

Open access



[github.com/syamanulm/geebook](https://github.com/syamanulm/geebook)

# Pemrosesan Citra Penginderaan Jauh Berbasis Python dan Google Earth Engine, Volume 1: Fundamentals

Disusun oleh: Syam'ani



Pusat Pengembangan Infrastruktur Informasi Geospasial  
Laboratorium Informasi Geospasial Fakultas Kehutanan  
Universitas Lambung Mangkurat



Masyarakat Ahli Penginderaan Jauh Indonesia  
Komisariat Wilayah Kalimantan Selatan



Pemrosesan Citra Penginderaan Jauh

# **Berbasis Python dan Google Earth Engine,**

**Volume 1: Fundamentals**

Syam'ani





# Pemrosesan Citra Penginderaan Jauh Berbasis Python dan Google Earth Engine, Volume 1: Fundamentals

Penulis:  
Syam'ani

Desain Cover:  
Syam'ani

Tata Letak:  
Hayatun Nisa

Editor:  
Dr. H. Abdi Fithria, S.Hut., M.P.

## **PENERBIT:**

ULM Press, 2025

d/a Publication Management Center ULM  
Lantai 1 Gedung Perpustakaan Pusat ULM  
Jl. Hasan Basri, Kayutangi, Banjarmasin 70123  
Telp./Fax. 0511 - 3305195  
ANGGOTA APPTI (004.035.1.03.2018)

Hak cipta dilindungi oleh Undang Undang  
Dilarang memperbanyak sebagian atau seluruh isi buku tanpa izin tertulis dari Penerbit, kecuali  
untuk kutipan singkat demi penelitian ilmiah dan resensi  
I - III + 376 hal, 18,2 × 25,7 cm  
Cetakan Pertama. Agustus 2025  
ISBN : 978-634-7195-31-9



---

*Buku ini dilarang untuk diperjualbelikan dalam bentuk apa pun. Setiap orang diizinkan untuk menggunakan atau membagikan buku ini secara gratis, dengan tidak merubah isinya atau mengklaim ulang hak ciptanya.*

---

**Petunjuk sitasi (APA Style):**

Syam'ani. (2025). *Pemrosesan Citra Penginderaan Jauh Berbasis Python dan Google Earth Engine, Volume 1: Fundamentals*. Banjarmasin: ULM Press.

Foto cover: ICEYE SAR Satellite (<https://www.iceye.com/press/media-assets>)



## PRAKATA

Di era perubahan iklim dan tekanan antropogenik yang semakin signifikan, penginderaan jauh menjadi salah satu instrumen paling krusial untuk memantau dinamika permukaan bumi. Dengan ketersediaan data satelit selama beberapa dekade terakhir, kita mampu menangkap perubahan bentang lahan bumi secara lebih mendetail. Namun, ragam sensor dan kompleksitas volume data menuntut pendekatan pemrosesan yang terstruktur dan efisien. Buku ini hadir sebagai panduan awal bagi peneliti, praktisi, dan mahasiswa yang ingin menguasai dasar-dasar pemrosesan citra penginderaan jauh. Lewat penerapan Python dan Google Earth Engine (GEE), pembaca akan mendapatkan fondasi teoritis sekaligus praktik langsung untuk menangani data observasi bumi dalam skala global. Volume pertama ini memfokuskan pada landasan konsep serta alur kerja fundamental yang dibutuhkan sebelum melangkah pada aplikasi terapan lanjutan.

Dalam beberapa tahun terakhir, perkembangan teknologi penginderaan jauh begitu pesat. Sebagai contoh, konstelasi satelit seperti Sentinel-2 memberikan data multispektral 10 meter secara gratis dan *near real-time* setiap 5 hari. Kemajuan di bidang komputasi awan—terutama GEE—memungkinkan pemrosesan petabyte data geospasial tanpa perlu infrastruktur lokal yang mahal. Di sisi perangkat lunak, ekosistem Python terus berkembang melalui pustaka seperti GDAL, Rasterio, Pandas/GeoPandas, Scikit-Learn, PyTorch, TensorFlow, dan sebagainya, yang membuka peluang integrasi metode *machine learning* dan *deep learning* ke dalam analisis citra.

Mengapa Python dan GEE? Karena kombinasi keduanya merevolusi cara kita mengelola dan menganalisis data penginderaan jauh dalam skala luas. Python, sebagai bahasa pemrograman *open-source*, memudahkan peneliti untuk berkolaborasi, mereproduksi hasil, dan membangun modul kustom sesuai kebutuhan riset. Sementara GEE menyediakan infrastruktur komputasi awan, akses langsung ke arsip citra satelit historis, serta API yang kaya fitur untuk geoanalisis.

Struktur buku ini dirancang untuk berkembang secara berjenjang. Bab pertama mengulas konsep citra digital, teknologi penginderaan jauh, dan persiapan teknis perangkat lunak. Bab berikutnya membahas tentang dasar-dasar Bahasa Python berikut pustaka-pustaka terkait. Selanjutnya, fokus beralih pada dasar-dasar dan aplikasi GEE. Pembaca jangan bingung atau terkejut, jika di dalam buku ini terdapat berbagai teknik yang berbeda untuk menyelesaikan satu masalah. Sebab buku ini secara implisit juga berusaha untuk memahamkan pembaca, bahwa di dalam pemrograman, terdapat banyak cara untuk menyelesaikan satu masalah yang sama. Akhir kata, dengan segala keterbatasan yang ada, semoga buku ini menjadi pintu gerbang Anda untuk menjelajahi dunia penginderaan jauh berbasis komputasi awan.

Banjarbaru, Agustus 2025  
Penulis,

Syam'ani

x

## SINOPSIS

Buku Pemrosesan Citra Penginderaan Jauh Berbasis Python dan Google Earth Engine, Volume 1: Fundamentals terdiri atas 3 bab. Bab pertama merupakan pengantar, yang menyajikan landasan teoritis citra digital penginderaan jauh, pengantar teknologi penginderaan jauh satelit, petunjuk penggunaan buku, instalasi dan pengantar perangkat lunak yang diperlukan, dan pengantar pemrograman berbasis komputasi awan. Bab kedua berisi tentang dasar-dasar Bahasa Pemrograman Python, dan pustaka-pustaka terkait dalam pemrosesan citra digital. Seperti NumPy, SciPy, Pandas, GeoPandas, Matplotlib, GDAL, dan Rasterio, berikut pembahasan tentang asisten-asisten AI di dalam ekosistem Python. Bab ketiga membahas tentang pemanfaatan Bahasa Python dan teknologi komputasi awan Google Earth Engine di dalam pemrosesan citra digital penginderaan jauh.

Jika Bab II membahas sintaks dasar Bahasa Python dan lebih diperuntukkan bagi para pembaca yang belum mengenal bahasa pemrograman ini, Bab III secara khusus memfokuskan pada operasi-operasi dasar pemrosesan citra digital penginderaan jauh berbasis teknologi Google Earth Engine. Seperti akses citra ke katalog Google Earth Engine, transformasi citra, pemfilteran citra, analisis regresi, klasifikasi digital, analisis multitemporal, hingga visualisasi informasi geospasial. Meskipun buku ini tidak membahas setiap metode lanjutan dalam pengolahan citra, fokus pada analisis dasar memungkinkan pembaca menguasai fondasi teknis yang diperlukan sebelum melangkah ke topik yang lebih kompleks.

Target utama kehadiran buku ini adalah efisiensi pekerjaan di dunia penginderaan jauh. Sebab di dalam pemrosesan citra penginderaan jauh, tantangan utama terletak pada besarnya kapasitas data yang diproses, beratnya spesifikasi perangkat keras yang diperlukan untuk memproses, dan mungkin mahalnya harga perangkat lunak yang digunakan untuk pemrosesan. Dengan komputasi awan, ketiga beban ini dapat diminimalisir, atau bahkan dieliminir sama sekali.

Bagi Anda yang berlatar belakang ahli atau praktisi penginderaan jauh, yang sudah malang melintang di dunia pemrosesan citra digital berbasis komputasi lokal, buku ini mungkin tidak akan mengajarkan konsep-konsep baru bagi Anda. Akan tetapi, buku ini mengajarkan cara baru terhadap apa yang sudah biasa Anda kerjakan selama ini. Selain akan membuat pekerjaan lama Anda menjadi lebih efisien, melalui cara-cara yang baru ini, diharapkan akan membuka peluang lahirnya ide-ide atau konsep-konsep baru di dalam pemrosesan citra penginderaan jauh. Mengingat dengan komputasi awan, akan membuat pekerjaan yang selama ini sulit, atau bahkan sama sekali tidak dapat dikerjakan, menjadi sangat mungkin atau sangat mudah untuk dikerjakan.

Secara singkat, buku referensi ini lebih menitikberatkan pada instruksi teknis dan penerapan praktis. Jika dipersentasekan secara kasar, mungkin buku ini memuat sekitar 30% teori dan sekitar 70% praktik. Sehingga pengetahuan tentang teori-teori penginderaan jauh dari sumber-sumber literatur lain tentu akan sangat membantu di dalam memahami buku ini. Dan mengingat buku ini lebih banyak berisi tentang penjelasan-penjelasan teknikal, maka metode paling efektif untuk mempelajari buku ini adalah dengan mengikuti latihan-latihan yang ada di dalamnya secara langsung, langkah demi langkah secara berurutan. Selain itu, mengingat Python, Google Earth Engine, Google Colab, dan perangkat lunak terkait akan terus berkembang setiap saat. Sebagai konsekuensinya, akan sangat mungkin bahwa pada saat Anda membaca beberapa bagian buku ini, langkah-langkah teknis yang disajikan sudah ketinggalan atau tidak berlaku lagi. Oleh karena itu, Anda direkomendasikan untuk menelusuri sendiri referensi-referensi terbaru untuk bagian-bagian yang sudah ketinggalan tersebut.



## DAFTAR ISI

<b>Prakata .....</b>	<b>ix</b>
<b>Sinopsis .....</b>	<b>xi</b>
<b>Daftar Isi.....</b>	<b>xiii</b>
<b>Daftar Tabel .....</b>	<b>xvii</b>
<b>Daftar Gambar .....</b>	<b>xix</b>
<b>Bab I Pengantar .....</b>	<b>1</b>
A. Struktur Citra Digital Penginderaan Jauh.....	3
B. Teknologi Penginderaan Jauh Satelit .....	9
Landsat Series.....	9
Sentinel Family .....	12
MODIS .....	18
GEDI .....	19
C. Petunjuk Menggunakan Buku Ini .....	20
Untuk siapa buku ini? .....	20
Apa yang diperlukan untuk menjalankan buku ini?.....	20
Bagaimana cara menggunakan buku ini? .....	20
D. Menggunakan Anaconda .....	21
Instalasi Anaconda.....	21
Anaconda Environment .....	25
Instalasi Paket Python.....	26
Menggunakan JupyterLab .....	30
Menggunakan Visual Studio Code.....	36
Menggunakan Spyder.....	40
E. Google Collaboratory.....	42
Login ke Google Colab .....	42
Membuat Notebook Baru.....	44
<b>Bab II Fundamental Python .....</b>	<b>53</b>
A. Mengenal Bahasa Python .....	55
B. Sintaks Dasar Bahasa Python .....	57
Variabel.....	57
Penggabungan String .....	58
Penamaan Variabel .....	59
Nilai Boolean.....	60
Operator Python.....	60
Prioritas Operator .....	62
Komentar .....	62
Kata-kata Kunci Python dan Sensitivitas Huruf.....	63
Indentasi.....	65
Jeda Baris Kode Python.....	65

Debugging Tips .....	66
Perulangan Instruksi dan Keputusan .....	66
Pernyataan Kondisional <code>if else</code> .....	67
Pernyataan Kondisional <code>if else</code> dalam Satu Baris .....	68
Pernyataan Perulangan <code>for</code> .....	68
Pernyataan Perulangan <code>while</code> .....	70
Pernyataan Lompatan <code>break</code> dan <code>continue</code> .....	71
Struktur Data .....	72
List .....	72
Tuple .....	78
Set .....	78
Dictionary .....	81
Fungsi .....	84
Fungsi dengan Jumlah Argumen Bebas .....	85
Fungsi Lambda .....	87
Pemrograman Berorientasi Objek .....	87
Kelas dan Objek .....	87
Lingkup dan Siklus Hidup Variabel .....	88
Kata kunci <code>del</code> .....	91
Destruktor <code>__del__</code> .....	92
Variabel Privat dan Fungsi Privat .....	92
Kata kunci <code>pass</code> .....	93
Metode Objek, Metode Kelas, dan Metode Statik .....	94
Metode Ajaib .....	94
Pewarisan .....	95
Modul .....	97
Import Paket Python .....	99
Penanganan Kesalahan dan Pengecualian .....	100
Kata Kunci <code>with</code> .....	101
<b>C. NumPy .....</b>	<b>102</b>
Konstruksi dan Akses NumPy Array .....	102
Transformasi Bentuk NumPy Array .....	106
Penggabungan dan Pemisahan NumPy Array .....	107
Mencari Elemen Unik .....	109
Kalkulus Pada NumPy Array .....	109
Kalkulus Vektor Pada NumPy Array .....	109
Konstanta NumPy .....	111
<b>D. SciPy .....</b>	<b>112</b>
<b>E. Pandas .....</b>	<b>114</b>
Dasar-dasar Pandas DataFrame .....	115
Mengakses Data Tabular Menggunakan Pandas .....	116
Mengakses Data di dalam DataFrame .....	117
<code>Toc</code> dan <code>iToc</code> .....	118
Mengedit Data di dalam DataFrame .....	120
Merubah Data .....	120
Menambahkan Baris dan Kolom Data .....	120
Kalkulasi di dalam DataFrame .....	121
Menghapus Data .....	122
Merubah Label Kolom .....	122
Menyimpan DataFrame ke dalam File .....	122
Visualisasi Pandas DataFrame .....	123
Konversi DataFrame Menjadi Data Geospasial .....	124

<b>F. GeoPandas.....</b>	<b>125</b>
Mengakses Data Geospasial .....	126
Seleksi Data Geospasial .....	126
Visualisasi Data Geospasial.....	128
Visualisasi Interaktif .....	130
Analisis Geospasial Dasar .....	130
<b>G. Matplotlib.....</b>	<b>133</b>
<b>H. GDAL.....</b>	<b>137</b>
Membaca dan Menulis Data Raster Menggunakan GDAL.....	137
<b>I. Rasterio .....</b>	<b>139</b>
<b>J. Gemini, AI-Powered Tools, Jupyter AI, dan Copilot .....</b>	<b>141</b>
Google Gemini .....	141
Fix error dan Explain error .....	142
Syntax Error vs Logical Error .....	144
Jupyter AI.....	145
GitHub Copilot .....	152
<b>Bab III Google Earth Engine .....</b>	<b>155</b>
<b>A. Mengenal Google Earth Engine .....</b>	<b>157</b>
Registrasi ke GEE.....	159
Katalog Data Earth Engine .....	163
Mengakses Citra Sentinel-2 MSI .....	165
Tips Jika Lupa Project ID Earth Engine Default Project.....	168
Memotong Citra Menggunakan Shapefile.....	172
Mengekspor Data ke Google Drive .....	174
Menggunakan GEE Assets .....	178
GEE Menggunakan JupyterLab/Visual Studio Code.....	181
Mengekspor Data ke Local Drive .....	187
Akses dan Unduh Citra Sentinel-2 Bebas Awan .....	187
Akses dan Unduh Citra Landsat-9 Bebas Awan .....	193
<b>B. Transformasi Citra.....</b>	<b>198</b>
Transformasi Spektral dan Indeks-indeks Multispektral .....	198
Thresholding dan Masking .....	202
Mengambil User ROI .....	207
Transformasi Spasial .....	208
Transformasi Temporal .....	213
<b>C. Pemfilteran Citra.....</b>	<b>222</b>
Filter Tekstur .....	222
Filter Morfologi.....	225
Spectral Distance .....	225
Spectral Dilation, Spectral Erosion, dan Spectral Gradient .....	228
Filter Konvolusi .....	232
<b>D. Analisis Regresi .....</b>	<b>235</b>
Regresi antara Data Lapangan dan Citra.....	235
Regresi Linier Menggunakan Scikit-Learn .....	241
Uji Akurasi Menggunakan MAPE dan RMSE.....	241
Visualisasi Grafik Regresi.....	242
Implementasi Model Regresi .....	243
Regresi antar Citra .....	244
<b>E. Klasifikasi Multispektral .....</b>	<b>255</b>

Klasifikasi Multispektral Terselia.....	256
Uji Akurasi Menggunakan Confusion Matrix .....	260
Klasifikasi Multispektral Tak Terselia .....	262
<b>F. Analisis Citra Berbasis Objek .....</b>	<b>264</b>
Uji Akurasi Menggunakan Confusion Matrix .....	270
Konversi Hasil Segmentasi ke Vektor .....	274
<b>G. Analisis Multitemporal .....</b>	<b>278</b>
Ekstraksi Informasi Geospasial Sawah.....	278
Uji Akurasi Menggunakan Intersection Over Union (IoU) .....	283
Multiclass IoU.....	287
Memantau Dinamika Hutan Mangrove.....	289
Membuat Timelapse Animation Hutan Mangrove.....	292
Pixelwise Change Detections.....	295
Image Differencing .....	296
Image Rationing .....	301
Image Regression .....	303
Change Vector Analysis .....	307
<b>H. Analisis dan Prediksi Deret Waktu .....</b>	<b>315</b>
Kecenderungan Linier.....	322
Kecenderungan Non-Linier.....	324
Kecenderungan Harmonik.....	326
Prediksi Deret Waktu.....	328
<b>I. Analisis Sub-Pixel .....</b>	<b>333</b>
<b>J. Visualisasi Data .....</b>	<b>350</b>
Timelapse Animation .....	350
Membuat Layout .....	356
Web Map Interaktif .....	363
Grafik 3 Dimensi dan Animasi Grafik .....	366
Web Map Interaktif Citra Satelit Multitemporal .....	368
Permukaan 3 Dimensi dan Animasi Permukaan.....	373
<b>Referensi .....</b>	<b>377</b>
<b>Indeks .....</b>	<b>383</b>

## DAFTAR TABEL

Tabel 1.1.	Koversi bilangan desimal, biner, oktal, dan heksadesimal .....	3
Tabel 1.2.	Perbandingan Landsat 7 EMT+ dan Landsat 8 OLI/TIRS .....	11
Tabel 1.3.	Perbandingan band Sentinel-2 dan Landsat 4, 5, 7, 8, 9.....	14
Tabel 1.4.	Sentinel-3 OLCI bands.....	16
Tabel 1.5.	Informasi gas-gas atmosfir bumi yang disediakan oleh Sentinel-5P .....	16
Tabel 1.6.	Spesifikasi Citra MODIS.....	18
Tabel 2.1.	Bahasa pemrograman terpopuler menurut TIOBE Index untuk Agustus 2025	55
Tabel 2.2.	Operator aritmatika .....	60
Tabel 2.3.	Operator penugasan .....	61
Tabel 2.4.	Operator perbandingan .....	61
Tabel 2.5.	Operator logika .....	61
Tabel 2.6.	Operator identitas .....	61
Tabel 2.7.	Operator keanggotaan.....	61
Tabel 2.8.	Operator bitwise .....	62
Tabel 2.9.	Prioritas operator Python .....	62
Tabel 2.10.	Kata-kata kunci Bahasa Python .....	64
Tabel 2.11.	Fungsi-fungsi yang dapat diterapkan pada list .....	77
Tabel 2.12.	Konstanta NumPy .....	111
Tabel 2.13.	Sub-paket SciPy.....	112
Tabel 3.1.	Tipe-tipe data GEE .....	158
Tabel 3.2.	USGS burn severity level classification .....	245
Tabel 3.3.	Interpretasi umum nilai Kappa .....	274



## DAFTAR GAMBAR

Gambar 1.1.	Ilustrasi teknologi digital .....	3
Gambar 1.2.	Ilustrasi nilai pixel.....	4
Gambar 1.3.	Ilustrasi pembentukan citra berwarna.....	5
Gambar 1.4.	Citra satelit multispektral.....	5
Gambar 1.5.	Ilustrasi reflectance pada citra optik.....	6
Gambar 1.6.	TOA vs TOC.....	7
Gambar 1.7.	Resolusi citra digital penginderaan jauh .....	8
Gambar 1.8.	Satelit Landsat 1.....	9
Gambar 1.9.	Perbandingan band seluruh seri Landsat.....	10
Gambar 1.10.	Satelit Landsat 9.....	10
Gambar 1.11.	Landsat Next Spectral Bands.....	11
Gambar 1.12.	Sentinel Family.....	12
Gambar 1.13.	Mode akuisisi Sentinel-1 SAR .....	13
Gambar 1.14.	Perbandingan band Landsat 7 dan Landsat 8 dengan Sentinel-2 .....	13
Gambar 1.15.	Sentinel-2 MSI tile untuk wilayah Kalimantan Selatan.....	14
Gambar 1.16.	Satelit Sentinel-3.....	15
Gambar 1.17.	Data gas-gas atmosfir bumi yang disediakan oleh Sentinel-5P.....	17
Gambar 1.18.	Citra gas CO di atas Pulau Kalimantan pada tanggal 5 Oktober 2023 .....	17
Gambar 1.19.	GEDI mengekstrak ketinggian kanopi di Hutan Amazon .....	19
Gambar 2.1.	Ilustrasi citra 3 band.....	103
Gambar 2.2.	Format-format data tabular yang dapat diakses Pandas .....	114
Gambar 2.3.	Struktur DataFrame Pandas .....	114
Gambar 2.4.	Laman Matplotlib Colormaps .....	124
Gambar 3.1.	Opsi bahasa dalam penggunaan GEE.....	158
Gambar 3.2.	Jenis klasifikasi digital citra penginderaan jauh menurut perspektif pixel....	255
Gambar 3.3.	Terminologi objek pada citra penginderaan jauh .....	264
Gambar 3.4.	Konsep uji akurasi menggunakan confusion matrix.....	270
Gambar 3.5.	Uji akurasi data atribut informasi geospasial.....	283
Gambar 3.6.	Konsep dasar IoU, Precision, dan Recall .....	283
Gambar 3.7.	Implementasi teknis uji akurasi menggunakan IoU .....	284
Gambar 3.8.	Ilustrasi pure pixel dan mixed pixel.....	333
Gambar 3.9.	Fraksi-fraksi endmember dalam satu mixed pixel.....	334





# Bab I

# Pengantar



## A. Struktur Citra Digital Penginderaan Jauh

Teknologi digital merupakan sebuah teknologi yang dirancang sedemikian rupa untuk menterjemahkan setiap data dan informasi menjadi *binary number* (bilangan biner atau bilangan berbasis dua). Sehingga semua data dan informasi yang disimpan, diolah, dan disajikan, pada hakikatnya berwujud bilangan biner. Bilangan biner sendiri adalah suatu sistem bilangan yang hanya mengenal dua simbol, yaitu 0 (nol) dan 1 (satu). Mungkin muncul pertanyaan, mengapa teknologi digital menggunakan bilangan biner sebagai format manajemen data dan informasi? Hal ini karena teknologi digital pada dasarnya mengacu kepada sifat-sifat listrik. Dimana listrik hanya mengenal dua keadaan, yaitu padam/mati atau menyala/hidup. Pada umumnya, keadaan listrik mati disimbolkan dengan 0, dan keadaan listrik hidup disimbolkan dengan 1. Sebagaimana diilustrasikan pada Gambar 1.1. Sehingga semua komunikasi data dan informasi di dalam teknologi digital secara keseluruhan dinotasikan dengan angka 0 dan 1. Hal ini mirip dengan Kode Morse yang hanya mengenal dua simbol, yaitu titik dan strip, untuk merepresentasikan karakter-karakter alfanumerik.



Gambar 1.1. Ilustrasi teknologi digital

Bilangan biner dapat dikonversi menjadi bilangan desimal, yaitu bilangan berbasis 10 yang kita kenal di dalam pertukaran data dan informasi sehari-hari. Bilangan biner juga dapat dikonversi menjadi bilangan berbasis 8 atau bilangan oktal, atau bilangan berbasis 16 atau yang dikenal sebagai bilangan heksadesimal. Sebagaimana terlihat pada Tabel 1.1.

Tabel 1.1. Koversi bilangan desimal, biner, oktal, dan heksadesimal

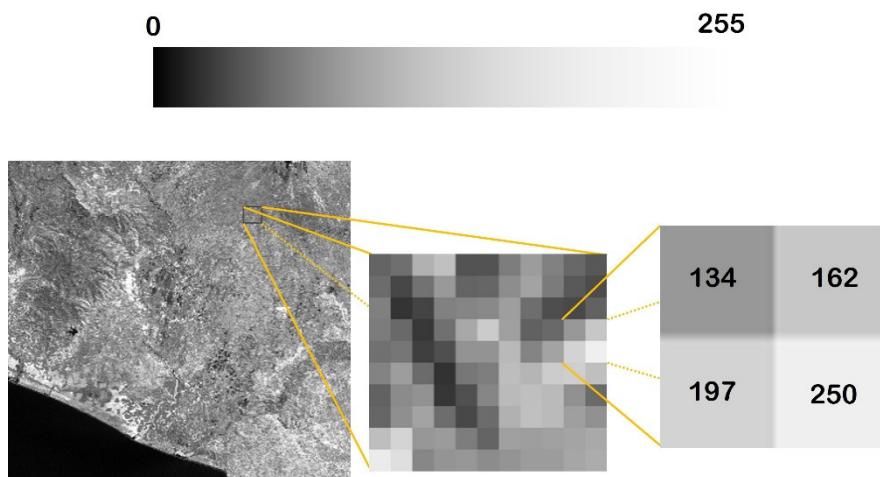
Desimal	Biner	Oktal	Heksadesimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

## Bab I Pengantar

Citra digital penginderaan jauh 8-bit, seperti Landsat 7 ETM+, memiliki presisi informasi radiometrik 8-bit (*binary digit*). Sehingga setiap pixel Citra Landsat 7 memiliki 8 digit bilangan biner. Yaitu dimulai dari 00000000 (semua bit “padam”) hingga 11111111 (semua bit “menyal”). 0000000 jika dikonversi menjadi desimal akan ekivalen dengan 0. Sementara 11111111 jika dikonversi menjadi desimal akan ekivalen dengan 255. Perhitungannya adalah sebagai berikut:

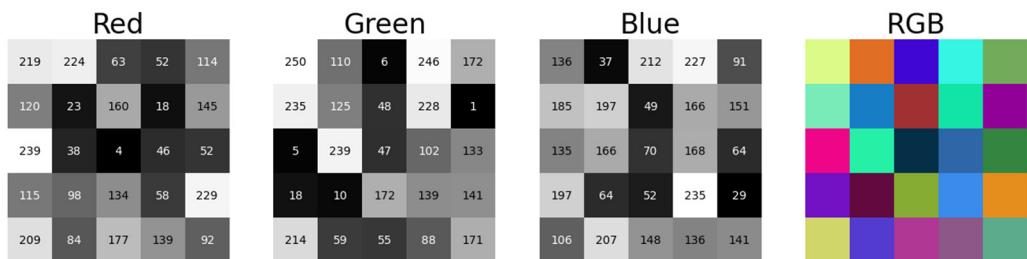
$$\begin{array}{rcl} 1 \cdot 2^0 & = & 1 \\ 1 \cdot 2^1 & = & 2 \\ 1 \cdot 2^2 & = & 4 \\ 1 \cdot 2^3 & = & 8 \\ 1 \cdot 2^4 & = & 16 \\ 1 \cdot 2^5 & = & 32 \\ 1 \cdot 2^6 & = & 64 \\ 1 \cdot 2^7 & = & 128 \\ & & \hline & & 255 \end{array}$$

Semua bit “padam” biasanya akan memberikan warna gelap atau hitam pada citra, sementara semua bit “menyal” akan memberikan warna putih pada citra. Sebagaimana ilustrasi pada Gambar 1.2. Sedangkan warna di antara hitam dan putih adalah abu-abu. Warna abu-abu tentu saja di antara 0 dan 255, atau dengan kata lain di antara 00000000 dan 11111111. Nilai pixel 162 misalnya akan identik dengan 10100010 biner.



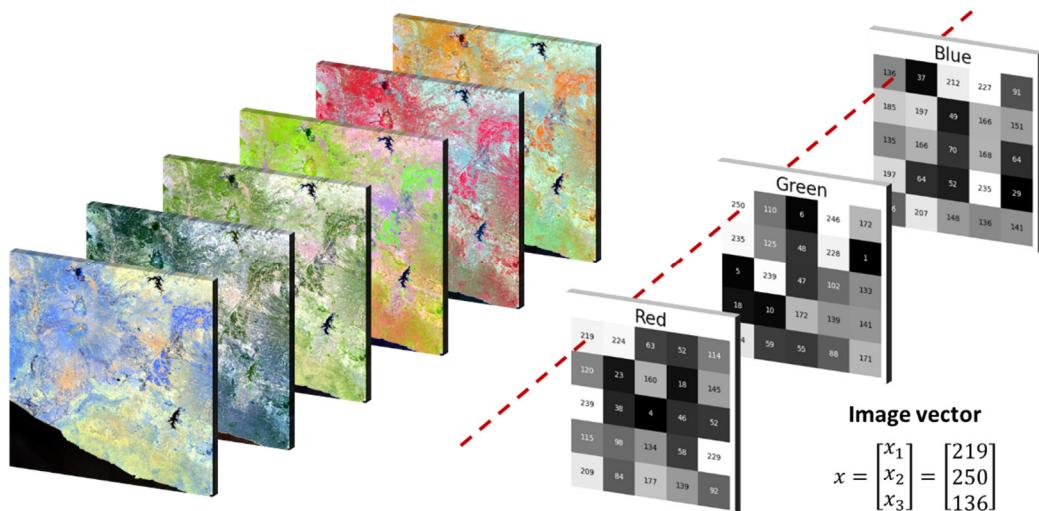
Gambar 1.2. Ilustrasi nilai pixel

Demikian halnya dengan citra berwarna *Red-Green-Blue* (RGB), yang pada dasarnya terkomposisi dari 3 buah citra, dimana masing-masing citra 8-bit. Perhatikan Gambar 1.3, untuk pixel dengan posisi paling kiri atas. Pada band Red nilai pixelnya 219, pada band Green nilai pixelnya 250, dan pada band Blue nilai pixelnya 136. Pada sistem formasi warna digital RGB 24-bit (3 band x 8-bit), masing-masing band akan memiliki kode biner 8-bit. Pada nilai-nilai pixel pada Gambar 1.3, 219 ekivalen dengan 11011011 biner atau DB heksadesimal, 250 ekivalen dengan 11111010 biner atau FA heksadesimal, dan 136 ekivalen dengan 10001000 biner atau 88 heksadesimal (kalkulasi dilakukan di laman <https://www.rapidtables.com/convert/number/decimal-to-binary.html>). Sehingga dalam notasi heksadesimal 24 bit, vektor citra tersebut dapat ditulis DBFA88. Notasi heksadesimal 24-bit seperti ini lumrah di dunia *programming* atau *web design*.



Gambar 1.3. Ilustrasi pembentukan citra berwarna

Sebuah citra dapat berupa satu saluran (*band/channel*) tunggal atau *single band*, tetapi dapat juga terdiri atas banyak saluran, yang disebut citra multispektral atau citra multisaluran. Pada citra multisaluran, setiap band direkam pada spektrum gelombang elektromagnetik yang berbeda. Sebagaimana kamera digital smartphone kita yang merekam pada spektrum gelombang elektromagnetik merah, hijau, dan biru, untuk menghasilkan warna foto yang alami sebagaimana warna objek aslinya. Sehingga foto JPEG yang dihasilkan oleh kamera smartphone kita sebenarnya merupakan sebuah citra 3 band. Akan tetapi, untuk citra satelit penginderaan jauh seperti Landsat 7 atau Sentinel-2, citranya tidak hanya terdiri atas 3 band, melainkan multiband atau multispektral. Hal ini bertujuan untuk menangkap kenampakan berbagai fitur atau objek, yang bahkan terkadang mata manusia sendiri tidak dapat melihatnya.



Gambar 1.4. Citra satelit multispektral

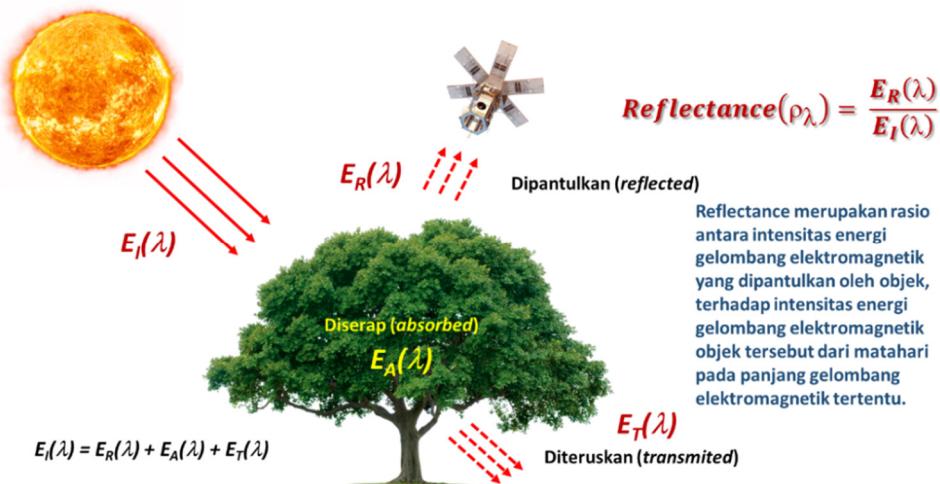
Gambar 1.4 menunjukkan bagaimana sebuah citra digital multispektral direpresentasikan. Citra digital multispektral pada dasarnya adalah sebuah *array* data numerik. Setiap pixelnya memiliki nilai, dan nilai pixel antar saluran pada lokasi yang sama sering dinotasikan dengan *image vector*. Pada Gambar 1.3 sebelumnya, untuk pixel di sudut paling kiri atas jika dinotasikan dalam bentuk vektor citra bentuknya akan seperti ini  $\begin{bmatrix} 219 \\ 250 \\ 136 \end{bmatrix}$ . Sebagaimana ilustrasi pada Gambar 1.4. Konstruksi vektor citra seperti ini merupakan dasar di dalam melakukan pemrosesan citra digital. Hampir setiap algoritma pengolah citra digital, seperti klasifikasi multispektral, segmentasi citra, bahkan *machine learning*, dan *deep learning*, semuanya berbasis vektor citra.

## Bab I Pengantar

Dikarenakan sebuah citra digital penginderaan jauh pada dasarnya hanya susunan angka, maka semua yang dapat kita lakukan pada angka tentu saja dapat kita lakukan pada citra penginderaan jauh. Tentu saja, yang dapat kita lakukan pada angka adalah semua kalkulasi matematika dan statistika. Demikian halnya dengan citra digital, kita dapat membagi antar band (penisbahan atau rasio antar band), misalnya band merah dibagi dengan band hijau, atau pun operasi-operasi matematika lainnya yang lebih kompleks. Tinggal yang harus kita fahami sekarang, angka atau nilai pixel pada citra digital penginderaan jauh tersebut menunjukkan nilai apa? Jawaban sederhananya, tergantung jenis citra penginderaan jauhnya.

Berdasarkan sumber energi gelombang elektromagnetik yang digunakan, sistem penginderaan jauh dapat dibagi menjadi dua bagian, yaitu sistem pasif dan sistem penginderaan jauh aktif. Sistem penginderaan jauh pasif berarti sumber energi gelombang elektromagnetik yang digunakan untuk pencitraan tidak berasal dari sensor, melainkan dari alam. Misalnya dari cahaya matahari atau energi panas di atas permukaan bumi. Sedangkan sistem penginderaan jauh aktif berarti sumber energi gelombang elektromagnetik yang digunakan untuk pencitraan berasal dari sensor itu sendiri. Dengan kata lain, sensornya membawa dan menembakkan gelombang elektromagnetik sendiri ke atas permukaan bumi.

Sistem penginderaan jauh yang tergolong pada sistem penginderaan jauh pasif adalah sistem penginderaan jauh optik dan sistem penginderaan jauh termal. Sementara sistem penginderaan jauh yang termasuk pada sistem penginderaan jauh aktif adalah sistem penginderaan jauh *Radio Detection and Ranging* (Radar) dan *Light Detection and Ranging* (Lidar). Buku ini akan lebih berfokus pada pembahasan sistem penginderaan jauh pasif, terutama penginderaan jauh optik. Sementara sistem penginderaan jauh aktif nanti mungkin hanya sedikit yang akan diulas dan diberikan contohnya di dalam buku ini.

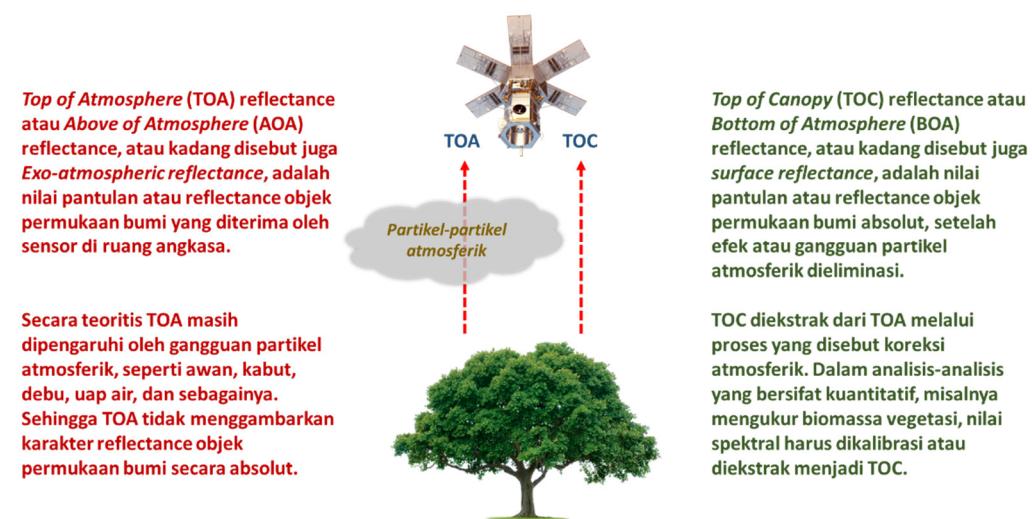


Gambar 1.5. Ilustrasi reflectance pada citra optik  
(Konsep diterjemahkan dari Müller and Pfeifroth, 2022)

Pada sistem penginderaan jauh termal, nilai pixel citranya merupakan hasil konversi dari nilai *emittance*. Emittance merupakan energi termal yang diemisi oleh objek-objek di atas permukaan bumi. Untuk selanjutnya, nilai emittance ini dapat dikonversi menjadi temperatur emisi atau temperatur permukaan (*Land Surface Temperature* (LST)). Citra-citra seperti NOAA dan MODIS menggunakan saluran-saluran termal untuk mendeteksi hotspot.

Pada sistem penginderaan jauh optik, nilai pixelnya merupakan representasi dari nilai *radiance*. Yaitu besaran energi pantulan gelombang elektromagnetik yang dipancarkan oleh objek-objek di atas permukaan bumi. Nilai radiance ini kemudian dapat dikonversi menjadi reflectance, yaitu rasio atau nisbah antara energi yang dipantulkan objek ke sensor penginderaan jauh terhadap energi yang diterima oleh objek tersebut dari matahari, pada spektrum gelombang elektromagnetik tertentu. Sebagaimana diilustrasikan pada Gambar 1.5. Nilai reflectance ini pada umumnya dijadikan dasar di dalam pemodelan kuantitatif citra digital penginderaan jauh, seperti model regresi antara indeks vegetasi dan biomassa vegetasi.

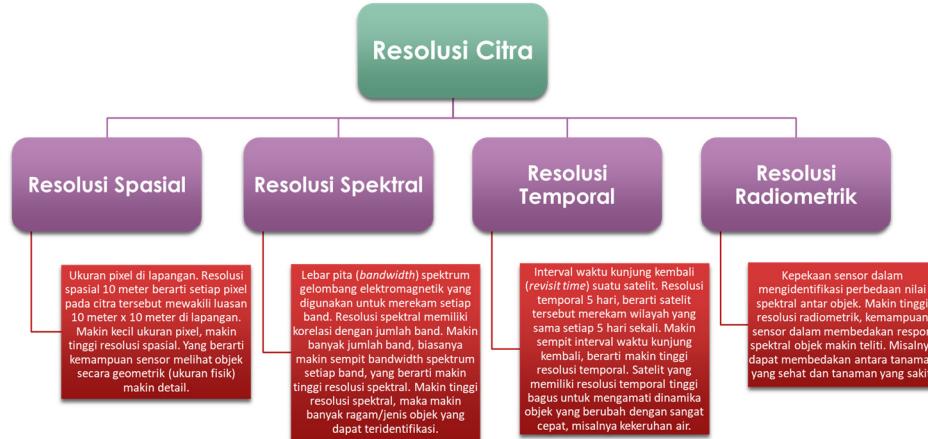
Reflectance sendiri ada dua jenis, yaitu *Top of Atmosphere* (TOA) reflectance dan *Top of Canopy* (TOC) reflectance. TOA atau sering juga disebut sebagai *exo-atmospheric reflectance* merupakan nilai reflectance yang terlihat dari ruang angkasa oleh sensor penginderaan jauh. Dengan kata lain, nilai TOA masih belum terkoreksi sehingga masih terpengaruh atau bias terhadap partikel-partikel atmosferik, seperti debu, kabut, asap, dan sebagainya. Sementara TOC atau dikenal juga sebagai *surface reflectance* merupakan TOA yang sudah terkoreksi. Sehingga nilai TOC seolah-olah adalah nilai reflectance absolut objek seperti ketika kita melihat objek tersebut secara langsung berhadapan di atas permukaan bumi. Perhatikan ilustrasi pada Gambar 1.6.



Gambar 1.6. TOA vs TOC

Selain memiliki nilai, setiap pixel pada citra penginderaan jauh juga memiliki ukuran. Karena pixel berbentuk kotak, maka setiap pixel citra penginderaan jauh akan mewakili luasan wilayah tertentu di atas permukaan bumi. Luasan wilayah yang diwakili oleh setiap pixel citra penginderaan jauh ini dikenal sebagai resolusi spasial.

**RESOLUSI SPASIAL** menggambarkan ukuran objek terkecil yang masih dapat dideteksi oleh sensor dan direpresentasikan oleh citra penginderaan jauh. Ukuran resolusi spasial biasanya dinyatakan dengan kilometer, meter, dan centimeter. Semakin kecil ukuran pixel, dikatakan bahwa resolusi spasial akan semakin tinggi. Sehingga Sentinel-2 yang memiliki resolusi spasial 10 meter, dikatakan bahwa resolusi spasialnya lebih tinggi dibandingkan dengan Landsat-8 yang resolusi spasialnya 30 meter. Selain resolusi spasial, terdapat juga resolusi spektral, resolusi temporal, dan resolusi radiometrik. Untuk lebih jelasnya dapat dilihat pada Gambar 1.7.



Gambar 1.7. Resolusi citra digital penginderaan jauh  
(Konsep diterjemahkan dari Schowengerdt, 2007)

**RESOLUSI SPEKTRAL** merupakan *bandwidth* atau lebar pita spektrum gelombang elektromagnetik untuk setiap band citra. Sebagai contoh, band 1 (*coastal aerosol*) Citra Sentinel-2 memiliki bandwidth 433-453 nm (*nanometer*). Sementara band 9 Citra MODIS (juga *coastal aerosol*) memiliki bandwidth 438-448 nm. Karena rentang spektrum atau lebar pita MODIS lebih sempit, yaitu dari 438 nm hingga 448 nm, dibandingkan dengan Sentinel-2 yang lebih lebar yaitu dari 433 nm hingga 453 nm, maka dikatakan resolusi spektral saluran *coastal aerosol* MODIS lebih tinggi dibandingkan dengan saluran *coastal aerosol* Sentinel-2.

**RESOLUSI TEMPORAL** merupakan interval waktu *revisit time* atau waktu kunjung kembali suatu wahana penginderaan jauh, yaitu satelit, di lokasi yang sama di atas permukaan bumi. Misalnya, di atas Kota Banjarbaru, Landsat 9 melakukan akuisisi setiap berapa hari sekali. Semakin pendek interval waktu akuisisi maka dikatakan resolusi temporalnya semakin tinggi. Sehingga Sentinel-2 yang memiliki interval waktu akuisisi setiap 5 hari sekali, dikatakan lebih tinggi resolusi temporalnya dibandingkan dengan Landsat 8 yang memiliki interval waktu akuisisi 16 hari. Citra dengan resolusi temporal yang tinggi sangat diperlukan untuk observasi fitur-fitur yang sangat dinamik atau dapat berubah dalam waktu yang sangat singkat, seperti kualitas air.

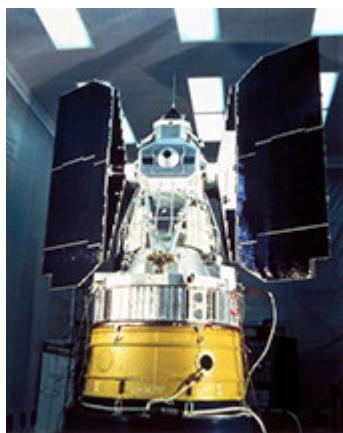
Terkait dengan resolusi temporal, tentu saja wahana penginderaan jauh seperti *Unmanned Aerial Vehicle* (UAV) atau *drone* tidak memiliki resolusi temporal. Sebab wahana ini dapat diterbangkan kapan pun sesuai keperluan kita. Khusus untuk satelit, terdapat pula istilah satelit geostasioner. Yaitu satelit yang kecepatan orbitalnya sama persis dengan kecepatan rotasi bumi. Sehingga satelit tersebut relatif diam terhadap lokasi yang sama di atas permukaan bumi. Dengan kata lain, satelit geostasioner secara terus-menerus merekam tempat yang sama di permukaan bumi. Contoh satelit geostasioner adalah satelit cuaca Himawari-8 (<https://himawari8.nict.go.jp>).

**RESOLUSI RADIOMETRIK** merupakan presisi atau kepekaan sensor dalam mengidentifikasi perbedaan nilai spektral antar objek di atas permukaan bumi. Resolusi radiometrik dinyatakan dalam satuan *bit (binary digit)*. Landsat 7 ETM+ memiliki resolusi radiometrik 8-bit, Landsat 8 OLI memiliki resolusi radiometrik 16-bit, sementara Sentinel-2 memiliki resolusi radiometrik 12-bit. Semakin tinggi bit-nya dikatakan resolusi radiometriknya semakin tinggi. Sensor yang memiliki resolusi radiometrik lebih tinggi, dapat lebih sensitif atau teliti dalam membedakan respons spektral yang sangat kecil dari objek-objek di atas permukaan bumi.

## B. Teknologi Penginderaan Jauh Satelit

### Landsat Series

Landsat merupakan program observasi bumi dari *United States Geological Survey (USGS)*. Sampai dengan saat ini, seri Landsat sudah mencapai generasi kesembilan, yaitu Landsat 9. Landsat mungkin merupakan citra satelit paling populer bagi para ahli dan praktisi penginderaan jauh. Khususnya sebelum kehadiran keluarga Sentinel. Citra-citra Landsat dari generasi 1 sampai dengan 9 dapat diunduh di laman resminya <https://earthexplorer.usgs.gov>. Dahulu seri Landsat merupakan citra satelit komersial, yang dijual seharga jutaan rupiah untuk setiap liputan (*scene*) perekamannya. Akan tetapi, sejak Oktober 2008 USGS menggratiskan Landsat 7 untuk publik, menyusul empat bulan kemudian seluruh seri Landsat digratiskan sepenuhnya untuk publik (<https://www.usgs.gov/landsat-missions/landsat-7>).



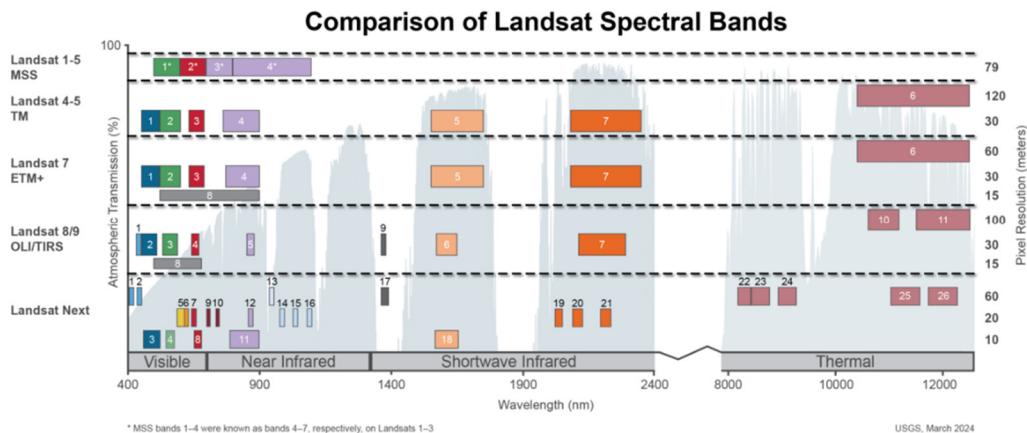
Gambar 1.8. Satelit Landsat 1  
(<https://www.usgs.gov/media/images/landsat-1-satellite>)

**LANDSAT 1** diluncurkan ke orbit pada 23 Juli 1972, pada saat itu Satelit Landsat 1 dikenal sebagai *Earth Resources Technology Satellite (ERTS)*. Landsat 1 merupakan satelit observasi bumi pertama yang diluncurkan dengan tujuan untuk mempelajari dan memantau daratan planet kita. Untuk melakukan pemantauan, Landsat 1 membawa dua instrumen yang disebut *Return Beam Vidicon (RBV)* dan *Multispectral Scanner (MSS)* (<https://landsat.gsfc.nasa.gov/satellites/landsat-1>). Penerus Landsat 1, yaitu Landsat 2 dan Landsat 3 masing-masing membawa sensor yang sama dengan Landsat 1, yaitu RBV dan MSS. **LANDSAT 2** diorbitkan pada tanggal 22 Januari 1975 (<https://landsat.gsfc.nasa.gov/satellites/landsat-2>) dan **LANDSAT 3** diorbitkan pada tanggal 5 Maret 1978 (<https://landsat.gsfc.nasa.gov/satellites/landsat-3>). Landsat 1 dan Landsat 2 memiliki resolusi spasial 80 meter. Khusus untuk sensor RBV, Landsat 3 memiliki resolusi spasial 40 meter (<https://www.usgs.gov/landsat-missions/landsat-3>).

**LANDSAT 4** diorbitkan pada tanggal 16 Juli 1982. Landsat 4 tidak membawa sensor RBV, melainkan menggantinya dengan *Thematic Mapper (TM)*, dan masih membawa sensor MSS (<https://landsat.gsfc.nasa.gov/satellites/landsat-4>). Untuk sensor optik TM memiliki resolusi spasial 30 meter, sementara sensor termalnya memiliki resolusi spasial 120 meter (<https://www.usgs.gov/landsat-missions/landsat-4>). **LANDSAT 5 TM** diluncurkan ke orbit pada

## Bab I Pengantar

tanggal 1 Maret 1984. Landsat 5 membawa sensor yang sama persis dengan Landsat 4 TM (<https://landsat.gsfc.nasa.gov/satellites/landsat-5>). Sepertinya Landsat 5 merupakan generasi Landsat yang paling panjang usianya, sebab beroperasi selama 29 tahun (sampai 5 Juni 2013).



Gambar 1.9. Perbandingan band seluruh seri Landsat  
(<https://www.usgs.gov/media/images/spectral-bandpasses-all-landsat-sensors>)

**LANDSAT 6** membawa sensor *Enhanced Thematic Mapper* (ETM) dan diluncurkan pada tanggal 5 Oktober 1993. Landsat 6 memiliki resolusi spasial 30 meter, dan terdapat tambahan 1 band pankromatik dengan resolusi spasial 15 meter. Akan tetapi, Satelit Landsat 6 gagal mencapai orbit dikarenakan pecahnya *hydrazine manifold*. Akibatnya, pesawat terjatuh alih-alih mengumpulkan energi yang cukup untuk mencapai orbit (<https://landsat.gsfc.nasa.gov/satellites/landsat-6>). Itu sebabnya kita tidak akan pernah mendengar atau melihat Citra Landsat 6. Penerus Landsat 6, **LANDSAT 7 ENHANCED THEMATIC MAPPER PLUS (ETM+)** diluncurkan ke orbit pada tanggal 15 April 1999. Landsat 7 memiliki resolusi spasial 30 meter, kecuali band termal yang memiliki resolusi spasial 60 meter, dan band pankromatik yang memiliki resolusi spasial 15 meter. Sayangnya, pada Juni 2003, Landsat 7 mengalami kegagalan *Scan Line Corrector* (SLC). Akibatnya, citra yang dihasilkan memiliki *stripping* (<https://www.usgs.gov/landsat-missions/landsat-7>).



Gambar 1.10. Satelit Landsat 9  
(<https://www.usgs.gov/landsat-missions/landsat-9>)

**LANDSAT 8** diluncurkan pada 11 Februari 2013, dan membawa sensor *Operational Land Imager* (OLI) dan *Thermal Infrared Sensor* (TIRS). Landsat 8 memiliki resolusi spasial 30 meter, kecuali band termal 100 meter, dan band pankromatik 15 meter. Landsat 8 juga menambahkan dua band baru, yaitu *Coastal Aerosol* dan *Cirrus* (<https://landsat.gsfc.nasa.gov/satellites/landsat-8>). **LANDSAT 9** diluncurkan pada 27 September 2021 (<https://landsat.gsfc.nasa.gov/satellites/landsat-9>). Mirip dengan Landsat 8, Landsat 9 juga membawa sensor OLI-2/TIRS-2. Landsat 8 dan Landsat 9 memiliki resolusi temporal 16 hari. Akan tetapi, terdapat *offset* atau selisih waktu 8 hari di antara keduanya. Sehingga Landsat 8 dan Landsat 9 akan merekam lokasi yang sama secara bergantian setiap 8 hari ([https://landsat.usgs.gov/landsat\\_acg](https://landsat.usgs.gov/landsat_acg)).

Tabel 1.2. Perbandingan Landsat 7 EMT+ dan Landsat 8 OLI/TIRS

ETM+ and OLI/TIRS Spectral Bands		
L7 ETM+ Bands	LDCM OLI/TIRS Band Requirements	
Band 1 30 m, Blue, 0.450–0.515 $\mu\text{m}$	30 m, Coastal/Aerosol, 0.433–0.453 $\mu\text{m}$ (*A)	Band 1
Band 2 30 m, Green, 0.525–0.605 $\mu\text{m}$	30 m, Blue, 0.450–0.515 $\mu\text{m}$	Band 2
Band 3 30 m, Red, 0.630–0.690 $\mu\text{m}$	30 m, Green, 0.525–0.600 $\mu\text{m}$	Band 3
Band 4 30 m, Near-IR, 0.775–0.900 $\mu\text{m}$	30 m, Red, 0.630–0.680 $\mu\text{m}$ (*B)	Band 4
Band 5 30 m, SWIR-1, 1.550–1.750 $\mu\text{m}$	30 m, Near-IR, 0.845–0.885 $\mu\text{m}$ (*B)	Band 5
Band 6 30 m, SWIR-2, 2.090–2.350 $\mu\text{m}$	30 m, SWIR-1, 1.560–1.660 $\mu\text{m}$ (*B)	Band 6
Band 7 30 m, SWIR-2, 2.090–2.350 $\mu\text{m}$	30 m, SWIR-2, 2.100–2.300 $\mu\text{m}$ (*B)	Band 7
Band 8 15 m, Pan, 0.520–0.900 $\mu\text{m}$	15 m, Pan 0.500–0.680 $\mu\text{m}$ (*B)	Band 8
	30 m, Cirrus, 1.360–1.390 $\mu\text{m}$ (*C)	Band 9
Band 9 60 m, LWIR, 10.00–12.50 $\mu\text{m}$	100 m, LWIR-1, 10.30–11.30 $\mu\text{m}$ (*D)	Band 10
	100 m, LWIR-2, 11.50–12.50 $\mu\text{m}$ (*D)	Band 11

#### \*Explanation of Differences

- A. Coastal Band added at request of ocean color investigators requiring higher resolution of coastal waters relative to MODIS and SeaWiFS.
- B. Bandwidth refinements made to avoid atmospheric absorption features (enabled by the higher signal-to-noise ratio inherent in push-broom architecture).
- C. Cirrus Band added to detect cirrus contamination in other channels.
- D. TIRS will acquire the data for these two thermal bands.

Sumber: <https://landsat.gsfc.nasa.gov>

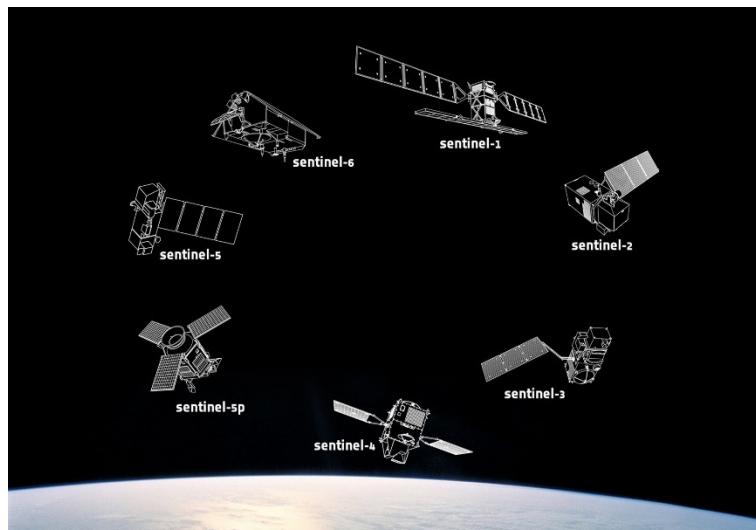
Citra Landsat dirilis ke publik dalam 3 level, yaitu Level 1 (L1) dalam format *Digital Number* (DN), Level 2 (L2) dalam format TOC, dan Level 3 (L3) yang merupakan produk-produk Landsat yang sudah siap pakai seperti *Burned Area* dan *Dynamic Surface Water Extent*. Setelah era Landsat 9, USGS merencanakan misi Landsat berikutnya yang rencananya akan diluncurkan pada tahun 2030, yang dikenal sebagai Landsat Next. **LANDSAT NEXT** akan memberikan kemampuan baru bagi pengguna Landsat generasi berikutnya. Peningkatan resolusi spasial dan temporal dari konstelasi Landsat Next “superspektral” 26-band akan membuka aplikasi baru untuk kualitas air, produksi tanaman dan tekanan tanaman, dinamika iklim dan salju, kesehatan tanah dan variabel lingkungan penting lainnya (<https://www.usgs.gov/landsat-missions/landsat-next>).



Gambar 1.11. *Landsat Next Spectral Bands*  
<https://www.usgs.gov/media/images/landsat-next-spectral-bands>

## Sentinel Family

Sentinel merupakan sebuah program observasi bumi yang diberi nama *Copernicus* dari *European Space Agency* (ESA). Keluarga Sentinel mencakup sejumlah besar satelit dengan berbagai jenis dan fungsinya. Satelit-satelit Sentinel yang sudah berada di orbit saat ini adalah Sentinel-1, Sentinel-2, Sentinel-3, Sentinel-5P, dan Sentinel-6. Harap diperhatikan bahwa kode penomoran pada keluarga Sentinel berbeda dengan seri Landsat. Pada seri Landsat, kode nomor (misalnya Landsat 8) menunjukkan generasi satelit. Sedangkan pada keluarga Sentinel, kode nomor (misalnya Sentinel-2) menunjukkan jenis dan fungsi satelit, bukan generasi satelit. Citra-citra Sentinel dapat diakses di laman resmi <https://dataspace.copernicus.eu>.

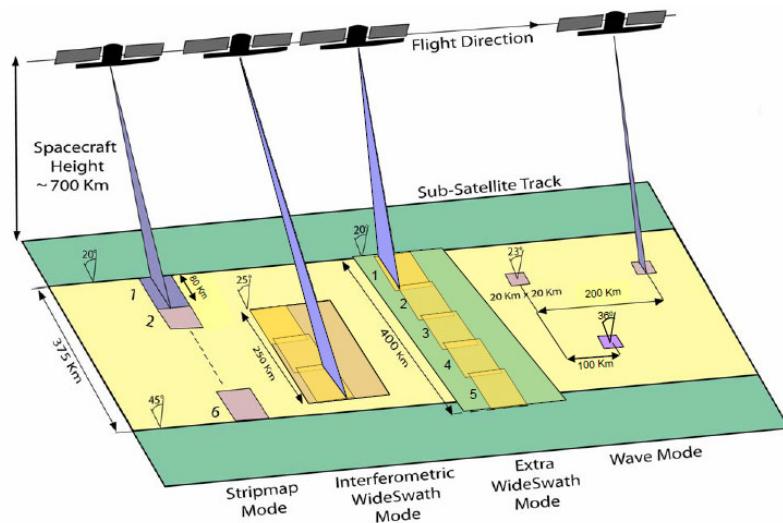


Gambar 1.12. Sentinel Family

([https://www.esa.int/Applications/Observing\\_the\\_Earth/Copernicus/The\\_Sentinel\\_missions](https://www.esa.int/Applications/Observing_the_Earth/Copernicus/The_Sentinel_missions))

**SENTINEL-1 SYNTHETIC APERTURE RADAR (SAR)** adalah misi pencitraan radar yang memiliki orbit polar, merekam untuk segala cuaca, siang dan malam untuk layanan darat dan laut. Sentinel-1A diluncurkan pada 3 April 2014 dan Sentinel-1B pada 25 April 2016. Keduanya dibawa ke orbit dengan roket Soyuz dari Pelabuhan Antariksa Eropa di Guyana Prancis. Misi Sentinel-1B berakhir pada tahun 2022, dan ada rencana untuk meluncurkan Sentinel-1C sesegera mungkin ([https://www.esa.int/Applications/Observing\\_the\\_Earth/Copernicus/The\\_Sentinel\\_missions](https://www.esa.int/Applications/Observing_the_Earth/Copernicus/The_Sentinel_missions)). Sentinel-1 SAR memiliki resolusi temporal 12 hari, dan merupakan C-band SAR dengan berbagai mode akuisisi. Sebagaimana terlihat pada Gambar 1.13.

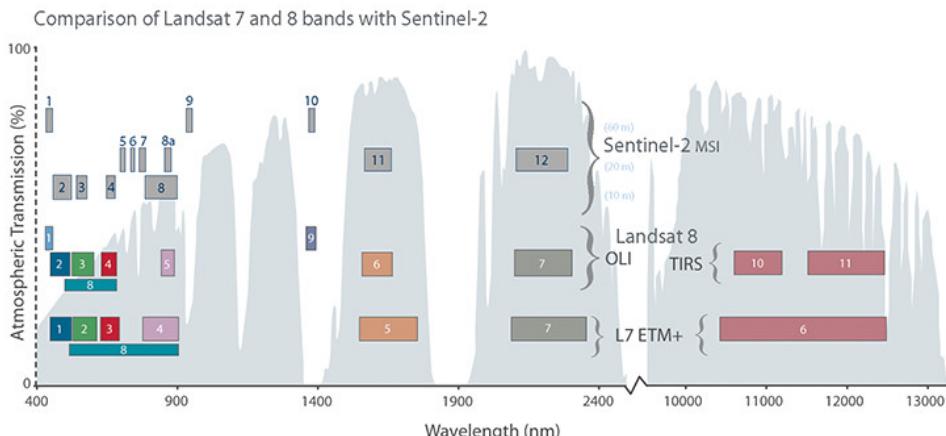
Citra Sentinel-1 yang tersedia untuk seluruh wilayah Indonesia secara *near real-time* adalah Sentinel-1 SAR *Interferometric Wide Swath* (IW) dengan *dual polarization*, yaitu VV dan VH. Sentinel-1 SAR dirilis ke publik dalam beberapa format. **Single-Look Complex (SLC)** yang membawa informasi paling lengkap, baik *amplitude* maupun *phase*. Sentinel-1 SAR SLC diperlukan untuk analisis interferometri, seperti pemetaan pergerakan permukaan bumi (*surface deformation*). **Ground Range Detected (GRD)** hanya membawa informasi *amplitude*, sementara informasi *phase*-nya dihilangkan. Sentinel-1 SAR GRD digunakan untuk analisis *polarimetric SAR*, seperti pemetaan banjir. **Ocean (OCN)** merupakan format Sentinel-1 SAR yang berisi informasi estimasi kecepatan dan arah angin di atas permukaan laut.



Gambar 1.13. Mode akuisisi Sentinel-1 SAR

(<https://sentinels.copernicus.eu/en/web/sentinel/technical-guides/sentinel-1-sar/sar-instrument/acquisition-modes>)

**SENTINEL-2 MULTISPECTRAL INSTRUMENT (MSI)** merupakan misi pencitraan optik multispektral resolusi tinggi yang berorbit polar untuk kepentingan pemantauan daratan, misalnya, vegetasi, penutupan lahan dan air, perairan daratan, dan wilayah pesisir. Sentinel-2A diluncurkan ke orbit pada tanggal 23 Juni 2015 dan Sentinel-2B menyusul pada tanggal 7 Maret 2017 ([https://www.esa.int/Applications/Observing\\_the\\_Earth/Copernicus/The\\_Sentinel\\_missions](https://www.esa.int/Applications/Observing_the_Earth/Copernicus/The_Sentinel_missions)). Sentinel-2 seolah-olah menjadi rival utama seri Landsat. Terlebih dari awal peluncuran, Citra Sentinel-2 sudah didedikasikan untuk terbuka ke seluruh publik. Sentinel-2 memiliki resolusi temporal 10 hari, akan tetapi karena Sentinel-2 merupakan sistem satelit kembar yang memiliki offset 5 hari satu sama lain, sehingga dengan 2 satelit, resolusi temporal Sentinel-2 menjadi 5 hari. Hal ini menjadikan Sentinel-2 efektif untuk pemantauan dinamika objek-objek permukaan bumi dalam interval yang relatif singkat.



Gambar 1.14. Perbandingan band Landsat 7 dan Landsat 8 dengan Sentinel-2  
(<https://landsat.gsfc.nasa.gov/wp-content/uploads/2015/06/Landsat.v.Sentinel-2.png>)

## Bab I Pengantar

Tabel 1.3. Perbandingan band Sentinel-2 dan Landsat 4, 5, 7, 8, 9

Band	Band range (nm)/Band Center (nm)	Spatial Resolution (m)	Name	Landsat-8/9 OLI Band	Landsat-4 TM, 5 TM, 7 ETM+ Band
B1	433-453 / 443	60	Coastal/Aerosol	B1	NA
B2	458-523 / 490	10	Blue	B2	B1
B3	543-578 / 560	10	Green	B3	B2
B4	650-680 / 665	10	Red	B4	B3
B5	698-713 / 705	20	Red Edge 1	NA	NA
B6	734-748 / 740	20	Red Edge 2	NA	NA
B7	765-785 / 783	20	Red Edge 3	NA	NA
B8	785-900 / 842	10	Near Infrared (NIR)	B5	B4
B8a	855-875 / 865	20	Narrow NIR	NA	NA
B9	930-950 / 945	60	Water-vapour	NA	NA
B10	1365-1385 / 1375	60	Cirrus	B9	NA
B11	1565-1655 / 1610	20	Short Wave Infrared-1 (SWIR-1)	B6	B5
B12	2100-2280 / 2190	20	Short Wave Infrared-2 (SWIR-2)	B7	B7

Sentinel-2 membawa payload sensor yang memiliki 13 saluran spektral, dengan variasi resolusi spasial 10 m, 20 m, dan 60 m, serta resolusi radiometrik 12-bit. Jika kita mengunduh Citra Sentinel-2 melalui laman <https://browser.dataspace.copernicus.eu>, kita akan mengunduh Citra Sentinel-2 MSI per tile. Satu tile Sentinel-2 memiliki luasan sekitar 110 km x 110 km (sekitar  $1^{\circ}$ ), dan terdapat interseksi antar tile. Sentinel-2 dirilis ke publik dalam format TOA (Level-1C) dan TOC (Level-2A). Silahkan kunjungi laman <https://sentiwiki.copernicus.eu/web/s2-mission> untuk mempelajari karakteristik Sentinel-2 secara lebih lengkap.



Gambar 1.15. Sentinel-2 MSI tile untuk wilayah Kalimantan Selatan

**SENTINEL-3 OLCI/SLSTR/STM** adalah satelit resolusi spasial medium multisensor yang membawa sensor *Ocean and Land Colour Instrument* (OLCI), *Sea and Land Surface Temperature Radiometer* (SLSTR), dan *Altimetry Surface Topography Mission* (STM). Sehingga secara praktis Sentinel-3 membawa sensor optik, termal, dan radar sekaligus. Sensor OLCI berbasis ENVISAT's *Medium Resolution Imaging Spectrometer* (MERIS). OLCI memiliki 21 band dengan resolusi spasial 300 meter. Selain membawa sensor optik, sensor OLCI juga membawa sensor termal untuk pemetaan *Land Surface Temperature* (LST). SLSTR berbasis ENVISAT's *Advanced Along Track Scanning Radiometer* (AATSR), untuk menentukan *global Sea Surface Temperatures* dengan akurasi lebih baik dari 0.3 Kelvin. SLSTR memiliki resolusi spasial 500 meter dan 1 kilometer. STM merupakan radar dual-frekuensi (Ku dan C-band), dengan resolusi spasial 300 meter. STM ditujukan untuk mengukur *sea surface topography* atau tinggi permukaan air laut (<https://sentiwiki.copernicus.eu/web/s3-mission>).



Gambar 1.16. Satelit Sentinel-3  
(<https://www.isardsat.space/project/s3ng2>)

Sebagaimana Sentinel-2, Sentinel-3 juga merupakan sistem konstelasi satelit kembar, yaitu Sentinel-3A dan Sentinel-3B. Sentinel-3A diluncurkan pada tanggal 16 Februari 2016 dan Sentinel-3B diluncurkan pada tanggal 25 April 2018. Dua satelit Sentinel-3 yang berada di orbit memungkinkan *revisit time* yang singkat, yaitu kurang dari dua hari untuk OLCI dan kurang dari satu hari untuk SLSTR di wilayah ekuator.

Sentinel-3 juga memiliki produk yang disebut S3 SYNERGY. S3 SYNERGY atau SYN merupakan produk kombinasi SLSTR dan OLCI. SYNERGY tersusun atas 3 jenis produk Level-2, yaitu parameter surface reflectance dan aerosol di atas lahan yang diproyeksikan pada grid OLCI 300 m untuk semua saluran SLSTR dan OLCI; produk kontinuitas SPOT yang menghasilkan karakteristik serupa dengan instrumen observasi VEGETASI; dan karakteristik aerosol di darat dan laut diproyeksikan pada grid resolusi spasial 4,5 kilometer (<https://sentiwiki.copernicus.eu/web/s3-synergy>). Jika Anda ingin mempelajari secara lebih detail tentang produk Sentinel-3 SYNERGY, silahkan akses *user handbook*-nya di laman <https://sentinel.esa.int/documents/247904/4598110/Sentinel-3-Synergy-Land-Handbook.pdf>.

## Bab I Pengantar

Tabel 1.4. Sentinel-3 OLCI bands

Name	Description	Wavelength centre (nm)	Resolution (m)
B01	Aerosol correction, improved water constituent retrieval	400	300
B02	Yellow substance and detrital pigments (turbidity)	412.5	300
B03	Chlorophyll absorption maximum, biogeochemistry, vegetation	442.5	300
B04	Chlorophyll	490	300
B05	Chlorophyll, sediment, turbidity, red tide	510	300
B06	Chlorophyll reference (minimum)	560	300
B07	Sediment loading	620	300
B08	2 <sup>nd</sup> Chlorophyll absorption maximum, sediment, yellow substance / vegetation	665	300
B09	Improved fluorescence retrieval	673.75	300
B10	Chlorophyll fluorescence peak, red edge	681.25	300
B11	Chlorophyll fluorescence baseline, red edge transition	708.75	300
B12	O <sub>2</sub> absorption / clouds, vegetation	753.75	300
B13	O <sub>2</sub> absorption / aerosol correction	761.25	300
B14	Atmospheric correction	764.375	300
B15	O <sub>2</sub> absorption used for cloud top pressure, fluorescence over land	767.5	300
B16	Atmospheric / aerosol correction	778.75	300
B17	Atmospheric / aerosol correction, clouds, pixel co-registration	865	300
B18	Water vapour absorption reference. Common reference band with SLSTR. Vegetation monitoring	885	300
B19	Water vapour absorption, vegetation monitoring (maximum REFLECTANCE)	900	300
B20	Water vapour absorption, atmospheric / aerosol correction	940	300
B21	Atmospheric / aerosol correction, snow grain size	1020	300

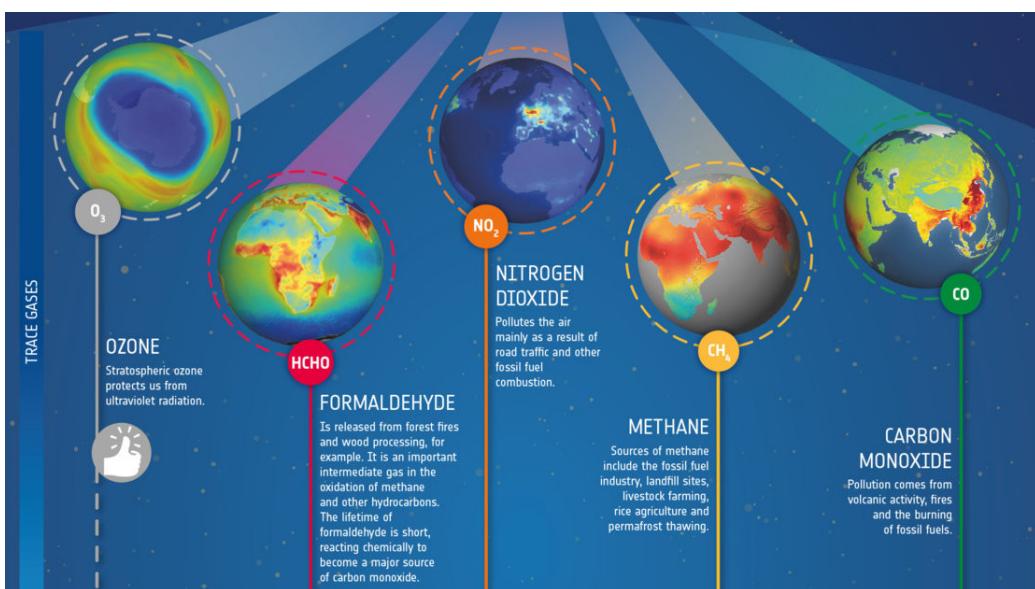
Sumber: <https://docs.sentinel-hub.com/api/latest/data/sentinel-3-olci-l1b>

**SENTINEL-5P (PRECURSOR)** merupakan misi pertama Copernicus yang didedikasikan untuk pemantauan atmosfir bumi. Sentinel-5P diluncurkan pada tanggal 13 Oktober 2017, dengan membawa *TROPOspheric Monitoring Instrument* (TROPOMI). Sentinel-5P membawa 4 band, yaitu Ultraviolet (UV) dan Visible (VIS) (270–495 nm), Near Infrared (NIR) (675–775 nm), dan Shortwave Infrared (SWIR) (2305–2385 nm). Sentinel-5P memiliki resolusi spasial rendah, yaitu 5,5 x 3,5 km (<https://sentiwiki.copernicus.eu/web/s5p-mission>). Misi Sentinel-5P bertujuan untuk memberikan informasi dan layanan mengenai kualitas udara dan iklim antara tahun 2017 dan setidaknya tahun 2023 (<https://docs.sentinel-hub.com>).

Tabel 1.5. Informasi gas-gas atmosfir bumi yang disediakan oleh Sentinel-5P

Product/Band	Physical Quantity	Units
CO	Carbon monoxide total column	(mol/m <sup>2</sup> )
HCHO	Formaldehyde tropospheric vertical column	(mol/m <sup>2</sup> )
NO <sub>2</sub>	Nitrogen dioxide tropospheric column	(mol/m <sup>2</sup> )
O <sub>3</sub>	Ozone total column	(mol/m <sup>2</sup> )
SO <sub>2</sub>	Sulfur dioxide total column	(mol/m <sup>2</sup> )
CH <sub>4</sub>	Column averaged dry air mixing ratio of methane	PPB

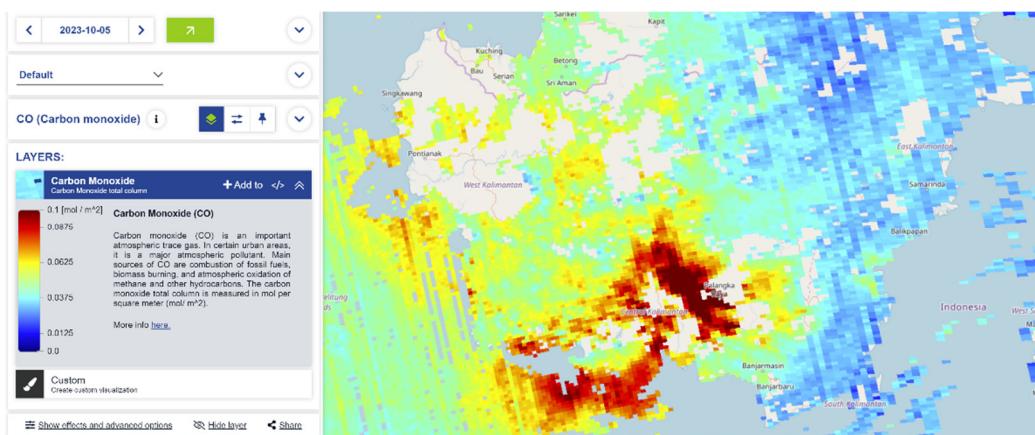
Sumber: <https://docs.sentinel-hub.com>



Gambar 1.17. Data gas-gas atmosfer bumi yang disediakan oleh Sentinel-5P  
<https://www.copernicus.eu/en/sentinel-5p-air-we-breathe>)

Melalui laman <https://browser.dataspace.copernicus.eu> kita dapat mengekstrak gambar atau pun membuat animasi *timelapse* distribusi geospasial gas-gas atmosfer dengan menggunakan produk dari Sentinel-5P. Gambar 1.18 menunjukkan sebaran gas karbonmonoksida (CO) di atas Pulau Kalimantan, pada saat kejadian kebakaran hutan dan lahan di Provinsi Kalimantan Selatan dan Kalimantan Tengah pada tanggal 5 Oktober 2023.

Sebagaimana disajikan pada Tabel 1.5, bahwa produk-produk data gas dari Sentinel-5P memiliki satuan mol/m<sup>2</sup>, kecuali untuk gas metana (CH<sub>4</sub>). Tentu saja, satuan ini dapat dikonversi ke dalam satuan volume sehari-hari, misalnya liter atau meter kubik. Jika diasumsikan gas berada dalam keadaan *Standard Temperature and Pressure* (STP), yaitu temperatur 0° Celcius dan tekanan 1 atmosfir, maka 1 mol gas akan ekivalen dengan 22,4 liter.



Gambar 1.18. Citra gas CO di atas Pulau Kalimantan pada tanggal 5 Oktober 2023  
<https://browser.dataspace.copernicus.eu>)

## MODIS

*Moderate Resolution Imaging Spectroradiometer* (MODIS) merupakan instrument yang terpasang pada Satelit Terra (<https://terra.nasa.gov>) dan Satelit Aqua (<https://aqua.nasa.gov>). Orbit Satelit Terra mengelilingi Bumi diatur sedemikian rupa sehingga melintasi dari utara ke selatan melintasi ekuator di pagi hari, sedangkan Satelit Aqua melewati selatan ke utara melewati ekuator di sore hari. Terra MODIS dan Aqua MODIS mengamati seluruh permukaan bumi setiap 1 hingga 2 hari, memperoleh data dalam 36 band-band spektral, atau grup spektral (<https://modis.gsfc.nasa.gov/about>). MODIS memiliki resolusi radiometrik 12-bit dan resolusi spasial bervariasi antar band, yaitu 250 m (band 1-2), 500 m (band 3-7), dan 1.000 m (band 8-36) (<https://modis.gsfc.nasa.gov/about/specifications.php>).

Tabel 1.6. Spesifikasi Citra MODIS

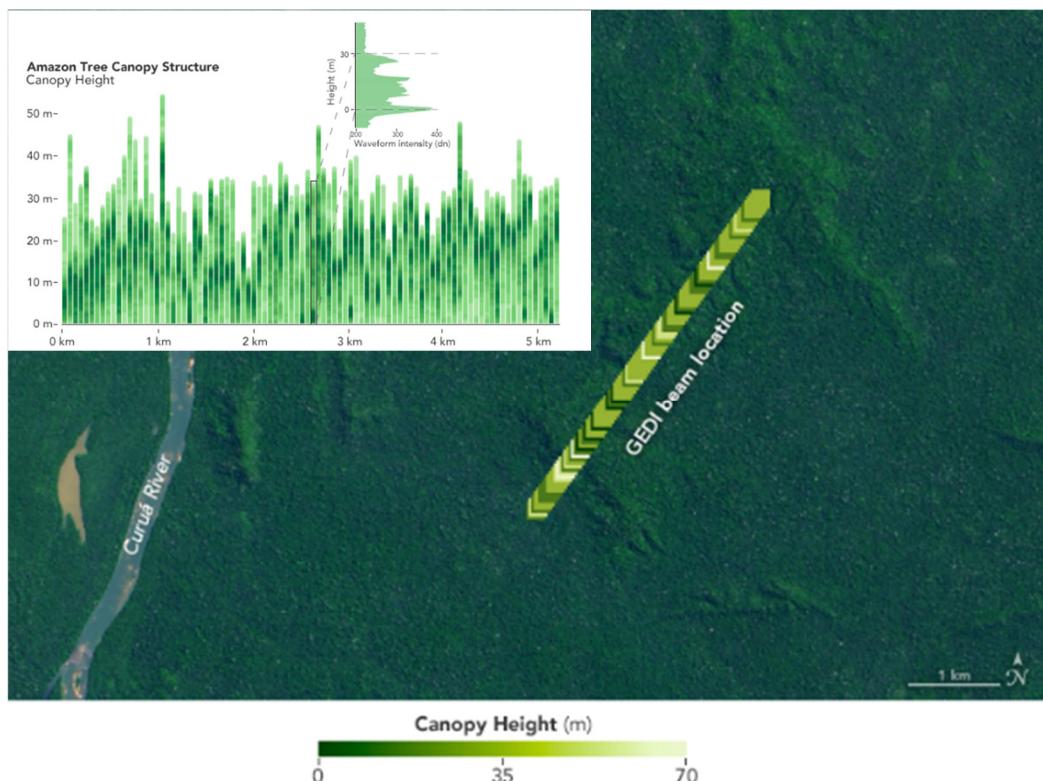
Primary Use	Band	Bandwidth	Radiance (W/m <sup>2</sup> -μm-sr)	Required SNR
Land/Cloud/Aerosols Boundaries	1	620 - 670	21.8	128
	2	841 - 876	24.7	201
Land/Cloud/Aerosols Properties	3	459 - 479	35.3	243
	4	545 - 565	29.0	228
	5	1230 - 1250	5.4	74
	6	1628 - 1652	7.3	275
	7	2105 - 2155	1.0	110
	8	405 - 420	44.9	880
	9	438 - 448	41.9	838
Ocean Color/ Phytoplankton/ Biogeochemistry	10	483 - 493	32.1	802
	11	526 - 536	27.9	754
	12	546 - 556	21.0	750
	13	662 - 672	9.5	910
	14	673 - 683	8.7	1087
	15	743 - 753	10.2	586
	16	862 - 877	6.2	516
	17	890 - 920	10.0	167
	18	931 - 941	3.6	57
	19	915 - 965	15.0	250
Surface/Cloud Temperature	20	3660 - 3840	0.45(300K)	0.05
	21	3929 - 3989	2.38(335K)	0.20
	22	3929 - 3989	0.67(300K)	0.07
	23	4020 - 4080	0.79(300K)	0.07
Atmospheric Temperature	24	4433 - 4498	0.17(250K)	0.25
	25	4482 - 4549	0.59(275K)	0.25
Cirrus Clouds Water Vapor	26	1360 - 1390	6.00	150(SNR)
	27	6535 - 6895	1.16(240K)	0.25
	28	7175 - 7475	2.18(250K)	0.25
Cloud Properties	29	8400 - 8700	9.58(300K)	0.05
Ozone	30	9580 - 9880	3.69(250K)	0.25
Surface/Cloud Temperature	31	10780 - 11280	9.55(300K)	0.05
	32	11770 - 12270	8.94(300K)	0.05
Cloud Top Altitude	33	13185 - 13485	4.52(260K)	0.25
	34	13485 - 13785	3.76(250K)	0.25
	35	13785 - 14085	3.11(240K)	0.25
	36	14085 - 14385	2.08(220K)	0.35

Sumber: <https://modis.gsfc.nasa.gov/about/specifications.php>

MODIS dirilis ke publik dalam berbagai format produk, bahkan produknya sangat bervariasi untuk berbagai keperluan. Salah satu produk MODIS yang sangat terkenal adalah data hotspot yang biasa digunakan untuk pemantauan kejadian kebakaran hutan dan lahan. Data hotspot MODIS dapat diunduh di laman <https://firms.modaps.eosdis.nasa.gov>. Produk-produk MODIS lainnya, seperti *vegetation indices*, *evapotranspiration*, *land surface temperature*, *landcover*, *Leaf Area Index* (LAI), *emissivity*, dan sebagainya, dapat diakses di <https://earthexplorer.usgs.gov> dan <https://search.earthdata.nasa.gov>.

## GEDI

*Global Ecosystem Dynamics Investigation* (GEDI) merupakan instrumen laser atau Lidar yang terpasang di *International Space Station* (ISS). GEDI (dibaca "Jedi" seperti dalam Film *Star Wars*) menyediakan informasi pengukuran yang presisi terhadap tinggi kanopi hutan, struktur vertikal kanopi, dan elevasi permukaan (<https://gedi.umd.edu/return-of-the-gedi>). Data GEDI juga dapat diolah menjadi informasi biomassa, cadangan karbon, hingga sekuestrasi dan emisi karbon. GEDI diluncurkan pada tahun 2018, dan merekam dari 2019 hingga 2024. GEDI memiliki resolusi spasial 25 meter (<https://www.earthdata.nasa.gov/sensors/gedi>). Sensor GEDI sempat mengalami hibernasi (tidak melakukan akuisisi) selama setahun terakhir, yaitu sejak Maret 2023. Akan tetapi aktif kembali mulai 22 April 2024. Sensor GEDI diharapkan akan melakukan akuisisi hingga tahun 2030 (<https://gedi.umd.edu/return-of-the-gedi>). Data GEDI dapat diakses secara terbuka di laman <https://lpdaac.usgs.gov> dan <https://daac.ornl.gov>.



Gambar 1.19. GEDI mengekstrak ketinggian kanopi di Hutan Amazon  
(<https://earthobservatory.nasa.gov/images/146203/a-new-measuring-stick-for-forests>)

## C. Petunjuk Menggunakan Buku Ini

### Untuk siapa buku ini?

Buku ini diperuntukkan bagi siapa saja yang ingin mempelajari teknik-teknik pemrosesan citra digital penginderaan jauh di lingkungan Bahasa Pemrograman Python. Baik yang berbasis komputasi awan menggunakan Google Earth Engine, maupun yang berbasis infrastruktur mesin sendiri menggunakan aplikasi seperti JupyterLab. Buku ini sesuai untuk berbagai kalangan, baik para praktisi di institusi pemerintahan atau perusahaan yang memiliki rutinitas pemrosesan citra penginderaan jauh, para dosen dan peneliti yang biasa mengeksplorasi atau mengembangkan metodologi pemrosesan citra penginderaan jauh, para mahasiswa yang sedang menyelesaikan tugas akhir/skripsi/tesis/dissertasi dengan melibatkan pemrosesan citra penginderaan jauh, atau organisasi-organisasi profesional atau lembaga-lembaga swadaya masyarakat yang sewaktu-waktu ingin memantau permukaan bumi menggunakan citra penginderaan jauh. Bahkan jika Anda adalah seorang pejabat publik, yang sewaktu-waktu memerlukan informasi tematik permukaan bumi secara instan, untuk segera dipaparkan di hadapan publik atau kepala daerah.

Mengingat pembahasan teori penginderaan jauh di dalam buku ini sangat singkat, maka pengguna buku ini diasumsikan sudah memiliki pengetahuan dasar penginderaan jauh maupun informasi geospasial secara umum. Terkait Python sendiri, buku ini berasumsi bahwa pembaca baru pertama kali ini mempelajari Bahasa Python. Meskipun, bagi yang sudah pernah mempelajari Bahasa Python atau pun sudah memiliki latar belakang pengetahuan bahasa pemrograman lain sebelumnya, akan sangat membantu di dalam memahami buku ini. Tentu saja, bagi programer Python berpengalaman, tidak perlu lagi untuk mempelajari Bab II Fundamental Python. Sebagai informasi tambahan, buku ini menggunakan Python 3 (Van Rossum and Drake, 2009).

### Apa yang diperlukan untuk menjalankan buku ini?

Jika Anda menggunakan Google Colab, relatif tidak ada batasan spesifikasi komputer yang diperlukan. Bahkan Anda dapat menggunakan *Tablet PC* atau *Smartphone*. Yang Anda perlukan tentu saja adalah akses internet yang cepat dan stabil, dan aplikasi browser seperti Google Chrome, Mozilla Firefox, Microsoft Edge, Apple Safari, Opera, atau yang lainnya. Jika Anda menggunakan aplikasi JupyterLab atau yang sejenisnya secara offline di mesin Anda sendiri, direkomendasikan Anda menggunakan komputer dengan spesifikasi prosesor minimal Intel Core i7 atau yang setara, RAM minimum 16 GB, dan *Solid Stated Drive (SSD)* hardrive dengan ruang yang cukup. Python sendiri merupakan bahasa pemrograman lintas platform, sehingga Anda bebas menggunakan sistem operasi apa pun. Contoh-contoh yang disajikan di dalam buku ini menggunakan Microsoft Windows 11. Jika Anda menggunakan Mac OS atau Ubuntu Linux, akan ada sedikit penyesuaian kode, terutama pada path atau lokasi file dan folder.

### Bagaimana cara menggunakan buku ini?



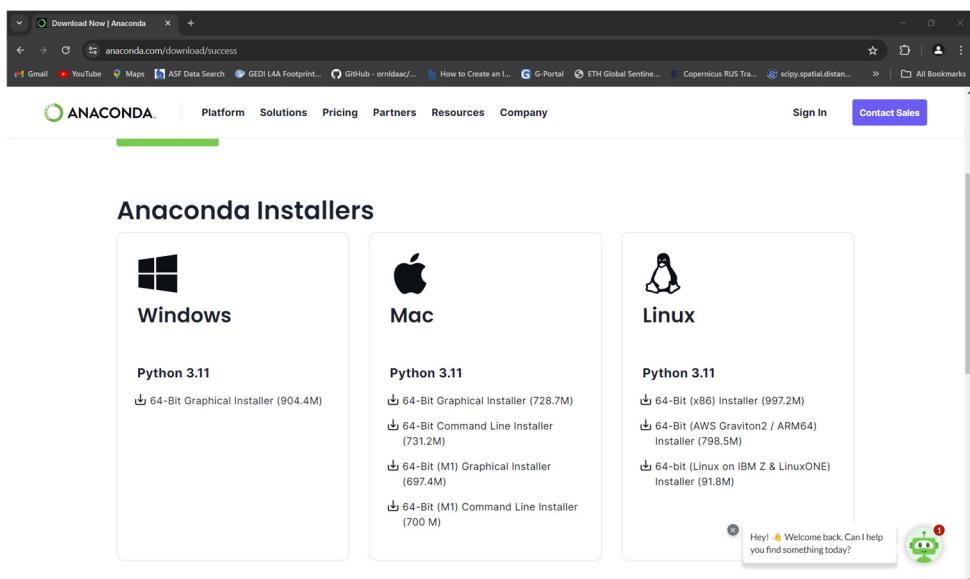
Jika Anda sudah familiar dengan Python dan GEE, buku ini dapat Anda jadikan sebagai bacaan pelengkap penambah pengetahuan. Jika Anda ingin mengikuti tutorial yang ada di dalam buku ini langkah demi langkah, kunjungi terlebih dahulu laman <https://github.com/syamaniulm/geebook>. Anda juga dapat memindai *qr code* di samping atau di sampul depan buku ini. Baca dan ikuti instruksi yang diberikan di laman GitHub tersebut.

## D. Menggunakan Anaconda

Anaconda (<https://anaconda.org/>) (Anaconda Software Distribution, 2020) merupakan platform aplikasi komputasi saintifik untuk Bahasa Python dan R. Anaconda menyediakan semacam lingkungan terintegrasi untuk pemrograman. Anaconda dapat berjalan di sistem operasi Microsoft Windows, GNU Linux, dan Apple Mac. Software Anaconda sendiri gratis, akan tetapi Anaconda menyediakan opsi layanan dan dukungan profesional yang bersifat komersial.

### Instalasi Anaconda

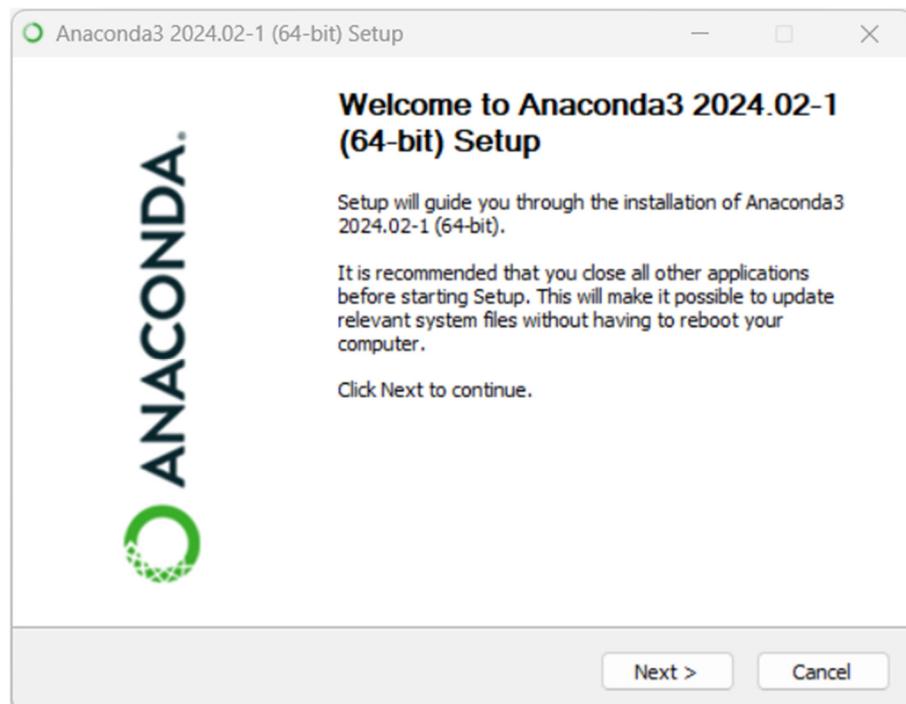
Download perangkat lunak Anaconda di laman <https://www.anaconda.com/download/success>, sesuaikan dengan sistem operasi yang digunakan. Jika Anda menggunakan Windows, download Anaconda untuk Windows.



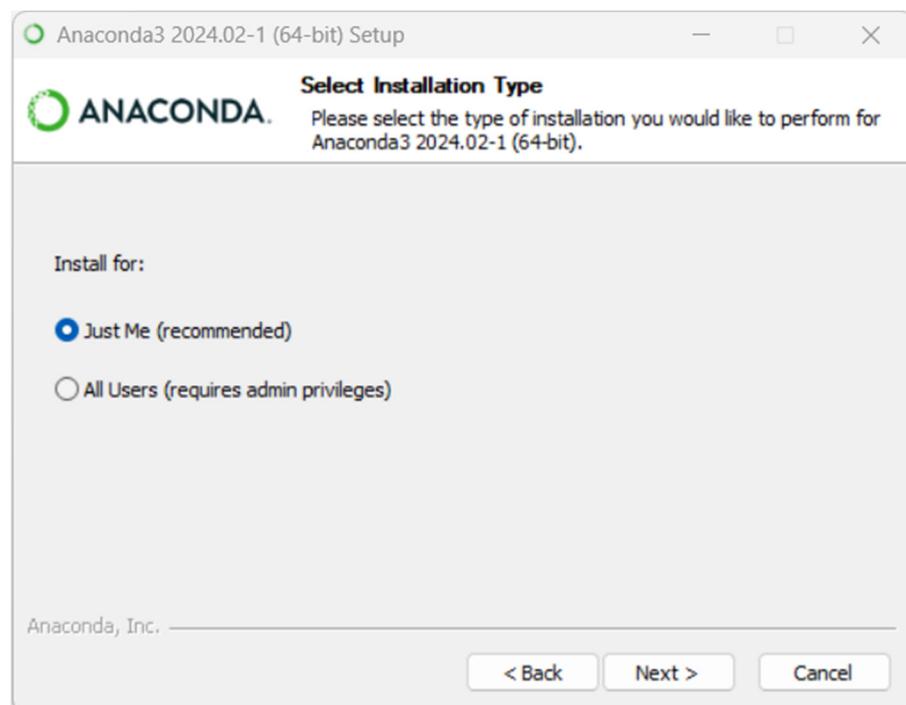
Pada saat tutorial ini dibuat, versi terbaru dari Anaconda adalah Anaconda 3 2024.02-1, yang dirilis pada Februari 2024. Anaconda versi ini sudah terintegrasi dengan Python 3.11. Meskipun jika kita akses laman <https://www.python.org/>, versi Python saat ini adalah Python 3.12. Dan pada saat Anda membaca buku ini, versi Anaconda dan Python terakhir pasti sudah berubah.



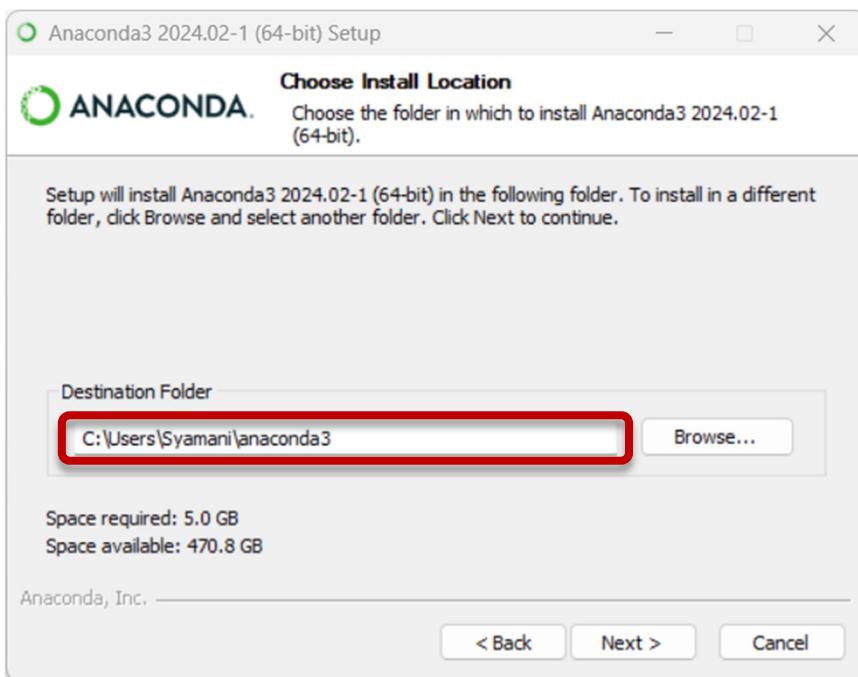
Bagi pengguna Windows, sesudah diunduh, icon perangkat lunak Anaconda akan tampak seperti pada gambar di atas. Untuk melakukan instalasi, klik ganda pada file aplikasi Anaconda yang sudah Anda unduh. Selanjutnya, akan muncul kotak dialog seperti pada gambar di bawah.



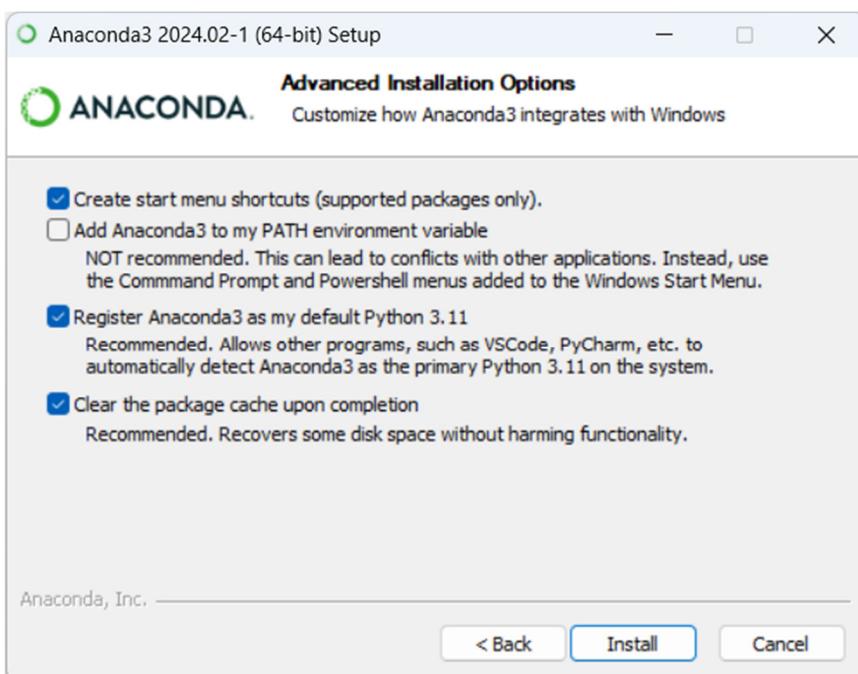
Pada kotak dialog di atas, klik **Next**. Selanjutnya, kita akan di bawah pada kotak dialog seperti pada gambar di bawah.



Pada kotak dialog di atas, pastikan opsi **Just Me** terpilih, kemudian klik **Next**. Berikutnya, akan ditampilkan kotak dialog sebagaimana pada gambar di bawah.

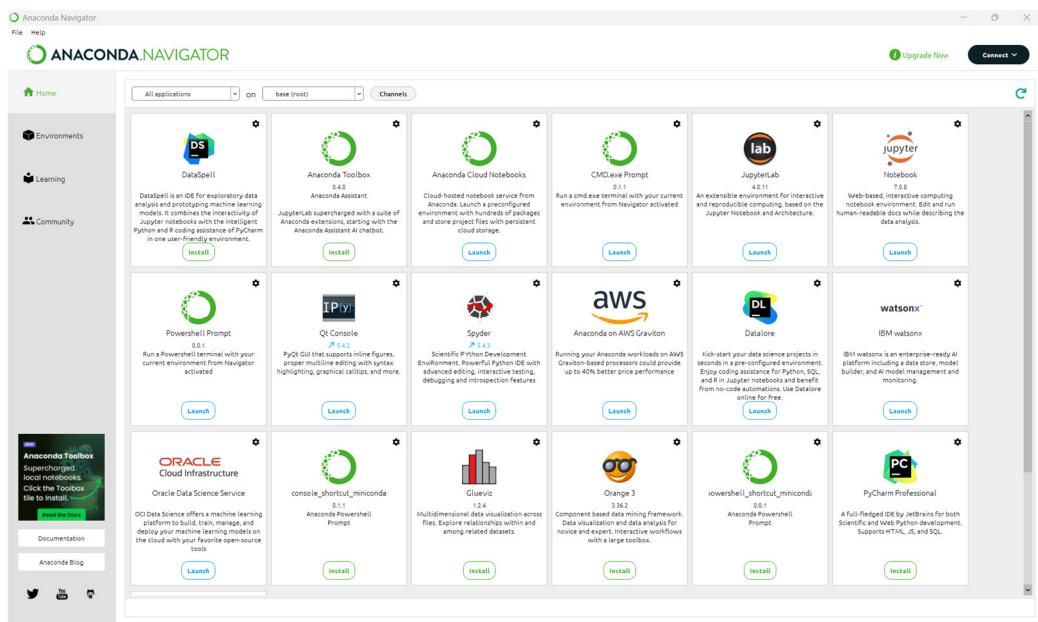


Pada kotak dialog di atas, pastikan Anaconda terinstal di lokasi **C:\Users>NamaUser\anaconda3**, kemudian klik **Next**. Dan kita akan di bawa pada kotak dialog berikutnya seperti gambar di bawah.

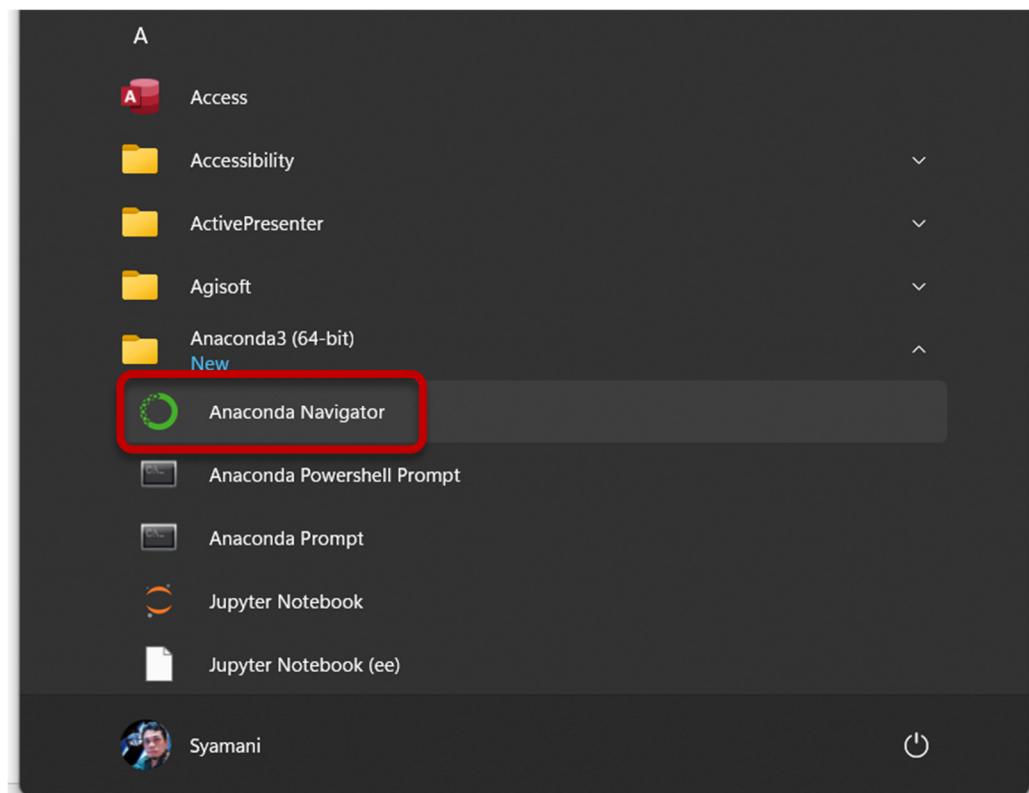


Pada kotak dialog di atas, pastikan opsi yang dipilih sebagaimana yang terlihat pada gambar, kemudian klik **Install**. Proses instalasi akan berlangsung beberapa saat, tergantung kecepatan komputer yang Anda gunakan. Jika proses instalasi selesai, biasanya akan ditampilkan aplikasi Anaconda Navigator seperti pada gambar berikut.

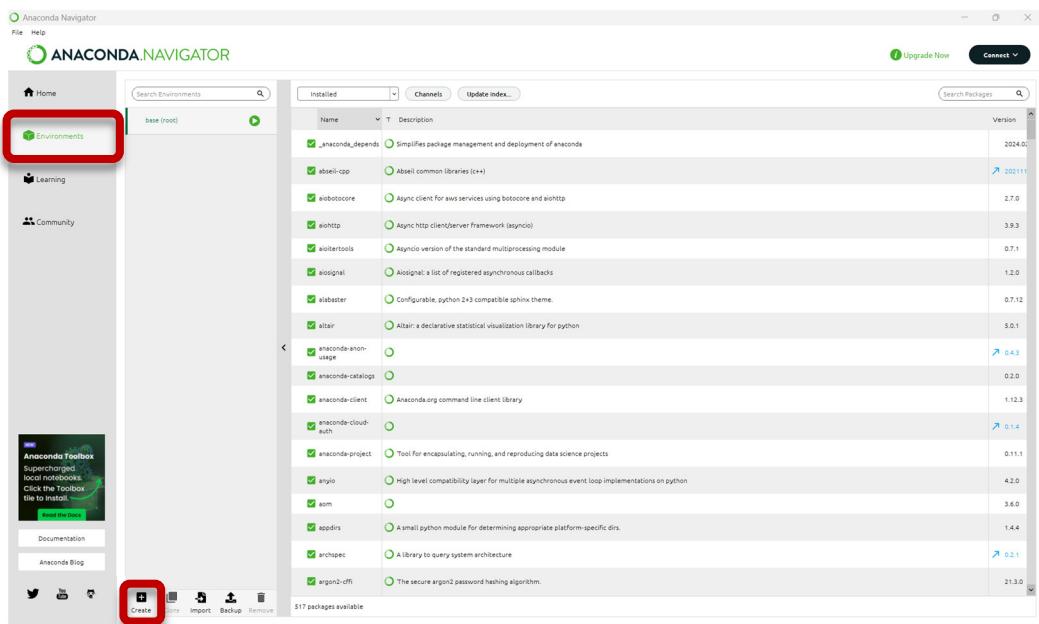
## Bab I Pengantar



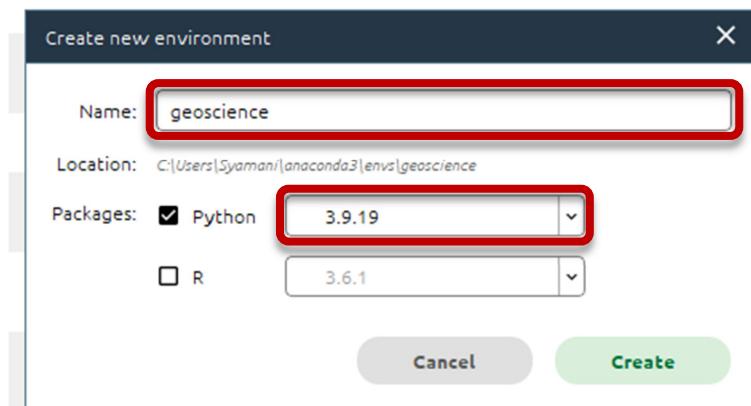
Catatan, jika Anaconda Navigator tidak terbuka pasca instalasi, Anda dapat membukanya secara manual dengan mengklik tombol Start pada taskbar Windows. Kemudian cari dan klik Anaconda Navigator, sebagaimana ditunjukkan pada gambar di bawah.



## Anaconda Environment



Pada aplikasi Anaconda Navigator, klik tab **Environments**, seperti pada gambar di atas. Kemudian klik tombol **Create** pada bagian bawah, sehingga ditampilkan kotak dialog **Create new environment** seperti pada gambar di bawah. Proses pembuatan environment baru harus dilakukan secara online.

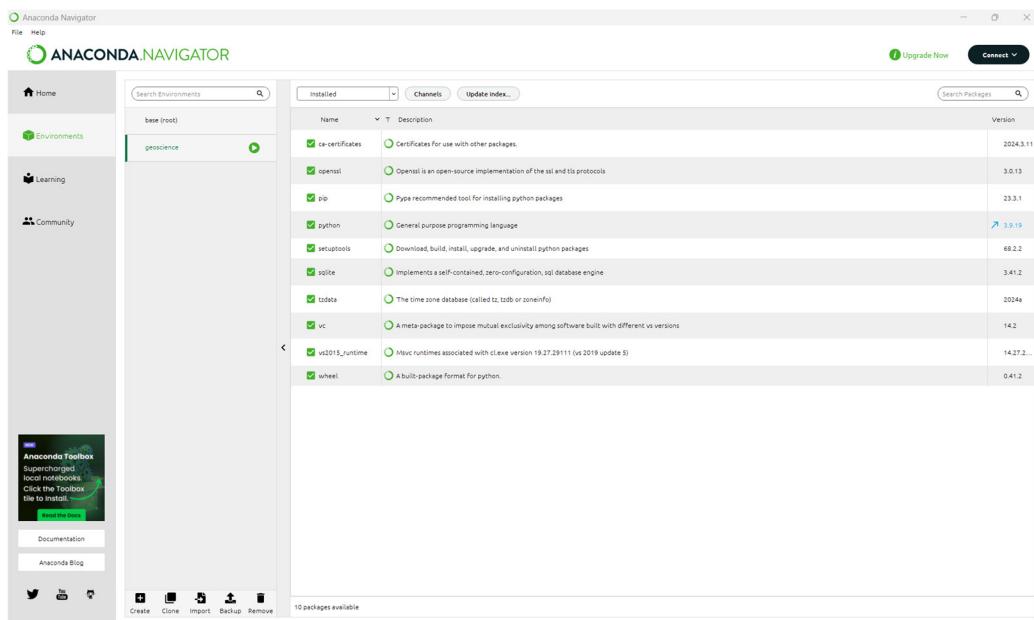


Pada kotak dialog di atas, isikan nama environment yang diinginkan, misalnya **geoscience**, atau nama apapun yang Anda inginkan. Syarat nama environment adalah tidak boleh ada spasi, dan tidak boleh memuat karakter-karakter khusus, seperti /, ?, \*, %, :, ;, dan sebagainya.

Kemudian pilih versi Python yang diinginkan, misalnya Python 3.9.19, kemudian klik tombol **Create**. Sehingga akan terbentuk sebuah Anaconda environment dengan nama **geoscience**.

Anaconda environment adalah semacam lingkungan pengembangan khusus yang terisolasi, sehingga di dalam setiap environment kita dapat memilih versi Python yang diinginkan, dan dapat memilih konfigurasi paket-paket Python yang diinstal. Tujuan pembuatan Anaconda environment adalah untuk menyesuaikan versi Python dengan proyek pemrograman yang kita kerjakan, sekaligus menghindari konflik antar paket-paket yang diinstal. Sebab biasanya, setiap pekerjaan pemrograman Python akan memerlukan versi Python yang berbeda-beda, dan paket-paket Python yang juga berbeda-beda.

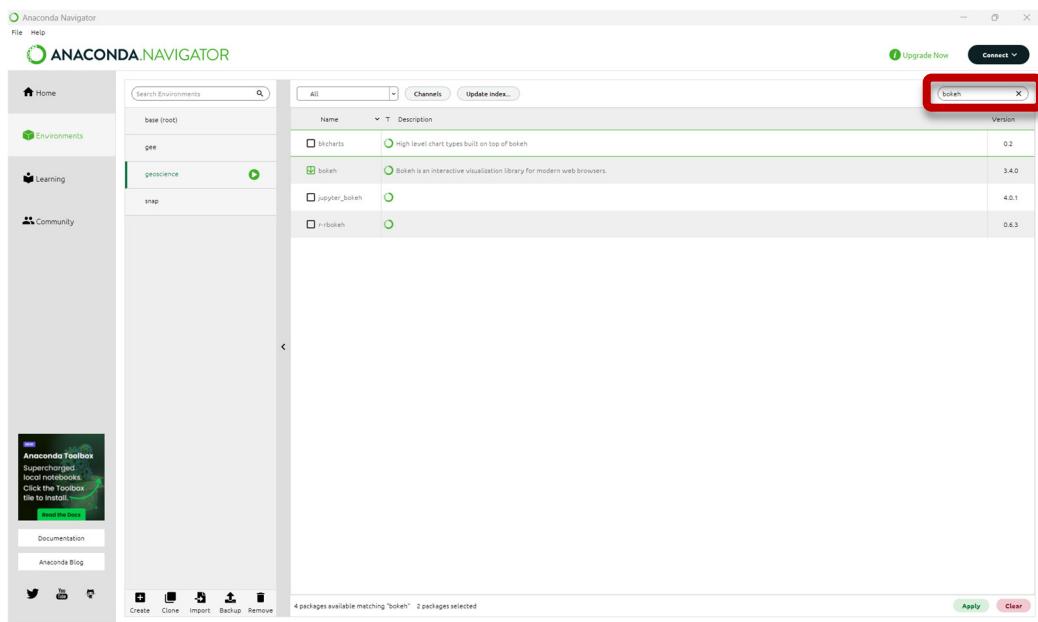
Jika tidak ada masalah (misalnya konflik nama environment) dalam pembuatan environment, maka pada Anaconda Navigator akan ditampilkan sebuah environment baru, misalnya **geoscience** seperti pada gambar di bawah.



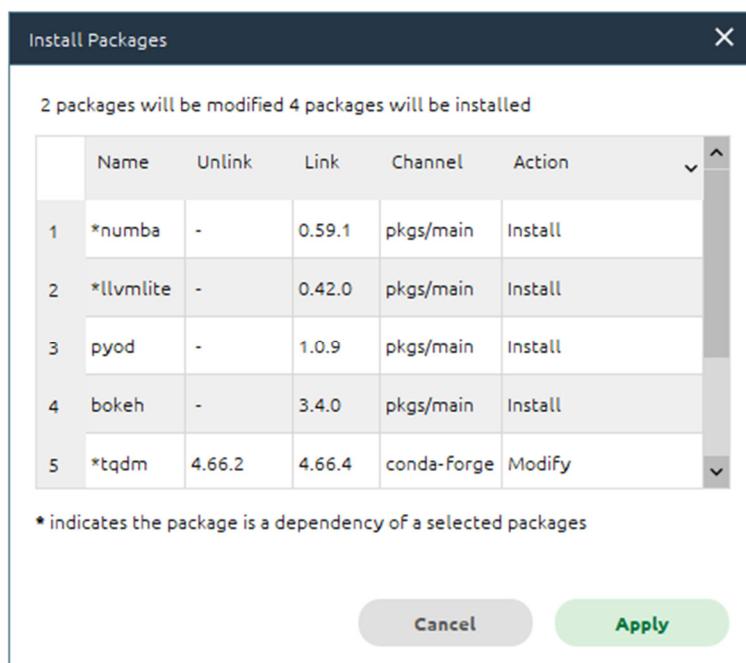
Di dalam environment **geoscience** yang baru kita buat, sudah terdapat paket-paket default Python. Kita dapat menginstal paket-paket Python yang diperlukan, tentu saja proses instalasi paket-paket Python harus dilakukan secara online.

### Instalasi Paket Python

Untuk menginstal paket Python, pilih environment terlebih dahulu. Kemudian pada panel kanan, pilih opsi **All** pada kolom di samping tombol **Channels**, kemudian di kolom **Search Packages**, ketik nama paket yang akan diinstal, misalnya **bokeh** (<https://bokeh.org>). Setelah nama paket ditemukan, klik (centang) nama paket, kemudian klik tombol **Apply** di bagian bawah. Sebagaimana terlihat pada gambar berikut.



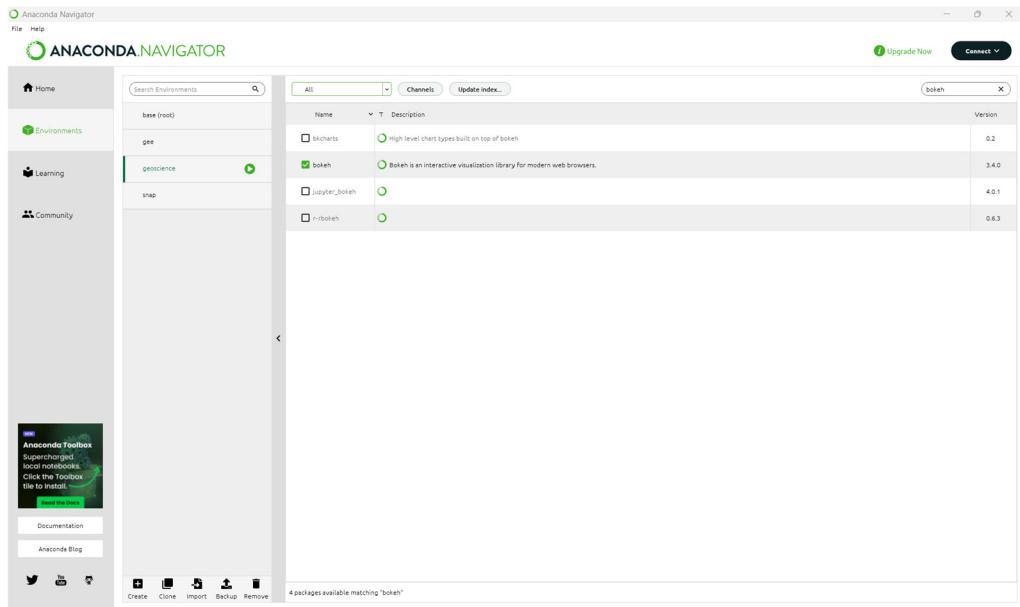
Jika ditampilkan kotak dialog seperti pada gambar di bawah, itu artinya Anaconda akan meminta untuk instalasi paket-paket pendukung atau dependensi dari paket yang akan kita instal, yaitu **bokeh**. Klik tombol **Apply** pada kotak dialog seperti pada gambar di bawah.



Proses instalasi akan berlangsung beberapa saat, tergantung kecepatan komputer dan kecepatan akses internet. Catatan, selama proses instalasi paket, akan terjadi download sejumlah file dari internet. Jadi pastikan koneksi internet tidak terputus dan dalam keadaan stabil. Instalasi akan terhenti dan gagal jika akses internet kita bermasalah.

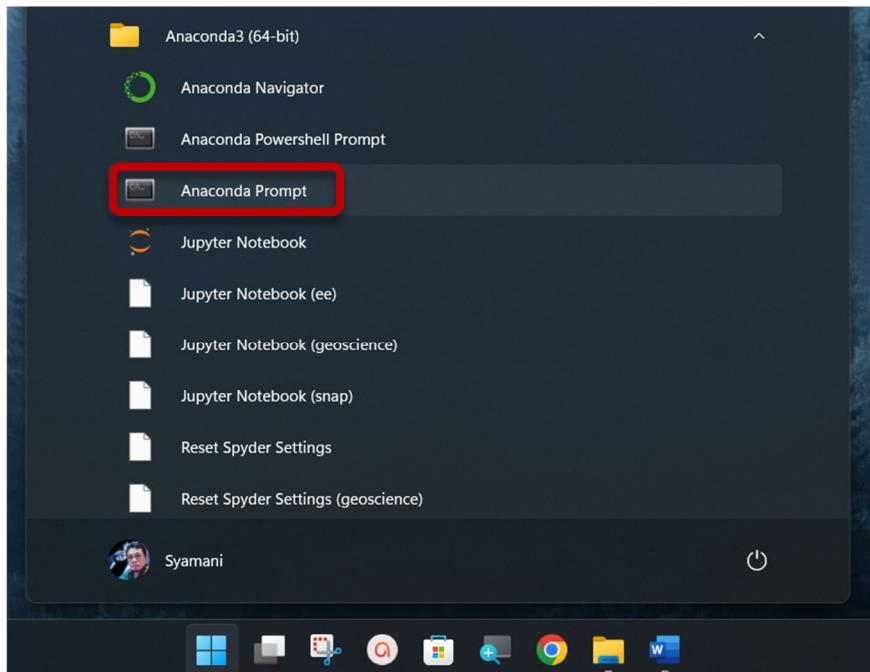
## Bab I Pengantar

Berikut adalah tampilan Anaconda Navigator setelah instalasi paket **bokeh** selesai. Sebagai informasi, bokeh adalah paket yang akan digunakan untuk visualisasi interaktif di web browser.



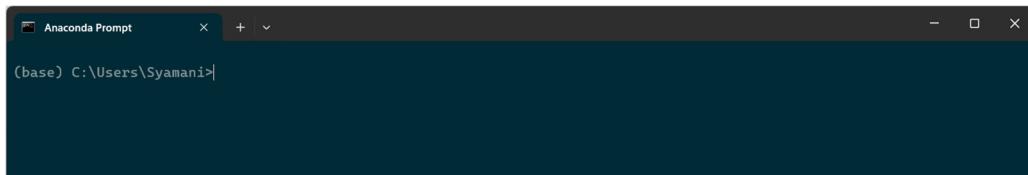
Alternatif lain untuk instalasi paket adalah melalui terminal atau Anaconda Prompt. Cara ini pada umumnya lebih disukai oleh para programer, sebab kita dapat lebih leluasa untuk memilih versi-versi paket yang akan diinstal.

Untuk membuka Anaconda Prompt, klik tombol Start Windows, kemudian cari dan klik Anaconda Prompt, seperti pada gambar di bawah.



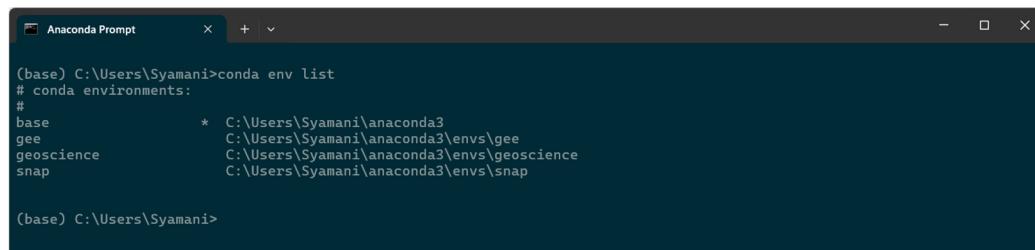
Untuk memudahkan ke depannya, klik kanan pada Anaconda Prompt seperti pada gambar di atas, kemudian pilih dan klik Pin to taskbar. Sehingga tombol Anaconda Prompt akan muncul di taskbar Windows. Hal ini akan memudahkan kita untuk mengaksesnya nanti.

Setelah kita mengklik Anaconda Prompt, akan tampil jendela terminal seperti pada gambar berikut.



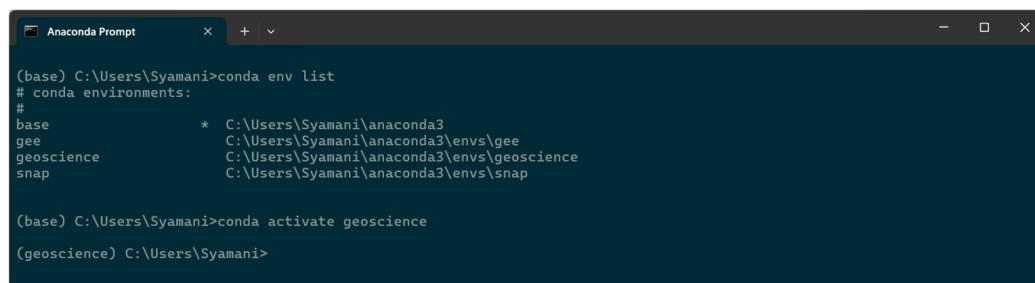
Notasi **(base) C:\Users\Nama User>** menandakan kita sedang berada di lingkungan dasar (*base*) Anaconda. Kita sangat tidak direkomendasikan untuk bekerja atau pun menginstal paket-paket Python di lingkungan *base* ini. Sebab nanti jika terjadi konflik atau *crash*, maka kita harus menginstal ulang Anaconda. Kita harus masuk ke dalam environment yang sudah kita buat terlebih dahulu, baru kemudian kita bekerja atau menginstal paket-paket Python yang diinginkan.

Ketikkan perintah **conda env list** dan tekan tombol **Enter** pada keyboard, untuk menampilkan seluruh environment yang ada di dalam Anaconda. Sebagaimana pada gambar di bawah.



```
(base) C:\Users\Syamani>conda env list
# conda environments:
#
base          *  C:\Users\Syamani\anaconda3
gee           C:\Users\Syamani\anaconda3\envs\gee
geoscience    C:\Users\Syamani\anaconda3\envs\geoscience
snap          C:\Users\Syamani\anaconda3\envs\snap
```

Untuk masuk ke dalam kedalam salah satu environment, misalnya **geoscience**, ketikkan perintah **conda activate geoscience** dan tekan **Enter**. Selanjutnya, kita akan berada di dalam environment **geoscience**. Sebagaimana pada gambar di bawah.



```
(base) C:\Users\Syamani>conda env list
# conda environments:
#
base          *  C:\Users\Syamani\anaconda3
gee           C:\Users\Syamani\anaconda3\envs\gee
geoscience    C:\Users\Syamani\anaconda3\envs\geoscience
snap          C:\Users\Syamani\anaconda3\envs\snap

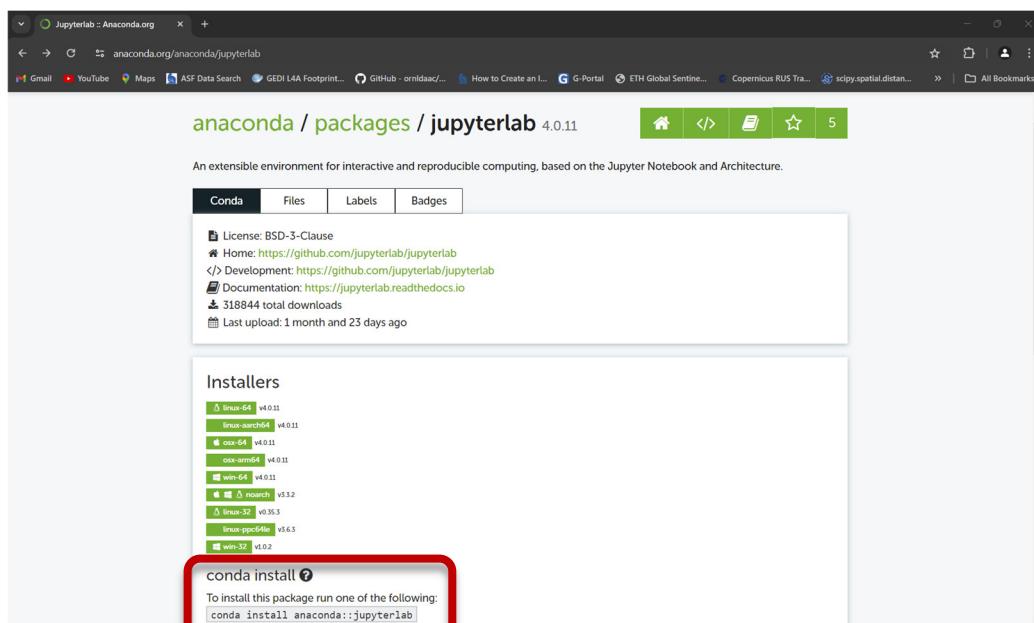
(base) C:\Users\Syamani>conda activate geoscience
(geoscience) C:\Users\Syamani>
```

Perhatikan gambar di atas, notasinya sekarang sudah berubah menjadi **(geoscience) C:\Users\Nama User>**. Hal ini menandakan kita sudah berada di dalam environment **geoscience**.

### Menggunakan JupyterLab

JupyterLab (*Julia Python Interpreter Laboratory*) (Kluyver et al., 2016) adalah lingkungan pengembangan interaktif berbasis web terbaru untuk notebook, kode, dan data. Antarmukanya yang fleksibel memungkinkan pengguna untuk mengonfigurasi dan mengatur alur kerja dalam ilmu data, komputasi ilmiah, jurnalisme komputasi, dan pembelajaran mesin. Desain modular mengundang ekstensi untuk memperluas dan memperkaya fungsionalitas (<https://jupyter.org/>).

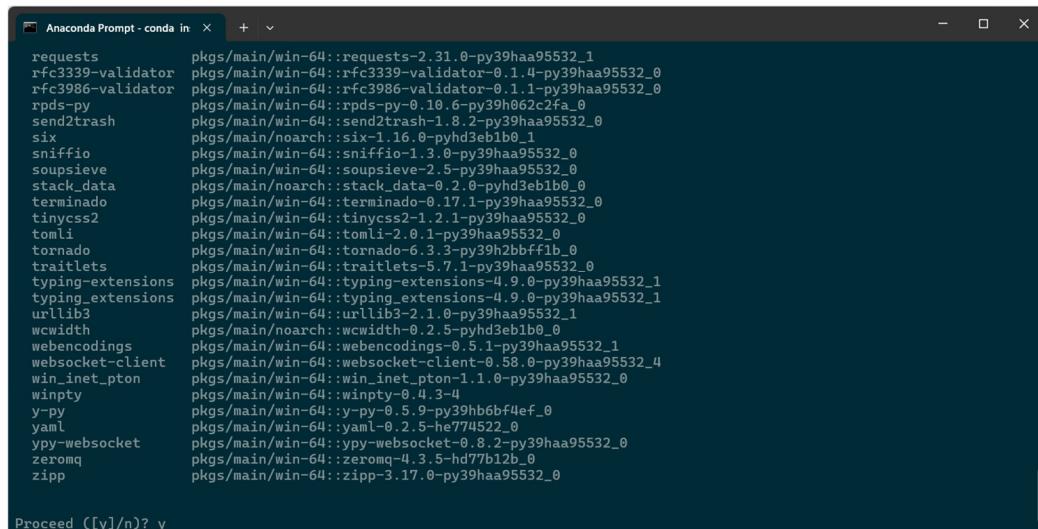
Untuk menginstal JupyterLab via Anaconda Prompt, terlebih dahulu buka laman <https://anaconda.org/anaconda/jupyterlab>. Anda bisa googling dengan kata kunci **conda jupyter lab** untuk menemukan laman ini.



Copy-paste perintah **conda install anaconda::jupyterlab** yang ada pada laman di atas ke Anaconda Prompt, kemudian tekan Enter, sebagaimana gambar di bawah.

```
(base) C:\Users\Syamani>conda env list
# conda environments:
#
base          *  C:\Users\Syamani\anaconda3
gee            C:\Users\Syamani\anaconda3\envs\gee
geoscience     C:\Users\Syamani\anaconda3\envs\geoscience
snap           C:\Users\Syamani\anaconda3\envs\snap

(base) C:\Users\Syamani>conda activate geoscience
(geoscience) C:\Users\Syamani>conda install anaconda::jupyterlab
```



```

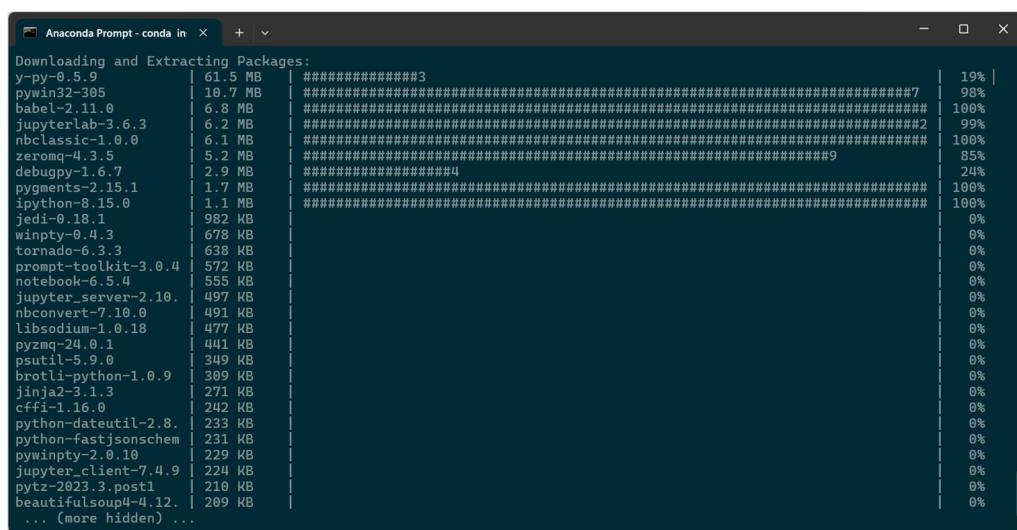
Anaconda Prompt - conda in + v

requests          pkgs/main/win-64::requests-2.31.0-py39haa95532_1
rfc3339-validator pkgs/main/win-64::rfc3339-validator-0.1.4-py39haa95532_0
rfc3986-validator pkgs/main/win-64::rfc3986-validator-0.1.1-py39haa95532_0
rpds-py           pkgs/main/win-64::rpds-py-0.10.6-py39haa95532_0
send2trash        pkgs/main/noarch::send2trash-1.8.2-py39haa95532_0
six               pkgs/main/win-64::six-1.16.0-pyhd3eb1b0_1
sniffio           pkgs/main/win-64::sniffio-1.3.0-py39haa95532_0
soup sieve        pkgs/main/win-64::soup sieve-2.5-py39haa95532_0
stack_data        pkgs/main/noarch::stack_data-0.2.0-pyhd3eb1b0_0
terminado         pkgs/main/win-64::terminado-0.17.1-py39haa95532_0
tinycc2           pkgs/main/win-64::tinycc2-1.2.1-py39haa95532_0
tomli             pkgs/main/win-64::tomli-2.0.1-py39haa95532_0
tornado           pkgs/main/win-64::tornado-6.3.3-py39h2bbff1b_0
traitlets         pkgs/main/win-64::traitlets-5.7.1-py39haa95532_0
typing-extensions pkgs/main/win-64::typing-extensions-4.9.0-py39haa95532_1
typing_extensions pkgs/main/win-64::typing_extensions-4.9.0-py39haa95532_1
urllib3           pkgs/main/win-64::urllib3-2.1.0-py39haa95532_1
wcwidth            pkgs/main/noarch::wcwidth-0.2.5-pyhd3eb1b0_0
webencodings      pkgs/main/win-64::webencodings-0.5.1-py39haa95532_1
websocket-client  pkgs/main/win-64::websocket-client-0.58.0-py39haa95532_4
win_inet_pton     pkgs/main/win-64::win_inet_pton-1.1.0-py39haa95532_0
winpty             pkgs/main/win-64::winpty-0.4.3-4
y-py               pkgs/main/win-64::y-py-0.5.9-py39hb6bf4ef_0
yaml               pkgs/main/win-64::yaml-0.2.5-he774522_0
yppy-websocket    pkgs/main/win-64::yppy-websocket-0.8.2-py39haa95532_0
zeromq            pkgs/main/win-64::zeromq-4.3.5-hd77b12b_0
zipp               pkgs/main/win-64::zipp-3.17.0-py39haa95532_0

Proceed ([y]/n)? y

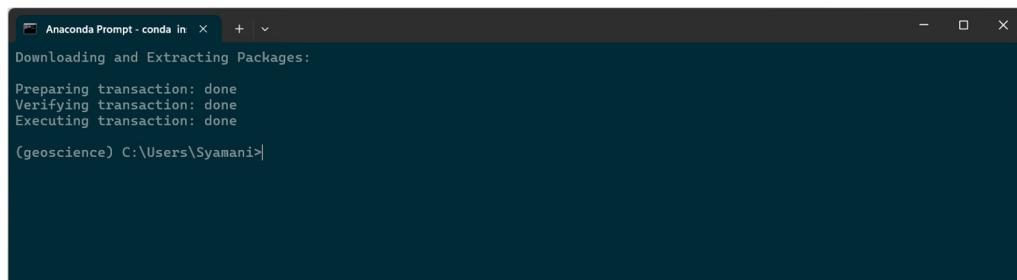
```

Anaconda akan mencari semua paket dependensi dari JupyterLab. Jika ditampilkan opsi seperti pada gambar di atas, untuk melanjutkan instalasi ketik **y** dan tekan Enter.



Paket	Ukuran	Status	Progress
y-py-0.5.9	61.5 MB	#####	19%
pywin32-305	10.7 MB	#####	98%
babel-2.11.0	6.8 MB	#####	100%
jupyterlab-3.6.3	6.2 MB	#####	99%
nbclassic-1.0.0	6.1 MB	#####	100%
zeromq-4.3.5	5.2 MB	#####	85%
debugpy-1.6.7	2.9 MB	#####	24%
pygments-2.15.1	1.7 MB	#####	100%
ipython-8.15.0	1.1 MB	#####	100%
jedi-0.18.1	982 KB	#####	0%
wipty-0.4.3	678 KB	#####	0%
tornado-6.3.3	638 KB	#####	0%
prompt-toolkit-3.0.4	572 KB	#####	0%
notebook-6.5.4	555 KB	#####	0%
jupyter_server-2.10.	497 KB	#####	0%
nbsconvert-7.10.0	491 KB	#####	0%
Libsodium-1.0.18	477 KB	#####	0%
pymzmq-24.0.1	441 KB	#####	0%
psutil-5.9.0	349 KB	#####	0%
brotli-python-1.0.9	309 KB	#####	0%
jinja2-3.1.3	271 KB	#####	0%
cffi-1.16.0	242 KB	#####	0%
python-dateutil-2.8.	233 KB	#####	0%
python-fastjsonschem	231 KB	#####	0%
pywipty-2.0.10	229 KB	#####	0%
jupyter_client-7.4.9	224 KB	#####	0%
pytz-2023.3.post1	210 KB	#####	0%
beautifulsoup4-4.12.	209 KB	#####	0%
... (more hidden) ...		#####	0%

Proses instalasi paket JupyterLab akan berlangsung beberapa saat, tergantung kecepatan komputer dan akses internet. Jika proses instalasi sudah berhasil, akan ditampilkan seperti gambar di bawah.



```

Anaconda Prompt - conda in + v

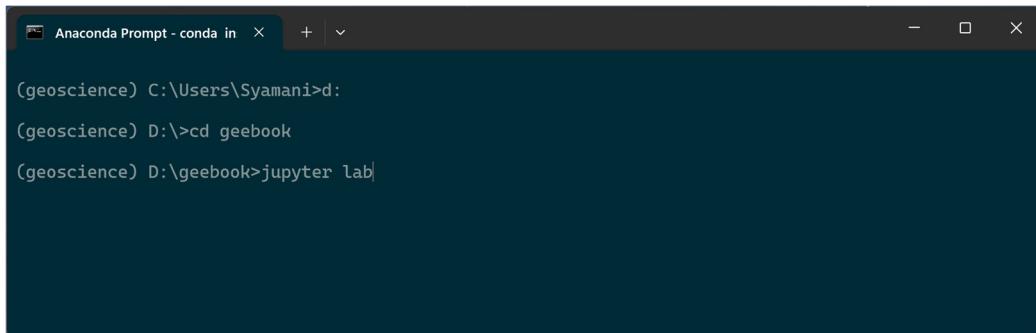
Downloading and Extracting Packages:
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(geoscience) C:\Users\Syamani>

```

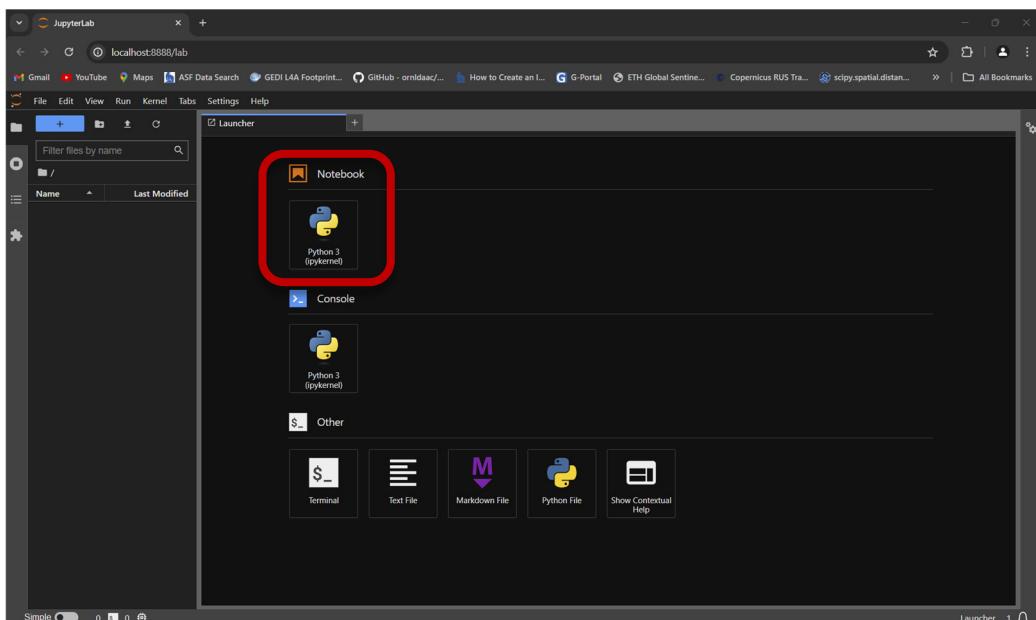
## Bab I Pengantar

Berpindahlah ke drive dan folder tempat proyek kita, misalnya proyek kita ada di dalam drive **D** folder **geebook**. Ketikkan perintah **d**: tekan Enter, kemudian **cd geebook** dan tekan Enter. Setelah kita berada di lokasi atau path **D:\geebook**, selanjutnya untuk menjalankan JupyterLab, ketikkan perintah **jupyter lab**, dan tekan **Enter**.



```
Anaconda Prompt - conda in × + ▾ (geoscience) C:\Users\Syamani>d: (geoscience) D:>cd geebook (geoscience) D:\geebook>jupyter lab
```

Kita akan dibawa ke jendela aplikasi JupyterLab sebagaimana gambar berikut.

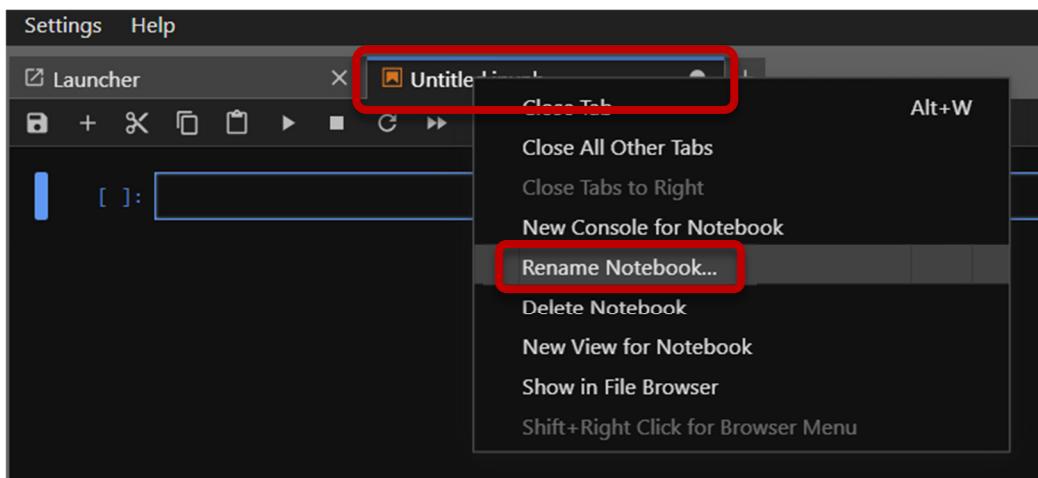


Untuk membuat sebuah notebook baru, klik tombol **Python 3 (ipykernel)** di bawah tulisan **Notebook**. Sebagaimana pada gambar di atas. Alternatifnya adalah klik menu **File → New → Notebook**. Selanjutnya, akan terbentuk sebuah notebook baru yang masih kosong, sebagaimana gambar di bawah.

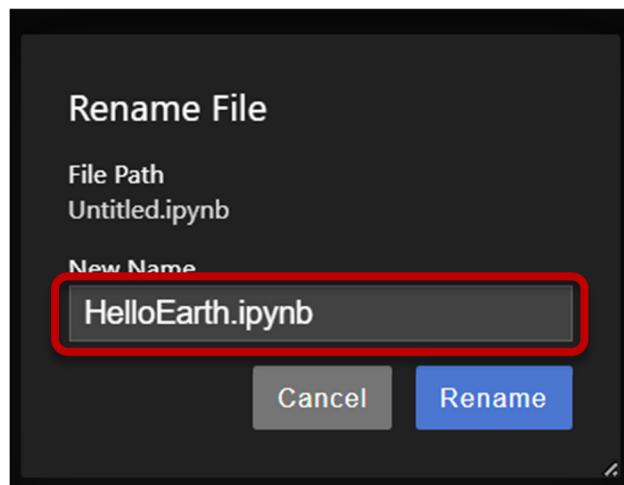
---

*Notebook adalah sebuah file berbasis web yang nantinya akan menjadi tempat kita mengetikkan kode-kode Python, berikut melihat luaran dari kode-kode program yang kita jalankan.*

---

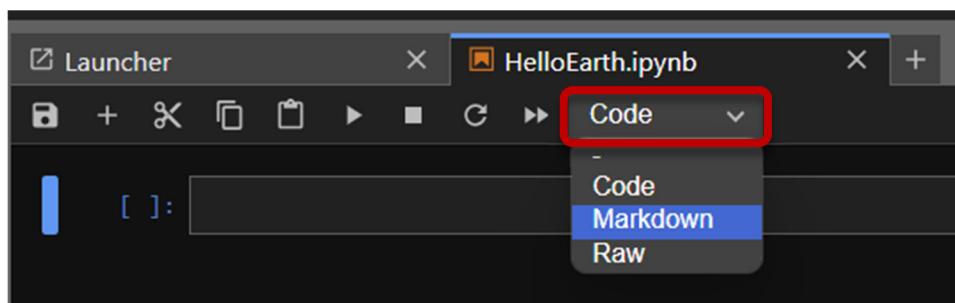


Untuk merubah nama notebook, klik kanan pada tab nama notebook sebagaimana ditunjukkan pada gambar di atas. Kemudian klik **Rename Notebook**.



Beri nama yang diinginkan, misalnya **HelloEarth.ipynb**, kemudian klik tombol **Rename**. Seperti pada gambar di atas. Catatan, ekstensi **ipynb** tidak boleh dihapus atau dirubah.

Setelah nama notebook berubah, klik **Code**, kemudian pilih dan klik **Markdown** sebagaimana ditunjukkan pada gambar di bawah.

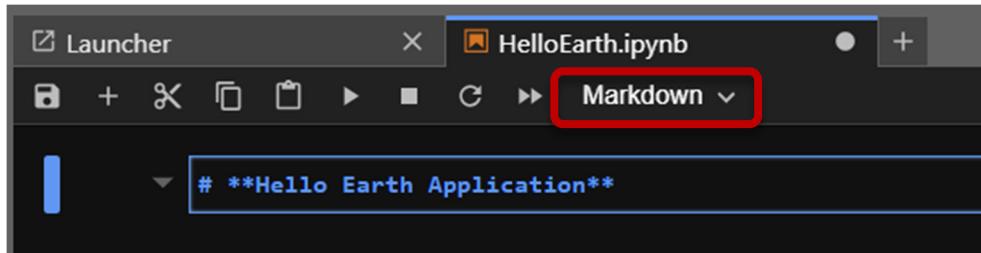


## Bab I Pengantar

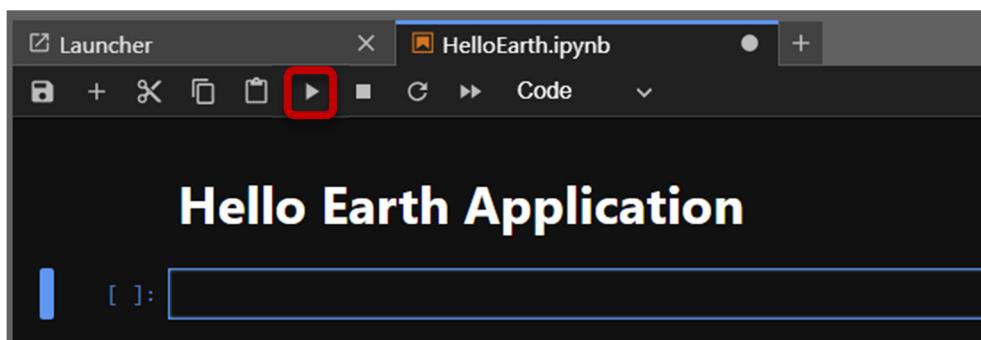
Kemudian pada sel notebook ketikkan teks berikut:

```
# **Hello Earth Application**
```

Perhatikan gambar berikut:



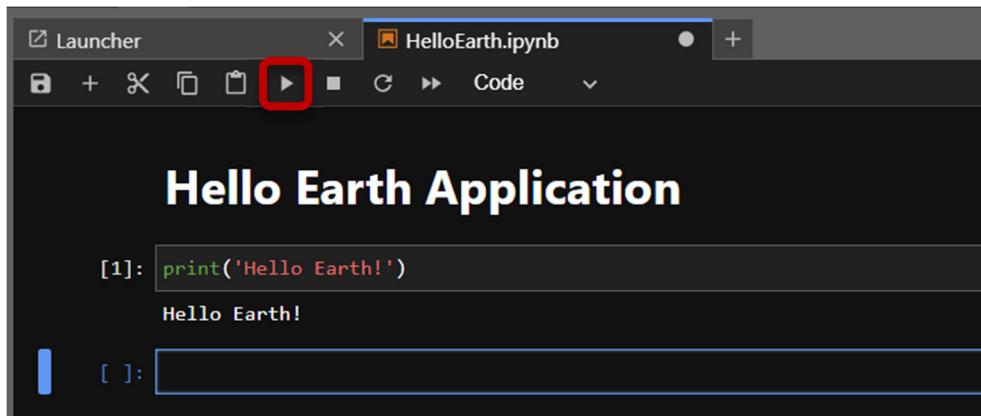
Kemudian klik tombol Run (▶), dan perhatikan hasilnya sebagaimana pada gambar berikut.



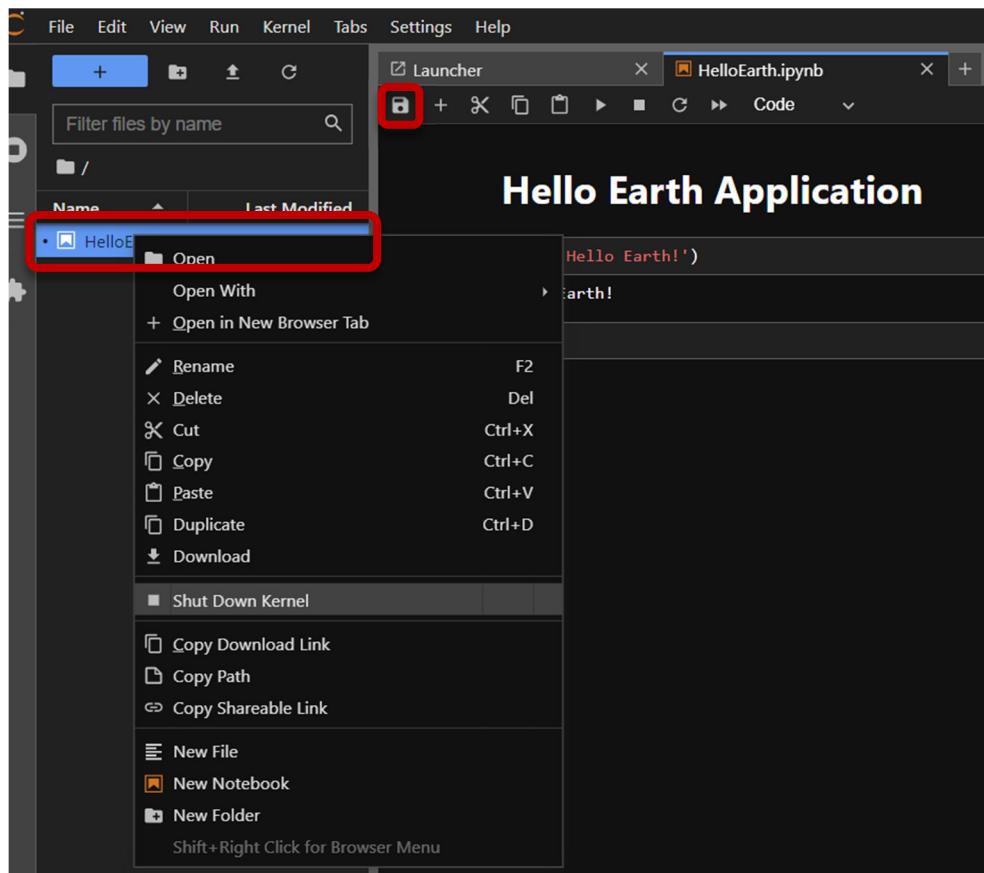
Selanjutnya, pada sel di bawahnya, ketikkan kode berikut:

```
print('Hello Earth!')
```

Kemudian klik tombol Run (▶), dan perhatikan hasilnya sebagaimana pada gambar berikut.

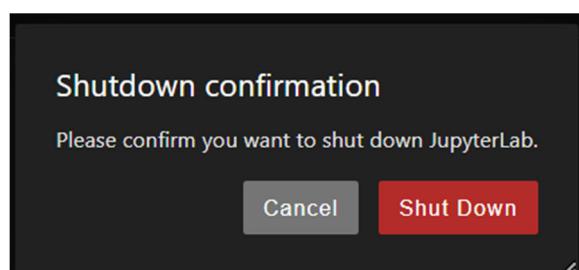


Sampai di sini, Anda sudah berhasil membuat sebuah aplikasi berbasis Python pertama Anda. Tulisan putih **Hello Earth!** yang muncul di bawah kode merupakan luaran dari kode program yang kita buat dan kita jalankan. Klik tombol **Save** (disk) untuk menyimpan notebook.

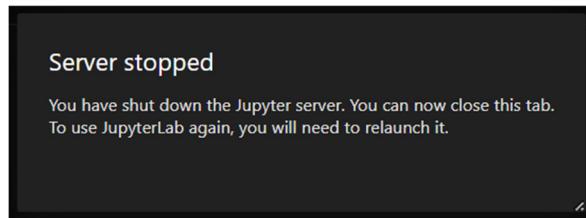


Untuk keluar dari sebuah notebook, klik tanda silang di samping nama **HelloEarth.ipynb** ( ). Kemudian pada daftar nama notebook, klik kanan nama notebook yang mau ditutup, dan pada menu yang muncul pilih dan klik **Shut Down Kernel**.

Untuk keluar dari JupyterLab, klik menu **File → Shut Down**.



Pada kotak dialog seperti pada gambar di atas, klik tombol **Shut Down**.



Setelah muncul informasi **Server stopped**, tutup manual jendela web JupyterLab sebagaimana kita menutup laman web biasa.

A screenshot of an Anaconda Prompt window titled "Anaconda Prompt - conda in". The terminal output shows the shutdown process of a Jupyter Notebook kernel. It includes logs from "ServerApp" and "geoscience" environments, and ends with the command "exit".

```
[I 2024-11-20 08:35:02.747 ServerApp] 404 GET /api/contents/HelloEarth.ipynb?content=0&hash=0&1732062902741 (e44e54b18afb475482c0fe4bd7da6b97@::1) 0.00ms referer=http://localhost:8888/lab
[I 2024-11-20 08:35:02.749 ServerApp] 404 GET /api/contents/HelloEarth.ipynb?content=0&hash=0&1732062902741 (::1): No such file or directory: HelloEarth.ipynb
[I 2024-11-20 08:35:06.867 ServerApp] Creating new notebook in
[I 2024-11-20 08:35:07.196 ServerApp] Kernel started: 77015f6d-1ed5-4ff3-820e-3fbb71981af8
[I 2024-11-20 08:35:07.857 ServerApp] Connecting to kernel 77015f6d-1ed5-4ff3-820e-3fbb71981af8.
[I 2024-11-20 08:35:07.863 ServerApp] Connecting to kernel 77015f6d-1ed5-4ff3-820e-3fbb71981af8.
[I 2024-11-20 08:35:07.883 ServerApp] Connecting to kernel 77015f6d-1ed5-4ff3-820e-3fbb71981af8.
[I 2024-11-20 08:35:26.455 ServerApp] Saving file at /HelloEarth.ipynb
[I 2024-11-20 08:35:27.710 ServerApp] Starting buffering for 77015f6d-1ed5-4ff3-820e-3fbb71981af8:bec74e79-bad9-41ea-b941-3535ecaaba79
[I 2024-11-20 08:35:29.930 ServerApp] Kernel shutdown: 77015f6d-1ed5-4ff3-820e-3fbb71981af8
[I 2024-11-20 08:35:33.749 ServerApp] Shutting down on /api/shutdown request.
[I 2024-11-20 08:35:33.750 ServerApp] Shutting down 4 extensions

(geoscience) D:\geeebook>conda deactivate
(base) D:\geeebook>exit
```

Kita akan dibawa kembali ke Anaconda Prompt. Jalankan perintah **conda deactivate** dan tekan **Enter**, untuk keluar dari environment. Setelah kembali ke base, kemudian ketik **exit** dan tekan **Enter** untuk keluar dari terminal Anaconda Prompt.

---

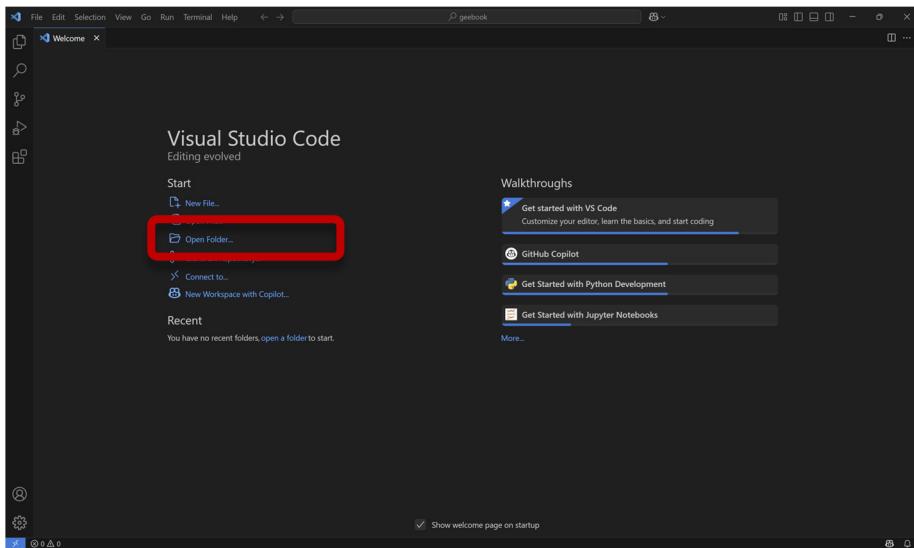
*Penting untuk selalu melakukan prosedur Shut Down dan deactivate environment jika kita sudah selesai bekerja menggunakan Notebook, JupyterLab, atau Anaconda Prompt. Dan jangan biasakan menutup jendela aplikasi JupyterLab atau Anaconda Prompt secara paksa tanpa prosedur Shut Down, sebab hal ini dapat menimbulkan potensi kerusakan file, hilangnya informasi penting, atau crash aplikasi di kemudian hari.*

---

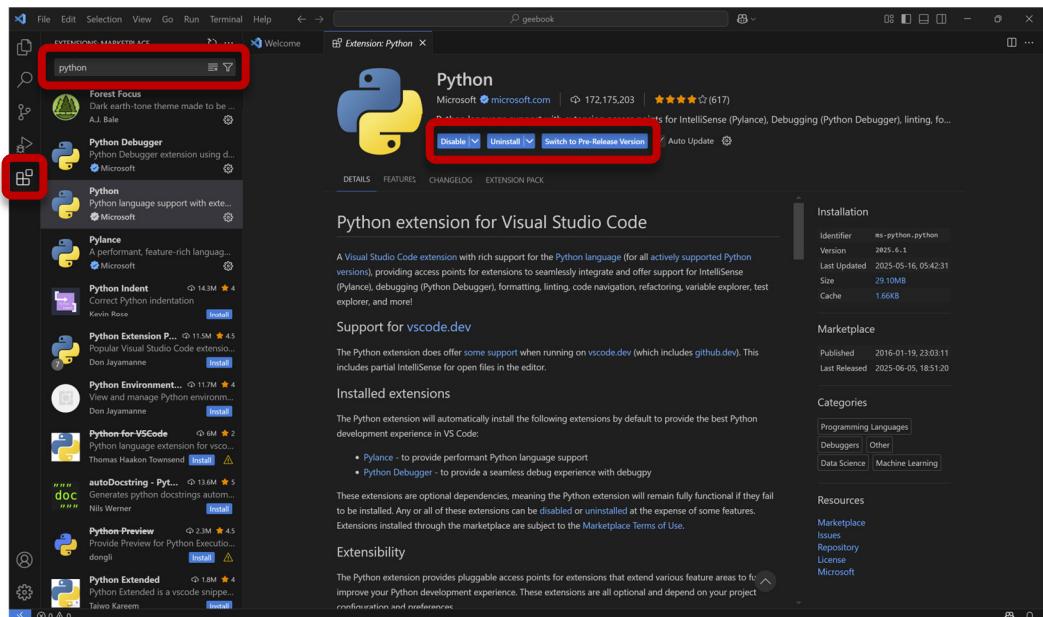
## Menggunakan Visual Studio Code

VS Code (*Visual Studio Code*) (<https://code.visualstudio.com/>) adalah sebuah platform lingkungan pengembangan lintas bahasa pemrograman. Meskipun dikembangkan oleh Microsoft Corporation, VS Code bersifat lintas sistem operasi dan gratis. VS Code dikembangkan dari kode open source di bawah lisensi MIT, akan tetapi rilis resminya memuat beberapa elemen properti milik Microsoft, sehingga tidak seluruh kode programnya open source. Versi full open source dari VS Code adalah VS Codium (<https://vscode.codium.com/>). Sebagai pengguna VS Code, kita tidak perlu khawatir untuk menggunakan versi resmi VS Code dari Microsoft, sebab sepenuhnya gratis untuk digunakan. Kecuali jika kita ingin terlibat dalam pengembangan source code aplikasinya, kita direkomendasikan untuk menggunakan VS Codium.

VS Code sangat mudah diinstal, unduh versi resminya di <https://code.visualstudio.com/> sesuai sistem operasi yang Anda gunakan. Kemudian instal sebagaimana Anda menginstal perangkat lunak biasa. Setelah diinstal, VS Code dapat langsung dijalankan sebagaimana menjalankan aplikasi biasa dengan mengklik icon . Perhatikan jendela VS Code seperti pada gambar berikut:



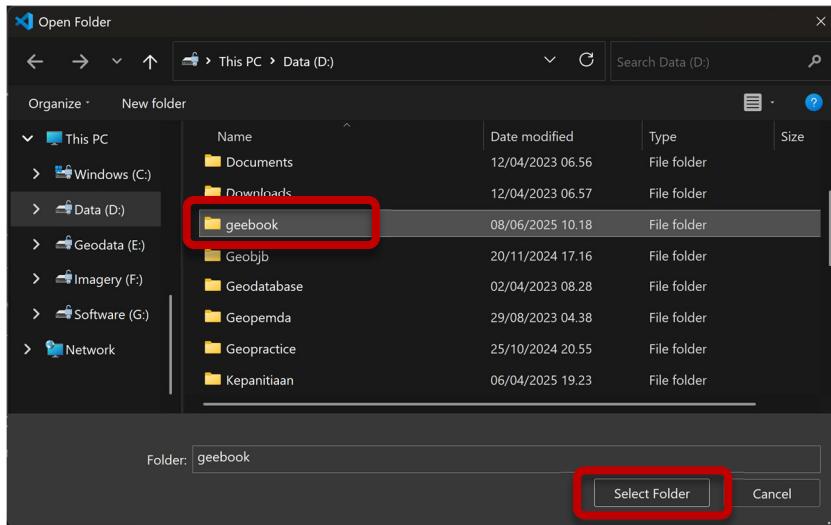
Agar kita dapat menggunakan Bahasa Python di lingkungan VS Code, kita harus menginstal ekstensi Python. Sebagaimana terlihat pada gambar berikut:



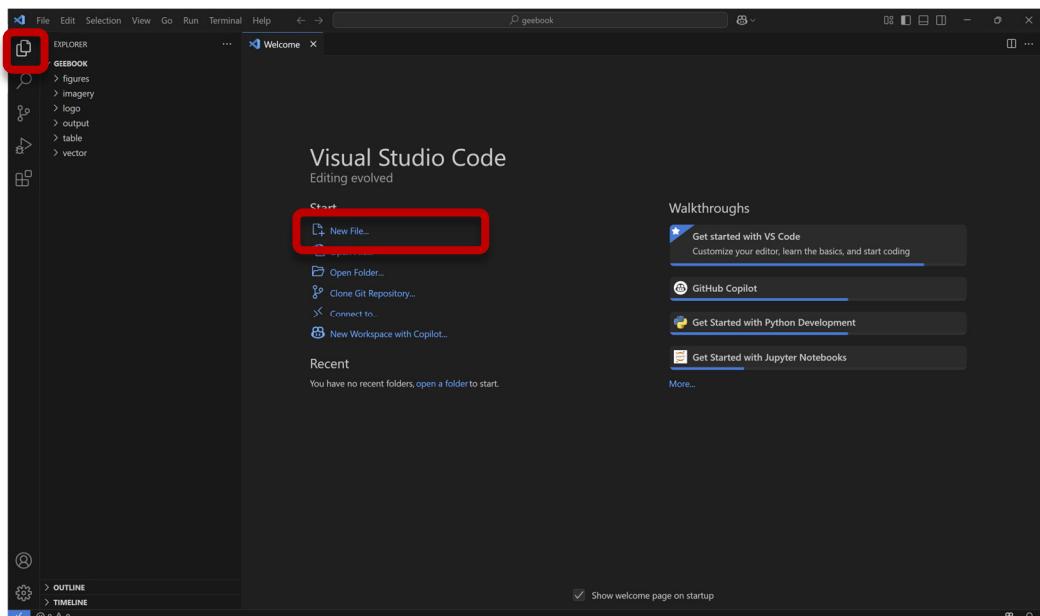
Dalam keadaan online, klik tombol **Extensions**, sebagaimana ditunjukkan pada gambar di atas. Kemudian ketik **Python** pada kolom **Search Extensions in Marketplace** sebagaimana ditunjukkan pada gambar di atas. Selanjutnya instal ekstensi **Python**, termasuk ekstensi-ekstensi lainnya yang diperlukan. Tanda centang () menunjukkan bahwa ekstensi tersebut sudah terinstal.

## Bab I Pengantar

Untuk memulai bekerja dengan VS Code, Anda harus membuka sebuah folder terlebih dahulu. Caranya, klik **File → Open Folder**, atau langsung klik link **Open Folder** pada laman depan jendela VS Code sebagaimana terlihat pada gambar terdahulu. Selanjutnya, pilih folder kerja Anda, dan klik tombol **Select Folder**, sebagaimana terlihat pada gambar berikut:

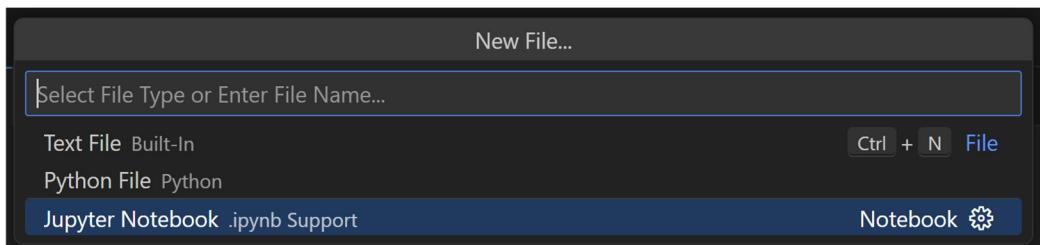


Setelah folder terbuka, klik tombol **Explorer**, sebagaimana ditunjukkan pada gambar di bawah, maka struktur folder kerja kita akan terlihat.

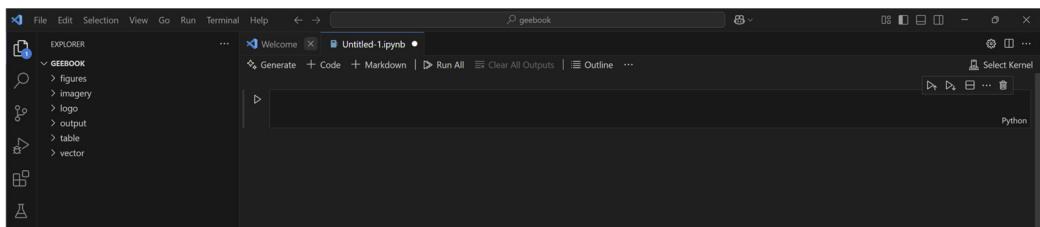


VS Code kompatibel dengan file Python (.py) atau pun file notebook (.ipynb) yang kita buat di dalam JupyterLab. Sehingga Anda dapat berpindah-pindah antara JupyterLab atau VS Code nantinya, tanpa harus menyesuaikan atau menulis ulang kode-kode Python Anda. Agar VS Code kompatibel dengan file notebook, Anda juga harus menginstal ekstensi **Jupyter**, yaitu melalui **Extensions** sebagaimana sudah dijelaskan pada saat instalasi ekstensi Python sebelumnya.

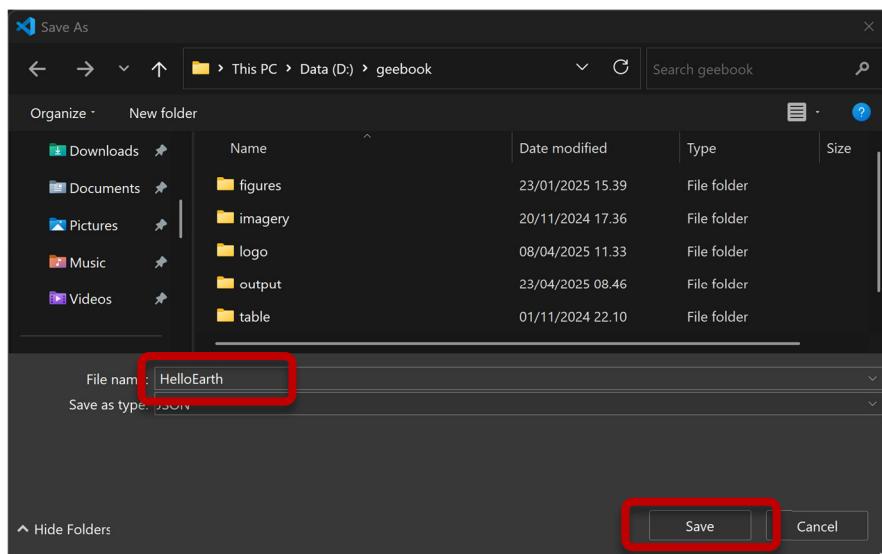
Untuk membuat sebuah file kode baru di dalam VS Code, klik **File → New File** atau langsung klik tombol **New File** sebagaimana ditunjukkan pada gambar di atas. Kemudian pada kolom **Command Palette** di bagian tengah atas jendela VS Code, pilih **Jupyter Notebook**, sebagaimana terlihat pada gambar berikut:



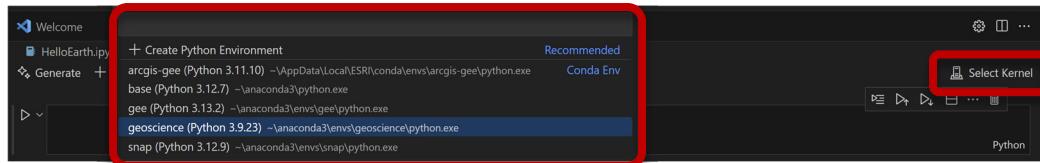
Selanjutnya, akan muncul sebuah file Jupyter Notebook baru dengan nama **Untitled-1.ipynb**, sebagaimana terlihat pada gambar berikut:



Simpan file Jupyter Notebook dengan klik **File → Save As** atau tekan tombol **Ctrl + Shift + S**.

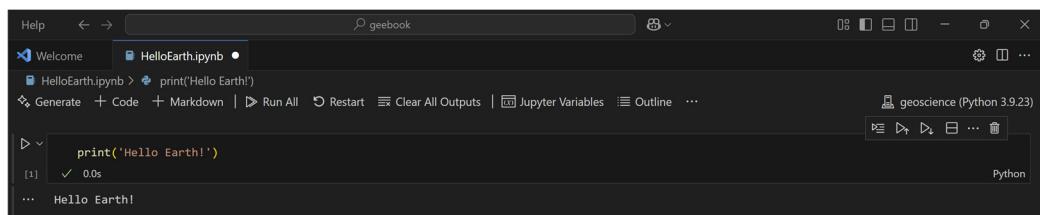


Beri nama file notebook, misalnya **HelloEarth**, kemudian klik tombol **Save**. Untuk mengkoneksikan file Jupyter Notebook dengan Anaconda Environment yang sudah kita buat sebelumnya, klik tombol **Select Kernel** di sudut kanan atas jendela VS Code, maka pada kolom **Command Palette** akan muncul sejumlah opsi. Pilih opsi **Select Another Kernel**, kemudian **Python Environments**, selanjutnya pilih Anaconda Environment yang kita inginkan, misalnya **geoscience**. Sebagaimana terlihat pada gambar berikut:



Pada **Command Palette**, kita juga dapat membuat sebuah Python environment. Python environment yang dibuat di dalam VS Code akan tersimpan di dalam folder kerja kita. Akan tetapi, Anda direkomendasikan untuk tetap menggunakan Anaconda Environment, agar nantinya kode-kode Python yang kita tulis dapat fleksibel ketika harus berpindah antar platform, misalnya dari VS Code ke JupyterLab atau Spyder. Sekaligus untuk menghindari pembuatan terlalu banyak environment yang mungkin berfungsi sama dan akan sangat memboroskan ruang penyimpanan. Sebaiknya Anda membuat Python environment di dalam VS Code hanya jika Anda tidak menginstal Anaconda secara terpisah.

Setelah memilih Python environment, selanjutnya kita dapat mengetikkan dan menjalankan kode-kode Python kita, termasuk membuat Markdown, sebagaimana JupyterLab. Untuk menjalankan kode pada satu sel klik tombol **Execute Cell** (▶) atau tekan tombol **Ctrl + Alt + Enter**. Perhatikan contohnya pada gambar berikut:



Tanda centang (✓) di samping kiri sel kode Python menunjukkan bahwa secara teknis tidak ada masalah dengan kode, dan kode sukses dieksekusi. Jangan lupa untuk menyimpan file kode Python Anda, dengan klik menu **File → Save** atau tekan tombol **Ctrl + S**. Jika kita ingin mengakhiri bekerja menggunakan VS Code, Anda dapat secara langsung menutup file notebook atau jendela aplikasi tanpa ada prosedur Shut Down sebagaimana JupyterLab.

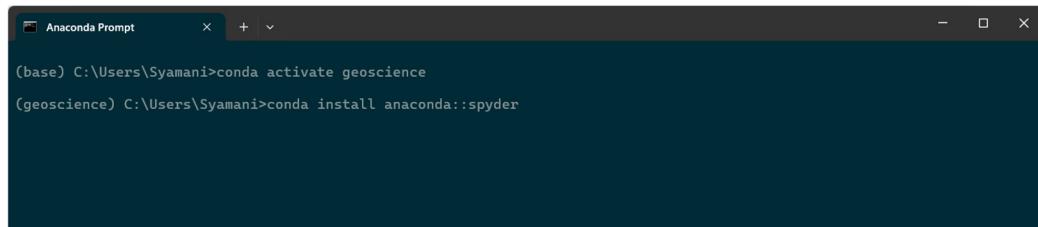
## Menggunakan Spyder

Spyder (*the Scientific Python Development Environment*) (<https://www.spyder-ide.org/>) (Raybaut, 2009) merupakan sebuah platform lingkungan pengembangan Python untuk *scientific and engineering*. Selain JupyterLab dan VS Code, Spyder juga dapat menjadi sebuah opsi lingkungan pemrosesan citra digital penginderaan jauh yang *powerfull*. Berdasarkan pengalaman penulis, pada beberapa kasus, ketika kita dihadapkan pada pemrosesan *big earth observation data*, eksekusi Python di dalam Spyder berjalan jauh lebih cepat dibandingkan dengan JupyterLab atau VS Code. Jadi jika nantinya Anda melakukan pemrosesan citra berbasis Python di komputer sendiri, dimana citra digitalnya sangat besar atau algoritma yang diimplementasikan sangat kompleks, Anda direkomendasikan menggunakan Spyder.

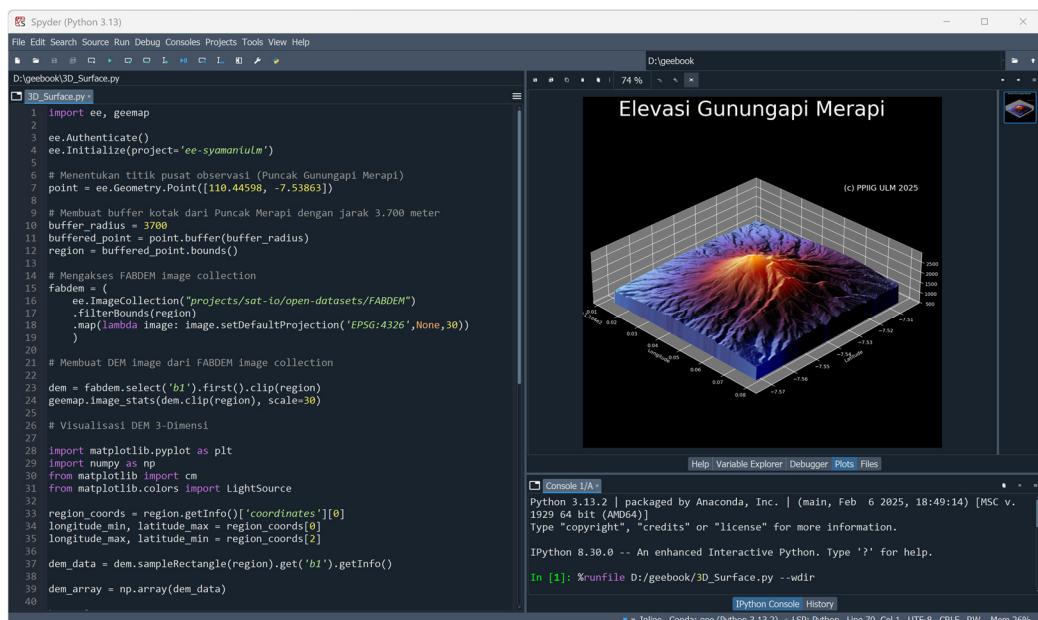
Untuk menginstal Spyder di lingkungan Anaconda, aktifkan environment Anda di dalam Anaconda Prompt, kemudian instal Spyder dengan perintah berikut:

```
conda install anaconda::spyder
```

Sebagaimana terlihat pada gambar berikut:



Setelah instalasi selesai, Spyder dapat dijalankan dengan perintah spyder di Anaconda Prompt. Berikut adalah tampilan jendela aplikasi Spyder.



Pada Spyder, kode Python disimpan langsung ke dalam format file Python (.py). Dimana pada aplikasi Spyder, kode program ditulis di jendela sebelah kiri. Di jendela bagian kanan terdapat sejumlah panel, di antaranya adalah panel *Console*, *Variabel Explorer*, *Plot*, dan sebagainya. Jika kita memproses citra di dalam Spyder, tampilan grafis (misalnya citra) akan tampil di panel *Plot*. Karena di dalam Spyder kode Python disimpan dalam format file .py, maka kode-kode Python di dalamnya harus dieksekusi secara keseluruhan sekaligus, tidak dapat dieksekusi per sel kode sebagaimana file notebook. Selain JupyterLab, Visual Studio Code, dan Spyder, ada beberapa platform aplikasi lain yang juga dapat digunakan untuk pemrograman Python, khususnya untuk pemrosesan citra digital penginderaan jauh. Misalnya PyCharm, NetBeans, PyDev, Atom, CodeLobster, Wing IDE, dan sebagainya. Jika Anda tertarik untuk menggunakan platform-platform ini, silahkan telusuri sendiri aplikasinya berikut petunjuk penggunaannya.

## E. Google Collaboratory

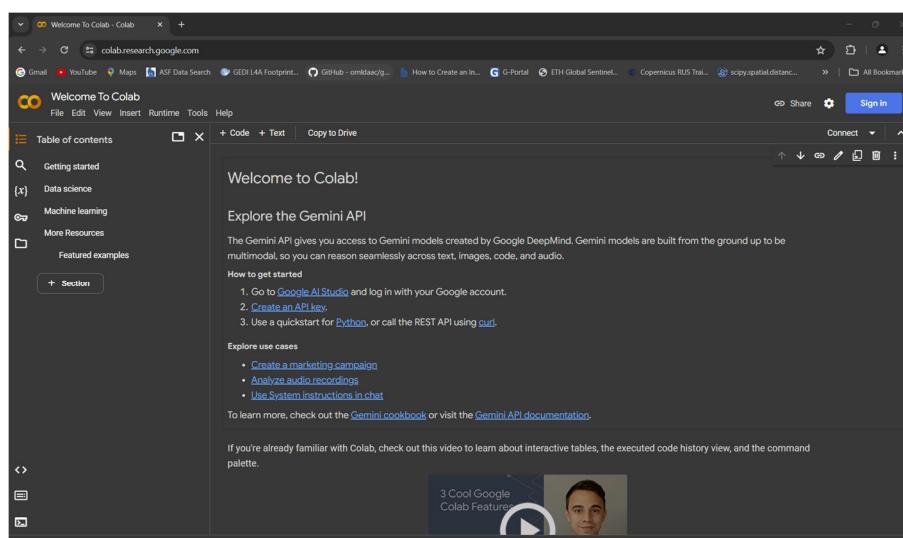
Google Collaboratory atau Google Colab adalah platform berbasis *cloud* untuk menulis, menjalankan, dan berbagi kode Python melalui *web browser*. Platform ini dirancang bagi analis, pengembang, peneliti, dan pendidik yang bekerja di bidang data sains dan machine learning dengan menyediakan environment komputasi yang fleksibel dan mudah diakses tanpa biaya. Google Colab juga menawarkan kemampuan untuk menjalankan Jupyter Notebook (*web app open-source* untuk kombinasi kode, teks terformat, dan visualisasi data) langsung dari web browser tanpa perlu konfigurasi apa pun (<https://revou.co/kosakata/google-colab>).

Google Colab menyediakan layanan gratis maupun berbayar. Yang paling menarik dari Google Colab adalah kita tidak memerlukan banyak instalasi paket-paket Python, sebab hampir semua paket yang biasa digunakan sudah tersedia. Termasuk integrasinya dengan GitHub. Terkecuali mungkin paket-paket Python khusus yang sifatnya sangat spesifik yang harus kita instal manual di Google Colab. Karena berbasis cloud, maka fasilitas utama yang harus dipenuhi untuk dapat menjalankan Google Colab adalah akses internet yang mumpuni selama kita bekerja.

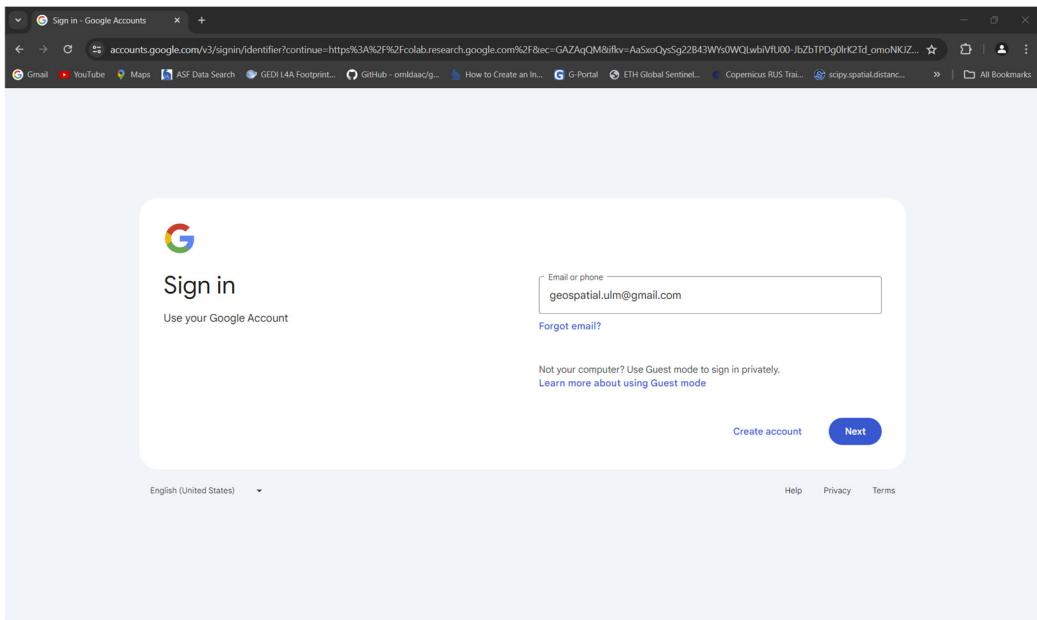
### Login ke Google Colab

Syarat utama untuk menggunakan Google Colab adalah kita harus sudah punya email di Gmail. Kita tidak bisa menggunakan email dengan domain yang lain, kecuali email domain khusus yang memang juga terintegrasi dengan sistem atau layanan Gmail. Jadi sebelum mengakses Google Colab, pastikan Anda sudah memiliki sebuah akun email di Gmail. Jika belum, silahkan buat sebuah email di laman <https://mail.google.com/>. Untuk menghindari tercampurnya transaksi data dengan email Gmail dan Google Drive kita yang sudah ada, tentu saja tidak ada salahnya bagi Anda yang sudah punya akun Gmail sebelumnya, untuk membuat akun Gmail lagi khusus untuk menggunakan Google Colab. Sebab ketika kita menggunakan Google Colab untuk pemrosesan citra via Google Earth Engine nantinya, kita berpotensi untuk menyimpan banyak file-file citra digital ke dalam Google Drive.

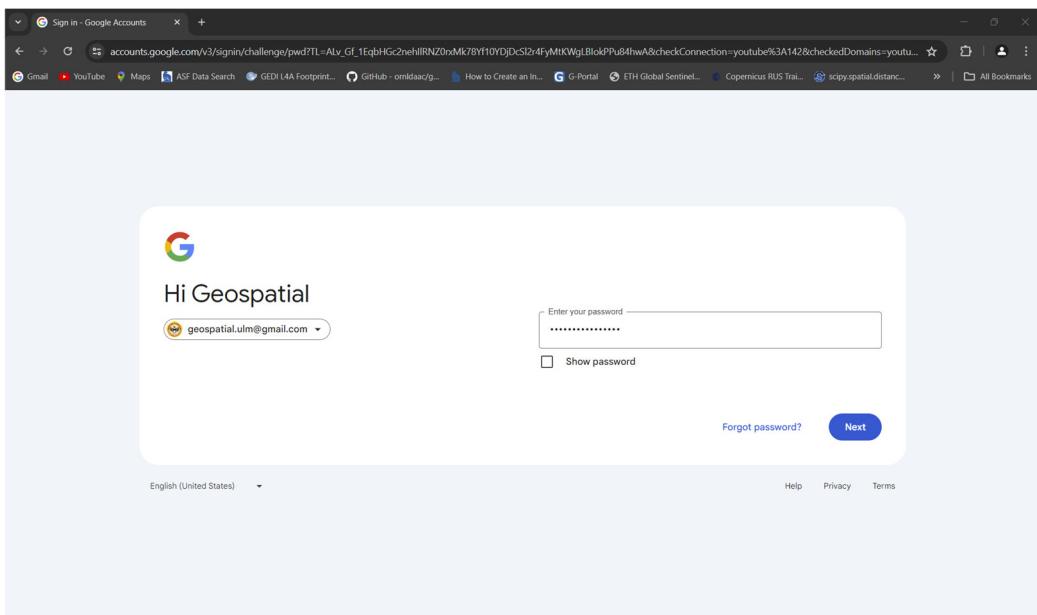
Untuk login ke Google Colab, buka laman <https://colab.research.google.com/> di web browser favorit Anda. Kemudian klik tombol **Sign in**, sebagaimana pada gambar di bawah.



Masukkan alamat email Gmail Anda, klik **Next**. Contohnya dapat dilihat pada gambar di bawah.

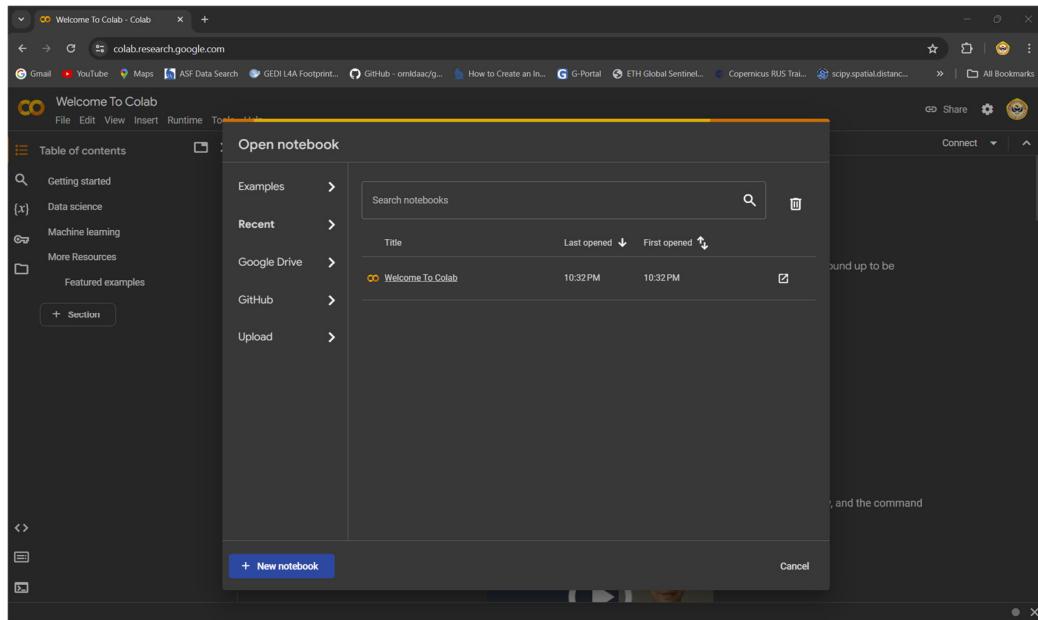


Tentu saja Anda juga harus memasukkan password email Anda, kemudian klik tombol **Next**. Sebagaimana gambar di bawah.



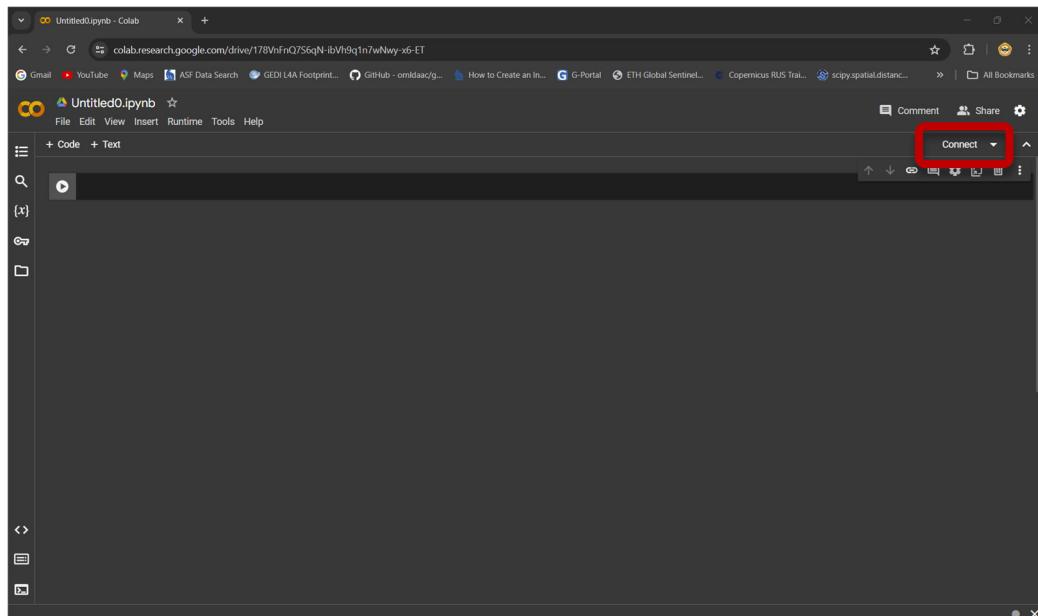
Selanjutnya, kita akan di bawa ke laman utama Google Colab. Sebagaimana ditunjukkan oleh gambar berikut.

## Bab I Pengantar

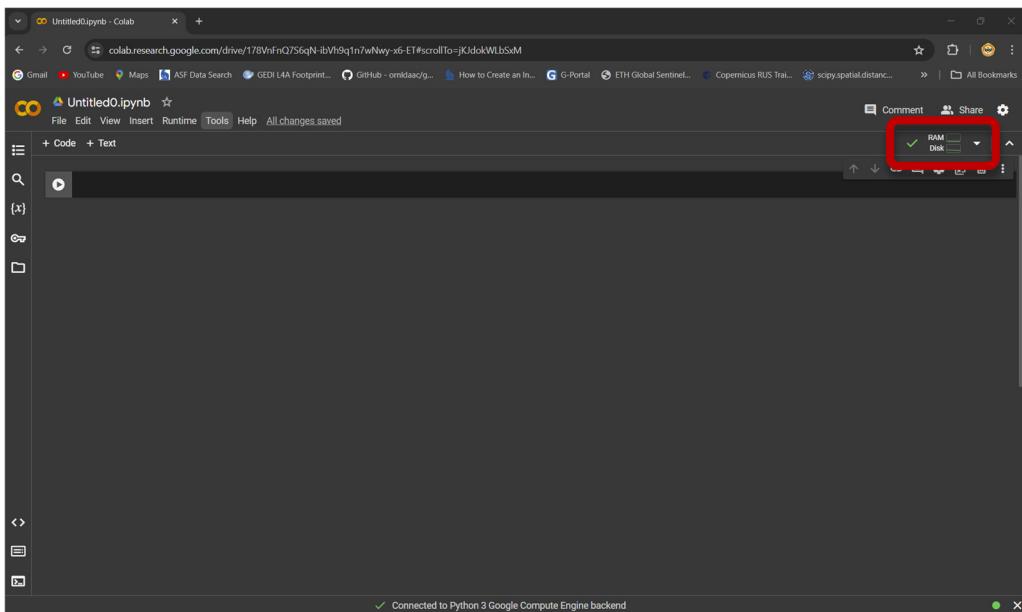


### Membuat Notebook Baru

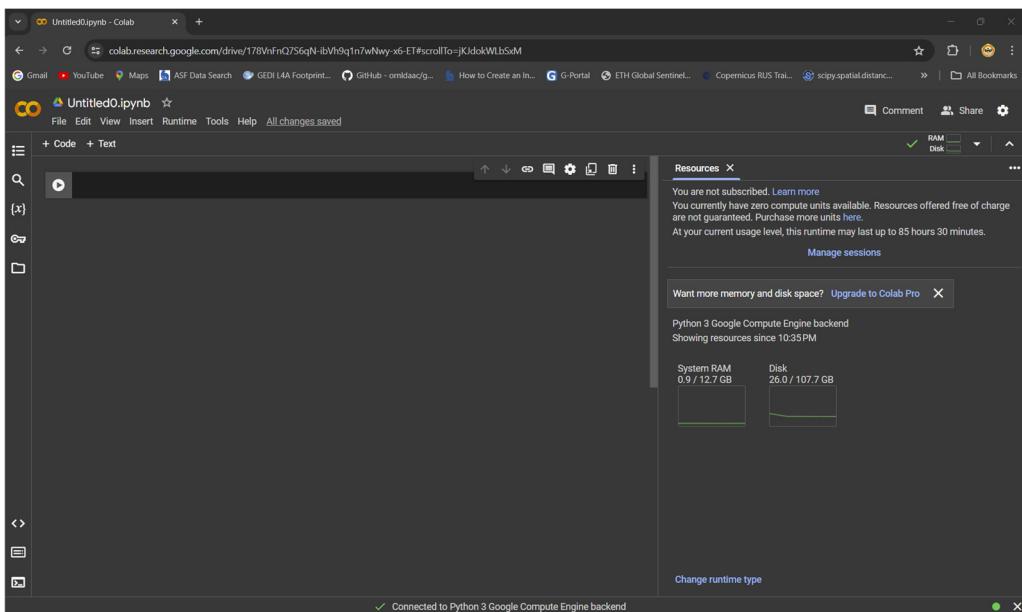
Untuk membuat sebuah notebook baru, klik tombol **New notebook** yang berwarna biru seperti pada gambar di atas. Atau klik menu **File → New notebook**. Sebuah notebook baru akan ditampilkan sebagaimana terlihat pada gambar di bawah.



Pada notebook baru di atas, klik tombol **Connect** di sudut kanan atas, agar kita terkoneksi dengan runtime Google Colab. Selanjutnya, tombol **Connect** akan berubah seperti gambar berikut.



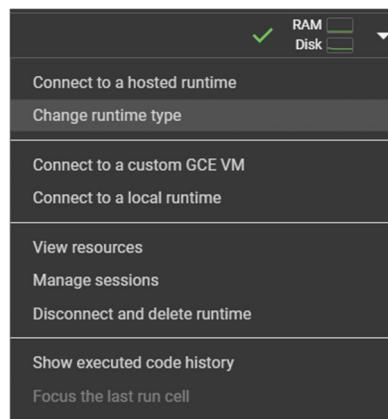
Klik tombol **RAM/Disk** (yang sebelumnya tombol Connect) pada notebook di atas untuk melihat sumberdaya (*resources*) yang dialokasikan oleh runtime Google Colab untuk kita bekerja. Yaitu memory (*RAM*) dan kapasitas ruang penyimpanan data (*Disk*).



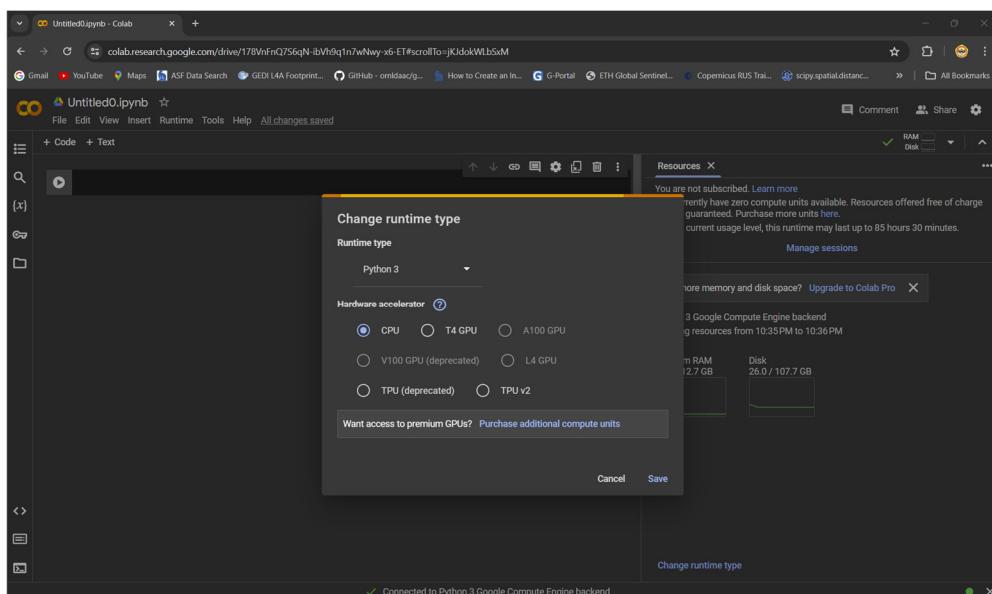
Untuk Google Colab versi gratisan, kita diberikan alokasi memory sekitar 12 GB dan ruang penyimpanan data sekitar 100 GB. Harap diingat, ruang 100 GB ini adalah ruang penyimpanan data di dalam runtime Google Colab, bukan ruang penyimpanan di dalam Google Drive kita. Kapasitas Google Drive kita yang gratisan tetaplah 15 GB. Jika kapasitas runtime Google Colab ini dirasakan kurang, tentu saja kita dapat menambahkannya dengan fasilitas berbayar. Dengan layanan Google Colab yang berbayar, kita akan mendapatkan RAM dan Disk yang lebih besar. Termasuk kapasitas Google Drive juga dapat diperbesar melalui layanan berbayar Google.

*Google Colab runtime adalah alokasi sumberdaya (CPU/GPU/TPU/RAM/Disk) berikut Python dan paket-paketnya yang akan terbentuk setiap kali kita membuat sebuah notebook baru. Sehingga setiap file notebook kita di Google Colab akan punya runtime-nya masing-masing. Kapasitas Disk di runtime berbeda dengan di Google Drive, sehingga tidak akan mempengaruhi ruang penyimpanan kita di Google Drive. Akan tetapi, setiap notebook yang kita buat, filenya akan tersimpan di dalam Google Drive.*

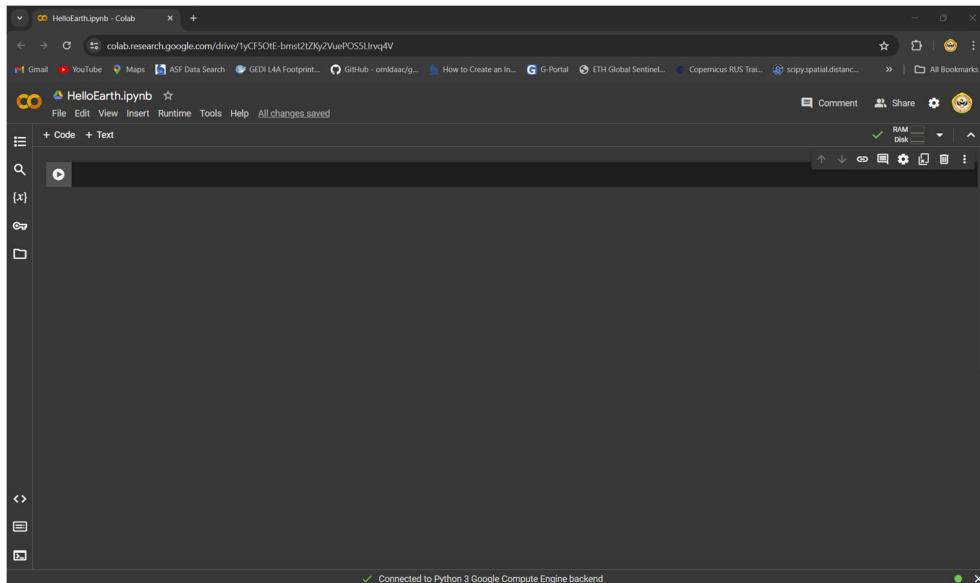
Kita dapat mengganti jenis runtime dengan mengklik tombol **RAM/Disk**, kemudian klik **Change runtime type**. Sebagaimana gambar berikut.



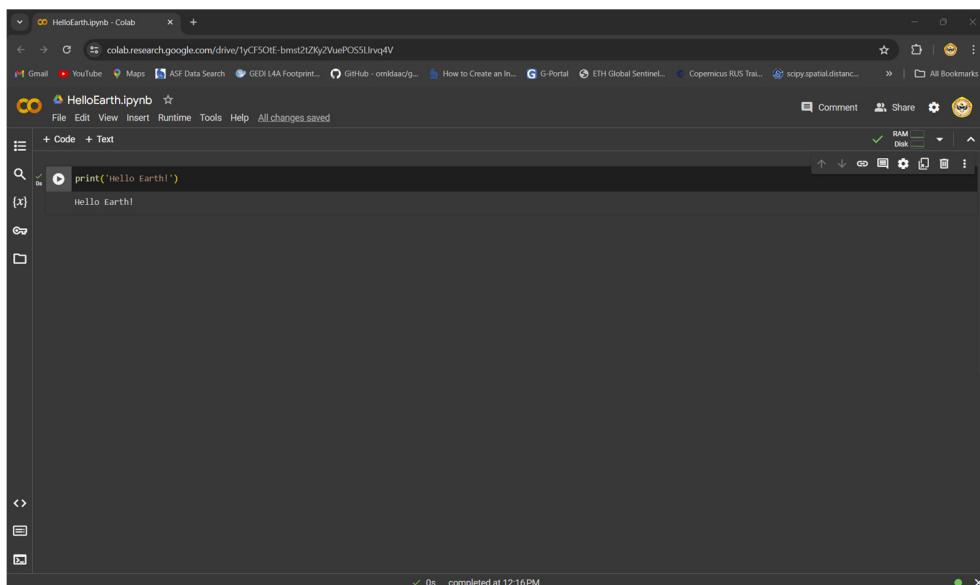
Selanjutnya akan tampil kotak dialog seperti pada gambar di bawah. Kita dapat memilih processing unit yang diinginkan, apakah CPU (*Central Processing Unit*), GPU (*Graphical Processing Unit*), atau TPU (*Tensor Processing Unit*).



Untuk saat ini, biarkan default CPU dulu. Sebab GPU atau TPU hanya efektif digunakan untuk aplikasi *deep learning* menggunakan framework TensorFlow atau PyTorch. CPU berarti processing unitnya mirip dengan laptop yang kita gunakan, seperti Intel Core, i3, i5, i7, atau i9. Laptop-laptop yang ada sekarang kebanyakan sudah *built-up* dengan GPU, seperti seri NVIDIA RTX.



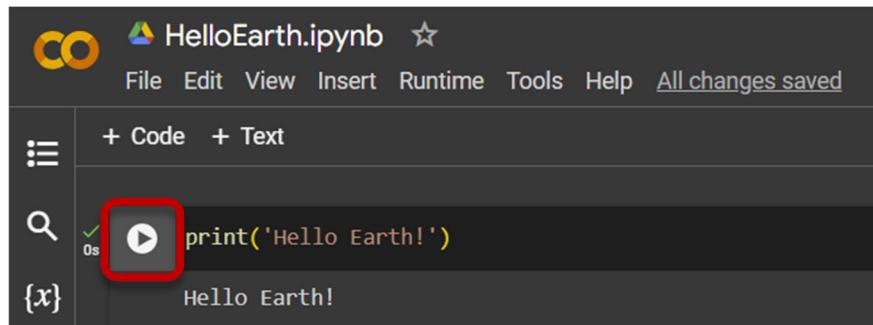
Klik menu **File → Rename** untuk merubah nama notebook. Kemudian ketik nama notebook yang diinginkan, misalnya **HelloEarth.ipynb**, sebagaimana gambar di atas.



Ketikkan kode Python berikut:

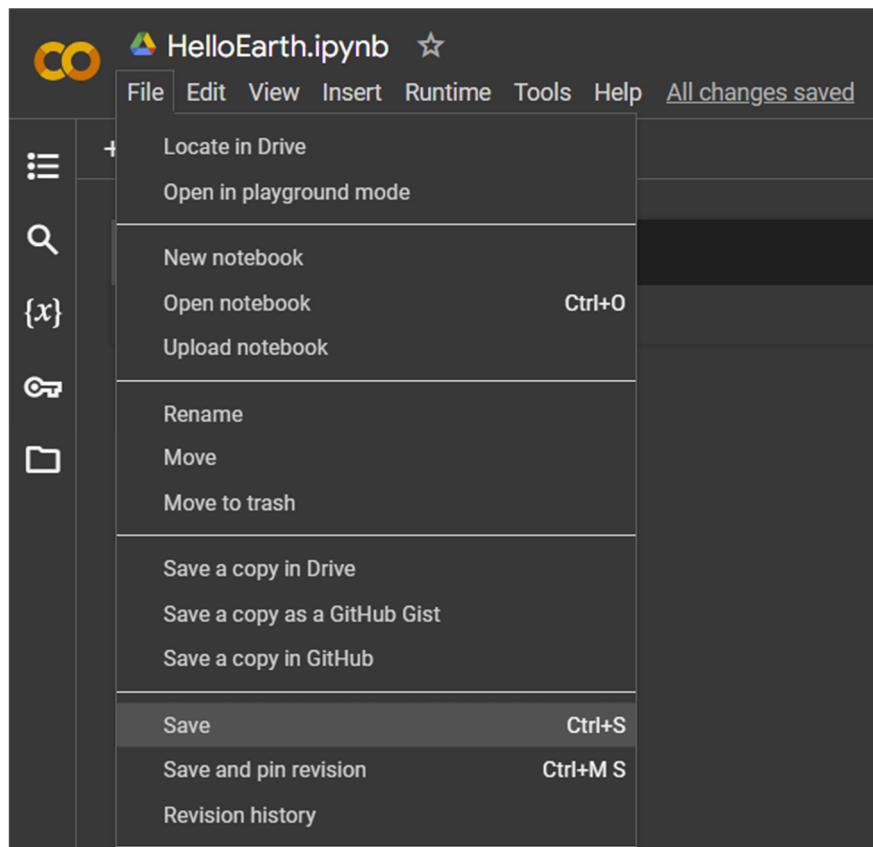
```
print('Hello Earth!')
```

Kemudian klik tombol **Run** (▶) di samping sel kode Python untuk menjalankan kode program. Jika tidak ada kesalahan sintaks, output kode program akan ditampilkan di bawah sel kode programnya, yaitu `Hello Earth!`.

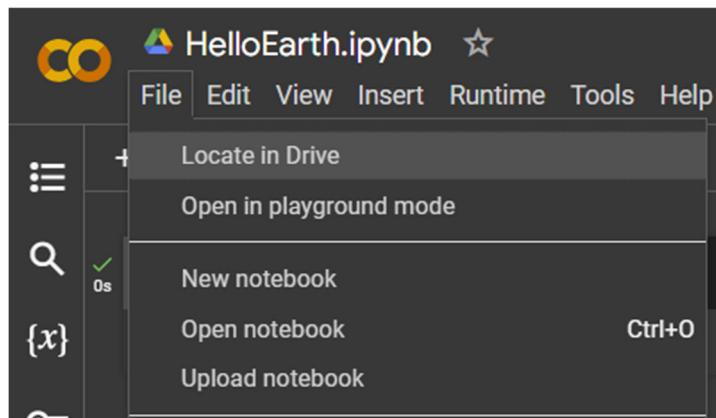


Untuk menambahkan sel kode program baru, klik tombol **+ Code**. Untuk menambahkan teks (markdown seperti di JupyterLab), klik tombol **+ Text**.

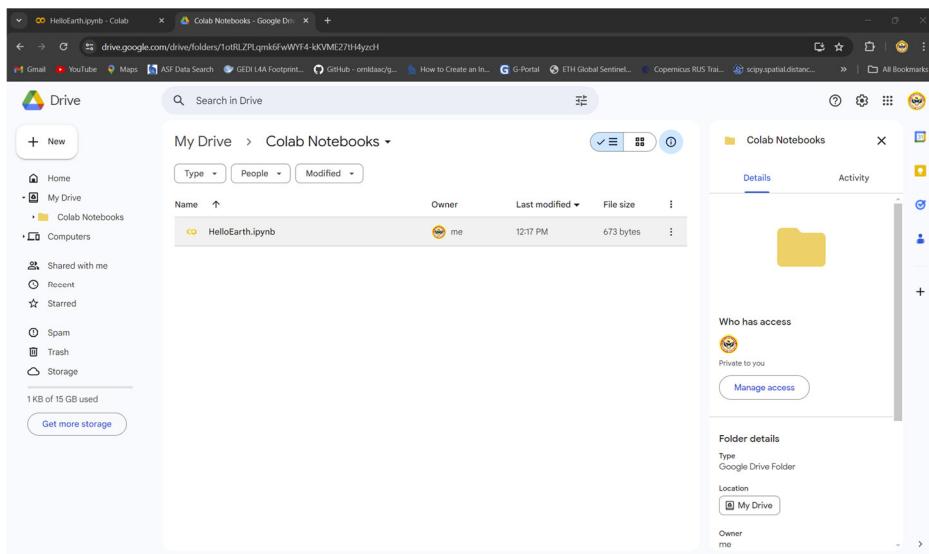
Klik menu **File → Save** untuk menyimpan notebook. Sebagaimana gambar berikut.



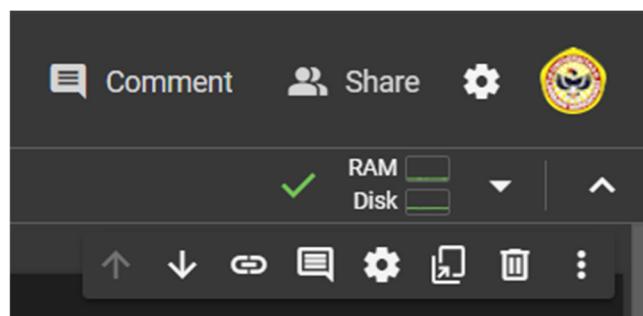
Untuk melihat lokasi penyimpanan file notebook kita di dalam Google Drive, klik menu **File → Locate in Drive**. Sebagaimana ditunjukkan pada gambar berikut.



Jendela browser baru akan terbuka dan akan menampilkan Google Drive kita. File notebook kita tersimpan di dalam Google Drive di folder **Colab Notebooks**. Perhatikan gambar berikut.

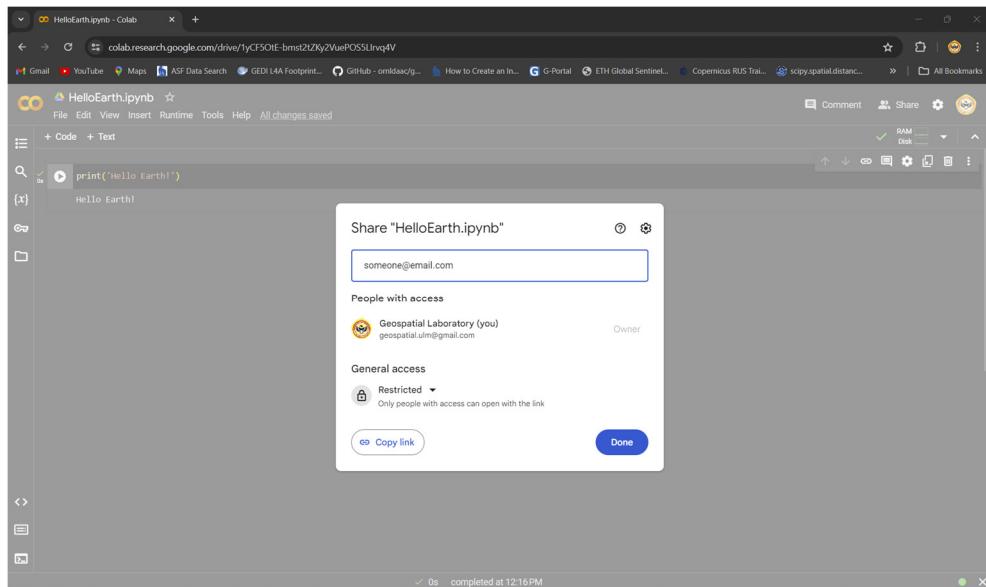


Tentu saja, kita dapat mendownload file notebook ke dalam laptop kita, untuk kemudian dijalankan di dalam Anaconda secara offline. Akan tetapi, perlu ada beberapa penyesuaian kode program. Kita juga dapat berbagi notebook kita ke orang lain yang juga menggunakan Google Colab. Caranya adalah klik tombol **Share**, sebagaimana pada gambar di bawah.

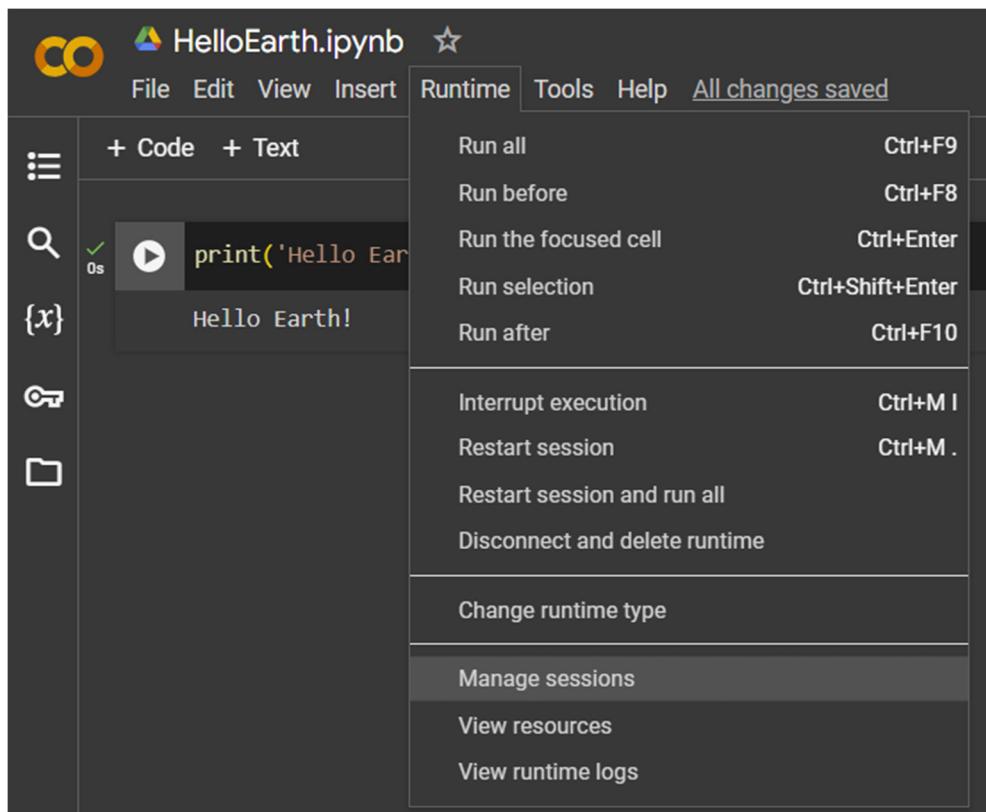


## Bab I Pengantar

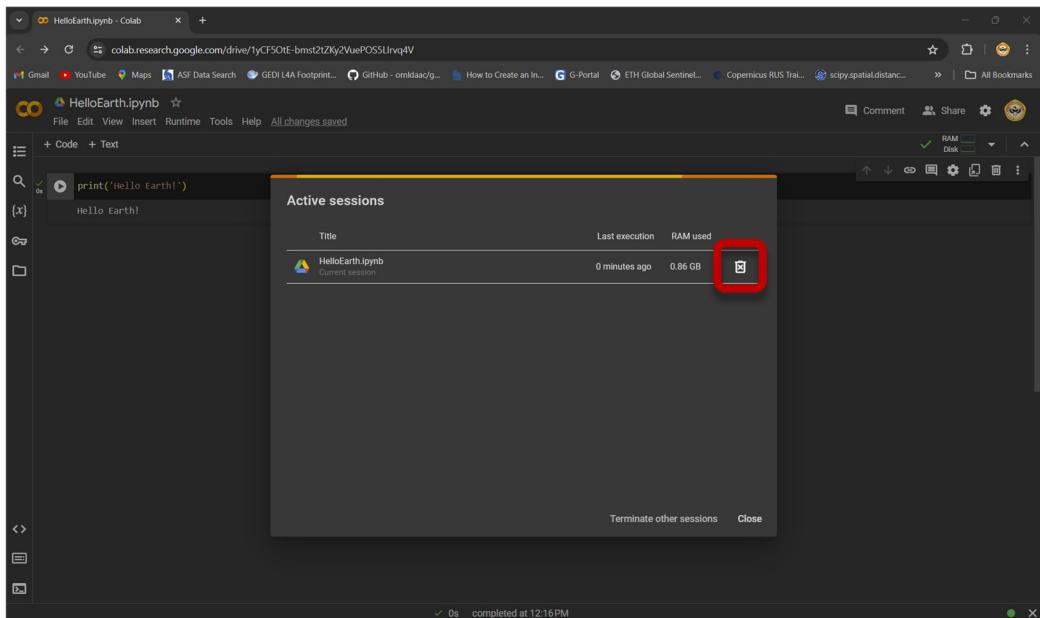
Kemudian pada kotak dialog share, masukkan alamat email kolega yang akan kita kirim salinan notebook kita, dan tekan tombol **Done**. Sebagaimana pada gambar berikut.



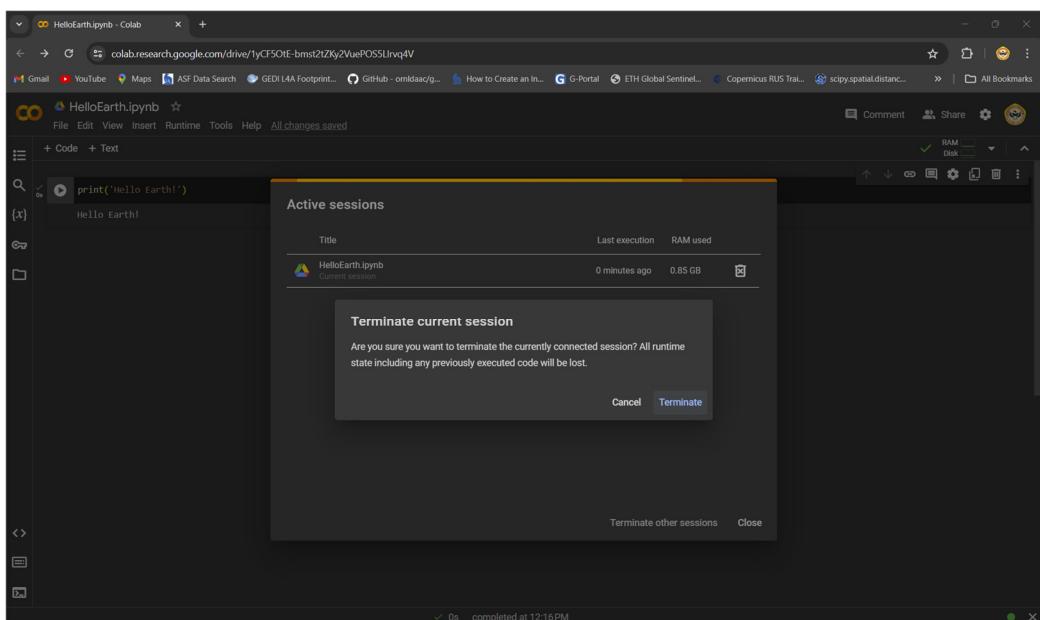
Jika kita sudah selesai bekerja dengan notebook kita, klik menu **Runtime → Manage sessions**. Sebagaimana pada gambar di bawah.



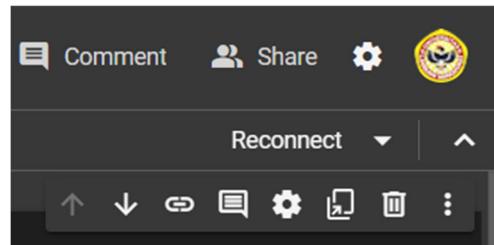
Selanjutnya, pada kotak dialog yang muncul, klik tombol **Terminate** (☒) pada notebook yang mau kita shut down. Sebagaimana terlihat pada gambar berikut.



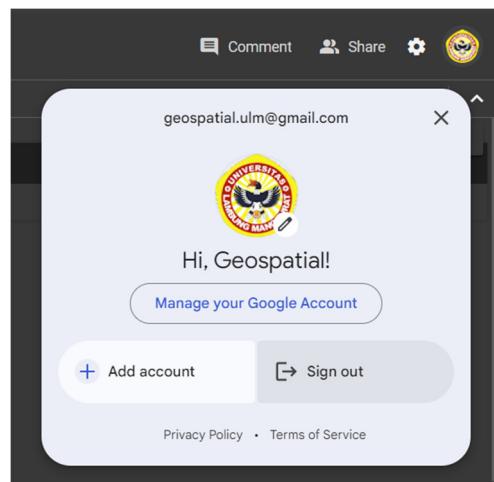
Jika muncul kotak dialog **Terminate current session** seperti pada gambar di bawah, klik tombol **Terminate**, jika kita memang ingin mengakhiri pekerjaan kita di notebook.



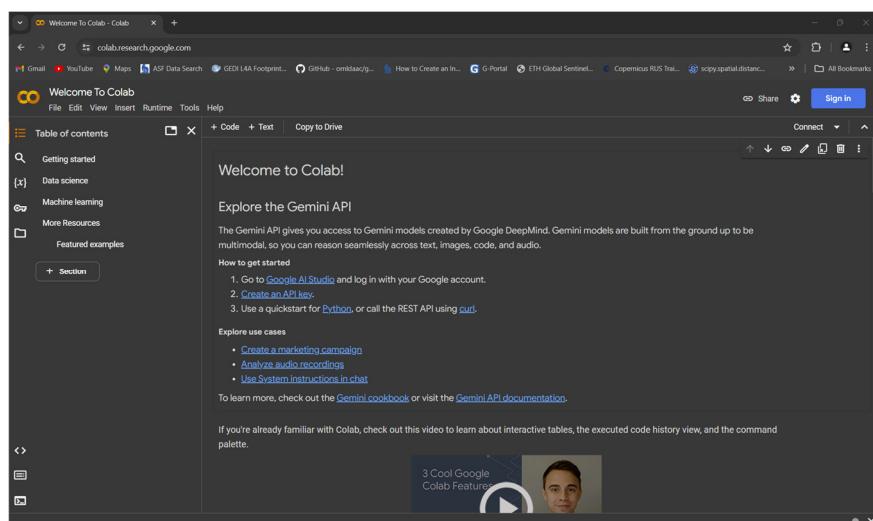
Setelah mengklik tombol **Terminate**, notebook akan terputus dari runtime-nya. Jika ternyata kita masih ingin bekerja dengan notebook tersebut kembali, klik tombol **Reconnect**. Sebagaimana terlihat pada gambar berikut.



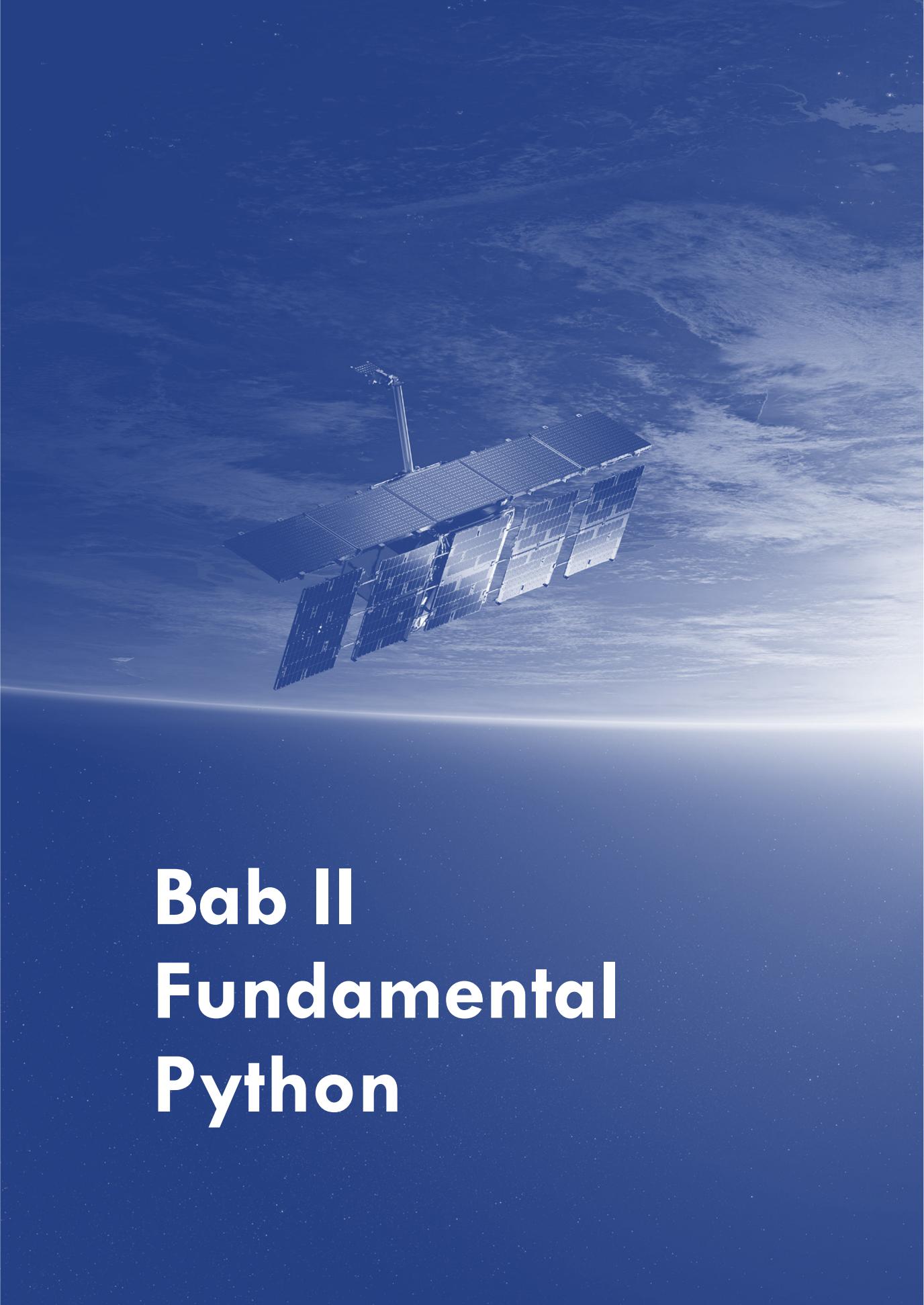
Jika Anda sudah selesai bekerja dengan Google Colab, Anda dapat **Sign out** sebagaimana kita logout dari email kita di Gmail. Sebagaimana terlihat pada gambar berikut.



Dan kita akan dibawa ke halaman Google Colab tanpa login sebagaimana gambar berikut.



Sampai di sini pengantar awal Google Colab sudah selesai. Khusus untuk materi Dasar-dasar Bahasa Python berikutnya, Anda bebas untuk menggunakan JupyterLab atau VS Code (offline), atau Google Colab (online). Terkecuali nanti jika diinstruksikan secara khusus untuk menggunakan JupyterLab/VS Code atau Google Colab.



# **Bab II**

# **Fundamental**

# **Python**



## A. Mengenal Bahasa Python

Jika Anda mendengar nama Python, mungkin yang terbayang di benak Anda adalah nama sejenis hewan melata, yang terkenal berbadan besar dan mampu menelan mangsanya bulat-bulat. Faktanya tidak seperti itu, nama Bahasa Pemrograman Python tidak ada hubungannya dengan hewan jenis apapun. Nama Python yang terdengar sedikit menakutkan tersebut juga tidak menggambarkan bagaimana sulitnya mempelajari Bahasa Pemrograman Python. Justru di antara semua bahasa pemrograman, Python merupakan salah satu bahasa pemrograman yang paling mudah untuk dipelajari. Itu sebabnya, sebagian besar programer pemula, pada umumnya akan memilih Python sebagai bahasa pemrograman pertamanya. Jadi, bagi Anda yang baru pertama kali ini mengenal bahasa pemrograman, tidak perlu takut dengan yang namanya “Python”.

Bahasa Pemrograman Python sebenarnya sudah mulai dikonsep oleh Guido van Rossum pada akhir tahun 1980. Meskipun implementasinya baru terlaksana mulai Desember 1989. Pada tahun 1990, Bahasa Python dirilis secara internal di *Centrum voor Wiskunde en Informatica* (CWI) atau Institut Riset Matematika dan Sains Komputer Nasional, Netherlands. Dan pada tanggal 20 Februari 1991, barulah Python versi 0.9.0 dirilis secara resmi ke publik. Tanggal inilah yang umumnya dikenal sebagai tanggal rilis resminya Bahasa Pemrograman Python. Nama Python diambil dari nama sebuah acara komedi televisi, yaitu *Monty Python's Flying Circus*, yang ditayangkan BBC One pada tahun 1969 sampai tahun 1974. Monty Python sendiri merupakan nama grup lawak berkebangsaan Inggris yang bermain dalam acara komedi televisi tersebut (<https://pythoninstitute.org>). Awalnya, Python didesain untuk sistem operasi kecil yang bernama Amoeba. Ketika sistem operasi GNU/Linux lahir pada tahun yang bersamaan dengan rilis resmi Python, Python banyak berjasa dalam menangani tugas-tugas administrator Linux.

Tabel 2.1. Bahasa pemrograman terpopuler menurut TIOBE Index untuk Agustus 2025

Aug 2025	Aug 2024	Change	Programming Language	Ratings	Change
1	1		Python	26.14%	+8.10%
2	2		C++	9.18%	-0.86%
3	3		C	9.03%	-0.15%
4	4		Java	8.59%	-0.58%
5	5		C#	5.52%	-0.87%
6	6		JavaScript	3.15%	-0.76%
7	8	▲	Visual Basic	2.33%	+0.15%
8	9	▲	Go	2.11%	+0.08%
9	25	▲	Perl	2.08%	+1.17%
10	12	▲	Delphi/Object Pascal	1.82%	+0.19%
11	10	▼	Fortran	1.75%	-0.03%
12	7	▼	SQL	1.72%	-0.49%
13	30	▲	Ada	1.52%	+0.91%
14	19	▲	R	1.37%	+0.26%
15	13	▼	PHP	1.27%	-0.19%
16	11	▼	MATLAB	1.19%	-0.53%
17	20	▲	Scratch	1.15%	+0.06%
18	14	▼	Rust	1.13%	-0.15%
19	18	▼	Kotlin	1.10%	-0.04%
20	17	▼	Assembly language	1.03%	-0.19%

Sumber: <https://www.tiobe.com/tiobe-index>

## Bab II Fundamental Python

---

Python merupakan bahasa pemrograman terpopuler saat ini. Bahkan sejak Agustus 2024 hingga Agustus 2025, Python menempati peringkat nomor 1 di dalam daftar bahasa-bahasa pemrograman terpopuler. Sebagaimana terlihat pada Tabel 2.1. Popularitas Python tidak terlepas dari kesederhanaan bahasanya, sehingga cukup memudahkan penggunaannya. Kelebihan Python dibandingkan bahasa pemrograman lainnya adalah sifatnya yang dinamis dan efisien. Bersifat dinamis, sebab di samping Python dapat berfungsi sebagai bahasa pemrograman murni seperti C/C++/C# yang memerlukan proses kompilasi sebelum dijalankan, Python juga dapat berfungsi sebagai *shell* atau *Command Line Interface (CLI)*, yaitu kode program yang hanya diketikkan satu atau beberapa baris, hanya dengan menekan tombol *Enter* kode langsung dieksekusi, tanpa melalui proses kompilasi terlebih dahulu.

Python juga dikenal efisien dalam penulisan kode program, salah satu penyebabnya adalah blok-blok program ditentukan oleh identasi. Bukan tanda kurung kurawal sebagaimana C/C++/C#, dan bukan kata-kata sebagaimana Bahasa Pascal atau BASIC. Dan tidak ada tanda atau notasi apa-apa (seperti titik koma *C-style*) di belakang baris pernyataan kode Python. Kelebihan Python lainnya adalah kita tidak perlu dipusingkan dengan urusan penanganan memori komputer. Sebab di dalam Python tidak ada pendeklarasian tipe variabel, semuanya sudah ditangani oleh interpreter. Berbeda dengan bahasa pemrograman statik seperti C/C++/C#, dimana kita sedikit banyaknya dituntut untuk memahami konsep memori komputer, khususnya untuk kepentingan pendeklarasian variabel dan manajemen memori menggunakan *pointer* atau *reference*.

Lebih jauh, dengan semakin berkembangnya era kecerdasan buatan (*Artificial Intelligence*) akhir-akhir ini, Python seakan-akan menjadi bahasa pemrograman utama di dalam menangani aplikasi-aplikasi kecerdasan buatan (Teoh and Rong, 2022). Berbagai *platform* dan *framework* populer *data science*, *machine learning*, dan *deep learning*, seperti Scikit-Learn, PyTorch, TensorFlow, dan Keras, dirancang untuk Bahasa Python. Di dunia geospasial sendiri, perangkat lunak-perangkat lunak Sistem Informasi Geografis (SIG) seperti ArcGIS Desktop, ArcGIS Pro, QGIS, dan sebagainya, juga menggunakan Python untuk otomasi proses-proses analisis geospasial. Sehingga secara singkat, kemampuan menggunakan Bahasa Pemrograman Python akan menjadi satu keuntungan bagi karir di bidang informasi geospasial dan kecerdasan buatan.

Upendra Patel (*Director and Founder at TriState Technology*) memberikan 10 (sepuluh) alasan mengapa Python adalah bahasa pemrograman terbaik untuk kecerdasan buatan, yaitu (<https://www.tristatetechnology.com/>):

1. *Simple and consistent* (sederhana dan konsisten);
2. *Better library ecosystem* (ekosistem pustaka yang lebih baik);
3. *Flexible* (fleksibel);
4. *Popular* (populer);
5. *Better visualization option* (opsi visualisasi yang lebih baik);
6. *Readability* (keterbacaan);
7. *Platform independence* (independensi platform);
8. *Rapid development* (perkembangan yang pesat);
9. *Less coding* (lebih sedikit pengkodean);
10. *Speed of execution* (kecepatan eksekusi).

## B. Sintaks Dasar Bahasa Python

### Variabel

Di dalam matematika, variabel adalah suatu notasi yang dapat menyimpan nilai, dan nilainya dapat berubah-ubah. Definisi variabel yang sama juga berlaku di dalam bahasa pemrograman, termasuk Python. Di dalam Python, suatu variabel akan digunakan untuk menyimpan suatu nilai atau data. Faktanya, setiap data di dalam Python akan disimpan di dalam variabel. Pendeklarasian variabel di dalam Python sangat simpel, tidak serumit bahasa lain seperti C++ yang harus mendefinisikan tipe variabel secara eksplisit. Deklarasi variabel di dalam Python cukup dengan menuliskan nama variabel dan langsung mengisinya dengan nilai, misalnya:

```
earth_radius = 6371
pjg_jalan = 32.5
bil_kompleks = 5 + 3j
profesi = 'Remote Sensing Expert'
opsi = True
var_kosong = None
```

Deklarasi variabel dengan penugasan nilai seperti di atas dikenal juga sebagai inisialisasi variabel. Variabel di dalam Python dapat bertipe numerik, teks (*string*), *boolean*, atau *None*. Variabel numerik dapat berupa bilangan bulat (*integer*), bilangan desimal (*float*), atau bilangan kompleks (*complex*). Pada contoh di atas, `earth_radius` merupakan contoh variabel numerik (*integer*), `pjg_jalan` merupakan contoh variabel numerik (*float*), `bil_kompleks` merupakan contoh variabel numerik (*complex*), `profesi` merupakan contoh variabel string (teks), `opsi` adalah contoh variable boolean yang hanya dapat berisi nilai `True` atau `False`, dan `var_kosong` adalah sebuah variabel bertipe `None` atau kosong.

Perhatikan bahwa variabel string wajib diketik diantara tanda '`'Ini string'` (tanda petik tunggal), atau '`"Ini string"` (tanda petik ganda). Baik tanda petik tunggal maupun tanda petik ganda dapat digunakan secara bergantian, sebab dalam konteks tertentu artinya akan sama. Akan tetapi, di dalam konteks teks yang memang sudah terdapat tanda petik tunggal di dalamnya, maka notasi string wajib menggunakan tanda petik ganda. Sebab jika tidak, nanti akan terjadi error. Contohnya pada kata "`Syam'ani`".

Untuk menampilkan isi variabel dan mengetahui tipe sebuah variabel, kita dapat mengetikkan perintah berikut:

```
earth_radius = 6371
print(earth_radius)
print(type(earth_radius))
```

Output:

```
6371
<class 'int'>
```

Kode seperti `print(earth_radius)` akan mencetak isi dari variabel `earth_radius` ke layar. Sementara kode seperti `print(type(earth_radius))` akan mencetak tipe dari variabel `earth_radius` ke layar.

## Bab II Fundamental Python

Kita juga dapat merubah tipe variabel, dimana di dalam Python hal ini dikenal sebagai *casting*. Ada *implicit casting* atau dikenal juga sebagai *automatic casting*, dan ada *explicit casting* atau dikenal juga sebagai *manual casting*.

Contoh implicit casting:

```
x = 5
y = 2.5
result = x + y
print(result)
```

Output:

```
7.5
```

Kode di atas adalah contoh *implicit casting*. Sebab x tipenya integer, y tipenya float. Ketika dalam proses kalkulasi, x yang integer akan dikonversi oleh Python secara otomatis menjadi float. Berikut adalah contoh *explicit casting*:

```
x = 5
y = 2.5
result = int(x) + int(y)
print(result)
```

Output:

```
7
```

Kode di atas adalah contoh *explicit casting*. Sebab variabel x dan y dikonversi secara eksplisit dengan kode `int(x)`. Hati-hati di dalam melakukan *explicit casting*, sebab langkah ini dapat merubah informasi di dalam variabel secara signifikan. Contoh pada kasus di atas, variabel y sebenarnya diisi dengan `2.5`, akan tetapi setelah dicasting menjadi integer, nilainya menjadi `2`.

## Pengabungan String

String dapat digabung, contoh:

```
first_name = 'Ethan'
last_name = 'Hunt'
full_name = first_name + ' ' + last_name
print(full_name)
```

Output:

```
Ethan Hunt
```

Perhatikan bahwa string `' '` ada spasi di tengahnya. Di dalam kasus tertentu, entri ini harus ditambahkan. Sebab jika tidak, outputnya akan seperti ini:

```
EthanHunt
```

Cara lain menggabungkan string:

```
luas_hutan = 250
string = 'Luas hutan adalah {} hektare.'.format(luas_hutan)
print(string)
```

Output:

```
Luas hutan adalah 250 hektare.
```

Dengan instruksi di atas, maka tanda `{}` di dalam string akan diisi dengan variabel `luas_hutan`. Teknik seperti ini biasanya digunakan untuk mempermudah penggabungan string dan variabel. Alternatif lain, instruksi `'Luas hutan adalah {} hektare.'.format(luas_hutan)` dapat juga ditulis ke dalam format *f-string* berikut:

```
f'Luas hutan adalah {luas_hutan} hektare.'
```

Dimana kedua format penulisan kode di atas akan memberikan hasil yang sama persis.

### Penamaan Variabel

Terkait penamaan variabel, tentu saja direkomendasikan nama variabel harus informatif. Untuk menyimpan data luas wilayah misalnya, buatlah nama variabel seperti `luas_wilayah` atau `region_area`. Jangan pakai nama-nama yang aneh-aneh dan tidak informatif, misalnya `wxyz` atau `abcd`. Sebab kalau nama variabel tidak informatif, nanti kita sebagai programer yang akan bingung sendiri. Dan tentu saja, nama variabel tidak boleh konflik atau sama persis satu sama lain di dalam satu file kode program (misalnya satu notebook). Serta tidak boleh juga sama dengan nama identitas lainnya, seperti nama kelas atau nama fungsi.

Menurut <https://www.w3schools.com/>, berikut adalah konvensi penamaan variabel di dalam Bahasa Pemrograman Python:

- Nama variabel harus diawali dengan huruf atau karakter garis bawah
- Nama variabel tidak boleh diawali dengan angka
- Nama variabel hanya boleh berisi karakter alfanumerik dan garis bawah (A-z, 0-9, dan `_`)
- Nama variabel sensitif huruf besar-kecil (`citra`, `Citra`, dan `CITRA` adalah tiga variabel berbeda)
- Nama variabel tidak boleh berupa kata kunci Python apa pun.

Pada umumnya, variabel di dalam bahasa pemrograman diberi nama huruf kecil semua. Jika nama variabel terdiri atas 2 atau lebih, bisa menggunakan garis bawah sebagai penghubung. Gaya penulisan dengan huruf kecil semua dan pemisah garis bawah antar kata seperti ini dikenal sebagai *snake case*. Untuk Bahasa Python, Hunt (2023) merekomendasikan penamaan variabel *snake case* seperti ini. Contohnya, `luas_area`, `koordinat_titik`, `elevasi_minimum`, dan sebagainya. Terkadang penamaan variabel juga dapat menggunakan *camel case*, dimana huruf pertama dari kata pertama menggunakan huruf kecil, sementara huruf pertama pada kata berikutnya menggunakan huruf kapital. Contohnya, `panjangGaris`, `diameterBumi`, `geoInfo`, dan sebagainya. Hanya saja, untuk Python agak jarang menggunakan *camel case*. Penamaan variabel dengan gaya *camel case* sering digunakan pada keluarga bahasa C/C++/C#.

### Nilai Boolean

Boolean merupakan ekspresi benar (*True*) atau salah (*False*). Sehingga, tipe boolean dapat berupa sebuah variabel yang berisi ekspresi *True* atau *False*. Perhatikan contoh berikut:

```
x = 5  
y = 8  
  
z = x > y  
  
print(z)
```

Output:

```
False
```

Pada contoh kode di atas, variabel *x* diisi dengan nilai *5*, dan variabel *y* diisi dengan nilai *8*. Kemudian variabel *z* diisi dengan operasi perbandingan *x > y*. Tentu saja, ekspresi *x > y* atau dengan kata lain *5 > 8* ini adalah salah secara matematis. Sehingga ekspresi *x > y* memberikan nilai *False* ke variabel *z*. Seandainya ekspresinya dirubah menjadi *x < y*, maka outputnya tentu saja akan *True*.

Pada praktiknya, nilai boolean nanti akan sangat berperan di dalam ekspresi logika seperti *if*, perulangan instruksi atau liukan (*loop*) seperti *for* dan *while*, atau pengambilan keputusan (*decision*). Sebab ekspresi logika seperti *if* baru akan mengeksekusi instruksi di bawahnya jika ekspresi yang diberikan bernilai *True*. Hal yang sama juga berlaku untuk perulangan instruksi seperti *for* dan *while*.

### Operator Python

Di dalam setiap pemrograman, dapat dipastikan bahwa kita akan melakukan berbagai proses kalkulasi, perbandingan, ekspresi logika, penugasan nilai, dan sebagainya. Dan untuk melakukan proses-proses ini tentu saja diperlukan operator. Operator dapat berupa simbol-simbol tertentu, seperti simbol-simbol matematika. Bahkan operator juga dapat berupa kata-kata yang kita kenal sehari-hari, seperti and, or, not, dan sebagainya.

Berikut adalah operator-operator yang digunakan oleh Bahasa Pemrograman Python menurut [https://www.w3schools.com/python/python\\_operators.asp](https://www.w3schools.com/python/python_operators.asp):

Tabel 2.2. Operator aritmatika

Operator	Nama	Contoh
+	Addition (Penjumlahan)	$x + y$
-	Subtraction (Pengurangan)	$x - y$
*	Multiplication (Perkalian)	$x * y$
/	Division (Pembagian)	$x / y$
%	Modulus (Sisa pembagian)	$x \% y$
**	Exponentiation (Perpangkatan)	$x ** y$
//	Floor division (Pembagian dengan pembulatan ke bawah)	$x // y$

Tabel 2.3. Operator penugasan

Operator	Contoh	Sama Dengan
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
=	x  = 3	x = x   3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3
:	print(x := 3)	x = 3 print(x)

Tabel 2.4. Operator perbandingan

Operator	Nama	Contoh
==	Equal (Sama dengan)	x == y
!=	Not equal (Tidak sama dengan)	x != y
>	Greater than (Lebih besar dari)	x > y
<	Less than (Lebih kecil dari)	x < y
>=	Greater than or equal to (Lebih besar dari atau sama dengan)	x >= y
<=	Less than or equal to (Lebih kecil dari atau sama dengan)	x <= y

Tabel 2.5. Operator logika

Operator	Deskripsi	Contoh
and	BerNilai benar jika kedua pernyataan benar	x < 5 and x < 10
or	BerNilai benar jika salah satu pernyataan benar	x < 5 or x < 4
not	BerNilai salah jika pernyataan benar, atau sebaliknya	not(x < 5 and x < 10)

Tabel 2.6. Operator identitas

Operator	Deskripsi	Contoh
is	BerNilai benar jika kedua variabel adalah objek yang sama	x is y
is not	BerNilai benar jika kedua variabel adalah objek yang tidak sama	x is not y

Tabel 2.7. Operator keanggotaan

Operator	Deskripsi	Contoh
in	BerNilai benar jika sebuah urutan dengan nilai spesifik terdapat di dalam objek	x in y
not in	BerNilai benar jika sebuah urutan dengan nilai spesifik tidak terdapat di dalam objek	x not in y

Tabel 2.8. Operator bitwise

Operator	Nama	Deskripsi	Contoh
&	AND	Setel setiap bit ke 1 jika kedua bit adalah 1	x & y
	OR	Setel setiap bit ke 1 jika salah satu dari dua bit adalah 1	x   y
^	XOR	Setel setiap bit ke 1 jika hanya satu dari dua bit bernilai 1	x ^ y
~	NOT	Membalikkan semua bit	~x
<<	Zero fill left shift	Geser ke kiri dengan menekan angka nol dari kanan dan biarkan bagian paling kiri terlepas	x << 2
>>	Signed right shift	Geser ke kanan dengan mendorong salinan bit paling kiri dari kiri, dan biarkan bit paling kanan terlepas	x >> 2

### Prioritas Operator

Prioritas operator (*operator precedence*) menentukan operasi apa yang terlebih dahulu dieksekusi, ketika kode program melibatkan operasi-operasi yang kompleks. Misalnya dalam sebuah persamaan matematika yang cukup panjang, terdapat perkalian, pembagian, penjumlahan, pengurangan, pengkuadratan, dan sebagainya.

Tabel 2.9. Prioritas operator Python

Operator	Deskripsi
()	Tanda kurung
**	Perpangkatan
+x -x ~x	Unary plus, unary minus, dan bitwise NOT
* / // %	Perkalian, pembagian, pembagian dengan pembulatan ke bawah, dan sisa hasil pembagian
+ -	Penjumlahan dan pengurangan
<< >>	Bitwise geser ke kiri dan ke kanan
&	Bitwise AND
^	Bitwise XOR
	Bitwise OR
== != > >= < <= is is not in not in	Perbandingan, identitas, and operator keanggotaan
not	Logika NOT
and	Logika AND
or	Logika OR

Sumber: <https://www.w3schools.com>

### Komentar

Di dalam setiap bahasa pemrograman, termasuk Bahasa Python, komentar adalah kalimat-kalimat yang disisipkan ke dalam kode program. Akan tetapi, oleh interpreter atau kompiler, komentar tidak akan dianggap sebagai bagian dari kode program, dan akan diabaikan oleh interpreter atau kompiler ketika kode program tersebut dijalankan. Dengan kata lain, komentar tidak akan berpengaruh terhadap luaran kode program atau perangkat lunak yang dibangun. Akan tetapi, komentar akan berpengaruh kepada kita sebagai programer.

Komentar merupakan catatan-catatan tertentu yang dibuat oleh programer di dalam kode program sebagai penanda atau pengingat. Misalnya untuk mengingatkan tentang fungsi dari suatu blok kode program. Tanpa adanya komentar, jika kode program yang dibangun cukup besar, maka programer biasanya akan kesulitan untuk mengingat fungsi masing-masing bagian kode program. Apalagi jika di dalam sebuah industri perangkat lunak atau proyek kolaborasi, kode program dibangun secara bersama-sama oleh banyak programer di dalam satu tim. Komentar akan sangat berguna untuk memberi catatan informasi kepada anggota tim yang lainnya.

Setiap bahasa pemrograman memiliki ciri yang khas di dalam memberikan notasi komentar di dalam kode programnya. Bahasa C/C++/C#/Java/JavaScript pada umumnya menggunakan notasi `//` atau `/* ... */` untuk penanda baris komentar. Sementara Python menggunakan notasi `#` atau `""" ... """` sebagai penanda baris komentar di dalam kode program. Di dalam Python, tanda `#` digunakan untuk komentar satu baris, sedangkan `""" ... """` digunakan untuk komentar panjang multibaris.

Berikut adalah contoh komentar yang valid di dalam Bahasa Python:

```
# Ini adalah contoh komentar satu baris tersendiri
# Ini adalah contoh komentar di baris kedua
```

```
y = x**2 + 3x + 5      # Ini adalah contoh komentar di belakang baris kode
```

```
"""
Ini adalah contoh komentar yang cukup panjang, sebab mungkin komentar yang pendek
tidak akan pernah cukup. Harap maklum, kita termasuk orang yang lebih banyak
berbicara dari pada berpikir dan bekerja. Itu sebabnya ketika ada berita viral
di sosmed, para komentator akan langsung berkumpul, tanpa terlebih dahulu cek n
ricek kebenaran beritanya.
"""
```

Tentu saja, kita jangan menambahkan komentar-komentar yang tidak berguna ke dalam kode program kita, sebagaimana contoh-contoh komentar di atas. Meskipun contoh-contoh komentar di atas valid secara teknis (sintaksnya benar), akan tetapi mubazir secara substansi. Tambahkan komentar jika memang diperlukan, tidak perlu setiap baris kode program ada komentarnya, dan tambahkan hanya komentar yang informatif. Jika kita membangun kode program secara kolaboratif, pastikan komentar-komentar yang kita buat dimengerti oleh anggota tim yang lain.

---

*Untuk JupyterLab, VS Code, dan Google Colab, comment dan uncomment dapat dilakukan dengan menekan kombinasi tombol **Ctrl** dan **/** secara bersamaan. Untuk platform pengembangan lainnya mungkin tekniknya akan berbeda.*

---

## Kata-kata Kunci Python dan Sensitivitas Huruf

Kata-kata kunci (*keywords*) Python adalah kata-kata tertentu yang sudah memiliki fungsi-fungsi spesifik dan sudah ditetapkan di dalam Python, seperti `if`, `for`, `and`, `or`, `True`, `False`, dan sebagainya. Sehingga kata-kata kunci ini tidak dapat dijadikan sebagai nama variabel, nama fungsi, nama kelas, dan sebagainya di dalam kode program Python.

## Bab II Fundamental Python

Tabel 2.10. Kata-kata kunci Bahasa Python

Keyword	Deskripsi
<u>and</u>	Operator logika and
<u>as</u>	Membuat sebuah alias
<u>assert</u>	Untuk debugging
<u>break</u>	Menghentikan putaran instruksi
<u>class</u>	Mendefinisikan sebuah kelas
<u>continue</u>	Melanjutkan putaran instruksi ke putaran berikutnya
<u>def</u>	Mendefinisikan sebuah fungsi
<u>del</u>	Menghapus sebuah variabel atau objek
<u>elif</u>	Pernyataan kondisional, sama dengan else if
<u>else</u>	Pernyataan kondisional jika semua if bernilai False
<u>except</u>	Apa yang akan dijalankan ketika sebuah eksepsi muncul
<u>False</u>	Nilai boolean False
<u>finally</u>	Apa yang akan dijalankan tanpa melihat kemunculan eksepsi
<u>for</u>	Membuat putaran instruksi for
<u>from</u>	Mengimpor bagian-bagian spesifik dari sebuah modul
<u>global</u>	Mendeklarasikan variabel global
<u>if</u>	Membuat pernyataan kondisional if
<u>import</u>	Mengimport sebuah modul
<u>in</u>	Mencek apakah sebuah nilai terdapat di dalam list, tuple, dan sebagainya
<u>is</u>	Menguji apakah dua variabel sama
<u>lambda</u>	Membuat fungsi tidak bernaama
<u>None</u>	Merepresentasikan nilai null (kosong)
<u>nonlocal</u>	Mendeklarasikan variabel non-lokal
<u>not</u>	Operator logika not
<u>or</u>	Operator logika or
<u>pass</u>	Pernyataan kosong, pernyataan yang tidak melakukan apa pun
<u>raise</u>	Membangkitkan sebuah eksepsi
<u>return</u>	Keluar dari sebuah fungsi dan mengembalikan sebuah nilai
<u>True</u>	Nilai boolean True
<u>try</u>	Membuat sebuah pernyataan eksepsi
<u>while</u>	Membuat putaran instruksi while
<u>with</u>	Digunakan untuk menyederhanakan penanganan pengecualian
<u>yield</u>	Untuk mengembalikan daftar nilai dari generator

Sumber: <https://www.w3schools.com>

Terkait sensitivitas huruf, sebagaimana Bahasa C/C++/C#/Java/JavaScript, Python adalah bahasa pemrograman yang *case sensitive*. Artinya huruf kapital dan huruf kecil akan dianggap berbeda. Baik di dalam kata-kata kunci, maupun di dalam penamaan identitas seperti variabel, fungsi, dan kelas. Sehingga di dalam Python, for akan berbeda dengan For atau FOR. Luas\_lahan, akan berbeda dengan luas\_lahan atau LUAS\_LAHAH.

## Indentasi

Setiap bahasa pemrograman memiliki teknik yang unik di dalam menandai blok kode-kode program. Visual Basic misalnya menggunakan kata kunci BEGIN dan END untuk mengawali dan mengakhiri blok kode program. Keluarga Bahasa C/C++/C# menggunakan tanda {} (kurung kurawal) untuk menandai blok kode program. Python sendiri memiliki cara yang unik di dalam menandai blok kode program, yaitu dengan teknik indentasi (*indentation*). Indentasi adalah penulisan baris paragraf yang agak menjorok masuk ke dalam. Hal ini yang menjadikan Bahasa Python menjadi unik dibandingkan dengan bahasa-bahasa pemrograman lainnya. Hal yang harus diwaspadai adalah programer Python pemula sering melakukan kesalahan di dalam teknik indentasi ini. Menurut <https://www.geeksforgeeks.org>, indentasi Python secara bawaan (*default*) adalah 4 (empat) spasi. Pada umumnya, platform pengembangan yang kita gunakan dalam menuliskan Bahasa Python, seperti JupyterLab, VS Code, atau Google Colab, memiliki fasilitas pengontrol indentasi secara otomatis.

Berikut adalah contoh indentasi dalam blok kode **if else** Python:

```
if nilai_sidang_skripsi >= 70:
    print('Selamat! Anda sudah berhasil lulus...')
else:
    print('Maaf ya, silahkan ngulang sidang lagi...')
```

Jika baris kode di bawah **if** atau di bawah **else** kehilangan indentasi, maka akan terjadi kesalahan sintaks. Dan kode program tidak dapat dijalankan. Akan tetapi, JupyterLab, VS Code, atau Google Colab biasanya akan membuatkan indentasi otomatis, untuk baris-baris kode yang memang teridentifikasi memerlukan indentasi. Jika tidak terbentuk indentasi otomatis, kita dapat membuat indentasi sendiri dengan menekan tombol **Tab** di keyboard. Tentu saja, ketika kita akan mengakhiri blok **if** atau **else**, jika terbentuk indentasi otomatis maka harus dihapus.

Hal yang harus diperhatikan lagi adalah, bahwa akhir baris-baris kode Python tidak diakhiri tanda apa pun, termasuk tanda ; (titik koma) sebagaimana keluarga Bahasa C/C++/C#/Java/JavaScript. Konon, programer C/C++/C#/Java/JavaScript yang pindah ke Python sering khilaf menambahkan tanda titik koma diakhir baris kode. Hal ini akan menyebabkan error pada kode Python.

## Jeda Baris Kode Python

Python memberikan fasilitas untuk memberikan jeda atau pemisah baris kode program yang terlalu panjang. Sehingga walaupun baris kodennya ditulis ke dalam beberapa baris, Python akan tetap menganggapnya sebagai satu baris kode. Karakter untuk memberi jeda atau memisahkan baris kode program tersebut adalah \ (backslash). Perhatikan contoh kode berikut:

```
a = 1 + 2 + 3 + 4 - 5 * 2
```

Kode program di atas, dapat juga ditulis sebagai berikut:

```
a = 1 + 2 + \
3 + 4 - \
5 * 2
```

Kode program di atas, keduanya akan tetap dianggap sebagai satu baris kode.

### Debugging Tips

Debugging merupakan proses untuk menemukan atau memperbaiki *bug* di dalam kode program. *bug* sendiri adalah istilah untuk kesalahan di dalam kode program. Bug pada umumnya akan selalu muncul pada saat programer membuat kode program. Bug dapat muncul di dalam kode program diakibatkan dari hal yang sangat sederhana, seperti lupa mengetikkan tanda titik, sampai yang paling kompleks, yaitu kesalahan alur atau logika pemrograman. Sehingga debugging merupakan aktivitas yang selalu dilakukan di dalam proses pembuatan kode program. Untuk mempermudah pekerjaan debugging, programer pada umumnya akan menjaga agar setiap baris kode programnya tidak terlalu kompleks. Dalam arti, jika terdapat sebuah formula atau algoritma yang kompleks, lebih baik kodennya dipisahkan ke dalam beberapa baris kode. Hal ini mengingat proses debugging biasanya dilakukan baris per baris kode program. Perhatikan contoh formula indeks vegetasi *Global Environmental Monitoring Index* (GEMI) (Pinty and Verstraete, 1992) berikut:

$$\text{GEMI} = \eta * (1 - 0.25 * \eta) - \frac{\text{Red} - 0.125}{1 - \text{Red}}$$

$$\text{Dimana: } \eta = \frac{2*(\text{NIR}^2 - \text{Red}^2) + 1.5*\text{NIR} + 0.5*\text{Red}}{\text{NIR} + \text{Red} + 0.5}$$

Formula GEMI di atas memuat sebuah parameter yaitu  $\eta$  (*eta*), yang harus dihitung dengan formula tersendiri. Dengan kata lain, pada GEMI di atas terdapat formula di dalam formula. Pada kasus seperti ini, kode programnya sebaiknya ditulis sebagai berikut:

```
eta = (2*(nir**2-red**2)+1.5*nir+0.5*red)/(nir+red+0.5)
gemi = eta*(1-0.25*eta)-((red-0.125)/(1-red))
```

Kode program di atas sebenarnya dapat ditulis sebagai berikut:

```
gemi = ((2*(nir**2-red**2)+1.5*nir+0.5*red)/(nir+red+0.5))*(1-0.25*((2*(nir**2-red**2)+1.5*nir+0.5*red)/(nir+red+0.5)))-((red-0.125)/(1-red))
```

Meskipun kedua format penulisan di atas secara teknis adalah legal, dan akan memberikan output yang sama. Akan tetapi, programer yang sudah berpengalaman tidak akan pernah melakukan opsi yang kedua. Sebab opsi yang kedua, yaitu menggabungkan langsung kalkulasi  $\eta$  ke dalam formula GEMI, akan membuat kode program menjadi sangat kompleks. Dan pada akhirnya nanti, jika pada baris kode itu terdapat kesalahan sintaks, tentu saja akan sangat menyulitkan proses *debugging*.

### Perulangan Instruksi dan Keputusan

Secara normal, ketika kode program Python dieksekusi, maka instruksi akan dieksekusi secara berurutan mulai dari kode pada baris yang paling atas menuju ke kode pada baris paling bawah. Akan tetapi, untuk keperluan tertentu, kita menginginkan agar eksekusi instruksi tidak berjalan linier dari atas ke bawah, melainkan berubah arah jika ditemukan suatu kondisi tertentu. Bahkan terkadang kita menginginkan baris-baris kode program tertentu hanya akan dieksekusi jika ditemukan kondisi tertentu, atau bahkan diulang-ulang sampai bertemu dengan kondisi tertentu, misalnya batasan nilai tertentu. Untuk keperluan ini, bahasa pemrograman seperti Python menyediakan fasilitas perulangan instruksi (*loop*) dan keputusan (*decision*).

## Pernyataan Kondisional `if else`

Pernyataan `if else` digunakan untuk menjalankan instruksi (atau sebaliknya tidak menjalankan instruksi), jika ditemukan kondisi tertentu. Kondisi yang dimaksud adalah ekspresi logika yang memberikan nilai boolean (`True` atau `False`). Contoh penggunaan `if else` yang paling sederhana adalah sebagai berikut:

```
nilai_kuliah = 85
if nilai_kuliah >= 60:
    print('Selamat! Anda lulus.')
```

Output:

```
Selamat! Anda lulus.
```

Jika ekspresi `nilai_kuliah >= 60` bernilai `True`, maka instruksi di bawah `if`, yaitu `print('Selamat! Anda lulus.')` akan dijalankan. Jika ekspresi `nilai_kuliah >= 60` bernilai `False`, maka instruksi `print('Selamat! Anda lulus.')` tidak akan dijalankan. Pada kode di atas, karena nilai kuliahnya adalah `85`, maka ekspresi `nilai_kuliah >= 60` bernilai `True`, dan instruksi `print('Selamat! Anda lulus.')` dijalankan. Coba Anda ganti nilai variabel `nilai_kuliah` dengan `45`, maka kode program tidak akan memberikan output apapun. Perhatikan kembali kode berikut:

```
nilai_kuliah = 45
if nilai_kuliah >= 60:
    print('Selamat! Anda lulus.')
else:
    print('wah, maaf ya. Anda harus mengulang lagi tahun depan.')
```

Output:

```
wah, maaf ya. Anda harus mengulang lagi tahun depan.
```

Pada kode di atas, karena ekspresi `nilai_kuliah >= 60` bernilai `False`, maka instruksi di bawah `if` tidak dieksekusi, dan yang akan dieksekusi adalah instruksi di bawah `else`.

Contoh lain:

```
nilai_kuliah = 65
if nilai_kuliah >= 70:
    print('Selamat! Anda lulus. Dan tidak perlu mengulang.')
elif nilai_kuliah < 70 and nilai_kuliah >= 60:
    print('Selamat! Anda lulus. Tetapi Anda direkomendasikan mengulang.')
else:
    print('wah, maaf ya. Anda harus mengulang lagi tahun depan.')
```

Output:

```
Selamat! Anda lulus. Tetapi Anda direkomendasikan mengulang.
```

## Bab II Fundamental Python

Pada kode di atas, `elif` berfungsi seperti `if` "alternatif", jika kondisi `if` di atasnya tidak terpenuhi (bernilai `False`). `if` juga dapat berada di dalam `if`, yang dikenal sebagai *nested if* atau lebih umum lagi *nested loop*. Perhatikan contoh berikut:

```
nilai_kuliah = 85
nilai_praktik = 67

if nilai_kuliah >= 70:
    if nilai_praktik >= 70:
        print('Selamat! Anda lulus. Dan tidak perlu mengulang.')
    else:
        print('Selamat! Anda lulus. Tetapi Anda harus remedial praktik.')
elif nilai_kuliah < 70 and nilai_kuliah >= 60:
    print('Selamat! Anda lulus. Tetapi Anda direkomendasikan mengulang.')
else:
    print('wah, maaf ya. Anda harus mengulang lagi tahun depan.')
```

Output:

```
Selamat! Anda lulus. Tetapi Anda harus remedial praktik.
```

Pada kode di atas, karena ekspresi `nilai_kuliah >= 70` di bawah `if` bernilai `True`, maka instruksi di bawahnya (yaitu ekspresi `if` lagi) akan dieksekusi. Akan tetapi, ekspresi `nilai_kuliah >= 70` di dalam `if` ini nilainya `False`, maka yang akan dieksekusi adalah instruksi di bawah `else` di dalam `if`.

### Pernyataan Kondisional `if else` dalam Satu Baris

Pernyataan kondisional `if else` dapat disingkat dalam satu baris. Hal ini kalau memang ekspresi logikanya sangat sederhana dan singkat. Formulanya adalah:

```
ekspresi if kondisi else eksepsi
```

ekspresi akan dieksekusi jika kondisi bernilai `True`. Jika kondisi bernilai `False` maka eksepsi yang akan dieksekusi.

Perhatikan contoh berikut:

```
nilai = 75
print('Lulus') if nilai>=60 else print('Mengulang')
```

Output:

```
Lulus
```

### Pernyataan Perulangan `for`

Perulangan `for` digunakan untuk melakukan iterasi instruksi secara berurutan. `for` dapat mengulang instruksi menurut list, tuple, dictionary, set, atau string. Di dalam pemrosesan citra digital, `for` memegang peranan yang sangat vital di dalam operasi pixel-pixel citra.

Perhatikan contoh penggunaan **for** berikut:

```
citra = [123, 456, 789]
for nilai_pixel in citra:
    print(nilai_pixel)
```

Output:

```
123
456
789
```

Contoh lain:

```
jlh_titik = int(input('Berapa titik yang Anda inginkan?'))
for i in range(jlh_titik):
    print('Titik', i+1)
```

Output:

```
Berapa titik yang Anda inginkan?
5
Titik 1
Titik 2
Titik 3
Titik 4
Titik 5
```

Pada contoh kode di atas, jika pengguna menginput **5**, kode program akan mencetak **Titik** sebanyak 5 kali. Variabel **i** dihitung mulai 0 (nol), oleh sebab itu harus ditambahkan dengan 1 (satu) agar perhitungan **Titik** dimulai dari 1. Fungsi **range** pada kode di atas adalah untuk merubah nilai integer yang diinput pengguna menjadi rentang perulangan. Maksudnya, jika pengguna menginput angka 5, maka akan diulang pada rentang 0 sampai 4 (5 kali).

**for** sering diimplementasikan untuk list atau string. Perhatikan contoh berikut:

```
nama_lengkap = input('Masukkan nama Anda:')
print(f'Hai, {nama_lengkap}!')
print(f'Nama Anda ada {len(nama_lengkap)} huruf.')
for i in range(len(nama_lengkap)):
    print(f'Huruf ke-{i+1} nama Anda adalah: {nama_lengkap[i]}')
```

Output:

```
Masukkan nama Anda:
EDWARD
Hai, EDWARD!
Nama Anda ada 6 huruf.
Huruf ke-1 nama Anda adalah: E
Huruf ke-2 nama Anda adalah: D
Huruf ke-3 nama Anda adalah: A
Huruf ke-4 nama Anda adalah: R
Huruf ke-5 nama Anda adalah: D
Huruf ke-6 nama Anda adalah: D
```

## Bab II Fundamental Python

Fungsi `len` pada kode di atas adalah untuk menghitung panjang list atau string. Untuk string, `len` akan menghitung jumlah hurufnya. Jika nama yang diinput pengguna adalah `EDWARD` sebagaimana contoh di atas, maka `len` akan memberikan angka 6, dan `range` akan membuat rentang pengulangan 0 sampai 5 (6 kali). `for` juga dapat ditulis dalam bentuk satu baris, sebagaimana contoh berikut:

```
daftar_angka = [angka for angka in range(10)]  
print(daftar_angka)
```

Output:

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

### Pernyataan Perulangan `while`

Pernyataan perulangan `while` memiliki fungsi yang mirip dengan `for`, yaitu mengulang instruksi. Perbedaannya adalah, `for` akan mengulang instruksi berdasarkan rentang jumlah perulangan yang sudah diketahui secara eksak. Sedangkan `while` akan terus mengulang instruksi sampai ekspresi bernilai `False`. Dengan kata lain, `while` digunakan untuk memutar instruksi yang jumlah putarannya tidak diketahui secara pasti.

Untuk contoh kasus perulangan di dunia nyata, perhatikan perbedaan dua perintah berikut:

*'Ayo, kalian masing-masing push up 50 kali...!!!'*

*'Ayo, kalian masing-masing push up sampai komandan datang ke tempat ini...!!!'*

Perintah pertama identik dengan `for`, sementara perintah kedua lebih tepat diidentikkan dengan `while`. Sebab perintah pertama jelas harus *push up* berapa kali, sementara perintah kedua tidak jelas harus *push up* berapa kali.

Contoh kode program sederhana yang menggunakan `while`:

```
i = 1  
while i <= 5:  
    print(i)  
    i += 1
```

Output:

```
1  
2  
3  
4  
5
```

Instruksi di atas akan memutar instruksi `while` sampai ekspresi `i <= 5` bernilai `False`. Atau dengan kata lain, jika `i` sudah bernilai lebih dari 5. Dimana variabel `i` sendiri pada awalnya sudah diinisiasi dengan nilai 1. Konsekuensi dari `while` adalah terdapat kemungkinan bahwa instruksi di bawahnya tidak akan dieksekusi sama sekali. Yaitu ketika ekspresi di belakang `while` dari awal sudah bernilai `False`.

Sebaliknya, instruksi di bawah `while` juga dapat berputar terus, hal ini akan terjadi jika ekspresi di belakang `while` selalu bernilai `True`. Perhatikan contoh kode berikut:

```
true_password = 'PythonIsTheBest'
password = input('Masukkan password Anda:')

while password != true_password:
    print('Password yang Anda masukkan salah!')
    password = input('Silahkan masukkan ulang password Anda:')

print('Terima kasih. Anda berhasil masuk.')
```

Kode di atas akan terus-terusan meminta password, sampai pengguna memasukkan password yang benar, yaitu `PythonIsTheBest`. Dan jika dari awal pengguna sudah memasukkan password yang benar, maka `while password != true_password` akan langsung bernilai `False`, dan instruksi di dalam `while` tidak akan pernah dieksekusi.

Peringatan keras. Jangan jalankan kode program berikut!

```
while True:
    print('Ayo, semangat belajar Python!!!')
```

Kode program di atas akan berputar “selamanya”, dan terus menerus mencetak tulisan `'Ayo, semangat belajar Python!!!'` ke layar. Sebab ekspresi di belakang `while` selalu `True`.

### Pernyataan Lompatan `break` dan `continue`

Pernyataan `break` dan `continue` memiliki fungsi yang saling berlawanan. Sesuai dengan namanya, `break` akan memaksa untuk menghentikan (keluar) dari putaran instruksi `for` atau `while`, jika bertemu dengan kondisi tertentu. Sebaliknya, `continue` akan memaksa untuk melanjutkan memutar instruksi ke putaran berikutnya, sebelum instruksi di dalam satu putaran benar-benar selesai dieksekusi.

Perhatikan contoh penggunaan `break` berikut:

```
i = 1

while i <= 100:
    print(f'Putaran instruksi yang ke-{i}')
    i += 1
    if i > 5:
        break

print('Eksekusi program selesai...')
```

Output:

```
Putaran instruksi yang ke-1
Putaran instruksi yang ke-2
Putaran instruksi yang ke-3
Putaran instruksi yang ke-4
Putaran instruksi yang ke-5
Eksekusi program selesai...
```

## Bab II Fundamental Python

Pada kode di atas, tanpa adanya `break`, putaran instruksi `while` akan diputar 100 kali. Akan tetapi, ketika kondisi di belakang `if` bernilai `True` (yaitu ketika variabel `i` nilainya lebih dari 5), maka pernyataan `break` akan dieksekusi. Pernyataan `break` akan menghentikan paksa putaran instruksi `while`, alur eksekusi akan keluar dari `while`, dan melanjutkan eksekusi ke baris kode berikutnya, yaitu `print('Eksekusi program selesai...')`. Dalam konteks pemrosesan citra penginderaan jauh, `break` biasa dipakai di dalam iterasi instruksi jika kita ingin menghentikan putaran `while` ketika sudah ditemukan nilai tertentu yang diinginkan. Misalnya pada saat akurasi atau error sudah mencapai ambang (*threshold*) nilai tertentu.

Perhatikan contoh penggunaan `continue` berikut:

```
for i in range(5):
    if (i+1) == 3:
        continue
    print(f'Putaran instruksi yang ke-{i+1}')
```

Output:

```
Putaran instruksi yang ke-1
Putaran instruksi yang ke-2
Putaran instruksi yang ke-4
Putaran instruksi yang ke-5
```

Pada kode di atas perhatikan outputnya yang aneh, dimana output putaran instruksi yang ke-3 tidak ada. Hal ini karena `continue` memaksa lompat ke putaran instruksi berikutnya sebelum putaran instruksi ketiga selesai dieksekusi sampai ke baris yang paling bawah.

## Struktur Data

### List

List merupakan sebuah kontainer yang menyimpan sejumlah data di dalamnya. List adalah solusi untuk variabel yang menyimpan banyak data. Faktanya, hal ini sering kita temukan pada aplikasi di dunia nyata. Misalnya, kita punya daftar nama 5 orang surveyor lapangan.

Dari pada menulis seperti ini:

```
surveyor1 = 'Noah'
surveyor2 = 'Owen'
surveyor3 = 'Liam'
surveyor4 = 'James'
surveyor5 = 'Ethan'
```

Lebih baik ditulis dalam bentuk list berikut:

```
surveyor = ['Noah', 'Owen', 'Liam', 'James', 'Ethan']
```

Sebuah list didefinisikan sebagaimana sebuah variabel, kemudian semua data di dalamnya ditempatkan di dalam kurung siku [], dengan pemisah tanda koma antar elemen. Setiap elemen di dalam list memiliki indeks (nomor urut), dimana indeks list selalu dimulai dari 0, bukan 1. Perhatikan ilustrasi elemen list dan indeks list berikut:

Elemen	Noah	Owen	Liam	James	Ethan
Indeks	0	1	2	3	4

Untuk selanjutnya, elemen-elemen list dapat diakses dengan menggunakan indeks-indeksnya. Perhatikan kode lengkapnya sebagai berikut:

```
surveyor = ['Noah', 'Owen', 'Liam', 'James', 'Ethan']

print(f'Output 1: {surveyor}')
print(f'Output 2: {surveyor[3]}')
print(f'Output 3: {surveyor[0:3]}')
print(f'Output 4: {surveyor[-1]}')
print(f'Output 5: {surveyor[-3:-1]}')
print(f'Output 6: {surveyor[:3]}')
```

Output:

```
Output 1: ['Noah', 'Owen', 'Liam', 'James', 'Ethan']
Output 2: James
Output 3: ['Noah', 'Owen', 'Liam']
Output 4: Ethan
Output 5: ['Liam', 'James']
Output 6: ['Noah', 'Owen', 'Liam']
```

Penjelasan output kode program di atas:

Output 1: `surveyor` berarti mengakses semua elemen di dalam list.

Output 2: `surveyor[3]` berarti mengakses indeks ke-3 atau elemen yang ke-4 di dalam list.

Output 3: `surveyor[0:3]` berarti mengakses dari elemen pertama hingga elemen ke-(4 – 1) di dalam list.

Output 4: `surveyor[-1]` berarti mengakses elemen ke-1 dari kanan di dalam list. Kalau indeksnya negatif berarti indeks dihitung dari kanan, sehingga elemen ke-(-1) identik dengan indeks ke-4 atau elemen ke-5.

Output 5: `surveyor[-3:-1]` berarti mengakses semua elemen ke-4 dan ke-5 di dalam list. Dihitung dari -1 (paling kanan) dan  $-3 + 1 = -2$  dari kanan.

Output 6: `surveyor[:3]` berarti mengakses semua elemen hingga elemen ke-(4 – 1) di dalam list. Instruksi ini identik dengan `surveyor[0:3]`.

Perhatikan akses elemen list menurut rentang tertentu seperti pada `surveyor[0:3]`. 0 di sini bersifat **inklusif**, artinya indeks yang ke-0 akan diikutkan untuk diakses. Sedangkan 3 di sini bersifat **eksklusif**, artinya indeks yang ke-3 tidak akan diikutkan untuk diakses. Dengan kata lain, instruksi `surveyor[0:3]` akan mengakses elemen list dari indeks yang ke-0 hingga indeks yang ke-2. Karakteristik inklusif dan eksklusif indeks ini harus benar-benar difahami, sebab nanti banyak potensi kesalahan teknis berasal dari sini.

List bersifat *mutable*, artinya kontennya dapat dirubah. Untuk menambahkan elemen digunakan instruksi `append`. Perhatikan contoh kode berikut:

```
surveyor = ['Noah', 'Owen', 'Liam', 'James', 'Ethan']
surveyor.append('Henry')
print(surveyor)
```

## Bab II Fundamental Python

Output:

```
['Noah', 'Owen', 'Liam', 'James', 'Ethan', 'Henry']
```

Sebuah list juga dapat digabung dengan list lainnya dengan instruksi `extend`. Perhatikan contoh kode berikut:

```
surveyor = ['Noah', 'Owen', 'Liam', 'James', 'Ethan']
surveyor_tambahan = ['Emma', 'olivia', 'Amelia']
surveyor.extend(surveyor_tambahan)
print(surveyor)
```

Output:

```
['Noah', 'Owen', 'Liam', 'James', 'Ethan', 'Emma', 'olivia', 'Amelia']
```

Untuk menyisipkan elemen pada indeks tertentu digunakan perintah `insert`. Perhatikan contoh kode berikut:

```
surveyor = ['Noah', 'Owen', 'Liam', 'James', 'Ethan']
surveyor.insert(2, 'Henry')
print(surveyor)
```

Output:

```
['Noah', 'Owen', 'Henry', 'Liam', 'James', 'Ethan']
```

Mirip dengan perintah `extend`, list juga dapat digabung seperti operasi penjumlahan biasa. Perhatikan contoh kode berikut:

```
surveyor1 = ['Noah', 'Owen', 'Liam', 'James', 'Ethan']
surveyor2 = ['Emma', 'olivia', 'Amelia']
surveyor = surveyor1 + surveyor2
print(surveyor)
```

Output:

```
['Noah', 'Owen', 'Liam', 'James', 'Ethan', 'Emma', 'olivia', 'Amelia']
```

Untuk menghapus elemen list digunakan perintah `remove`. Perhatikan contoh kode berikut:

```
surveyor = ['Noah', 'Owen', 'Liam', 'James', 'Ethan']
print(surveyor)
surveyor.remove('Ethan')
print(surveyor)
```

Output:

```
[['Noah', 'Owen', 'Liam', 'James', 'Ethan']
 ['Noah', 'Owen', 'Liam', 'James']]
```

Untuk menghapus elemen berdasarkan indeksnya digunakan fungsi `pop`. Perhatikan contoh kode berikut:

```
surveyor = ['Noah', 'Owen', 'Liam', 'James', 'Ethan']
print(surveyor)
surveyor.pop(2)
print(surveyor)
```

Output:

```
['Noah', 'Owen', 'Liam', 'James', 'Ethan']
 ['Noah', 'Owen', 'James', 'Ethan']]
```

Menghapus elemen list berdasarkan indeks juga dapat dilakukan dengan kata kunci `del`. Perhatikan contoh kode berikut:

```
surveyor = ['Noah', 'Owen', 'Liam', 'James', 'Ethan']
print(surveyor)
del surveyor[:2]
print(surveyor)
```

Output:

```
['Noah', 'Owen', 'Liam', 'James', 'Ethan']
 ['Liam', 'James', 'Ethan']]
```

Dalam hal ini, penggunaan `del` mirip dengan teknik mengakses elemen list menggunakan indeksnya. Sehingga `del` dapat menghapus banyak elemen menggunakan rentang indeksnya, dibandingkan dengan fungsi `pop` yang hanya bisa menghapus satu elemen berdasarkan indeksnya.

### *Menghitung Panjang List*

Menghitung jumlah elemen atau panjang sebuah list dapat dilakukan dengan fungsi `len`. Perhatikan contoh kode berikut:

```
surveyor = ['Noah', 'Owen', 'Liam', 'James', 'Ethan']
jlh_surveyor = len(surveyor)
print(jlh_surveyor)
```

Output:

5

### List Multidimensi

Tidak hanya satu dimensi sebagaimana contoh-contohnya, sebuah list juga dapat multidimensi. Atau dikenal juga sebagai list di dalam list. Perhatikan contoh kode berikut:

```
daftar = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]  
print(daftar)
```

Output:

```
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

### Perulangan dan List

Instruksi perulangan **for** sering diasosiasikan dengan list. Perhatikan contoh kode berikut:

```
surveyor = ['Noah', 'Owen', 'Liam', 'James', 'Ethan']  
for nama_surveyor in surveyor:  
    print(nama_surveyor)
```

Output:

```
Noah  
Owen  
Liam  
James  
Ethan
```

Alternatif lainnya, dengan output yang sedikit berbeda:

```
surveyor = ['Noah', 'Owen', 'Liam', 'James', 'Ethan']  
for i in range(len(surveyor)):  
    print(f'Nama surveyor ke-{i+1} adalah {surveyor[i]}')
```

Output:

```
Nama surveyor ke-1 adalah Noah  
Nama surveyor ke-2 adalah Owen  
Nama surveyor ke-3 adalah Liam  
Nama surveyor ke-4 adalah James  
Nama surveyor ke-5 adalah Ethan
```

Apa bedanya perulangan **for** yang pertama dengan alternatif? Untuk yang pertama, kita memutar **for** dengan menggunakan elemen list. Sedangkan untuk yang alternatif, kita memutar **for** menggunakan indeks list. Pada praktiknya, kedua opsi ini akan digunakan sesuai keperluan.

### List Kosong

Pada kasus tertentu, misalnya ketika kita akan menyimpan data hasil dari sejumlah kalkulasi yang dilakukan beberapa kali, kita akan membuat list kosong terlebih dahulu sebelum kalkulasi dilakukan. Kemudian ketika kalkulasi dilakukan, satu per satu hasilnya akan ditambahkan ke dalam list. Sehingga hasil kalkulasi nantinya akan tersimpan di dalam satu tempat, yaitu satu list. Perhatikan contoh kode programnya berikut:

```
print('Ayo anak-anak! Mari berhitung 1 sampai 10...')

hitungan = [] # Deklarasi sebuah list kosong

for i in range(10):
    hitungan.append(i+1)

print(hitungan)
```

Output:

```
Ayo anak-anak! Mari berhitung 1 sampai 10...
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Kombinasi sebuah list kosong dan perulangan `for` seperti contoh kode di atas merupakan sebuah teknik yang lumrah dilakukan di dalam komputasi saintifik. Misalnya dalam sebuah algoritma, kita sedang mencari parameter paling optimum untuk algoritma tersebut. Parameter paling optimum adalah parameter yang dapat memberikan akurasi paling maksimum. Konsekuensinya, sejumlah parameter akan dicoba dan kalkulasi algoritma akan dieksekusi berulang-ulang menggunakan instruksi `for`. Setiap satu kalkulasi, akan dihitung akurasinya, dan untuk setiap data akurasi yang keluar akan disimpan ke dalam list. Langkah berikutnya adalah kita mencari nilai maksimum akurasinya dengan menggunakan fungsi `max(nama_list)`.

### Metode-metode List

Berikut adalah fungsi-fungsi atau metode-metode yang dapat diterapkan pada sebuah list:

Tabel 2.11. Fungsi-fungsi yang dapat diterapkan pada list

Method	Description
<code>append()</code>	Adds an element at the end of the list
<code>clear()</code>	Removes all the elements from the list
<code>copy()</code>	Returns a copy of the list
<code>count()</code>	Returns the number of elements with the specified value
<code>extend()</code>	Add the elements of a list (or any iterable), to the end of the current list
<code>index()</code>	Returns the index of the first element with the specified value
<code>insert()</code>	Adds an element at the specified position
<code>pop()</code>	Removes the element at the specified position
<code>remove()</code>	Removes the first item with the specified value
<code>reverse()</code>	Reverses the order of the list
<code>sort()</code>	Sorts the list

Sumber: <https://www.w3schools.com>

### Tuple

Tuple adalah kontainer data yang mirip dengan list. Perbedaannya, tuple bersifat *immutable*. Yang berarti isinya bersifat permanen, tidak bisa dirubah-rubah. Sebuah tuple didefinisikan sebagai berikut:

```
surveyor = ('Noah', 'Owen', 'Liam', 'James', 'Ethan')
print(surveyor)
print(surveyor[1:3])
```

Output:

```
('Noah', 'Owen', 'Liam', 'James', 'Ethan')
('Owen', 'Liam')
```

Perhatikan bahwa sebuah tuple didefinisikan dengan kurung biasa (), bukan kurung siku seperti list. Akan tetapi, sistem indeks dan cara mengakses indeks tuple sama persis dengan list. Di dalam Python, list pada umumnya hanya menyimpan satu jenis data, misalnya angka saja atau teks saja. Sedangkan tuple pada umumnya digunakan untuk menyimpan data dengan tipe yang heterogen. Sebuah tuple juga dapat memuat tuple lagi di dalamnya, dikenal sebagai *nested tuple*.

Sebuah list dapat dikonversi menjadi sebuah tuple dengan fungsi **tuple**. Perhatikan contoh kode berikut:

```
daftar_surveyor_edit = ['Noah', 'Owen', 'Liam', 'James', 'Ethan']
daftar_surveyor_fix = tuple(daftar_surveyor_edit)

print(daftar_surveyor_edit)
print(daftar_surveyor_fix)
```

Output:

```
['Noah', 'Owen', 'Liam', 'James', 'Ethan']
('Noah', 'Owen', 'Liam', 'James', 'Ethan')
```

Sebaliknya, sebuah tuple juga dapat dikonversi menjadi sebuah list dengan fungsi **list**. Tekniknya sama dengan konversi list ke tuple pada contoh kode di atas.

### Set

Set atau himpunan merupakan kontainer untuk menyimpan data yang unik. Set mirip dengan list karena bersifat *mutable*, atau isinya dapat dirubah. Perbedaannya dengan list adalah, set tidak mengizinkan adanya duplikasi data. Perhatikan contoh kode berikut:

```
himpunan = {3, 3, 3, 2, 1, 5, 4, 4}
print(himpunan)
```

Output:

```
{1, 2, 3, 4, 5}
```

Sebuah set didefinisikan dengan kurung kurawal {}. Pada contoh kode di atas, meskipun terdapat banyak duplikasi data. Akan tetapi, ketika setnya terbentuk, duplikasi data akan dihapus oleh sistem. Sebuah list juga dapat dikonversi menjadi sebuah set menggunakan fungsi **set**. Perhatikan contoh kode berikut:

```
daftar = [3, 3, 3, 2, 1, 5, 4, 4]
himpunan = set(daftar)
print(himpunan)
```

Output:

```
{1, 2, 3, 4, 5}
```

Set tidak memiliki indeks sebagaimana list atau tuple. Sehingga kita tidak dapat mengakses elemen-elemen di dalam set menggunakan indeksnya seperti list atau tuple. Pada contoh kode di atas, jika kita mencoba mengakses indeks set, misalnya `himpunan[5]`, hasilnya akan error. Akan tetapi, elemen-elemen set bersifat *iterable* sebagaimana list. Sehingga elemen-elemen set dapat diiterasi menggunakan pernyataan **for**. Perhatikan contoh kode berikut:

```
himpunan = {3, 3, 3, 2, 1, 5, 4, 4}
for elemen in himpunan:
    print(elemen)
```

Output:

```
1
2
3
4
5
```

Kita dapat memverifikasi keberadaan suatu elemen di dalam set. Perhatikan contoh kode berikut:

```
surveyor = {'Noah', 'Owen', 'Liam', 'James', 'Ethan'}
cek_surveyor = 'Ethan' in surveyor
print(cek_surveyor)
```

Output:

```
True
```

Pada kode di atas, ekspresi '`Ethan`' **in** `surveyor` akan bernilai `True` jika entri `Ethan` memang berada di dalam set `surveyor`.

Elemen di dalam set dapat ditambahkan dengan fungsi `add`. Perhatikan contoh kode berikut:

```
surveyor = {'Noah', 'Owen', 'Liam', 'James', 'Ethan'}
surveyor.add('Henry')
print(surveyor)
```

Output:

## Bab II Fundamental Python

```
{'Ethan', 'Liam', 'Noah', 'James', 'Owen' 'Henry'}
```

Jika elemen yang ditambahkan lebih dari satu, gunakan fungsi `update`. Tekniknya sama dengan fungsi `add` pada contoh kode di atas. Sebagaimana list, untuk menghitung panjang atau jumlah data di dalam set kita juga dapat menggunakan fungsi `len`. Dan untuk mencari nilai maksimum dan minimum di dalam set, kita dapat menggunakan fungsi `max` dan `min`. Untuk menghapus sebuah elemen di dalam set dapat digunakan fungsi `remove`, `discard`, atau `pop` seperti list. Untuk menghapus semua elemen di dalam set dapat digunakan fungsi `clear`. Fungsi `remove` akan menghasilkan error jika elemen yang mau dihapus tidak ada di dalam set. Sementara fungsi `pop` akan menghapus elemen terakhir di dalam sebuah set.

Set di dalam Python identik dengan set (himpunan) dalam matematika. Sehingga berlaku logika-logika himpunan di dalam matematika. Seperti logika *or/union* (`|`), logika *and/intersect* (`&`), logika *difference* (`-`), dan logika *symmetric difference* (`^`). Tentu saja, hal ini mengacu kepada teori-teori himpunan di dalam ilmu matematika. Di sinilah perbedaan dan keunggulan utama set dibandingkan dengan list dan tuple. Perhatikan contoh kode berikut:

```
citra1 = {'Sentinel-2', 'Landsat-8', 'Pleiades', 'Sentinel-1', 'Landsat-9',  
'Sentinel-3'}  
citra2 = {'Pleiades', 'Sentinel-2', 'Landsat-9', 'ASTER'}  
  
print(f'Union citra1 dan citra2: {citra1 | citra2}')  
print(f'Intersect citra1 dan citra2: {citra1 & citra2}')  
print(f'Difference citra1 dan citra2: {citra1 - citra2}')  
print(f'Symmetric difference citra1 dan citra2: {citra1 ^ citra2}')
```

Output:

```
Union citra1 dan citra2: {'Sentinel-1', 'ASTER', 'Pleiades', 'Landsat-8',  
'Landsat-9', 'Sentinel-2', 'Sentinel-3'}  
Intersect citra1 dan citra2: {'Landsat-9', 'Sentinel-2', 'Pleiades'}  
Difference citra1 dan citra2: {'Landsat-8', 'Sentinel-1', 'Sentinel-3'}  
Symmetric difference citra1 dan citra2: {'ASTER', 'Landsat-8', 'Sentinel-1',  
'Sentinel-3'}
```

Alternatif lain dari menggunakan operator (`|`, `&`, `-`, `^`) sebagaimana contoh di atas, kita juga dapat menggunakan fungsi-fungsi berikut:

```
citra1.union(citra2)  
citra1.intersect(citra2)  
citra1.difference(citra2)  
citra1.symmetric_difference(citra2)
```

Hasilnya akan sama persis dengan yang menggunakan operator. Kita dapat mengeksplorasi kemampuan set untuk melakukan analisis-analisis yang memang memerlukan logika-logika himpunan matematika. Dan pada kenyataannya, analisis-analisis seperti union, intersect, dan sebagainya, merupakan analisis-analisis geospasial yang paling populer diterapkan.

Sebuah set dapat menyimpan tuple di dalamnya, tetapi tidak dapat menyimpan list di dalamnya. Contoh kasus pemanfaatan tuple di dalam set misalnya pada kasus mencari perbedaan atau kesamaan koordinat. Di bawah ini disajikan contoh kasus dimana terdapat dua orang surveyor yang mengambil sejumlah koordinat titik di lapangan. Kemudian tugas kita sebagai analis adalah mengkompilasi (menggabungkan) data koordinat hasil survei menjadi satu data. Akan tetapi, kita harus mencari dan membuang titik-titik yang kemungkinan saling tumpang tindih antar surveyor.

Berikut adalah contoh kode programnya:

```
koordinat_surveyor1 = {(114.328, -3.112), (114.326, -3.110), (114.318, -3.108)}
koordinat_surveyor2 = {(114.324, -3.107), (114.328, -3.112), (114.311, -3.092),
(114.323, -3.104), (114.318, -3.108)}

koordinat_kompilasi = koordinat_surveyor1 | koordinat_surveyor2
koordinat_overlap = koordinat_surveyor1 & koordinat_surveyor2

print(f'Koordinat titik hasil kompilasi: {koordinat_kompilasi}')
print(f'Koordinat titik tumpang tindih: {koordinat_overlap}')
```

Output:

```
Koordinat titik hasil kompilasi: {(114.311, -3.092), (114.328, -3.112), (114.324,
-3.107), (114.326, -3.11), (114.323, -3.104), (114.318, -3.108)}
Koordinat titik tumpang tindih: {(114.328, -3.112), (114.318, -3.108)}
```

## Dictionary

Dictionary (atau **dict**) adalah sebuah set atau himpunan berpasangan antara *key* (kunci) dan *value* (nilai). Sebagaimana set, dictionary bersifat *mutable*, isinya dapat diperbarui. Dictionary pada dasarnya adalah sebuah struktur basisdata (*database*), sehingga dictionary memegang peranan yang sangat vital di dalam manajemen dan analisis data. Terutama analisis data tabular menggunakan *Python Data Analysis* (Pandas) nantinya. Dictionary dapat digunakan untuk mengkonversi list atau *numpy array* menjadi tabel untuk kepentingan analisis data.

Perhatikan contoh dasar sebuah dictionary berikut:

```
data_kota = {
    'Nama': 'Banjarbaru',
    'Status': 'Ibukota provinsi',
    'Provinsi': 'Kalimantan Selatan'
}

print(data_kota)
print(data_kota['Provinsi'])
```

Output:

```
{'Nama': 'Banjarbaru', 'Status': 'Ibukota provinsi', 'Provinsi': 'Kalimantan
Selatan'}
Kalimantan Selatan
```

Dictionary menyerupai set, yaitu didefinisikan menggunakan kurung kurawal `{}`. Akan tetapi, data di dalam dictionary saling berpasangan antara *key* (kunci) dan *value* (nilai). Dalam hal ini, **Nama**, **Status**, dan **Provinsi** adalah *key*. Sedangkan **Banjarbaru**, **Ibukota provinsi**, dan **Kalimantan Selatan** adalah *value*. *Value* di dalam dictionary dapat diakses menggunakan *key*-nya. Sebagaimana set yang wajib memiliki data unik, tidak boleh ada duplikasi *key* di dalam dictionary. Untuk *value*-nya sendiri tidak bermasalah kalau misalnya terdapat duplikasi data. Jika dictionary dikonversi menjadi sebuah tabel menggunakan Pandas, maka *key* akan menjadi header atau kolom dari tabelnya. Dictionary tidak memiliki indeks sebagaimana list atau tuple. Isi dictionary juga dapat dirubah-rubah sebagaimana list dan set.

## Bab II Fundamental Python

### Fungsi Konstruktor `dict()`

Dictionary dapat dibuat menggunakan fungsi `dict`. Dalam hal ini, ada tiga opsi variasi penggunaan fungsi `dict`. Perhatikan contoh kode berikut:

```
data_kota_1 = dict(Nama='Banjarbaru', Status='Ibukota provinsi',  
Provinsi='Kalimantan Selatan')  
data_kota_2 = dict([('Nama', 'Banjarbaru'), ('Status', 'Ibukota provinsi'),  
('Provinsi', 'Kalimantan Selatan')])  
data_kota_3 = dict([('Nama', 'Banjarbaru'), ['Status', 'Ibukota provinsi'],  
['Provinsi', 'Kalimantan Selatan']])  
  
print(data_kota_1)  
print(data_kota_2)  
print(data_kota_3)
```

Output:

```
{'Nama': 'Banjarbaru', 'Status': 'Ibukota provinsi', 'Provinsi': 'Kalimantan  
Selatan'}  
{'Nama': 'Banjarbaru', 'Status': 'Ibukota provinsi', 'Provinsi': 'Kalimantan  
Selatan'}  
{'Nama': 'Banjarbaru', 'Status': 'Ibukota provinsi', 'Provinsi': 'Kalimantan  
Selatan'}
```

Ketiga opsi variasi penggunaan fungsi `dict` di atas akan memberikan output yang sama. Opsi yang pertama seperti deklarasi variabel, sehingga kuncinya tidak perlu dijadikan string (pakai tanda petik). Opsi yang kedua, kunci dan nilai berpasangan di dalam tuple. Dan opsi yang ketiga, kunci dan nilai berpasangan di dalam list. Sebuah kunci dapat dirubah nilainya, atau sebuah kunci baru dapat ditambahkan ke dalam list dengan cara seperti berikut:

```
data_kota = {'Nama': 'Banjarbaru', 'Status': 'Ibukota provinsi', 'Provinsi':  
'Kalimantan Selatan'}  
  
data_kota['Nama'] = 'Banjarmasin'  
data_kota['Status'] = 'Kota'  
data_kota['Luas'] = '98.46 km2'  
  
print(data_kota)
```

Output:

```
{'Nama': 'Banjarmasin', 'Status': 'Kota', 'Provinsi': 'Kalimantan Selatan',  
'Luas': '98.46 km2'}
```

Pada kode di atas, kunci `Nama` dan `Status` dirubah nilainya. Sementara sebuah kunci baru, yaitu `Luas` ditambahkan ke dalam dictionary. Entri di dalam dictionary dapat dihapus berdasarkan kuncinya menggunakan `pop` dan `del`. Metode lainnya, yaitu `clear` akan menghapus semua entri di dalam dictionary. Perhatikan contoh penggunaan `del` di bawah:

```
data_kota = {'Nama': 'Banjarbaru', 'Status': 'Ibukota provinsi', 'Provinsi':  
'Kalimantan Selatan'}  
  
del data_kota['Status']  
  
print(data_kota)
```

Output:

```
{'Nama': 'Banjarbaru', 'Provinsi': 'Kalimantan Selatan'}
```

### Konstruksi Dictionary dari List

Nilai-nilai dictionary dapat diambil dari list. Perhatikan contoh kode berikut:

```
no_kab = [1, 2, 3, 4, 5]
nama_kab = ['Banjar', 'Tanah Laut', 'Barito Kuala', 'Tapin', 'Hulu Sungai Selatan']
ibukota = ['Martapura', 'Pelaihari', 'Marabahan', 'Rantau', 'Kandangan']

kabupaten = {
    'Nomor': no_kab,
    'Nama Kabupaten': nama_kab,
    'Ibukota': ibukota
}

print(kabupaten)
```

Output:

```
{'Nomor': [1, 2, 3, 4, 5], 'Nama Kabupaten': ['Banjar', 'Tanah Laut', 'Barito Kuala', 'Tapin', 'Hulu Sungai Selatan'], 'Ibukota': ['Martapura', 'Pelaihari', 'Marabahan', 'Rantau', 'Kandangan']}
```

Konstruksi dictionary menggunakan list seperti pada contoh kode di atas umum dilakukan ketika nilai-nilai dictionary didapatkan dari hasil proses-proses kalkulasi atau analisis di dalam Python. Dengan kata lain, hal yang seperti ini sangat diperlukan ketika kita sedang merekap nilai-nilai hasil perhitungan ke dalam sebuah tabel. Misalnya, hasil perhitungan nilai-nilai koefisien korelasi dari sejumlah model persamaan regresi. Perhatikan lagi contoh kode program di bawah:

```
no_kab = [1, 2, 3, 4, 5]
nama_kab = ['Banjar', 'Tanah Laut', 'Barito Kuala', 'Tapin', 'Hulu Sungai Selatan']
ibukota = ['Martapura', 'Pelaihari', 'Marabahan', 'Rantau', 'Kandangan']

kabupaten = {
    'Nomor': no_kab,
    'Nama Kabupaten': nama_kab,
    'Ibukota': ibukota
}

print(kabupaten.keys())
print(kabupaten.values())
print(kabupaten.items())
```

Output:

```
dict_keys(['Nomor', 'Nama Kabupaten', 'Ibukota'])
dict_values([1, 2, 3, 4, 5], ['Banjar', 'Tanah Laut', 'Barito Kuala', 'Tapin', 'Hulu Sungai Selatan'], ['Martapura', 'Pelaihari', 'Marabahan', 'Rantau', 'Kandangan'])
dict_items([('Nomor', [1, 2, 3, 4, 5]), ('Nama Kabupaten', ['Banjar', 'Tanah Laut', 'Barito Kuala', 'Tapin', 'Hulu Sungai Selatan']), ('Ibukota', ['Martapura', 'Pelaihari', 'Marabahan', 'Rantau', 'Kandangan'])])
```

## Bab II Fundamental Python

Pada kode program di atas `keys()` digunakan untuk menampilkan semua key, `values()` digunakan untuk menampilkan semua value, dan `items()` digunakan untuk menampilkan pasangan key dan value. Faktanya, selain dapat menyimpan list, sebuah dictionary juga dapat menyimpan tuple, set, atau bahkan dictionary lain. Pemanfaatan dictionary nanti akan lebih mudah difahami ketika membahas topik Pandas pada bagian berikutnya di dalam buku ini.

### Fungsi

Fungsi merupakan bagian utama dalam bahasa pemrograman. Di masa kejayaan bahasa pemrograman terstruktur, fungsi memegang peranan terpenting dalam kode program. Sebab fungsi akan memisahkan/memecah-belah masalah besar menjadi masalah-masalah kecil, dan menyelesaiakannya secara terpisah. Pada pembahasan terdahulu sudah dijelaskan tentang blok kode program. Blok adalah kode program di dalam Python yang ditandai dengan pola indentasi.

Fungsi (*function*) sendiri sesungguhnya merupakan blok kode program yang diberi nama, sehingga jika diperlukan dapat dipanggil berulang-ulang di dalam kode program. Fungsi mirip dengan variabel, dimana variabel menyimpan data, sedangkan fungsi menyimpan serangkaian instruksi dan data juga. Peran dari fungsi sangat vital di dalam kode program, sebab akan membuat kode program menjadi lebih efisien. Instruksi yang sama tidak perlu ditulis berulang-ulang. Sebuah fungsi dapat memiliki sejumlah parameter di dalamnya, yaitu nilai-nilai yang dijadikan input atau argumen pada saat fungsi tersebut dipanggil.

Di dalam Python, deklarasi dan definisi fungsi diawali dengan kata kunci `def`. Tentu saja, sebuah fungsi harus didefinisikan dan dieksekusi terlebih dahulu sebelum dipanggil. Penamaan fungsi memiliki aturan yang sama dengan aturan penamaan variabel. Hanya saja, nama fungsi pada umumnya menggunakan kata kerja, seperti `hitung_luas`, `verifikasi_ulang`, `ujikurasi`, dan sebagainya. Berikut adalah contoh penggunaan fungsi di dalam Python:

```
def hitung_luas_persegi(panjang, lebar):
    luas = panjang * lebar
    return luas

data_pjg = float(input('Input data panjang persegi: '))
data_lbr = float(input('Input data lebar persegi: '))

data_luas = hitung_luas_persegi(data_pjg, data_lbr)

print('Luas persegi adalah:', data_luas)
```

Data panjang persegi dan lebar persegi harus diinput manual oleh pengguna program, tekan **Enter** setiap selesai menginput. Jika pengguna menginput nilai **20** untuk panjang persegi dan **5** untuk lebar persegi, maka outputnya adalah:

```
Input data panjang persegi:
10
Input data lebar persegi:
5
Luas persegi adalah: 50.0
```

Fungsi `input` merupakan fungsi bawaan Python yang digunakan untuk meminta input nilai dari pengguna. Karena fungsi `input` akan memberikan nilai string, maka harus dikonversi menjadi numerik (misalnya `float`) untuk bisa dilibatkan dalam operasi matematika, seperti menghitung luas persegi sebagaimana kode di atas.

Perhatikan fungsi `hitung_luas_persegi(panjang, lebar)` di atas, `hitung_luas_persegi` merupakan nama fungsi, sementara `(panjang, lebar)` merupakan parameter fungsi. Parameter harus diisi dengan argumen atau nilai ketika fungsi dipanggil. Pada saat fungsi dipanggil, yaitu baris `data_luas = hitung_luas_persegi(data_pjg, data_lbr)`, kita menggunakan variabel `data_pjg` dan `data_lbr` sebagai argumen, dimana data kedua variabel ini kita minta dari pengguna program. Bagaimana jika pengguna tidak memberikan input argumen pada saat memanggil sebuah fungsi? Jawabannya, eksekusi program akan error. Akan tetapi, ada solusinya, yaitu dengan memberikan argumen bawaan (*default argument*). Contohnya seperti ini:

```
def hitung_luas_persegi(panjang=20, lebar=5):
```

Pada kode di atas, karena parameter `panjang` dan `lebar` diisi nilai bawaan, yaitu `20` dan `5`, maka ketika fungsi `hitung_luas_persegi` dipanggil tanpa argumen, maka argumen `20` dan `5` akan digunakan. Seperti ini lah cara memanggil fungsi tanpa argumen:

```
hitung_luas_persegi()
```

Jika kemudian fungsi dipanggil dengan argumen, misalnya:

```
hitung_luas_persegi(17, 8)
```

maka nilai argumen itu lah yang akan dijadikan input oleh fungsi untuk kalkulasi. Pemanggilan argumen juga dapat dilakukan secara eksplisit:

```
hitung_luas_persegi(lebar=12, panjang=32)
```

Perhatikan, ketika argumen dipanggil secara eksplisit, maka jika posisi argumennya yang tertukar (`lebar` dulu baru `panjang`) tidak akan menjadi masalah. Akan tetapi, jika argumen dipanggil secara implisit, yaitu tanpa menyebut nama parameternya terlebih dahulu, seperti `hitung_luas_persegi(17, 8)`, maka posisi argumen tidak boleh tertukar. Sebab angka pertama pasti akan dianggap `panjang`, dan yang kedua baru `lebar`, berapa pun nilai yang kita input. Kita harus berhati-hati dalam hal ini, sebab banyak potensi kesalahan terjadi di sini.

Kata kunci `return` di dalam fungsi berperan di dalam mengembalikan atau memberikan nilai yang dihasilkan oleh fungsi tersebut. Jika kita menginginkan fungsi memberikan nilai keluar, sebagaimana hasil perhitungan luas persegi di atas, maka nilai yang mau dikeluarkan harus di-return. Kalau nilai yang mau dikeluarkan dari fungsi adalah luas, maka kodennya adalah `return luas`. Di dalam Python, sebuah fungsi juga dibolehkan tanpa nilai `return`.

### Fungsi dengan Jumlah Argumen Bebas

Perhatikan kembali definisi fungsi `def hitung_luas_persegi(panjang, lebar)`. Fungsi seperti ini memberikan jumlah parameter tetap (*fixed*), yaitu 2 parameter, `panjang` dan `lebar`. Meskipun keduanya atau salah satunya dapat diisi dengan argumen bawaan. Pertanyaannya, bagaimana jika kita ingin mendefinisikan sebuah fungsi dengan jumlah argumen bebas, berapa pun yang kita inginkan sesuai keperluan? Jawabannya adalah dengan membuat parameter bertanda `*` (tanda bintang tunggal) dan `**` (tanda bintang ganda).

## Bab II Fundamental Python

Tanda bintang tunggal menunjukkan *non-keyword argument*, dan tanda bintang ganda menunjukkan *keyword argument*. Berikut adalah contoh penggunaan kedua jenis parameter ini:

```
def list_surveyor(*args):
    for nama in args:
        print(nama)

list_surveyor('Bambang', 'Rojali', 'Joni', 'Santoso', 'Udin')
```

Output:

```
Bambang
Rojali
Joni
Santoso
Udin
```

Perhatikan bahwa `list_surveyor` merupakan fungsi yang memiliki parameter `*args`. Yang artinya dapat diisi argumen dengan jumlah bebas, atau tidak diisi sama sekali. Sehingga ketika fungsi ini dipanggil, jumlah argumennya bebas. Jika dipanggil seperti berikut ini:

```
list_surveyor('Bambang', 'Rojali')
```

maka outputnya adalah:

```
Bambang
Rojali
```

Perhatikan lagi fungsi berikut:

```
def list_surveyor(**kwargs):
    for jabatan, nama in kwargs.items():
        print("%s: %s" % (jabatan, nama))

list_surveyor(Korlap='Bambang', Asisten='Rojali', Anggota='Santoso')
```

Output:

```
Korlap: Bambang
Asisten: Rojali
Anggota: Santoso
```

Mirip dengan tanda bintang tunggal, fungsi tanda bintang ganda hanya menambahkan kata kunci pada argumen. Jika fungsinya dipanggil seperti berikut ini:

```
list_surveyor(korlap='Bambang', Asisten='Rojali', Konsumsi='Joni', Anggota='Santoso')
```

maka outputnya adalah:

```
Korlap: Bambang
Asisten: Rojali
Konsumsi: Joni
Anggota: Santoso
```

## Fungsi Lambda

Fungsi Lambda adalah sebuah fungsi kecil (hanya satu baris kode) yang tidak memiliki nama (*anonymous*). Sintaks dasarnya adalah sebagai berikut:

```
lambda parameters : expression
```

Contoh implementasi fungsi lambda dalam fungsi kuadrat matematika  $f(x) = x^2 + 3x + 5$ :

```
f = lambda x: x**2 + 3*x + 5          # Definisi fungsi lambda
x = 5
y = f(x)                                # Fungsi lambda dipanggil
print(y)
```

Output:

```
45
```

Jika parameter/argumennya lebih dari satu, fungsi lambda akan berbentuk seperti berikut:

```
model_regresi = lambda x1, x2, x3: 2*x1 + 3*x2 + 5*x3 + 10
x1 = 4
x2 = 8
x3 = 12
y = model_regresi(x1,x2,x3)
print(y)
```

Output:

```
102
```

Kode program di atas adalah implementasi model regresi linier berganda  $y = 2x_1 + 3x_2 + 5x_3 + 10$ .

## Pemrograman Berorientasi Objek

### Kelas dan Objek

Pada bagian sebelumnya kita sudah membahas tentang berbagai tipe data di dalam Python. Seperti variabel (numerik dan string), list, tuple, set, dan dictionary. Pertanyaannya, bagaimana jika kita ingin memproses data yang tipenya tidak masuk ke dalam salah satu tipe data yang sudah kita bahas? Misalnya citra satelit, shapefile, bahkan hingga data suara atau video. Bahasa pemrograman tentu saja didesain untuk menyelesaikan permasalahan kompleks di dunia nyata, tidak hanya sekedar membuat list teks atau operasi matematika. Untuk keperluan ini, bahasa pemrograman berorientasi objek seperti Python menyediakan fasilitas yang disebut kelas (*class*). Kelas digunakan untuk mendefinisikan tipe data atau objek baru. Singkatnya, kelas merupakan sebuah tipe data sebagaimana variabel numerik atau variabel string. Hanya saja, tipe data kelas ini kita definisikan sendiri. Dan data yang diinisiasi dengan kelas dikenal sebagai objek.

## Bab II Fundamental Python

Secara teknis, kelas merupakan teknik enkapsulasi variabel dan fungsi ke dalam satu entitas. Fungsi yang terdapat di dalam kelas sering disebut sebagai metode (*method*). Bentuk umum deklarasi kelas adalah sebagai berikut:

```
class NamaKelas:  
    variabel_satu = ...  
    variabel_dua = ...  
  
    def fungsi_satu():  
        ...  
        return ...  
  
    def fungsi_dua():  
        ...  
        return ...
```

Berikut adalah contoh deklarasi dan implementasi kelas:

```
class Kota:  
    def __init__(self, nama_kota, status_kota):  
        self.nama = nama_kota  
        self.status = status_kota  
  
    def info_kota(self):  
        print('Nama kota: ', self.nama)  
        print('Status kota: ', self.status)  
  
banjarbaru = Kota('Banjarbaru', 'Ibukota Provinsi Kalsel')  
banjarbaru.info_kota()
```

Output:

```
Nama kota: Banjarbaru  
Status kota: Ibukota Provinsi Kalsel
```

Pada contoh kode program di atas terdapat fungsi `__init__`. Fungsi `__init__` di dalam suatu kelas dikenal sebagai *constructor*. Yaitu suatu fungsi atau metode yang akan dipanggil secara otomatis ketika sebuah objek diinisiasi atau diciptakan dengan kelas tersebut. Sementara parameter `self` adalah sebuah referensi (dikenal juga sebagai *pointer*) ke objek yang dideklarasikan dengan kelas tersebut. Parameter ini digunakan oleh objek (*instance*) dari suatu kelas untuk mengakses variabel-variabel atau fungsi-fungsi yang dimiliki oleh kelas tersebut. Tanpa adanya `self`, objek yang dideklarasikan dengan suatu kelas tidak akan dapat mengakses variabel-variabel dan fungsi-fungsi yang ada di dalam kelas tersebut.

Untuk pemberian nama kelas sendiri, aturannya mirip dengan penamaan variabel. Hanya saja, kelas pada umumnya diberi nama berupa kata benda. Dan gaya penulisan nama kelas biasanya menggunakan *Pascal case*, yaitu gaya penulisan bahasa pemrograman Pascal. Dimana setiap awal kata menggunakan huruf kapital. Misalnya `DataGeospasial`, `Raster`, `SistemKoordinat`, dan sebagainya.

### Lingkup dan Siklus Hidup Variabel

Di dalam Python dan bahasa pemrograman lainnya, pada umumnya sebuah variabel hanya dapat diakses di dalam tempat dia didefinisikan. Variabel seperti ini dikenal sebagai variabel lokal. Sebuah variabel dapat bersifat lokal di dalam sebuah fungsi. Perhatikan contoh berikut:

```

def my_func():
    nama = 'James T. Kirk'
    print(f'Hello, {nama}')

my_func()

print(f'Kamu tinggal dimana, {nama}?'')
```

Output:

```

Hello, James T. Kirk
Traceback (most recent call last):
  File "main.py", line 7, in <module>
    print(f'Kamu tinggal dimana, {nama}?')
NameError: name 'nama' is not defined
```

Pada kode di atas, variabel `nama` didefinisikan di dalam fungsi `my_func`, sehingga variabel `nama` bersifat lokal dan hanya dapat diakses dari dalam fungsi `my_func`. Ketika variabel `nama` dicoba untuk diakses di luar fungsi `my_func`, hasilnya adalah error. Jika variabel `nama` diletakkan di luar fungsi `my_func`, maka variabel `nama` akan menjadi variabel global dan dapat diakses dari mana saja di dalam kode program, termasuk oleh fungsi `my_func`. Contohnya sebagai berikut:

```

nama = 'James T. Kirk'

def my_func():
    print(f'Hello, {nama}')

my_func()

print(f'Kamu tinggal dimana, {nama}?')
```

Output:

```

Hello, James T. Kirk
Kamu tinggal dimana, James T. Kirk?
```

Sebuah variabel lokal lingkupnya hanya lokal, dan variabel global lingkupnya global di dalam keseluruhan kode program tersebut. Demikian juga dengan siklus hidup variabelnya. Untuk variabel lokal di dalam fungsi, variabel tersebut akan musnah jika eksekusi fungsi sudah berakhir. Sementara variabel global akan tetap ada dan dapat diakses selama eksekusi kode program belum berakhir.

Meskipun Python, dan bahasa pemrograman lainnya menyediakan fasilitas untuk definisi variabel global. Akan tetapi, pembuatan variabel global sebenarnya merupakan sebuah teknik yang buruk di dalam pemrograman. Sebab jika kode program kita cukup besar, nantinya berpotensi akan terjadi konflik di dalam penamaan identitas variabel. Disamping variabel global akan menempati ruang di dalam memori sepanjang eksekusi program belum berakhir. Tentu saja hal ini akan menjadikan program tidak efisien, sebab boros memori komputer. Jadi, sebaiknya selalu terapkan teknik variabel lokal di dalam fungsi, dan gunakan variabel global hanya jika benar-benar diperlukan atau ketika tidak ada opsi lain.

Sebuah pertanyaan mungkin muncul, bagaimana jika terdapat variabel lokal yang namanya sama persis dengan variabel global? Perhatikan contoh berikut:

## Bab II Fundamental Python

```
nama = 'James T. Kirk'

def my_func():
    nama = 'Leonard McCoy'
    print(f'Hello, {nama}')

my_func()

print(f'Kamu tinggal dimana, {nama}?')
```

Output:

```
Hello, Leonard McCoy
Kamu tinggal dimana, James T. Kirk?
```

Perhatikan output contoh kode program di atas. Variabel global `nama` dan variabel lokal `nama` di dalam fungsi `my_func` tidak saling mempengaruhi. Masing-masing berdiri sendiri. Akibatnya, `nama` yang dipanggil di outputnya menjadi berbeda. Jika kita ingin merubah nilai dari variabel global `nama` dari dalam fungsi `my_func`, kita harus meletakkan kata kunci `global` di depan variabel lokal `nama` yang ada di dalam fungsi `my_func`, sebelum nilainya diisi ulang. Perhatikan contoh berikut:

```
nama = 'James T. Kirk'

def my_func():
    global nama
    nama = 'Leonard McCoy'
    print(f'Hello, {nama}')

my_func()

print(f'Kamu tinggal dimana, {nama}?')
```

Output:

```
Hello, Leonard McCoy
Kamu tinggal dimana, Leonard McCoy?
```

Pada contoh kode di atas, dengan penggunaan kata kunci `global`, nilai dari variabel global `nama` diisi dengan nilai baru dari dalam fungsi `my_func`. Tentu saja, nilai asalnya semula menjadi tertimpa atau terhapus, dan digantikan dengan nilai yang baru.

Terdapat sebuah kata kunci lagi yang dapat merubah akses ke lingkup variabel yang berbeda, yaitu `nonlocal`. Kata kunci `nonlocal` biasanya digunakan ketika ada fungsi di dalam fungsi. Perhatikan contoh kode berikut:

```
def fungsi_ayah():
    nama = 'Si Ayah'

    def fungsi_anak():
        nonlocal nama
        nama = 'Si Anak'

    fungsi_anak()

    print(nama)

fungsi_ayah()
```

Output:

```
Si Anak
```

Kata kunci `nonlocal` di dalam `fungsi_anak` akan merubah atau menimpa nilai variabel dengan nama sama di dalam fungsi di atasnya, yaitu `fungsi_ayah`.

### Kata Kunci `del`

Sebuah variabel atau sebuah objek akan menempati ruang di dalam RAM (*Random Access Memory*) komputer selama variabel atau objek tersebut belum habis siklus hidupnya. Jika suatu saat sebuah variabel atau sebuah objek sudah tidak diperlukan lagi eksistensinya di dalam program, meskipun siklus hidupnya masih belum habis, kita dapat menghapusnya secara manual. Terdapat sebuah kata kunci yang dapat digunakan untuk menghapus variabel atau objek di dalam kode program Python, yaitu `del`. `del` akan menghapus suatu variabel atau objek, meskipun siklus hidup variabel atau objek tersebut sebenarnya belum berakhir. Tentu saja, sebuah variabel atau sebuah objek hanya akan kita hapus dengan sengaja jika kehadirannya memang sudah tidak diperlukan lagi. Di dalam pemrosesan citra digital, hal ini pada umumnya dilakukan untuk mengosongkan ruang di dalam memori. Tentu saja, tujuannya adalah untuk manajemen memori komputer. Sehingga kode program yang dihasilkan nantinya dapat berjalan secara lebih efisien atau hemat memori, serta mencegah kode program atau komputer macet akibat memori penuh.

Perhatikan contoh kode program berikut:

```
citra_satelit = 'Ini adalah sebuah citra satelit berukuran 5 GB'
print(citra_satelit)
del citra_satelit
print(citra_satelit)
```

Output:

```
Ini adalah sebuah citra satelit berukuran 5 GB
Traceback (most recent call last):
  File "main.py", line 7, in <module>
    print(citra_satelit)
NameError: name 'citra_satelit' is not defined
```

Kode program di atas adalah sebuah ilustrasi ketika Anda memproses sebuah citra satelit berukuran 5 GB menggunakan Python. Bayangkan jika RAM laptop Anda hanya 8 GB. Tentu saja, akan ada potensi memori penuh dan kode program macet. Sebab sisa ruang memori yang 3 GB pasti sudah dialokasikan oleh laptop untuk perangkat lunak lainnya, misalnya sistem operasi, browser, dan sebagainya. Jika memang objek `citra_satelit` sudah selesai digunakan dan tidak diperlukan lagi, kita dapat menghapusnya menggunakan `del`. Setelah objek `citra_satelit` dihapus menggunakan `del`, objek `citra_satelit` sudah tidak ada lagi di dalam kode program. Jika kita mencoba mengakses `citra_satelit` setelah dihapus, hasilnya adalah error. `del` juga dapat digunakan untuk menghapus beberapa variabel atau objek sekaligus, dengan cara berikut:

```
del sentinel2, landsat9, srtm, shp_admin
```

## Destruktor `__del__`

Python memiliki sebuah metode khusus di dalam kelas yang akan dijalankan secara otomatis ketika sebuah objek dihapus atau berakhir siklus hidupnya. Fungsi tersebut adalah *destructor* `__del__`. `__del__` merupakan kebalikan dari `__init__` yang sudah kita bahas pada bagian terdahulu, dimana `__init__` merupakan konstruktur. Destruktor sebenarnya tidak terlalu diperlukan di dalam Python, tidak sebagaimana bahasa pemrograman lain seperti C++. Hal ini karena Python memiliki *garbage collector* yang berfungsi untuk menangani manajemen memori komputer secara otomatis ketika kode Python dieksekusi. Pada umumnya, destruktur di dalam Python hanya perlu didefinisikan secara manual ketika kita ingin mengambil alih sebagian kendali manajemen memori di dalam Python.

Perhatikan contoh penggunaan destruktur pada kode program berikut:

```
class CitraSatelit:
    # Konstruktur
    def __init__(self):
        print('Citra satelit dimuat ke dalam program.')

    # Destruktor
    def __del__(self):
        print('Citra satelit dihapus dari dalam program.')
sentinel2 = CitraSatelit()
del sentinel2
```

Output:

```
Citra satelit dimuat ke dalam program.
Citra satelit dihapus dari dalam program.
```

Pada contoh kode program di atas, konstruktur `__init__` akan dipanggil secara otomatis ketika sebuah objek, yaitu `sentinel2`, diciptakan dengan kelas `CitraSatelit`. Dan destruktur `__del__` akan dipanggil secara otomatis ketika objek dari kelas `CitraSatelit`, yaitu `sentinel2`, dihapus menggunakan `del`. Sebenarnya, tanpa dihapus menggunakan `del` pun destruktur `__del__` akan dipanggil secara otomatis, yaitu ketika siklus hidup objek `sentinel2` sudah berakhir. Pada prinsipnya, destruktur di dalam Python hanya perlu didefinisikan secara eksplisit jika kita ingin memerintahkan program untuk melakukan sesuatu, ketika sebuah objek yang diinisialisasi dengan suatu kelas dihapus atau berakhir siklus hidupnya di dalam program.

## Variabel Privat dan Fungsi Privat

Variabel privat dan fungsi privat merupakan suatu variabel dan fungsi yang hanya dapat diakses di dalam kelas atau fungsi tempat dimana dia dideklarasikan. Sehingga variabel dan fungsi privat tidak dapat diakses dari luar lingkup kelas atau fungsi yang mengenkapsulasinya. Variabel privat dan fungsi privat ini bertujuan untuk menyembunyikan data atau fungsi, agar hanya kelas dan fungsi pemiliknya yang dapat mengakses atau mengeksekusi. Sekaligus untuk menghindari konflik dengan variabel-variabel atau fungsi-fungsi di luarnya.

Di dalam Python, variabel atau fungsi privat dinotasikan dengan dua garis bawah di depan nama variabel atau fungsinya. Contoh penggunaan variabel privat:

```

class Kota:
    __provinsi = 'Kalimantan Selatan'
    def __init__(self, nama_kota, status_kota):
        self.nama = nama_kota
        self.status = status_kota
    def info_kota(self):
        print('Nama kota: ', self.nama)
        print('Status kota: ', self.status, self.__provinsi)
banjarbaru = Kota('Banjarbaru', 'Ibukota Provinsi')
banjarbaru.info_kota()

```

Output:

```

Nama kota: Banjarbaru
Status kota: Ibukota Provinsi Kalimantan Selatan

```

Pada kode di atas, `__provinsi` merupakan variabel privat yang hanya dapat diakses oleh kelas pemiliknya, yaitu kelas `Kota`. Jika objek `banjarbaru` mencoba mengakses variabel `__provinsi`, misalnya:

```
print('Nama provinsi: ', banjarbaru.__provinsi)
```

hasilnya akan error seperti berikut:

```

Traceback (most recent call last):
  File "main.py", line 15, in <module>
    print('Nama provinsi: ', banjarbaru.__provinsi)
AttributeError: 'Kota' object has no attribute '__provinsi'

```

Informasi '`Kota` object has no attribute `'__provinsi'`' menginformasikan bahwa objek yang diinisiasi dengan kelas `Kota`, yaitu `banjarbaru`, tidak memiliki atribut `__provinsi`.

### Kata Kunci `pass`

Kata kunci `pass` berfungsi seperti instruksi “*free memory*” di dalam gladi resik acara wisuda universitas yang rencananya akan berlangsung besok hari. Dengan kata lain, kata ini berfungsi untuk melewatkannya yang nantinya akan diisi di masa yang akan datang. Biasanya hal ini dilakukan ketika kita masih dalam tahap membuat semacam *framework* atau kerangka dari kode program. `pass` dapat diletakkan di dalam fungsi/metode atau kelas. Hal yang pasti adalah `pass` tidak akan memberikan output, fungsinya hanya untuk menghindari error ketika fungsi/metode atau kelas tersebut digunakan di dalam kode program. Contoh penggunaan kata kunci `pass` adalah sebagai berikut:

```

def my_function():
    pass

my_function()

```

Kode di atas akan berjalan normal dan tidak terjadi error, tetapi tidak akan memberikan output apapun. Akan tetapi, jika `pass` dihapus, ketika `my_function` dipanggil akan terjadi error.

## Metode Objek, Metode Kelas, dan Metode Statik

Pada bagian sebelumnya kita sudah membahas tentang metode, yaitu suatu fungsi di dalam kelas. Metode-metode yang sudah kita buat sebelumnya merupakan metode objek atau *instance method*, atau dikenal juga sebagai *regular/plain method*. Metode ini biasanya memiliki parameter `self` di dalamnya. Metode objek diakses oleh objek yang dideklarasikan oleh suatu kelas, dan sifatnya unik untuk setiap objek. Selain metode objek, di dalam Python juga terdapat metode kelas (*class method*) dan metode statik (*static method*). Metode kelas didefinisikan dengan dekorator `@classmethod` dan metode statik didefinisikan dengan dekorator `@staticmethod`. Perhatikan template kodennya sebagai berikut:

```
class Kelassaya:  
    def metode_objek(self):  
        ...  
  
    @classmethod  
    def metode_kelas(cls):  
        ...  
  
    @staticmethod  
    def metode_statik():  
        ...
```

Metode kelas memiliki parameter `cls` di dalamnya, dimana `cls` ini merupakan pointer ke kelas itu sendiri. Berbeda dengan `self` yang merupakan pointer ke objek yang didefinisikan dengan kelas tersebut. Lalu dimana letak perbedaan metode objek, metode kelas, dan metode statik? Metode objek dapat mengakses atau diakses oleh masing-masing objek yang didefinisikan oleh kelas tersebut dengan menggunakan parameter `self`. Metode kelas dapat dijalankan secara langsung oleh kelasnya. Metode kelas tidak dapat mengakses atau diakses oleh objeknya, tetapi dia dapat mengakses atau diakses oleh kelasnya sendiri dengan menggunakan parameter `cls`. Metode statik pada dasarnya tidak memiliki ikatan dengan kelas atau pun objeknya. Akan tetapi, metode statik didefinisikan di dalam kelas dan dipanggil menggunakan kelas tersebut, sebagaimana metode kelas. Metode statik sebenarnya lebih mirip fungsi biasa yang berdiri sendiri. Akan tetapi, metode statik biasanya disimpan ke dalam sebuah kelas untuk tujuan pengelompokan fungsi-fungsi yang mirip sehingga memudahkan penggunaan.

## Metode Ajaib

*Magic method* atau metode ajaib di dalam Python merupakan fungsi atau metode khusus yang namanya diawali dan diakhiri dengan *double underscore* (garis bawah ganda). *Magic method* sendiri tidak dirancang untuk kita panggil secara langsung, akan tetapi ia akan dipanggil oleh sistem secara internal pada saat-saat tertentu. Di dalam python, *magic method* juga dinamakan dengan *dunder method*. Dinamakan “*dunder*” karena dia merupakan singkatan dari “*double underscore*” (<https://www.tutorialsteacher.com>).

Konstruktor `__init__` yang ada di dalam kelas merupakan salah satu *magic method*. “Keajaiban” dari metode atau fungsi `__init__` adalah dia dipanggil secara otomatis ketika sebuah objek dideklarasikan dengan kelas tersebut. Selain `__init__` masih banyak metode-metode ajaib lainnya. `print` yang sudah sering kita gunakan sebelumnya, dan akan kita gunakan terus dalam pemrograman Python, pada dasarnya adalah sebuah fungsi. Untuk mengetahui *magic methods* yang ada di dalam `print`, ketikkan kode berikut:

```
print(dir(print))
```

Output:

```
['__call__', '__class__', '__delattr__', '__dir__', '__doc__', '__eq__',  
 '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__', '__init__',  
 '__init_subclass__', '__le__', '__lt__', '__module__', '__name__', '__ne__',  
 '__new__', '__qualname__', '__reduce__', '__reduce_ex__', '__repr__', '__self__',  
 '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__text_signature__']
```

Output dari kode di atas merupakan daftar *magic methods* yang dimiliki oleh `print`. Yang pasti, *magic methods* tidak dapat kita panggil sebagaimana kita panggil seperti metode-metode atau fungsi-fungsi lainnya, *magic methods* khusus dipanggil oleh sistem untuk menjalankan tugas-tugas tertentu. Ketika kita membuat dan menjalankan kode-kode program dengan Bahasa Python, secara tidak sadar kita sebenarnya sering menjalankan *magic methods* di balik layar. Sebagaimana ketika kita menjalankan fungsi `print` untuk menampilkan teks ke layar.

## Pewarisan

Pewarisan (*inheritance*) merupakan mekanisme yang disediakan oleh bahasa pemrograman berorientasi objek, agar suatu kelas dapat mengambil variabel atau fungsi yang dimiliki oleh sebuah kelas. Sebagai contoh, kita akan mendeklarasikan sebuah kelas dengan nama `Raster`. Kita dapat mendeklarasikan sebuah kelas baru, misalnya dengan nama `JPG`, akan tetapi kelas `JPG` ini akan mewarisi data dari kelas `Raster`.

```
class Raster:  
    def struktur(self):  
        print('Tersusun atas pixel.')  
  
class JPG(Raster):  
    def jlh_band(self):  
        print('Maksimum 3 band.')  
  
gambar = JPG()  
gambar.struktur()  
gambar.jlh_band()
```

Output:

```
Tersusun atas pixel.  
Maksimum 3 band.
```

Pada kode di atas, `class JPG(Raster)` merupakan instruksi agar kelas `JPG` mewarisi kelas `Raster`. Meskipun *child class*, yaitu `JPG` tidak memiliki fungsi `struktur`, akan tetapi `JPG` dapat mengakses fungsi `struktur`. Sebab kelas `JPG` mewarisi kelas dari *parent class*-nya, yaitu `Raster`. Tentu saja, konsep pewarisan ini digunakan untuk efisiensi penulisan kode program. Sebab jika misalnya sudah ada definisi kelas `Raster`, yang berisi ciri-ciri data raster secara umum. Selanjutnya, untuk mendefinisikan kelas `JPG`, kita tidak perlu lagi membuat ciri-ciri detail sebuah data raster secara umum. Sebab `JPG` adalah salah satu jenis data raster, yang pastinya memiliki karakteristik umum sebagai data raster, yaitu tersusun atas pixel. Kita tinggal mendefinisikan kelas `JPG` dengan mewarisi kelas `Raster`. Selanjutnya, di dalam kelas `JPG` kita masukkan ciri-ciri khusus gambar JPEG yang memang tidak ditemukan di data raster lain.

## Bab II Fundamental Python

Lebih jauh, di dalam Python ada istilah *multiple inheritance* atau pewarisan jamak, dan *multilevel inheritance* atau pewarisan bertingkat. Pewarisan jamak berarti satu kelas mewarisi lebih dari satu kelas. Pewarisan bertingkat berarti suatu kelas mewarisi kelas lain, dimana kelas lain yang diwarisinya juga mewarisi kelas lainnya di atasnya. Sebagai contoh pewarisan jamak, misalnya terdapat sebuah definisi kelas **DataGeospasial**, kemudian terdapat lagi sebuah definisi kelas **Raster**. Jika nantinya ada sebuah definisi kelas **GeoTIFF**, maka kelas **GeoTIFF** akan mewarisi kedua kelas tersebut. Sebab kita tahu bahwa GeoTIFF adalah data raster yang memiliki atribut geospasial, yaitu koordinat. Pewarisan jamak sebenarnya cukup dihindari di dalam pemrograman. Sebab akan menyebabkan kode program menjadi sangat kompleks. Beberapa bahasa, seperti C# dan Java, tidak mendukung pewarisan jamak. Bentuk definisi pewarisan jamak kelas **GeoTIFF** dari kelas **DataGeospasial** dan **Raster** adalah sebagai berikut:

```
class DataGeospasial:
    def info_geo(self):
        print('Punya koordinat.')

class Raster:
    def info_raster(self):
        print('Tersusun atas pixel.')

class GeoTIFF(DataGeospasial, Raster):
    def __init__(self, nama):
        self.nama = nama
        print(f'Nama citra: {self.nama}.')

    def info_geotiff(self):
        print('Raster multiband.')

19 = GeoTIFF('Landsat 9')
19.info_geo()
19.info_raster()
19.info_geotiff()
```

Output:

```
Nama citra: Landsat 9.
Punya koordinat.
Tersusun atas pixel.
Raster multiband.
```

Bentuk pewarisan bertingkat adalah sebagaimana contoh berikut:

```
class DataGeospasial:
    def info_geo(self):
        print('Punya koordinat.')

class Vektor(DataGeospasial):
    def info_vektor(self):
        print('Tersusun atas verteks.')

class Poligon(Vektor):
    def __init__(self, nama):
        self.nama = nama
        print(f'Nama data: {self.nama}.')

    def info_poligon(self):
        print('Memiliki atribut luas.')

polygon_administrasi = Poligon('Poligon Administrasi')

polygon_administrasi.info_geo()
polygon_administrasi.info_vektor()
polygon_administrasi.info_poligon()
```

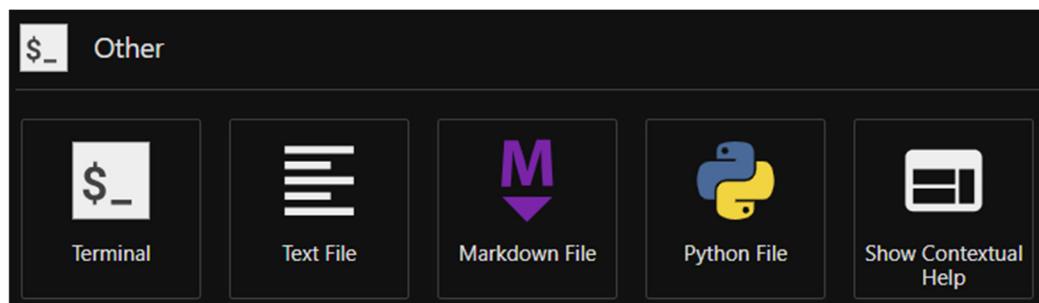
Output:

```
Nama data: Poligon Administrasi.
Punya koordinat.
Tersusun atas verteks.
Memiliki atribut luas.
```

Pada contoh kode di atas, logikanya adalah kelas **Poligon** merupakan turunan dari kelas **Vektor**, dan kelas **Vektor** merupakan turunan dari kelas **DataGeospasial**. Tentu saja, kita dapat menurunkan lagi kelas-kelas lainnya, kelas **Garis** dan kelas **Titik** misalnya, dimana keduanya mewarisi kelas **Vektor**.

## Modul

Agar dapat digunakan seterusnya di file-file kode Python yang lain, definisi kelas dapat disimpan di dalam file tersendiri. Di dalam Python, hal ini dikenal sebagai modul (*module*). Modul pada dasarnya adalah sebuah file berekstensi **.py** atau **.pyc** (*compiled Python*), yang berisi kode-kode Python berupa kelas, fungsi, atau variabel. Modul bertujuan agar kode program yang sama tidak perlu harus ditulis berulang-ulang jika suatu saat nanti diperlukan. Sebagai latihan, buka JupyterLab, kemudian klik Python File. Sebagaimana terlihat pada gambar berikut.



Ganti nama file Python yang baru kita buat dengan nama **Geodesi.py**. Kemudian di dalam file **Geodesi.py** ketikkan kode berikut:

```
import math

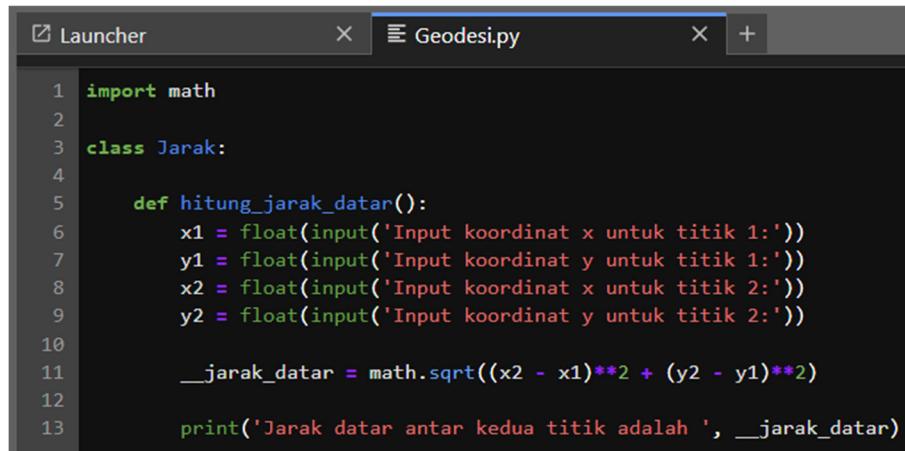
class Jarak:

    def hitung_jarak_datar():
        x1 = float(input('Input koordinat x untuk titik 1:'))
        y1 = float(input('Input koordinat y untuk titik 1:'))
        x2 = float(input('Input koordinat x untuk titik 2:'))
        y2 = float(input('Input koordinat y untuk titik 2:'))

        jarak_datar = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)

        print('Jarak datar antar kedua titik adalah ', jarak_datar)
```

Sebagaimana gambar berikut:

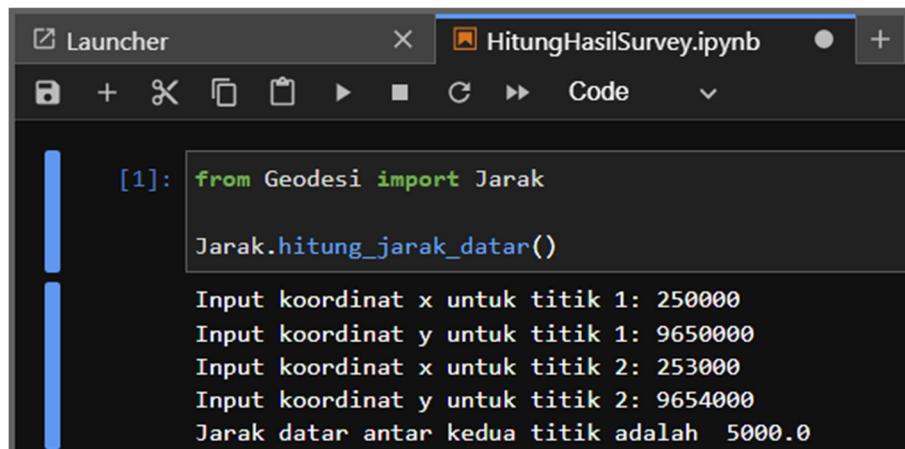


```
1 import math
2
3 class Jarak:
4
5     def hitung_jarak_datar():
6         x1 = float(input('Input koordinat x untuk titik 1:'))
7         y1 = float(input('Input koordinat y untuk titik 1:'))
8         x2 = float(input('Input koordinat x untuk titik 2:'))
9         y2 = float(input('Input koordinat y untuk titik 2:'))
10
11         jarak_datar = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
12
13         print('Jarak datar antar kedua titik adalah ', jarak_datar)
```

Simpan dan tutup file **Geodesi.py**, kemudian buat sebuah notebook baru. Beri nama misalnya **HitungHasilSurvey.ipynb**. Kemudian di dalam notebook baru tersebut ketikkan kode berikut:

```
from Geodesi import Jarak
Jarak.hitung_jarak_datar()
```

dan jalankan. Outputnya akan tampak seperti pada gambar berikut:



```
[1]: from Geodesi import Jarak
Jarak.hitung_jarak_datar()

Input koordinat x untuk titik 1: 250000
Input koordinat y untuk titik 1: 9650000
Input koordinat x untuk titik 2: 253000
Input koordinat y untuk titik 2: 9654000
Jarak datar antar kedua titik adalah 5000.0
```

Angka-angka koordinat harus diinput manual oleh pengguna ketika kode program dieksekusi. Dan sebagai catatan, di dalam file Python **Geodesi.py** dapat didefinisikan banyak kelas. Misalnya **Lereng**, **Luas**, **Elevasi**, dan sebagainya, dengan fungsi/metodenya masing-masing.

---

*Ketika kita mengimport paket-paket Python dalam kode program kita nantinya, misalnya import math, import numpy, import scikit-learn, import tensorflow, from osgeo import gdal, dan sebagainya, pada dasarnya kita mengimport kelas-kelas atau fungsi-fungsi/metode-metode yang sudah dibuatkan oleh orang lain.*

---

## Import Paket Python

Pada bagian terdahulu kita sudah membahas tentang instalasi paket Python menggunakan Anaconda. Paket Python pada dasarnya berisi modul-modul, kelas-kelas, dan fungsi-fungsi yang sudah siap untuk dipakai. Ketika sebuah paket sudah diinstal, ketika akan digunakan di dalam kode program Python, paket tersebut harus diimport terlebih dahulu. Kata kunci import di dalam Python digunakan untuk mengimport paket ke dalam kode program Python. Untuk Google Colab, paket-paket standar Python pada umumnya sudah terintegrasi, sehingga tidak perlu diinstal lagi. Terkecuali paket-paket spesifik yang memang kurang umum digunakan. Perhatikan contoh-contoh cara mengimport paket Python berikut:

```
import pandas
```

Instruksi di atas akan mengimport Pandas ke dalam kode Python.

```
import pandas as pd
```

Instruksi di atas akan mengimport Pandas sekaligus membuat semacam nama alias atau singkatan nama untuk Pandas. Sehingga ketika Pandas akan digunakan nantinya kita tidak perlu mengetikkan `pandas`, cukup `pd` saja.

```
import math, numpy, pandas
```

Instruksi di atas digunakan untuk mengimport lebih dari satu paket sekaligus.

```
from sklearn.linear_model import LinearRegression
```

atau

```
from sklearn.linear_model import LinearRegression as LR
```

Instruksi di atas digunakan untuk mengimport secara langsung kelas `LinearRegression` dari paket Scikit-Learn. Sehingga ketika kita melakukan analisis regresi linier menggunakan Scikit-Learn, kita cukup mengetikkan `LinearRegression` atau `LR` saja. Jika seandainya pada saat impor paket regresi linier Scikit-Learn kita hanya melakukan `import sklearn.linear_model.LinearRegression` saja, maka setiap kali kita melakukan analisis regresi linier di dalam Python, kita harus mengetikkan kode `sklearn.linear_model.LinearRegression`. Tentu saja, hal ini membuat penulisan kode program menjadi tidak efisien. Kita juga dapat mengimport langsung fungsi yang ada di dalam suatu kelas dari suatu paket Python. Perhatikan contoh kode berikut:

```
from sklearn.linear_model.LinearRegression import predict
```

`predict` merupakan sebuah fungsi atau metode yang tersimpan di dalam kelas `LinearRegression` dari paket Scikit-Learn. Dengan mengimport metode `predict` seperti itu, kita tidak perlu lagi menggunakan kode `LinearRegression.predict`. Tetapi cukup mengetikkan `predict` saja.

## Penanganan Kesalahan dan Pengecualian

Perhatikan kembali contoh kode program hitung jarak datar di atas. Bagaimana jika seandainya pengguna program menginput data yang bukan angka? Misalnya pengguna memasukkan teks/string `abcdefg`. Tentu saja, akan muncul error, dan eksekusi program akan terhenti. Di dalam pemrograman, kondisi seperti ini tentu saja tidak dikehendaki. Bayangkan, dalam sebuah form registrasi online, pendaftar salah format dalam memasukkan alamat email, akibatnya formulir online atau laman webnya menjadi macet atau *crash*. Tentu saja, ini bukan merupakan teknik pemrograman yang baik. Sebab setiap potensi kesalahan yang muncul harusnya dapat diantisipasi dan kode program harusnya memberikan reaksi yang lebih baik dan lebih informatif. Di sini lah diperlukan teknik *try and exception*.

Perhatikan contoh berikut:

```
def fungsiku():
    try:
        varaiabelku = float(input('Input sembarang data numerik:'))
        print('OK. Terima kasih atas kerjasamanya... ')
    except:
        print('Tolong masukkan hanya data numerik.')
fungsiuk()
```

Output:

```
Input sembarang data numerik:
5
OK. Terima kasih atas kerjasamanya...
```

Output alternatif:

```
Input sembarang data numerik:
abc
Tolong masukkan hanya data numerik.
```

Pada kode di atas, ketika kode dijalankan dan pengguna menginput data sesuai permintaan, yaitu data numerik (`5`), maka kode di bawah `try` akan dieksekusi. Jika pengguna menginput selain data numerik (`abc`), maka akan terjadi error dan kode di bawah `except` akan dieksekusi.

Kode di atas dapat juga ditulis sebagai berikut (dan akan memberikan efek yang sama persis):

```
def fungsiku():
    try:
        varaiabelku = float(input('Input sembarang data numerik:'))
    except:
        print('Tolong masukkan hanya data numerik.')
    else:
        print('OK. Terima kasih atas kerjasamanya... ')
fungsiuk()
```

Pada kode di atas, instruksi `print('OK. Terima kasih atas kerjasamanya...')` ditaruh di bawah `else`. Dimana kode di bawah `else` hanya akan dieksekusi jika tidak terjadi error. Jika terjadi error, kode di bawah `else` tidak akan dieksekusi, kode di bawah `except` yang akan dieksekusi.

Berikut adalah contoh lain penggunaan **try** dan **except**:

```
def fungsiku():
    try:
        varaiabelku = float(input('Input sembarang data numerik:'))
    except:
        print('Tolong masukkan hanya data numerik.')
    else:
        print('OK. Terima kasih atas kerjasamanya... ')
    finally:
        print('Eksekusi program diakhiri.')

fungsku()
```

Output:

```
Input sembarang data numerik:
abc
Tolong masukkan hanya data numerik.
Eksekusi program diakhiri.
```

Pada kode di atas, tidak peduli apakah input dari pengguna salah atau benar, instruksi di bawah **finally** akan dieksekusi.

### Kata Kunci **with**

Kata kunci **with** digunakan untuk menyederhanakan blok penanganan pengecualian (*exception handling*). Biasanya hal ini dilakukan ketika kita mengakses (baik membaca maupun menulis) sebuah file. Perhatikan contoh kode program yang diambil dan dimodifikasi dari laman <https://www.geeksforgeeks.org> berikut:

```
# (1) Tanpa penanganan pengecualian
# dan tanpa menggunakan kata kunci with
file = open('File_path', 'w')
file.write('Hello Earth!')
file.close()

# (2) Penanganan pengecualian menggunakan try finally
file = open('File_path', 'w')
try:
    file.write('Hello Earth!')
finally:
    file.close()

# (3) Penanganan pengecualian menggunakan with
with open('File_path', 'w') as file:
    file.write('Hello Earth!')
```

Kode di atas hanya sebagai contoh, sehingga tidak akan memberikan output apa-apa, dikarenakan path dan filenya hanya simulasi. Kode nomor (1) merupakan contoh yang tidak direkomendasikan di dalam mengakses sebuah file. Sebab tanpa penanganan pengecualian, sehingga jika filenya tidak ada atau terjadi kesalahan di dalam penulisan path/nama file maka kode program akan berpotensi crash. Kode nomor (2) merupakan teknik yang baik di dalam mengakses sebuah file, sebab menggunakan metode **try** dan **finally** yang sudah dijelaskan sebelumnya. Kode nomor (3) merupakan bentuk penyederhanaan dari kode nomor (2), yaitu mengganti **try** dan **finally** dengan kata kunci **with**. Dengan kata lain, kode nomor (3) dan nomor (2) akan memberikan output yang sama persis.

## C. NumPy

NumPy (*Numerical Python*) (Harris et al., 2020) merupakan paket dasar yang digunakan untuk komputasi saintifik di lingkungan Python. Bahasa Python secara bawaan tidak memiliki fasilitas array. Sehingga NumPy di dalam Python akan berfungsi sebagai fasilitas untuk manajemen array berikut analisis-analisis yang terkait. Tentu saja, karena citra digital penginderaan jauh pada dasarnya merupakan sebuah array data numerik, maka NumPy memegang peranan yang sangat penting di dalam pemrosesan citra penginderaan jauh. Terutama untuk manajemen pixel-pixel citra. Di lingkungan Anaconda Prompt, NumPy dapat diinstal dengan instruksi berikut:

```
conda install anaconda::numpy
```

Sementara di lingkungan Google Colab, paket NumPy sudah tersedia dan dapat langsung dipanggil ke dalam kode program Python.

### Konstruksi dan Akses NumPy Array

Perhatikan contoh kode Python berikut:

```
import numpy as np  
  
my_array = np.array([1, 2, 3, 4, 5])  
  
print(my_array)  
print(my_array[3])  
print(my_array[2:4])
```

Output:

```
[1 2 3 4 5]  
4  
[3 4]
```

NumPy array menyerupai list, baik proses konstruksinya maupun cara mengakses elemen-elemen di dalamnya. Perbedaannya dengan list adalah, bahwa NumPy array hanya dapat memuat data dengan tipe sama, misalnya integer. Jika kita memasukkan data dengan tipe yang berbeda, misalnya terdapat string dan integer, NumPy akan mengkonversi datanya agar tipe data dalam satu array sama. List (dan juga tuple) dapat dikonversi menjadi NumPy array. Tentu saja, sebagai fasilitas utama untuk array data numerik di dalam Python, NumPy sangat mendukung multidimensional array. Perhatikan contoh kode program berikut:

```
import numpy as np  
  
band_1 = [[10, 20, 30], [40, 50, 60], [70, 80, 90], [90, 80, 70]]  
band_2 = [[10, 12, 13], [14, 15, 16], [17, 18, 19], [19, 18, 17]]  
band_3 = [[11, 21, 31], [41, 51, 61], [71, 81, 91], [91, 81, 71]]  
  
citra = np.array([band_1, band_2, band_3])  
  
print(f'Dimensi citra: {citra.shape}')  
print(f'Jumlah band citra: {citra.shape[0]}')  
print(f'Jumlah baris citra: {citra.shape[1]}')  
print(f'Jumlah kolom citra: {citra.shape[2]}')  
print('Nilai-nilai pixel citra:')  
print(citra)
```

Output:

```
Dimensi citra: (3, 4, 3)
Jumlah band citra: 3
Jumlah baris citra: 4
Jumlah kolom citra: 3
Nilai-nilai pixel citra:
[[[10 20 30]
 [40 50 60]
 [70 80 90]
 [90 80 70]]

 [[10 12 13]
 [14 15 16]
 [17 18 19]
 [19 18 17]]

 [[11 21 31]
 [41 51 61]
 [71 81 91]
 [91 81 71]]]
```

Contoh kode di atas merupakan dasar untuk memahami struktur sebuah citra digital multi saluran (*band*) sebagaimana citra penginderaan jauh multispektral. Untuk mempermudah pemahaman, pada kode di atas masing-masing band nilai-nilai pixelnya direkonstruksi menggunakan list. Kemudian ketiga list dikombinasikan menggunakan NumPy array untuk menghasilkan nilai-nilai pixel citra multispektral 3 band. Dimana jika diilustrasikan ke dalam bentuk gambar, Numpy array `citra` seperti pada kode di atas akan terlihat seperti berikut:

10	20	30
40	50	60
70	80	90
90	80	70

Band 1

10	12	13
14	15	16
17	18	19
19	18	17

Band 2

11	21	31
41	51	61
71	81	91
91	81	71

Band 3

Gambar 2.1. Ilustrasi citra 3 band

Instruksi `citra.shape` digunakan untuk menampilkan *shape* atau dimensi dari sebuah NumPy array, atau dengan kata lain dimensi dari sebuah citra. Dimensi citra yang ditunjukkan adalah dalam bentuk tuple `(3, 4, 3)`. Dimana secara berurutan, angka yang pertama (indeks ke-0) menunjukkan jumlah band, angka yang kedua (indeks ke-1) menunjukkan jumlah baris, dan angka yang ketiga (indeks ke-2) menunjukkan jumlah kolom. Sehingga instruksi `citra.shape[0]` akan memberikan informasi jumlah band, instruksi `citra.shape[1]` akan memberikan informasi jumlah baris, dan instruksi `citra.shape[2]` akan memberikan informasi jumlah kolom. Sebagaimana contoh kode program di atas. Demikianlah pixel-pixel sebuah citra digital disimpan di dalam NumPy array dan dikelola di dalam Python.

## Bab II Fundamental Python

Untuk mengakses array multidimensi sebagaimana contoh kode di atas, digunakan teknik-teknik sebagai berikut:

```
import numpy as np

band_1 = [[10, 20, 30], [40, 50, 60], [70, 80, 90], [90, 80, 70]]
band_2 = [[10, 12, 13], [14, 15, 16], [17, 18, 19], [19, 18, 17]]
band_3 = [[11, 21, 31], [41, 51, 61], [71, 81, 91], [91, 81, 71]]

citra = np.array([band_1, band_2, band_3])

print('Mengakses seluruh pixel di band 1:')
print(citra[0,:,:])

print('Mengakses seluruh pixel pada baris pertama di band 1:')
print(citra[0,0,:])

print('Mengakses seluruh pixel pada kolom ketiga di band 2:')
print(citra[:, :, 2])

print('Mengakses 1 pixel pada baris ketiga kolom kedua di band 3:')
print(citra[2, 2, 1])

print('Mengakses beberapa pixel pada baris pertama hingga baris ketiga kolom kedua di band 3:')
print(citra[2, 0:3, 1])
```

Output:

```
Mengakses seluruh pixel di band 1:
[[10 20 30]
 [40 50 60]
 [70 80 90]
 [90 80 70]]
Mengakses seluruh pixel pada baris pertama di band 1:
[10 20 30]
Mengakses seluruh pixel pada kolom ketiga di band 2:
[13 16 19 17]
Mengakses 1 pixel pada baris ketiga kolom kedua di band 3:
81
Mengakses beberapa pixel pada baris pertama hingga baris ketiga kolom kedua di band 3:
[21 51 81]
```

Bandingkan antara cara mengakses pixel-pixel citra pada kode program di atas, output kode programnya, dan Gambar 2.1 di atas.

---

*Penting untuk diperhatikan bahwa ilustrasi citra 3 band di atas hanya untuk mempermudah pemahaman struktur citra digital dan NumPy array.*

*Pada kenyataannya, ketika dihadapkan pada citra digital asli, struktur NumPy array yang terbentuk akan sangat tergantung pada framework yang digunakan untuk mengakses citra. Jika citra dengan format GeoTIFF misalnya dibuka menggunakan GDAL, maka struktur pixel dan NumPy array-nya adalah (baris, kolom, band). Bukan (band, baris, kolom) sebagaimana contoh di atas. Akan tetapi, kita tidak perlu khawatir dengan hal ini, sebab struktur NumPy array dapat ditransformasi atau dirubahbentuk sesuai keperluan.*

---

Kita dapat membuat sebuah NumPy array dengan isi 0 (nol) atau kosong dengan instruksi berikut:

```
import numpy as np
array_nol = np.zeros((2,3), dtype=np.uint8)
array_kosong = np.empty((2,3), dtype=np.float16)

print(array_nol)
print(array_kosong)
```

Output:

```
[[0 0 0]
 [0 0 0]]
 [[0. 0. 0.]
 [0. 0. 0.]]
```

Pada kode di atas, `(2,3)` menunjukkan dimensi array yang kita buat, sementara `dtype=np.uint8` dan `dtype=np.float16` adalah untuk menentukan tipe data array. Terkait dengan tipe data NumPy seperti `np.uint8`, `np.float16`, dan sebagainya, selengkapnya dapat diakses dan dipelajari sendiri di laman resmi NumPy <https://numpy.org/doc/stable/reference/arrays.dtypes.html>.

Pembuatan array kosong sebagaimana contoh di atas sangat lumrah dilakukan, yaitu ketika kita sedang menyiapkan semacam band atau citra kosong dengan struktur yang spesifik (band, baris, kolom) untuk menampung pixel-pixel hasil pemrosesan menggunakan algoritma tertentu.

Konstruksi NumPy array juga dapat dilakukan menggunakan `range`, `arange`, dan `linspace`. Perhatikan contoh kode program berikut:

```
import numpy as np
range_array = np.array(range(1,22,3))
arange_array = np.arange(1,10)
linspace_array = np.linspace(0,100,5)

print(range_array)
print(arange_array)
print(linspace_array)
```

Output:

```
[ 1  4  7 10 13 16 19]
[1 2 3 4 5 6 7 8 9]
[ 0. 25. 50. 75. 100.]
```

Pada contoh kode di atas, `range` akan menghasilkan array dari 1 sampai (sebelum) 22 dengan kelipatan 3. Ingat kembali aturan pada list, bahwa angka seperti 22 di atas bersifat eksklusif. `arange` akan menghasilkan array dari 1 sampai (sebelum) 10 secara berurutan. Sementara `linspace` akan menghasilkan 5 elemen array dari 0 sampai 100, dengan *equal interval* antar elemen array. Dalam hal ini, khusus pada `linspace`, 100 bersifat inklusif.

## Transformasi Bentuk NumPy Array

NumPy array dapat dirubah-rubah bentuknya sesuai keperluan. Misalnya dari  $(3, 3)$  menjadi  $(1, 9)$  atau  $(9, 1)$ . Perhatikan contoh kode berikut:

```
import numpy as np

my_array = np.array([[1,2,3],[4,5,6]])

print('my_array.shape:')
print(my_array.shape)
print('-----')
print('my_array.reshape(3,2):')
print(my_array.reshape(3,2))
print('-----')
print('my_array.reshape(6):')
print(my_array.reshape(6))
print('-----')
print('my_array.reshape(1,6):')
print(my_array.reshape(1,6))
print('-----')
print('my_array.reshape(6,1):')
print(my_array.reshape(6,1))
print('-----')
print('my_array.reshape(-1):')
print(my_array.reshape(-1))
print('-----')
print('my_array.reshape(1,-1):')
print(my_array.reshape(1,-1))
print('-----')
print('my_array.reshape(-1,1):')
print(my_array.reshape(-1,1))
print('-----')
print('np.transpose(my_array):')
print(np.transpose(my_array))
```

Output:

```
my_array.shape:
(2, 3)
-----
my_array.reshape(3,2):
[[1 2]
 [3 4]
 [5 6]]
-----
my_array.reshape(6):
[1 2 3 4 5 6]
-----
my_array.reshape(1,6):
[[1 2 3 4 5 6]]
-----
my_array.reshape(6,1):
[[1]
 [2]
 [3]
 [4]
 [5]
 [6]]
-----
my_array.reshape(-1):
[1 2 3 4 5 6]
-----
my_array.reshape(1,-1):
[[1 2 3 4 5 6]]
-----
my_array.reshape(-1,1):
[[1]
 [2]
 [3]]
```

```
[4]
[5]
[6]]
np.transpose(my_array):
[[1 4]
 [2 5]
 [3 6]]
```

Array `my_array` merupakan sebuah array dengan dimensi `(2, 3)`. Dimensinya dapat dirubah menjadi `(3, 2)` dengan kode `my_array.reshape(3, 2)`. `my_array.reshape(6)` dan `my_array.reshape(-1)` menghasilkan output yang sama, dimana instruksi ini dikenal juga sebagai *flattening array*. Sebab instruksi ini akan menghasilkan array 1-D (satu dimensi). Instruksi ini dapat diganti dengan `my_array.flatten()`, outputnya akan sama persis. `my_array.reshape(1, 6)` dan `my_array.reshape(1, -1)` akan menghasilkan output yang sama, yaitu array 1 baris dengan 2x3 atau 6 kolom. `my_array.reshape(6, 1)` dan `my_array.reshape(-1, 1)` akan menghasilkan output yang sama, yaitu array 2x3 atau 6 baris dengan 1 kolom. Baik array 1 baris 6 kolom atau pun 6 baris 1 kolom, keduanya disebut array 2-D (dua dimensi). Sementara array seperti contoh citra sebelumnya, yaitu 3 band, 4 baris, dan 3 kolom, disebut juga sebagai array 3-D (tiga dimensi).

Bagian yang terakhir, yaitu `np.transpose(my_array)` akan memutar array, sehingga baris menjadi kolom dan kolom menjadi baris. Transpose array dengan dimensi `(2, 3)` akan merubahnya menjadi array `(3, 2)`. Akan tetapi, `np.transpose(my_array)` akan berbeda dengan `my_array.reshape(3, 2)`. Sebab `my_array.reshape(3, 2)` tidak memutar dan menukar posisi elemen array sebagaimana `np.transpose(my_array)`. `my_array.reshape(3, 2)` hanya memindah posisi, sementara urutan eleman array-nya tetap. Coba cermati output kode program di atas, bandingkan antara output `my_array.reshape(3, 2)` dan output `np.transpose(my_array)`. Faktanya, transpose array banyak ditemukan pada algoritma-algoritma statistika yang sering diterapkan di dalam pemrosesan citra penginderaan jauh. Perhatikan algoritma *decision function* dari *Bayesian Learning (Maximum Likelihood Classifier)* berikut (Richards, 2022):

$$d_j(x) = \operatorname{argmax} \left( \ln p(\omega_j) - \frac{1}{2} \ln |C_j| - \frac{1}{2} \left[ (x - m_j)^T C_j^{-1} (x - m_j) \right] \right)$$

$(x - m_j)^T$  merupakan transpose array dari array hasil pengurangan antara vektor nilai pixel dengan vektor rerata nilai pixel untuk setiap kelas spektral.

Tentu saja, transformasi dimensi array harus mengacu dan kompatibel dengan dimensi array-nya, sehingga tidak dapat dilakukan sembarangan. Misalnya `my_array` dengan dimensi `(2, 3)` di atas jika ditransformasi menjadi `(4, 2)` atau `(1, 7)`, tentu saja akan error. Sebab NumPy array tidak dapat menyesuaikan bentuk array terhadap elemen-elemennya. Transformasi array pada dasarnya seperti menyusun ulang elemen-elemen yang sudah ada tanpa merubah jumlahnya. Pada praktiknya, transformasi array sering dilakukan di dalam komputasi nilai-nilai pixel citra digital. Misalnya di dalam *machine learning* atau *deep learning*.

## Penggabungan dan Pemisahan NumPy Array

Beberapa NumPy array dapat digabungkan menjadi satu array dengan berbagai instruksi, diantaranya dapat menggunakan `numpy.concatenate`, `numpy.stack`, `numpy.dstack`, `numpy.hstack`, `numpy.vstack`, `numpy.column_stack`, dan `numpy.row_stack`. Dan sebuah NumPy array dapat dipisahkan menjadi beberapa array menggunakan instruksi

## Bab II Fundamental Python

`numpy.split`, `numpy.array_split`, `numpy.dsplit`, `numpy.hsplit`, dan `numpy.vsplit`. Buku ini tidak akan mengulas semua fasilitas ini. Untuk membaca penjelasannya secara lebih komprehensif, Anda dapat mengakses laman resmi NumPy di <https://numpy.org/doc/stable/reference/routines.array-manipulation.html>.

Fasilitas yang umum digunakan di dalam visualisasi citra komposit RGB di dalam Python adalah `numpy.dstack`. `d` disini menunjukkan *depth-wise*, yang artinya NumPy array ditumpuk (*stack*) pada *axis* atau dimensi yang ketiga di dalam array 3-Dimensi seperti citra satelit. Misalnya terdapat sebuah citra satelit digital dengan 3 band, yaitu *Red*, *Green*, *Blue*. Dimana masing-masing band memiliki dimensi 10 baris pixel dan 10 kolom pixel. Kemudian ketiga band ini akan dikombinasikan untuk membentuk citra komposit RGB di dalam Python, dengan menggunakan Matplotlib. Perhatikan contoh implementasi kode programnya sebagai berikut:

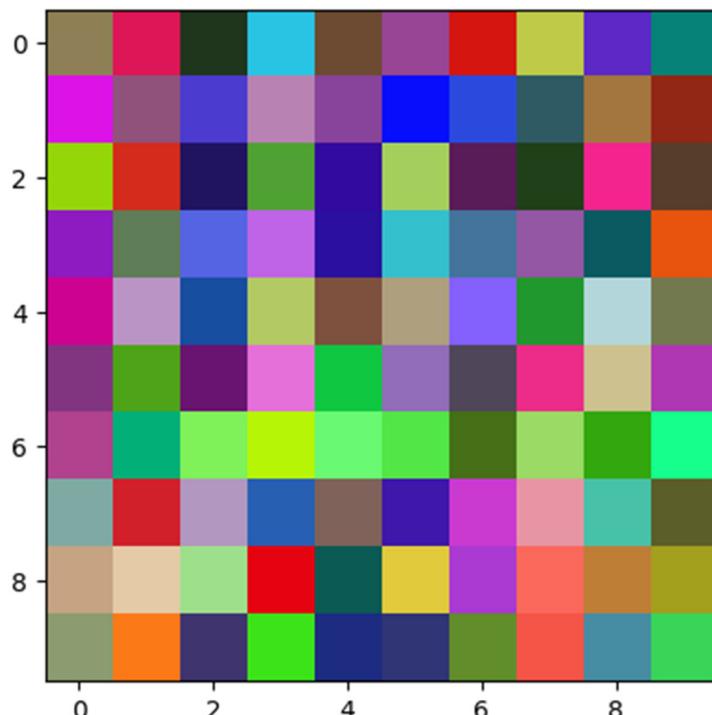
```
import numpy as np
import matplotlib.pyplot as plt

Red = np.random.rand(10,10)
Green = np.random.rand(10,10)
Blue = np.random.rand(10,10)

RGB = np.dstack((Red,Green,Blue))

plt.imshow(RGB)
```

Output:



Instruksi `np.random.rand` pada contoh di atas digunakan untuk membuat NumPy array dengan elemen angka acak. Teknik `numpy.dstack` seperti di atas akan sering kita lakukan di dalam pemrosesan citra digital. Tentu saja di dalam pembuatan citra komposit RGB.

## Mencari Elemen Unik

Instruksi `numpy.unique` dapat digunakan untuk mencari elemen-elemen unik di dalam NumPy array. Perhatikan contoh kode berikut:

```
import numpy as np
x = np.array([1, 1, 1, 2, 2, 2, 3, 3, 3])
y = np.unique(x)
print(y)
```

Output:

```
[1 2 3]
```

## Kalkulus Pada NumPy Array

Pada prinsipnya, kalkulasi pada Numpy array bersifat “*element-wise*”, yang artinya operasi matematika antar array dilakukan elemen per elemen. Perhatikan contoh kode berikut:

```
import numpy as np
array_satu = np.array([[1,2,3,4],[5,6,7,8]])
array_dua = np.array([[4,3,2,1],[8,7,6,5]])
print(array_satu + array_dua)
print(array_satu - array_dua)
print(array_satu * array_dua)
print(array_dua / array_satu)
```

Output:

```
[[ 5  5  5  5]
 [13 13 13 13]
 [[-3 -1  1  3]
 [-3 -1  1  3]
 [[ 4  6  6  4]
 [40 42 42 40]
 [[4.          1.5        0.66666667 0.25
 [1.6         1.16666667 0.85714286 0.625      ]]
```

## Kalkulus Vektor Pada NumPy Array

NumPy memiliki “semua” alat yang tersedia untuk melakukan kalkulasi aljabar linier, seperti determinan, *eigenvalue*, *eigenvector*, dan banyak lagi (Hazrat, 2023). Perhatikan matriks A dan matriks B yang memiliki dimensi sama berikut:

$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \quad B = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$$

Perkalian matriks A dan matriks B dengan menggunakan NumPy dilakukan sebagai berikut:

```
import numpy as np
A = np.array([[1,2],[3,4]])
```

## Bab II Fundamental Python

```
B = np.array([[5,6],[7,8]])
print(np.dot(A,B))
print(A.dot(B))
print(np.dot(B,A))
print(B.dot(A))
```

Output:

```
[[19 22]
 [43 50]]
 [[19 22]
 [43 50]]
 [[23 34]
 [31 46]]
 [[23 34]
 [31 46]]
```

Perhatikan kode di atas, `np.dot(A,B)` sama dengan `A.dot(B)`. Dan `np.dot(B,A)` sama dengan `B.dot(A)`. Di dalam aturan matematika kita tahu bahwa perkalian matriks tidak bersifat komutatif. Sehingga matriks A dan matriks B tidak akan sama dengan perkalian matriks B dan matriks A. Dan sebagaimana aturan matematika juga bahwa perkalian matriks sebagaimana contoh di atas adalah perkalian antara baris dan kolom.

Sekarang versi 3.5, Python memperkenalkan operator `@` untuk mempermudah dan mempercepat perkalian matriks. Perhatikan contoh kode berikut:

```
import numpy as np
A = np.array([[1,2],[3,4]])
B = np.array([[5,6],[7,8]])
print(A @ B)
print(B @ A)
```

Output:

```
[[19 22]
 [43 50]]
 [[23 34]
 [31 46]]
```

Perhatikan lagi contoh kode Python berikut:

```
import numpy as np
from numpy.linalg import det, inv
A = np.array([[6,2],[5,4]])
print(det(A))
print(inv(A))
print(np.dot(A,inv(A)))
```

Output:

```
14.000000000000004
 [[ 0.28571429 -0.14285714]
 [-0.35714286  0.42857143]]
 [[1. 0.]
 [0. 1.]]
```

Pada contoh kode di atas, `det(A)` digunakan untuk menghitung determinan matriks A, dan `inv(A)` digunakan untuk mencari matriks kebalikan (*inverse matrix*) dari matriks A, atau dinotasikan dengan  $A^{-1}$ . Di dalam matematika, jika suatu matriks dikalikan dengan matriks kebalikannya (`np.dot(A, inv(A))`) maka akan menghasilkan matriks identitas. Matriks identitas adalah matriks yang elemennya pada diagonal utama 1 semuanya, sedangkan elemen lainnya di luar diagonal utama 0 semuanya, sebagaimana pada contoh di atas.

Inverse matrix banyak ditemukan pada algoritma statistika yang umum diterapkan di dalam pemrosesan citra penginderaan jauh. Perhatikan kembali algoritma Bayesian Learning pada halaman 107 sebelumnya, dimana parameter  $C_j^{-1}$  merupakan merupakan inverse matrix dari covariance matrix  $C_j$  untuk setiap kelas spektral. Di dalam NumPy, covariance matrix sendiri dapat diekstrak sebagaimana contoh kode berikut:

```
import numpy as np
my_array = np.array([[12, 32], [41, 35]])
print('my_array: ')
print(my_array)
print('-----')
print('np.cov(my_array): ')
print(np.cov(my_array))
```

Output:

```
my_array:
[[12 32]
 [41 35]]
-----
np.cov(my_array):
[[200. -60.]
 [-60. 18.]]
```

Instruksi `np.cov` pada contoh kode di atas digunakan untuk mengekstrak covariance matrix.

### Konstanta NumPy

NumPy memiliki sejumlah konstanta yang sering kita libatkan di dalam pemrosesan citra digital. Terutama di dalam algoritma-algoritma matematika dan statistika. Berikut adalah konstanta-konstanta yang disediakan oleh NumPy (<https://numpy.org>):

Tabel 2.12. Konstanta NumPy

Nomor	Konstanta	Arti
1	<code>numpy.e</code>	Konstanta Euler e = 2.718281828459...
2	<code>numpy.euler_gamma</code>	$\gamma = 0.57721566490153286...$
3	<code>numpy.inf</code>	Infinity (bilangan tak terhingga)
4	<code>numpy.nan</code>	Not a Number (bukan angka)
5	<code>numpy.newaxis</code>	Merupakan alias dari <code>None</code>
6	<code>numpy.pi</code>	$\pi = 3.1415926535897932384626433...$

## D. SciPy

SciPy (*Scientific Python*) (Virtanen et al., 2020) merupakan librari Python yang menyediakan algoritma untuk optimasi, integrasi, interpolasi, *eigenvalue*, persamaan aljabar, persamaan diferensial, statistik, dan problematika lainnya (<https://scipy.org>). Di dalam pemrosesan citra digital penginderaan jauh, SciPy memegang peranan penting di dalam implementasi algoritma-algoritma pemrosesan citra digital. SciPy diorganisir menjadi beberapa sub-paket yang mencakup domain-domain komputasi ilmiah berbeda (<https://scipy.org>). SciPy sudah terintegrasi di dalam Google Colab. Sementara di lingkungan Anaconda Prompt, SciPy diinstal dengan instruksi berikut:

```
conda install anaconda::scipy
```

Tabel 2.13. Sub-paket SciPy

Sub-paket	Deskripsi
<a href="#">cluster</a>	Algoritma-algoritma pengelompokkan
<a href="#">constants</a>	Konstanta-konstanta fisika dan matematik
<a href="#">fft</a>	Transformasi Fourier Diskrit
<a href="#">fftpack</a>	Fungsi Transformasi Fourier Cepat
<a href="#">integrate</a>	Integrasi dan pemecah persamaan diferensial biasa
<a href="#">interpolate</a>	Interpolasi spline dan penghalusan
<a href="#">io</a>	Fasilitas input dan output
<a href="#">linalg</a>	Aljabar linier
<a href="#">ndimage</a>	Pemrosesan citra n-dimensional
<a href="#">odr</a>	Regresi jarak ortogonal
<a href="#">optimize</a>	Optimasi dan fungsi pencarian akar
<a href="#">signal</a>	Pemrosesan sinyal
<a href="#">sparse</a>	Matriks sparse dan fungsi-fungsi terkait
<a href="#">spatial</a>	Struktur data spasial dan algoritma
<a href="#">special</a>	Fungsi-fungsi khusus
<a href="#">stats</a>	Distribusi dan fungsi-fungsi statistik

Sumber: <https://docs.scipy.org/doc/scipy/tutorial/index.html#user-guide>

Sebagaimana pembahasan NumPy sebelumnya, tentu saja tidak mungkin untuk membahas semua fungsi-fungsi SciPy di dalam buku ini. Berikut ini hanya disajikan contoh-contoh kode program untuk komputasi statistik dengan menggunakan SciPy. Selebihnya Anda dapat membaca sendiri petunjuk-petunjuk penggunaan SciPy di laman resminya, atau membaca literatur-literatur lain yang lebih komprehensif dalam membahas SciPy.

Perhatikan contoh-contoh kode berikut:

```
# Kalkulasi jarak spasial di antara 2 titik koordinat
from scipy.spatial.distance import cityblock, euclidean, chebyshev, cosine,
minkowski

titik_A = [250000, 9650000]
titik_B = [253000, 9654000]

manhattan_distance = cityblock(titik_A,titik_B)
euclidean_distance = euclidean(titik_A,titik_B)
chebysev_distance = chebyshev(titik_A,titik_B)
```

```

cosine_distance = cosine(titik_A,titik_B)
minkowski_distance = minkowski(titik_A,titik_B)

print(f'Manhattan distance dari titik A ke titik B: {manhattan_distance}')
print(f'Euclidean distance dari titik A ke titik B: {euclidean_distance}')
print(f'Chebysev distance dari titik A ke titik B: {chebysev_distance}')
print(f'Cosine distance dari titik A ke titik B: {cosine_distance}')
print(f'Minkowski distance dari titik A ke titik B: {minkowski_distance}')

```

Output:

```

Manhattan distance dari titik A ke titik B: 7000
Euclidean distance dari titik A ke titik B: 5000.0
Chebysev distance dari titik A ke titik B: 4000
Cosine distance dari titik A ke titik B: 4.4944326038631743e-08
Minkowski distance dari titik A ke titik B: 5000.0

```

```

# Konstanta-konstanta fisika dan matematika

from scipy.constants import pi, c, g, Stefan_Boltzmann

print(f'Konstanta PI {pi}.')
print(f'Kecepatan cahaya di ruang vakum {c} m/s.')
print(f'Akselerasi gravitasi {g} m/s2.')
print(f'Konstanta Stefan Boltzmann {Stefan_Boltzmann} W/m2K4.')

```

Output:

```

Konstanta PI 3.141592653589793.
Kecepatan cahaya di ruang vakum 299792458.0 m/s.
Akselerasi gravitasi 9.80665 m/s2.
Konstanta Stefan Boltzmann 5.670374419e-08 W/m2K4.

```

```

# Aljabar linier: Menghitung inverse matrix

import numpy as np
from scipy.linalg import inv

mat = np.array([[6,2],[5,4]])

inv_mat = inv(mat)

print('Matriks:')
print(mat)
print('Inverse matriks:')
print(inv_mat)
print('Hasil kali matriks dan inverse matriksnya:')
print(mat @ inv_mat)

```

Output:

```

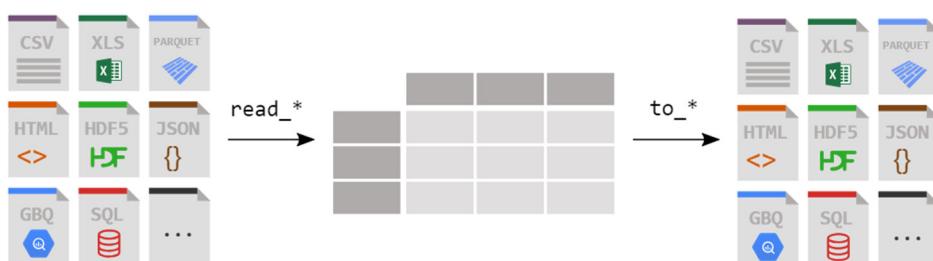
Matriks:
[[6 2]
 [5 4]]
Inverse matriks:
[[ 0.28571429 -0.14285714]
 [-0.35714286  0.42857143]]
Hasil kali matriks dan inverse matriksnya:
[[1. 0.]
 [0. 1.]]

```

## E. Pandas

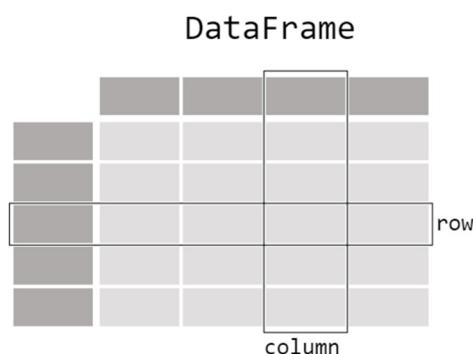
Pandas (*Python Data Analysis*) (McKinney, 2010; The pandas development team, 2020) adalah librari Python yang diperuntukkan untuk manajemen basisdata (*database*) dan analisis data di dalam Python. Di dalam pemrosesan citra digital penginderaan jauh, Pandas pada umumnya digunakan untuk mengintegrasikan data dari Sistem Informasi Geografis (SIG), atau data dari hasil survei lapangan (misalnya dalam format Excel atau CSV), dengan pixel-pixel citra digital penginderaan jauh. Sebagai contoh, ketika kita ingin membangun model regresi antara *Normalized Difference Vegetation Index* (NDVI) dengan data lapangan biomassa vegetasi. Informasi tabular biomassa vegetasi hasil kompilasi dari data survei lapangan harus diakses dengan Pandas terlebih dahulu ke dalam lingkungan Python. Untuk mempelajari Pandas secara ringkas, kita dapat menggunakan cheatsheet resmi yang disediakan oleh Pandas di [https://pandas.pydata.org/Pandas\\_Cheat\\_Sheet.pdf](https://pandas.pydata.org/Pandas_Cheat_Sheet.pdf).

Secara bawaan, Pandas mendukung akses (*read and write*) data dengan format-format berikut:



Gambar 2.2. Format-format data tabular yang dapat diakses Pandas (<https://pandas.pydata.org>)

Di dalam Pandas, sebuah tabel data disebut DataFrame, yang memiliki struktur sebagaimana Gambar 2.3 berikut:



Gambar 2.3. Struktur DataFrame Pandas (<https://pandas.pydata.org>)

Di lingkungan Google Colab, Pandas sudah terintegrasi dan dapat langsung dipanggil. Sementara di lingkungan Anaconda Prompt, Pandas dapat diinstal dengan instruksi berikut:

```
conda install -c conda-forge pandas
```

## Dasar-dasar Pandas DataFrame

Sebagai dasar untuk memahami Pandas DataFrame, kita akan menggunakan contoh data yang kita buat sebelumnya pada topik dictionary.

Perhatikan contoh kode berikut:

```
import pandas as pd

# Membuat list
no_kab = [1, 2, 3, 4, 5]
nama_kab = ['Banjar', 'Tanah Laut', 'Barito Kuala', 'Tapin', 'Hulu Sungai Selatan']
ibukota = ['Martapura', 'Pelaihari', 'Marabahan', 'Rantau', 'Kandangan']

# Membuat dictionary dari list
kabupaten = {
    'Nomor': no_kab,
    'Nama Kabupaten': nama_kab,
    'Ibukota': ibukota
}

# Membuat Pandas DataFrame dari dictionary
df = pd.DataFrame.from_dict(kabupaten)
df
```

Output:

	Nomor	Nama Kabupaten	Ibukota
0	1	Banjar	Martapura
1	2	Tanah Laut	Pelaihari
2	3	Barito Kuala	Marabahan
3	4	Tapin	Rantau
4	5	Hulu Sungai Selatan	Kandangan

Pada contoh kode di atas, sebuah dictionary yang dikonstruksi dari beberapa list, dikonversi menjadi sebuah Pandas DataFrame menggunakan instruksi `pd.DataFrame.from_dict`.

Sebuah Pandas DataFrame juga dapat dikonstruksi secara langsung dari sebuah list atau sebuah NumPy array, atau kombinasi list dan NumPy array. Yaitu dengan menggunakan instruksi `pd.DataFrame.from_records`. Perhatikan contoh kode berikut:

```
import pandas as pd
import numpy as np

provinsi = ['Kalbar', 'Kalsel', 'Kalteng', 'Kaltim', 'Kaltara']
plat = np.array(['KB', 'DA', 'KH', 'KT', 'KU'])

df = pd.DataFrame.from_records({'Provinsi': provinsi, 'Kode Plat': plat})
df
```

Output:

	Kode	Plat	Provinsi
0		KB	Kalbar
1		DA	Kalsel
2		KH	Kalteng
3		KT	Kaltim
4		KU	Kaltara

Perhatikan perbedaannya antara konstruksi Pandas DataFrame dari dictionary dan dari list/NumPy array. Pada proses konstruksi Pandas DataFrame dari dictionary, kita tidak perlu menambahkan label kolom. Hal ini karena elemen-elemen di dalam dictionary sudah memiliki label. Akan tetapi, ketika list/NumPy array dikonversi menjadi Pandas DataFrame, kita perlu menambahkan label kolom secara eksplisit.

### Mengakses Data Tabular Menggunakan Pandas

Sebagaimana ditunjukkan pada Gambar 2.2, kita dapat mengakses file *Comma-separated values* (CSV) dengan menggunakan Pandas. Perhatikan contoh kode berikut:

```
import pandas as pd
from google.colab import drive

drive.mount('/content/gdrive')

ground_samples =
pd.read_csv('/content/gdrive/MyDrive/geebook/table/ground_samples.csv')
ground_samples
```

Output:

	PLOT_ID	CATEGORY	LAT	LONG	BIOMASS
0	Plot 01	Validation Plot	-3.51182	114.940	96.637
1	Plot 02	Training Plot	-3.51071	114.940	129.327
2	Plot 03	Training Plot	-3.50907	114.942	274.912
3	Plot 04	Training Plot	-3.50960	114.943	268.307
4	Plot 05	Validation Plot	-3.51043	114.943	74.497
5	Plot 06	Training Plot	-3.51066	114.944	235.203
6	Plot 07	Training Plot	-3.51014	114.945	87.461
7	Plot 08	Training Plot	-3.50953	114.945	90.639
8	Plot 09	Training Plot	-3.50892	114.946	114.997

Perhatikan bahwa contoh kode di atas dijalankan menggunakan Google Colab. Dan dengan asumsi Anda sudah menjalankan petunjuk penggunaan buku ini sebagaimana pada halaman 20. Jika Anda menjalankannya menggunakan JupyterLab secara offline, maka kode programnya memerlukan sedikit penyesuaian, yaitu di bagian argumen `pd.read_csv`. Jika data `ground_samples.csv` tersimpan di laptop Anda di dalam folder `D:\geebok\table`, maka ganti kodennya menjadi `pd.read_csv('D:/geebok/table/ground_samples.csv')`. Dan jika Anda menjalankan kode program di atas secara offline menggunakan JupyterLab, tentu saja tidak perlu ada instruksi `from google.colab import drive` dan `drive.mount('/content/gdrive')`. Karena kedua baris instruksi ini digunakan untuk mengakses data yang tersimpan di dalam Google Drive.

Jika DataFrame kita cukup panjang, `ground_samples` dapat diganti menjadi `ground_samples.head`, untuk menampilkan 5 baris pertama DataFrame, atau `ground_samples.head(10)` untuk menampilkan 10 baris pertama DataFrame. Untuk menampilkan 5 baris terakhir DataFrame, gunakan `ground_samples.tail(5)`. Perhatikan bahwa setiap baris data di dalam Pandas DataFrame akan memiliki indeks dimulai 0 (nol) sebagaimana list atau NumPy array. Untuk menampilkan DataFrame tanpa indeks gunakan instruksi `ground_samples.style.hide()`.

## Mengakses Data di dalam DataFrame

Untuk mengakses 1 kolom DataFrame gunakan instruksi seperti `ground_samples[['PLOT_ID']]`. Untuk mengakses 2 kolom DataFrame gunakan instruksi seperti `ground_samples[['PLOT_ID', 'BIOMASS']]`. Berikut adalah contoh output dari 2 kolom DataFrame yang diakses dengan instruksi `ground_samples[['PLOT_ID', 'BIOMASS']].head(5).style.hide()`.

PLOT_ID	BIOMASS
Plot 01	96.637000
Plot 02	129.327000
Plot 03	274.912000
Plot 04	268.307000
Plot 05	74.497000

Jika kita menggunakan instruksi `ground_samples['BIOMASS']`, hasilnya akan berbeda:

	BIOMASS
0	96.637
1	129.327
2	274.912
3	268.307
4	74.497

Perhatikan output di atas, dimana label kolomnya tidak ditampilkan. Dan jika instruksinya seperti ini `ground_samples['BIOMASS'].to_numpy()`, maka hasilnya adalah sebuah NumPy array dari kolom data **BIOMASS**. Instruksi `ground_samples['BIOMASS']` sama dengan `ground_samples.BIOMASS`, dan `ground_samples['BIOMASS'].to_numpy()` sama dengan `ground_samples.BIOMASS.to_numpy()`.

### loc dan iloc

`loc` dan `iloc` merupakan dua metode yang dapat digunakan untuk mengakses baris-baris dan kolom-kolom data di dalam Pandas DataFrame. `loc` mengakses berdasarkan label indeks, sedangkan `iloc` berdasarkan nomor indeks (dimulai dari 0).

`loc` akan lebih efektif jika indeksnya diberi label informatif, bukan indeks otomatis dari 0 sampai n. Atau salah satu kolom di dalam tabel data kita jadikan sebagai indeks. Misalnya **PLOT\_ID** sebagaimana contoh DataFrame di atas. instruksinya adalah sebagai berikut:

```
ground_samples.set_index('PLOT_ID', inplace=True)
```

Instruksi `inplace=True` digunakan agar perubahan indeks dilakukan pada DataFrame original. Bukan membuat DataFrame yang baru. Perhatikan outputnya seperti berikut:

	CATEGORY	LAT	LONG	BIOMASS
PLOT_ID				
<b>Plot 01</b>	Validation Plot	-3.51182	114.940	96.637
<b>Plot 02</b>	Training Plot	-3.51071	114.940	129.327
<b>Plot 03</b>	Training Plot	-3.50907	114.942	274.912
<b>Plot 04</b>	Training Plot	-3.50960	114.943	268.307
<b>Plot 05</b>	Validation Plot	-3.51043	114.943	74.497

Setelah **PLOT\_ID** dijadikan indeks sebagaimana tabel di atas, maka untuk mengakses **PLOT\_ID** dari 1 sampai 3 dengan menggunakan `loc` instruksinya adalah berikut:

```
ground_samples.loc['Plot 01':'Plot 03']
```

Perhatikan outputnya:

	CATEGORY	LAT	LONG	BIOMASS
PLOT_ID				
<b>Plot 01</b>	Validation Plot	-3.51182	114.940	96.637
<b>Plot 02</b>	Training Plot	-3.51071	114.940	129.327
<b>Plot 03</b>	Training Plot	-3.50907	114.942	274.912

Jika kita menggunakan `iloc` sebagaimana instruksi berikut:

```
ground_samples.iloc[1:3]
```

Outputnya akan berbeda:

	CATEGORY	LAT	LONG	BIO MASS
PLOT_ID				
<b>Plot 02</b>	Training Plot	-3.51071	114.940	129.327
<b>Plot 03</b>	Training Plot	-3.50907	114.942	274.912

Mengapa demikian? Sebab `iloc` menggunakan indeks, sebagaimana ketika kita mengakses list atau NumPy array. Sehingga `iloc[1:3]` berarti **PLOT\_ID** di indeks 1 (yaitu Plot 02), sebab indeks selalu dimulai dari 0. Dan **PLOT\_ID** di indeks 2 (yaitu Plot 03). Ingat kembali, bahwa 3 di dalam `iloc[1:3]` bersifat eksklusif.

Untuk memfilter data **Training Plot** di kolom **CATEGORY** dapat digunakan instruksi berikut:

```
ground_samples.loc[(ground_samples['CATEGORY'] == 'Training Plot')]
```

atau

```
ground_samples.loc[ground_samples.CATEGORY.str.contains('Training Plot')]
```

Output dari kedua instruksi di atas adalah sama, yaitu hanya menampilkan data yang entri di kolom **CATEGORY** hanya **Training Plot**. Sebagaimana tabel berikut:

	CATEGORY	LAT	LONG	BIO MASS
PLOT_ID				
<b>Plot 02</b>	Training Plot	-3.51071	114.940	129.327
<b>Plot 03</b>	Training Plot	-3.50907	114.942	274.912
<b>Plot 04</b>	Training Plot	-3.50960	114.943	268.307
<b>Plot 06</b>	Training Plot	-3.51066	114.944	235.203
<b>Plot 07</b>	Training Plot	-3.51014	114.945	87.461

Teknik di atas pada umumnya diterapkan jika kita ingin memisahkan (*slicing*) data dengan kriteria tertentu, sebagaimana ketika kita melakukan *select* atau *query* di dalam basisdata. Misalnya pada kasus ketika kita memisahkan data untuk *training data* dan *validation data*.

## Mengedit Data di dalam DataFrame

### Merubah Data

Perhatikan kembali contoh DataFrame sebelumnya:

PLOT_ID	CATEGORY	LAT	LONG	BIOMASS
Plot 01	Validation Plot	-3.51182	114.940	96.637
Plot 02	Training Plot	-3.51071	114.940	129.327
Plot 03	Training Plot	-3.50907	114.942	274.912
Plot 04	Training Plot	-3.50960	114.943	268.307
Plot 05	Validation Plot	-3.51043	114.943	74.497

Untuk merubah sebuah sel data, misalnya pada entri **CATEGORY** untuk **Plot 03** akan kita rubah menjadi **Testing Plot**. Instruksinya adalah metode `at` sebagaimana contoh kode berikut:

```
ground_samples.at['Plot 03', 'CATEGORY'] = 'Testing Plot'
```

Perhatikan outputnya:

PLOT_ID	CATEGORY	LAT	LONG	BIOMASS
Plot 01	Validation Plot	-3.51182	114.940	96.637
Plot 02	Training Plot	-3.51071	114.940	129.327
Plot 03	Testing Plot	-3.50907	114.942	274.912
Plot 04	Training Plot	-3.50960	114.943	268.307
Plot 05	Validation Plot	-3.51043	114.943	74.497

### Menambahkan Baris dan Kolom Data

Untuk menambahkan sebaris data baru di dalam DataFrame, kita dapat menggunakan teknik dictionary berikut:

```
baris_baru = {'PLOT_ID': 'Plot 00', 'CATEGORY': 'Training Plot', 'LAT': -3.51283, 'LONG': 114.943, 'BIOMASS': '123.45'}  
ground_samples.loc['Plot 41'] = baris_baru
```

Instruksi di atas akan menambahkan baris baru di dalam DataFrame dengan **PLOT\_ID Plot 41**.

Untuk menambahkan sebuah kolom kosong baru, kita dapat menggunakan teknik berikut:

```
ground_samples['CARBON'] = pd.Series(dtype='float16')
```

Perhatikan outputnya:

	CATEGORY	LAT	LONG	BIOMASS	CARBON
PLOT_ID					
<b>Plot 01</b>	Validation Plot	-3.51182	114.940	96.637	NaN
<b>Plot 02</b>	Training Plot	-3.51071	114.940	129.327	NaN
<b>Plot 03</b>	Testing Plot	-3.50907	114.942	274.912	NaN
<b>Plot 04</b>	Training Plot	-3.50960	114.943	268.307	NaN
<b>Plot 05</b>	Validation Plot	-3.51043	114.943	74.497	NaN

Pada contoh kode di atas, kita menambahkan sebuah kolom kosong dengan nama **CARBON** dan tipe data yang akan diisi nantinya adalah **float16**. Karena kolomnya merupakan tipe data numerik dan isinya kosong, maka oleh Pandas akan diisi dengan **NaN** (*Not a Number*).

### Kalkulasi di dalam DataFrame

Untuk menambahkan sebuah kolom baru dengan nama **CARBON**, sekaligus mengisinya dengan angka hasil kalkulasi dari kolom lainnya yang sudah ada, dimana **CARBON** ekivalen dengan **BIOMASS** dikali **0.47**, digunakan instruksi sebagai berikut:

```
ground_samples['CARBON'] = ground_samples['BIOMASS'] * 0.47
```

Perhatikan outputnya:

	CATEGORY	LAT	LONG	BIOMASS	CARBON
PLOT_ID					
<b>Plot 01</b>	Validation Plot	-3.51182	114.940	96.637	45.41939
<b>Plot 02</b>	Training Plot	-3.51071	114.940	129.327	60.78369
<b>Plot 03</b>	Testing Plot	-3.50907	114.942	274.912	129.20864
<b>Plot 04</b>	Training Plot	-3.50960	114.943	268.307	126.10429
<b>Plot 05</b>	Validation Plot	-3.51043	114.943	74.497	35.01359

Untuk memperoleh ringkasan numerik dari sebuah DataFrame digunakan instruksi berikut:

```
ground_samples.describe()
```

## Bab II Fundamental Python

Perhatikan outputnya:

	LAT	LONG	BIOMASS	CARBON
count	40.000000	40.000000	40.000000	40.000000
mean	-3.509395	114.941425	171.025675	80.382067
std	0.001910	0.003358	86.349092	40.584073
min	-3.513530	114.935000	23.781000	11.177070
25%	-3.510673	114.938750	113.618750	53.400813
50%	-3.509420	114.942000	161.771500	76.032605
75%	-3.508105	114.944250	243.125250	114.268867
max	-3.505700	114.946000	405.096000	190.395120

### Menghapus Data

Untuk menghapus data di dalam DataFrame dapat digunakan metode `drop`. Untuk menghapus kolom **CATEGORY** pada DataFrame sebelumnya digunakan instruksi berikut:

```
ground_samples.drop(['CATEGORY'], axis = 'columns', inplace = True)
```

Untuk menghapus sebaris data pada **PLOT\_ID Plot 05** digunakan instruksi berikut:

```
ground_samples.drop(['Plot 05'], axis = 'index', inplace = True)
```

### Merubah Label Kolom

Untuk merubah label kolom **CATEGORY** menjadi **DESCRIPTION** digunakan metode `rename` sebagaimana contoh kode berikut:

```
ground_samples.rename(columns={'CATEGORY': 'DESCRIPTION'}, inplace=True)
```

Untuk merubah label 2 kolom sekaligus digunakan instruksi berikut:

```
ground_samples.rename(columns={'LAT': 'LATITUDE', 'LONG': 'LONGITUDE'}, inplace=True)
```

### Menyimpan DataFrame ke dalam File

Untuk menyimpan sebuah DataFrame ke dalam file Excel digunakan instruksi berikut:

```
ground_samples.to_excel('/content/gdrive/MyDrive/geebook/table/carbon_samples.xlsx')
```

Perhatikan kembali Gambar 2.2 sebelumnya, tentang format-format data tabular yang didukung oleh Pandas. Untuk menyimpan DataFrame ke dalam format CSV misalnya, instruksinya tinggal diganti menjadi `to_csv` seperti berikut:

```
ground_samples.to_csv('/content/gdrive/MyDrive/geebook/table/carbon_samples.csv')
```

## Visualisasi Pandas DataFrame

Kita dapat mengostumisasi tampilan DataFrame sesuai dengan keperluan atau keinginan kita, dengan menggunakan fasilitas DataFrame.style. Tentu saja, tidak mungkin untuk membahas semua teknik-teknik visualisasi menggunakan DataFrame.style di dalam buku ini. Selengkapnya, visualisasi Pandas DataFrame menggunakan DataFrame.style dapat Anda pelajari di laman resmi Pandas [https://pandas.pydata.org/pandas-docs/stable/user\\_guide/style.html](https://pandas.pydata.org/pandas-docs/stable/user_guide/style.html), atau pun dari sumber-sumber lainnya yang lebih lengkap beserta contoh-contohnya.

Perhatikan kode contoh berikut:

```
import pandas as pd
from google.colab import drive
drive.mount('/content/gdrive')

ground_samples =
pd.read_csv('/content/gdrive/MyDrive/geebook/table/ground_samples.csv')
ground_samples.set_index('PLOT_ID', inplace=True)
ground_samples.head(10).style.format(precision=3, thousands='.', decimal=',').background_gradient(subset='BIOMASS', cmap='plasma')
```

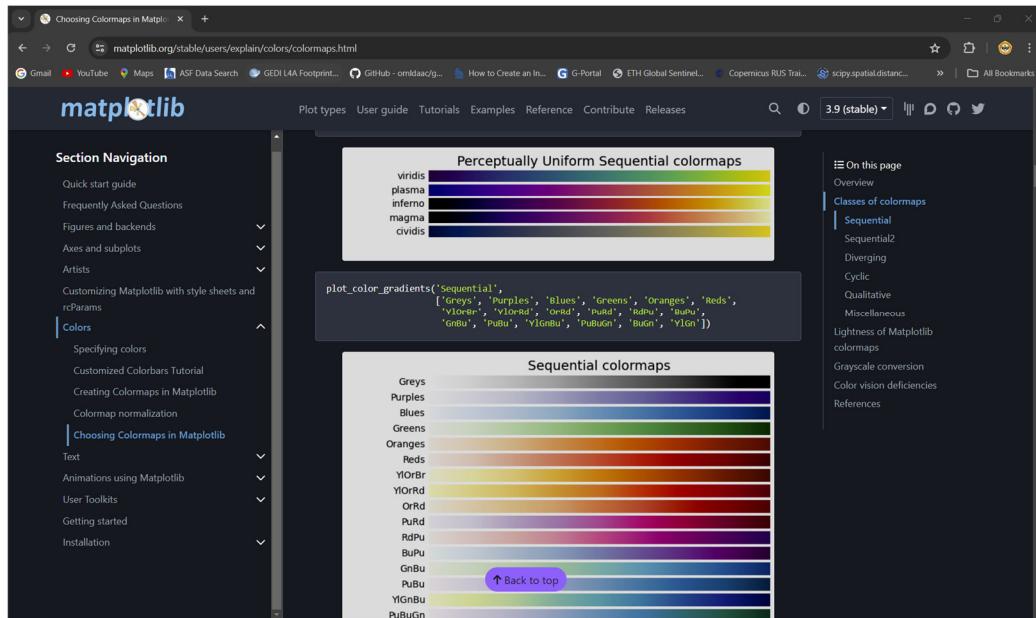
Output:

	CATEGORY	LAT	LONG	BIOMASS
PLOT_ID				
<b>Plot 01</b>	Validation Plot	-3,512	114,940	96,637
<b>Plot 02</b>	Training Plot	-3,511	114,940	129,327
<b>Plot 03</b>	Training Plot	-3,509	114,942	274,912
<b>Plot 04</b>	Training Plot	-3,510	114,943	268,307
<b>Plot 05</b>	Validation Plot	-3,510	114,943	74,497
<b>Plot 06</b>	Training Plot	-3,511	114,944	235,203
<b>Plot 07</b>	Training Plot	-3,510	114,945	87,461
<b>Plot 08</b>	Training Plot	-3,510	114,945	90,639
<b>Plot 09</b>	Training Plot	-3,509	114,946	114,997
<b>Plot 10</b>	Training Plot	-3,508	114,946	69,867

Pada contoh kode di atas, instruksi `format(precision=3, thousands='.', decimal=',')` akan menetapkan ketelitian desimal hingga 3 angka di belakang koma untuk semua kolom data numerik. Sekaligus merubah format data numeriknya menjadi format Indonesia, yaitu tanda titik untuk pemisah ribuan dan tanda koma untuk notasi desimal. Instruksi `background_gradient(subset='BIOMASS', cmap='plasma')` akan memberi

## Bab II Fundamental Python

warna gradasi khusus hanya pada kolom **BIOMASS**. Warna gradasi ditentukan dengan argumen `cmap='plasma'`. Tentu saja, Anda dapat mengganti warna dengan mengganti kode warna `plasma` dengan kode warna lain atau pun memberi warna kolom lainnya sesuai keinginan Anda. Silahkan buka laman <https://matplotlib.org/stable/users/explain/colors/colormaps.html> untuk memilih colormaps yang Anda inginkan. Sebagaimana terlihat pada Gambar 2.4.



Gambar 2.4. Laman Matplotlib Colormaps  
(<https://matplotlib.org/stable/users/explain/colors/colormaps.html>)

## Konversi DataFrame Menjadi Data Geospasial

Di dalam pemrosesan citra digital penginderaan jauh, misalnya ketika proses konstruksi model regresi antara nilai NDVI dan data lapangan, kita diharuskan untuk mengkonversi data tabular hasil survei lapangan menjadi data geospasial. Tentu saja, syarat yang harus dipenuhi adalah terdapat kolom-kolom yang berisi data koordinat geografis di dalam data lapangan kita. Dan untuk mengkonversi Pandas DataFrame menjadi data geospasial, kita memerlukan GeoPandas (<https://geopandas.org>). GeoPandas (*Geospatial Python Data Analysis*) merupakan bentuk ekstensi geospasial dari Pandas.

Perhatikan contoh kode berikut:

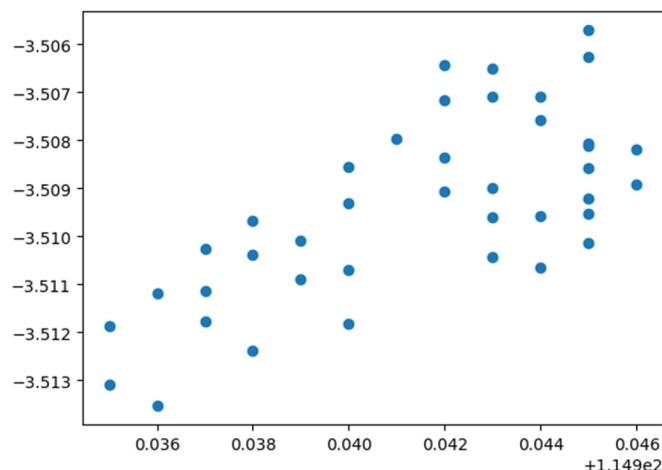
```
import pandas as pd
import geopandas as gpd
from google.colab import drive

drive.mount('/content/gdrive')

ground_samples =
pd.read_csv('/content/gdrive/MyDrive/geebook/table/ground_samples.csv')

sampling_points = gpd.GeoDataFrame(ground_samples,
geometry=gpd.points_from_xy(ground_samples.LONG, ground_samples.LAT))
sampling_points.plot()
```

Output:



Untuk selanjutnya, jika diperlukan, kita dapat menyimpan data geospasial di atas ke dalam *shapefile*, yaitu dengan menggunakan instruksi berikut:

```
sampling_points.to_file('/content/gdrive/MyDrive/geebook/vector/ground_samples.shp', crs='EPSG:4326')
```

`crs` pada kode di atas merupakan akronim dari *coordinate reference system* atau sistem referensi koordinat, dan '[EPSG:4326](#)' merupakan kode *European Petroleum Survey Group* (EPSG) (<https://epsg.io>) untuk *Geographic Coordinate System* (GCS) WGS 1984.

Tentu saja, fungsi-fungsi Pandas sangat luas dan tidak dapat dibahas seluruhnya di dalam buku singkat ini. Terlebih buku ini tidak diperuntukkan untuk membahas Pandas secara khusus. Untuk mempelajari Pandas secara komprehensif, kami merekomendasikan Anda untuk membaca dari buku-buku lain yang memang didedikasikan khusus untuk mengulas Pandas.

## F. GeoPandas

Sebagaimana sudah disinggung pada bagian terdahulu, bahwa GeoPandas (Jordahl et al., 2020) adalah bentuk peningkatan fasilitas dari Pandas yang ditujukan untuk pemrosesan data geospasial. Dengan menggunakan GeoPandas, kita dapat mengakses format data geospasial seperti *ESRI Shapefile*, membuat shapefile, melakukan seleksi data geospasial, hingga visualisasi data geospasial dan analisis-analisis geospasial.

GeoPandas mengimplementasikan dua struktur data utama, `GeoSeries` dan `GeoDataFrame`. Keduanya merupakan subkelas dari `pandas.Series` dan `pandas.DataFrame`. `GeoSeries` pada dasarnya adalah sebuah vektor di mana setiap entri dalam vektor adalah sekumpulan bentuk yang sesuai dengan satu pengamatan. Sebuah entri dapat terdiri dari hanya satu bentuk (seperti satu poligon) atau beberapa bentuk yang dimaksudkan untuk dianggap sebagai satu pengamatan (seperti banyak poligon yang membentuk Negara Bagian Hawaii atau negara seperti Indonesia). Sementara `GeoDataFrame` adalah struktur data tabular yang berisi `GeoSeries` (<https://geopandas.org>).

## Mengakses Data Geospasial

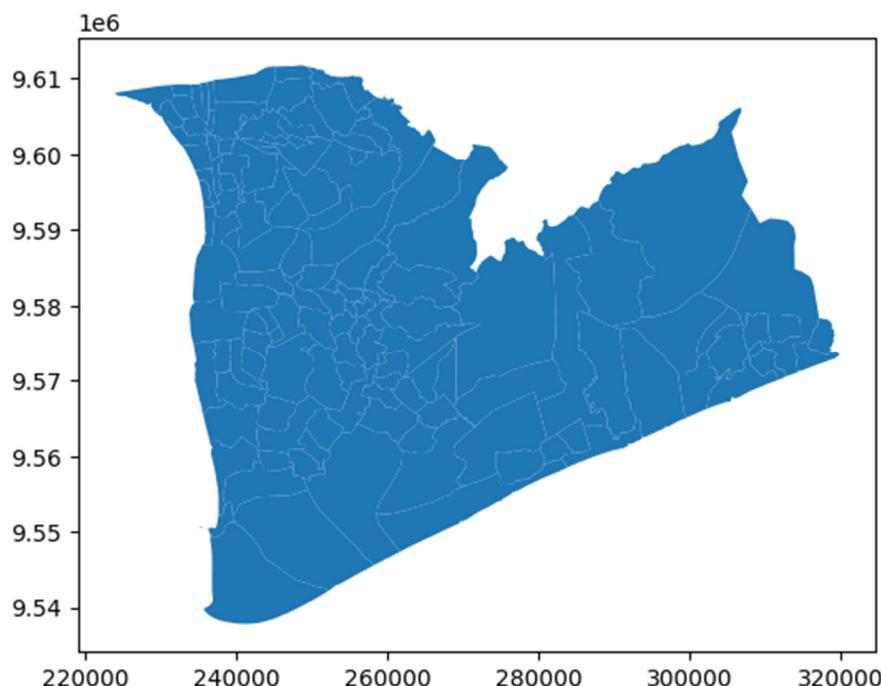
Untuk mengakses dan menampilkan shapefile yang disimpan di dalam Google Drive dari Google Colab digunakan metode `read_file` sebagaimana contoh kode berikut:

```
import geopandas as gpd
from google.colab import drive

drive.mount('/content/gdrive')

tanah_laut =
gpd.read_file('/content/gdrive/MyDrive/geebook/vector/Tanah_Laut.shp')
tanah_laut.plot()
```

Output:



## Seleksi Data Geospasial

Untuk menampilkan atribut dari shapefile yang kita akses, kita cukup menghapus instruksi `plot()` pada kode di atas. Sebagaimana contoh kode berikut:

```
import geopandas as gpd
from google.colab import drive

drive.mount('/content/gdrive')

tanah_laut =
gpd.read_file('/content/gdrive/MyDrive/geebook/vector/Tanah_Laut.shp')
```

Output:

	DESA	KECAMATAN	KABUPATEN	PROVINSI	PENDUDUK	LUAS_KM2	geometry
0	Alur	Jorong	Tanah Laut	Kalimantan Selatan	2372	78.80470	POLYGON ((271590.646 9563803.636, 269045.48 95...
1	Ambawang	Batu Ampar	Tanah Laut	Kalimantan Selatan	1978	14.59340	POLYGON ((268744.562 9566662.943, 269243.484 9...
2	Ambungan	Pelaihari	Tanah Laut	Kalimantan Selatan	2154	53.27730	POLYGON ((258448.58 9588775.597, 258415.47 958...
3	Angsau	Pelaihari	Tanah Laut	Kalimantan Selatan	13024	9.28098	POLYGON ((252993.724 9581736.308, 253001.5 958...
4	Asam-Asam	Jorong	Tanah Laut	Kalimantan Selatan	5753	125.39900	POLYGON ((285955.954 9590588.894, 286544.582 9...
...	...	...	...	...	...	...	...
130	Tirta Jaya	Bajuin	Tanah Laut	Kalimantan Selatan	2705	4.16950	POLYGON ((258285.275 9579862.661, 258283.632 9...
131	Tungkaran	Pelaihari	Tanah Laut	Kalimantan Selatan	824	15.41550	POLYGON ((245967.178 9584750.386, 246500.561 9...
132	Ujung	Bati Bati	Tanah Laut	Kalimantan Selatan	2956	16.71400	POLYGON ((245584.756 9605583.267, 245651.098 9...
133	Ujung Baru	Bati Bati	Tanah Laut	Kalimantan Selatan	2693	8.22303	POLYGON ((250700.436 9602270.457, 250189.261 9...
134	Ujung Batu	Pelaihari	Tanah Laut	Kalimantan Selatan	3022	36.51430	POLYGON ((248163.705 9590093.17, 248137.97 958...

135 rows × 7 columns

Untuk selanjutnya, proses akses dan pembaharuan data untuk data atribut adalah sama persis dengan Pandas yang sudah dibahas pada bagian sebelumnya. Seperti menggunakan metode `loc` atau `iloc`, sesuai keperluan. Sebab biar bagaimana pun juga, GeoPandas pada dasarnya adalah Pandas dengan penambahan elemen geometri.

Perhatikan data atribut shapefile di atas, dimana terdapat *field* atau kolom **geometry**. Yang menandakan bahwa data tersebut adalah GeoPandas GeoDataFrame. Kolom khusus **geometry** itu lah yang membedakan GeoPandas GeoDataFrame dengan Pandas DataFrame biasa.

Sebagai contoh, untuk menampilkan data atribut sebuah kecamatan saja, yaitu Kecamatan Tambang Ulang, digunakan instruksi seperti berikut:

```
tanah_laut.loc[(tanah_laut['KECAMATAN'] == 'Tambang ulang')]
```

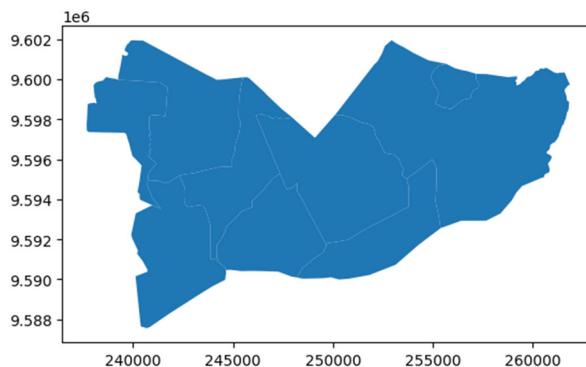
Output:

	DESA	KECAMATAN	KABUPATEN	PROVINSI	PENDUDUK	LUAS_KM2	geometry
23	Bingkulu	Tambang Ulang	Tanah Laut	Kalimantan Selatan	1821	21.65030	POLYGON ((244220.392 9590909.78, 244627.044 95...
38	Gunung Raja	Tambang Ulang	Tanah Laut	Kalimantan Selatan	2247	24.92850	POLYGON ((240054.553 9601938.359, 240376.339 9...
58	Kayu Abang	Tambang Ulang	Tanah Laut	Kalimantan Selatan	978	11.71990	POLYGON ((239252.109 9599967.972, 239378.496 9...
71	Martadah	Tambang Ulang	Tanah Laut	Kalimantan Selatan	1768	56.98260	POLYGON ((255686.026 9600757.126, 255571.826 9...
72	Martadah Baru	Tambang Ulang	Tanah Laut	Kalimantan Selatan	2711	3.56281	POLYGON ((257174.132 9600264.384, 257165.168 9...
93	Pulau Sari	Tambang Ulang	Tanah Laut	Kalimantan Selatan	2579	16.75550	POLYGON ((247969.957 9598075.671, 247733.519 9...
112	Sungai Jelai	Tambang Ulang	Tanah Laut	Kalimantan Selatan	2436	15.78090	POLYGON ((255392.097 9592556.034, 255286.654 9...
113	Sungai Pinang	Tambang Ulang	Tanah Laut	Kalimantan Selatan	1897	29.96600	POLYGON ((247969.957 9598075.671, 248230.134 9...
123	Tambang Ulang	Tambang Ulang	Tanah Laut	Kalimantan Selatan	2319	17.14960	POLYGON ((249635.38 9591785.597, 249650.063 95...

Untuk menampilkan geometri shapefile hasil seleksi data, kita tinggal menambahkan metode `plot()` pada instruksi di atas. Sebagaimana contoh berikut:

```
tanah_laut.loc[(tanah_laut['KECAMATAN'] == 'Tambang ulang')].plot()
```

Output:



Kita dapat menyimpan data hasil seleksi sebuah kecamatan di atas dengan instruksi berikut:

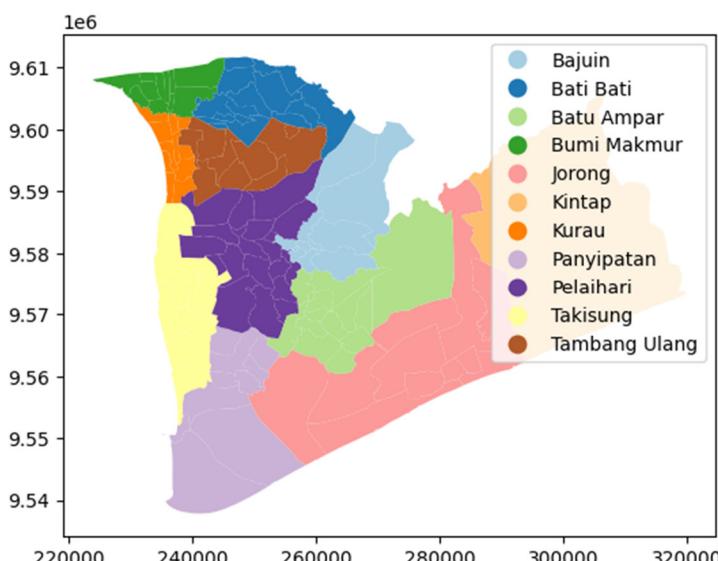
```
tbg_ulang = tanah_laut.loc[(tanah_laut['KECAMATAN'] == 'Tambang Ulang')]  
tbg_ulang.to_file('/content/gdrive/MyDrive/geebook/vector/Tambang_Ulang.shp')
```

### Visualisasi Data Geospasial

Untuk memvisualisasikan data geospasial agar lebih informatif, terutama simbologinya, digunakan instruksi seperti berikut:

```
import geopandas as gpd  
from google.colab import drive  
drive.mount('/content/gdrive')  
  
tanah_laut =  
gpd.read_file('/content/gdrive/MyDrive/geebook/vector/Tanah_Laut.shp')  
tanah_laut.plot(column='KECAMATAN', legend=True, cmap='Paired')
```

Output:



Instruksi `column='KECAMATAN'` digunakan untuk memilih atau menentukan kolom **KECAMATAN** sebagai dasar pewarnaan unik. Sementara instruksi `cmap='Paired'` digunakan untuk memiliki skema pewarnaan, mengacu ke kode warna Matplotlib colormaps di laman <https://matplotlib.org/stable/users/explain/colors/colormaps.html>.

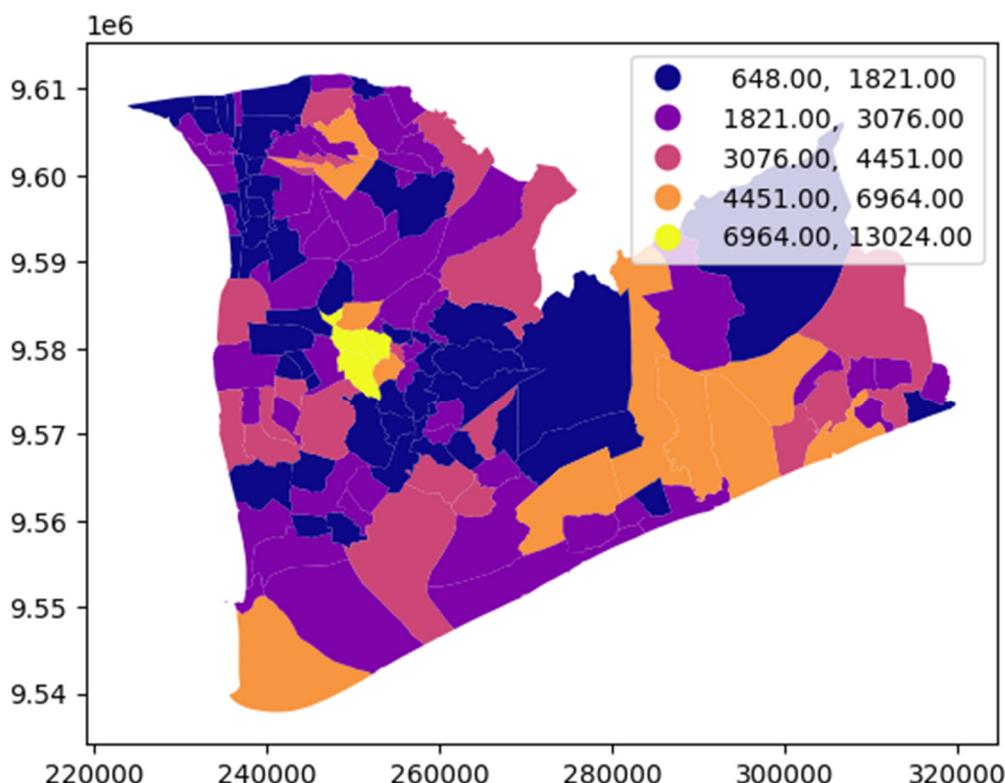
Untuk membuat *choropleth*, yaitu visualisasi data geospasial kuantitatif dengan gradasi warna, dapat digunakan instruksi sebagaimana contoh kode berikut:

```
import geopandas as gpd
from google.colab import drive

drive.mount('/content/gdrive')

tanah_laut =
gpd.read_file('/content/gdrive/MyDrive/geebook/vector/Tanah_Laut.shp')
tanah_laut.plot(column='PENDUDUK', legend=True, cmap='plasma',
scheme='natural_breaks')
```

Output:



Catatan, jika visualisasi *choropleth* menggunakan *scheme* tidak bekerja, maka kita harus menginstall librari **mapclassify** terlebih dahulu.

Untuk Google Colab instruksinya adalah sebagai berikut:

```
!pip install mapclassify
```

## Bab II Fundamental Python

Untuk Anaconda Prompt instruksinya adalah sebagai berikut:

```
conda install anaconda::mapclassify
```

### Visualisasi Interaktif

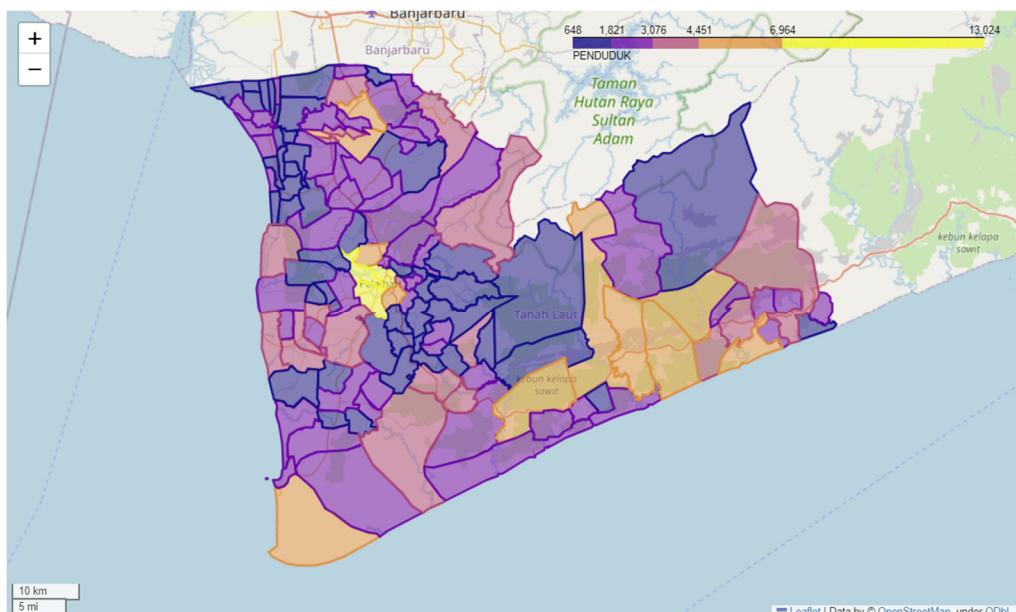
Untuk menampilkan visualisasi data geospasial secara interaktif, kita tinggal mengganti metode `plot` pada kode di atas dengan metode `explore`. Sebagaimana contoh kode berikut:

```
import geopandas as gpd
from google.colab import drive

drive.mount('/content/gdrive')

tanah_laut =
gpd.read_file('/content/gdrive/MyDrive/geebook/vector/Tanah_Laut.shp')
tanah_laut.explore(column='PENDUDUK', legend=True, cmap='plasma',
scheme='natural_breaks')
```

Output:



Tentu saja visualisasi interaktif di atas lebih menarik, sebab dapat *zoom in* dan *zoom out* sesuai keperluan sebagaimana kita menggunakan aplikasi Google Maps. Serta data atributnya dapat ditampilkan secara interaktif menggunakan panah *mouse*.

### Analisis Geospasial Dasar

Dengan menggunakan GeoPandas, kita dapat melakukan berbagai analisis atau *geoprocessing* dasar. Seperti *buffer*, *overlay*, *dissolve*, dan sebagainya. Sebagaimana ketika kita menggunakan perangkat lunak SIG seperti ArcGIS, QGIS, dan sebagainya. Memang sampai dengan saat ini, tool-analisis GeoPandas tidak selengkap perangkat lunak SIG yang biasa kita gunakan. Akan tetapi, GeoPandas tentu saja akan selalu berkembang untuk membenahi fasilitas-fasilitasnya.

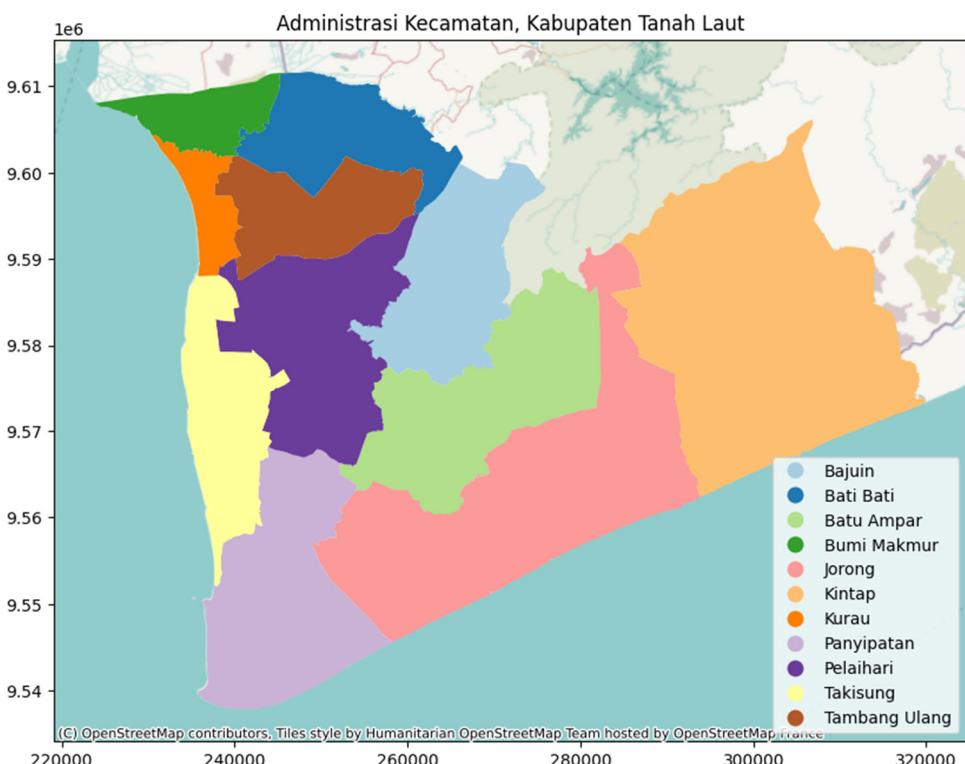
Berikut adalah contoh kode untuk analisis *dissolve* menggunakan GeoPandas:

```
import geopandas as gpd
import contextily as cx
from google.colab import drive

drive.mount('/content/gdrive')

tanah_laut =
gpd.read_file('/content/gdrive/MyDrive/geebook/vector/Tanah_Laut.shp')
kecamatan = tanah_laut.dissolve(by='KECAMATAN', as_index=False)
layout = kecamatan.plot(figsize=(10, 10), column='KECAMATAN', legend=True,
legend_kwds={'loc': 'lower right'}, cmap='Paired')
layout.set_title('Administrasi Kecamatan, Kabupaten Tanah Laut')
cx.add_basemap(layout, crs=kecamatan.crs)
```

Output:



Contoh kode di atas digunakan untuk melakukan *dissolve* menggunakan *field* atau kolom **KECAMATAN** (`by='KECAMATAN'`). Untuk melakukan *dissolve* pada kolom lain misalnya, tentu saja Anda harus mengganti entri '**KECAMATAN**' dengan nama kolom lainnya. Adapun `as_index=False` digunakan agar kolom **KECAMATAN** tidak menjadi indeks di dalam GeoDataFrame. Secara bawaan, `as_index` selalu bernilai `True`.

Jika diperlukan, data hasil analisis seperti *dissolve* di atas dapat disimpan ke dalam sebuah shapefile baru. Instruksi yang digunakan adalah sebagai berikut:

```
kecamatan.to_file('/content/gdrive/MyDrive/geebook/vector/Kecamatan_Tanah_Laut.shp', crs='EPSG:32750')
```

## Bab II Fundamental Python

---

Instruksi seperti di atas akan membuat sebuah shapefile baru dengan nama **Kecamatan\_Tanah\_Laut.shp**. Dimana **EPSG:32750** adalah kode EPSG untuk Sistem Koordinat Universal Transverse Mercator (UTM) Zone 50S dengan datum geodetik WGS 1984.

Pada visualisasi data geospasial di atas juga ditambahkan parameter-parameter baru untuk mempercantik layout. Diantaranya adalah argumen `figsize=(10, 10)` yang digunakan untuk mengatur ukuran plot, yaitu lebar dan tinggi plot masing-masing 10 inch. Dan argumen dictionary `legend_kwds={'loc': 'lower right'}` yang digunakan untuk memindahkan posisi legenda ke kanan bawah. Sementara metode `set_title` digunakan untuk memberi judul pada plot.

Contextily merupakan librari yang digunakan untuk menambahkan latar belakang peta dasar (*basemap*) pada plot visualisasi data geospasial. Pada contoh kode di atas, contextily mengambil sistem koordinat data geospasial yang digunakan sebagai referensi (`crs=kecamatan.crs`). Dan sebagaimana visualisasi interaktif sebelumnya, contextily menggunakan *Open Street Map* sebagai peta dasar bawaan. Tentu saja, Anda dapat mengganti peta dasar ini sesuai keinginan. Buka laman [https://contextily.readthedocs.io/en/latest/intro\\_guide.html](https://contextily.readthedocs.io/en/latest/intro_guide.html) untuk memilih peta dasar contextily yang Anda inginkan, berikut mempelajari petunjuk penggunaannya.

Catatan, jika librari contextily tidak berfungsi di dalam Google Colab, JupyterLab, atau VS Code, maka harus diinstal terlebih dahulu. Untuk Google Colab instruksinya adalah sebagai berikut:

```
!pip install contextily
```

Untuk Anaconda Prompt instruksinya adalah sebagai berikut:

```
conda install contextily
```

Tentu saja, beberapa elemen lain masih dapat ditambahkan ke dalam visualisasi data geospasial di atas, seperti *scale bar*, *north arrow*, dan sebagainya. Silahkan Anda mencari referensinya sendiri yang banyak tersedia secara online. Pada praktiknya, untuk lebih mempercantik visualisasi informasi geospasial, GeoPandas sering dikombinasikan dengan Matplotlib (<https://matplotlib.org>), Folium (<https://python-visualization.github.io/folium>), CartoPy (<https://scitools.org.uk/cartopy>), Seaborn (<https://seaborn.pydata.org>), Bokeh (<https://bokeh.org>), bahkan visualisasi 3-Dimensi menggunakan PyVista (<https://pyvista.org>). Dimana librari-librari ini tidak dibahas di dalam buku ini. Silahkan Anda menelusuri lamannya masing-masing untuk mencari informasinya lebih jauh.

---

*Di dalam pemrosesan citra digital penginderaan jauh, GeoPandas nantinya akan banyak berperan ketika kita menentukan lokasi atau melakukan pemotongan citra yang akan dianalisis. GeoPandas juga digunakan ketika kita melakukan analisis yang melibatkan integrasi data vektor dengan citra digital, seperti membangun model regresi antara indeks-indeks vegetasi dan data hasil survei lapangan.*

---

## G. Matplotlib

Matplotlib (<https://matplotlib.org>) (Hunter, 2007) merupakan librari untuk visualisasi grafis paling populer di lingkungan Python. Matplotlib biasa digunakan untuk memvisualisasikan data numerik ke dalam bentuk grafik (*chart*), termasuk tentu saja visualisasi informasi geospasial seperti shapefile atau citra digital penginderaan jauh. Pada bagian terdahulu kita sudah menggunakan Matplotlib untuk mensimulasikan citra digital. Untuk mempelajari Matplotlib secara singkat, Anda dapat mengakses sendiri *cheatsheet* dan *handout* resmi Matplotlib di laman <https://matplotlib.org/cheatsheets>.

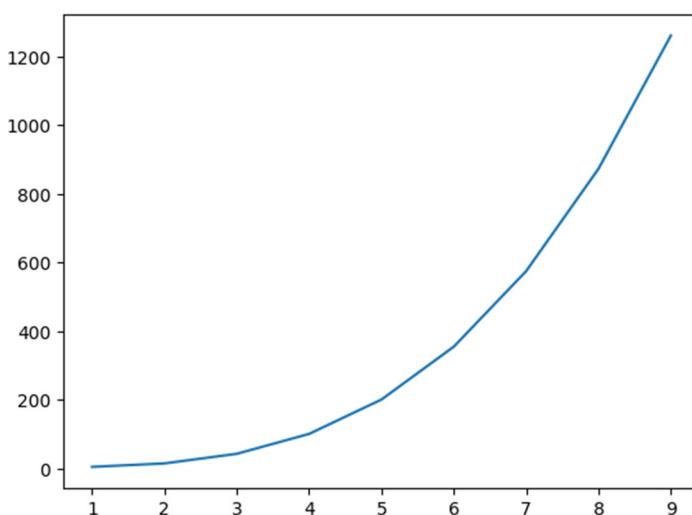
Matplotlib sudah terintegrasi di lingkungan Google Colab. Untuk di lingkungan Anaconda Prompt, Matplotlib dapat diinstal dengan instruksi berikut:

```
conda install conda-forge::matplotlib
```

Berikut adalah sebuah contoh kode paling sederhana untuk memplotkan grafik dari persamaan kuadrat  $y = 2x^3 + 3x^2 + 5x + 1$  menggunakan Matplotlib:

```
import matplotlib.pyplot as plt
x = np.arange(1,10)
y = 2*x***3 - 3*x***2 + 5*x + 1
plt.plot(x,y)
```

Output:



Berikut adalah contoh Matplotlib yang lebih kompleks untuk memvisualisasikan gambar ke dalam bentuk multiplot:

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_sample_image
china = load_sample_image('china.jpg')
```

```
flower = load_sample_image('flower.jpg')
fig, ax = plt.subplots(figsize=(8,6), nrows=2, ncols=2)

ax[0,0].imshow(china)
ax[0,0].axis('off')
ax[0,0].set_title('China 1', fontsize=12)
ax[0,1].imshow(flower)
ax[0,1].axis('off')
ax[0,1].set_title('Flower 1', fontsize=12)
ax[1,0].imshow(flower)
ax[1,0].axis('off')
ax[1,0].set_title('Flower 2', fontsize=12)
ax[1,1].imshow(china)
ax[1,1].axis('off')
ax[1,1].set_title('China 2', fontsize=12)

plt.suptitle('China and Flower', fontsize=16)
plt.show()
```

Output:

China and Flower

China 1



Flower 1



Flower 2



China 2



Kode di atas menggunakan sampel gambar JPEG dari dataset Scikit-Learn (from sklearn.datasets import load\_sample\_image). Pada contoh kode program di atas, plot yang ditampilkan adalah multiplot atau *multi axes*. Dengan 2 baris (nrows=2) dan 2 kolom (ncols=2), dan ukuran plotnya secara keseluruhan adalah lebar 8 inch dan tinggi 6 inch (figsize=(8,6)). Axes pada Matplotlib akan tersusun menyerupai list atau NumPy array. ax[0,0] berarti untuk axes paling kiri atas, demikian seterusnya hingga ax[1,1] paling kanan bawah. Tentu saja, jumlah dan indeks axes akan tergantung dari parameter nrows dan ncols. Metode axis('off') digunakan untuk menghilangkan grid sumbu pada gambar, dimana grid sumbu ini biasanya hanya diperlukan untuk grafik, bukan untuk foto. set\_title('China 1', fontsize=12) digunakan untuk mengatur judul, termasuk ukuran huruf, untuk setiap axes. Sementara plt.suptitle('China and Flower', fontsize=16) digunakan untuk mengatur *super title*, yaitu judul utama untuk plot.

Berikut adalah sebuah contoh bar plot berwarna:

```
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt

from google.colab import drive
drive.mount('/content/gdrive')
tanah_laut =
gpd.read_file('/content/gdrive/MyDrive/geebook/vector/Tanah_Laut.shp')
tanah_laut.head(5)
```

Output:

	DESA	KECAMATAN	KABUPATEN	PROVINSI	PENDUDUK	LUAS_KM2	geometry
0	Alur	Jorong	Tanah Laut	Kalimantan Selatan	2372	78.80470	POLYGON ((271590.646 9563803.636, 269045.48 95...
1	Ambawang	Batu Ampar	Tanah Laut	Kalimantan Selatan	1978	14.59340	POLYGON ((268744.562 9566662.943, 269243.484 9...
2	Ambungan	Pelaihari	Tanah Laut	Kalimantan Selatan	2154	53.27730	POLYGON ((258448.58 9588775.597, 258415.47 958...
3	Angsau	Pelaihari	Tanah Laut	Kalimantan Selatan	13024	9.28098	POLYGON ((252993.724 9581736.308, 253001.5 958...
4	Asam-Asam	Jorong	Tanah Laut	Kalimantan Selatan	5753	125.39900	POLYGON ((285955.954 9590588.894, 286544.582 9...

Lanjutkan kode di atas dengan kode berikut di kolom baru Google Colab:

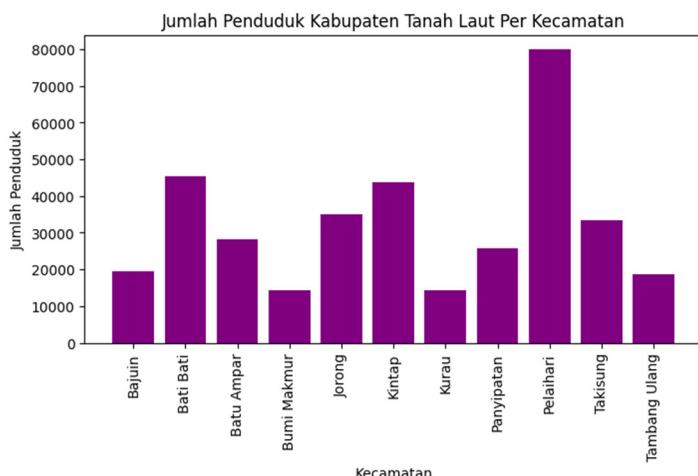
```
tala_kec = tanah_laut.dissolve(by='KECAMATAN', aggfunc={'PENDUDUK': 'sum',
'LUAS_KM2': 'sum'}, as_index=False)

kecamatan = tala_kec.KECAMATAN.to_numpy()
penduduk = tala_kec.PENDUDUK.to_numpy()
luas = tala_kec.LUAS_KM2.to_numpy()

fig, ax = plt.subplots(figsize=(8,4))

ax.bar(kecamatan, penduduk, label=kecamatan, color='purple')
ax.set_xlabel('Kecamatan')
ax.set_ylabel('Jumlah Penduduk')
ax.set_title('Jumlah Penduduk Kabupaten Tanah Laut Per Kecamatan')
plt.xticks(rotation=90)
plt.show()
```

Output:



## Bab II Fundamental Python

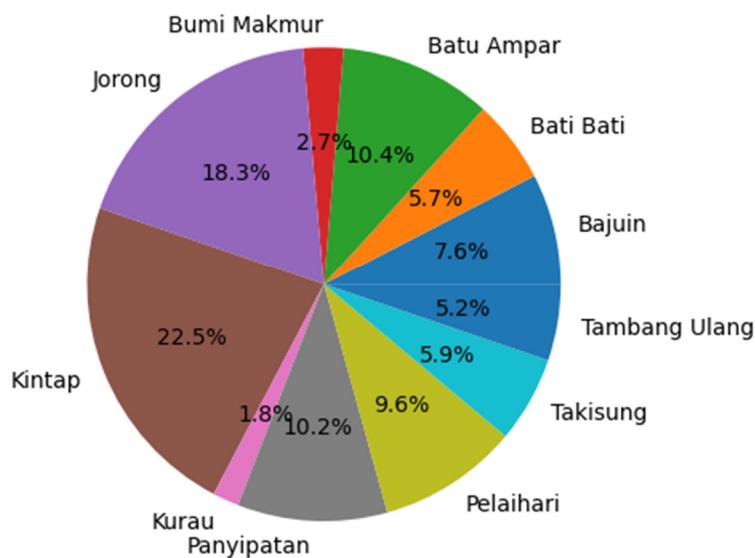
Pada kode di atas, GeoDataFrame `tanah_laut` yang merupakan sebuah shapefile didissolve menggunakan field KECAMATAN, sekaligus dengan summary (menjumlahkan) PENDUDUK dan LUAS\_KM2 menurut KECAMATAN (`aggfunc={'PENDUDUK': 'sum', 'LUAS_KM2': 'sum'}`). Instruksi `plt.xticks(rotation=90)` digunakan untuk memutar label sumbu X (nama-nama kecamatan) 90°, hal ini karena nama-nama kecamatannya cukup panjang. Selanjutnya, data PENDUDUK divisualisasikan ke dalam bentuk bar plot. Sementara data LUAS\_KM2 divisualisasikan ke dalam pie plot pada contoh kode program di bawah.

Pada contoh kode di atas, tambahkan kode berikut pada kolom baru di Google Colab:

```
fig, ax = plt.subplots()
ax.pie(luas, labels=kecamatan, autopct='%1.1f%%')
ax.set_title('Persentase Luas Wilayah Kecamatan, Kabupaten Tanah Laut')
plt.show()
```

Output:

Persentase Luas Wilayah Kecamatan, Kabupaten Tanah Laut



Jika diperlukan plot dapat disimpan dengan menggunakan instruksi berikut:

```
plt.savefig('/lokasi_file/nama_file.jpg')
```

Selain plot-plot 2-Dimensi seperti contoh-contoh di atas, dengan Matplotlib kita juga dapat membuat plot-plot 3-Dimensi yang lebih menarik. Silahkan Anda mempelajari konsepnya dan melihat contoh-contohnya di laman <https://matplotlib.org/stable/gallery/mplot3d/index.html>. Tentu saja, tidak sembarang data dapat diplotkan ke dalam visualisasi 3-Dimensi. Datanya sendiri harus menyediakan informasi hingga dimensi ketiga. Misalnya terdapat data koordinat XY dan data Z (elevasi, kedalaman, temperatur, curah hujan, biomassa, dan sebagainya). Matplotlib juga sering dikombinasikan dengan Seaborn (<https://seaborn.pydata.org>) untuk plotting data statistik.

## H. GDAL

*The Geospatial Data Abstraction Library (GDAL)* (GDAL/OGR contributors, 2024) adalah librari penerjemah untuk format data geospasial raster dan vektor yang dirilis di bawah *MIT style Open Source License* oleh *Open Source Geospatial Foundation* (<https://gdal.org>). GDAL mendukung hampir seluruh format data raster geospasial yang umum digunakan. GDAL adalah semacam “core of the core” dari sebagian besar perangkat lunak geospasial. Seperti ArcGIS, QGIS, ENVI, ERDAS, GRASS GIS, gvSIG, Orfeo Toolbox, SAGA GIS, Google Earth, GeoPandas, dan sebagainya. Dengan kata lain, di dalam perangkat lunak-perangkat lunak populer ini, terdapat GDAL yang bertugas sebagai antarmuka untuk menangani interaksi perangkat lunak dengan berbagai format data raster dan vektor. Di lingkungan Google Colab, GDAL dapat diinstal dengan instruksi berikut:

```
!pip install gdal
```

Di lingkungan Anaconda Prompt GDAL diinstal dengan instruksi berikut:

```
conda install conda-forge::gdal
```

### Membaca dan Menulis Data Raster Menggunakan GDAL

Untuk membaca dan menulis citra digital penginderaan jauh dalam format GeoTIFF dengan menggunakan GDAL pada umumnya digunakan instruksi-instruksi sebagai berikut:

```
import numpy as np
import matplotlib.pyplot as plt

from osgeo import gdal
from skimage.exposure import equalize_hist
from google.colab import drive

drive.mount('/content/gdrive')

path = '/content/gdrive/MyDrive/geebook/imagery/'

# Reading image files
print('Reading image files...')
s2_image = gdal.Open(path + 'S2_Muara_Barito.tif')

# Reading image parameters
img_proj = s2_image.GetProjection()
geotransform = s2_image.GetGeoTransform()
img_height = s2_image.RasterYSize
img_width = s2_image.RasterXSize
img_band = s2_image.RasterCount

# Reading image bands and arrays
image = np.empty((img_height, img_width, img_band))

for i in range(img_band):
    band = s2_image.GetRasterBand(i+1)
    image[:, :, i] = band.ReadAsArray()

# Calculating MNDWI
mndwi = (image[:, :, 2] - image[:, :, 1]) / (image[:, :, 2] + image[:, :, 1])
water = np.zeros((img_height, img_width), dtype=np.ubyte)
```

```
water[np.where(mndwi > 0)] = 1
# writing image files
print('Writing image to file...')

image_name = path + 'water.tif'
image_raster = gdal.GetDriverByName("GTiff").Create(image_name, img_width,
img_height, 1, gdal.GDT_Byte)
image_raster.GetRasterBand(1).writeArray(water)

image_raster.SetGeoTransform(geotransform)
image_raster.SetProjection(img_proj)
image_raster.FlushCache()

image_raster = None
# Output visualization
print('Starting visualization...')

fig, ax = plt.subplots(figsize=(12,4), nrows=1, ncols=3)
rgb = np.dstack((image[:, :, 3], image[:, :, 2], image[:, :, 1]))

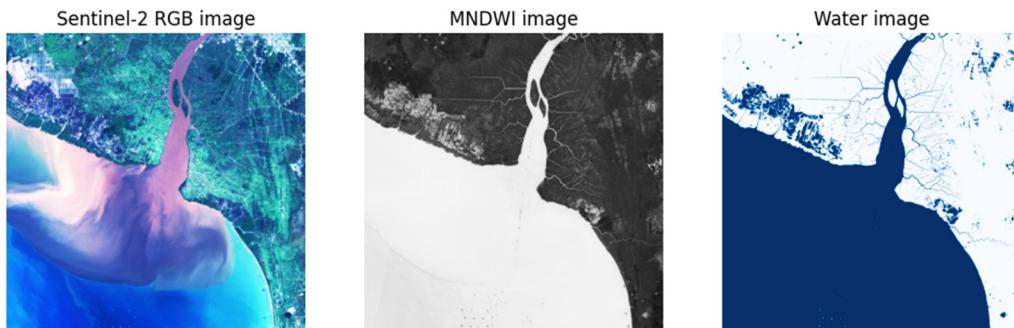
ax[0].imshow(equalize_hist(rgb))
ax[0].set_title('Sentinel-2 RGB image')
ax[0].axis('off')

ax[1].imshow(mndwi, cmap='gray')
ax[1].set_title('MNDWI image')
ax[1].axis('off')

ax[2].imshow(water, cmap='Blues')
ax[2].set_title('Water image')
ax[2].axis('off')

plt.show()
```

Output:



Kode di atas akan mengakses sebuah citra multispektral, yaitu Sentinel-2 MSI, dalam format GeoTIFF. Kemudian melakukan transformasi *Modified Normalized Difference Water Index* (MNDWI). Dimana formula MNDWI adalah sebagai berikut (Xu, 2006):

$$\text{MNDWI} = \frac{\text{Green} - \text{SWIR1}}{\text{Green} + \text{SWIR1}}$$

Selanjutnya melakukan pemisahan antara tubuh air dan bukan tubuh air di atas Citra MNDWI, dengan kriteria nilai MNDWI lebih dari 0 adalah fitur air. Citra hasil ekstraksi fitur air ini kemudian disimpan ke dalam file GeoTIFF.

Catatan, kode program di atas diperuntukkan untuk Google Colab dan citra dibaca dan ditulis dari dan ke dalam Google Drive. Jika Anda bekerja menggunakan JupyterLab/VS Code secara offline, maka perlu penyesuaian *path* sebagai berikut:

```
path = 'D:/geebok/imagery/'
```

Path di atas tentu saja dengan asumsi data Anda berada di drive D, jika drive-nya berbeda maka Anda harus menyesuaikan. Dan Anda harus menghapus instruksi `from google.colab import drive` dan `drive.mount('/content/gdrive')`.

Elemen yang terpenting untuk diperhatikan adalah bahwa pada kode program di atas, GDAL akan mengembalikan citra dalam bentuk NumPy array dengan dimensi **[jumlah baris, jumlah kolom, jumlah band]**. Sehingga untuk mengambil band 3 dari citra misalnya, kodennya adalah `image[:, :, 2]`. Ingat kembali, bahwa indeks NumPy array selalu dimulai dari 0. Sehingga indeks 2 merupakan band 3. Dan karena Citra Sentinel-2 yang digunakan di atas lengkap 13 band, maka *green band* adalah `image[:, :, 2]` dan *SWIR1 band* adalah `image[:, :, 11]`.

## I. Rasterio

Rasterio (<https://rasterio.readthedocs.io>) (Gillies et al., 2019) merupakan librari Python lain yang dapat digunakan untuk mengakses dan melakukan pemrosesan citra penginderaan jauh. Tentu saja, di dalam Rasterio sebenarnya terdapat GDAL juga, sebagaimana perangkat lunak geospasial lainnya. Akan tetapi, dibandingkan dengan GDAL, Rasterio lebih efisien dalam penulisan kode program. Sehingga bagi programer pemula yang ingin menganalisis citra penginderaan jauh di lingkungan Python, Rasterio relatif lebih mudah untuk dipelajari dibanding GDAL. Di lingkungan Google Colab, Rasterio dapat diinstal dengan instruksi berikut:

```
!pip install rasterio
```

Di lingkungan Anaconda Prompt Rasterio diinstal dengan instruksi berikut:

```
conda install anaconda::rasterio
```

Kode Rasterio berikut memiliki output yang sama persis dengan kode GDAL sebelumnya:

```
import rasterio
import numpy as np
import matplotlib.pyplot as plt

from google.colab import drive
from skimage.exposure import equalize_hist
from google.colab import drive

drive.mount('/content/gdrive')

path = '/content/gdrive/MyDrive/geebok/imagery/'

# Reading image file
print('Reading image files...')
with rasterio.open(path + 'S2_Muara_Barito.tif') as s2_image:

    img_profile = s2_image.profile
    img_width = s2_image.width
```

## Bab II Fundamental Python

```
img_height = s2_image.height
img_band = s2_image.count

image = np.empty((img_height, img_width, img_band))

for i in range(img_band):
    image[:, :, i] = s2_image.read(i+1)

# Calculating MNDWI

mndwi = (image[:, :, 2] - image[:, :, 11]) / (image[:, :, 2] + image[:, :, 11])
water = np.zeros((img_height, img_width), dtype=np.ubyte)
water[np.where(mndwi > 0)] = 1

# writing image files

print('Writing image to file...')

img_profile.update(count=1)

with rasterio.open(path + 'water.tif', 'w', **img_profile) as img_file:
    img_file.write(water, 1)
    img_file.close()

# output visualization

print('Starting visualization...')

fig, ax = plt.subplots(figsize=(12, 4), nrows=1, ncols=3)

rgb = np.dstack((image[:, :, 3], image[:, :, 2], image[:, :, 1]))

ax[0].imshow(equalize_hist(rgb))
ax[0].set_title('Sentinel-2 RGB image')
ax[0].axis('off')

ax[1].imshow(mndwi, cmap='gray')
ax[1].set_title('MNDWI image')
ax[1].axis('off')

ax[2].imshow(water, cmap='Blues')
ax[2].set_title('Water image')
ax[2].axis('off')

plt.show()
```

Rasterio tampak sedikit lebih singkat di dalam penulisan kode programnya. Terutama pada saat membaca citra digital dari file dan menulis citra digital ke dalam file. Dan sebagaimana GDAL, Rasterio akan mengembalikan citra menjadi NumPy array dengan format yang sama persis. Rasterio memiliki modul petunjuk penggunaan yang sangat lengkap yang dapat diakses di laman [https://rasterio.readthedocs.io/\\_/downloads/en/stable/pdf](https://rasterio.readthedocs.io/_/downloads/en/stable/pdf). Silahkan Anda mengakses sendiri tutorialnya untuk mempelajari Rasterio secara lebih komprehensif.

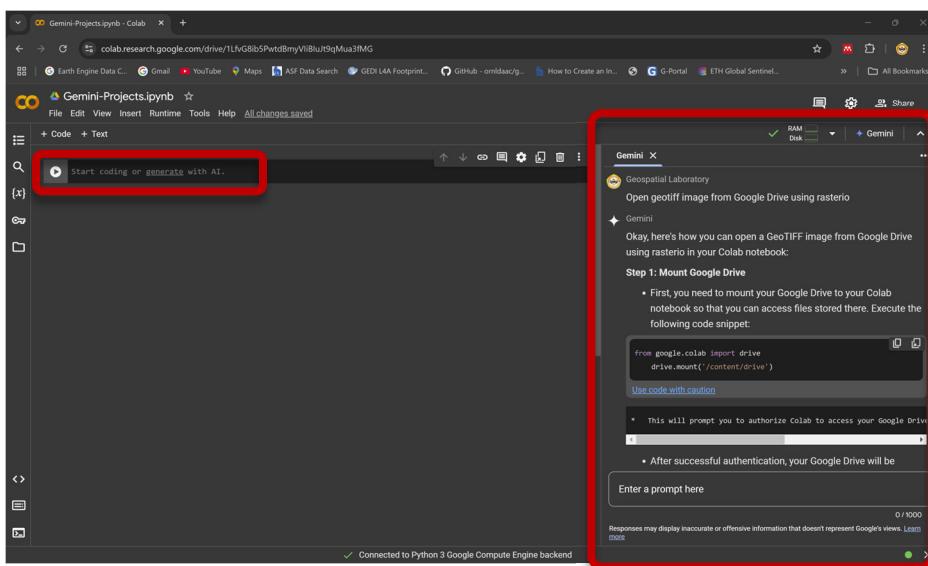
Baik dengan menggunakan GDAL maupun Rasterio, sebuah citra satelit digital multispektral seperti Sentinel-2 MSI dapat dirubah menjadi NumPy array tanpa kehilangan seluruh metadata geospasialnya, seperti proyeksi, koordinat, resolusi spasial, dan sebagainya. Teknik merubah citra digital menjadi array pixel per pixel seperti ini sangat diperlukan ketika kita ingin mengambil kontrol sepenuhnya di dalam manipulasi nilai-nilai pixel di lingkungan bahasa pemrograman seperti Python. Dan tentu saja, teknik ini merupakan dasar bagi algoritma-algoritma pemrosesan yang lebih kompleks, seperti *machine learning* atau *deep learning*.

## J. Gemini, AI-Powered Tools, Jupyter AI, dan Copilot

### Google Gemini

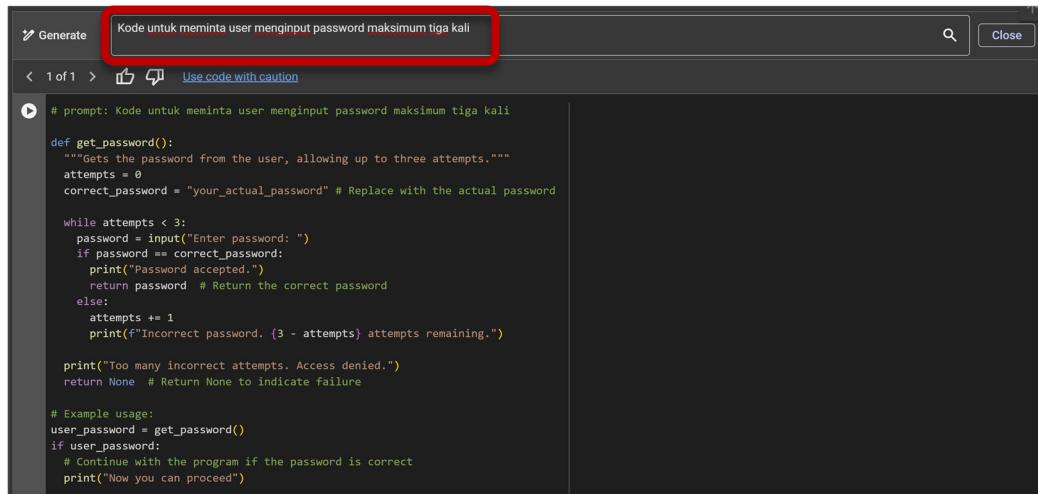
Pada saat penulisan buku ini, yaitu sekitar bulan Juni 2024, muncul fasilitas baru di dalam Google Colab, yaitu Google Gemini (<https://gemini.google.com/>). Dimana pada saat awal-awal penulisan buku ini, Google Gemini belum terintegrasi secara langsung dengan Google Colab. Itu sebabnya, jika Anda mencermati Bab I dari buku ini, khususnya pada Sub Bab E. **Google Collaboratory**, dan beberapa bagian pada Bab III dari buku ini, Anda belum melihat tombol Google Gemini pada gambar-gambar tangkapan layar Google Colab. Jika Anda sudah terbiasa menggunakan ChatGPT (<https://chatgpt.com/>), pasti Anda akan dengan mudah dapat menggunakan Gemini. Sebab Google Gemini dan ChatGPT memiliki fungsi yang sama persis. Yaitu platform *Artificial Intelligence* (AI) online yang mampu merespon pertanyaan-pertanyaan atau ide-ide Anda dalam bentuk *prompt* yang ditulis dalam bahasa sehari-hari. Integrasi Google Gemini ke dalam Google Colab memungkinkan proses penulisan kode Bahasa Python menjadi lebih efisien.

Untuk menjalankan Google Gemini di dalam Google Colab, klik tombol  Gemini di bagian kanan atas jendela Google Colab, sebagaimana terlihat pada gambar di bawah. Kemudian pada kolom **Enter a prompt here** ketikkan prompt yang Anda inginkan, misalnya “*Open geotiff image from Google Drive using rasterio*”, dan jalankan. Gemini akan memberikan kode-kode yang siap pakai sebagaimana terlihat pada gambar di bawah. Anda tinggal klik **Copy** () dan paste kode ke dalam sel kode Google Colab yang sudah ada, atau klik **Add code cell** () untuk menambahkan langsung sel kode baru berikut kode-kodenya di dalam Google Colab. Selanjutnya, eksekusi kode-kode Python yang diberikan oleh Gemini. Pada panel Gemini, tidak hanya kode-kode Python yang kita dapatkan, penjelasan tentang bagian-bagian kode pun juga diberikan oleh Gemini.



Anda juga dapat menggunakan fasilitas *AI-Powered Tools* dengan cara langsung mengetikkan prompt Anda di dalam sel-sel kode Google Colab, dengan cara mengklik **generate** pada **Start coding or generate with AI** seperti pada gambar di atas. Misalnya Anda ketikkan prompt dalam Bahasa Indonesia, “*Kode untuk meminta user menginput password maksimum tiga kali*”, maka AI-Powered Tools akan memberikan kode-kode siap pakai seperti pada gambar berikut.

## Bab II Fundamental Python



The screenshot shows a Google Colab interface. At the top, there's a search bar with the placeholder "Kode untuk meminta user menginput password maksimum tiga kali". Below the search bar, there are navigation icons and a "Close" button. The main area contains a code cell with the following Python code:

```
# prompt: Kode untuk meminta user menginput password maksimum tiga kali

def get_password():
    """Gets the password from the user, allowing up to three attempts."""
    attempts = 0
    correct_password = "your_actual_password" # Replace with the actual password

    while attempts < 3:
        password = input("Enter password: ")
        if password == correct_password:
            print("Password accepted.")
            return password # Return the correct password
        else:
            attempts += 1
            print(f"Incorrect password. {3 - attempts} attempts remaining.")

    print("Too many incorrect attempts. Access denied.")
    return None # Return None to indicate failure

# Example usage:
user_password = get_password()
if user_password:
    # Continue with the program if the password is correct
    print("Now you can proceed")
```

Meskipun keberadaan Google Gemini dan AI-Powered Tools sangat memudahkan di dalam pemrograman Python di lingkungan Google Colab, tentu saja Anda tetap dituntut untuk memahami konsep-konsep dasar Bahasa Python. Sebab kode-kode Python siap pakai yang diberikan AI terkadang harus kita sesuaikan dengan situasi, keinginan, atau kebutuhan kita. Dan berdasarkan pengalaman penulis, terkadang kode-kode yang diberikan oleh Gemini tidak seluruh bagiannya sesuai dengan harapan kita. Bahkan terkadang sebagian kode memuat *bug* atau *out of date*, terutama ketika kode-kode yang kita minta cukup kompleks. Sehingga masih memerlukan campur tangan kita sebagai programer untuk menyunting kode-kode secara manual.

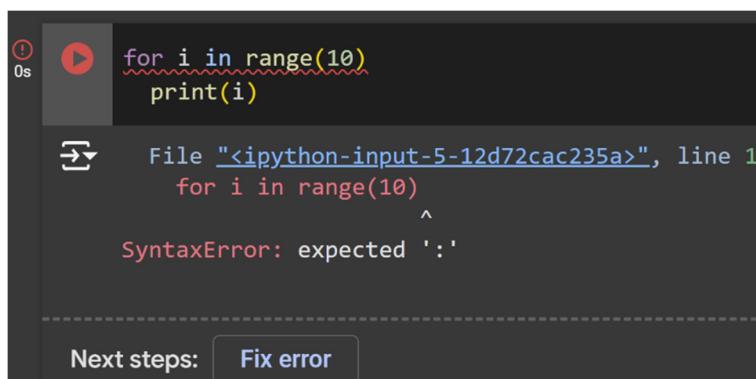
Keuntungan utama yang kita dapatkan dengan integrasi Googla Gemini dan AI-Powered Tools ke dalam Google Colab adalah kita tidak terlalu perlu untuk menghafalkan detail teknis sintaks dasar Bahasa Python, yang kita perlukan adalah memahami konsep-konsep, fungsi-fungsi, atau logika-logikanya. Misalnya, kita cukup memahami konsep apa itu instruksi **for** dan kapan instruksi **for** digunakan, tanpa perlu kita menghafal detail sintaks **for** secara teknis, seperti dimana menaruh tanda titik dua (:), indentasinya bagaimana, dan sebagainya. Nanti jika kita memerlukan instruksi **for** di dalam kode kita, kita tinggal memasukkan prompt di Google Gemini atau AI-Powered Tools untuk meminta instruksi ini.

Apakah prompt harus dalam Bahasa Inggris? Jawabannya, tidak. Pada contoh di atas digunakan prompt berbahasa Indonesia, Gemini dan AI-Powered Tools dapat memahami apa yang diminta dengan baik. Akan tetapi, berdasarkan pengalaman penulis, instruksi dalam Bahasa Inggris akan lebih efektif dari pada Bahasa Indonesia. Hal penting lainnya yang harus difahami adalah bahwa Gemini dan AI-Powered Tools akan berusaha untuk memberikan kode-kode Python secara kontekstual dengan apa yang sedang kita kerjakan. Secara teknis, Gemini dan AI-Powered Tools akan melihat kode-kode yang sudah kita tulis pada sel-sel kode sebelumnya pada file Notebook yang sama. Misalnya, pada sel-sel kode sebelumnya kita sedang mengerjakan analisis statistik untuk ekonomi, maka ketika kita memberikan prompt, Gemini atau AI-Powered Tools akan memberikan kode-kode yang terkait statistik untuk ekonomi.

### Fix error dan Explain error

Fasilitas lainnya yang juga sangat memudahkan di dalam Google Colab, terutama ketika terjadi kesalahan sintaks di dalam penulisan kode-kode Python adalah Fix error dan Explain error.

Perhatikan contoh-contoh berikut:

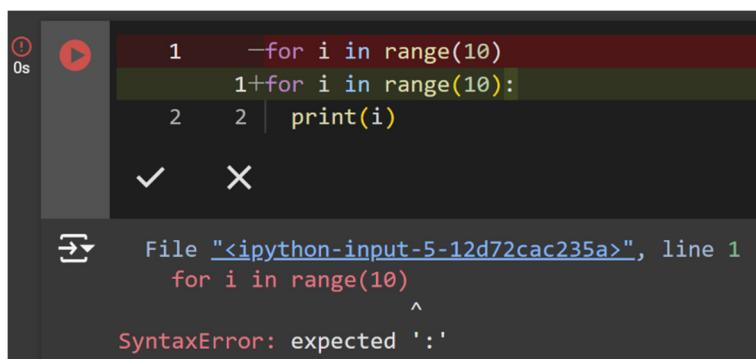


```
0s ⏴ for i in range(10)
      print(i)

→ File "<ipython-input-5-12d72cac235a>", line 1
      for i in range(10)
                  ^
SyntaxError: expected ':'
```

Next steps: **Fix error**

Pada kode di atas terjadi kesalahan sintaks, yaitu lupa menambahkan tanda titik dua (:) pada akhir baris instruksi `for`. Karena kesalahannya kecil dan mudah diperbaiki, maka Google Colab akan memunculkan tombol **Fix error**. Anda dapat langsung mengklik tombol **Fix error** dan akan ada opsi dua tombol sebagaimana gambar di bawah:

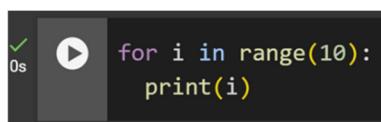


```
0s ⏴ 1 -for i in range(10)
      1+for i in range(10):
      2 2 | print(i)

✓ ✗

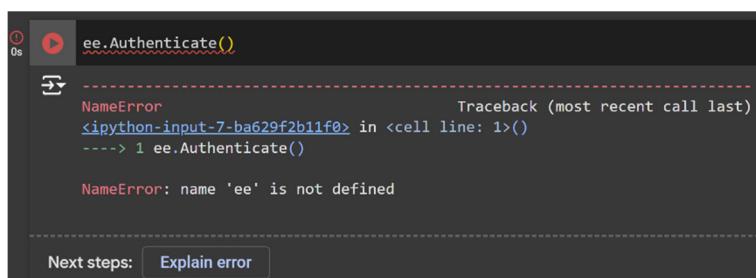
→ File "<ipython-input-5-12d72cac235a>", line 1
      for i in range(10)
                  ^
SyntaxError: expected ':'
```

Jika Anda setuju dengan rekomendasi perbaikan kode yang diberikan, langsung klik tombol **✓**. Dan hasilnya adalah seperti pada gambar berikut:



```
0s ⏴ ✓ for i in range(10):
      print(i)
```

Jika kesalahan sintaksnya cukup kompleks, biasanya yang muncul bukan tombol **Fix error**, melainkan **Explain error**. Sebagaimana pada contoh berikut:



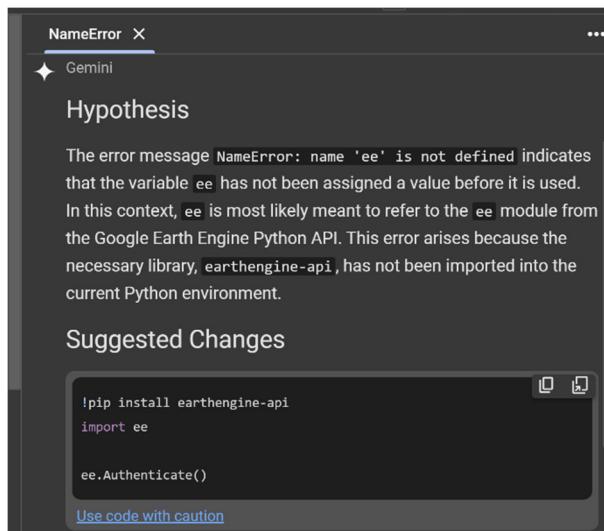
```
0s ⏴ ee Authenticate()

→ -----
          Traceback (most recent call last)
          <ipython-input-7-ba629f2b11f0> in <cell line: 1>()
          ----> 1 ee Authenticate()

NameError: name 'ee' is not defined
```

Next steps: **Explain error**

Jika Anda mengklik tombol **Explain error** seperti pada gambar di atas, maka panel Google Gemini akan tampil dan menjelaskan seputar error yang terjadi. Dan Google Gemini juga akan memberikan rekomendasi perbaikan atas kesalahan yang terjadi pada kode. Sebagaimana terlihat pada gambar berikut:



Karena kesalahan yang terjadi cukup fatal, Anda harus memperbaikinya secara manual, tidak semudah mengklik tombol **Fix error** seperti pada contoh sebelumnya. Dalam proses perbaikan kode, Anda dapat secara langsung mengikuti rekomendasi yang diberikan oleh Google Gemini, maupun Anda mencari solusinya sendiri dari sumber-sumber lain. Misalnya dari laman yang sangat terkenal, yaitu <https://stackoverflow.com/>.

### Syntax Error vs Logical Error

Ada 2 (dua) jenis kesalahan di dalam kode program, yaitu kesalahan sintaks (*syntax error*) dan kesalahan logika (*logical error*). Kesalahan sintaks merupakan kesalahan teknis di dalam penulisan pernyataan-pernyataan di dalam kode program, dimana terjadi pelanggaran terhadap aturan-aturan sintaks dari suatu bahasa pemrograman, seperti Python. Seperti lupa menyertakan tanda titik dua, lupa indentasi, lupa atau kurang tanda kurung, terjadi penjumlahan data yang bukan angka, dan sebagainya. Kesalahan sintaks pada umumnya akan lebih mudah untuk dilacak dan diperbaiki. Mengingat kesalahan sintaks di dalam kode program biasanya akan langsung terdeteksi oleh interpreter atau kompiler, dan akan menyebabkan proses eksekusi atau kompilasi kode program terhenti secara paksa. Sebagaimana dua contoh kesalahan sebelumnya.

Hal yang paling mengkhawatirkan di dalam kode-kode program yang kita bangun adalah terjadi kesalahan logika. Sebab kesalahan-kesalahan logika pada umumnya tidak terdeteksi secara langsung oleh Python interpreter, selama sintaksnya sudah benar. Kesalahan logika adalah kesalahan yang terjadi di dalam alur logika kode program yang kita bangun. Kesalahan logika biasanya baru akan kelihatan setelah kode program dieksekusi, dan hasil yang diberikan tidak seperti yang diharapkan. Contoh kesalahan logika misalnya di dalam formula matematika, bilangan yang harusnya dijumlahkan, ternyata justru kita kalikan. Tentu saja operasi matematika dan kode program akan tetap sukses berjalan, tetapi luaran dari formula matematika yang kita tulis tidak akan memberikan hasil sesuai harapan.

## Jupyter AI

Secara default, Anda bisa secara langsung menggunakan fasilitas Gemini AI via Google Colab. Akan tetapi, bagaimana jika Anda mau menggunakan fasilitas AI untuk membuat kode secara otomatis di lingkungan JupyterLab? Untuk keperluan ini, Anda dapat memanfaatkan Jupyter AI (<https://jupyter-ai.readthedocs.io/>). Dengan Jupyter AI, Anda dapat mengintegrasikan platform AI seperti Google Gemini atau ChatGPT ke dalam JupyterLab. Tentu saja, untuk menjalankan tool AI ini di dalam JupyterLab, komputer tetap harus dalam keadaan online. Sebab prompt yang Anda input akan langsung terhubung ke server dan dijawab secara langsung dari server.

Terdapat banyak model AI yang didukung oleh Jupyter AI, antara lain Gemini, OpenAI/ChatGPT, NVIDIA, dan sebagainya. Kunjungi <https://jupyter-ai.readthedocs.io/en/latest/users/index.html> untuk mendapatkan penjelasan yang lebih lengkap. Di dalam buku ini, kita akan belajar bagaimana mengintegrasikan Gemini ke dalam JupyterLab. Untuk model-model AI lainnya silahkan Anda menelusuri sendiri literatur atau teknik-tekniknya.

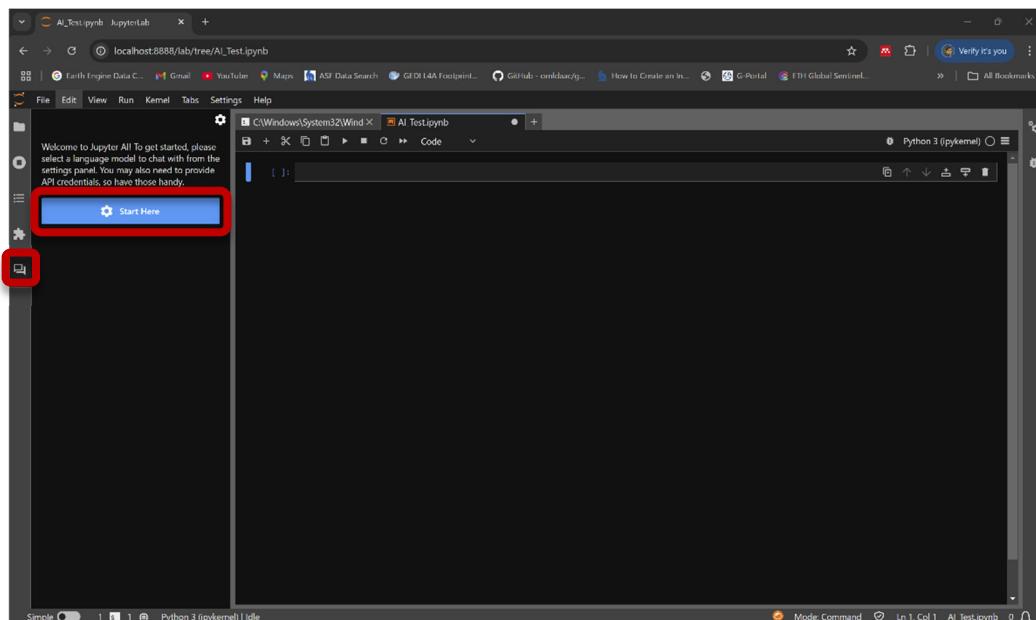
Untuk menginstal Jupyter AI, masuk terlebih dahulu ke dalam sebuah environment yang akan Anda gunakan untuk pengembangan. Misalnya, conda activate gee. Asumsinya JupyterLab sudah terinstal sebelumnya. Kemudian instal Jupyter AI dengan perintah berikut:

```
pip install jupyter-ai[all]
```

Setelah instalasi selesai, jalankan JupyterLab dengan perintah:

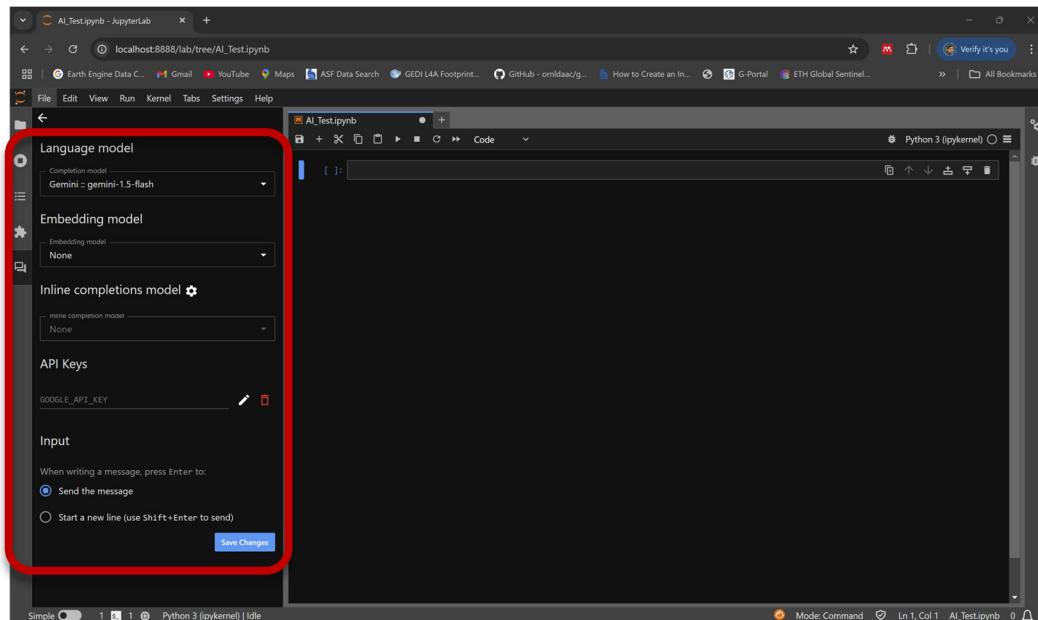
```
jupyter lab
```

Anda bisa membuat sebuah notebook baru, misalnya beri nama **AI\_Test.ipynb**. Klik tombol Jupyter AI Chat pada panel kiri, sebagaimana ditunjukkan pada gambar di bawah:

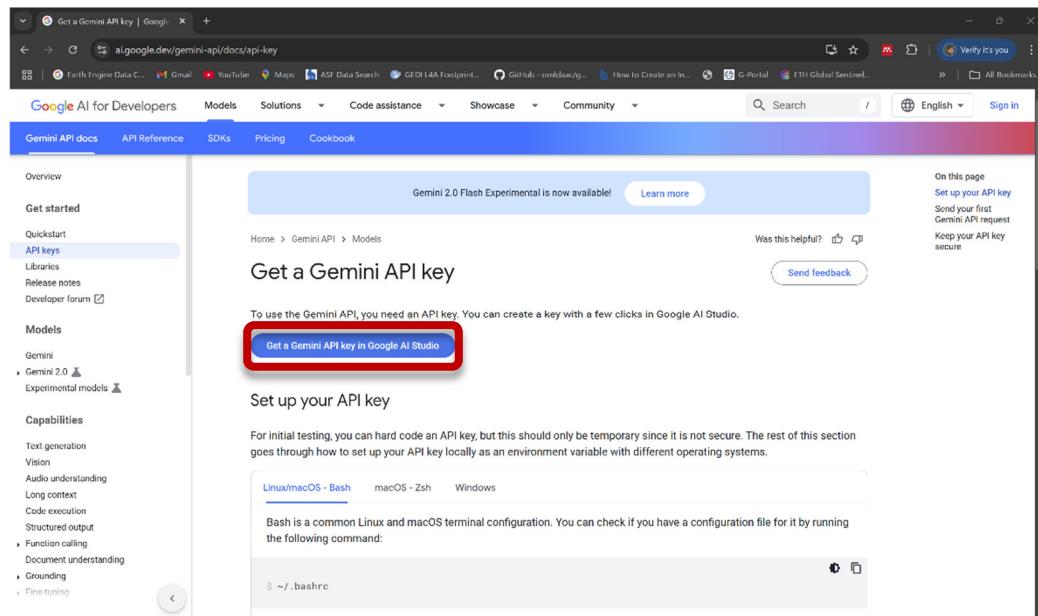


## Bab II Fundamental Python

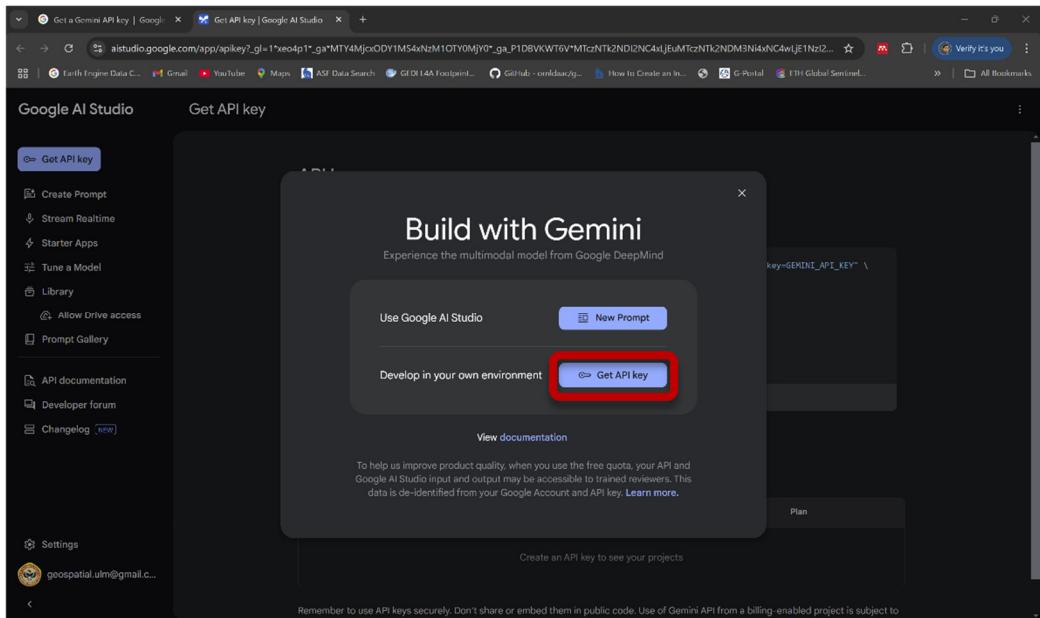
Kemudian klik tombol Start Here, panel bagian kiri akan terlihat seperti pada gambar berikut:



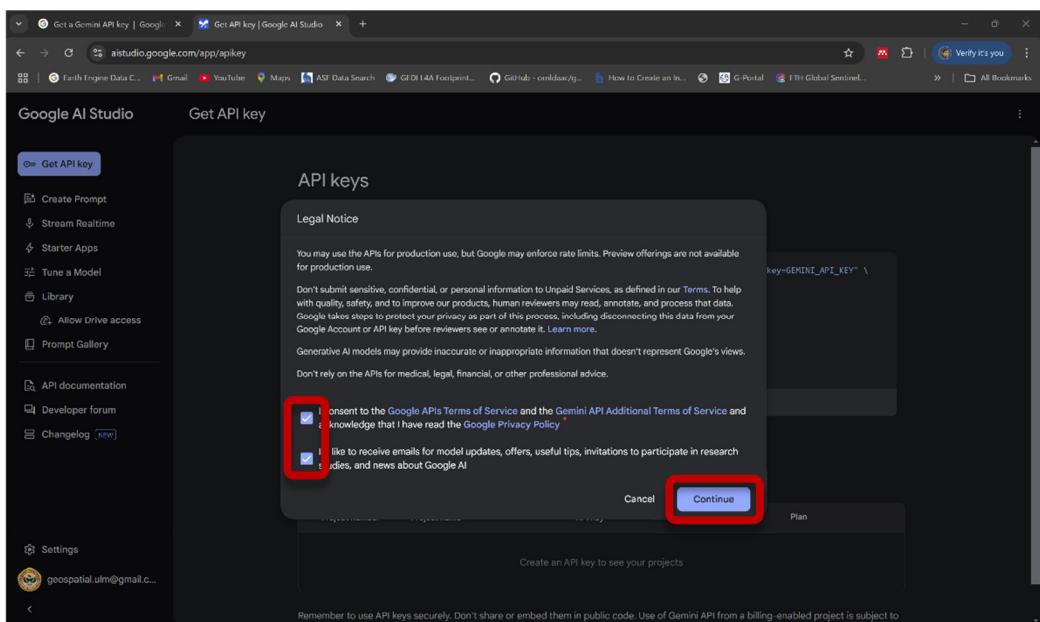
Pada gambar di atas, pilih **Gemini::gemini-1.5-flash** pada kolom **Language model**. Kita diharuskan untuk mengisi **GOOGLE\_API\_KEY** pada kolom **API Keys**. Untuk membuat Gemini API Keys, login ke akun Google Anda, sebaiknya akun yang sama dengan yang Anda gunakan untuk Google Colab dan GEE nantinya. Kemudian buka laman <https://ai.google.dev/gemini-api/docs/api-key>, sebagaimana pad gambar berikut:



Pada gambar di atas, klik tombol **Get a Gemini API key in Google AI Studio**. Selanjutnya, Anda akan di bawah ke laman berikutnya seperti terlihat pada gambar berikut:

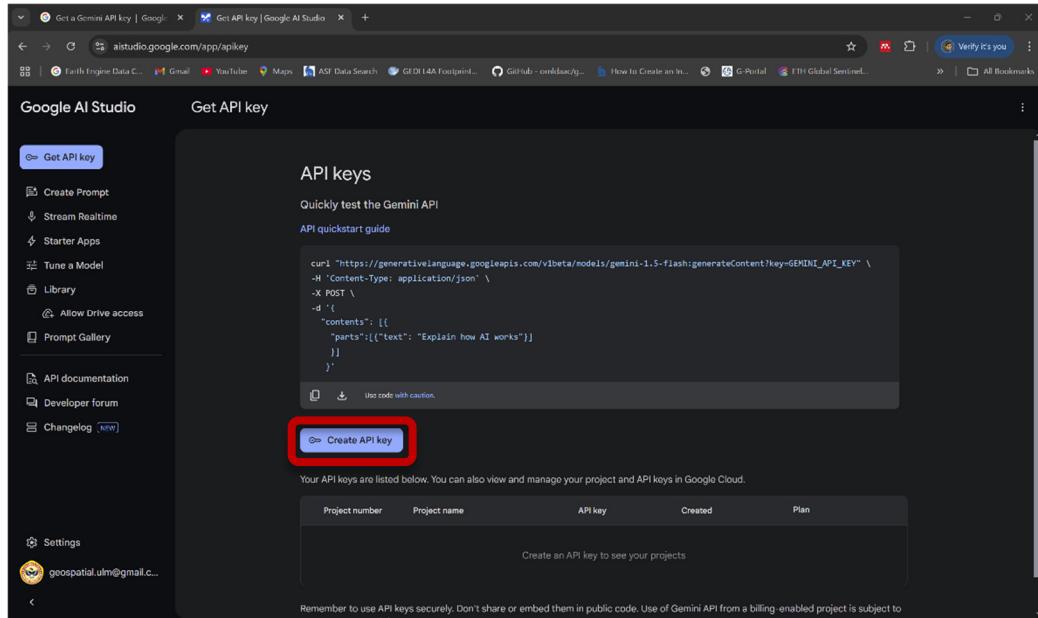


Pada gambar di atas, klik tombol **Get API key**, selanjutnya akan muncul kotak dialog berikut:

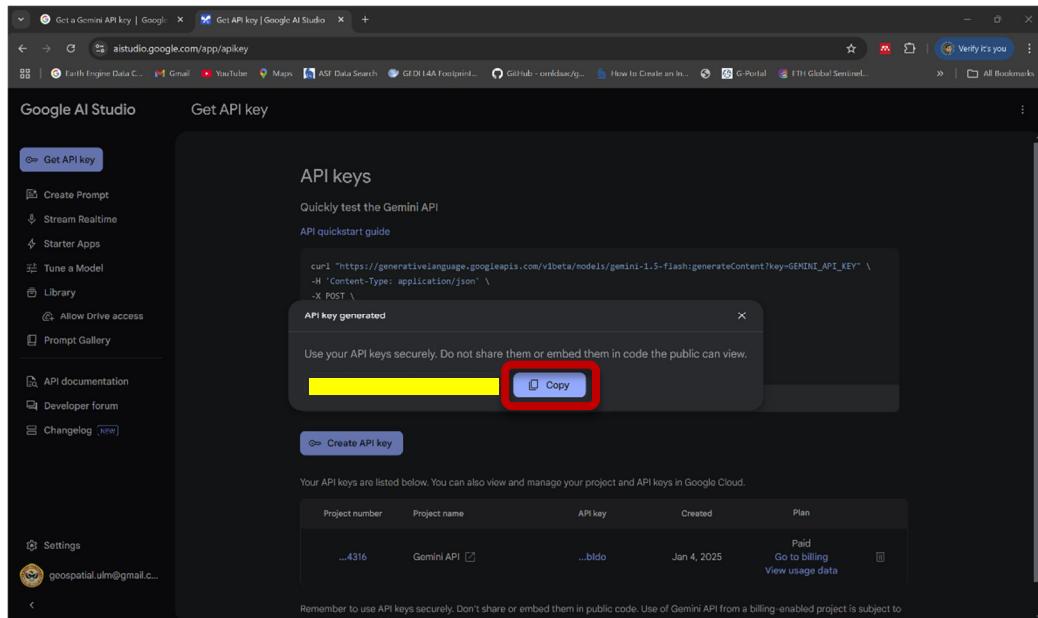


Pada gambar di atas, klik/centang semua opsi, kemudian klik tombol **Continue**. Kemudian kita akan di bawa ke laman seperti pada gambar berikut

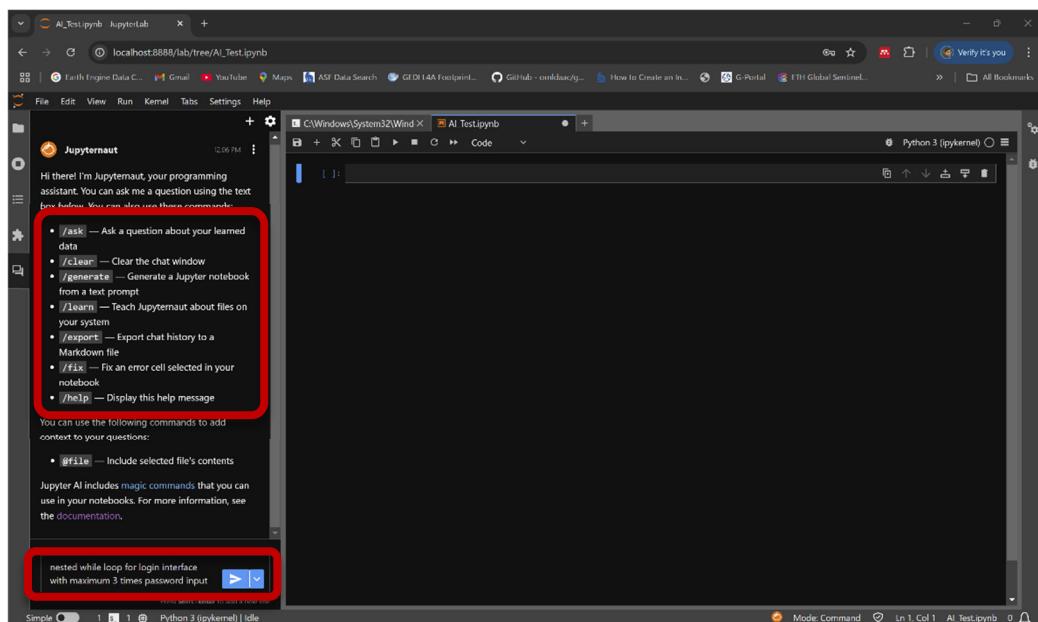
## Bab II Fundamental Python



Pada gambar di atas, klik tombol **Create API key**. Kemudian sebuah **API key** akan muncul seperti pada gambar di bawah:

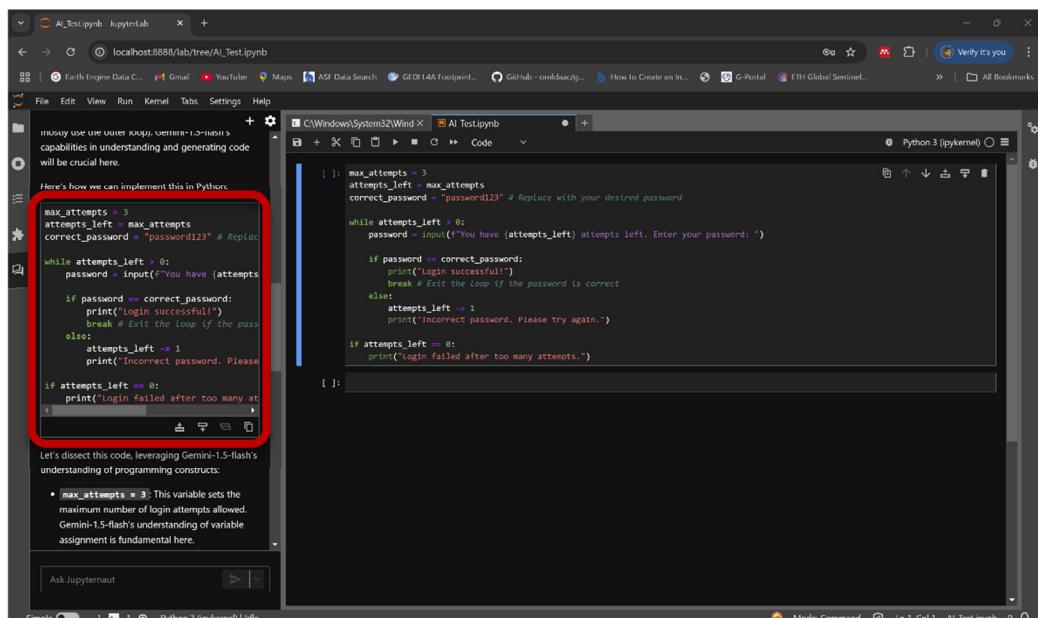


Klik tombol **Copy** pada gambar di atas. Selanjutnya, **paste** Gemini API key yang sudah Anda copy ke kolom **API Keys** pada panel kiri JupyterLab (timpa teks **GOOGLE\_API\_KEY**). Kemudian klik tombol **Save Changes** di bagian bawah. Untuk mulai menggunakan Gemini, klik terlebih dahulu tombol pada panel kiri Jupyter Lab. Selanjutnya, kita dapat mengetikkan prompt pada kolom chat yang disediakan, sebagaimana contoh pada gambar berikut:



Pada gambar di atas, diketikkan sebuah contoh prompt yang berbunyi `nested while loop for login interface with maximum 3 times password input`. Selanjutnya, tekan **Enter** pada keyboard Anda atau klik tombol untuk mengirimkan chat. Gemini AI akan langsung merespon chat kita secara online, dan Gemini AI akan memberikan jawaban berupa kode-kode Python yang kita inginkan.

Catatan, jika koneksi internet Anda sedang terputus ketika menjalankan chat, maka Gemini tidak akan dapat memberikan jawaban. Berikut adalah contoh output kode Python yang dihasilkan dari contoh prompt yang kita input di atas.



## Bab II Fundamental Python

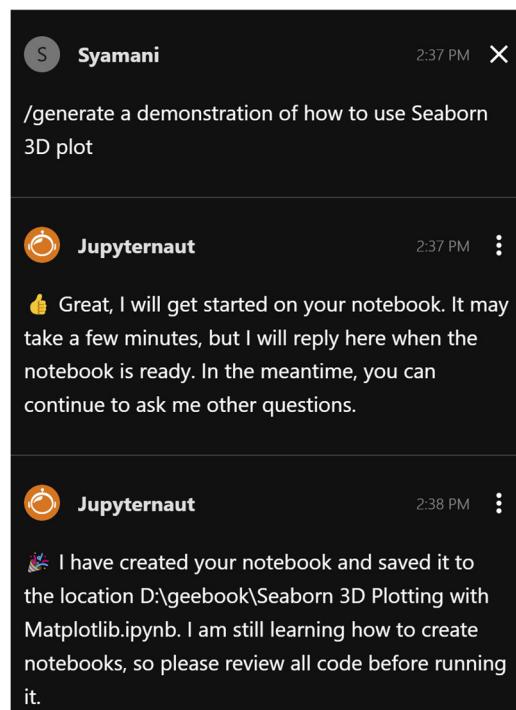
Ada 3 tombol yang terdapat di bagian bawah kode yang diberikan oleh Gemini, yaitu . Tombol paling kiri adalah untuk menambahkan kode dari Gemini ke sebelah atas sel kode yang sedang kita pilih, tombol kedua dari kiri adalah untuk menambahkan kode dari Gemini ke sebelah bawah sel kode yang sedang kita pilih, tombol kedua dari kanan adalah untuk mengganti sel kode yang kita pilih dengan sel kode dari gemini, dan tombol paling kanan adalah untuk mengcopy kode-kode dari Gemini untuk dipaste manual ke dalam sel kode JupyterLab. Gunakan tombol-tombol ini sesuai keperluan Anda.

Perhatikan kembali gambar pada halaman sebelumnya, yaitu pada panel kiri. Perhatikan perintah-perintah dengan garis miring (*slash*), seperti `/ask`, `/clear`, `/generate`, dan sebagainya. Perintah-perintah ini disebut *magic commands*. Silahkan Anda baca sendiri penjelasan tentang fungsi dari masing-masing perintah tersebut.

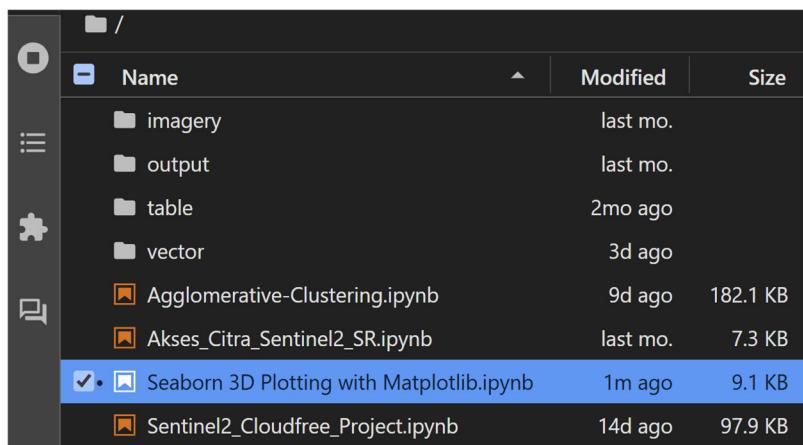
Sebagai contoh, Anda dapat menjalankan perintah berikut pada kolom chat:

```
/generate a demonstration of how to use Seaborn 3D plot
```

Prompt di atas digunakan untuk menghasilkan sebuah notebook baru secara otomatis, dimana nantinya kode-kode Python yang dihasilkan akan sesuai dengan konteks informasi yang kita berikan di dalam perintah. Selanjutnya, pada panel bagian kiri akan muncul pesan-pesan seperti pada gambar di bawah. Proses pembuatan notebook baru secara otomatis dapat berlangsung lama, tergantung pada panjang-pendeknya kode-kode Python yang dihasilkan.

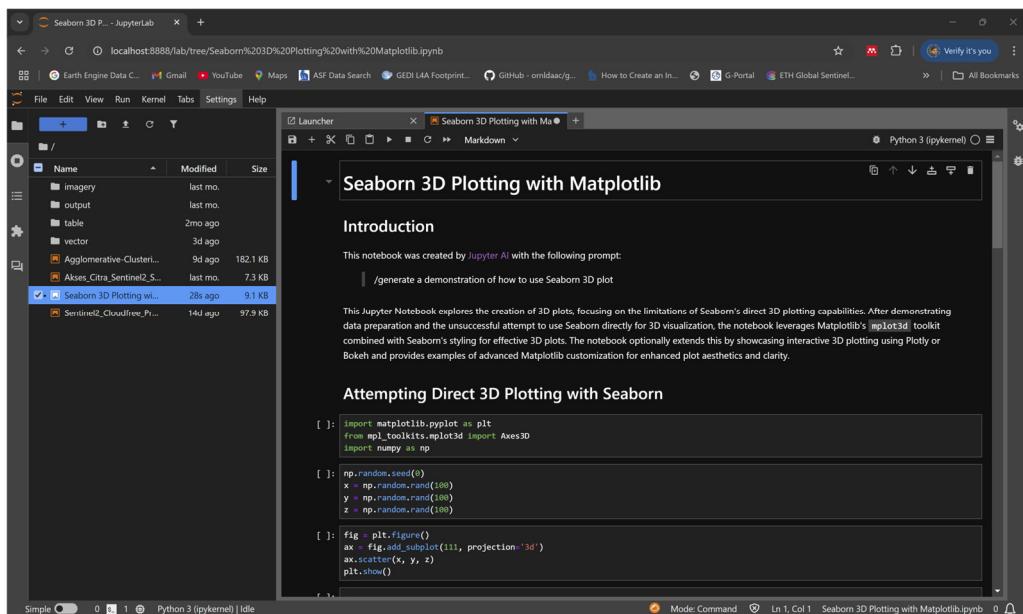


Pada gambar di atas, ada konfirmasi bahwa sebuah notebook baru dengan nama **Seaborn 3D Plotting with Matplotlib.ipynb** sudah berhasil dibuat di folder **D:/geeebook/**. Sebagaimana terlihat pada gambar berikut:



Secara default, lokasi folder tempat terbentuknya file baru akan ditentukan dari lokasi atau *path* ketika Anda menjalankan perintah untuk membuka JupyterLab. Terkecuali jika di dalam JupyterLab Anda masuk atau pindah ke dalam folder lain. Anda dapat langsung membuka file notebook baru yang dihasilkan secara otomatis tersebut.

Contoh kode-kode yang dihasilkan terlihat seperti pada gambar di bawah. Catatan, meskipun nantinya Anda memberikan perintah yang sama pada chat, ada kemungkinan kode-kode yang diberikan bisa berbeda. Hal ini karena algoritma Gemini AI akan terus belajar secara mandiri, dan akan terus berusaha untuk memberikan jawaban-jawaban terbaik ketika kita mengirim chat.

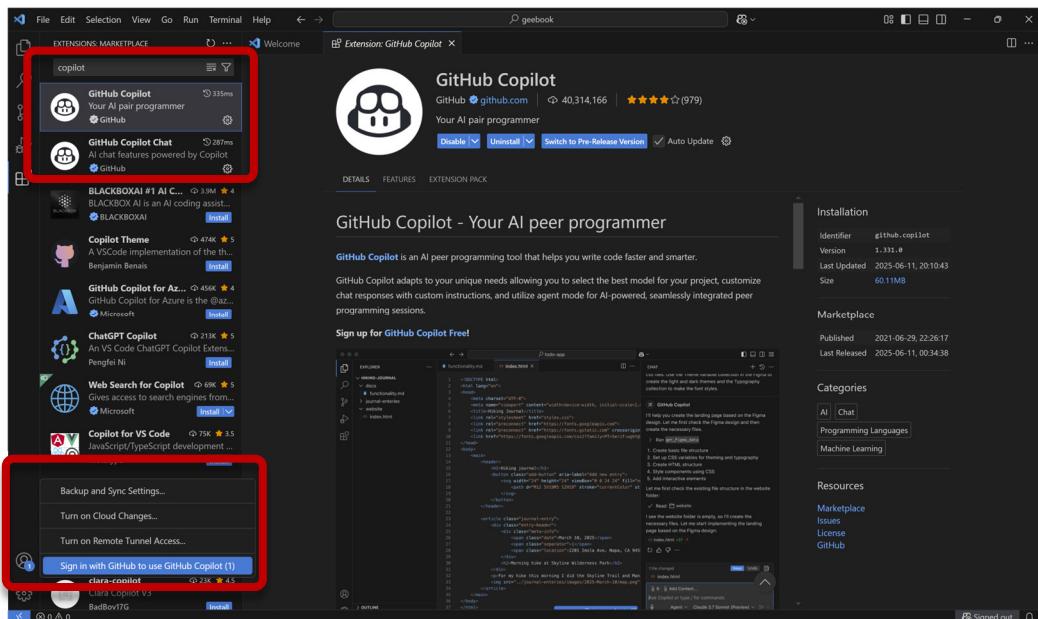


Jika diinginkan, Anda dapat melakukan perubahan pada kode-kode di atas sesuai keperluan.

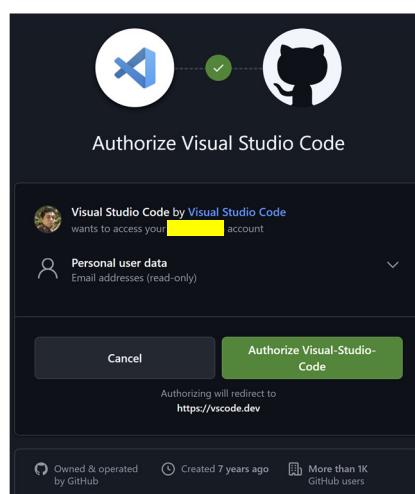
Untuk selanjutnya, silahkan Anda mengeksplorasi sendiri model-model AI yang lain, seperti OpenAI/ChatGPT dan yang lainnya. Teknik-tekniknya tentu saja akan berbeda dengan Gemini yang sudah dijelaskan di atas. Dan sebagian model-model AI, khususnya untuk model-model profesional, kemungkinan layanannya harus berbayar.

### GitHub Copilot

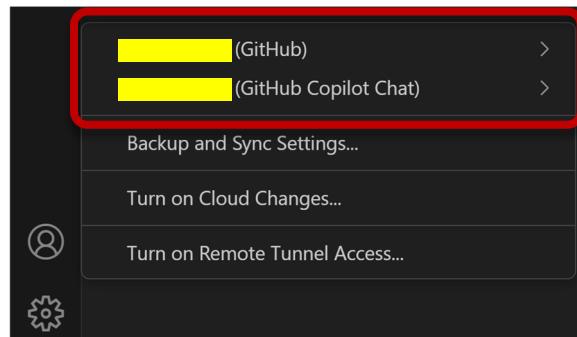
Jika Anda pengguna VS Code, maka GitHub Copilot adalah asisten AI Anda untuk coding. GitHub Copilot bersifat gratis untuk developer individual. Syarat untuk menggunakan GitHub Copilot adalah Anda harus memiliki akun GitHub. Silahkan Anda registrasi terlebih dahulu di laman <https://github.com/> untuk mendapatkan akun gratis GitHub. Setelah Anda memiliki akun GitHub, Anda harus menginstal minimal dua ekstensi, yaitu **GitHub Copilot** dan **GitHub Copilot Chat**. Sebagaimana terlihat pada gambar berikut:



Setelah kedua ekstensi di atas terinstal, klik tombol **Accounts** (👤) sebagaimana ditunjukkan pada gambar di atas, kemudian klik **Sign in with GitHub to use GitHub Copilot**. Anda akan dibawa ke laman login GitHub, silahkan login menggunakan username dan password Anda, kemudian ikuti instruksi di laman. Jika muncul laman seperti pada gambar di bawah, langsung klik tombol **Authorize Visual-Studio-Code**, dan tunggu proses otorisasi hingga selesai.

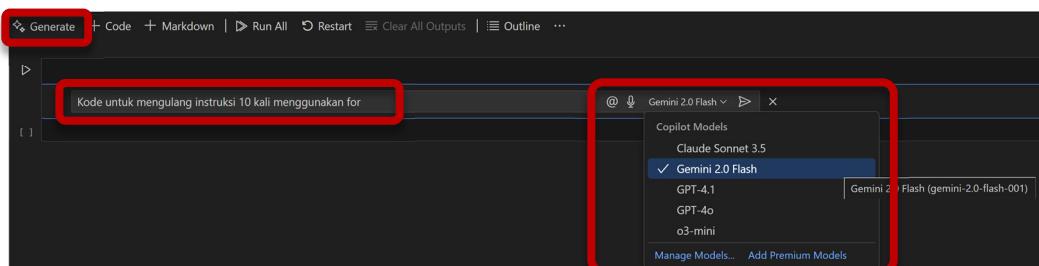


Setelah proses otorisasi selesai, kita biasanya akan dibawa ke (atau disuruh membuka) VS Code. Jika Anda mengklik tombol **Accounts** dan ada tambahan menu seperti pada gambar di bawah, berarti VS Code Anda sudah siap menggunakan GitHub Copilot.

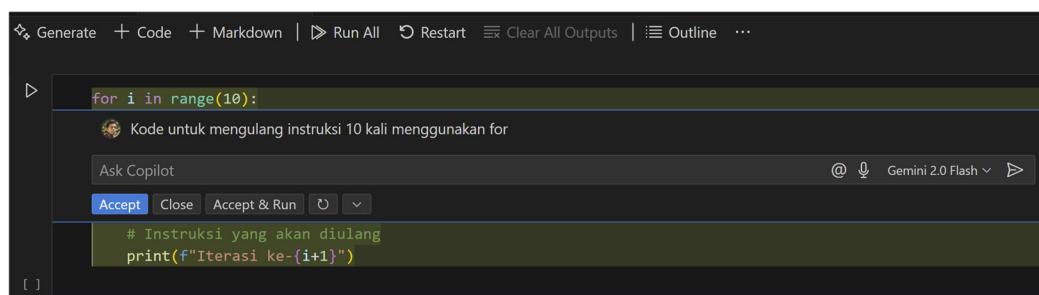


*Jika proses otorisasi GitHub Copilot sudah selesai sepenuhnya, Anda dapat logout dari laman GitHub di browser Anda. Hal ini tidak akan membuat GitHub Copilot logout dari VS Code.*

Untuk bertanya kepada Copilot, pastikan kursor berada di dalam sebuah sel kode, klik tombol **Generate** atau tekan tombol **Ctrl + I** di keyboard secara bersamaan. Kemudian pada kolom **Ask Copilot**, ketikkan prompt Anda, baik dalam Bahasa Inggris (direkomendasikan) maupun dalam Bahasa Indonesia. Sebagaimana dicontohkan pada gambar di bawah:



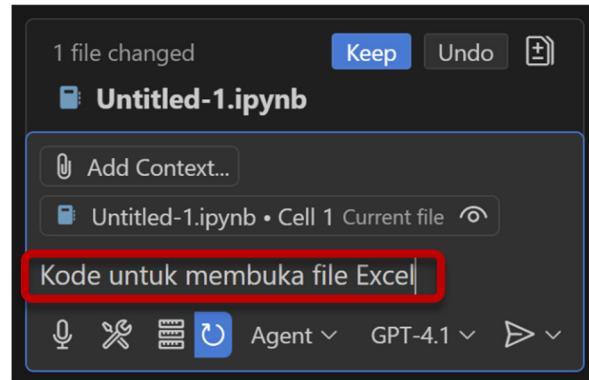
Anda dapat memilih berbagai asisten AI, baik Gemini, GPT, atau yang lainnya. Jika sudah selesai menginput prompt Anda, klik tombol **Send and Dispatch** (▶) atau tekan **Enter**. Kode-kode Python akan muncul sebagaimana pada gambar di bawah:



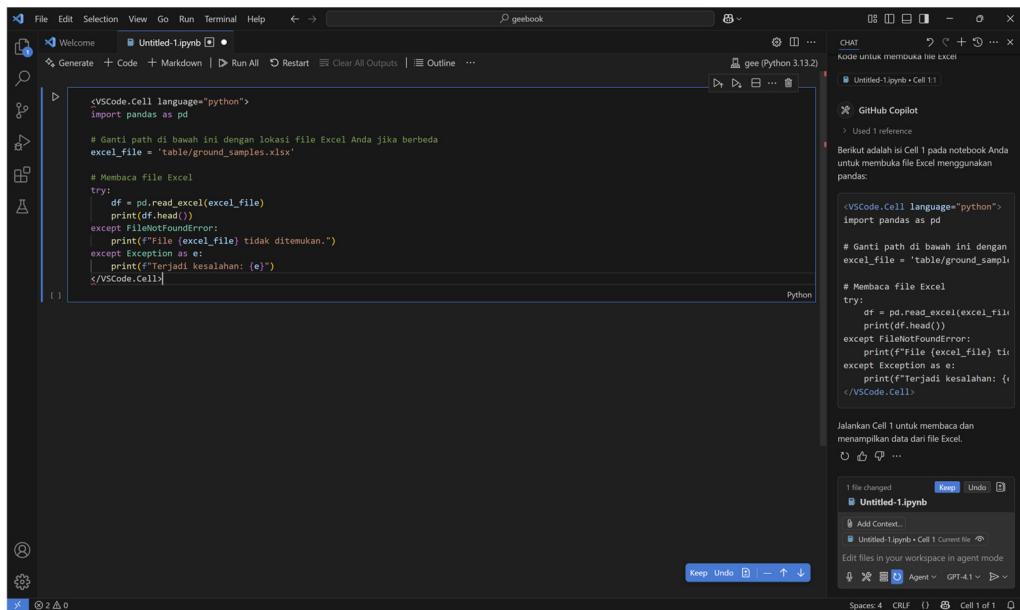
## Bab II Fundamental Python

Jika Anda setuju dengan kode-kode yang sudah diberikan, Anda dapat langsung mengklik tombol **Accept**. Jika kode-kode yang muncul tidak sesuai keinginan Anda, langsung rubah prompt Anda, dan klik lagi tombol **Send and Dispatch**.

Untuk menggunakan GitHub Copilot Chat, tempatkan kursor pada sebuah sel kosong, kemudian klik menu **View – Chat** atau tekan tombol **Ctrl + Alt + I**. Kemudian ketikkan prompt Anda sebagaimana dicontohkan pada gambar berikut:



Selanjutnya, klik tombol **Send** () atau tekan **Enter**. Kode-kode yang Anda minta akan langsung muncul sebagaimana dicontohkan pada gambar berikut:



Anda dapat langsung memasukkan kode-kode yang diberikan ke dalam sel, kemudian lakukan beberapa perubahan kode sesuai keinginan Anda.

Sebagai tambahan, login ke akun GitHub Copilot dari VS Code akan menyebabkan VS Code terkoneksi langsung dengan akun GitHub Anda. Artinya, Anda dapat langsung mempublikasikan kode-kode ke akun GitHub Anda. Terkait publikasi kode-kode Python ke GitHub tidak akan dibahas lebih jauh di dalam buku ini, silahkan Anda mencari literatur lain untuk mempelajarinya.



# Bab III

# Google Earth

# Engine



## A. Mengenal Google Earth Engine

Google Earth Engine (GEE) (Gorelick et al., 2017) adalah platform komputasi dan analisis berbasis *cloud* yang dikembangkan oleh Google untuk memfasilitasi pemrosesan dan analisis citra satelit serta data geospasial lainnya secara efisien dan skalabel. Platform ini menyediakan akses ke kumpulan data citra satelit global yang luas, termasuk data dari Landsat, Sentinel, MODIS, dan banyak lagi. Dengan GEE, pengguna dapat melakukan berbagai jenis analisis spasial dan temporal, seperti pemantauan perubahan lahan, pemetaan tutupan lahan, analisis cuaca dan iklim, pemantauan kebakaran hutan, dan banyak lagi.

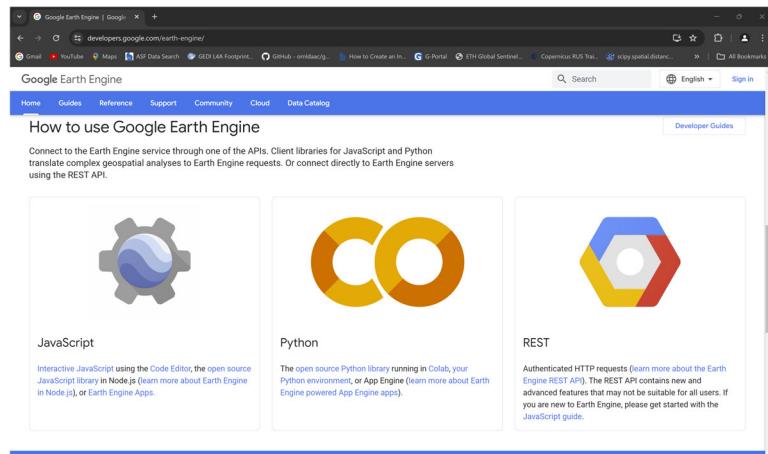
Platform ini juga dilengkapi dengan berbagai alat analisis, visualisasi, dan pemrograman yang memungkinkan pengguna untuk mengembangkan dan menjalankan skrip-skrip untuk analisis data geospasial. GEE merupakan salah satu alat yang sangat berguna bagi para ilmuwan, peneliti, mahasiswa, dan praktisi di bidang pemetaan, penginderaan jauh, dan ilmu kebumian untuk melakukan analisis data geospasial dengan skala global dan waktu yang panjang. GEE memungkinkan analisis data observasi bumi yang pada umumnya memiliki kapasitas besar (*big earth data*) secara lebih efisien, dengan cara memfokuskan pengguna pada *point of interest* dan mengunduh hanya data yang diperlukan.

GEE memiliki beberapa kelebihan yang membuatnya menjadi platform yang kuat untuk akses dan analisis data observasi bumi. Berikut adalah pernyataan-pernyataan beberapa ahli tentang GEE:

- GEE, platform pemrosesan *big data* penginderaan jauh berbasis *cloud*, mengintegrasikan citra penginderaan jauh skala besar yang telah digunakan secara luas selama lebih dari 40 tahun. Platform ini juga mencakup data topografi, iklim dan cuaca, tutupan lahan, demografi, dan data lingkungan lainnya (Xiaona et al., 2022).
- GEE mendobrak pemisahan data tradisional, algoritma model, dan daya komputasi, mewujudkan penerapan *cloud* pada ketiganya, dan menunjukkan potensi besar dalam pemrosesan dan analisis cepat data besar melalui penggabungan data besar dan komputasi *cloud* bersama-sama (Xiaona et al., 2022).
- Dalam perspektif aplikasi, GEE tidak hanya menghadirkan peluang pengembangan untuk analisis jangka panjang berskala global yang cepat, tetapi juga mempromosikan pembagian data, algoritma, dan produk secara cepat, yang selanjutnya menandai dimulainya era keterbukaan dan pembagian penginderaan jarak jauh (Xiaona et al., 2022).
- GEE berfungsi sebagai platform serbaguna untuk memproses dan memvisualisasikan kumpulan data geospasial, dengan tujuan utamanya adalah menyediakan platform terbuka untuk analisis geospasial skala planet (Vijayakumar et al., 2024).
- Platform GEE telah menunjukkan kemampuannya untuk melakukan analisis spasial dan temporal pada data skala global dengan kecepatan komputasi yang jauh lebih cepat, menjadikannya alat yang menarik bagi komunitas ilmiah, menawarkan fleksibilitas dan aksesibilitas (Vijayakumar et al., 2024).
- GEE adalah platform berbasis komputasi awan yang memfasilitasi geoprosesing, menjadikannya alat yang sangat diminati oleh dunia akademis dan penelitian. GEE telah terbukti menjadi platform web yang sedang berkembang dengan potensi untuk mengelola data satelit berukuran besar dengan mudah. GEE dianggap sebagai alat multidisiplin dengan beragam aplikasi di berbagai bidang ilmu pengetahuan (Velastegui-Montoya et al., 2023).

### Bab III Google Earth Engine

Secara bawaan, GEE dapat diimplementasikan dengan Bahasa JavaScript via Code Editor <https://code.earthengine.google.com>. Akan tetapi, GEE juga dapat diimplementasikan menggunakan Bahasa Python atau *REpresentational State Transfer* (REST). Sebagaimana terlihat pada Gambar 3.1. Pengguna Python tentu saja mengimplementasikan GEE melalui Google Colab <https://colab.research.google.com>, atau dapat juga menggunakan JupyterLab. Baik JavaScript, Python, maupun REST, masing-masing memiliki kelebihan dan kekurangan. Untuk mengakses dan mempelajari referensi *Application Program Interface* (API) dari GEE selengkapnya silahkan kunjungi laman <https://developers.google.com/earth-engine/apidocs>.



Gambar 3.1. Opsi bahasa dalam penggunaan GEE

Dalam konteks GEE, JavaScript lebih interaktif. Di samping komunitas, literatur, dan tutorial yang sangat luas. Sementara Python merupakan bahasa pemrograman utama di dalam *scientific computing*. Kelebihan Python adalah terintegrasi dengan berbagai framework *data science*, *machine learning*, dan *deep learning* yang sangat *powerfull*. Seperti Scikit-Learn, PyTorch, TensorFlow, Keras, dan sebagainya. Sehingga developer yang ingin mengeksplorasi GEE untuk kepentingan *artificial intelligence* tentu saja akan lebih efektif jika menggunakan Python. Terlebih lagi Python memiliki beragam librari untuk visualisasi data saintifik, seperti Matplotlib, Seaborn, Bokeh, dan sebagainya.

GEE memiliki sejumlah tipe data bawaan, yaitu sebagai berikut:

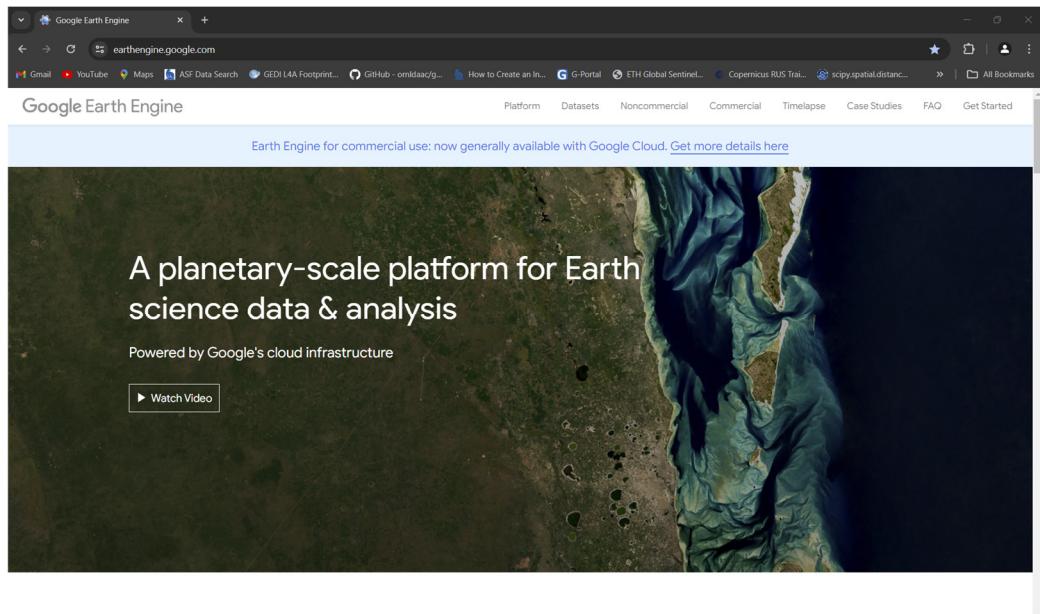
Tabel 3.1. Tipe-tipe data GEE

Data Type	Description
Image	The fundamental raster data type in Earth Engine.
ImageCollection	A stack or time-series of images.
Geometry	The fundamental vector data type in Earth Engine.
Feature	A Geometry with attributes.
FeatureCollection	A set of features.
Reducer	An object used to compute statistics or perform aggregations.
Join	How to combine datasets (Image or Feature collections) based on time, location, or an attribute property.
Array	For multi-dimensional analyses.

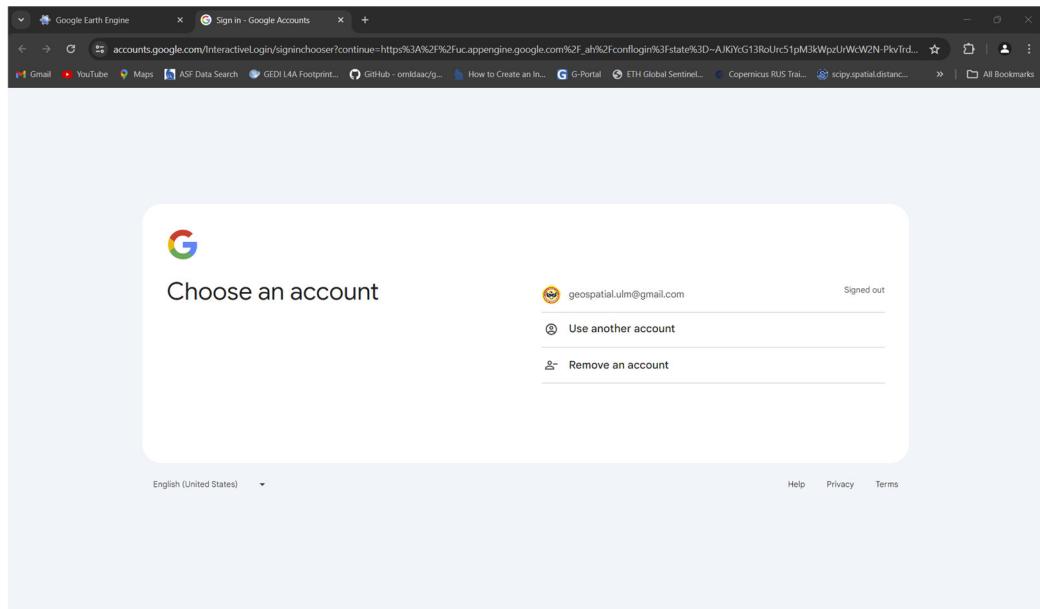
Sumber: <https://developers.google.com/earth-engine/guides>

## Registrasi ke GEE

Untuk dapat menggunakan GEE, kita terlebih dahulu harus registrasi dan login ke laman <https://earthengine.google.com>. Sebagaimana terlihat pada gambar di bawah:

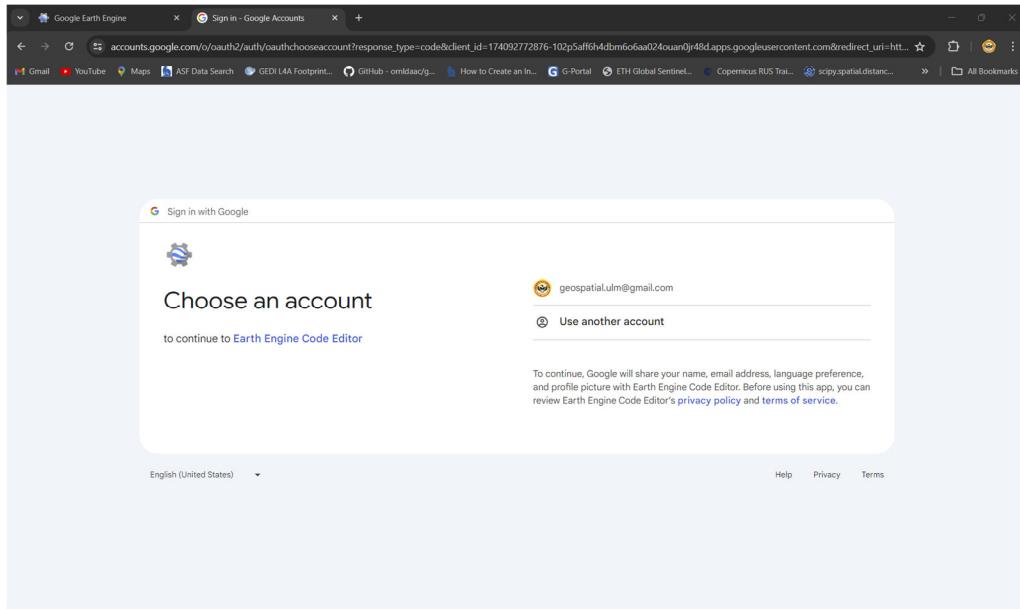


Jika nantinya kita mengakses GEE via Google Colab, maka akun email yang digunakan harus sama dengan yang kita gunakan di Google Colab. Masukkan atau ketikkan alamat email Anda, sebagaimana pada gambar berikut:

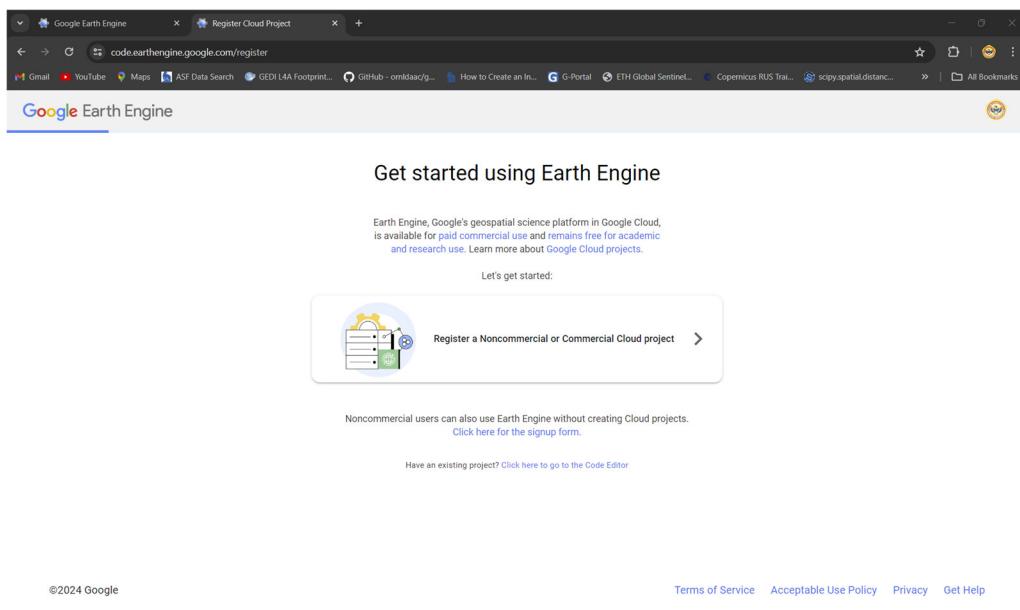


### Bab III Google Earth Engine

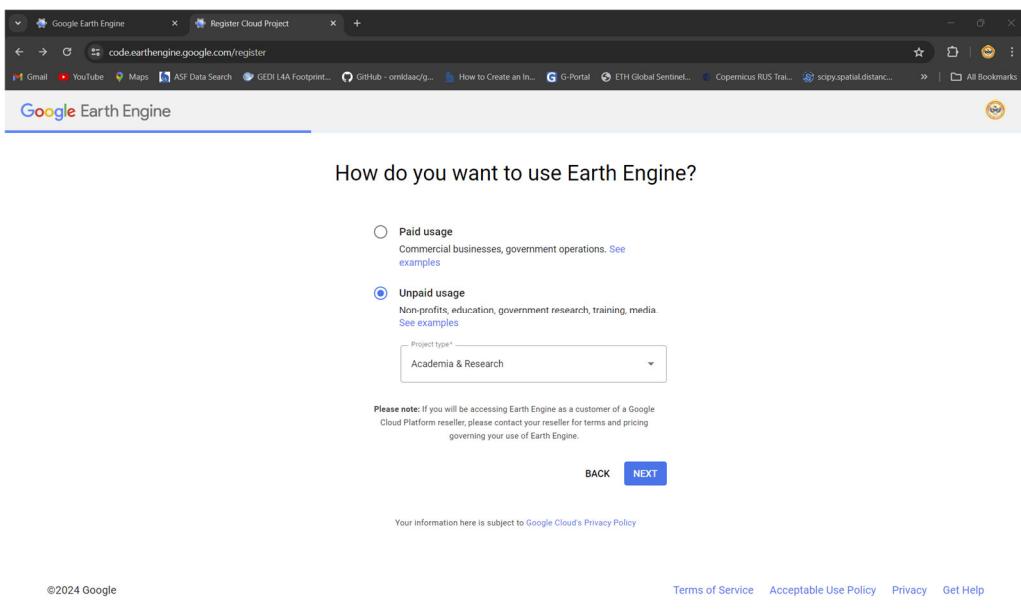
Pada laman **Choose an account**, pastikan kembali bahwa akun Gmail yang kita gunakan adalah akun yang sama dengan yang kita gunakan di Google Colab. Kemudian klik akun kita, sebagaimana pada gambar di bawah:



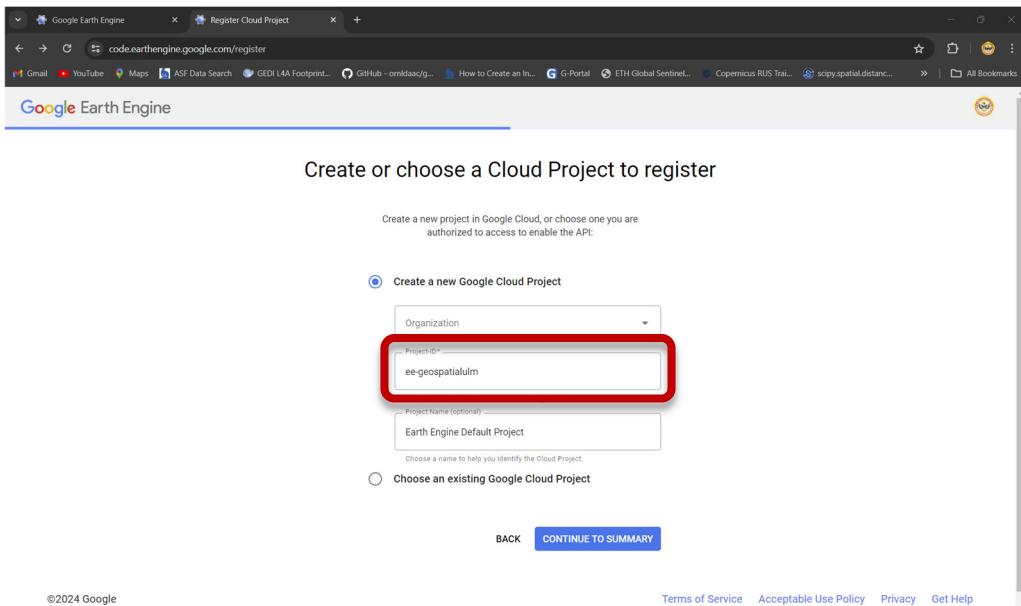
Pada laman **Get started using Earth Engine** seperti pada gambar di bawah, klik **Register a Noncommercial or Commercial Cloud project**.



Pada laman **How do you want to use Earth Engine?**, pilih **Unpaid usage**, kemudian pada kolom **Project type** pilih dan klik sesuai peruntukan penggunaan Anda, misalnya **Academia & Research**, jika nantinya Earth Engine akan digunakan untuk kepentingan pendidikan dan penelitian. Sebagaimana pada gambar berikut:



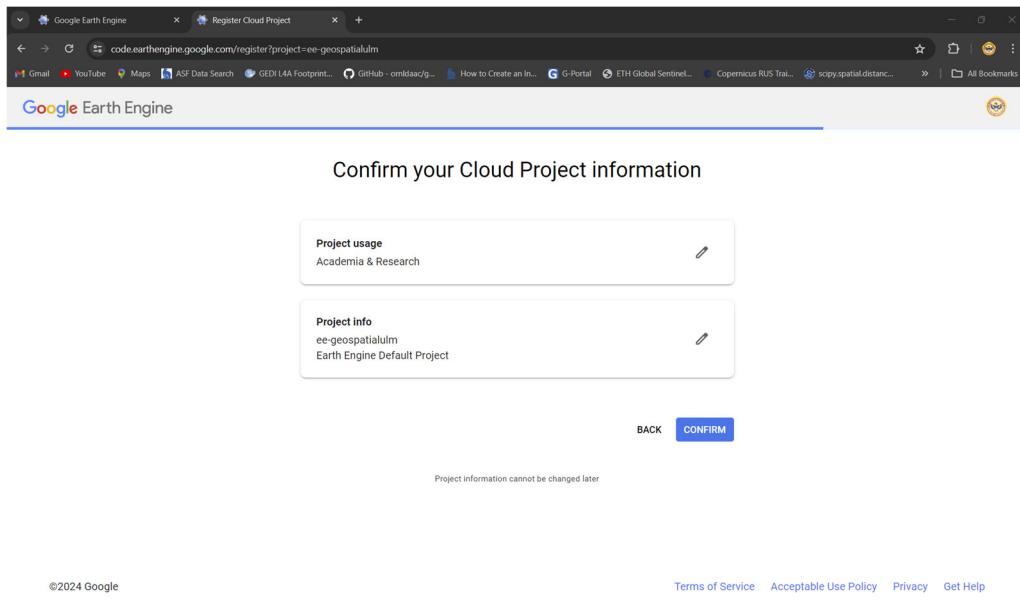
Pada laman **Create or choose a Cloud Project to register**, klik opsi **Create a new Google Cloud Project**, sebagaimana pada gambar di bawah. Tentukan sendiri nama *Project ID* Anda, misalnya saya memilih nama **ee-geospatialulm**. Kemudian klik tombol **CONTINUE TO SUMMARY**.



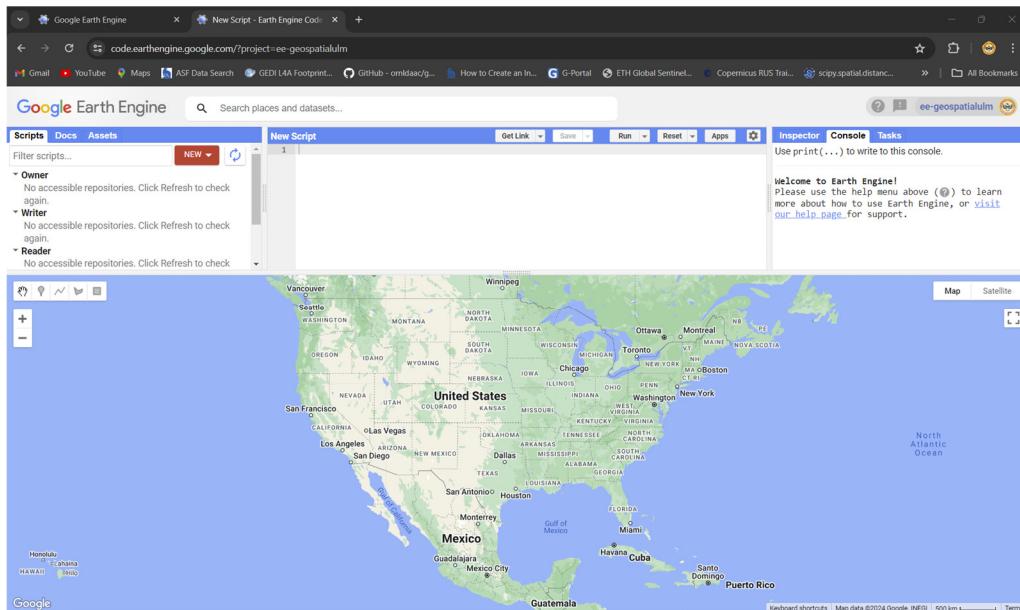
Perhatikan dan ingat bahwa *Project ID* seperti **ee-geospatialulm** di atas adalah nama yang akan digunakan setiap kali inisialisasi project GEE di dalam Google Colab nantinya. Tentu saja, *Project ID* Anda akan berbeda namanya, sesuai dengan nama yang Anda tentukan sendiri.

### Bab III Google Earth Engine

Selanjutnya, kita akan dibawa ke laman **Confirm your Cloud Project information**, sebagaimana pada gambar berikut:



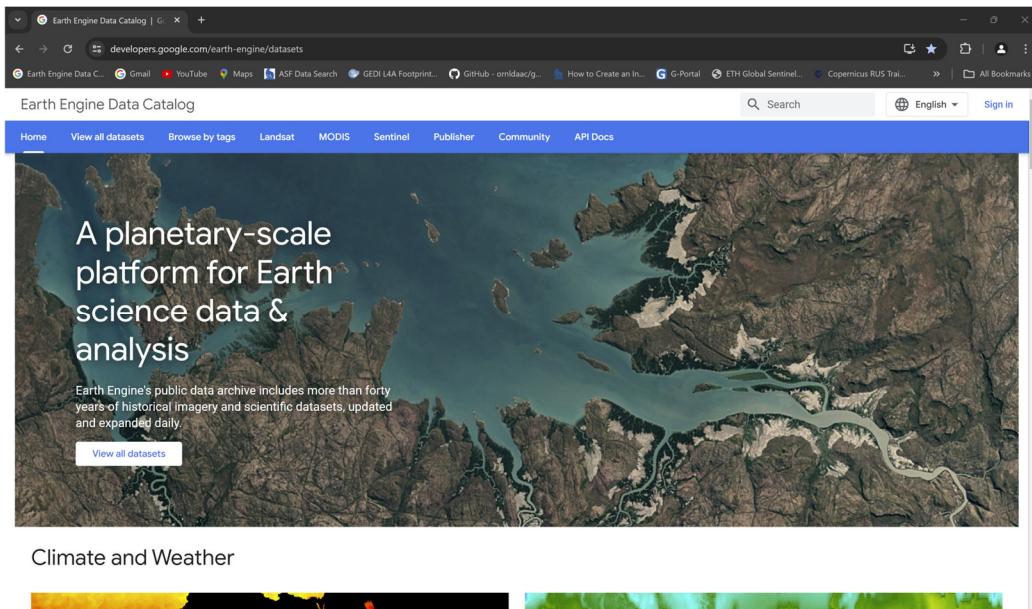
Jika pengaturan cloud project kita dirasa sudah sesuai dengan yang diinginkan, langsung klik tombol **CONFIRM**. Jika tidak sesuai, klik tombol **BACK**, dan atur kembali cloud project kita.



Jika tombol **CONFIRM** sudah diklik, kita akan dibawa ke laman GEE sebagaimana terlihat pada gambar di atas. Laman di atas merupakan Code Editor GEE dengan antar muka Bahasa JavaScript. Tentu saja, jika Anda menggunakan GEE dengan Bahasa JavaScript, maka Anda dapat langsung bekerja di laman tersebut. Tetapi, jika kita menggunakan GEE dengan Bahasa Python via Google Colab, silahkan tutup laman di atas secara manual seperti kita menutup laman web biasa.

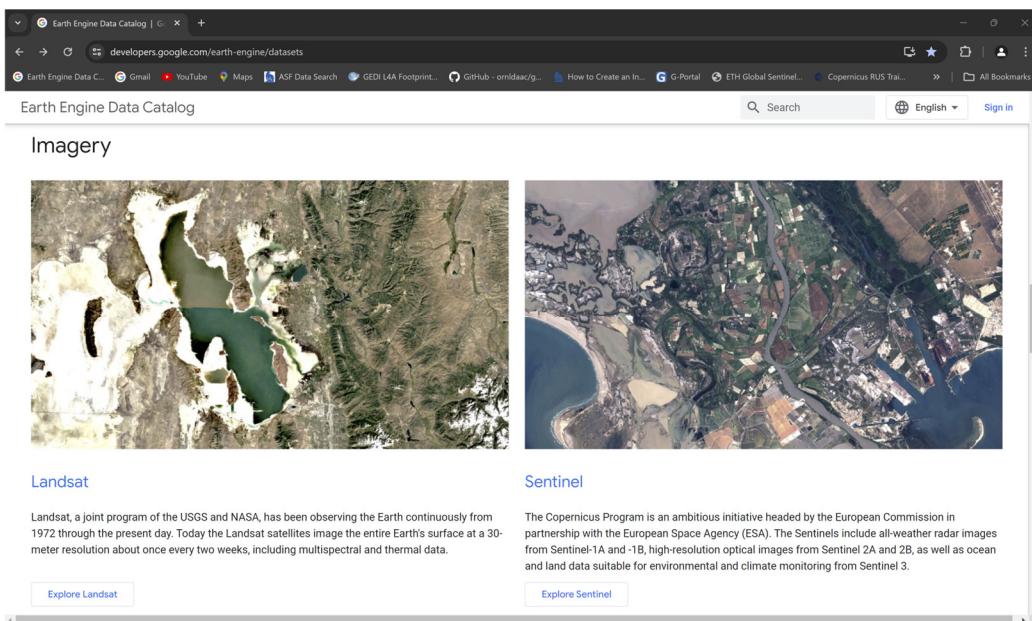
## Katalog Data Earth Engine

Sebelum melakukan pemrosesan citra satelit digital via Google Colab, terlebih dahulu jelajahi Katalog Data Earth Engine di laman <https://developers.google.com/earth-engine/datasets>. Sebagaimana terlihat pada gambar di bawah:



Climate and Weather

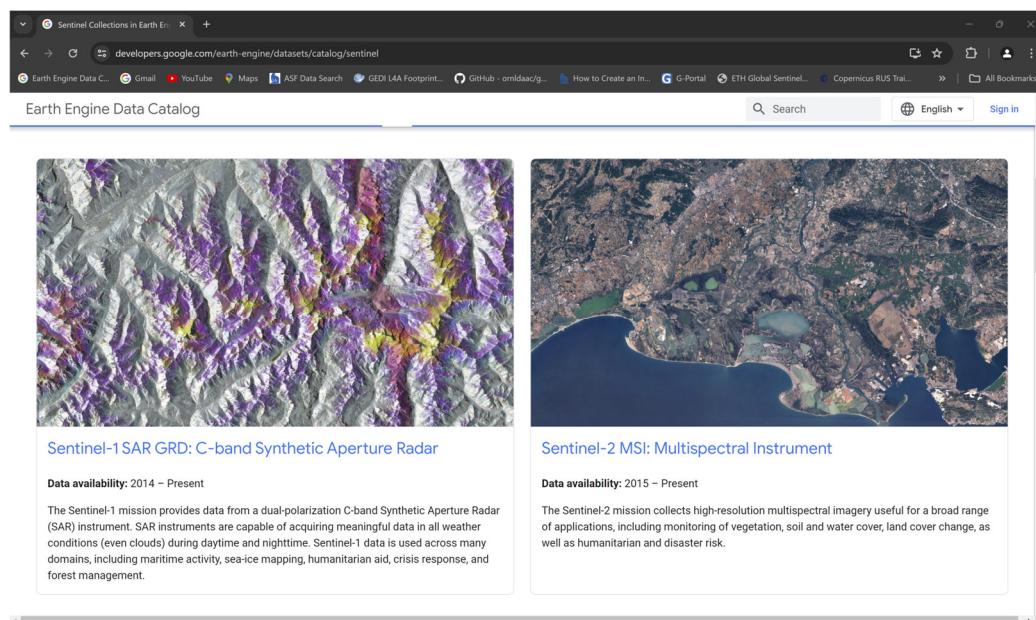
Di laman katalog data Earth Engine ini kita dapat menemukan data geospasial yang diperlukan untuk kita proses nantinya. Misalnya kita pilih Citra Sentinel, sebagaimana gambar di bawah:



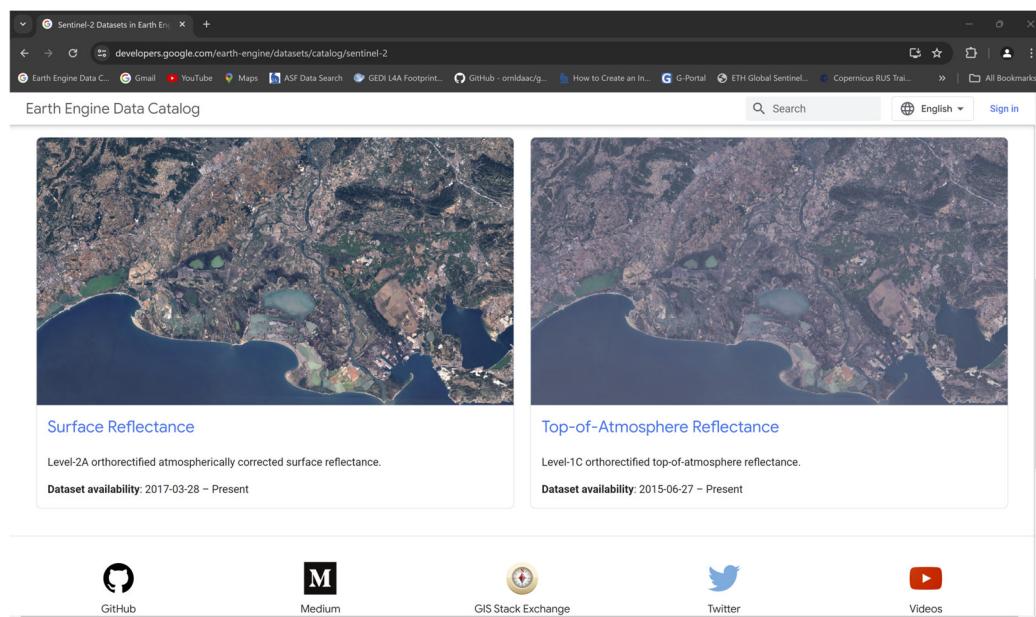
Klik **Sentinel** pada laman di atas. Kemudian kita akan dibawa pada laman seperti gambar berikut:

### Bab III Google Earth Engine

Pada laman di bawah, klik **Sentinel-2 MSI: Multispectral Instrument**.



Pada laman di bawah, klik **Top-of-Atmosphere Reflectance**. Lihat kembali penjelasan pada Bab I halaman 7, tentang perbedaan Surface Reflectance dan Top-of-Atmosphere Reflectance.



Perhatikan dengan cermat laman yang muncul berikutnya, sebagaimana terlihat seperti pada gambar berikut:

Pada laman di atas, entri `ee.ImageCollection('COPERNICUS/S2_HARMONIZED')` merupakan Earth Engine Snippet yang digunakan untuk mengakses Sentinel-2 image collection. Copy kode `ee.ImageCollection('COPERNICUS/S2_HARMONIZED')`, nantinya kode ini akan kita paste di dalam notebook Google Colab untuk mengakses Citra Sentinel-2.

## Mengakses Citra Sentinel-2 MSI

Buka laman <https://colab.research.google.com/>, kemudian login dan buat sebuah notebook baru. Ganti namanya menjadi **Sentinel-2-Project.ipynb** (atau nama apa pun yang Anda suka). Pada sel teks paling atas, sisipkan teks berikut.

```
# **Sentinel-2 MSI Project**
```

Sehingga dihasilkan kenampakan seperti pada gambar di bawah. Kemudian pada sel kode di bawahnya, ketikkan kode berikut kemudian klik **Run**:

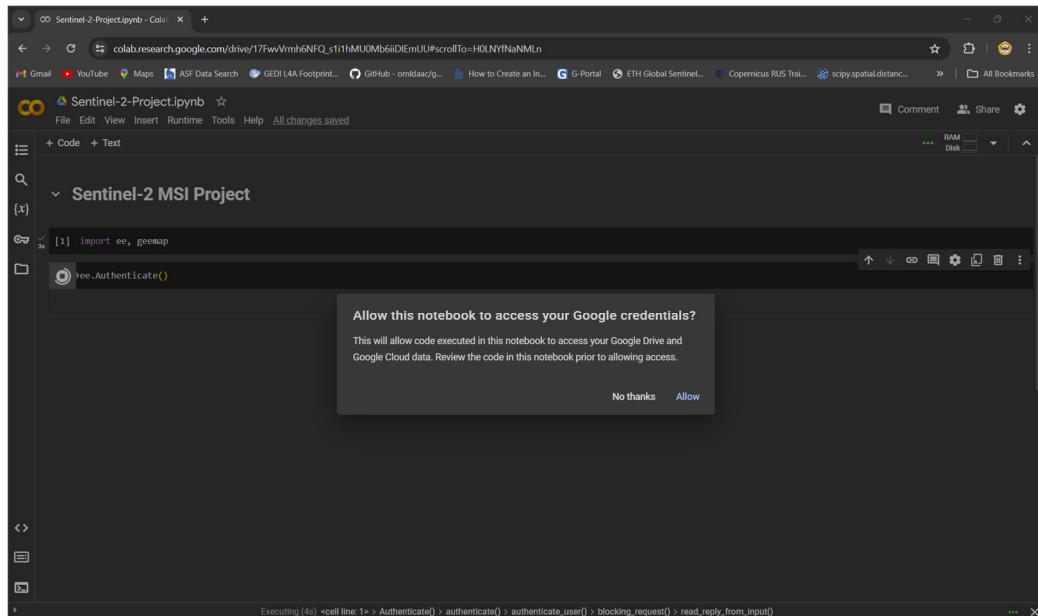
```
import ee, geemap
```

Kode seperti di atas berfungsi untuk mengimport paket Earth Engine dan GEE Map. Klik tombol **+ Code** untuk menambahkan sel kode baru di bawah, kemudian ketikkan kode berikut dan **Run**:

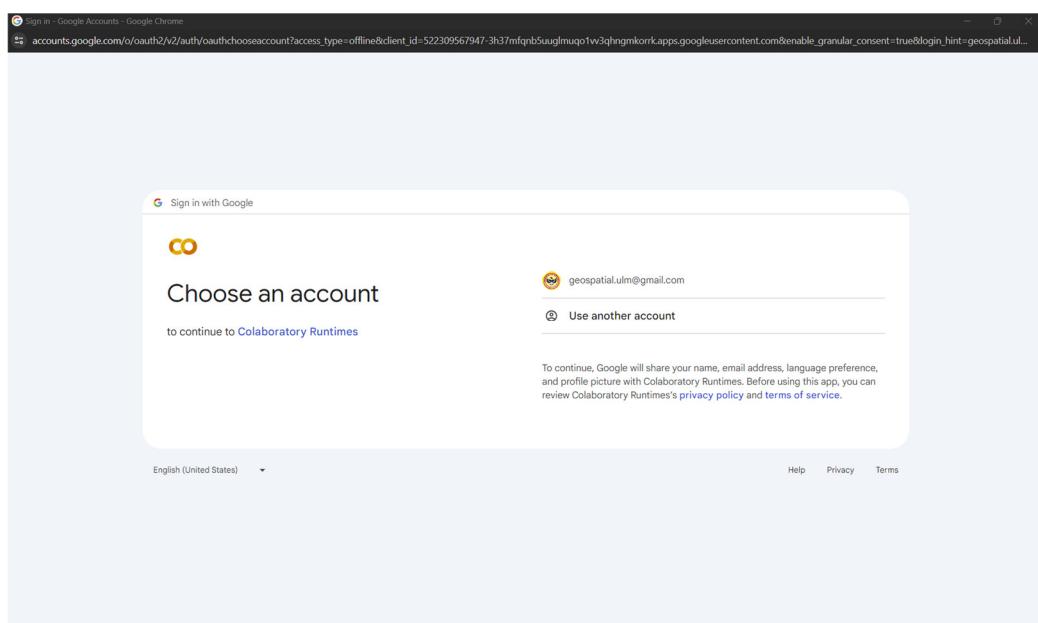
```
ee.Authenticate()
```

Kode di atas adalah untuk autentikasi atau pemberian izin akses Earth Engine ke dalam notebook Google Colab. Jika kemudian muncul kotak dialog **Allow this notebook to access your Google credentials?** sebagaimana pada pada gambar di bawah, klik tombol **Allow**.

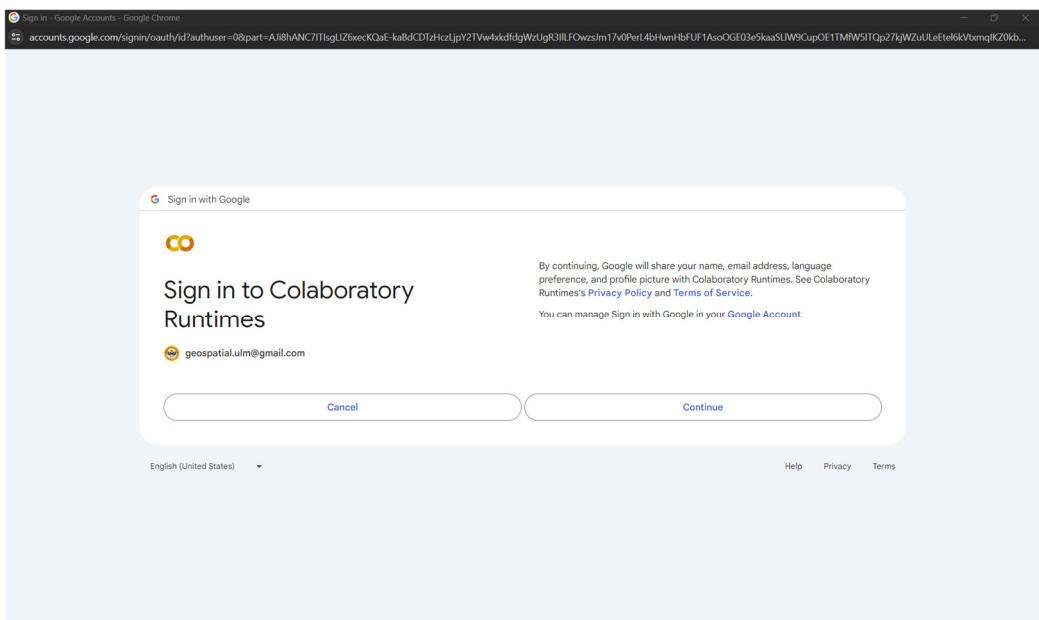
### Bab III Google Earth Engine



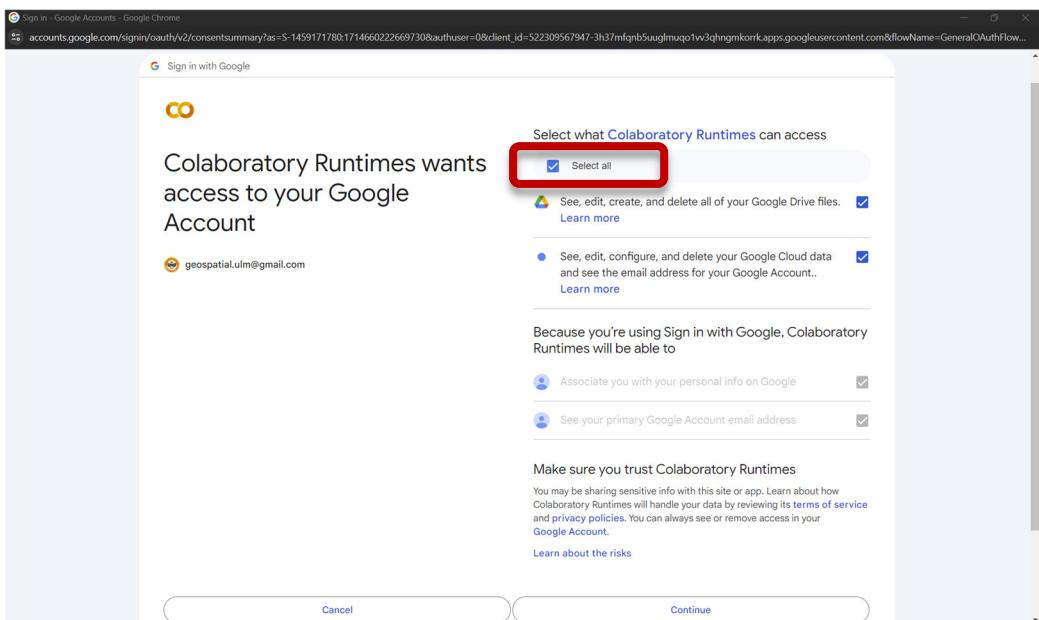
Selanjutnya, kita akan di bawa ke laman seperti pada gambar berikut:



Pada laman **Choose an account** di atas, klik atau masukkan akun Gmail GEE kita. Kemudian akan ditampilkan laman seperti gambar berikut:



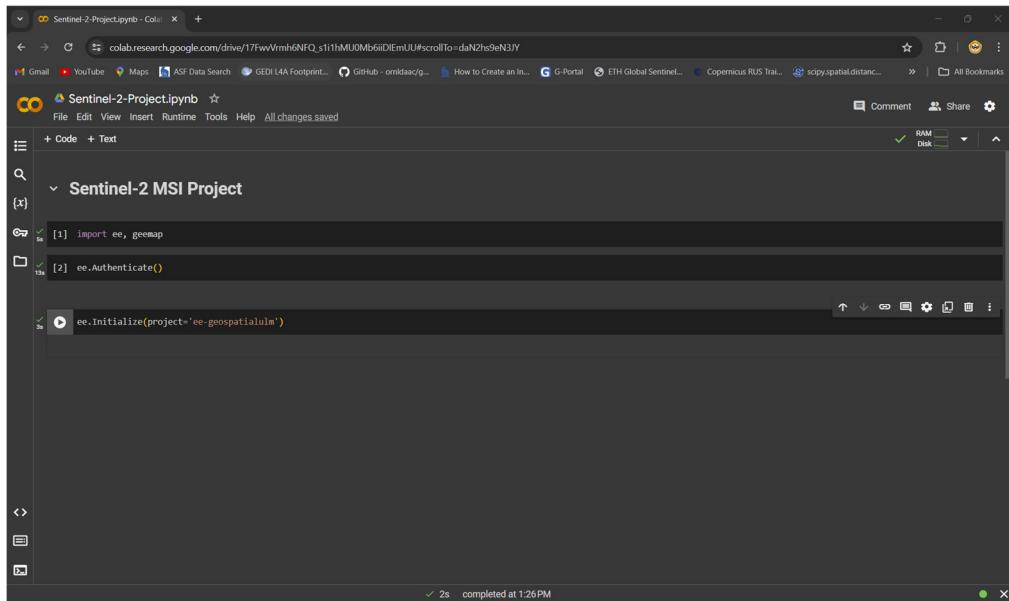
Pada laman di atas, klik tombol **Continue**. Dan kita akan di bawa ke laman seperti pada gambar di bawah:



Pada laman di atas, klik opsi **Select all** dan klik tombol **Continue**. Biasanya proses centang seperti di atas hanya terjadi ketika kita pertama kali menggunakan GEE di lingkungan Google Colab.

Selanjutnya, kita akan dibawa kembali ke laman notebook Google Colab sebagaimana terlihat pada gambar berikut:

### Bab III Google Earth Engine



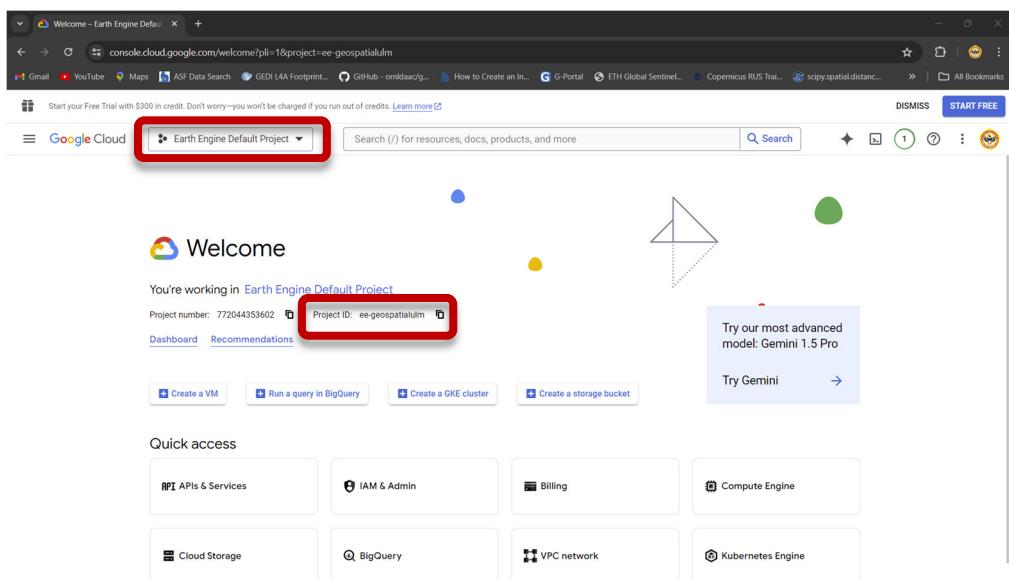
Tambahkan sel kode baru di bawah dengan mengklik tombol **+ Code**. Kemudian di sel kode baru tersebut ketikkan kode berikut dan **Run**.

```
ee.Initialize(project='ee-geospatialulm')
```

Perhatikan bahwa entri **ee-geospatialulm** adalah Project ID yang sudah kita buat pada saat registrasi Earth Engine sebelumnya. Sesuaikan entri ini dengan nama Project ID yang Anda buat.

#### Tips Jika Lupa Project ID Earth Engine Default Project

Buka laman <https://console.cloud.google.com>, sebagaimana pada gambar berikut:



Pada laman Google Cloud di atas, pastikan opsi yang terpilih di samping logo Google Cloud pada sudut kiri atas adalah *Earth Engine Default Project*. Selanjutnya, silahkan cari dan lihat entri *Project ID*. Klik tombol *Copy to clipboard* di samping Project ID untuk mengcopy-paste nama Project ID Anda ke dalam notebook Google Colab. Contoh di atas nama Project ID-nya adalah **ee-geospatialulm**. Tentu saja, Project ID Anda namanya akan berbeda, sesuai dengan nama yang Anda berikan sendiri. Jika nantinya Anda memiliki beberapa akun Google Colab, direkomendasikan nama Project ID-nya sama. Hal ini agar nantinya Anda akan lebih mudah ketika harus copy-paste kode antar akun tanpa harus merombak lagi kode-kode Python Anda.

Berikut adalah kode program lengkap yang harus Anda ketikkan di dalam notebook Google Colab untuk mengakses Citra Sentinel-2 MSI. Kode-kode berikut diketikkan dan dijalankan per sel kode di dalam notebook. Dimana kotak-kotak pembatas (*borders*) antar baris-baris kode di bawah merupakan panduan pembatas antar sel. Hal ini lebih baik dari pada mengetikkan kode-kode ke dalam satu sel kode notebook dan menjalankannya sekaligus.

```
import ee, geemap

ee.Authenticate()

ee.Initialize(project='ee-geospatialulm')

# Penentuan rentang tanggal akuisisi citra
tanggal_awal = '2023-09-01'
tanggal_akhir = '2023-09-30'

# Menentukan titik lokasi citra yang akan ditampilkan
titik = ee.Geometry.Point([114.776874, -3.449037])

s2_col = (
    ee.ImageCollection('COPERNICUS/S2_HARMONIZED')
        .filterDate(tanggal_awal,tanggal_akhir)
        .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE',20))
)

# Visualisasi RGB Sentinel-2 MSI
# menggunakan Geemap (wu, 2020; wu et al., 2019)
rgb_vis = {'min': 0, 'max': 5000, 'bands': ['B12','B11','B4']}

peta = geemap.Map()
peta.centerObject(titik,12)
peta.add_layer(s2_col.median(), rgb_vis, 'Sentinel-2 RGB')
peta
```

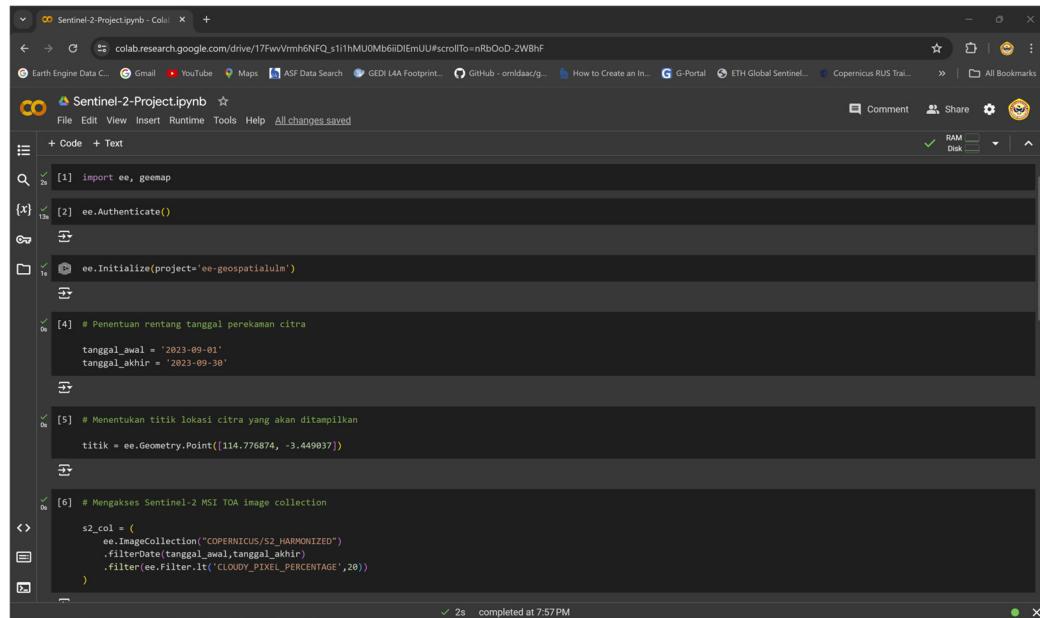
---

*Khusus untuk bagian seterusnya di dalam buku ini, kotak-kotak pemisah pada kode program Python, sebagaimana pada contoh kode di atas, akan berfungsi sebagai panduan pemisah sel kode di dalam notebook Google Colab. Yang artinya kode-kode pada masing-masing kotak harus ditulis di dalam sel terpisah dan dijalankan masing-masing per sel kode.*

---

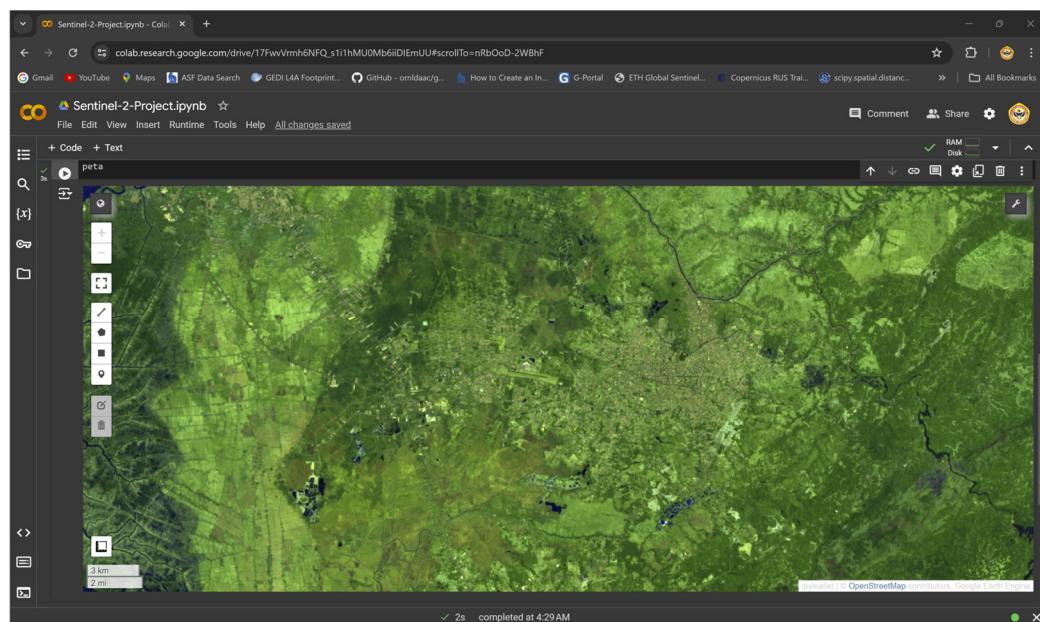
Perhatikan contoh pengetikan kodennya pada notebook Google Colab seperti gambar berikut:

### Bab III Google Earth Engine



```
[1]: import ee, geemap
[2]: ee.Authenticate()
[3]: ee.Initialize(project='ee-geospatialulm')
[4]: # Menentukan rentang tanggal perekaman citra
tangkal_awal = '2023-09-01'
tangkal_akhir = '2023-09-30'
[5]: # Menentukan titik lokasi citra yang akan ditampilkan
titik = ee.Geometry.Point([114.776874, -3.449037])
[6]: # Mengakses Sentinel-2 MSI TOA image collection
s2_col = (
ee.ImageCollection("COPERNICUS/S2_HARMONIZED")
.filterDate(tangkal_awal,tangkal_akhir)
.filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE',20))
```

Output dari kode Python di atas adalah sebagai berikut:



Jika citra komposit Sentinel-2 ditampilkan sebagaimana terlihat pada gambar di atas, *Selamat!* Anda sudah berhasil mengakses koleksi citra Sentinel-2 MSI TOA pada rentang tanggal dan wilayah yang Anda tentukan. Tentu saja, kode di atas baru terbatas hanya untuk menampilkan citra, dan belum mengekstrak atau pun menyimpan informasi apa-apa. Untuk tanggal awal dan tanggal akhir Anda bebas menentukan sesuai keinginan. Termasuk lokasi titik, yang sebenarnya pada contoh kode di atas, koordinat titiknya diambil menggunakan laman Google Map. Anda dapat mengganti koordinat tersebut sesuai keinginan, yang harus diperhatikan format koordinatnya adalah Bujur (*Longitude*) – Lintang (*Latitude*) atau XY.

Entri `ee.ImageCollection("COPERNICUS/S2_HARMONIZED")` tentu saja dicopas dari laman catalog Harmonized Sentinel-2 MSI: MultiSpectral Instrument, Level-1C [https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS\\_S2\\_HARMONIZED](https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_S2_HARMONIZED). Image collection Sentinel-2 difilter secara temporal dengan tanggal awal dan tanggal akhir. Persentase awan pada rentang tanggal Sentinel-2 tersebut juga difilter menggunakan instruksi `filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20))`. Dimana dipilih citra-citra dengan persentase awan kurang dari (*lt: less than*) 20% pada rentang waktu 25-09-2023 hingga 30-09-2023.

Hal yang harus diperhatikan adalah, sebagaimana aturan di dalam list Python atau NumPy array, `tanggal_akhir = '2023-09-30'` ini bersifat eksklusif. Artinya, jika pada lokasi yang kita tunjuk kebetulan terdapat akuisisi Sentinel-2 tepat pada tanggal 30 September 2023, maka akuisisi pada tanggal 30 September 2023 ini tidak akan dimasukkan ke dalam image collection yang kita akses.

Berdasarkan Image collection yang kita filer ini, kemudian nilai spektral diambil nilai tengahnya (median) untuk divisualisasikan, yaitu dengan instruksi `s2_image.median()`. Terkait nilai median ini, silahkan akses laman <https://developers.google.com/earth-engine/apidocs/ee-imagecollection-median> untuk mendapatkan penjelasan yang lebih detail. Selain median, kita juga dapat menggunakan `first`, `mean`, `min`, `max`, atau pun parameter-parameter lainnya. Jika pada rentang waktu yang kita tentukan hanya ditemukan satu akuisisi Sentinel-2 saja, tentu saja hanya satu citra yang akan diakses dan ditampilkan oleh GEE. Jika kemudian GEE tidak menampilkan citra, berarti pada rentang waktu tersebut tidak ditemukan akuisisi citra, atau tidak ditemukan akuisisi citra dengan persentase awan kurang dari 20%. Solusinya adalah rubah rentang waktunya atau rubah persentase awannya.

`rgb_vis = {'min': 0, 'max': 5000, 'bands': ['B12', 'B11', 'B4']}` digunakan untuk mengatur parameter visualisasi citra komposit. Pada entri ini, Anda dapat merubah kombinasi band RGB sesuai komposit warna yang Anda inginkan, misalnya untuk *True Color Composite* (TCC) kombinasi band-band-nya adalah `['B4', 'B3', 'B2']`. `'min': 0, 'max': 5000` merupakan perentangan linier (*linear stretching*) nilai spektral Sentinel-2 yang akan ditampilkan. Nilai ini dapat dirubah sesuai keperluan, jika citra terlalu gelap, turunkan nilai `max` menjadi `3000` misalnya. Jika citra terlalu cerah, naikkan nilai `max` menjadi `7000` misalnya.

Penting untuk diperhatikan bahwa, nilai TOA reflectance (termasuk TOC reflectance juga) Sentinel-2 MSI yang tersedia di server, sudah dikonversi nilainya oleh provider Sentinel-2 menjadi rentang 0 hingga 10000. Hal ini bertujuan untuk efisiensi transfer data via internet. Sebab jika nilai reflectance disimpan dalam format desimal 0 sampai 1, data harus disimpan di server dan kita unduh dalam format *floating point*. Bilangan desimal (*float*) memiliki kapasitas penyimpanan lebih besar dibandingkan dengan format bilangan bulat (*integer*) 0 sampai 10000. Jika kita membuat model kuantitatif (seperti regresi) menggunakan nilai-nilai spektral Sentinel-2 MSI nantinya, tentu saja rentang nilai reflectance 0 sampai 10000 ini harus dikonversi menjadi notasi desimal 0 sampai 1. Yaitu dengan cara membaginya dengan 10000.

Entri `peta.centerObject(titik, 12)` digunakan untuk mengatur titik tengah visualisasi citra, berdasarkan `titik = ee.Geometry.Point([114.776874, -3.449037])` yang sudah kita atur sebelumnya. Adapun angka 12 merupakan level *zooming* tampilan wilayah citra. Tentu saja, angka 12 ini sifatnya coba-coba. Misalnya kita coba 10 dulu, jika ternyata tampilan wilayahnya terlalu kecil, perbesar angkanya menjadi 12 misalnya.

Kita sudah berhasil mengakses salah satu koleksi citra dengan menggunakan GEE dan Python via Google Colab, yaitu Citra Sentinel-2 MSI TOA. Tentu saja, Anda dapat mencoba-coba untuk mengakses data geospasial lainnya, dengan terlebih dahulu menjelajahi katalog data Earth Engine di laman <https://developers.google.com/earth-engine/datasets> atau <https://gee-community-catalog.org/projects/>.

## Memotong Citra Menggunakan Shapefile

Pada contoh kode sebelumnya kita mengakses Sentinel-2 MSI TOA tanpa memotong (*clip*) citra menggunakan wilayah tertentu. Jika tanpa pemotongan seperti ini, tentu saja Sentinel-2 yang terakses dan ditampilkan adalah seluruh dunia. Terhadap citra-citra yang memiliki resolusi spasial tinggi seperti Sentinel-2, jarang sekali kasusnya dimana kita melakukan analisis untuk seluruh dunia. Bahkan untuk satu wilayah negara atau satu provinsi pun sangat jarang kemungkinannya, kecuali provinsi-provinsi yang wilayah administrasinya tidak terlalu besar.

Pada umumnya analisis Sentinel-2 hanya dilakukan pada wilayah yang relatif kecil, misalnya satu kota atau satu kabupaten. Untuk keperluan seperti ini, kita harus melakukan pemotongan citra mengikuti batas wilayah yang diperlukan terlebih dahulu. Dalam hal ini, kita sudah harus menyediakan batas wilayah yang akan digunakan untuk memotong citra, misalnya dalam format vektor (shapefile).

Sebagai contoh, masih melanjutkan contoh kode sebelumnya, tambahkan sel-sel kode baru pada notebook Google Colab. Kemudian ketikkan dan eksekusi kode-kode di bawah ini di dalam sel-sel kode terpisah:

```
!pip install mapclassify
```

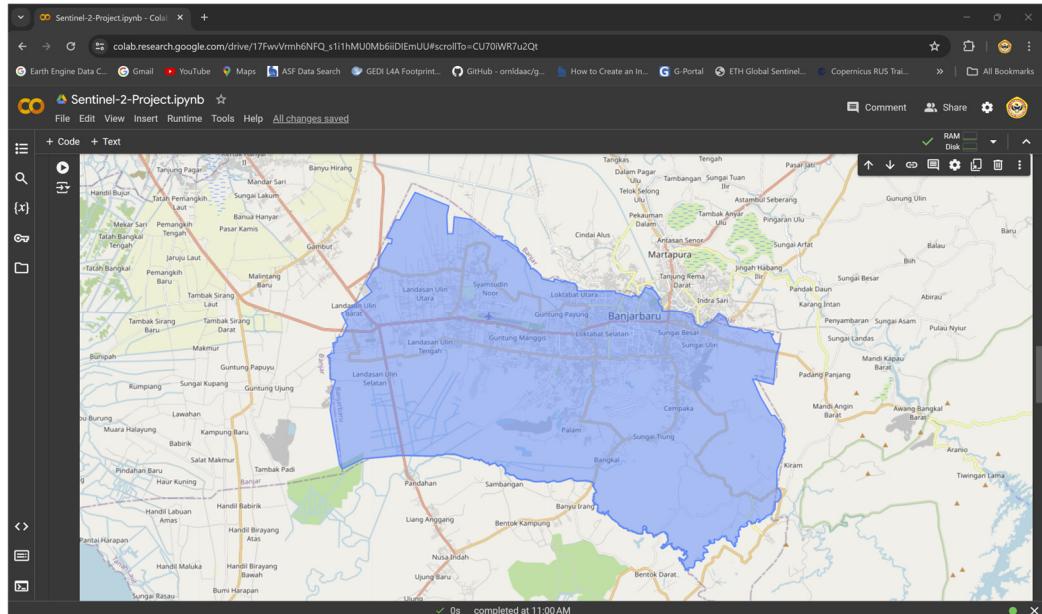
```
# Mengimpor shapefile
import geopandas as gpd
from google.colab import drive
drive.mount('/content/gdrive')
path = '/content/gdrive/MyDrive/geebook'
banjarbaru_shp = gpd.read_file(path + '/vector/Banjarbaru.shp')
banjarbaru_shp.explore()
```

```
# Konversi shapefile ke Geometry GEE
import json
banjarbaru_js = json.loads(banjarbaru_shp.to_json())
banjarbaru_fc = ee.FeatureCollection(banjarbaru_js)
banjarbaru = ee.Geometry(banjarbaru_fc.geometry())
```

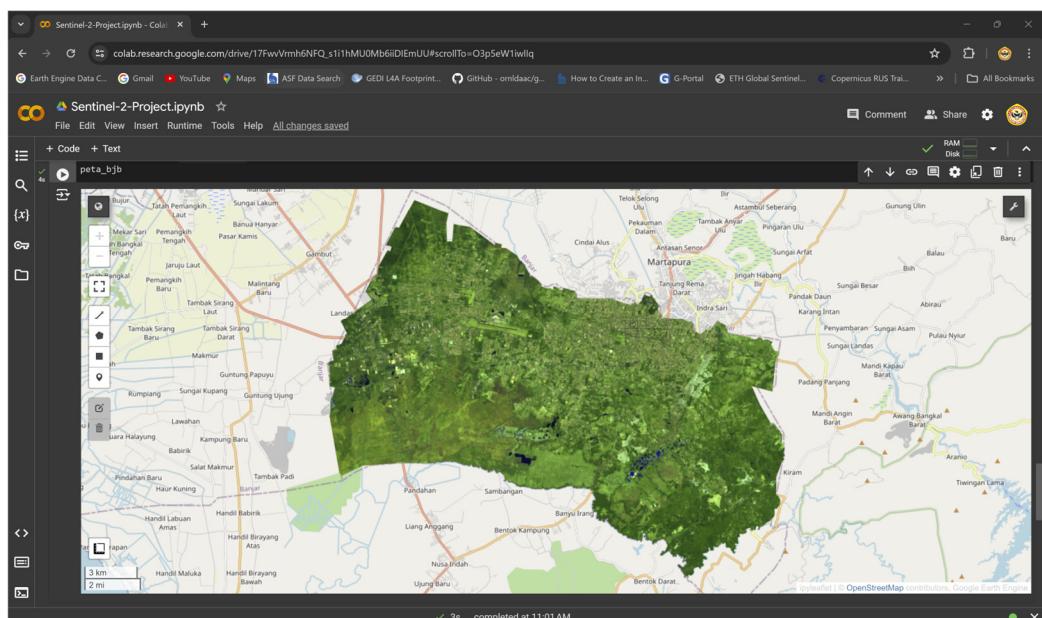
```
# Memilih band dan memotong citra menggunakan batas wilayah
s2_image = (
    (s2_col.median())
    .select('B2','B3','B4','B8','B11','B12'))
s2_banjarbaru = s2_image.clip(banjarbaru)
```

```
# Visualisasi RGB Sentinel-2 MSI Kota Banjarbaru
rgb_vis = {'min': 0, 'max': 5000, 'bands': ['B12', 'B11', 'B4']}
peta_bjb = geemap.Map()
peta_bjb.centerObject(banjarbaru, 12)
peta_bjb.add_layer(s2_banjarbaru, rgb_vis, 'Sentinel-2 RGB Banjarbaru')
peta_bjb
```

Output dari kode-kode di atas adalah sebagai berikut:



Dan Citra Sentinel-2 MSI yang sudah dipotong akan tampak seperti gambar berikut:



### Bab III Google Earth Engine

Shapefile yang dibuka menggunakan GeoPandas tidak dapat digunakan secara langsung untuk memotong citra dalam format Earth Engine (EE) Image. Dalam hal ini, shapefile harus dikonversi terlebih dahulu menjadi *Geometry*. Lihat kembali Tabel 3.1 pada bagian sebelumnya. Faktanya, shapefile bahkan tidak dapat langsung dikonversi menjadi Geometry.

Sebagaimana terlihat pada kode di atas, tahapan lengkapnya adalah, shapefile dikonversi menjadi format *JavaScript Object Notation* (JSON) (Crockford, 2006; Bray, 2014; Pezoa et al., 2016), kemudian JSON dikonversi menjadi *Feature Collection* (lihat Tabel 3.1), selanjutnya Feature Collection baru dikonversi menjadi Geometry. Dengan Geometry ini lah kita mengclip GEE Image.

Instruksi `.select('B2', 'B3', 'B4', 'B8', 'B11', 'B12')` digunakan untuk memilih band-band yang diperlukan. Terutama untuk diekspor menjadi GeoTIFF nantinya. Instruksi `s2_image.clip(banjarbaru)` akan memotong citra mengikuti batas wilayah spesifik yang sudah kita tentukan sebelumnya.

Perhatikan kode `peta_bjb.centerObject(banjarbaru, 12)`. Kode ini sedikit berbeda dibandingkan contoh kode sebelumnya, yaitu `peta.centerObject(titik, 12)`. Pada kode `peta.centerObject(titik, 12)`, kita menggunakan geometri titik (`titik`) sebagai titik pusat atau fokus di dalam visualisasi citra menggunakan GEE. Sementara pada kode `peta_bjb.centerObject(banjarbaru, 12)`, kita mengambil titik tengah dari poligon administrasi Banjarbaru (`banjarbaru`) sebagai titik pusat di dalam visualisasi citra. Alternatif lainnya, kita juga dapat secara langsung mengambil titik pusat dari citra yang akan divisualisasikan, yaitu dengan kode `peta_bjb.centerObject(s2_banjarbaru, 12)`. Tentu saja, citranya sudah harus dipotong menurut batas wilayah tertentu terlebih dahulu.

### Mengekspor Data ke Google Drive

Citra Sentinel-2 yang sudah dipotong menurut wilayah tertentu, dapat diekspor ke dalam Google Drive kita. Sehingga dapat digunakan untuk keperluan selanjutnya, misalnya diunduh ke dalam laptop kita untuk nantinya akan dianalisis secara offline.

Masih melanjutkan kode sebelumnya, untuk mengekspor data GEE Image ke Google Drive ketikkan kode berikut pada sel kode baru:

```
# Ekspor citra ke Google Drive
task_s2 = ee.batch.Export.image.toDrive(
    image=s2_banjarbaru,
    description='S2_Banjarbaru',
    folder='Citra',
    scale=10,
    crs='EPSG:32750'
)
task_s2.start()
```

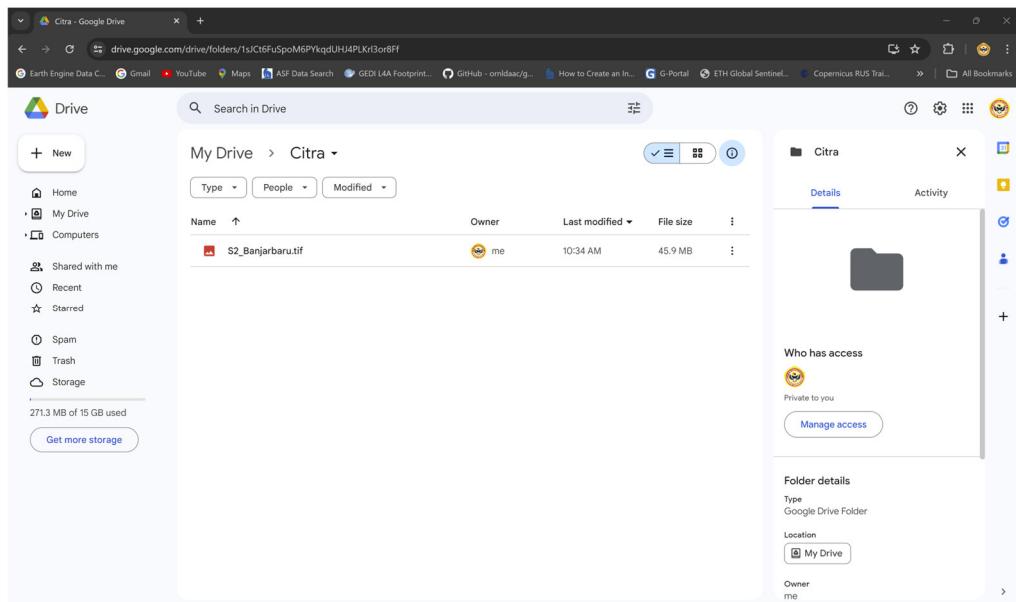
Kode di atas akan mengekspor Sentinel-2 ke Google Drive ke dalam folder `Citra`, dengan nama `S2_Banjarbaru` (format default GeoTIFF). `scale=10` adalah resolusi spasial yang kita tentukan, sementara `crs='EPSG:32750'` adalah Sistem Koordinat UTM WGS 1984 Zone 50S. Setelah kode di atas dijalankan, untuk mengetahui status ekspor data, jalankan kode berikut:

```
task_s2.status()
```

Jika outputnya seperti berikut:

```
{'state': 'RUNNING',
'description': 'S2_Banjarbaru',
'priority': 100,
'creation_timestamp_ms': 1717850819671,
'update_timestamp_ms': 1717850826813,
'start_timestamp_ms': 1717850823256,
'task_type': 'EXPORT_IMAGE',
'attempt': 1,
'id': 'BDQRYOWFM3RAN5WP3UV5SC60',
'name': 'projects/ee-geospatialuim/operations/BDQRYOWFM3RAN5WP3UV5SC60'}
```

Berarti proses ekspor masih berlangsung atau belum selesai, sebab ada entri status '`state`': '`RUNNING`'. Jika nanti statusnya sudah berubah menjadi '`state`': '`COMPLETED`', berarti proses ekspor sudah selesai. Jika statusnya nanti justru menjadi '`state`': '`READY`', besar kemungkinan sudah ada citra dengan nama yang sama persis di dalam folder yang sama. Berikut adalah Sentinel-2 yang sudah selesai diekspor ke dalam Google Drive, di dalam folder `Citra`.

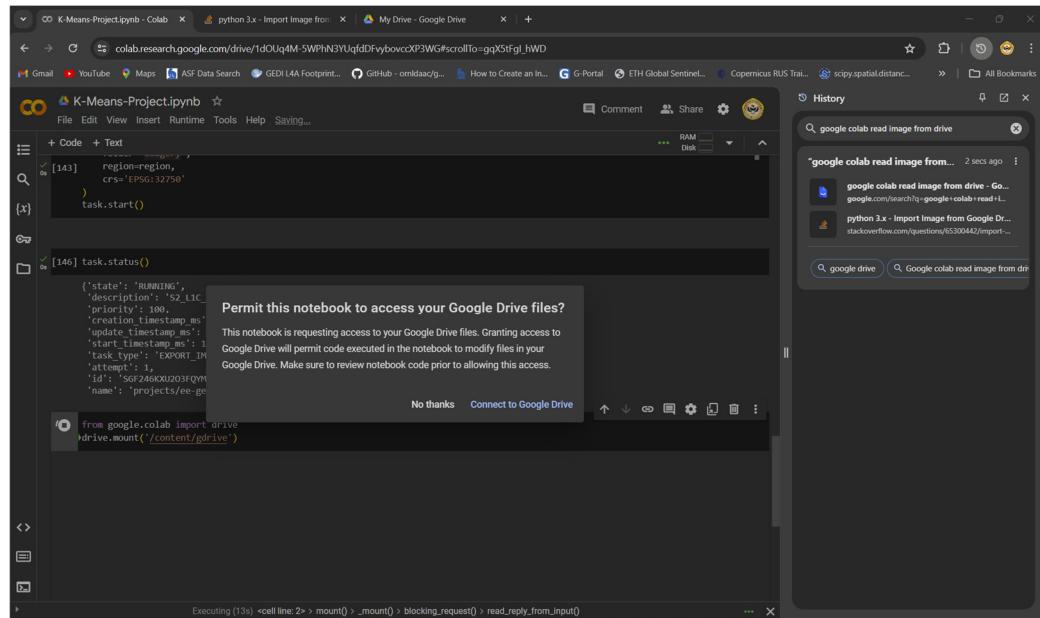


Secara bawaan, GEE hanya memungkinkan ekspor citra ke Google Drive maksimum `1e8` atau 100.000.000 pixel. Jika kita mengekspor citra dengan jumlah pixel melebihi `1e8`, maka akan terjadi error. Akan tetapi, GEE sebenarnya masih mendukung ekspor citra ke dalam Google Drive hingga `1e13` atau 10.000.000.000.000 pixel. Caranya adalah kita tambahkan parameter `maxPixels=1e13`, sebagaimana pada contoh kode berikut:

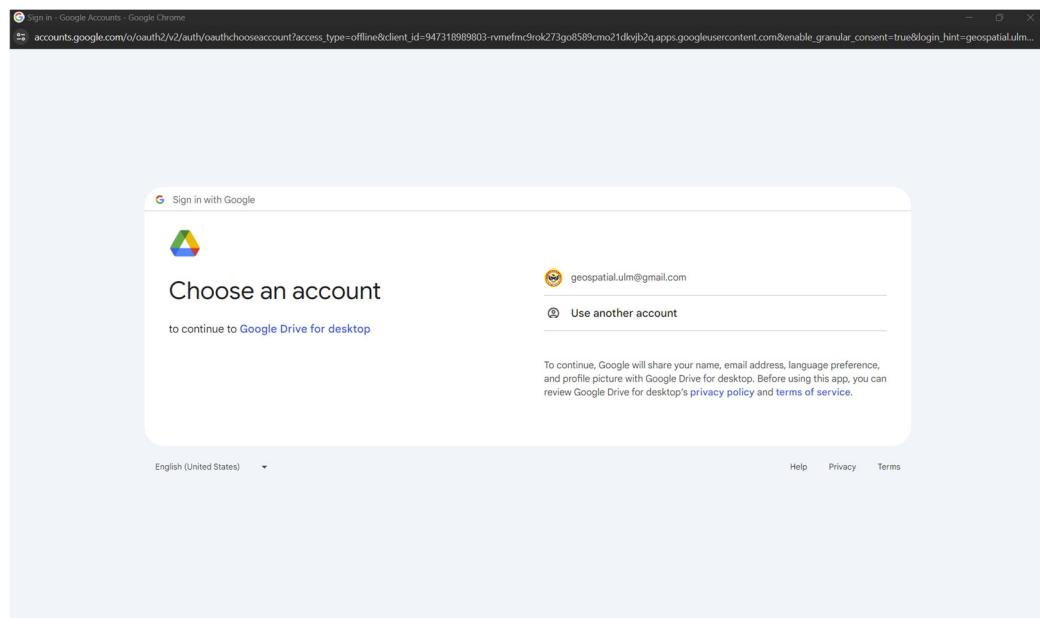
```
# Eksport citra ke Google Drive
task_s2 = ee.batch.Export.image.toDrive(
    image=s2_banjarbaru,
    description='S2_Banjarbaru',
    folder='citra',
    scale=10,
    maxPixels=1e13,
    crs='EPSG:32750'
)
task_s2.start()
```

### Bab III Google Earth Engine

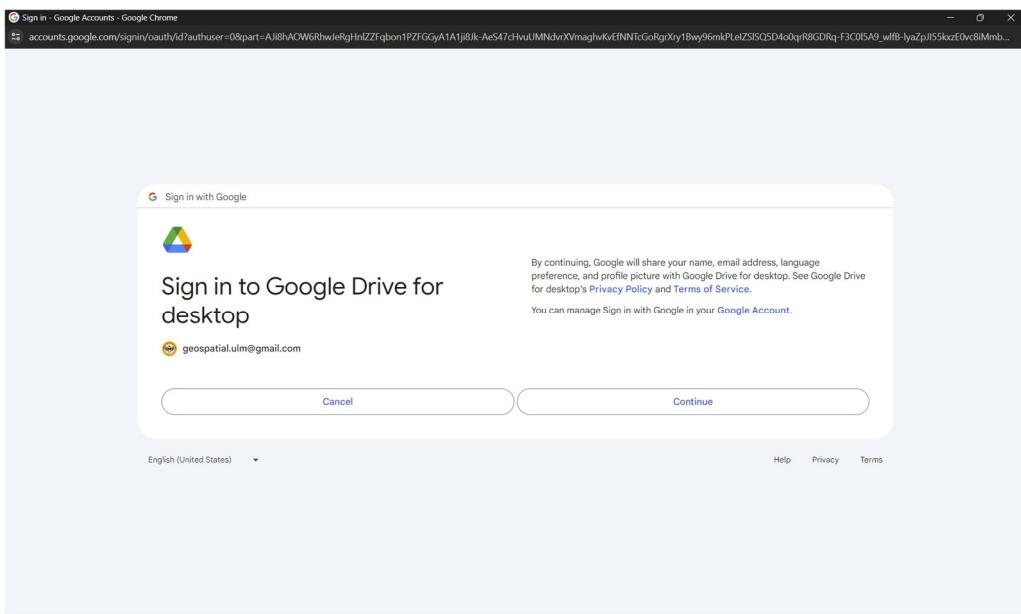
**CATATAN**, ketika kita mengekspor citra dari GEE ke Google Drive, untuk alasan sekuriti biasanya akan muncul laman-laman konfirmasi seperti pada gambar-gambar berikut:



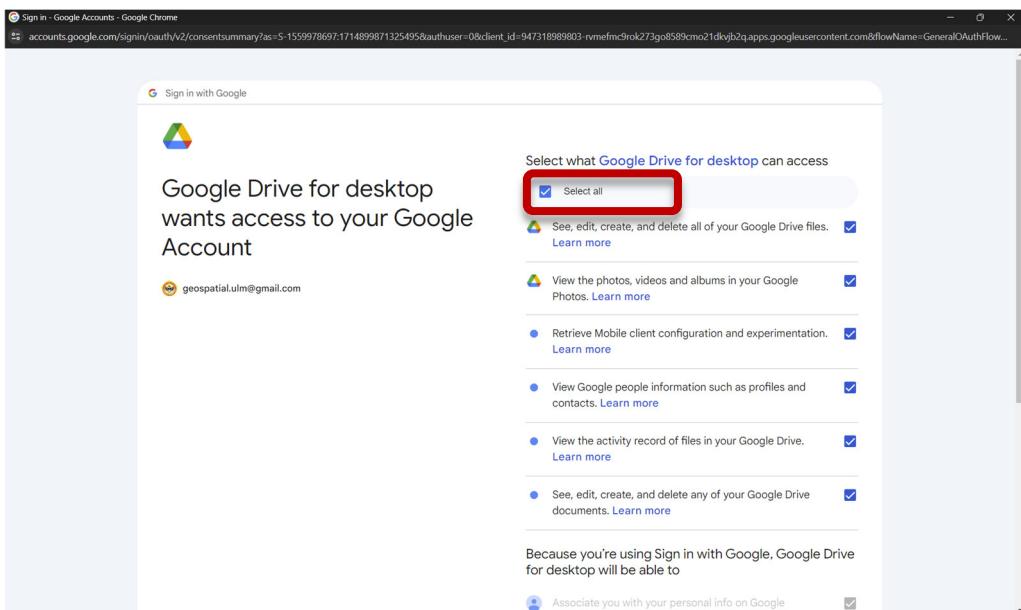
Pada laman di atas, langsung klik tombol **Connect to Google Drive**. Selanjutnya kita akan dibawa ke laman seperti berikut:



Pada laman di atas, klik atau ketikkan akun Google Drive kita. Jika akunnya sama dengan akun Google Colab kita, biasanya kita tidak akan diminta *password* lagi. Selanjutnya, kita akan di bawa ke laman seperti berikut:



Pada laman di atas langsung klik tombol **Continue**.



Pada laman di atas, centang **Select all** dan klik tombol **Continue**. Biasanya proses centang seperti di atas hanya terjadi ketika kita pertama kali mengekspor citra dari GEE ke Google Drive.

Setelah citra berhasil diekspor ke Google Drive dalam format GeoTIFF, tentu saja file citra dapat diunduh manual sebagaimana kita mengunduh file-file biasa dari Google Drive. Dan untuk selanjutnya, citra dalam format GeoTIFF ini dapat dibuka dengan semua perangkat lunak geospasial, atau pun dianalisis menggunakan Python secara offline. Karena kita hanya memilih band-band tertentu, yaitu `.select('B2', 'B3', 'B4', 'B8', 'B11', 'B12')`, maka hanya band-band ini yang akan terekspor menjadi GeoTIFF ke dalam Google Drive kita.

*Berdasarkan pengalaman, kita tidak harus menunggu proses ekspor ke Google Drive sampai selesai, jika kita memang harus keluar dari Google Colab. Proses ekspor citra dari GEE ke dalam Google Drive akan terus berlangsung sampai selesai, meskipun kita logout dari Google Colab, memutuskan akses internet, atau pun mematikan laptop.*

## Menggunakan GEE Assets

Pada bagian sebelumnya kita sudah membahas tentang bagaimana mengakses data geospasial dari Google Drive, maupun mengekspor data geospasial ke Google Drive. Penyimpanan data di dalam Google Drive sebenarnya hanya seperti memindah file dari *local drive* di laptop kita ke drive di internet. Dengan kata lain, akses GEE ke data geospasial kita di dalam Google Drive sebenarnya masih tidak begitu interaktif. Jika kita ingin akses GEE ke data geospasial kita lebih interaktif, kita dapat menyimpan data geospasial kita ke dalam *GEE Assets*.

Di dalam GEE Assets, kita dapat menyimpan data geospasial ke dalam format GEE, seperti image collection, feature collection, dan sebagainya (lihat kembali Tabel 3.1). Bahkan jika diinginkan, kita dapat membagi (*sharing*) data kita ke publik, misalnya produk riset kita, agar nantinya dapat dengan mudah untuk diakses dan digunakan publik, sebagaimana Earth Engine Data Catalog (<https://developers.google.com/earth-engine/datasets>).

Sebagai contoh, masih melanjutkan kode sebelumnya, tambahkan sel kode baru pada notebook Google Colab. Kemudian ketikkan dan jalankan kode-kode berikut:

```
# Ekspor citra ke GEE Assets
task_s2 = ee.batch.Export.image.toAsset(
    image=s2_banjarbaru,
    description='Sentinel2_Banjarbaru',
    assetId='projects/ee-geospatialulm/assets/Sentinel-2_Banjarbaru',
    scale=10,
    crs='EPSG:32750'
)
task_s2.start()
```

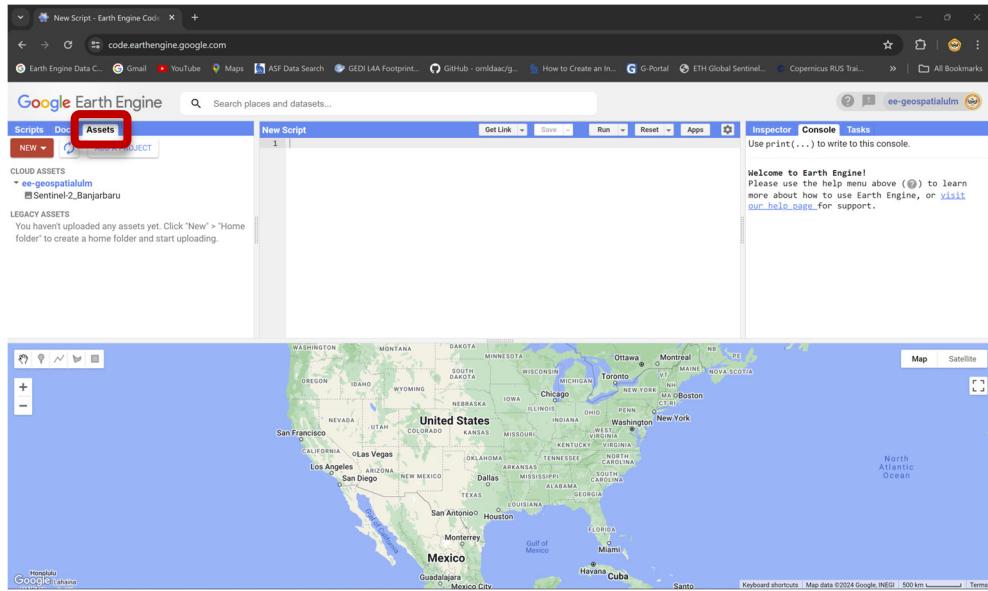
Untuk mengetahui status proses ekspor data, jalankan kode berikut:

```
task_s2.status()
```

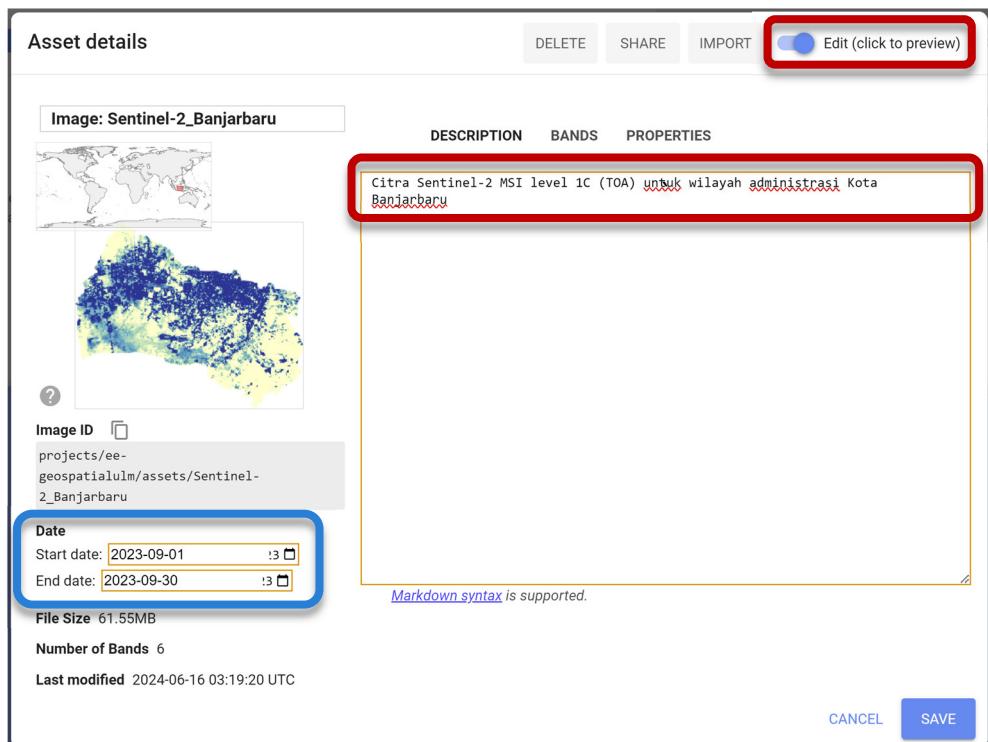
Jika state-nya **RUNNING** seperti di bawah, berarti proses ekspor masih berlangsung. Jika proses ekspor data selesai, state-nya akan berubah menjadi **COMPLETED**.

```
{'state': 'RUNNING',
'description': 'Sentinel2_Banjarbaru',
'priority': 100,
'creation_timestamp_ms': 1718507592390,
'update_timestamp_ms': 1718507597266,
'start_timestamp_ms': 1718507597231,
'task_type': 'EXPORT_IMAGE',
'attempt': 1,
'id': 'Z67VVCRXXMZWHE33N6VVHK42',
'name': 'projects/ee-geospatialulm/operations/Z67VVCRXXMZWHE33N6VVHK42'}
```

Jika prosesnya sudah **COMPLETED**, buka laman <https://code.earthengine.google.com>, pada laman yang muncul klik tab **Assets**, sebagaimana terlihat pada gambar berikut:



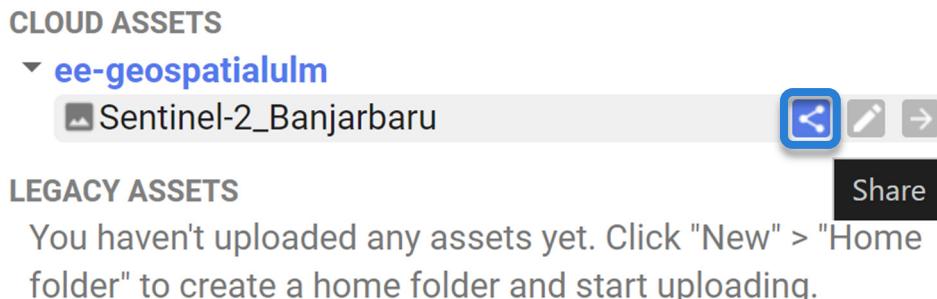
Pada gambar di atas, pada **CLOUD ASSETS ee-geospatialulm** terdapat sebuah image baru, dengan nama **Sentinel-2\_Banjarbaru**. Sesuai dengan assetId yang pada waktu ekspor data. Klik ganda pada **Sentinel-2\_Banjarbaru** di atas, sehingga ditampilkan jendela seperti berikut:



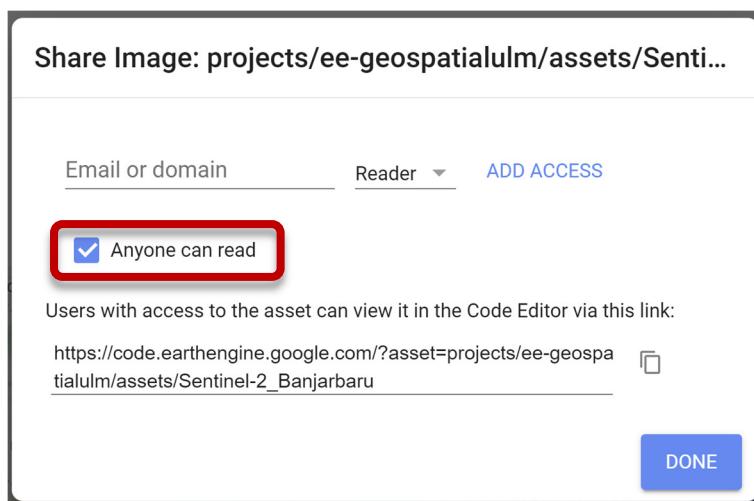
### Bab III Google Earth Engine

Pada jendela di atas, aktifkan tombol **Edit** di sudut kanan atas, kemudian lakukan input atau perubahan metadata sesuai keperluan. Misalnya **DESCRIPTION**, **Start date** dan **End date**, dan sebagainya. Jika sudah selesai, klik tombol **SAVE**.

Selanjutnya, jika kita perlu untuk membagi data kita ke publik atau tujuan tertentu, klik tombol **Share**. Sebagaimana terlihat pada gambar berikut:



Selanjutnya, akan ditampilkan jendela berikut:



Pada jendela di atas, jika ingin dibagi ke publik, centang **Anyone can read**. Kemudian klik **Done**. Tentu saja, jika data tidak ingin dibagi ke publik, masukkan hanya alamat email orang-orang yang memang kita beri hak untuk mengakses data kita. Jika kita punya institusi yang punya alamat email dengan domain sendiri, kita juga dapat memasukkan domain seperti **ulm.ac.id**. Sehingga siapa pun yang memiliki email dengan domain **ulm.ac.id** dapat mengakses data tersebut.

Selanjutnya, jika data kita terbuka untuk publik, baik kita atau pun orang lain, dapat mengakses image yang sudah kita ekspor dengan sintaks sebagai berikut:

```
# Mengakses GEE Assets
s2_bjb = (
    ee.Image('projects/ee-geospatialulm/assets/Sentinel-2_Banjarbaru')
)
```

## GEE Menggunakan JupyterLab/Visual Studio Code

Salah satu kelebihan implementasi GEE menggunakan Python adalah kita dapat melakukan akses dan pemrosesan citra di dalam JupyterLab atau VS Code yang sudah terinstal di laptop kita. Tentu saja, ketika kita mengakses data geospasial dari GEE, laptop kita tetap harus terkoneksi ke jaringan internet. Jika sumberdaya laptop kita, misalnya kapasitas RAM atau GPU, lebih tinggi dibandingkan dengan yang disediakan oleh runtime Google Colab, tentu saja pemrosesan data geospasial di dalam laptop kita akan lebih efisien. Jika RAM laptop Anda 16 GB atau lebih, Anda direkomendasikan menjalankan GEE via JupyterLab atau VS Code, dari pada Google Colab versi gratis yang hanya menyediakan RAM sekitar 12 GB. Terlebih, Google Colab free hanya memungkinkan runtime untuk berjalan maksimum 12 jam, tergantung pada ketersediaan sumberdaya dan pola penggunaan kita (<https://research.google.com/colaboratory/faq.html>).

Sebagaimana sudah dijelaskan pada Bab I, bahwa sangat direkomendasikan Anda menjalankan GEE di dalam lingkungan Anaconda tersendiri atau terpisah. Hal ini untuk menghindari konflik atau inkompatibilitas paket. Silahkan ikuti instruksi pada Bab I halaman 25, untuk membuat Anaconda environment. Buat sebuah Anaconda environment baru, misalnya dengan nama gee, kemudian aktifkan environment gee dan instal JupyterLab. Untuk VS Code bisa langsung menggunakan environment gee sebagaimana dijelaskan pada halaman 40. Di dalam environment gee, Anda wajib menginstal paket Earth Engine API dan Geemap.

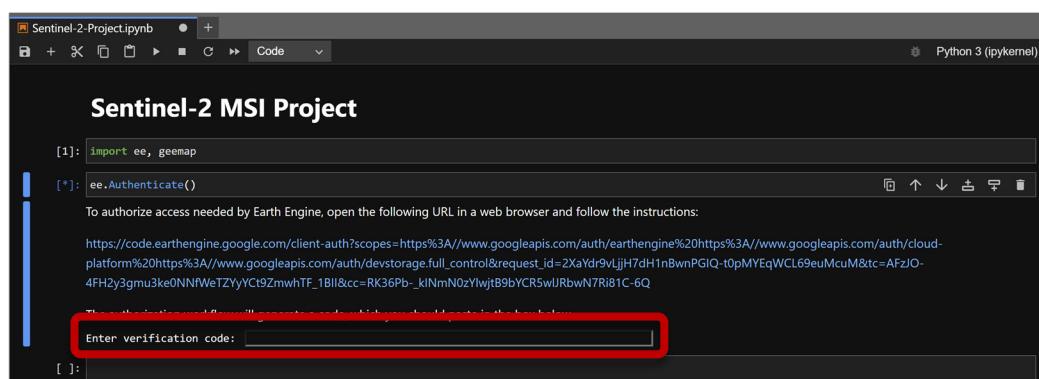
Earth Engine API dapat diinstal via Anaconda Prompt dengan instruksi berikut.

```
conda install conda-forge::earthengine-api
```

Dan Geemap dapat diinstal via Anaconda Prompt dengan instruksi berikut.

```
conda install conda-forge::geemap
```

Tentu saja jika diperlukan, di dalam environment gee dapat juga diinstal paket-paket lain, seperti Pandas, GeoPandas, Matplotlib, GDAL, Rasterio, NumPy, SciPy, Mapclassify, dan sebagainya. Setelah semua paket yang diperlukan terinstal, coba jalankan JupyterLab atau VS Code, sesuai keinginan Anda. Kemudian buat sebuah notebook baru. Beri nama **Sentinel-2-Project** misalnya, sebagaimana gambar di bawah, dimana gambar di bawah menggunakan JupyterLab.



```
[1]: import ee, geemap
[*]: ee.Authenticate()

To authorize access needed by Earth Engine, open the following URL in a web browser and follow the instructions:
https://code.earthengine.google.com/client-auth?scopes=https%3A//www.googleapis.com/auth/earthengine%20https%3A//www.googleapis.com/auth/cloud-platform%20https%3A//www.googleapis.com/auth/devstorage.full_control&request_id=2XaYdr9LijH7dH1nBvnPGIQ-t0pMYEqWCL69euMcuM&tc=AFzJO-4FH2y3gmu3ke0NNWeTZyYCT9ZmwhTF_1BII&cc=RK36Pb_kINmN0zYwjB9bYCR5wJRbwN7R181C-6Q

Enter verification code:
```

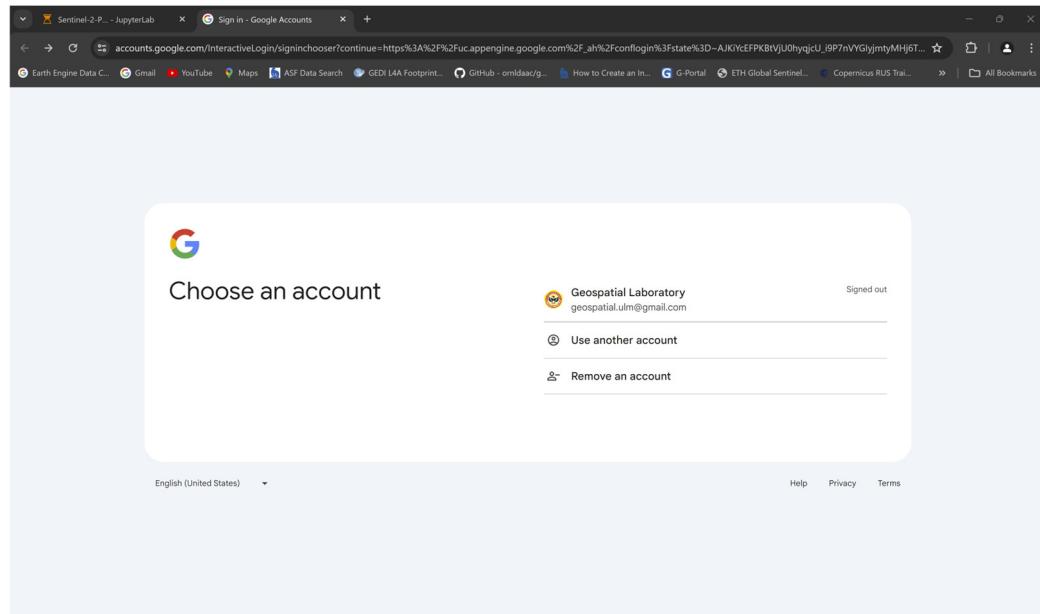
Sebagaimana terlihat pada gambar di atas, ketikkan kode-kode berikut di sel kode terpisah:

### Bab III Google Earth Engine

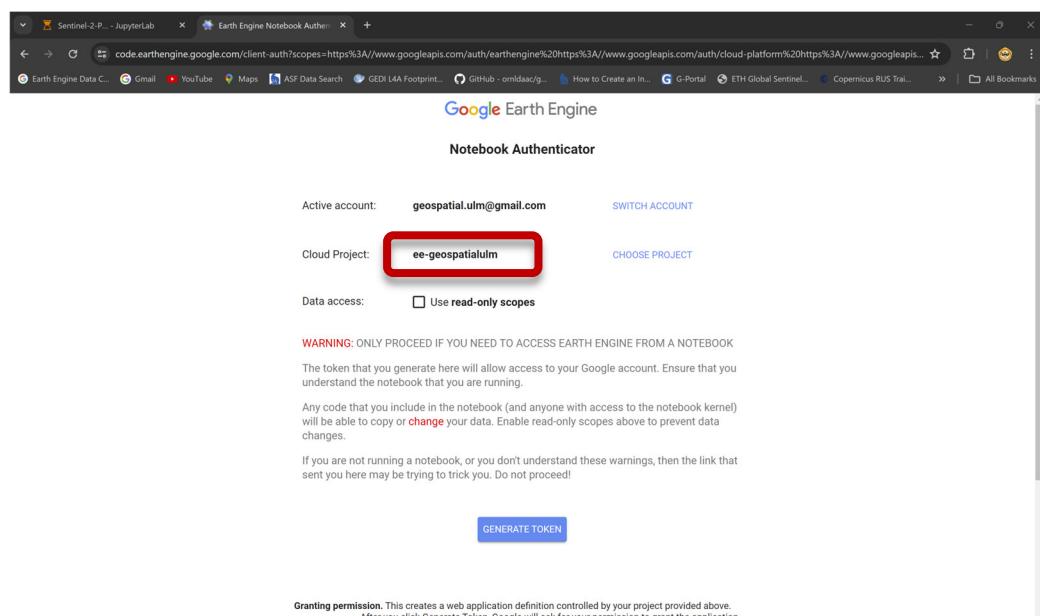
```
import ee, geemap
```

```
ee Authenticate()
```

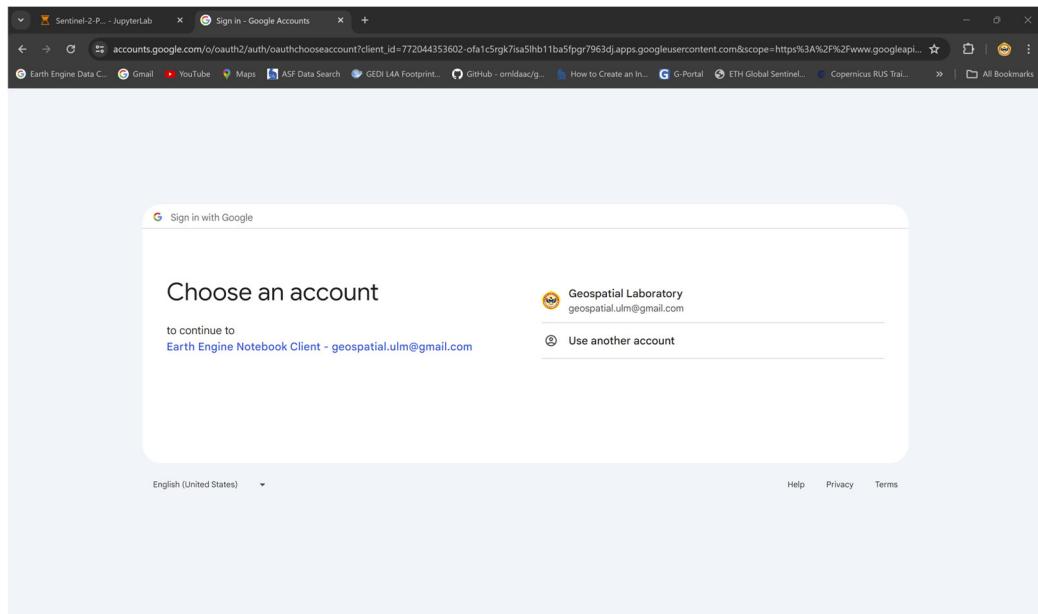
Ketika kita menjalankan instruksi `ee.Authenticate()` di dalam JupyterLab, akan muncul instruksi **Enter verification code:** sebagaimana terlihat pada gambar di atas. Dan kita akan dibawa ke laman **Choose an account** seperti pada gambar di bawah:



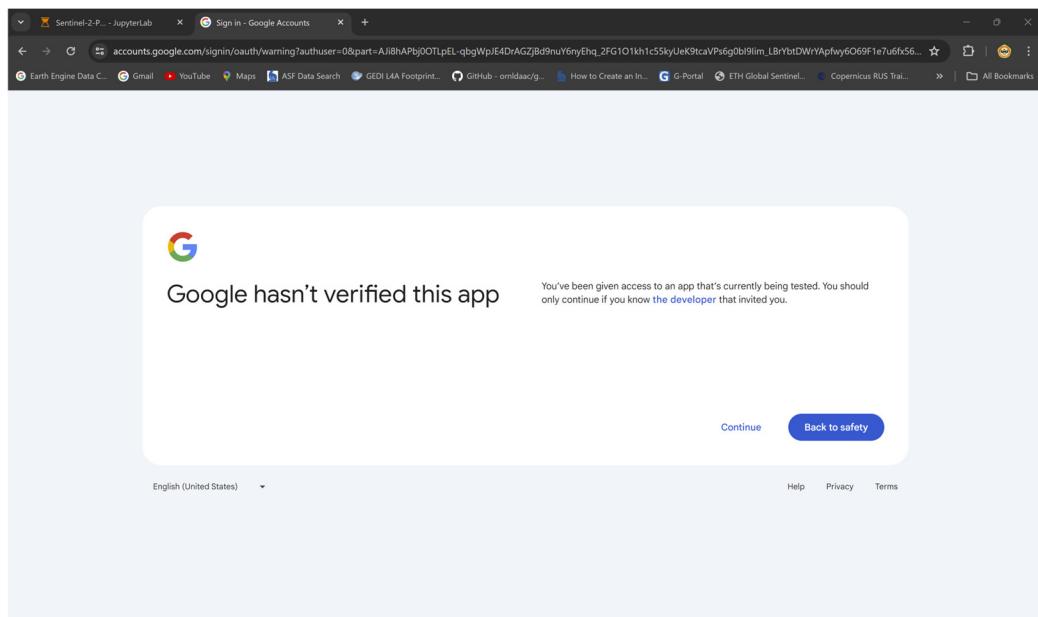
Pilih dan klik atau ketikkan Gmail akun kita yang sudah teregistrasi di GEE sebelumnya, kita juga akan diminta memasukkan password. Selanjutnya kita akan di bawa ke laman seperti berikut:



Pada laman di atas, pastikan akun dan cloud project-nya sesuai dengan yang kita gunakan di GEE. Kemudian klik tombol biru **GENERATE TOKEN**. Kita akan dibawa lagi ke laman berikut:

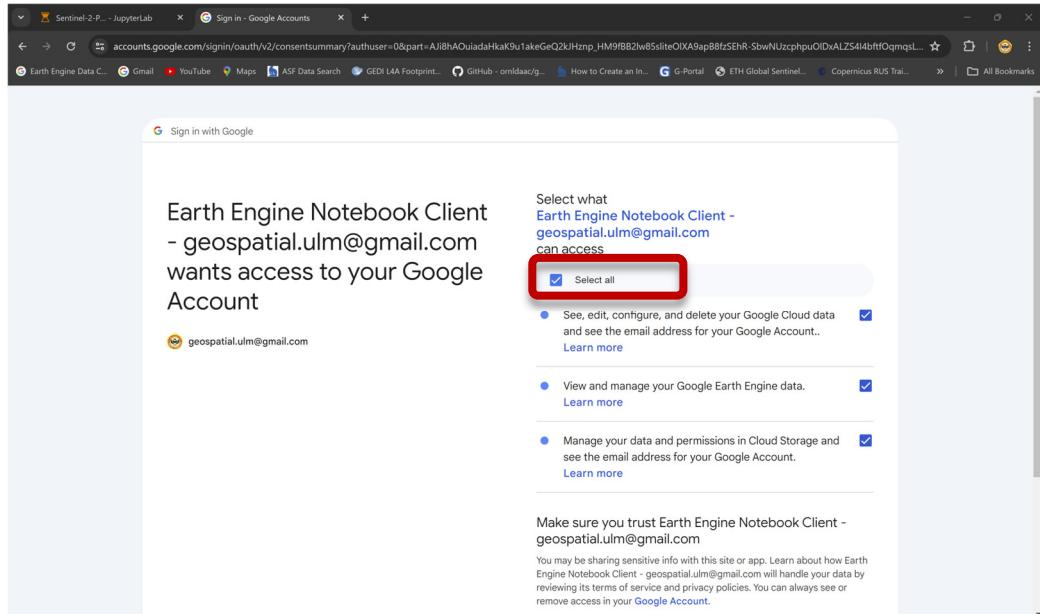


Pilih akun GEE kita, dan kita akan dibawa ke laman berikut:

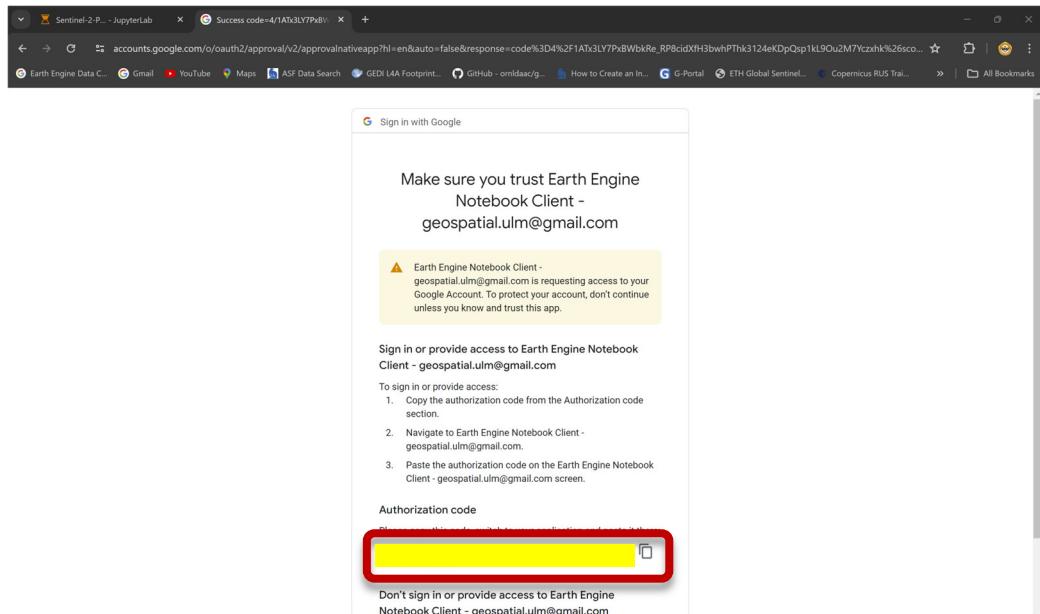


Pada laman di atas, klik tombol **Continue**. Kita akan dibawa lagi ke laman berikut:

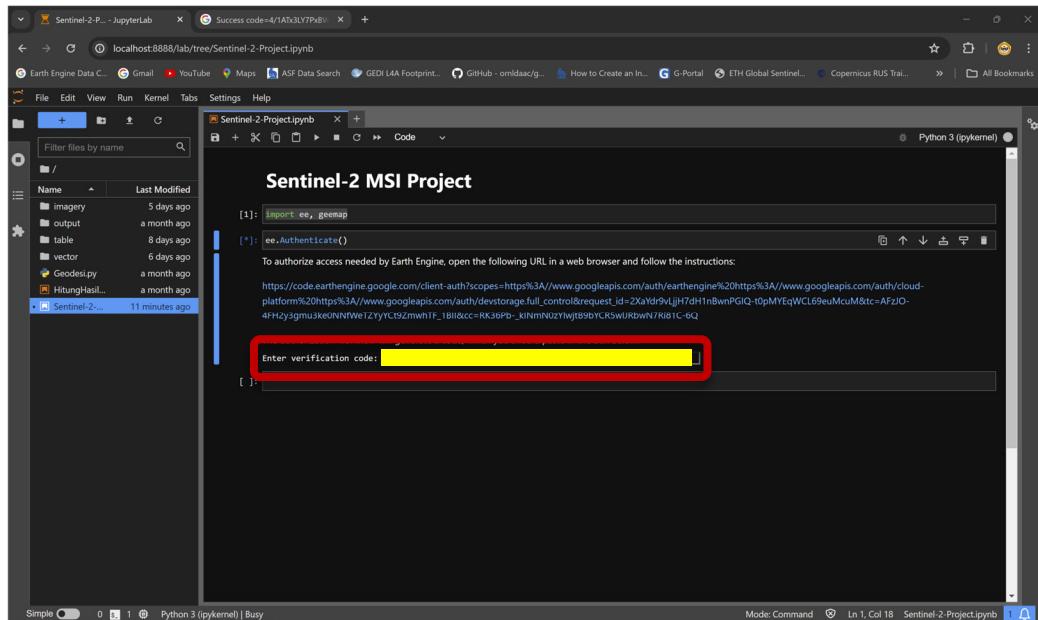
### Bab III Google Earth Engine



Pada laman di atas, centang **Select all** dan klik tombol **Continue**. Kita akan dibawa ke laman **Authorization code** seperti gambar berikut:

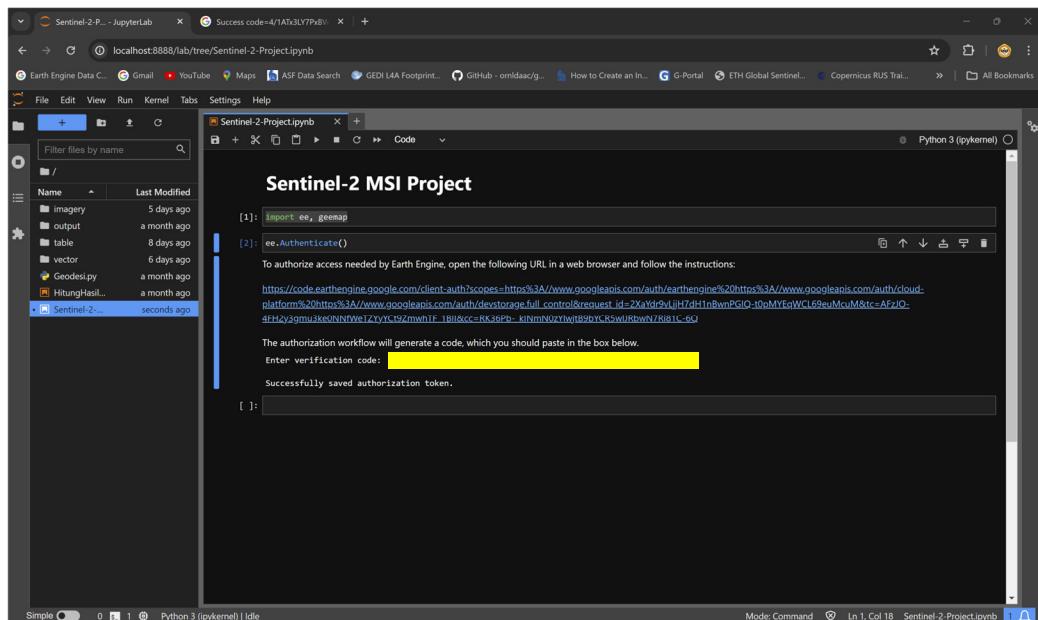


Pada laman di atas, copy **Authorization code** yang diberikan. Kemudian kembali ke JupyterLab dan paste kode yang diberikan ke kolom **Enter verification code**: sebagaimana gambar berikut:



The screenshot shows a JupyterLab interface with a Python 3 (ipykernel) kernel. In the code editor, line 1 contains `[1]: import ee, geemap` and line 2 contains `[2]: ee.Authenticate()`. A URL for authorization is displayed below the code. The terminal output shows a long URL starting with `https://code.earthengine.google.com/client-auth?scopes=https%3A//www.googleapis.com/auth/earthengine%20https%3A//www.googleapis.com/auth/cloud-platform%20https%3A//www.googleapis.com/auth/devstorage.full_control&request_id=2XaYdr9vUjjH7dh1nBwnPGI0-10pMYEqWCL69euMcuM&tc=AFzJO-4fHzY3gmuskeONNWeTZYyC19z/mwhTF-18ll&cc=RK36Pd-_kINmN0ZyIwjIB99DYCR5wURbwN7R81C-6Q`. Below the URL, there is an input field labeled "Enter verification code:" which is highlighted with a red box.

Setelah kode otorisasi dipaste, tekan tombol **Enter** di keyboard. Selanjutnya, akan ada keterangan **Successfully saved authorization token** sebagaimana pada gambar berikut:



The screenshot shows a JupyterLab interface with a Python 3 (ipykernel) kernel. In the code editor, line 1 contains `[1]: import ee, geemap` and line 2 contains `[2]: ee.Authenticate()`. A URL for authorization is displayed below the code. The terminal output shows the message "Successfully saved authorization token." followed by a blank line.

Setelah otorisasi sukses, kita sudah dapat mengakses GEE via JupyterLab atau VS Code. Sampai di sini, jika diinginkan, kita dapat logout dari akun Gmail kita. Jika Anda menggunakan Microsoft Windows, maka kode otorisasi GEE akan tersimpan di dalam sebuah file dengan nama **credentials**, di dalam folder **C:\Users>Nama\_User\.config\earthengine**. Selama file **credentials** ini tidak terhapus, pindah lokasi, atau berubah, maka kita tidak perlu melakukan otorisasi berulang-ulang setiap kali mengakses GEE via JupyterLab atau VS Code. Sepanjang akun GEE dan cloud project yang digunakan juga tetap sama.

### Bab III Google Earth Engine

Selanjutnya, ketikkan kode-kode berikut ke dalam sel-sel kode terpisah:

```
ee.Initialize(project='ee-geospatialulm')

# Penentuan rentang tanggal akuisisi citra
tanggal_awal = '2023-09-01'
tanggal_akhir = '2023-09-30'

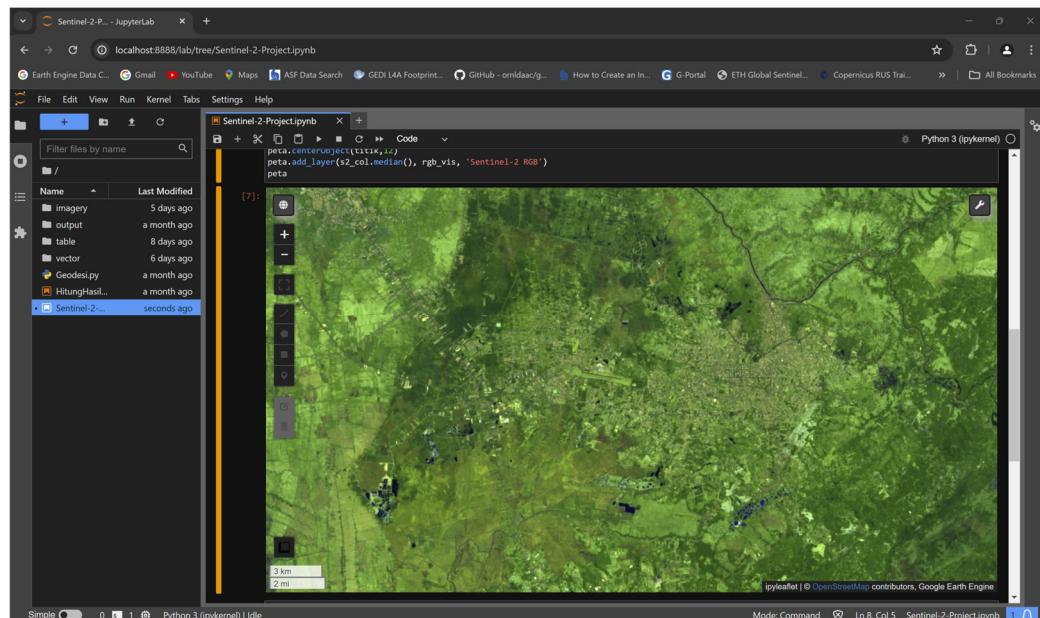
# Menentukan titik lokasi citra yang akan ditampilkan
titik = ee.Geometry.Point([114.776874, -3.449037])

# Mengakses Sentinel-2 image collection
s2_col = (
    ee.ImageCollection('COPERNICUS/S2_HARMONIZED')
        .filterDate(tanggal_awal, tanggal_akhir)
        .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20))
)

# Visualisasi RGB Sentinel-2 MSI
rgb_vis = {'min': 0, 'max': 5000, 'bands': ['B12','B11','B4']}

peta = geemap.Map()
peta.centerobject(titik,12)
peta.add_layer(s2_col.median(), rgb_vis, 'Sentinel-2 RGB')
peta
```

Output:



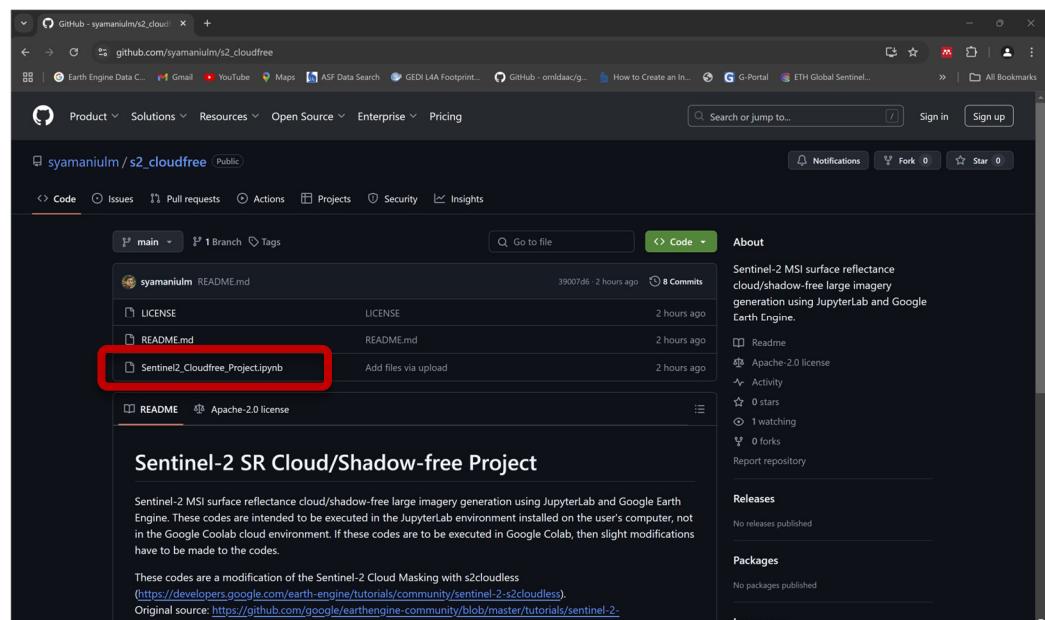
Tentu saja, outputnya relatif akan sama persis dengan Google Colab. Sebagaimana terlihat pada gambar di atas.

## Mengekspor Data ke Local Drive

Kelebihan lain GEE menggunakan JupyterLab/VS Code adalah kita dapat mengekspor langsung data seperti citra satelit secara langsung ke dalam *local drive* komputer kita. Sehingga nantinya dapat diproses secara luring menggunakan perangkat lunak yang sudah terinstal di komputer kita, misalnya QGIS (<https://www.qgis.org/>). Lebih jauh, kita juga dapat mengunduh citra yang sangat besar ke dalam local drive kita tanpa dibatasi oleh aturan `maxPixels=1e13`, misalnya mengunduh Citra Sentinel-2 untuk satu wilayah Provinsi Kalimantan Selatan. Hal ini dapat kita lakukan menggunakan fasilitas `geemap.download_ee_image_tiles`. Sebenarnya, fasilitas `geemap.download_ee_image_tiles` juga dapat mengekspor citra ke Google Drive. Akan tetapi, jika Google Drive kita versi gratisan yang memiliki kapasitas maksimum 15 GB, tentu saja tidak mungkin menyimpan citra resolusi spasial tinggi seperti Sentinel-2 untuk satu wilayah Kalimantan Selatan. Kita memerlukan Google Drive berbayar dengan kapasitas besar, atau opsi gratisnya adalah mengekspor citra secara langsung ke local drive komputer kita. Dan kalau kita mengekspor citra secara langsung ke komputer kita, tentu saja yang harus menjadi perhatian adalah seberapa besar ruang kosong yang masih tersisa di media penyimpanan kita.

### Akses dan Unduh Citra Sentinel-2 Bebas Awan

Permasalahan yang umumnya akan kita temukan ketika mengunduh citra satelit pada wilayah yang sangat luas adalah semakin besarnya potensi kehadiran awan, berikut bayangannya. Terlebih lagi pada wilayah yang dekat dengan garis ekuator seperti Pulau Kalimantan. Sehingga terkadang citra dengan wilayah yang luas menjadi tidak efektif, sebab hanya sedikit informasi permukaan bumi yang dapat terlihat akibat banyaknya tutupan awan. Untuk mengatasi problematika ini kita perlu untuk melakukan koreksi awan dan bayangan awan. Koreksi awan yang relatif mudah untuk diterapkan secara otomatis adalah dengan menggunakan mosaik citra multitemporal. Atau dengan kata lain, kita akan mencari pixel-pixel yang paling bersih dari awan pada rentang waktu tertentu untuk dipilih dan dimosaik menjadi sebuah citra. Untuk keperluan seperti ini, khususnya untuk Citra Sentinel-2 MSI, silahkan Anda mengakses laman [https://github.com/syamaniulm/s2\\_cloudfree](https://github.com/syamaniulm/s2_cloudfree), sebagaimana terlihat pada gambar di bawah.



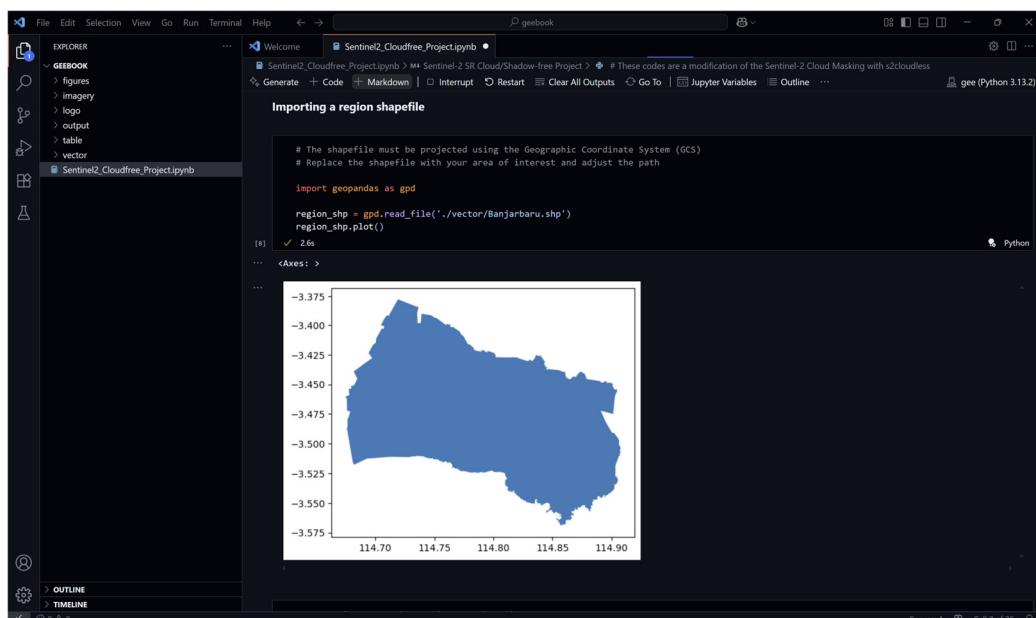
### Bab III Google Earth Engine

Pada laman di atas, download file **Sentinel2\_Cloudfree\_Project.ipynb**, kemudian simpan ke dalam folder kerja Anda, misalnya **D:\geebok**. Untuk menjalankan kode-kode program yang terdapat pada file **Sentinel2\_Cloudfree\_Project.ipynb** diperlukan instalasi sejumlah paket, yaitu:

```
earthengine-api (https://anaconda.org/conda-forge/earthengine-api)
geemap (https://anaconda.org/conda-forge/geemap)
geopandas (https://anaconda.org/conda-forge/geopandas)
json (https://anaconda.org/jmcurray/json)
folium (https://anaconda.org/conda-forge/folium)
geedim (https://anaconda.org/conda-forge/geedim)
```

Jika semua paket yang diperlukan di atas sudah terinstal, selanjutnya silahkan buka file **Sentinel2\_Cloudfree\_Project.ipynb** dengan menggunakan JupyterLab atau VS Code. Dimana di dalam buku ini yang digunakan adalah VS Code, sebagaimana terlihat pada gambar di bawah. Sebelum kode-kode Python dieksekusi, ada beberapa bagian kode yang harus diatur atau disesuaikan terlebih dahulu. Bagian pertama yang harus disesuaikan adalah inisialisasi proyek ee.Initialize(project='ee-geospatialulm'), tentu saja harus disesuaikan nama cloud project-nya dengan milik Anda. Bagian yang kedua adalah kode untuk membuka shapefile region\_shp = gpd.read\_file('./vector/Banjarbaru.shp'), kode ini berasumsi shapefile yang dibuka adalah **Banjarbaru.shp** di dalam folder **D:\geebok\vector**, sebagaimana terlihat pada gambar di bawah.

Anda bebas untuk mengganti shapefile yang diinginkan sesuai wilayah kerja Anda. Dalam hal ini, shapefile diwajibkan memiliki proyeksi atau sistem koordinat geografis (bujur-lintang) dan tanpa atribut Z atau M. Jika Anda menggunakan shapefile yang memiliki sistem koordinat UTM atau yang lainnya, maka wajib untuk dikonversi terlebih dahulu ke dalam sistem koordinat geografis. Terkait teknik konversi sistem koordinat shapefile menggunakan Geopandas, silahkan buka laman [https://geopandas.org/en/stable/docs/reference/api/geopandas.GeoDataFrame.to\\_crs.html](https://geopandas.org/en/stable/docs/reference/api/geopandas.GeoDataFrame.to_crs.html).



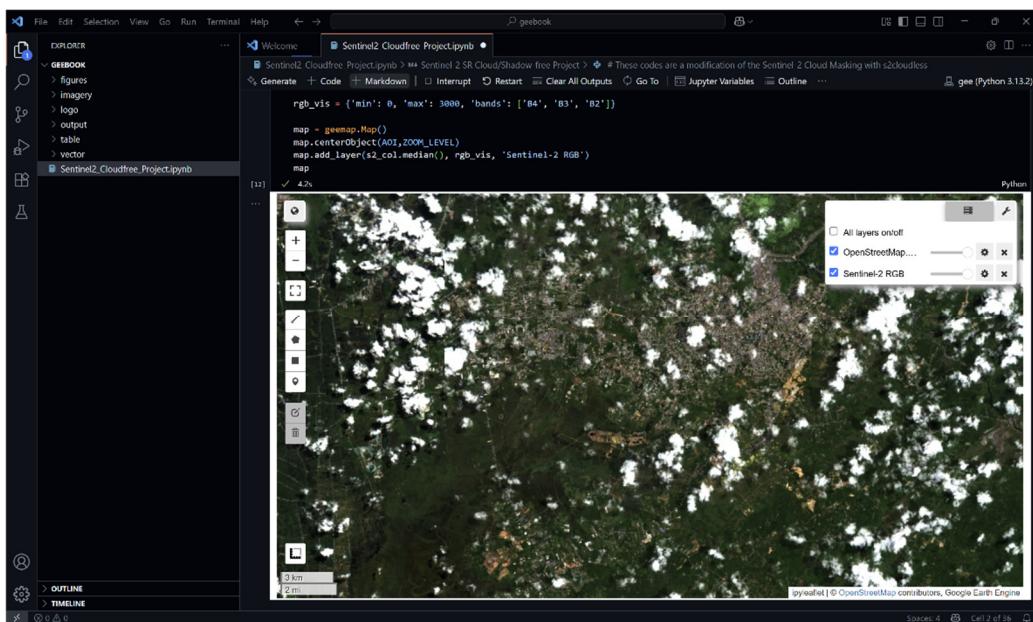
The screenshot shows a VS Code interface with a Jupyter Notebook tab open. The notebook cell contains Python code to import geopandas and read a shapefile named 'Banjarbaru.shp'. A plot of the resulting polygon is displayed in the output area. The plot shows a blue-shaded polygon representing the administrative boundary of Banjarbaru, with coordinates ranging from approximately 114.70 to 114.90 on the x-axis and -3.375 to -3.575 on the y-axis.

```
# The shapefile must be projected using the Geographic Coordinate System (GCS)
# Replace the shapefile with your area of interest and adjust the path

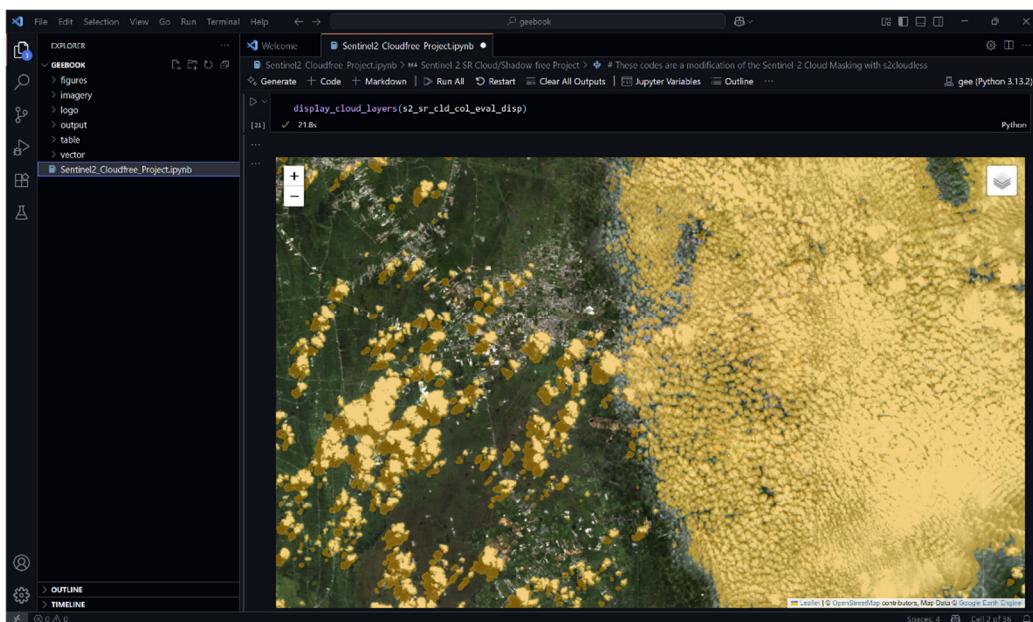
import geopandas as gpd

region_shp = gpd.read_file('./vector/Banjarbaru.shp')
region_shp.plot()
```

Gambar jendela VS Code di atas memperlihatkan shapefile poligon administrasi Kota Banjarbaru. Yang nantinya akan digunakan untuk memotong Citra Sentinel-2 yang akan diunduh. Selanjutnya, silahkan Anda menggulir (scroll) ke sel-sel kode di bagian bawah.

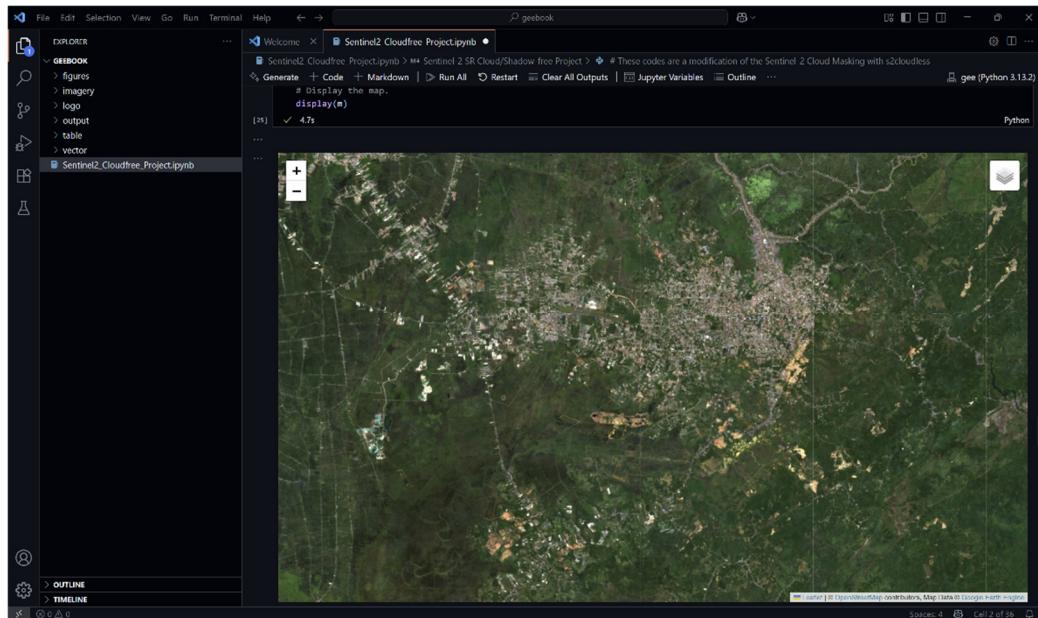


Gambar di atas memperlihatkan mosaik Citra Sentinel-2 asli yang akan kita unduh sebelum dibersihkan dari awan dan bayangan awan.

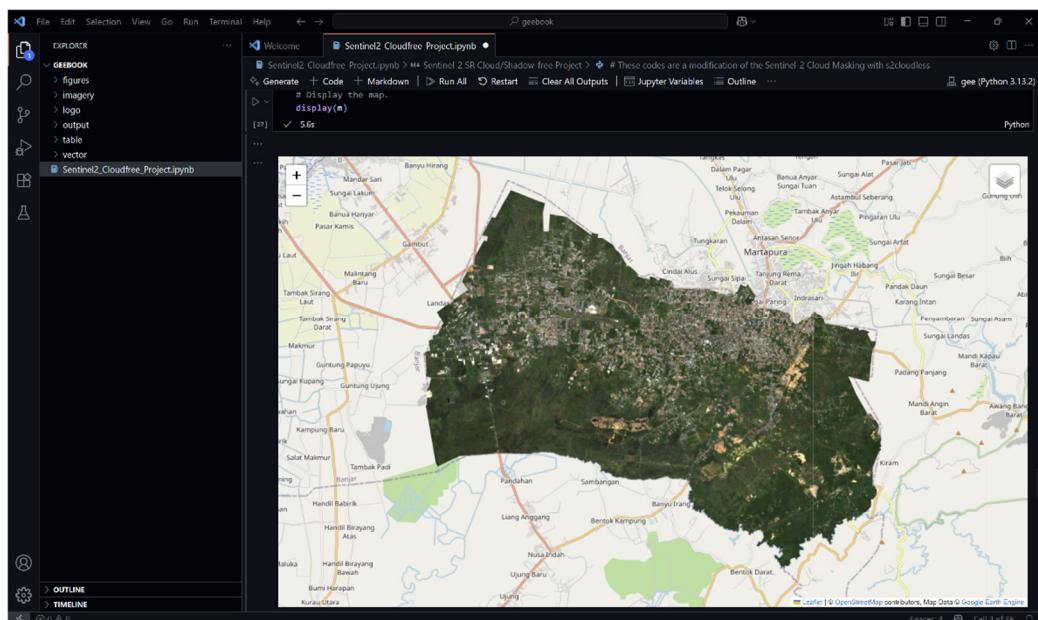


Gambar di atas memperlihatkan proses masking atau penandaan awan-awan yang nantinya akan dibuang secara otomatis.

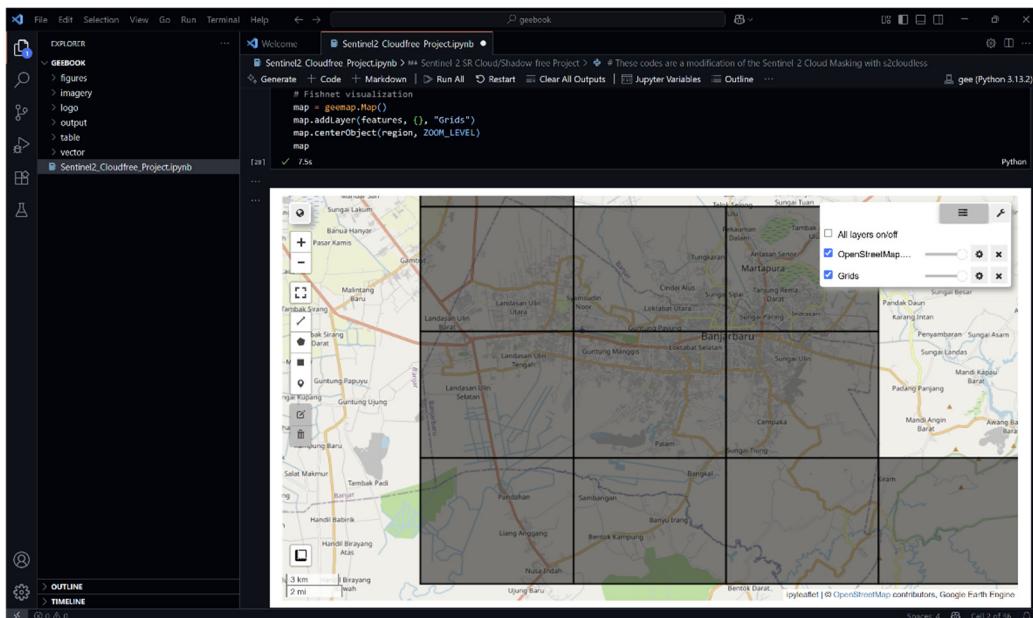
### Bab III Google Earth Engine



Gambar pada halaman sebelumnya memperlihatkan mosaik Citra Sentinel-2 yang sudah dibersihkan dari awan dan bayangan awan.



Gambar di atas memperlihatkan Citra Sentinel-2 bersih yang sudah dipotong menggunakan shapefile wilayah kerja kita.



Gambar di atas memperlihatkan fitur fishnet. Fishnet nantinya akan digunakan untuk memotong-motong sebuah citra yang besar menjadi *image tile* atau kepingan-kepingan citra yang kecil, sehingga nantinya dapat diunduh dengan mudah menggunakan GEE. Hal yang perlu diperhatikan dalam pembuatan fishnet adalah, semakin besar wilayah citra, semakin tinggi resolusi spasial citra, dan semakin banyak jumlah saluran yang akan diunduh, maka pengaturan di bagian `row_num = 3` dan `col_num = 3` harus diperbesar masing-masing nilainya. Misalnya, untuk shapefile seluas wilayah Kalimantan Selatan, Citra Sentinel-2 dengan resolusi spasial 10 meter dan 10 saluran, dapat diatur `row_num = 12` dan `col_num = 10`. Yang jelas, semakin besar nilai `row_num` dan `col_num` maka akan semakin banyak image tile yang akan diunduh.

Untuk sel kode terakhir pada file **Sentinel2\_Cloudfree\_Project.ipynb**, sebelum sel ini dieksekusi, variabel `out_dir` juga harus Anda modifikasi. Jika Anda ingin menyimpan citra-citra ke dalam folder **D:\geebok\output**, maka `out_dir = './output'`, sebagaimana terlihat seperti pada kode dan gambar di bawah. Lokasi penyimpanan citra dapat juga Anda tulis dengan path lengkap (*absolute path*) `out_dir = 'D:/geebok/output'`.

```
# Set the output directory, adjust it to your needs
out_dir = './output'

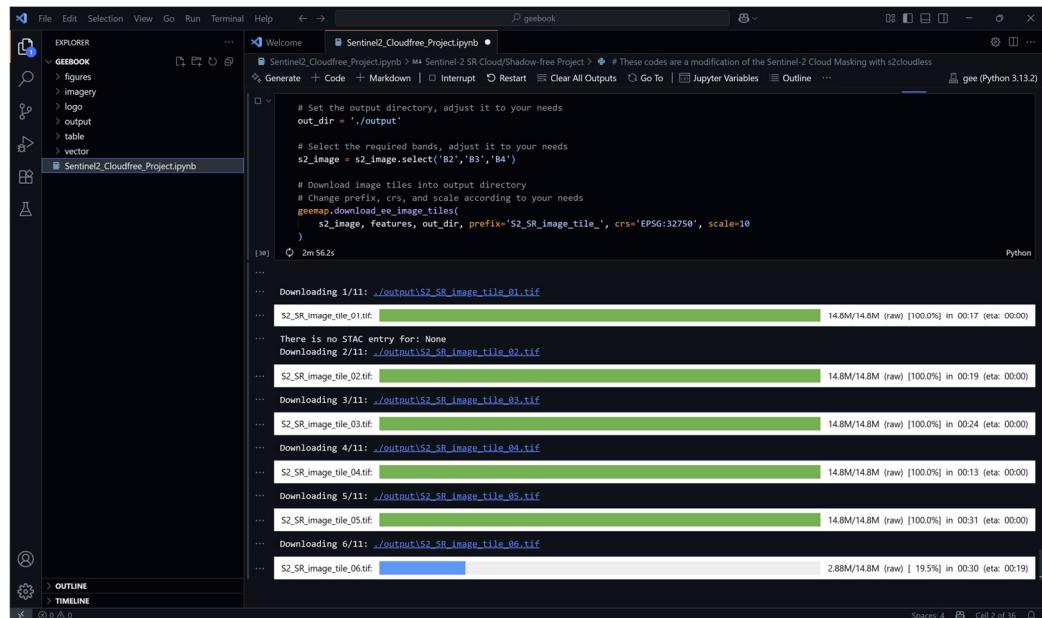
# Select the required bands, adjust it to your needs
s2_image = s2_image.select('B2','B3','B4')

# Download image tiles into output directory
# Change prefix, crs, and scale according to your needs
geemap.download_ee_image_tiles(
    s2_image, features, out_dir, prefix='S2_SR_image_tile_',
    crs='EPSG:32750', scale=10
)
```

Saluran-saluran yang diunduh (`s2_image = s2_image.select('B2','B3','B4')`), inisial nama file image tile (`prefix='S2_SR_image_tile_'`), sistem koordinat (`crs='EPSG:32750'`), dan resolusi spasial (`scale=10`), jika diperlukan juga dapat Anda sesuaikan dengan keperluan dan wilayah kerja Anda.

### Bab III Google Earth Engine

Perhatikan proses pengunduhan Citra Sentinel-2 ke dalam image tile sebagaimana gambar di bawah. Tentu saja, kita harus menunggu sampai semua tile selesai untuk diunduh.



The screenshot shows a Jupyter Notebook interface within VS Code. The code cell contains Python code for downloading Sentinel-2 image tiles. The output pane displays the progress of the download for 11 tiles, each labeled S2\_SR\_image\_tile\_{number}. The tiles are being downloaded from a directory named 'output'.

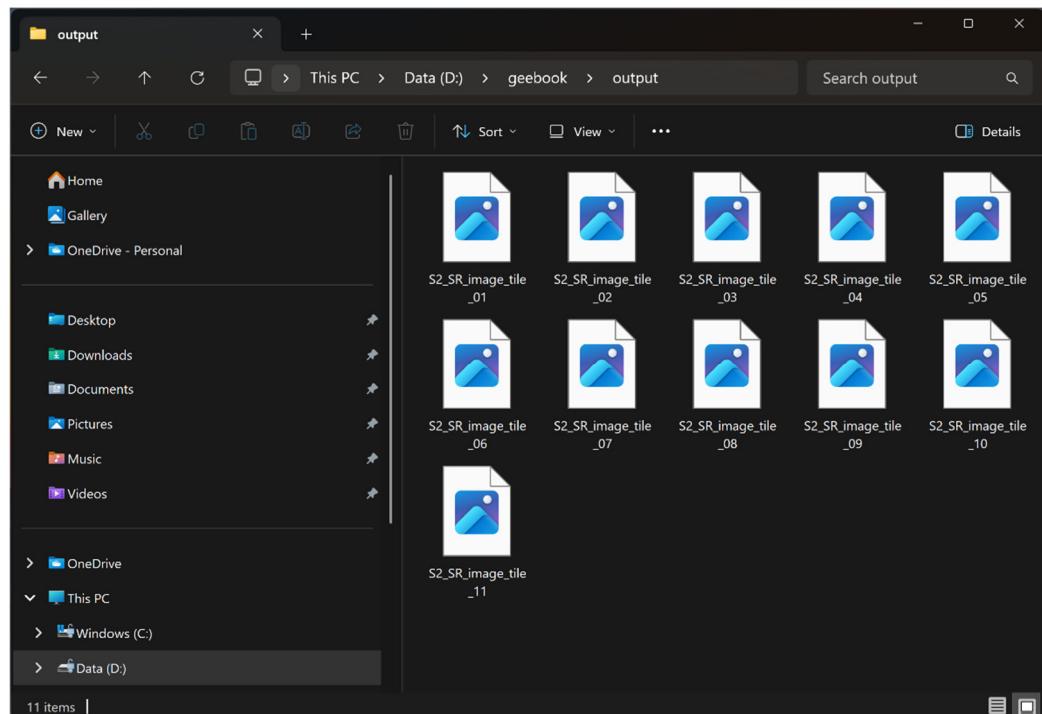
```
# Set the output directory, adjust it to your needs
out_dir = './output'

# Select the required bands, adjust it to your needs
s2_image = s2_image.select('B2','B3','B4')

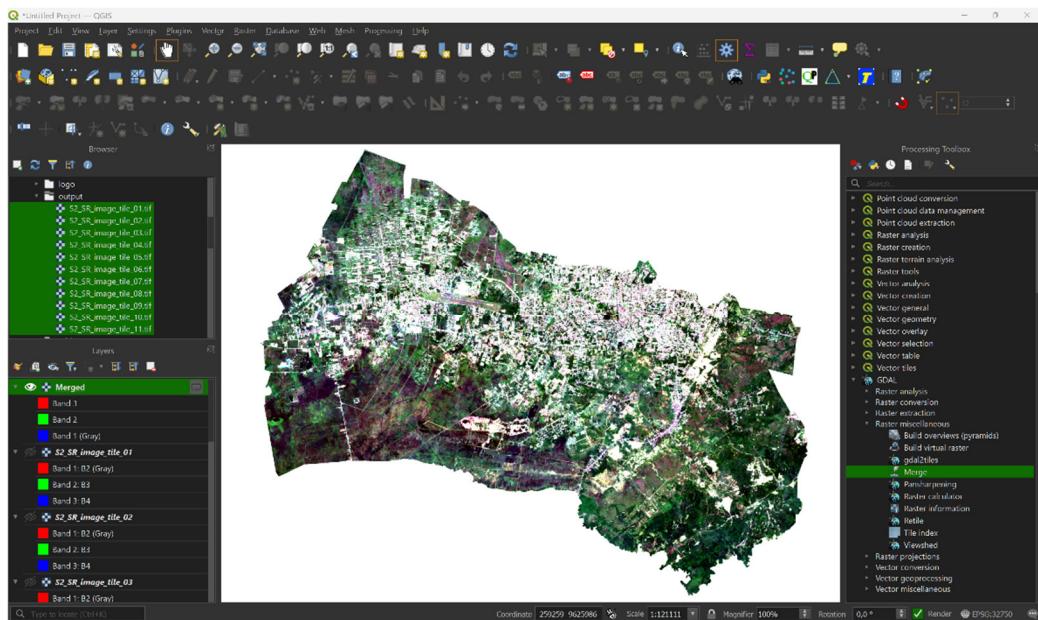
# Download image tiles into output directory
# Change prefix, crs, and scale according to your needs
gemap.download_ee_image_tiles(
    s2_image, features, out_dir, prefix='S2_SR_image_tile_', crs='EPSG:32750', scale=10
)
```

...  
...  
... Downloading 1/11: ./output/S2\_SR\_image\_tile\_01.tif  
... S2\_SR\_image\_tile\_01.tif: 14.8M/14.8M (raw) [100.0%] in 00:17 (eta: 00:00)  
... There is no STAC entry for: None  
... Downloading 2/11: ./output/S2\_SR\_image\_tile\_02.tif  
... S2\_SR\_image\_tile\_02.tif: 14.8M/14.8M (raw) [100.0%] in 00:19 (eta: 00:00)  
... Downloading 3/11: ./output/S2\_SR\_image\_tile\_03.tif  
... S2\_SR\_image\_tile\_03.tif: 14.8M/14.8M (raw) [100.0%] in 00:24 (eta: 00:00)  
... Downloading 4/11: ./output/S2\_SR\_image\_tile\_04.tif  
... S2\_SR\_image\_tile\_04.tif: 14.8M/14.8M (raw) [100.0%] in 00:13 (eta: 00:00)  
... Downloading 5/11: ./output/S2\_SR\_image\_tile\_05.tif  
... S2\_SR\_image\_tile\_05.tif: 14.8M/14.8M (raw) [100.0%] in 00:31 (eta: 00:00)  
... Downloading 6/11: ./output/S2\_SR\_image\_tile\_06.tif  
... S2\_SR\_image\_tile\_06.tif: 2.88M/14.8M (raw) [ 19.5%] in 00:30 (eta: 00:19)

Berikut adalah image tile Sentinel-2 yang sudah selesai diunduh:



Selanjutnya, Anda dapat menggunakan perangkat lunak favorit Anda untuk membuka citra secara offline, misalnya QGIS, sebagaimana terlihat pada gambar di bawah:



Langkah terakhir, untuk dapat dianalisis lebih jauh, seluruh image tile harus dimosaik terlebih dahulu menggunakan perangkat lunak favorit Anda. Sebaimana contoh di atas, citra sudah dimosaik menggunakan tool **GDAL** → **Raster miscellaneous** → **Merge** pada toolbox QGIS.

### Akses dan Unduh Citra Landsat-9 Bebas Awan

Landsat-9 memiliki sensor yang identik dengan Landsat-8, sehingga meskipun di dalam buku ini yang dicontohkan adalah Landsat-9, untuk mengakses dan mengunduh Landsat-8 bebas awan caranya sama dengan Landsat 9. Hanya ada sedikit penyesuaian pada akses image collection-nya. Untuk menghasilkan mosaik Landsat-8 atau Landsat-9 bebas awan dan bayangan awan, tekniknya jauh lebih simpel dibandingkan dengan Sentinel-2 sebelumnya.

Sebagai contoh, dengan menggunakan VS Code, buat sebuah file notebook baru dengan nama **Landsat9\_Download.ipynb**. Tentu saja, Anda dapat menggunakan JupyterLab atau Google Colab.

Langkah berikutnya, ketikkan dan jalankan kode-kode Python berikut:

```
import ee, geemap, geopandas as gpd

ee.Authenticate()

ee.Initialize(project='ee-geospatialulm')

region_shp = gpd.read_file('./vector/Kalsel_Polygon.shp')
region_shp.plot()
```

Kode-kode di atas digunakan untuk mengimpor paket-paket yang diperlukan, otorisasi, dan membuka sebuah shapefile dengan nama **Kalsel\_Polygon.shp**. Anda dapat mengganti shapefile sesuai dengan wilayah kerja Anda. Perhatikan hasilnya sebagaimana terlihat pada gambar berikut:

### Bab III Google Earth Engine

The screenshot shows a Jupyter Notebook interface with a Python kernel. The code cell contains:

```
region_shp = gpd.read_file('../vector/Kalsel_Polygon.shp')
region_shp.plot()
```

A plot window displays a blue polygon representing the region\_shp on a coordinate system with x-axis from 114.5 to 117.5 and y-axis from -5.0 to -1.5.

The next code cell contains:

```
region = geemap.geopandas_to_ee(region_shp).geometry()
```

At the bottom right, status indicators show In 2, Col 1, Spaces: 4, Cell 12 of 12.

Selanjutnya, ketikkan dan jalankan sebaris kode berikut:

```
region = geemap.geopandas_to_ee(region_shp).geometry()
```

Kode di atas digunakan untuk mengkonversi shapefile menjadi GEE geometry. Kode ini merupakan versi singkat dari kode sebelumnya pada halaman 174, dimana pada contoh kode sebelumnya kita harus mengkonversi shapefile ke format JSON dan Feature Collection terlebih dahulu. Tentu saja kedua versi kode ini fungsinya sama persis dan Anda cukup memilih salah satu.

Berikutnya, ketikkan dan jalankan kode-kode berikut:

```
# Applying scaling factors.
def apply_scale_factors(image):
    optical_bands = image.select('SR_B.*').multiply(0.0000275).add(-0.2)
    thermal_bands = image.select('ST_B.*').multiply(0.00341802).add(149.0)

    image = (
        image.addBands(optical_bands, None, True)
        .addBands(thermal_bands, None, True)
    )

    return image
```

Kode di atas adalah sebuah fungsi untuk mengkonversi nilai-nilai pixel saluran-saluran optik Landsat-9 yang aslinya memiliki rentang dari 1 hingga 65455, menjadi nilai spectral absolut 0 hingga 1. Termasuk juga untuk mengkonversi saluran termal (Band 10) Landsat-9 yang memiliki rentang nilai-nilai pixel dari 0 hingga 65535, menjadi *brightness temperature* (temperatur emisi) dalam satuan Kelvin (K). Penjelasan tentang hal ini dapat dilihat pada laman [https://developers.google.com/earth-engine/datasets/catalog/LANDSAT\\_LC09\\_C02\\_T1\\_L2](https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LC09_C02_T1_L2).

Anda juga dapat langsung copy-paste fungsi `apply_scale_factors` dari laman ini.

Langkah selanjutnya, ketikkan dan jalankan kode-kode berikut:

```
# Masking clouds and shadows
def mask_clouds(image):
    qa = image.select('QA_PIXEL')

    dilated_bit_mask = 1 << 1
    cirrus_bit_mask = 1 << 2
    cloud_bit_mask = 1 << 3
    shadow_bit_mask = 1 << 4

    mask = (
        qa.bitwiseAnd(dilated_bit_mask).eq(0)
        .And(qa.bitwiseAnd(cirrus_bit_mask).eq(0))
        .And(qa.bitwiseAnd(cloud_bit_mask).eq(0))
        .And(qa.bitwiseAnd(shadow_bit_mask).eq(0))
    )

    return image.updateMask(mask)
```

Kode-kode di atas digunakan untuk masking atau membuang (melubangi pada citra) awan dan bayangan awan pada Citra Landsat-9 dengan menggunakan saluran QA\_PIXEL. Penjelasan lengkap tentang hal ini dapat dilihat pada tautan [https://developers.google.com/earth-engine/datasets/catalog/LANDSAT\\_LC09\\_C02\\_T1\\_L2#bands](https://developers.google.com/earth-engine/datasets/catalog/LANDSAT_LC09_C02_T1_L2#bands). Dimana bit 1 merupakan peluasan (buffer) tepi awan, bit ini digunakan untuk meningkatkan akurasi masking awan. Bit 2 merupakan penanda awan cirrus, bit 3 merupakan penanda awan, dan bit 4 merupakan penanda bayangan awan. Lanjutkan dengan mengetikkan dan menjalankan kode-kode berikut:

```
# Load Landsat 9 collection
l9_col = (
    ee.ImageCollection('LANDSAT/LC09/C02/T1_L2')
    .filterDate('2022-01-01', '2025-01-01')
    .filter(ee.Filter.lt('CLOUD_COVER', 50))
    .map(apply_scale_factors)
    .map(mask_clouds)
)
```

Kode-kode di atas digunakan untuk mengakses image collection Landsat-9. Jika Anda ingin mengakses Landsat-8, cukup ganti **LC09** menjadi **LC08**. Kode-kode di atas juga sekaligus untuk memfilter tanggal akuisisi citra, dan memfilter persentase tutupan awan Landsat-9 agar kurang dari 50%. Fungsi **map** pada kode-kode di atas diterapkan untuk memanggil fungsi **apply\_scale\_factors** untuk mengkonversi nilai-nilai pixel, dan fungsi **mask\_clouds** untuk masking awan dan bayangan awan. Selanjutnya, ketikkan dan jalankan kode-kode berikut:

```
# Calculate median image for the region
l9_image = l9_col.median().clip(region)
```

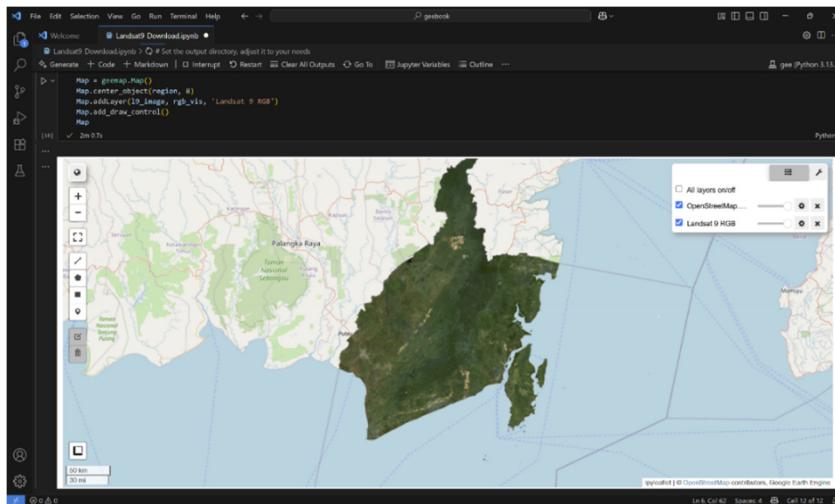
Kode di atas digunakan untuk mengambil nilai tengah (*median*) pixel sekaligus memotong citra.

```
# Visualizing the RGB composite
rgb_vis = {
    'bands': ['SR_B4', 'SR_B3', 'SR_B2'],
    'min': 0,
    'max': 0.25,
    'gamma': 1.4
}

Map = geemap.Map()
Map.center_object(region, 8)
Map.addLayer(l9_image, rgb_vis, 'Landsat 9 RGB')
Map.add_draw_control()
Map
```

### Bab III Google Earth Engine

Kode di atas digunakan untuk visualisasi Citra Landsat-9 komposit 432 (*true color composite*) yang sudah bersih dari awan dan bayangan awan. Perhatikan hasilnya sebagaimana gambar berikut:



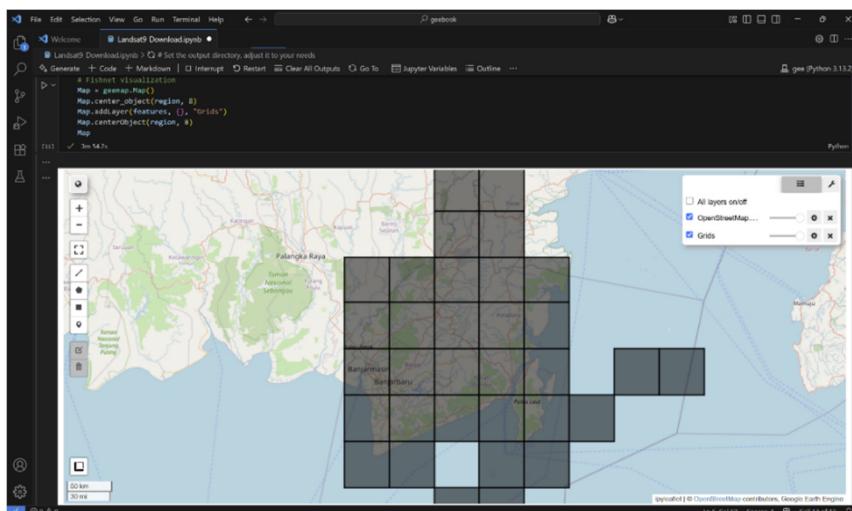
Selanjutnya, ketikkan dan jalankan kode-kode untuk membuat dan menampilkan fishnet berikut:

```
# Setting the number of rows and the number of columns
row_num = 8
col_num = 7

# Creating fishnet features
features = geemap.fishnet(region, rows=row_num, cols=col_num)

# Fishnet visualization
Map = geemap.Map()
Map.center_object(region, 8)
Map.addLayer(features, {}, "Grids")
Map.centerObject(region, 8)
Map
```

Untuk citra Landsat-9 yang memiliki resolusi spasial 30 meter dan 7 saluran, dengan shapefile seluas wilayah Kalimantan Selatan, maka `row_num = 8` dan `col_num = 7` sudah cukup.



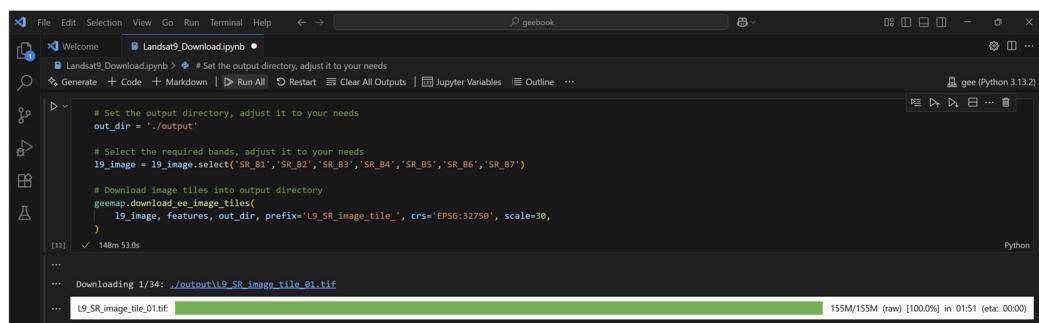
Berikut adalah kode-kode terakhir yang harus Anda ketikkan dan jalankan untuk mengunduh Citra Landsat-9 bebas awan:

```
# Set the output directory
out_dir = './output'

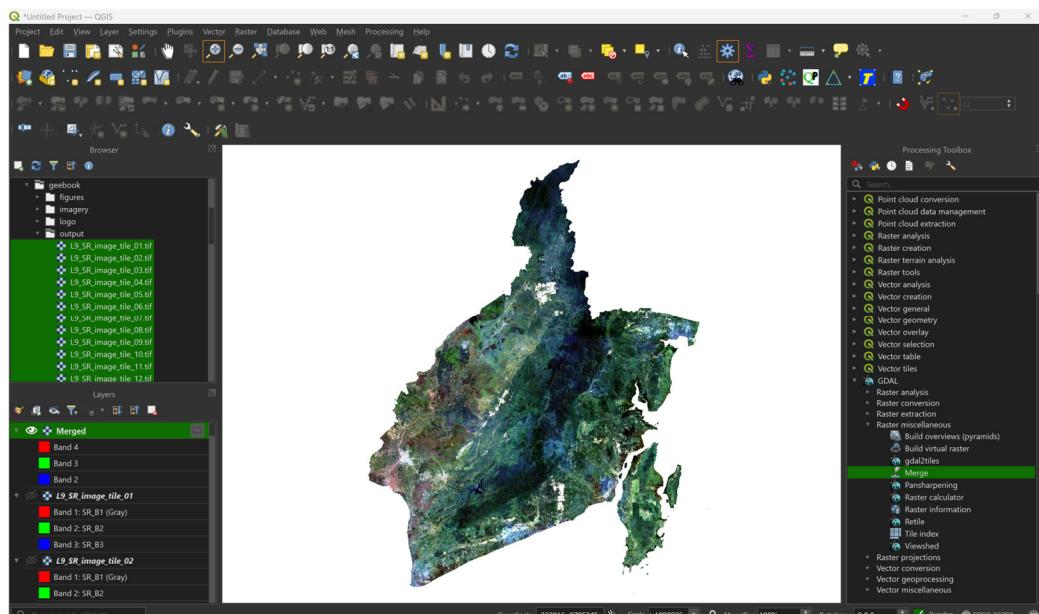
# Select the required bands
l9_image =
l9_image.select('SR_B1','SR_B2','SR_B3','SR_B4','SR_B5','SR_B6','SR_B7')

# Download image tiles into output directory
geemap.download_ee_image_tiles(
    l9_image, features, out_dir, prefix='L9_SR_image_tile_', crs='EPSG:32750',
    scale=30,
)
```

Pada kode-kode di atas, Anda dapat menyesuaikan lokasi output citra, saluran-saluran yang akan diunduh, inisial nama image tile, sistem koordinat, dan resolusi spasial, sesuai keperluan Anda.



Perlu waktu hampir 150 menit (tergantung kecepatan akses internet) untuk mengunduh Citra Landsat-9 band 1 sampai band 7, resolusi spasial 30 meter, dan seluas Kalimantan Selatan (total 34 tile). Perhatikan hasil citranya yang sudah dimosaik menggunakan QGIS seperti pada gambar:



## B. Transformasi Citra

Sesuai dengan namanya, transformasi citra (*image transformation*) berarti merubah bentuk citra penginderaan jauh. Dalam konteks ini, yang dirubah adalah dimensi spektral, dimensi spasial, atau dimensi temporalnya. Transformasi citra pada umumnya melibatkan proses-proses kalkulasi matematika dan statistika. Dimulai dari teknik yang paling sederhana, seperti *band rationing* atau penisbahan saluran pada transformasi spektral. Hingga yang sangat kompleks yaitu melibatkan algoritma-algoritma kecerdasan buatan. Transformasi citra pada umumnya dilakukan untuk tujuan mengekstrak informasi yang spesifik di atas citra penginderaan jauh, seperti menajamkan fitur-fitur vegetasi, air, urban, dan sebagainya. Selain itu, transformasi citra juga sering dilakukan di dalam proses koreksi atau restorasi citra, atau peningkatan kualitas citra seperti peningkatan resolusi spasial (*image super resolution*).

### Transformasi Spektral dan Indeks-indeks Multispektral

Transformasi spektral mungkin adalah jenis transformasi citra yang paling umum diterapkan. Secara sederhana, transformasi spektral berarti merubah nilai spektral. Sehingga nilai spektral citra (*reflectance*) yang pada umumnya memiliki rentang nilai dari 0 sampai 1, setelah ditransformasi rentang nilainya berubah menjadi -1 sampai 1 misalnya, atau rentang nilai lainnya. Indeks-indeks multispektral seperti indeks-indeks vegetasi, indeks-indeks air, indeks-indeks urban, dan sebagainya, merupakan metode-metode transformasi spektral.

Anda dapat mengunjungi laman <https://www.indexdatabase.de/db/i.php> untuk mencari indeks-indeks multispektral. Tersedia lebih dari 500 indeks-indeks spektral di laman ini. Bahkan termasuk indeks-indeks hiperspektral, yaitu metode-metode transformasi untuk citra hiperspektral. Citra hiperspektral sendiri pada dasarnya adalah citra multispektral, akan tetapi jumlah band-nya sangat banyak, mungkin ratusan atau ribuan band.

Indeks multispektral paling populer adalah *Normalized Difference Vegetation Index* (NDVI). Indeks vegetasi legendaris yang diformulasikan oleh para ilmuwan dari Texas A&M University (<https://www.tamu.edu>) ini usianya sudah lebih dari setengah abad, dan merupakan indeks vegetasi yang paling luas pemanfaatannya di dunia penginderaan jauh. NDVI pada khususnya dimanfaatkan untuk ekstraksi parameter-parameter biofisik vegetasi, seperti biomassa, volume kayu, kerapatan kanopi, hingga keanekaragaman hayati, dan sebagainya. NDVI diformulasikan sebagai berikut (Rouse et al., 1973):

$$\text{NDVI} = \frac{\text{NIR} - \text{Red}}{\text{NIR} + \text{Red}}$$

Dimana *NIR* adalah *Near Infrared* atau band inframerah dekat (IMD), dan *Red* adalah band merah. Tentu saja, posisi NIR dan Red ada di band berapa akan tergantung pada jenis citranya. Pada Landsat 7 NIR ada di band 4 dan Red ada di band 3, pada Landsat 8/9 NIR ada di band 5 dan Red ada di band 4, pada Sentinel-2 NIR ada di band 8 dan Red ada di band 4. Pada perkembangan-perkembangan selanjutnya, NDVI banyak mengalami modifikasi. Bahkan banyak indeks-indeks spektral lainnya yang dikembangkan karena terinspirasi NDVI, baik itu indeks-indeks vegetasi maupun indeks-indeks non-vegetasi.

Sebagai contoh implementasi metode transformasi NDVI pada GEE, buat sebuah notebook baru di dalam Google Colab. Kemudian ketikkan kode-kode berikut:

```

import ee, geemap

ee.Authenticate()

ee.Initialize(project='ee-geospatialulm')

# Mengakses Sentinel-2 image collection
s2_col = (
    ee.ImageCollection("COPERNICUS/S2_SR_HARMONIZED")
    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 5))
)

# Mencari informasi Image ID Sentinel-2 MSI TOC
info_citra = (
    ee.Image(s2_col.filterDate('2023-01-01', '2023-12-31')
            .filterMetadata('MGRS_TILE', 'equals', '50MKB'))
)
info_citra

```

Output:

```

▼ ImageCollection COPERNICUS/S2_SR_HARMONIZED (2 elements)
  type: ImageCollection
  id: COPERNICUS/S2_SR_HARMONIZED
  version: 1718105170518083
  ▶ bands: List (23 elements)
  ▶ properties: Object (21 properties)
  ▼ features: List (2 elements)
    ▼ 0: Image COPERNICUS/S2_SR_HARMONIZED/20230808T022539_20230808T023805_T50MKB (23 bands)
      type: Image
      id: COPERNICUS/S2_SR_HARMONIZED/20230808T022539_20230808T023805_T50MKB
      version: 1718105170518083
      ▶ bands: List (23 elements)
      ▶ properties: Object (101 properties)
    ▶ 1: Image COPERNICUS/S2_SR_HARMONIZED/20230927T022539_20230927T023803_T50MKB (23 bands)

```

Perhatikan dengan cermat, bahwa kode di atas digunakan untuk mencari Citra Sentinel-2 MSI dalam format TOC untuk tile spesifik, yaitu 50MKB. Lihat kembali penjelasan dan Gambar 1.5 pada Bab I halaman 14. Sehingga contoh kode ini sedikit berbeda dari contoh sebelumnya dimana kita mengakses GEE image collection. Sedangkan di dalam contoh kode ini kita langsung mengakses GEE image pada waktu dan tile yang spesifik.

Pada kode di atas, pada rentang waktu dari 1 Januari 2023 hingga 31 Desember 2023, hanya ditemukan 2 tile 50MKB yang memiliki persentase penutupan awan kurang dari atau sama dengan 5%. Dalam hal ini, tile 50MKB yang akan kita pilih adalah **20230808T022539\_20230808T023805\_T50MKB**. Entri 20230808T022539 menunjukkan tanggal dan jam akuisisi. Artinya, citra diakuisisi pada tanggal 2023-08-08, dan jam 02:25:39 GMT, atau ekivalen dengan jam 10:25:39 WITA (waktu di lokasi citra). Sementara entri T50MKB jelas menunjukkan tile citra.

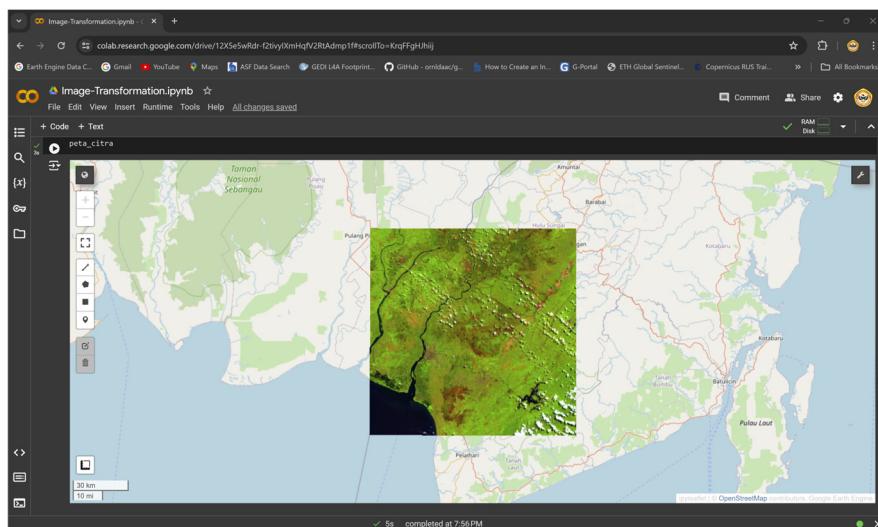
Copy COPERNICUS/S2\_SR\_HARMONIZED/20230808T022539\_20230808T023805\_T50MKB pada output di atas, untuk nanti kita paste ke dalam kode lanjutannya sebagaimana berikut:

### Bab III Google Earth Engine

```
# Mengakses Citra Sentinel-2 MSI TOC
s2_image = (
    ee.Image('COPERNICUS/S2_SR_HARMONIZED/20230808T022539_20230808T023805_T50MKB')
    .divide(10000)
)

# Visualisasi citra
rgb_vis = {'min': 0, 'max': 0.5, 'bands': ['B11', 'B8', 'B2']}
peta_citra = geemap.Map()
peta_citra.centerObject(s2_image)
peta_citra.add_layer(s2_image, rgb_vis, 'Sentinel-2 RGB')
peta_citra
```

Output:



Perhatikan bahwa pada kode di atas kita menjalankan fungsi `.divide(10000)` pada saat kita mengakses GEE image. Hal ini untuk mengembalikan nilai spektral Sentinel-2 MSI dari rentang 0 sampai 10000, menjadi nilai reflectance absolutnya pada rentang 0 sampai 1. Proses pembagian dengan 10000 ini sangat direkomendasikan sebelum kita melakukan transformasi citra, seperti kalkulasi NDVI. Terlebih jika indeks spektral nantinya dijadikan parameter pemodelan kuantitatif.

Setidaknya ada 3 teknik di dalam mengimplementasikan formula NDVI, atau pun indeks-indeks spektral lainnya di dalam GEE. Perhatikan kode-kode berikut:

```
# Kalkulasi NDVI: Teknik 1
red = s2_image.select('B4')
nir = s2_image.select('B8')

ndvi = (nir.subtract(red)).divide(nir.add(red)).rename('NDVI')
```

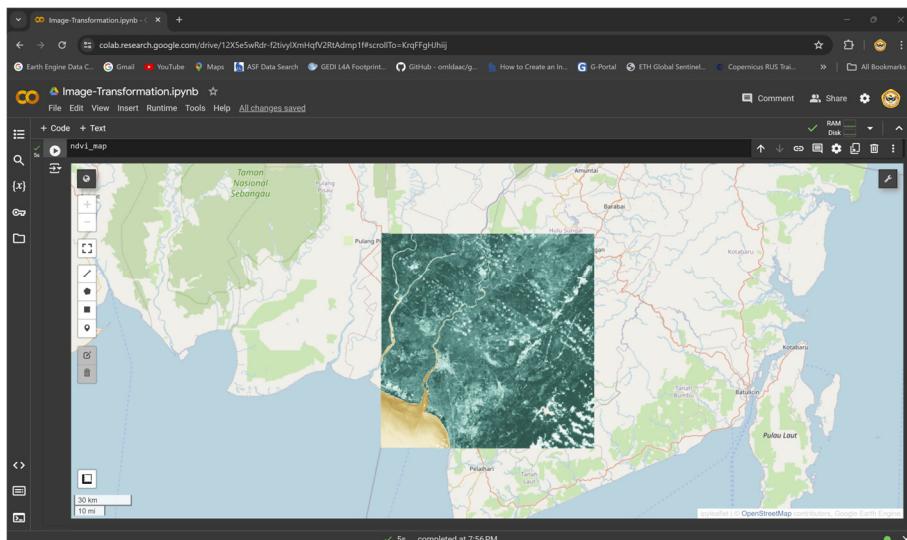
```
# Kalkulasi NDVI: Teknik 2
ndvi = s2_image.expression(
    '(IMD - Merah) / (IMD + Merah)',
    {'Merah': s2_image.select('B4'),
     'IMD': s2_image.select('B8')}).rename('NDVI')
```

```
# Kalkulasi NDVI: Teknik 3
ndvi = s2_image.normalizedDifference(['B8','B4']).rename('NDVI')
```

**PERINGATAN**, Teknik 1, 2, dan 3 pada contoh kode di atas akan memberikan output yang sama persis. Jadi cukup pilih dan jalankan salah satu saja dari 3 teknik ini, jangan dijalankan ketiganya. Selanjutnya ketik dan jalankan kode berikut:

```
# visualisasi NDVI
ndvi_vis = {'min': -1, 'max': 1, 'palette': 'BrBG'}
ndvi_map = geemap.Map()
ndvi_map.centerObject(ndvi)
ndvi_map.add_layer(ndvi, ndvi_vis, 'NDVI')
ndvi_map
```

Output:



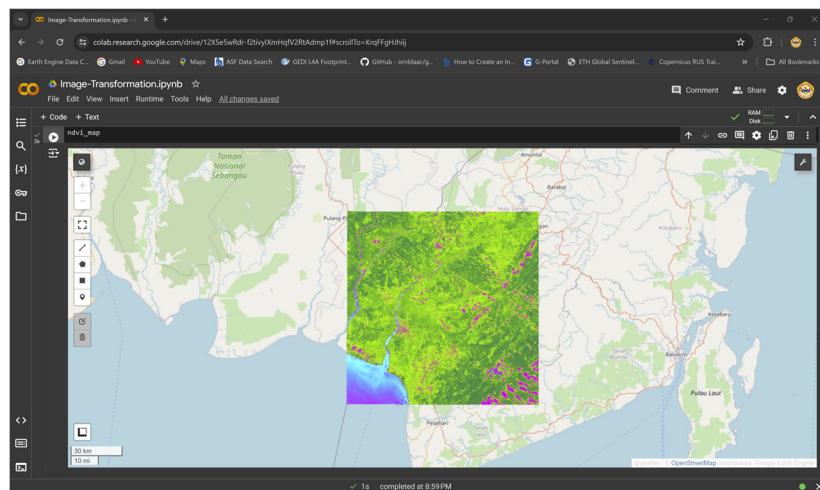
Teknik 1, 2, dan 3 pada kalkulasi NDVI di atas masing-masing memiliki kelebihan dan kekurangan. Teknik 1 menggunakan algebra atau operasi matematika standar GEE, dimana teknik ini pada umumnya kurang disukai bagi pemula, khususnya pemula di dunia pemrograman. Sebab teknik seperti ini sangat memerlukan ketelitian, terutama di dalam menentukan prioritas urutan operasinya. Teknik 2 menggunakan fungsi `expression`. Dimana ekspresinya ditulis dalam bentuk string dengan notasi variabel terserah programer (`IMD`, `Merah`). Kemudian masing-masing variabel didefinisikan menggunakan dictionary. Teknik 2 ini terlihat lebih ribet, karena kodennya cukup panjang. Akan tetapi sesungguhnya Teknik 2 ini lebih memberikan kejelasan di dalam penulisan formula, seperti halnya formula NDVI. Teknik 3 merupakan yang paling simpel, sebab menggunakan fungsi bawaan `normalizedDifference`. Keterbatasan Teknik 3 adalah hanya dapat diterapkan pada formula *NDVI-like*, yaitu indeks-indeks spektral yang formulanya dalam bentuk *normalize difference* seperti NDVI. Sementara indeks-indeks spektral lain yang formulanya lebih kompleks (atau lebih sederhana) dari NDVI, tentu saja tidak dapat diimplementasikan menggunakan fungsi `normalizedDifference`.

### Bab III Google Earth Engine

Instruksi `rename('NDVI')` sifatnya opsional, artinya boleh ditiadakan, akan tetapi sangat direkomendasikan untuk diterapkan. Instruksi ini akan memberi atau mengganti nama band yang kita transformasi sesuai yang diinginkan, misalnya `NDVI`. Pada analisis tertentu, terkadang kita perlu untuk secara langsung mengambil suatu citra hasil pemrosesan, seperti citra hasil transformasi NDVI, dengan cara mengakses namanya secara langsung. Perhatikan juga perbedaan antara teknik visualisasi citra komposit RGB dengan citra *single band* seperti NDVI. Pada visualisasi citra komposit RGB kita menggunakan teknik seperti `'bands': ['B11', 'B8', 'B2']` untuk memilih band-band yang dikompositkan. Sedangkan pada citra band tunggal seperti NDVI kita menggunakan teknik seperti `'palette': 'BrBG'`. Dimana kode warna `BrBG` diambil dari Matplotlib colormaps. Alternatif lain untuk pewarnaan citra band tunggal adalah dengan menggunakan kode warna heksadesimal. Sebagai contoh:

```
# Visualisasi NDVI
ndvi_vis = {'min': -1,
            'max': 1,
            'palette': ['0000FF', '00FFFF', 'FF00FF', '90FF00', '006060']}
ndvi_map = geemap.Map()
ndvi_map.centerobject(ndvi)
ndvi_map.add_layer(ndvi, ndvi_vis, 'NDVI')
ndvi_map
```

Output:



Sebagai tambahan, Anda dapat mengunjungi laman <https://colorhunt.co/> untuk mencari *color palette* yang cantik sesuai keinginan Anda.

### Thresholding dan Masking

*Thresholding* atau pembatasan ambang/batas nilai merupakan hal yang sering dilakukan pada citra-citra produk transformasi spektral. Thresholding biasanya digunakan untuk memisahkan objek atau fitur di atas citra, dengan mengacu kepada literatur hasil riset tertentu. Sebagai contoh, nilai *Modified Normalized Difference Water Index* (MNDWI) lebih dari 0 (nol) menurut Xu (2006) dipastikan merupakan fitur air. Nilai threshold juga dapat dikalkulasi berdasarkan algoritma tertentu, seperti *Otsu thresholding* (Otsu, 1979).

Pada contoh kasus berikut, kita akan memantau tingkat kekeruhan sekaligus memetakan *Total Suspended Solid* (TSS) atau muatan suspensi air muara Sungai Barito. Metode yang digunakan adalah kombinasi MNDWI, *Normalized Difference Turbidity Index* (NDTI), dan formula TSS menurut Liu et al. (2017). Yang diimplementasikan pada Citra Sentinel-2 MSI level-2A (TOC). NDTI diformulasikan sebagai berikut (Lacaux et al., 2007):

$$\text{NDTI} = \frac{\text{Red} - \text{Green}}{\text{Red} + \text{Green}}$$

Sementara formula TSS menurut hasil riset Liu et al. (2017) adalah sebagai berikut:

$$\text{TSS} = 2950 * \text{B7}^{1.357}$$

Formula TSS di atas akan menghasilkan informasi muatan suspensi air dalam satuan mg/Liter. Kelemahan NDTI adalah bahwa formula ini akan mengekstrak seluruh fitur tanah (*soil*) di atas citra, tidak peduli apakah tanah kering atau tanah di dalam air (kekeruhan). Sehingga untuk memetakan kekeruhan air (*turbidity*), kita harus melakukan *masking* citra NDTI menggunakan MNDWI. Sebab MNDWI akan memisahkan antara fitur air dan selain air.

```
import ee, geemap

ee.Authenticate()

ee.Initialize(project='ee-geospatialulm')

# Mencari informasi Image ID Sentinel-2 MSI TOC
s2_col = (
    ee.ImageCollection("COPERNICUS/S2_SR_HARMONIZED")
        .filterDate('2019-06-25', '2019-07-05')
        .filter(ee.Filter.lte('CLOUDY_PIXEL_PERCENTAGE', 50))
)

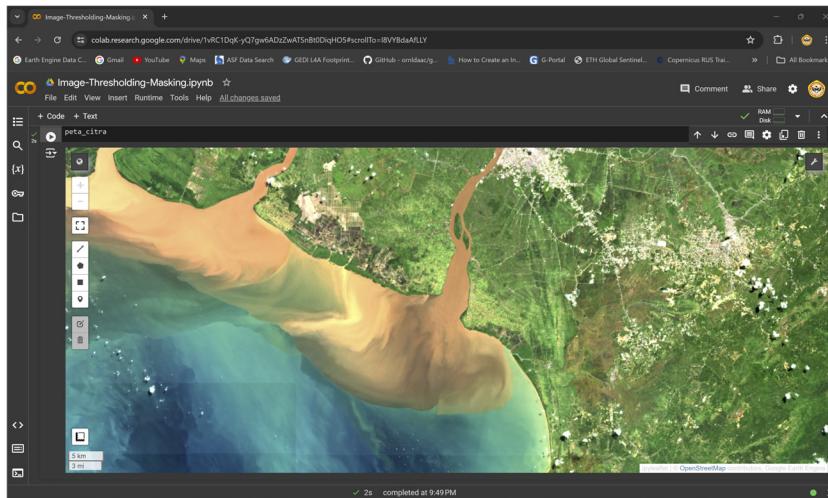
# Mengakses Citra Sentinel-2 MSI TOC
s2_image = s2_col.median().divide(10000)

# Menentukan titik lokasi citra yang akan ditampilkan
muara_barito = ee.Geometry.Point([114.490637, -3.520689])

# Visualisasi citra
rgb_vis = {'min': 0, 'max': 0.1, 'bands': ['B4', 'B3', 'B2']}
peta_citra = geemap.Map()
peta_citra.centerObject(muara_barito, 11)
peta_citra.add_layer(s2_image, rgb_vis, 'Sentinel-2 RGB')
peta_citra
```

Output:

### Bab III Google Earth Engine



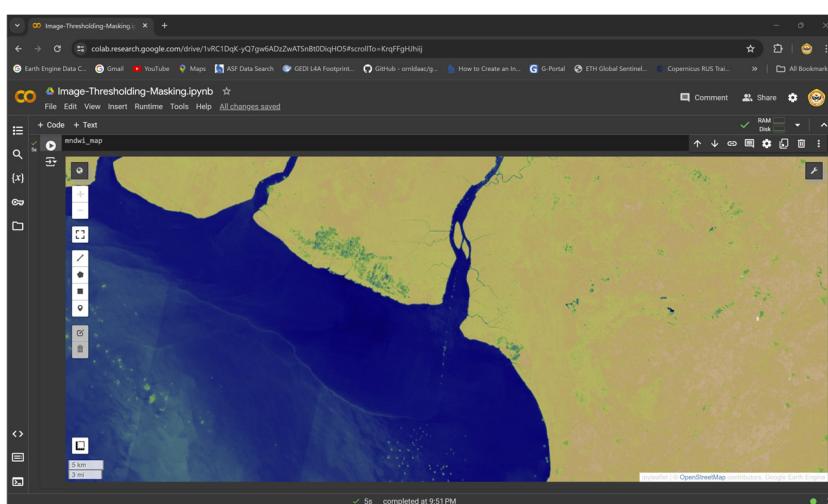
Kode di atas digunakan untuk mengakses citra Sentinel-2 MSI TOC. Sekaligus mengkonversi nilai spektralnya menjadi rentang 0 sampai 1. Hal ini dilakukan sebab nanti band-band citra akan digunakan untuk pemodelan kuantitatif.

Tambahkan sel-sel kode baru, ketikkan dan jalankan kode-kode berikut:

```
# Modified Normalized Difference Water Index (MNDWI)
mndwi = s2_image.normalizedDifference(['B3', 'B11']).rename('MNDWI')

# Visualisasi MNDWI
mndwi_vis = {'min': -1, 'max': 1, 'palette': 'gist_earth_r'}
mndwi_map = geemap.Map()
mndwi_map.centerObject(muara_barito, 11)
mndwi_map.add_layer(mndwi, mndwi_vis, 'MNDWI')
mndwi_map
```

Output:



Kode di atas digunakan untuk transformasi MNDWI. Perhatikan bahwa kode warna '`palette`': '`'gist_earth_r'`' digunakan untuk memberi warna pada citra MNDWI. Tambahan karakter `_r` pada `gist_earth_r` berfungsi untuk membalik (*reverse*) skema warna. Perhatikan gambar skema warna `gist_earth` dan `gist_earth_r` berikut:



Selanjutnya, tambahkan dan jalankan kode-kode berikut:

```
# Normalized Difference Turbidity Index (NDTI)
ndti = s2_image.normalizedDifference(['B4','B3']).rename('NDTI')
```

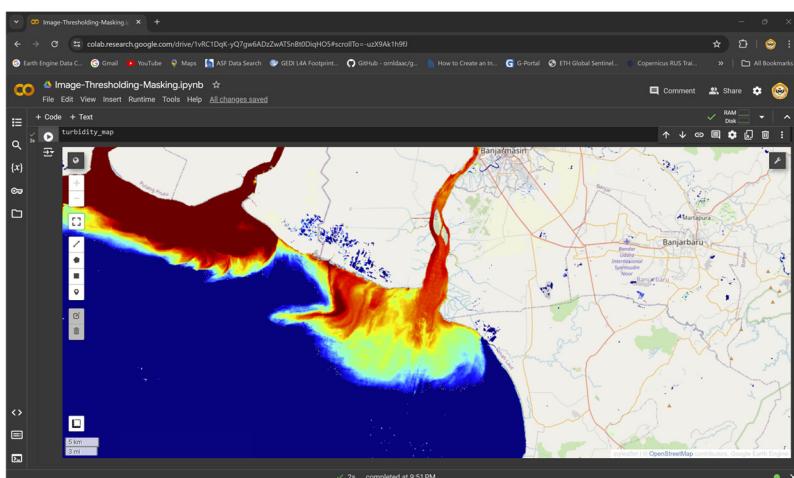
```
# Thresholding MNDWI untuk mengekstrak tubuh air
water = mndwi.gt(0)

# Masking kekeruhan air (turbidity)
turbidity = ndti.multiply(water).rename('Turbidity')
turbidity = turbidity.updateMask(turbidity.neq(0))
```

```
# Turbidity visualization
turbidity_vis = {'min': 0, 'max': 0.2, 'palette': 'jet'}

turbidity_map = geemap.Map()
turbidity_map.centerobject(muara_barito, 11)
turbidity_map.add_layer(turbidity, turbidity_vis, 'Turbidity')
turbidity_map
```

Output:



Pada kode di atas, `water = mndwi.gt(0)` merupakan thresholding nilai MNDWI. Instruksi ini akan menghasilkan sebuah citra baru dengan nama `water`. Dimana `water` merupakan *binary image*, yaitu citra yang hanya memiliki 2 nilai pixel, yaitu 0 dan 1. Pixel-pixel yang memenuhi ekspresi  $MNDWI > 0$  akan diberi nilai 1, sedangkan sisanya akan diberi nilai 0. Sehingga secara praktis, 1 berarti fitur air sedangkan 0 berarti fitur selain air.

### Bab III Google Earth Engine

Instruksi `turbidity = ndti.multiply(water)` akan mengalikan NDTI dan citra biner water. Hasil perkaliannya adalah nilai-nilai NDTI yang berada di luar fitur air akan menjadi 0, sementara pixel-pixel NDTI yang berada di dalam fitur air, yaitu kekeruhan air, nilainya akan tetap sebagai NDTI. `turbidity = turbidity.updateMask(turbidity.neq(0))` akan menerapkan *masking* untuk citra NDTI (`turbidity`), sehingga nilai `turbidity` 0 akan dibuat transparan. `turbidity.neq(0)` dibaca `turbidity not equal 0`. Hasilnya dapat dilihat pada gambar di atas, dimana kekeruhan hanya diekstrak di atas tubuh air. Sementara kekeruhan di atas daratan (tanah kering) menjadi transparan/tidak terlihat.

Lanjutkan dengan mengetikkan dan menjalankan kode-kode berikut:

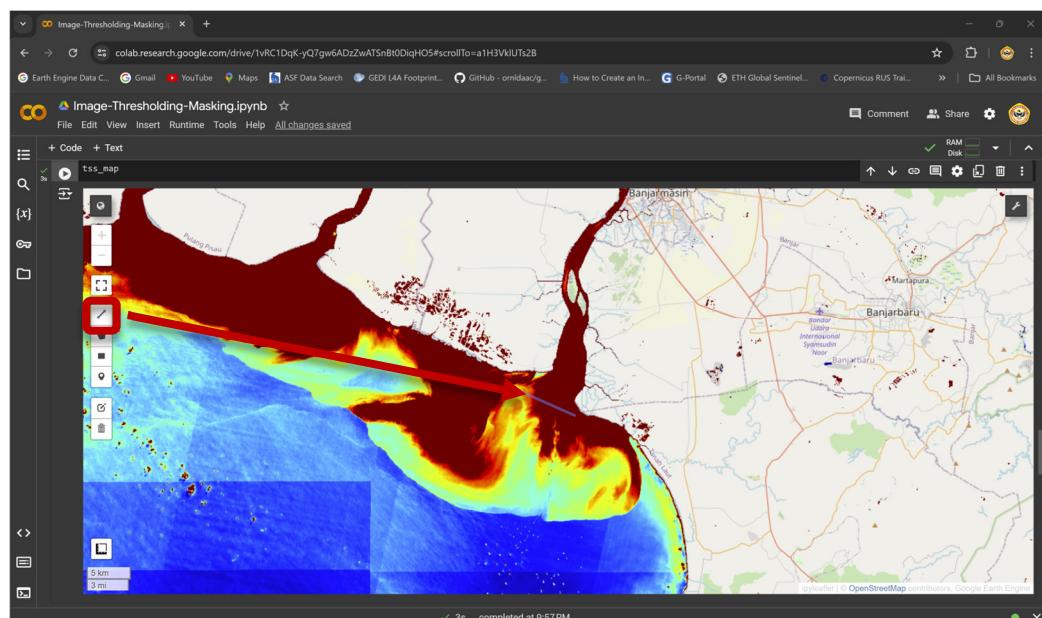
```
# Mengambil band 7 (Red Edge 3) Sentinel-2 MSI
re3 = s2_image.select('B7')

# Menghitung muatan suspensi air (TSS)
# Berdasarkan formula Liu et al. (2017)
tss = (re3.pow(1.357)).multiply(2950).rename('TSS')

# Masking TSS menggunakan tubuh air
tss = tss.multiply(water)
tss = tss.updateMask(tss.neq(0))

# TSS visualization
tss_vis = {'min': 0, 'max': 25, 'palette': 'jet'}
tss_map = geemap.Map()
tss_map.centerObject(muara_barito, 11)
tss_map.add_layer(tss, tss_vis, 'TSS')
tss_map
```

Output:



Kode di atas digunakan untuk mengekstrak TSS menggunakan formula Liu et al. (2017), yaitu dengan instruksi `tss = (re3_barito.pow(1.357)).multiply(2950)`. Sama seperti NDTI, TSS juga dimasking menggunakan fitur air (variabel water).

### Mengambil User ROI

Di atas visualisasi seperti pada gambar di atas, pengguna (*user*) dapat menggambar *Region of Interest* (ROI). Baik yang berwujud titik, garis, kotak, maupun poligon. Sebagaimana ditunjukkan pada gambar di atas, klik tombol ROI garis. Kemudian buat ROI dengan bentuk garis secara melintang di muara Sungai Barito dari arah Barat ke Timur. Sebagaimana terlihat pada gambar di atas. Kemudian ketikkan dan jalankan kode-kode berikut:

```
# Mengambil user transect
line = tss_map.user_roi

# Mengekstrak data transect
tss_data = geemap.extract_transect(
    tss, line, n_segments=100, scale=10,
    reducer='mean', to_pandas=True
)
tss_data
```

Output:

	distance	mean	
0	0.000000	16.053246	
1	67.544013	16.014989	
2	135.088025	16.029363	
3	202.632038	15.846122	
4	270.176050	16.682819	
...	...	...	
95	6416.681188	39.292470	
96	6484.225201	40.201618	
97	6551.769213	40.882808	
98	6619.313226	40.468011	
99	6686.857238	40.471232	

100 rows × 2 columns

Pada kode di atas, instruksi `line = tss_map.user_roi` akan mengambil ROI *transect* (jalur/garis) yang kita buat, dan menyimpannya ke dalam variabel `line`. Instruksi `tss_data = geemap.extract_transect` akan mengekstrak nilai-nilai TSS sepanjang ROI garis yang kita buat. Dan menyimpannya ke dalam bentuk Pandas DataFrame. Parameter `n_segments=100` akan mengambil 100 sampel TSS sepanjang garis. `scale=10` menunjukkan ukuran pixel yang kita tentukan, `reducer='mean'` menunjukkan bahwa jika nilai TSS yang diambil adalah nilai rata-rata di antara titik sepanjang garis, dan `to_pandas=True` berfungsi untuk mengkonversi output menjadi Pandas DataFrame.

### Bab III Google Earth Engine

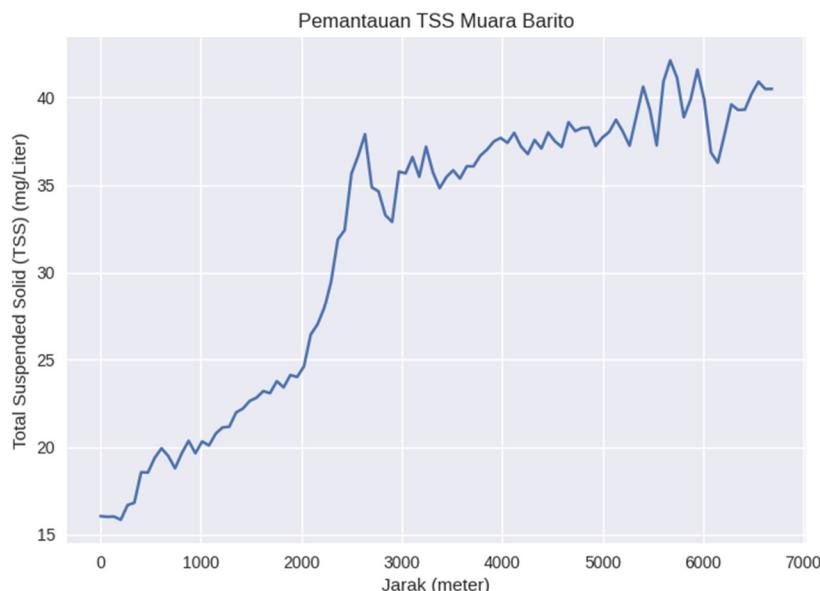
Selanjutnya, dari Pandas DataFrame di atas, akan kita buat sebuah grafik nilai-nilai TSS sepanjang transect. Untuk membuatnya, ketikkan dan jalankan kode-kode berikut:

```
# Visualisasi Transect TSS
import matplotlib.pyplot as plt
plt.style.use('seaborn-v0_8')

fig = plt.figure()
plt.plot(tss_data['distance'], tss_data['mean'])
plt.xlabel('Jarak (meter)')
plt.ylabel('Total Suspended Solid (TSS) (mg/Liter)')
plt.title('Pemantauan TSS Muara Barito')
plt.show()
```

Pada kode di atas, instruksi `plt.style.use('seaborn-v0_8')` digunakan untuk mengatur *Matplotlib plot style* pada notebook kita. `seaborn-v0_8` adalah jenis style yang kita pilih. Untuk memilih style-style lainnya, Anda dapat mengakses laman *Matplotlib style sheets reference* ([https://matplotlib.org/stable/gallery/style\\_sheets/style\\_sheets\\_reference.html](https://matplotlib.org/stable/gallery/style_sheets/style_sheets_reference.html)).

Output:



### Transformasi Spasial

Transformasi spasial dapat diterjemahkan sebagai perubahan resolusi spasial atau ukuran pixel citra digital penginderaan jauh. Salah satu metode transformasi spasial yang cukup terkenal adalah Transformasi Brovey (*Brovey Transformation*). Dengan Transformasi Brovey, kita dapat merubah band-band multispektral Landsat 7, 8, dan 9, yang memiliki resolusi spasial 30 meter menjadi 15 meter, dengan menggunakan band Pankromatiknya yang memiliki resolusi spasial 15 meter. Transformasi Brovey diformulasikan sebagai berikut (Abedini, 2000; Jensen, 2015):

$$\text{Red}_{\text{high res}} = \frac{\text{Red}_{\text{low res}}}{\text{Red}_{\text{low res}} + \text{Green}_{\text{low res}} + \text{Blue}_{\text{low res}}} * \text{Pan}_{\text{high res}}$$

$$\text{Green}_{\text{high res}} = \frac{\text{Green}_{\text{low res}}}{\text{Red}_{\text{low res}} + \text{Green}_{\text{low res}} + \text{Blue}_{\text{low res}}} * \text{Pan}_{\text{high res}}$$

$$\text{Blue}_{\text{high res}} = \frac{\text{Blue}_{\text{low res}}}{\text{Red}_{\text{low res}} + \text{Green}_{\text{low res}} + \text{Blue}_{\text{low res}}} * \text{Pan}_{\text{high res}}$$

Output dari Transformasi Brovey adalah sebuah citra komposit RGB (3 band) dengan resolusi spasial yang lebih tinggi. Untuk kasus Landsat 7, 8, dan 9, kita bebas memilih band-band yang akan dijadikan sebagai *Red*, *Green*, dan *Blue*, tergantung keperluan atau keinginan kita. Selain, Landsat, citra-citra lain yang juga memiliki band pankromatik, seperti citra-citra resolusi spasial tinggi Pleiades, WorldView, dan GeoEye, dan sebagainya, tentu saja juga dapat diterapkan Transformasi Brovey. Bahkan beberapa riset mencoba melakukan fusi citra (*image fusion*) antara Landsat-8 dan Sentinel-2 dengan menggunakan Transformasi Brovey. Sehingga dihasilkan Citra Landsat-8 dengan resolusi spasial 10 meter. Di dalam buku ini, akan disajikan sebuah contoh Transformasi Brovey Citra Landsat-9 OLI menggunakan band pankromatiknya. Buat sebuah notebook baru di dalam Google Colab, kemudian ketikkan dan jalankan kode-kode berikut:

```
import ee, geemap

ee.Authenticate()

ee.Initialize(project='ee-geospatialulm')

# Mencari informasi Image ID Landsat-9 OLI/TIRS TOA
lokasi = ee.Geometry.Point([110.057974, -7.901918])

l9_col = (
    ee.ImageCollection("LANDSAT/LC09/C02/T1_TOA")
    .filterBounds(lokasi)
    .filterDate('2022-01-01', '2024-06-15')
    .filter(ee.Filter.lte('CLOUD_COVER', 5))
)

# Menampilkan informasi citra Landsat-9
info_citra = ee.Image(l9_col)
info_citra
```

Output:

```
▼ ImageCollection LANDSAT/LC09/C02/T1_TOA (2 elements)
  type: ImageCollection
  id: LANDSAT/LC09/C02/T1_TOA
  version: 1718310049574191
  ▶ bands: List (17 elements)
  ▶ features: List (2 elements)
    ▶ 0: Image LANDSAT/LC09/C02/T1_TOA/LC09_120065_20220727 (17 bands)
    ▶ 1: Image LANDSAT/LC09/C02/T1_TOA/LC09_120066_20231221 (17 bands)
      type: Image
      id: LANDSAT/LC09/C02/T1_TOA/LC09_120066_20231221
      version: 1718310049574191
      ▶ bands: List (17 elements)
      ▶ properties: Object (106 properties)
```

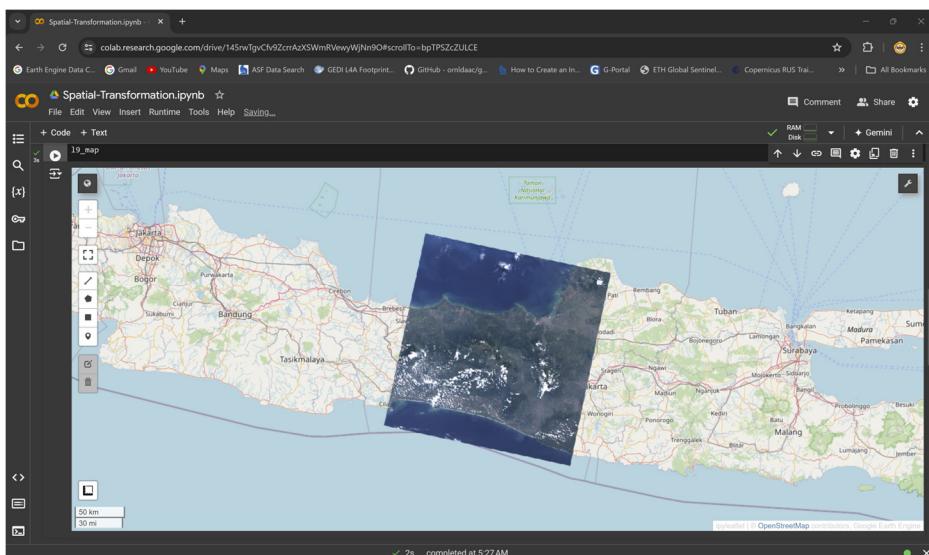
### Bab III Google Earth Engine

Pada output di atas, copy LANDSAT/LC09/C02/T1\_TOA/LC09\_120065\_20220727 dan paste ke kode berikut:

```
# Mengakses Citra Landsat-9 OLI TOA
19_image = ee.Image('LANDSAT/LC09/C02/T1_TOA/LC09_120065_20220727')
```

```
# Visualisasi citra
rgb_vis = {'min': 0, 'max': 0.25, 'bands': ['B4', 'B3', 'B2']}
19_map = geemap.Map()
19_map.centerObject(19_image)
19_map.add_layer(19_image, rgb_vis, 'Landsat-9 RGB')
19_map
```

Output:



Masih melanjutkan kode program di atas. Tambahkan sel-sel kode baru, ketikkan dan jalankan kode-kode berikut:

```
# Memilih band-band citra
red = 19_image.select('B4')
green = 19_image.select('B3')
blue = 19_image.select('B2')
pan = 19_image.select('B8')
```

```
# Red + Green + Blue
rgb_sum = red.add(green).add(blue)
```

```
# Transformasi Brovey
red_new = (red.divide(rgb_sum)).multiply(pan)
green_new = (green.divide(rgb_sum)).multiply(pan)
blue_new = (blue.divide(rgb_sum)).multiply(pan)
```

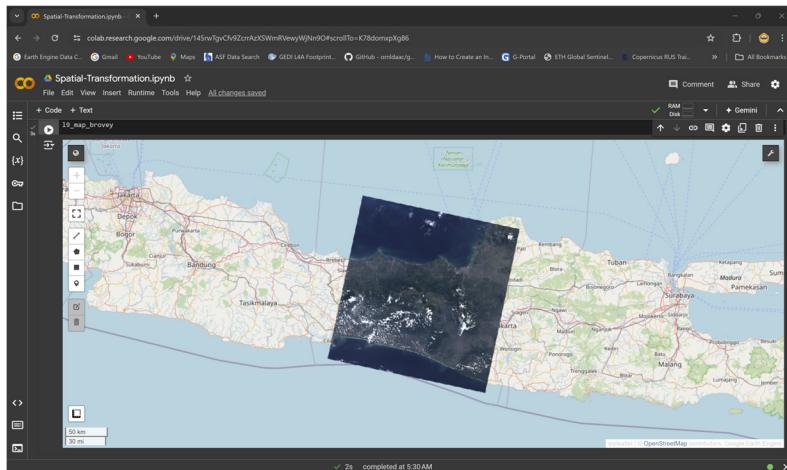
```
# Menggabungkan band-band hasil Transformasi Brovey
19_image_new = red_new.addBands([green_new, blue_new])
```

```
# visualisasi Citra transformasi Brovey

rgb_vis = {'min': 0, 'max': 0.1, 'bands': ['B4', 'B3', 'B2']}

19_map_brovey = geemap.Map()
19_map_brovey.centerObject(19_image_new)
19_map_brovey.add_layer(19_image_new, rgb_vis, 'Landsat-9 RGB')
19_map_brovey
```

Output:



Pada kedua output citra di atas, sepintas memang tidak terlihat perbedaan antara Landsat-9 komposit RGB yang original (resolusi spasial 30 meter), dengan hasil Transformasi Brovey (resolusi spasial 15 meter). Akan tetapi, jika kita menunjuk ke lokasi tertentu secara detail, maka akan terlihat jelas perbedaannya.

Pada kode-kode di atas, ganti instruksi `19_map.centerObject(19_image)` dengan instruksi `19_map.centerObject(lokasi, 16)` sebagaimana terlihat seperti pada kode di bawah. Kemudian jalankan:

```
# visualisasi citra

rgb_vis = {'min': 0, 'max': 0.25, 'bands': ['B4', 'B3', 'B2']}

19_map = geemap.Map()
19_map.centerObject(lokasi, 16)
19_map.add_layer(19_image, rgb_vis, 'Landsat-9 RGB')
19_map
```

Ganti juga instruksi `19_map_brovey.centerObject(19_image_new)` dengan instruksi `19_map_brovey.centerObject(lokasi, 16)` sebagaimana terlihat seperti pada kode di bawah. Kemudian jalankan:

```
# visualisasi Citra transformasi Brovey

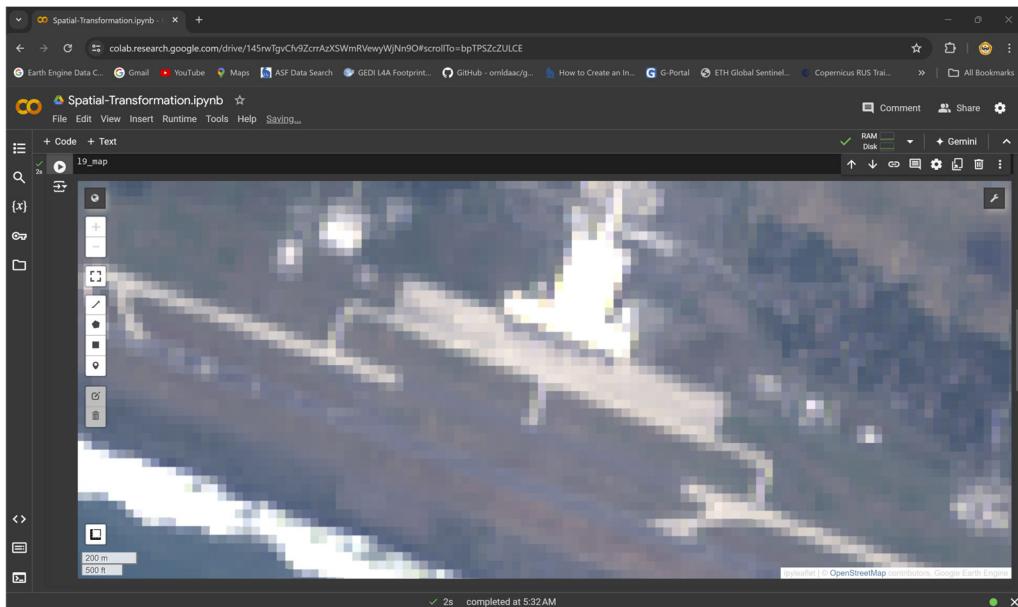
rgb_vis = {'min': 0, 'max': 0.1, 'bands': ['B4', 'B3', 'B2']}

19_map_brovey = geemap.Map()
19_map_brovey.centerObject(lokasi, 16)
19_map_brovey.add_layer(19_image_new, rgb_vis, 'Landsat-9 RGB')
19_map_brovey
```

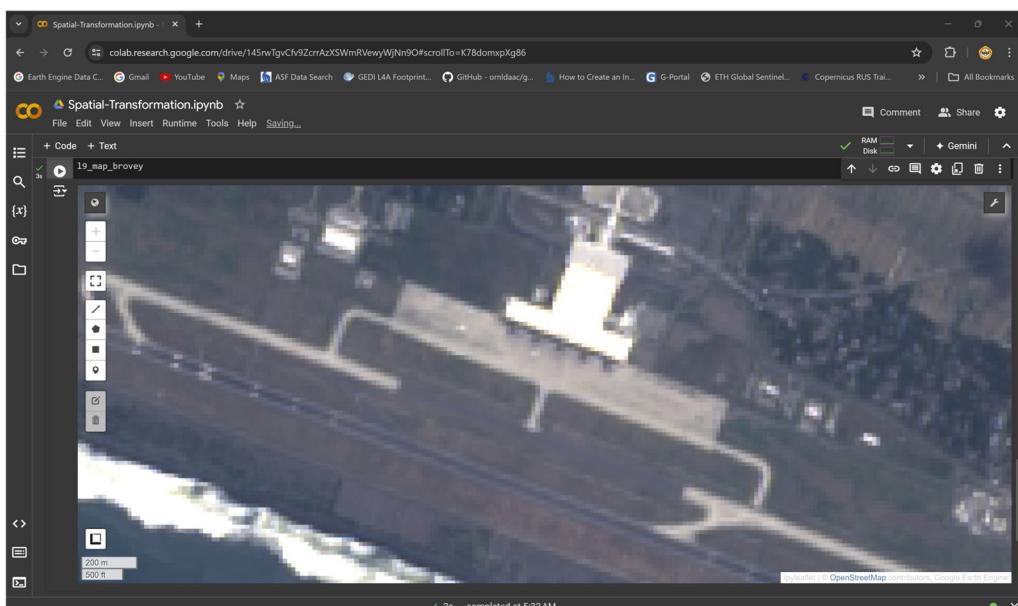
### Bab III Google Earth Engine

Perhatikan dengan teliti output kedua kode program di atas.

Output citra original:



Output Citra Landsat-9 hasil Transformasi Brovey:



Sekarang perhatikan perbedaannya, antara Landsat-9 RGB original, dengan Landsat-9 RGB hasil Transformasi Brovey. lokasi = ee.Geometry.Point([110.057974, -7.901918]) merupakan koordinat Yogyakarta International Airport, Kulon Progo. Terlihat jelas perbedaannya secara visual antara Yogyakarta International Airport pada Landsat-9 RGB resolusi spasial 30 meter, dengan Yogyakarta International Airport pada Landsat-9 RGB resolusi spasial 15 meter.

## Transformasi Temporal

Bagi sebagian praktisi penginderaan jauh, transformasi temporal mungkin akan terdengar sedikit tidak biasa. Akan tetapi, jika kita baca beberapa artikel di jurnal ilmiah, sudah banyak yang mengembangkan metode transformasi temporal ini. Transformasi temporal pernah diterapkan oleh Zeng et al. (2013) untuk memulihkan pixel-pixel Landsat 7 ETM+ SLC-off (*stripping*) yang hilang, dan Wu et al. (2024) untuk mengisi diskontinuitas pixel-pixel multitemporal yang hilang akibat penutupan awan atau malfungsi pada sensor optik. Secara singkat, transformasi temporal dapat diterjemahkan sebagai perubahan waktu akuisisi citra. Apakah itu mungkin? Tentu saja, waktu akuisisi citra secara definitif tidak dapat dirubah. Waktu dirubah dalam konteks ini maksudnya adalah waktu akuisisi citranya disesuaikan atau disimulasikan. Sehingga citra akuisisi 20 Januari 2024 misalnya, dibuat seolah-olah menjadi akuisisi 15 Juni 2024.

Cukup banyak potensi aplikasi transformasi temporal. Contoh konvensional adalah di dalam proses koreksi awan. Dimana awan pada citra akuisisi terbaru dimasking (dibuang), kemudian citra ditambal dengan citra akuisisi yang lebih lama, pada bagian yang bersih dari awan. Jika ditambahkan begitu saja tanpa ada proses transformasi, biasanya “tambalannya” akan terlihat. Sebagaimana kain baru ditambal dengan kain lama, meskipun jenis dan motif atau warna kainnya sama persis, tentu kain lama akan terlihat pudar. Sehingga di dalam proses tambal-sulam citra multitemporal, diperlukan penyesuaian waktu akuisisi citra, agar seakan-akan citra diakuisisi pada waktu yang sama. Sebagaimana transformasi spasial, transformasi temporal pada dasarnya adalah transformasi spektral juga, sebab pasti akan merubah nilai spektral. Disamping formula transformasinya pada dasarnya adalah transformasi spektral. Akan tetapi, untuk membedakan tujuan dan luaran transformasinya, perubahan resolusi spasial dapat kita sebut sebagai transformasi spasial, dan simulasikan waktu akuisisi dapat kita sebut sebagai transformasi temporal.

Sebagai contoh kasus, sebuah riset dilakukan pada pertengahan tahun 2023. Dimana pengambilan sampel lapangan dilakukan pada akhir bulan Juni 2023. Untuk menghasilkan sebuah model statistika yang representatif dan akurat nantinya, maka waktu akuisisi citra juga harus menyesuaikan, yaitu pada sekitar akhir Juni 2023. Permasalahannya adalah, Citra-citra Sentinel-2 MSI akuisisi sekitar akhir Juni 2023 untuk lokasi kajian tidak ada yang sepenuhnya bersih dari awan. Sehingga dalam hal ini harus dilakukan tambal-sulam citra multitemporal. Dengan mengambil waktu akhir Juni 2023 sebagai acuan waktu akuisisi citra. Buat sebuah notebook Google Colab baru, beri nama misalnya **Temporal-Transformation.ipynb**. Kemudian ketikkan dan jalankan kode-kode berikut:

```
import ee, geemap
ee.Authenticate()
ee.Initialize(project='ee-geospatialulm')

# Mengimpor shapefile
import geopandas as gpd
from google.colab import drive
drive.mount('/content/gdrive')
path = '/content/gdrive/MyDrive/geebook/'
babaris_shp = gpd.read_file(path + 'vector/Babarisi.shp')
```

### Bab III Google Earth Engine

```
# Konversi shapefile ke Geometry GEE
import json
babaris_js = json.loads(babarism_shp.to_json())
babaris_fc = ee.FeatureCollection(babarism_js)
babaris = ee.Geometry(babarism_fc.geometry())
```

```
# Menentukan tanggal citra
tgl_awal = '2023-01-01'
tgl_akhir = '2023-06-30'
```

```
# Mengakses Sentinel-2 MSI TOC image collection
s2_col = (
    ee.ImageCollection("COPERNICUS/S2_SR_HARMONIZED")
    .filterDate(tgl_awal,tgl_akhir)
    .filterBounds(babarism)
    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE',20))
)
ee.Image(s2_col)
```

Output:

```
▼ ImageCollection COPERNICUS/S2_SR_HARMONIZED (4 elements)
  type: ImageCollection
  id: COPERNICUS/S2_SR_HARMONIZED
  version: 1718565998621576
  ▶ bands: List (23 elements)
  ▶ properties: Object (21 properties)
  ▼ features: List (4 elements)
    ▶ 0: Image COPERNICUS/S2_SR_HARMONIZED/20230525T022531_20230525T023627_T50MKB (23 bands)
    ▶ 1: Image COPERNICUS/S2_SR_HARMONIZED/20230530T022539_20230530T023804_T50MKB (23 bands)
      type: Image
      id: COPERNICUS/S2_SR_HARMONIZED/20230530T022539_20230530T023804_T50MKB
      version: 1718565998621576
      ▶ bands: List (23 elements)
      ▶ properties: Object (101 properties)
    ▶ 2: Image COPERNICUS/S2_SR_HARMONIZED/20230609T022539_20230609T023805_T50MKB (23 bands)
    ▶ 3: Image COPERNICUS/S2_SR_HARMONIZED/20230624T022531_20230624T023804_T50MKB (23 bands)
      type: Image
      id: COPERNICUS/S2_SR_HARMONIZED/20230624T022531_20230624T023804_T50MKB
      version: 1718565998621576
      ▶ bands: List (23 elements)
      ▶ properties: Object (101 properties)
```

Kode-kode di atas digunakan untuk mencari Citra Sentinel-2 MSI TOC di sebuah wilayah kajian, pada rentang waktu 1 Januari 2023 hingga 30 Juni 2023, dan dengan tutupan awan minimum (kurang dari 20%). Citra yang akan diambil adalah akuisisi tanggal 24 Juni 2023 (**COPERNICUS/S2\_SR\_HARMONIZED/20230624T022531\_20230624T023804\_T50MKB**). Hal ini menyesuaikan dengan waktu pengambilan data lapangan. Dikarenakan citra akuisisi 24 Juni 2023 ini terdapat penutupan awan, maka akan dicari citra akuisisi sebelum atau sesudah 24 Juni 2023, dengan sebaran awan yang tidak sama dengan akuisisi 24 Juni 2023. Maksudnya lokasi dimana terdapat awan pada citra 24 Juni 2023, akan dicarikan pasangan citra di tanggal yang berbeda dimana di lokasi tersebut bersih dari awan.

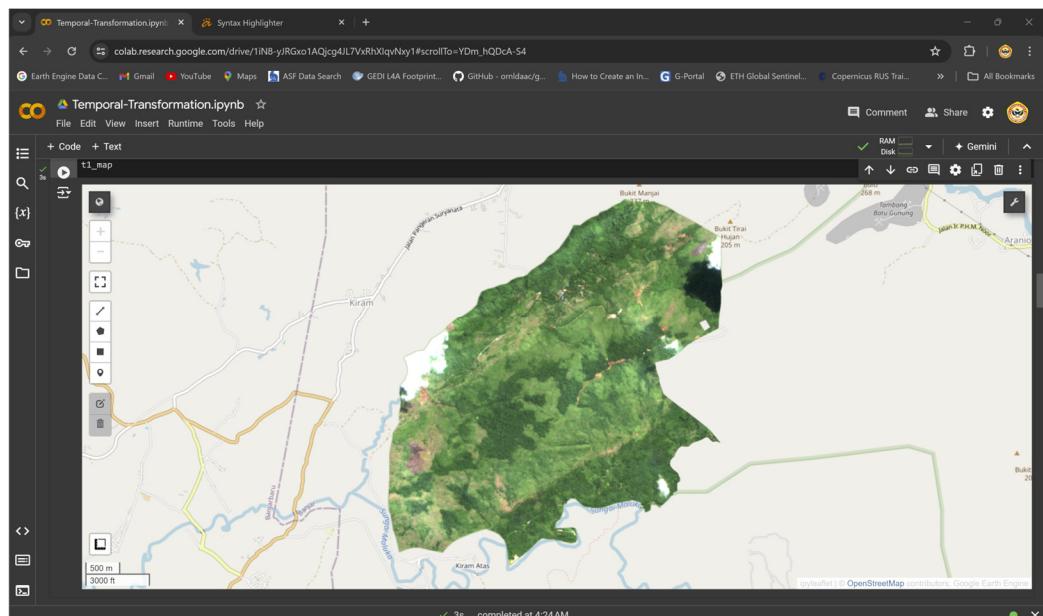
Proses pencarian pasangan citra ini sifatnya *trial and error*, dimana kita coba lihat secara visual semua citra yang terdekat dengan 24 Juni 2023, dan sebaran awannya berbeda dengan citra akuisisi 24 Juni 2023. Dari hasil penelusuran, ditemukan bahwa Sentinel-2 akuisisi 30 Mei 2023 (**COPERNICUS/S2\_SR\_HARMONIZED/20230530T022539\_20230530T023804\_T50MKB**) adalah yang paling dekat dan paling sesuai untuk menambal awan pada Sentinel-2 akuisisi tanggal 24 Juni 2023. Untuk selanjutnya, ketikkan dan jalankan kode-kode berikut:

```
# Mengakses citra multitemporal
s2_t1_image = (
    ee.Image('COPERNICUS/S2_SR_HARMONIZED/20230530T022539_20230530T023804_T50MKB')
        .select('B2', 'B3', 'B4')
        .divide(10000)
)
s2_t2_image = (
    ee.Image('COPERNICUS/S2_SR_HARMONIZED/20230624T022531_20230624T023804_T50MKB')
        .select('B2', 'B3', 'B4')
        .divide(10000)
)

# Clipping citra
s2_babarais_t1 = s2_t1_image.clip(babarais)
s2_babarais_t2 = s2_t2_image.clip(babarais)

# Visualisasi citra Sentinel-2 MSI TOC t1
rgb_vis = {'min': 0, 'max': 0.1, 'bands': ['B4', 'B3', 'B2']}
t1_map = geemap.Map()
t1_map.centerobject(babarais, 14)
t1_map.addlayer(s2_babarais_t1, rgb_vis, 'Sentinel-2 RGB Gunung Babarais')
t1_map
```

Output:

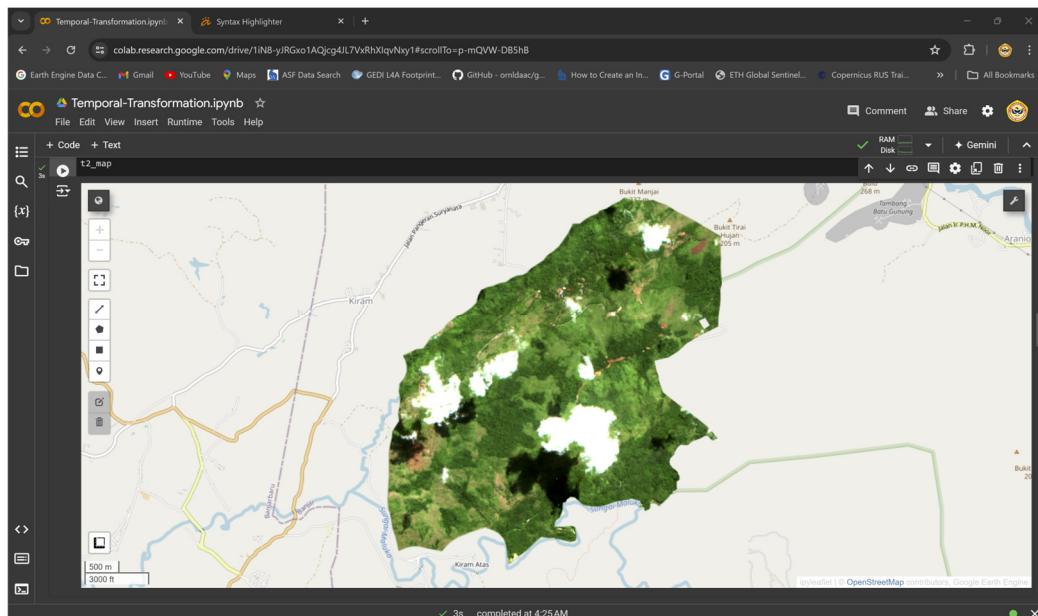


### Bab III Google Earth Engine

Lanjutkan dengan mengetikkan dan menjalankan kode-kode berikut:

```
# Visualisasi citra Sentinel-2 MSI TOC t2
rgb_vis = {'min': 0, 'max': 0.1, 'bands': ['B4','B3','B2']}
t2_map = geemap.Map()
t2_map.centerobject(babarisi,14)
t2_map.add_layer(s2_babarisi_t2, rgb_vis, 'Sentinel-2 RGB Gunung Babarisi')
t2_map
```

Output:



Pada kode-kode di atas, Sentinel-2 akuisisi 24 Juni 2023 disebut  $t_2$ , dan Sentinel-2 akuisisi sebelumnya, yaitu 30 Mei 2023 disebut  $t_1$ . Perhatikan kenampakan sebaran awannya pada kedua gambar di atas. Sebenarnya citra  $t_1$  relatif lebih bersih dari awan, akan tetapi waktu akuisisinya terlalu jauh dengan waktu survey lapangan. Sehingga dalam hal ini, citra  $t_2$  akan ditambah menggunakan citra  $t_1$ . Dikarenakan proses transformasi temporal untuk koreksi awan dilakukan band per band, maka untuk mempercepat proses dan mempersingkat contoh kode, koreksi awan kita batasi hanya untuk band 2, band 3, dan band 4 Sentinel-2 MSI TOC.

Lanjutkan dengan mengetikkan dan menjalankan kode berikut:

```
# Mengakses Sentinel-2: Cloud Probability
s2_cloud = (
    ee.ImageCollection("COPERNICUS/S2_CLOUD_PROBABILITY")
    .filterDate(tgl_awal,tgl_akhir)
    .filterBounds(babarisi)
) ee.Image(s2_cloud)
```

Output:

```

▶ 30: Image COPERNICUS/S2_CLOUD_PROBABILITY/20230530T022539_20230530T023804_T50MKB (1 band)
▶ 31: Image COPERNICUS/S2_CLOUD_PROBABILITY/20230604T022531_20230604T023804_T50MKB (1 band)
▶ 32: Image COPERNICUS/S2_CLOUD_PROBABILITY/20230609T022539_20230609T023805_T50MKB (1 band)
▶ 33: Image COPERNICUS/S2_CLOUD_PROBABILITY/20230614T022541_20230614T023805_T50MKB (1 band)
▶ 34: Image COPERNICUS/S2_CLOUD_PROBABILITY/20230619T022539_20230619T023805_T50MKB (1 band)
▼ 35: Image COPERNICUS/S2_CLOUD_PROBABILITY/20230624T022531_20230624T023804_T50MKB (1 band)
  type: Image
  id: COPERNICUS/S2_CLOUD_PROBABILITY/20230624T022531_20230624T023804_T50MKB
  version: 1687609673545928
  ▶ bands: List (1 element)
  ▶ properties: Object (5 properties)
▶ 36: Image COPERNICUS/S2_CLOUD_PROBABILITY/20230629T022539_20230629T024757_T50MKB (1 band)

```

Kode-kode di atas digunakan untuk mengakses dataset *Sentinel-2: Cloud Probability* ([https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS\\_S2\\_CLOUD\\_PROBABILITY](https://developers.google.com/earth-engine/datasets/catalog/COPERNICUS_S2_CLOUD_PROBABILITY)) dari Earth Engine Data Catalog. *Sentinel-2: Cloud Probability* menyediakan informasi probabilitas kehadiran awan di atas Citra *Sentinel-2* MSI untuk setiap waktu akuisisi. Probabilitas awan disajikan dalam bentuk persen, pada kisaran 0% hingga 100%. Sehingga *Sentinel-2: Cloud Probability* pada dasarnya adalah sebuah citra yang memiliki nilai pixel 0 sampai 100. Dalam hal ini, *Sentinel-2: Cloud Probability* dapat digunakan untuk proses masking awan pada *Sentinel-2*. Tentu saja, karena yang akan dimasking awannya adalah *Sentinel-2* akuisisi 24 Juni 2023, maka *Sentinel-2: Cloud Probability* yang digunakan juga harus tanggal yang sama, 24 Juni 2023 (**COPERNICUS/S2\_CLOUD\_PROBABILITY/20230624T022531\_20230624T023804\_T50MKB**).

Lanjutkan proses dengan mengetikkan dan menjalankan kode-kode berikut:

```

# Mengakses dan memotong Sentinel-2 t2 Cloud Probability
s2_t2_cloud = (
ee.Image('COPERNICUS/S2_CLOUD_PROBABILITY/20230624T022531_20230624T023804_T50MKB'
')

s2_babarис_t2_cloud = s2_t2_cloud.clip(babarис)

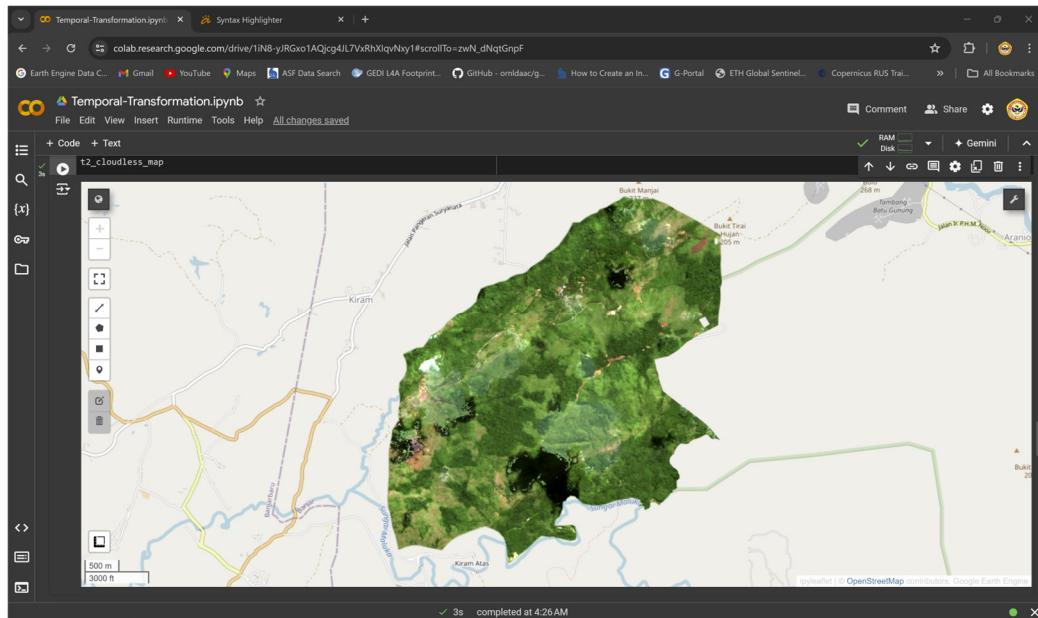
# Masking awan menggunakan Sentinel-2 Cloud Probability
s2_babarис_t1_mask = s2_babarис_t2_cloud.gte(10)
s2_babarис_t2_mask = s2_babarис_t2_cloud.lt(10)

# Menambal Sentinel-2 t2 menggunakan t1
s2_babarис_t1_masked = s2_babarис_t1.multiply(s2_babarис_t1_mask)
s2_babarис_t2_masked = s2_babarис_t2.multiply(s2_babarис_t2_mask)
s2_babarис_t2_cloudless = s2_babarис_t2_masked.add(s2_babarис_t1_masked)

# Visualisasi citra Sentinel-2 MSI TOC t2 cloudless
rgb_vis = {'min': 0, 'max': 0.1, 'bands': ['B4', 'B3', 'B2']}
t2_cloudless_map = geemap.Map()
t2_cloudless_map.centerobject(babarис, 14)
t2_cloudless_map.add_layer(s2_babarис_t2_cloudless, rgb_vis, 'Sentinel-2 RGB
Gunung Babarис')
t2_cloudless_map

```

Output:



Kode-kode di atas digunakan untuk proses masking awan, sekaligus tambal-sulam Sentinel-2 akuisisi 24 Juni 2023 dengan Sentinel-2 akuisisi 30 Mei 2023. Sentinel-2 24 Juni 2023 dimasking dengan logika `gte(10)`, artinya bagian citra dengan probabilitas kehadiran awan lebih besar dari atau sama dengan (*greater than or equal*) pada Sentinel-2 24 Juni 2023 10% dibuang. Sedangkan Sentinel-2 30 Juni 2023 dimasking dengan logika `lt(10)`, artinya bagian citra dengan probabilitas kehadiran awan lebih kecil dari (*less than*) pada Sentinel-2 24 Juni 2023 10% dibuang.

Perhatikan hasil tambalannya sebagaimana terlihat pada gambar di atas. Jika diperhatikan dengan detail, maka akan terlihat batas tambalannya. Hal ini dikarenakan berbagai faktor yang dapat menjadi penyebabnya, antara lain perbedaan kondisi atmosferik, perbedaan intensitas penyinaran matahari, dan perbedaan sudut pandang matahari pada saat akuisisi kedua citra. Meskipun pada faktanya perbedaan waktu akuisisi kedua citra tidak sampai 1 bulan.

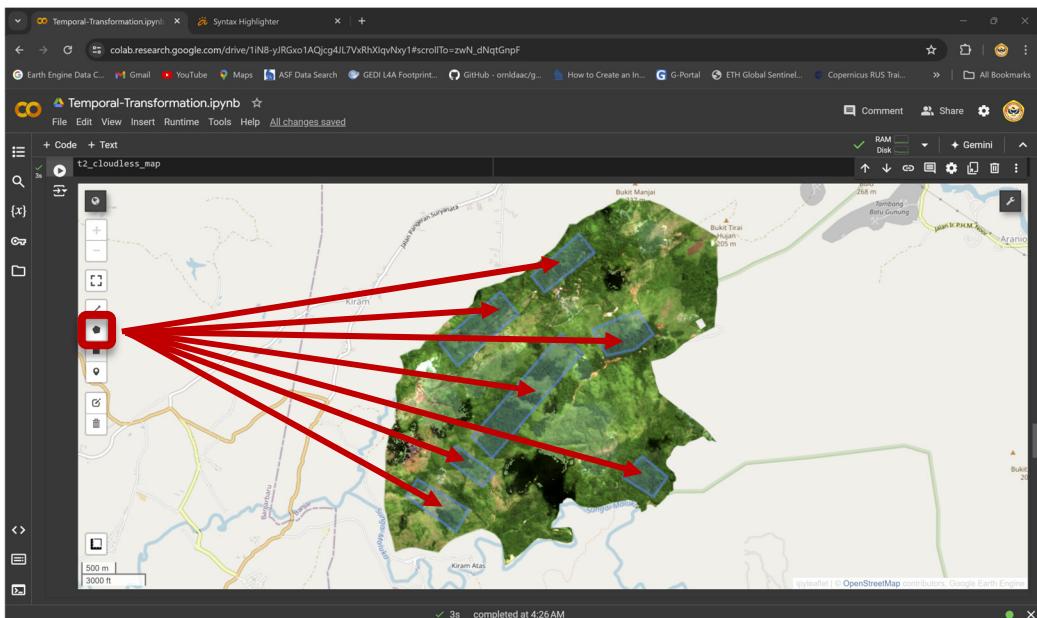
Untuk mengatasi hal tersebut, maka Sentinel-2 akuisisi 30 Mei 2023 harus ditransformasi secara temporal. Sehingga waktu akuisisinya seolah-olah menjadi sama dengan Sentinel-2 24 Juni 2023. Metode transformasi temporal yang diadaptasi di dalam buku ini diadopsi dari teknik normalisasi citra multitemporal menurut Ilsever and Unsalan (2012), dengan formula sebagai berikut:

$$\tilde{I}_{t1}(x, y) = \frac{\sigma_{t2}}{\sigma_{t1}}(I_{t1}(x, y) - \mu_{t2}) + \mu_{t1}$$

Dimana:

- $\tilde{I}_{t1}(x, y)$  : Intensitas (nilai spektral) citra  $t_1$  hasil transformasi temporal untuk setiap pixel  $(x, y)$
- $I_{t1}(x, y)$  : Intensitas (nilai spektral) citra original  $t_1$  yang akan ditransformasi temporal
- $\sigma_{t1}$  : Standar deviasi nilai spektral citra  $t_1$  (citra penambal yang ditransformasi temporal)
- $\sigma_{t2}$  : Standar deviasi nilai spektral citra  $t_2$  (citra yang akan ditambah)
- $\mu_{t1}$  : Rerata nilai spektral citra  $t_1$  (citra penambal yang ditransformasi temporal)
- $\mu_{t2}$  : Rerata nilai spektral citra  $t_2$  (citra yang akan ditambah)

Parameter-parameter standar deviasi dan rerata nilai spektral pada kedua citra untuk formula di atas harus diestimasi dari bagian-bagian citra yang bersih dari segala gangguan, baik bersih dari awan maupun bersih dari bayangan awannya. Sehingga untuk keperluan ini, kita harus membuat sejumlah sampel citra pada bagian yang bebas gangguan, sebagaimana pada gambar berikut:



Pada gambar di atas, kita membuat sejumlah sampel atau ROI dalam bentuk poligon pada bagian-bagian citra yang bersih dari gangguan awan dan bayangan awannya. Bersih di sini maksudnya adalah harus bersih pada kedua citra  $t_1$  dan  $t_2$ , bukan hanya bersih untuk citra  $t_2$  saja. Sebab ROI yang sama akan kita pakai untuk estimasi parameter-parameter statistik pada kedua citra  $t_1$  dan  $t_2$ . Sehingga pada saat pembuatan ROI, kita harus benar-benar teliti dalam melihat sebaran awan pada kedua citra. Pastikan ROI kita tempatkan pada bagian yang bersih pada citra  $t_1$  dan bersih juga pada citra  $t_2$ . Sebenarnya, ROI yang ideal adalah per fitur atau per kelas penutupan lahan. Agar lebih representatif secara statistik, akan tetapi hal ini akan jauh lebih rumit.

Selanjutnya, ketikkan dan jalankan kode-kode berikut:

```
# Mengambil user ROI
clear_roi = t2_cloudless_map.user_roi
```

```
# Menghitung statistik Sentinel-2 t1
t1_stats = (
    geemap.image_stats(s2_babarisan_t1.clip(clear_roi), scale=10)
    .getInfo()
)
b2_t1_mean = t1_stats['mean']['B2']
b3_t1_mean = t1_stats['mean']['B3']
b4_t1_mean = t1_stats['mean']['B4']

b2_t1_std = t1_stats['std']['B2']
b3_t1_std = t1_stats['std']['B3']
b4_t1_std = t1_stats['std']['B4']
```

### Bab III Google Earth Engine

```
# Menghitung statistik Sentinel-2 t2
t2_stats = (
    geemap.image_stats(s2_babarisis_t2.clip(clear_roi), scale=10)
    .getInfo()
)

b2_t2_mean = t2_stats['mean']['B2']
b3_t2_mean = t2_stats['mean']['B3']
b4_t2_mean = t2_stats['mean']['B4']

b2_t2_std = t2_stats['std']['B2']
b3_t2_std = t2_stats['std']['B3']
b4_t2_std = t2_stats['std']['B4']

# Transformasi temporal untuk normalisasi nilai spektral t1
# berdasarkan formula Ilsever and Unsalan (2012)

b2_t1 = s2_babarisis_t1.select('B2')
b3_t1 = s2_babarisis_t1.select('B3')
b4_t1 = s2_babarisis_t1.select('B4')

b2_t1_norm = (
    (b2_t1.subtract(b2_t1_mean))
    .multiply(b2_t2_std/b2_t1_std)
    .add(b2_t2_mean)
)

b3_t1_norm = (
    (b3_t1.subtract(b3_t1_mean))
    .multiply(b3_t2_std/b3_t1_std)
    .add(b3_t2_mean)
)

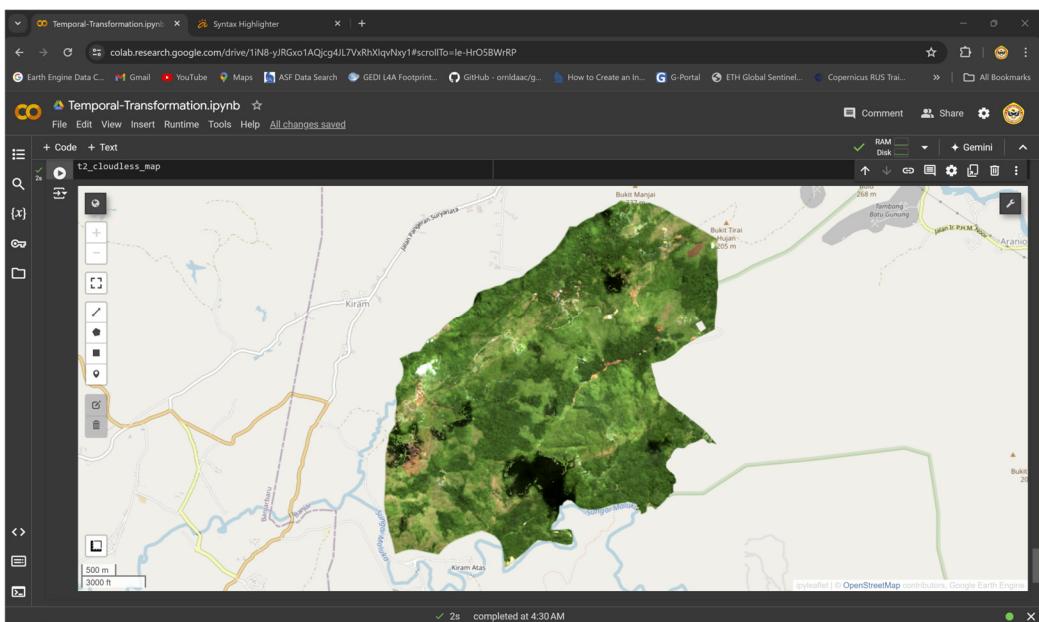
b4_t1_norm = (
    (b4_t1.subtract(b4_t1_mean))
    .multiply(b4_t2_std/b4_t1_std)
    .add(b4_t2_mean)
)

# Menggabungkan band-band hasil transformasi temporal
s2_babarisis_t1_norm = b2_t1_norm.addBands([b3_t1_norm, b4_t1_norm])

# Menambal Sentinel-2 t2 menggunakan t1 normalized
s2_babarisis_t1_masked = s2_babarisis_t1_norm.multiply(s2_babarisis_t1_mask)
s2_babarisis_t2_masked = s2_babarisis_t2.multiply(s2_babarisis_t2_mask)
s2_babarisis_t2_cloudless = s2_babarisis_t2_masked.add(s2_babarisis_t1_masked)

# Visualisasi citra Sentinel-2 MSI TOC t2 cloudless
rgb_vis = {'min': 0, 'max': 0.1, 'bands': ['B4', 'B3', 'B2']}
t2_cloudless_map = geemap.Map()
t2_cloudless_map.centerObject(babarisis, 14)
t2_cloudless_map.add_layer(s2_babarisis_t2_cloudless, rgb_vis, 'Sentinel-2 RGB Gunung Babarisis')
t2_cloudless_map
```

Output:



Perhatikan hasil akhirnya sebagaimana terlihat pada gambar di atas. Jika diperhatikan dengan detail, tambalan citranya sekarang jauh lebih mirip dengan area-area di sekitarnya, terutama untuk objek-objek yang sama. Jika masih terdapat perbedaan secara visual, atau dengan kata lain tambalannya masih terlihat jelas, hal itu disebabkan karena ketidakakuratan estimasi parameter-parameter statistik yang digunakan di dalam transformasi. Hal ini dapat terjadi karena kekurangtelitian pada saat pembuatan ROI, atau sampel-sampel ROI-nya yang memang kurang sehingga tidak merepresentasikan karakter statistik nilai spektral kedua citra secara keseluruhan. Jangan ragu-ragu untuk mengulang proses pembuatan ROI dan mengulang kembali proses transformasi temporal, sehingga dihasilkan kenampakan citra yang benar-benar terlihat tanpa tambalan.

Tambalan citra yang terlihat jelas juga dapat terjadi jika jarak waktu akuisisi kedua citra terpaut terlalu jauh, misalnya beberapa bulan atau 1 tahun. Sehingga secara faktual di lapangan sudah terjadi perubahan penutupan lahan. Dalam hal ini, kita harus mengusahakan secara maksimal agar kedua citra sedekat mungkin waktu akuisisinya. Lebih jauh, contoh proses di atas hanya diterapkan pada 3 band, yaitu band 2, band 3, dan band 4 Citra Sentinel-2 MSI TOC. Jika Anda ingin seluruh band dikoreksi awan, tentu saja estimasi parameter-parameter statistik dan transformasi temporal harus diterapkan pada seluruh band. Termasuk ketika metode di atas diterapkan pada citra lain, misalnya Landsat, tekniknya akan sedikit berbeda. Terutama perbedaan pada teknik masking awannya.

Jika diperhatikan dengan detail, proses pemilihan citra penambalan, yaitu citra akuisisi 30 Mei 2023 di atas sebenarnya belum begitu tepat. Sebab ternyata masih terdapat awan dengan posisi sama antar kedua citra. Akibatnya, yang terjadi adalah awan ditambal dengan awan, sebagaimana terlihat pada gambar di atas. Jika memang sulit untuk menemukan 2 citra dengan lokasi awan yang sepenuhnya berbeda, maka akan digunakan citra multitemporal dengan lebih dari 2 akuisisi. Sehingga prosesnya akan lebih rumit dari contoh kode di atas, sebab akan melibatkan banyak waktu akuisisi citra. Termasuk bayangan-bayangan awannya juga harus dideteksi dan dimasking, untuk ditambal dengan citra di waktu akuisisi yang berbeda. Terkait dengan teknik pembuatan mosaik Citra Sentinel-2 multitemporal bebas awan, silahkan lihat kembali pembahasan pada bagian terdahulu dari halaman 187 sampai halaman 193.

## C. Pemfilteran Citra

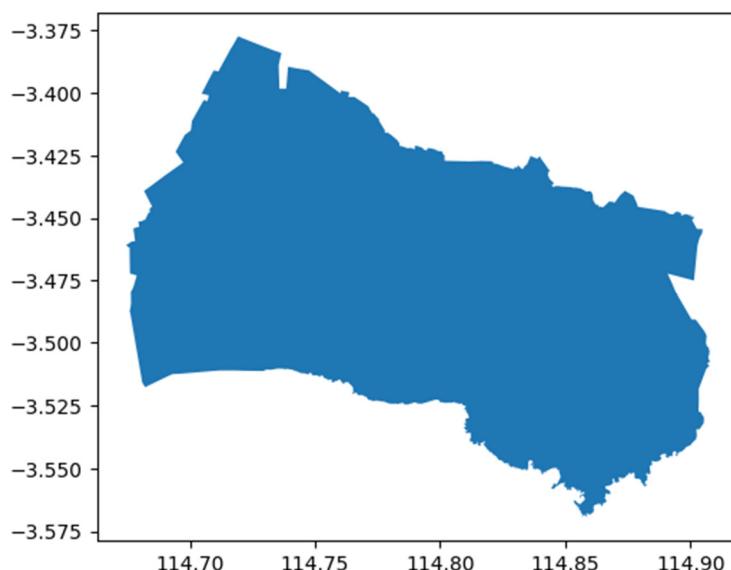
Pemfilteran citra (*image filtering*) merupakan proses matematis untuk memanipulasi nilai piksel dalam citra penginderaan jauh agar meningkatkan kualitas visual, mengurangi *noise*, atau menonjolkan fitur tertentu sebelum analisis lanjutan. Tujuan utamanya meliputi pereduksian gangguan (*random noise, speckle, striping*), penajaman (*enhancement*) tepi dan tekstur, penghalusan (*smoothing*) area seragam, dan penguatan fitur spasial (tepi sungai, pola aliran, batas vegetasi). Terdapat banyak sekali metode-metode pemfilteran citra, dan seiring waktu akan terus berkembang. Di dalam buku ini hanya disajikan sebagian contoh metode dan teknik pemfilteran citra. Yaitu filter tekstur, filter morfologi, dan filter konvolusi.

### Filter Tekstur

Buat sebuah notebook Google Colab baru, beri nama misalnya **Filter-Tekstur.ipynb**. Kemudian ketikkan dan jalankan kode-kode berikut:

```
import ee, geemap
ee.Authenticate()
ee.Initialize(project='ee-geospatialulm')
# Mengakses Google Drive
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

# Membuka shapefile
import geopandas as gpd
path = '/content/drive/My Drive/geebook/vector/'
region_shp = gpd.read_file(path + 'Banjarbaru.shp')
region_shp.plot()
```



```
# Konversi shapegile ke Earth Engine geometry
region = geemap.geopandas_to_ee(region_shp).geometry()
```

```
# Menentukan rentang tanggal akuisisi citra
start_date = '2023-04-01'
end_date = '2023-08-31'
```

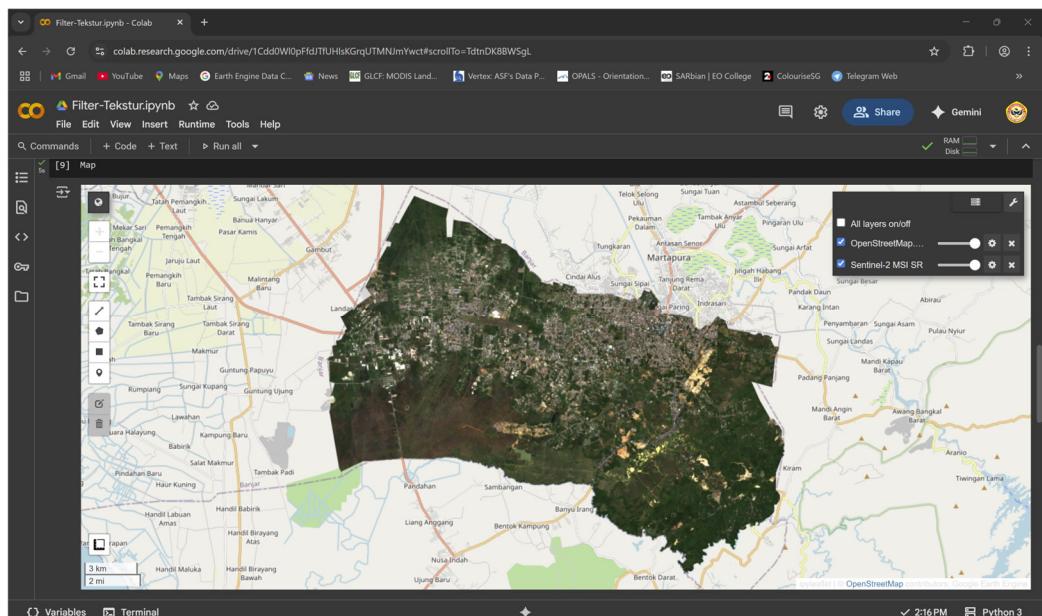
```
# Mengakses Sentinel-2 image collection
s2_col = (
    ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
        .filterDate(start_date, end_date)
        .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 50))
)
```

```
# Reduksi Sentinel-2 image collection menjadi image
s2_image = s2_col.median().clip(region)
s2_image = s2_image.select(['B2', 'B3', 'B4', 'B8', 'B11', 'B12'])
```

```
# Visualisasi citra komposit warna
rgb_vis = {
    'min': 0.0,
    'max': 2500,
    'bands': ['B4', 'B3', 'B2']
}

Map = geemap.Map()
Map.centerobject(region, 12)
Map.addLayer(s2_image, rgb_vis, 'Sentinel-2 MSI SR')
Map
```

Kode-kode di atas mengakses dan menampilkan citra Sentinel-2 sebagaimana gambar berikut:



Lanjutkan dengan mengetikkan dan menjalankan kode-kode berikut:

```
# Implementasi filter tekstur
glcm = s2_image.toInt32().glcmTexture(3)
```

```
▼ Image (108 bands)
  type: Image
  ▼ bands: List (108 elements)
    ► 0: "B2_asm", double, EPSG:4326, 2x3 px
    ► 1: "B2_contrast", double, EPSG:4326, 2x3 px
    ► 2: "B2_corr", double, EPSG:4326, 2x3 px
    ► 3: "B2_var", double, EPSG:4326, 2x3 px
    ► 4: "B2_idm", double, EPSG:4326, 2x3 px
    ► 5: "B2_savg", double, EPSG:4326, 2x3 px
    ► 6: "B2_svar", double, EPSG:4326, 2x3 px
    ► 7: "B2_sent", double, EPSG:4326, 2x3 px
    ► 8: "B2_ent", double, EPSG:4326, 2x3 px
    ► 9: "B2_dvar", double, EPSG:4326, 2x3 px
    ► 10: "B2_dent", double, EPSG:4326, 2x3 px
    ► 11: "B2_imcorr1", double, EPSG:4326, 2x3 px
    ► 12: "B2_imcorr2", double, EPSG:4326, 2x3 px
    ► 13: "B2_maxcorr", double, EPSG:4326, 2x3 px
    ► 14: "B2_diss", double, EPSG:4326, 2x3 px
    ► 15: "B2_inertia", double, EPSG:4326, 2x3 px
    ► 16: "B2_shade", double, EPSG:4326, 2x3 px
```

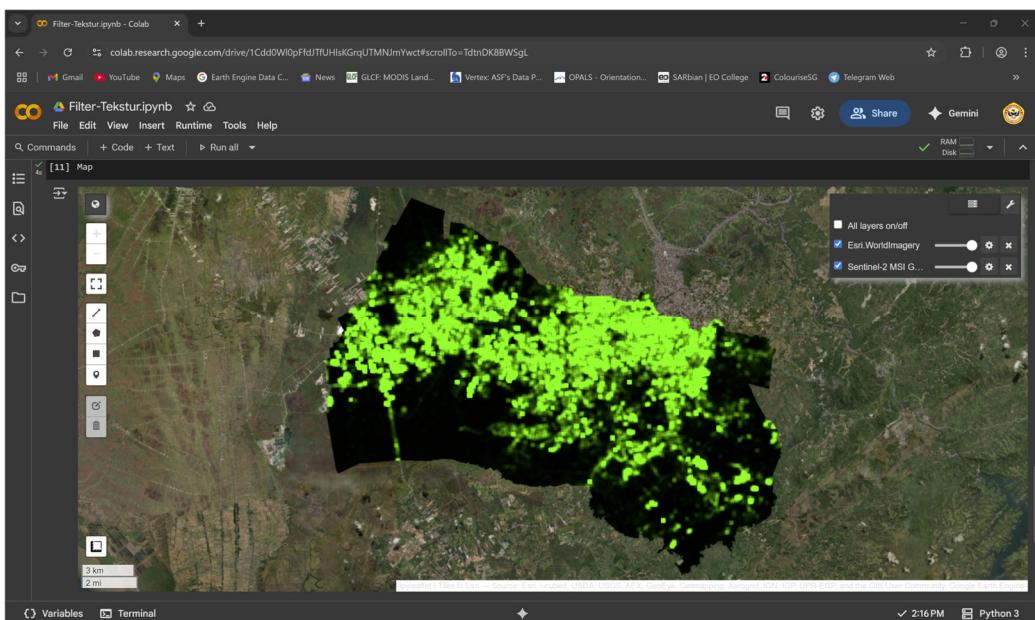
Instruksi di atas merupakan implementasi filter tekstur *Gray Level Co-occurrence Matrix* (GLCM) (Haralick et al., 1973) dengan ukuran jendela pixel (*kernel*) 3x3. Anda dapat merubah ukuran kernel ini sesuai keperluan, dan melihat perbedaan hasilnya. Hasil pemfilteran tekstur GLCM ini akan menghasilkan matriks GLCM untuk setiap band, sebagaimana terlihat pada gambar di atas. Penjelasan selengkapnya terkait matriks GLCM ini dapat dibaca secara langsung di laman <https://developers.google.com/earth-engine/apidocs/ee-image-glcmttexture>.

Sebagai contoh, pada kode-kode di bawah kita akan mengambil dan memvisualisasikan 1 matriks saja ke dalam komposit RGB, yaitu *cluster shade*. *Cluster Shade* adalah salah satu fitur tekstur statistik yang digunakan untuk mengukur kemiringan (*skewness*) dan asimetris distribusi intensitas piksel dalam citra. Ia termasuk dalam fitur orde kedua karena mempertimbangkan hubungan antara pasangan piksel. Tentu saja, Anda bebas untuk memvisualisasikan matriks-matriks lainnya sesuai keperluan.

```
# Visualisasi citra terfilter tekstur GLCM
glcm_vis = {'min': 0,
            'max': 100000,
            'band': ['B4_shade', 'B3_shade', 'B2_shade']}
}

Map = geemap.Map(basemap='HYBRID')
Map.centerObject(region, 12)
Map.addLayer(glcm, glcm_vis, 'Sentinel-2 MSI GLCM')
Map
```

Perhatikan output visualisasi matriks *cluster shade* GLCM seperti pada gambar berikut:



Hasil filter tekstur GLCM untuk matriks *cluster shade* seperti pada gambar di atas sepertinya mampu menonjolkan kenampakan urban dan lahan-lahan terbuka yang ada di Kota Banjarbaru. Silahkan Anda eksplorasi sendiri lebih lanjut, terutama matriks-matriks lainnya.

## Filter Morfologi

Filter morfologi pada citra penginderaan jauh adalah teknik pemrosesan citra berbasis struktur geometris yang digunakan untuk mengekstraksi dan menganalisis bentuk objek dalam citra, terutama dalam konteks spasial dan struktur. Teknik ini sangat berguna untuk memisahkan fitur-fitur penting seperti vegetasi, badan air, atau bangunan dari latar belakang atau *noise*. Beberapa jenis filter morfologi antara lain spectral distance, spectral gradient, spectral dilation, spectral erosion, dan sebagainya.

### Spectral Distance

Spectral distance digunakan untuk mengukur jarak spektral per pixel di antara dua citra. Biasanya teknik ini diterapkan pada citra bitemporal. Cukup banyak potensi pemanfaatan spectral distance, antara lain observasi perkembangan wilayah, pemetaan deforestasi, pemetaan area terbakar, bahkan pemetaan genangan banjir. Sebagai latihan, buat sebuah notebook baru dan beri nama misalnya **Spectral-Distance.ipynb**. Kemudian ketikkan dan eksekusi kode-kode berikut:

```
import ee, geemap
```

```
ee.Authenticate()
```

```
ee.Initialize(project='ee-geospatialulm')
```

```
# Akses Google Drive
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

### Bab III Google Earth Engine

```
# Membuka shapefile
import geopandas as gpd
path = '/content/drive/My Drive/geebook/vector/'
region_shp = gpd.read_file(path + 'Banjarbaru.shp')
```

```
# Konversi shapefile menjadi Earth Engine geometry
region = geemap.geopandas_to_ee(region_shp).geometry()
```

Anda dapat dapat mengganti wilayah dan waktu akuisisi citra jika diperlukan. Pada Latihan ini kita menggunakan wilayah administrasi Kota Banjarbaru, dan akuisisi citra pada tahun 2019 ( $t_1$ ) dan 2024 ( $t_2$ ). Silahkan ketikkan dan sesuaikan seperlunya, dan jalankan kode-kode berikut:

```
# Penentuan tanggal akuisisi citra t1
start_date_image1 = '2019-01-01'
end_date_image1 = '2019-12-31'

# Mengimpor Sentinel-2 surface reflectance
s2_col1 = (
    ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
        .filterDate(start_date_image1, end_date_image1)
        .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 50))
)
```

```
# Penentuan tanggal akuisisi citra t2
start_date_image2 = '2024-01-01'
end_date_image2 = '2024-12-31'

# Mengimpor Sentinel-2 surface reflectance
s2_col2 = (
    ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
        .filterDate(start_date_image2, end_date_image2)
        .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 50))
)
```

```
# Reduksi image collection menjadi image
s2_image1 = s2_col1.median().clip(region)
s2_image1 = s2_image1.select(['B2', 'B3', 'B4', 'B8', 'B11', 'B12'])

s2_image2 = s2_col2.median().clip(region)
s2_image2 = s2_image2.select(['B2', 'B3', 'B4', 'B8', 'B11', 'B12'])
```

Kode-kode dibawah merupakan implementasi spectral distance. Ada empat Teknik pengukuran jarak spectral yang dapat dipilih di dalam spectral distance, yaitu SAM (*Spectral Angle Mapper*), SID (*Spectral Information Divergence*), SED (*Squared Euclidean Distance*), atau EMD (*Earth Movers Distance*). Penjelasan dapat dilihat di laman <https://developers.google.com/earth-engine/apidocs/ee-image-spectraldistance>. Default-nya menggunakan metode pengukur jarak SAM, sebagaimana kode di bawah.

```
# Implementasi filter morfologi: Spectral Distance
sd = s2_image1.spectralDistance(s2_image2)
geemap.image_stats(sd, scale=10)
```

Jika Anda ingin menggunakan metode lain, misalnya SID, maka instruksinya dirubah menjadi seperti ini `sd = s2_image1.spectralDistance(s2_image2, 'sid')`. Hal ini cukup menarik, mengingat keempat teknik ini dapat menjadi potensi riset, misalnya komparasi keempat teknik pengukur jarak ini di dalam pemetaan area terbakar. Nanti akan diuji teknik yang mana yang akan menghasilkan peta area terbakar yang paling akurat.

```

▼ Object (5 properties)
  ▼ max: Object (1 property)
    distance: 1.0522735859900911
  ▼ mean: Object (1 property)
    distance: 0.14041539645471302
  ▼ min: Object (1 property)
    distance: 0.0024682309834777793
  ▼ std: Object (1 property)
    distance: 0.09271591354601941
  ▼ sum: Object (1 property)
    distance: 432042.857498669

```

Output dari kode di atas juga akan memberikan informasi statistik citra hasil spectral distance, sebagaimana terlihat pada gambar di atas. Hal itu berguna untuk visualisasi citra, sebagaimana kode di bawah:

```

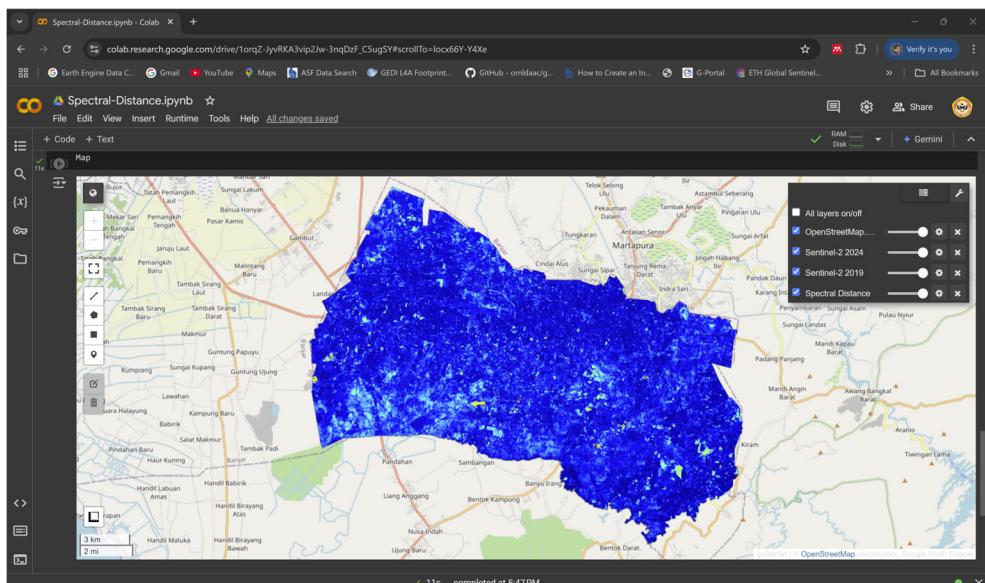
# Visualisasi citra terfilter morfologi
sd_vis = {'min': 0, 'max': 1, 'palette': 'jet'}

rgb_vis = {
  'min': 0.0,
  'max': 2000,
  'bands': ['B4', 'B3', 'B2']
}

Map = geemap.Map()
Map.centerobject(region, 12)
Map.addLayer(s2_image2, rgb_vis, 'Sentinel-2 2024')
Map.addLayer(s2_image1, rgb_vis, 'Sentinel-2 2019')
Map.addLayer(sd, sd_vis, 'Spectral Distance')
Map

```

Berikut adalah citra hasil filter spectral distance:



#### Spectral Dilation, Spectral Erosion, dan Spectral Gradient

Pembahasan tiga jenis filter morfologi, yaitu spectral dilation, spectral erosion, dan spectral gradient (Plaza et al., 2002) dijadikan satu. Sebab ketiga jenis filter ini saling memiliki keterkaitan. Spectral dilation digunakan untuk menonjolkan fitur tertentu. Sedangkan spectral erosion merupakan kebalikan dari spectral dilation, yaitu menghaluskan fitur atau menghilangkan fitur-fitur yang menonjol. Spectral gradient merupakan kombinasi keduanya, dimana secara teoritis spectral gradient akan mengukur perbedaan antara spectral dilation dan spectral erosion.

Sebagai latihan, silahkan buat sebuah notebook Google Colab baru. Kemudian beri nama misalnya **Dilation-Erosion-Gradient.ipynb**. Kemudian ketikkan dan eksekusi kode-kode berikut:

```
import ee, geemap  
  
ee.Authenticate()  
  
ee.Initialize(project='ee-geospatialulm')  
  
# Titik pusat lokasi pengamatan  
point = ee.Geometry.Point([114.711793, -2.170247])
```

Pada kode di atas, kita menunjuk sebuah titik lokasi hutan gambut yang ada di Provinsi Kalimantan Tengah. Anda dapat mengganti koordinat lokasinya sesuai keinginan Anda. Selanjutnya, lokasi akan kita buffer dengan radius 1.000 meter, dan wilayah buffer dikonversi menjadi kotak (*bounds*), dengan menggunakan kode-kode di bawah:

```
# Buffer titik pusat untuk membuat wilayah observasi  
region = point.buffer(1000).bounds()
```

Kode-kode di atas akan menghasilkan sebuah area kotak berukuran 2.000 meter x 2.000 meter, sebab radius kotak dari titik koordinat adalah 1.000 meter ke Utara, Selatan, Timur, dan Barat. Lanjutkan dengan mengetikan dan menjalankan kode-kode berikut:

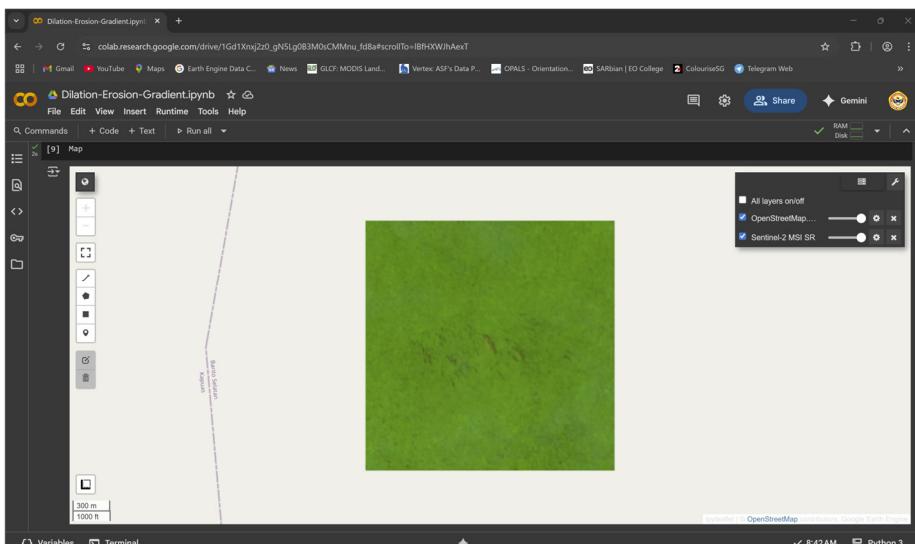
```
# Masking awan  
def mask_s2_clouds(image):  
    qa = image.select('QA60')  
    cloud_bit_mask = 1 << 10  
    cirrus_bit_mask = 1 << 11  
  
    mask = (  
        qa.bitwiseAnd(cloud_bit_mask)  
        .eq(0)  
        .And(qa.bitwiseAnd(cirrus_bit_mask).eq(0))  
    )  
  
    return image.updateMask(mask).divide(10000)  
  
# Mengakses Sentinel-2 image collection  
s2_col = (  
    ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')  
    .filterDate('2019-04-01', '2019-10-01')  
    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 50))  
    .map(mask_s2_clouds)  
)
```

```
# Memilih band dan reduksi image collection menjadi image
bands = ['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B8', 'B8A', 'B11', 'B12']
s2_image = s2_col.median().select(bands).clip(region)
```

```
# visualisasi Sentinel-2 RGB
rgb_vis = {
    'min': 0.0,
    'max': 0.5,
    'bands': ['B11', 'B8', 'B3']
}

Map = geemap.Map()
Map.centerObject(region, 15)
Map.addLayer(s2_image, rgb_vis, 'Sentinel-2 MSI SR')
```

Berikut adalah output visualisasi Citra Sentinel-2 RGB hasil eksekusi kode-kode di atas:



Untuk mengimplementasikan filter spectral dilation, ketikkan dan jalankan kode-kode berikut:

```
# Implementasi filter morfologi: spectral dilation
sd = s2_image.spectralDilation(kernel=ee.Kernel.square(5))
geemap.image_stats(sd, region=region, scale=10)
```

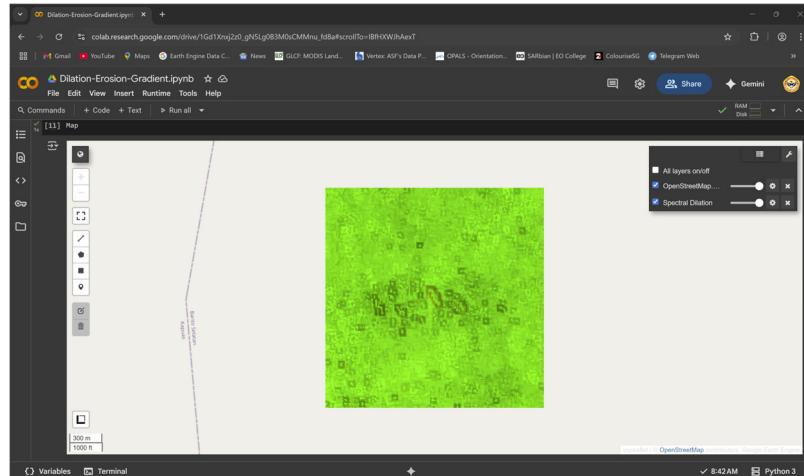
Opsi `kernel=ee.Kernel.square(5)` dapat dikosongkan (*default*), atau diganti dengan kernel-kernel lainnya, misalnya `kernel=ee.Kernel.gaussian(9)`, dengan berbagai variasi ukuran radius pemfilteran. Untuk penjelasan yang lebih lengkap, silahkan baca dokumentasi `ee.Kernel` pada laman <https://developers.google.com/earth-engine/apidocs/>.

```
# Visualisasi citra hasil filter spectral dilation
sd_vis = {'min': 0.1, 'max': 0.3, 'bands': ['B11', 'B8', 'B3']}

Map = geemap.Map()
Map.centerObject(region, 15)
Map.addLayer(sd, sd_vis, 'Spectral Dilation')
```

Berikut adalah output visualisasi citra hasil filter spectral dilation:

### Bab III Google Earth Engine

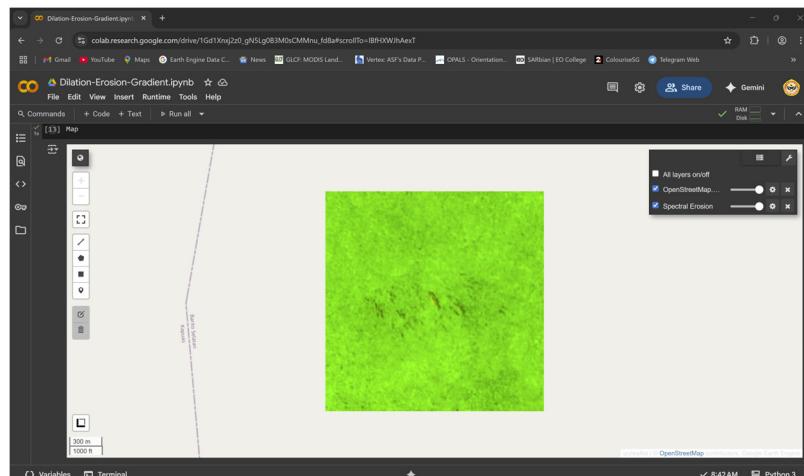


Jika dibandingkan dengan Citra Sentinel-2 original sebelum difilter, akan terlihat beberapa fitur yang visualnya semakin menonjol di atas citra hasil filter spectral dilation. Selanjutnya, untuk mengimplementasikan filter spectral erosion, ketikkan dan jalankan kode-kode berikut:

```
# Implementasi filter morfologi: spectral erosion
se = s2_image.spectralErosion(kernel=ee.Kernel.square(5))
geemap.image_stats(se, region=region, scale=10)
```

```
# Visualisasi citra hasil filter spectral erosion
se_vis = {'min': 0.1, 'max': 0.3, 'bands': ['B11', 'B8', 'B3']}
Map = geemap.Map()
Map.centerObject(region, 15)
Map.addLayer(se, se_vis, 'Spectral Erosion')
Map
```

Berikut adalah output visualisasi citra hasil filter spectral erosion:



Jika dilihat secara sepintas, mungkin tidak banyak terlihat perbedaan antara Citra Sentinel-2 original dengan hasil filter spectral dilation. Akan tetapi, jika citra dizoom akan terlihat beberapa fitur yang sebelumnya lebih dominan akan terlihat lebih kabur (*blur*) setelah difilter. Atau untuk

mendapatkan efek yang lebih nyata, Anda dapat merubah ukuran radius atau kernel filernya. Misalnya kita rubah ukuran radiusnya saja dari 5, `kernel=ee.Kernel.square(5)`, menjadi 9, `kernel=ee.Kernel.square(9)`. Silahkan Anda lihat sendiri efeknya.

Selanjutnya, untuk mengimplementasikan filter spectral gradient, secara langsung ketikkan dan jalankan kode-kode berikut:

```
# Implementasi filter morfologi: spectral gradient
sg = s2_image.spectralGradient(kernel=ee.Kernel.square(5))
geemap.image_stats(sg, region=region, scale=10)
```

Selain citra hasil filter spectral gradient, output lainnya dari kode di atas adalah nilai-nilai statistik citra spectral gradient. Sebagaimana hasil spectral dilation dan spectral erosion sebelumnya.

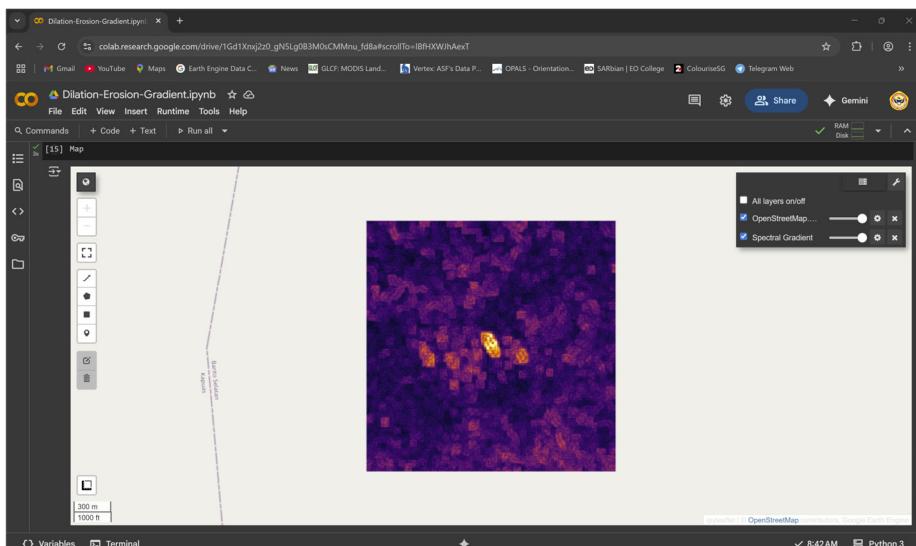
```
▼ Object (5 properties)
  ▼ max: Object (1 property)
    gradient: 0.24210697339773873
  ▶ mean: Object (1 property)
  ▼ min: Object (1 property)
    gradient: 0.013232515691967872
  ▶ std: Object (1 property)
  ▶ sum: Object (1 property)
```

Nilai statistik ini, terutama nilai maksimum dan minimumnya, akan digunakan untuk penentuan nilai '`min`' dan '`max`'. Sebagaimana kode-kode visualisasi citra spectral gradient berikut:

```
# Visualisasi citra hasil filter spectral gradient
sg_vis = {'min': 0, 'max': 0.2, 'palette': 'inferno'}

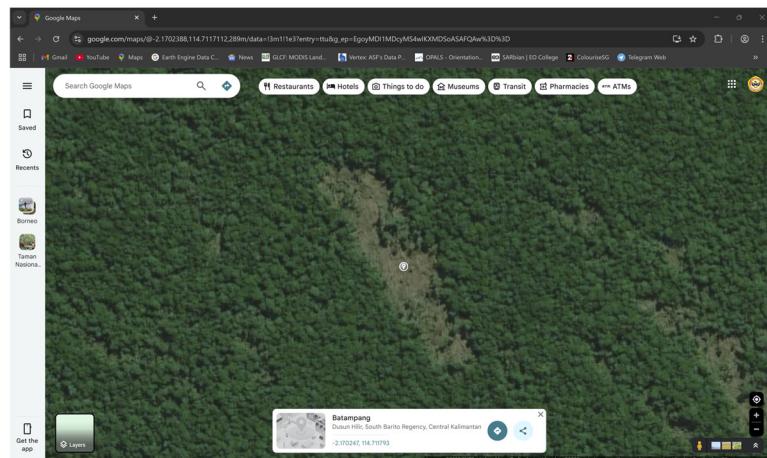
Map = geemap.Map()
Map.centerobject(region, 15)
Map.addLayer(sg, sg_vis, 'Spectral Gradient')
Map
```

Berikut adalah output visualisasi citra hasil filter spectral gradient:



### Bab III Google Earth Engine

Jika dilihat pada Google Maps, area yang paling terang pada citra hasil filter morfologi spectral gradient di atas sesungguhnya adalah deforestasi, sebagaimana terlihat pada gambar di bawah. Jika diobservasi lebih detail, terlihat seperti ada sisa-sisa potongan kayu hasil penebangan pohon (*logging*). Jika kita analisis lebih jauh, misalnya kita batasi (*thresholding*) nilai-nilai pixel spectral gradient-nya, maka area deforestasi yang cerah tersebut dapat dihitung luasannya, dan diekstrak menjadi informasi tematik deforestasi atau bahkan dikonversi menjadi poligon. Sehingga salah satu potensi aplikasi dari filter spectral gradient ini adalah untuk pemantauan deforestasi secara otomatis, terutama di hutan-hutan yang masih lebat. Kemungkinan akan lebih menarik lagi jika spectral gradient tidak diterapkan langsung pada citra mentah, melainkan pada citra hasil transformasi indeks-indeks vegetasi, seperti NDVI.



### Filter Konvolusi

Filter konvolusi adalah teknik pemrosesan citra yang digunakan untuk menerapkan transformasi spasial pada gambar, seperti penghalusan, deteksi tepi, atau peningkatan fitur tertentu. Konvolusi bekerja dengan menggeser kernel (matriks kecil berisi bobot) di atas citra dan menghitung kombinasi linear antara nilai piksel dan bobot kernel. Hasilnya adalah citra baru yang telah dimodifikasi sesuai dengan fungsi kernel. Jenis-jenis kernel yang umum digunakan di dalam filter konvolusi di antaranya adalah:

- Low-pass (penghalusan): Mengurangi noise dan detail kecil
- High-pass (deteksi tepi): Menyoroti perubahan intensitas, seperti batas wilayah
- Laplacian: Deteksi tepi isotropik
- Sobel, Prewitt, Roberts: Deteksi tepi arah tertentu
- Gaussian: Penghalusan dengan distribusi bobot berbentuk lonceng

Sebagai latihan, silahkan buat notebook Google Colab baru dan beri nama misalnya **Convolution-Filter.ipynb**. Kemudian ketikkan dan jalankan kode-kode berikut:

```
import ee, geemap  
  
ee.Authenticate()  
  
ee.Initialize(project='ee-geospatialulm')
```

```
# Membuat titik pengamatan dan buffer titik untuk menghasilkan wilayah observasi
point = ee.Geometry.Point([106.659358, -6.124270])
region = point.buffer(3000).bounds()
```

Wilayah yang akan kita observasi adalah Bandara Internasional Soekarno-Hatta. Anda dapat merubah wilayah sesuai dengan keinginan Anda.

```
# Masking awan
def mask_s2_clouds(image):
    qa = image.select('QA60')

    cloud_bit_mask = 1 << 10
    cirrus_bit_mask = 1 << 11

    mask = (
        qa.bitwiseAnd(cloud_bit_mask).eq(0)
        .And(qa.bitwiseAnd(cirrus_bit_mask).eq(0))
    )

    return image.updateMask(mask).divide(10000)
```

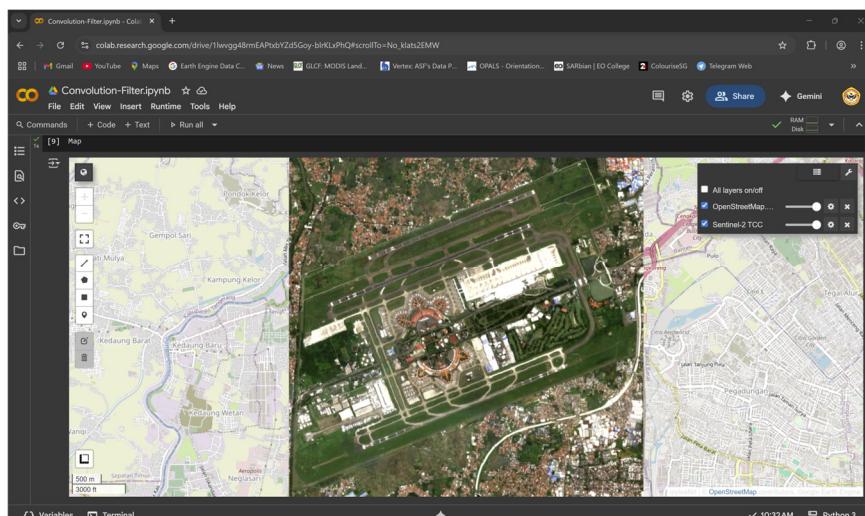
```
# Mengakses Sentinel-2 image collection
s2_col = (
    ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
    .filterDate('2022-01-01', '2022-06-01')
    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 50))
    .map(mask_s2_clouds)
)
```

```
# Reduksi Sentinel-2 image collection menjadi image
s2_image = s2_col.median().clip(region).select('B4', 'B3', 'B2')
```

```
# visualisasi Sentinel-2 True Color Composite (TCC)
rgb_vis = {'min': 0, 'max': 0.25, 'bands': ['B4', 'B3', 'B2']}

Map = geemap.Map()
Map.centerobject(region, 14)
Map.addLayer(s2_image, rgb_vis, 'Sentinel-2 TCC')
Map
```

Berikut adalah output visualisasi Citra Sentinel-2 RGB sebelum difilter:



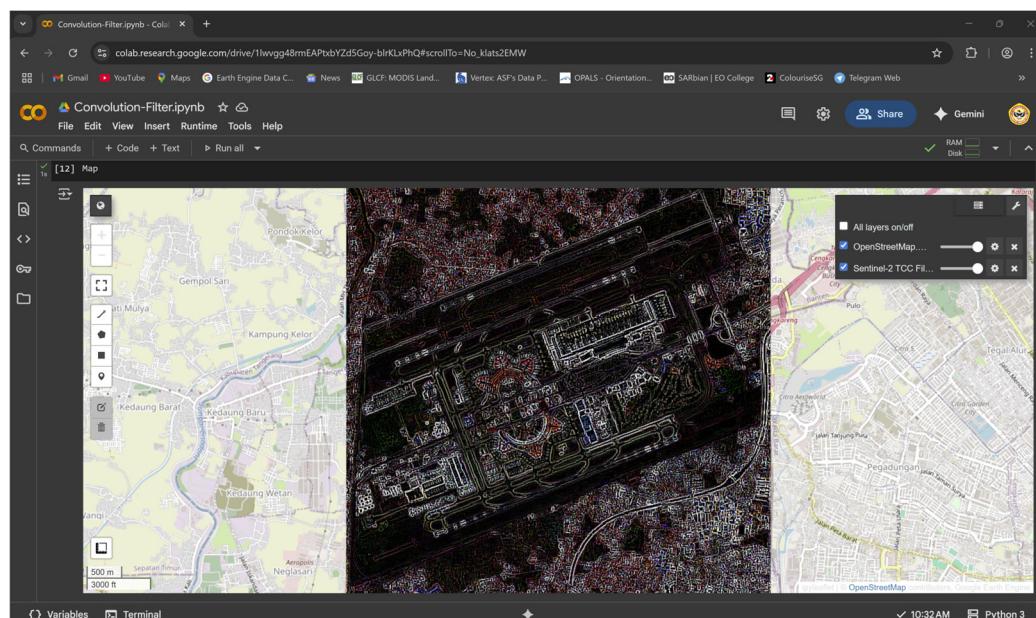
### Bab III Google Earth Engine

Untuk mengimplementasikan filter konvolusi dengan kernel Laplacian-8, ketikkan dan jalankan kode-kode berikut:

```
# Filter citra menggunakan Laplacian-8 edge-detection kernel
filter_kernel = ee.Kernel.laplacian8()
filtered_image = s2_image.convolve(filter_kernel)
```

Sebagaimana filter morfologi sebelumnya, Anda dapat merubah ukuran radius filter atau kernel filternya. Untuk penjelasan selengkapnya, silahkan baca dokumentasi ee.Kernel pada laman <https://developers.google.com/earth-engine/apidocs/>.

```
# Visualisasi citra terfilter konvolusi
Map = geemap.Map()
Map.centerObject(region, 14)
Map.addLayer(filtered_image, rgb_vis, 'Sentinel-2 TCC Filtered')
Map
```



Kernel Laplacian-8 (<https://developers.google.com/earth-engine/apidocs/ee-kernel-laplacian8>) digunakan untuk deteksi tepi (*edge detection*) objek pada citra. Hal ini biasa dilakukan ketika kita akan melakukan observasi atau bahkan interpretasi visual pada objek-objek yang sulit dilihat batas-batasnya satu sama lain di atas citra. Misalnya batas-batas antara hutan rapat, hutan sedang, dan hutan jarang. Filter Laplacian juga umum digunakan untuk menajamkan fitur-fitur garis seperti jaringan jalan atau aliran sungai.

Selanjutnya, silahkan Anda coba terapkan kernel filter lainnya yang juga digunakan untuk deteksi tepi, misalnya High-pass, Sobel, Prewitt, Roberts, dan sebagainya. Kemudian bandingkan hasilnya dengan filter Laplacian. Hal ini akan sangat menarik, sebab dapat menjadi potensi riset tersendiri nantinya. Misalnya studi kasus klasifikasi kerapatan hutan untuk survey dan pemetaan biomassa, atau pemetaan sebaran permukiman-permukiman kumuh di perkotaan. Coba juga terapkan filter-filter ini pada citra hasil transformasi NDVI misalnya.

## D. Analisis Regresi

Sesuai dengan namanya, indeks-indeks nilai spektral seperti NDVI dan yang lainnya, pada umumnya hanya memberikan informasi indeks atau urutan tinggi rendahnya suatu fitur atau fenomena kuantitatif, misalnya biomassa vegetasi. Semakin tinggi nilai NDVI misalnya, diasumsikan akan semakin tinggi biomassa vegetasi. Akan tetapi, seberapa besar kuantitas biomassa vegetasinya, secara langsung NDVI tidak dapat memberikan informasinya. Untuk keperluan ekstraksi informasi geospasial kuantitatif seperti biomassa vegetasi, biasanya diperlukan konstruksi model statistik antara indeks vegetasi seperti NDVI, dengan data faktual biomassa vegetasi hasil pengukuran di lapangan. Dan model statistik yang paling umum digunakan untuk keperluan seperti ini adalah regresi.

Model-model regresi dapat memberikan informasi matematis tentang hubungan atau korelasi antara variabel bebas (e.g. indeks vegetasi), dan variabel terikat (e.g. biomassa vegetasi). Model-model regresi yang dikonstruksi berdasarkan hasil sebuah riset, untuk selanjutnya dapat digunakan di dalam mengestimasi objek atau fenomena yang sama di wilayah lain, atau di wilayah yang sama tetapi di waktu yang berbeda. Di sini lah letak kekuatan utama model regresi, sehingga regresi mendapatkan tempat yang luas di dunia pemodelan citra penginderaan jauh.

Sebuah model regresi NDVI dan biomassa vegetasi yang dikonstruksi di hutan Kalimantan Tengah pada tahun 2024 misalnya, modelnya dapat dipakai untuk estimasi biomassa vegetasi di hutan Kalimantan Selatan, tanpa harus *ground check* ulang di Kalimantan Selatan. Tentu saja, citra satelit yang digunakan sama, dan tipe vegetasi atau hutannya diasumsikan juga sama atau mirip. Model yang sama juga dapat dipakai untuk estimasi biomassa vegetasi di hutan yang sama di Kalimantan Tengah, akan tetapi di waktu yang berbeda, misalnya estimasi biomassa vegetasi hutan pada tahun 2004. Tentu saja, citra satelit yang dipakai harus sama, atau setidaknya memiliki karakteristik yang sama.

Selain model-model regresi antara data lapangan dan indeks-indeks vegetasi, di dalam penginderaan jauh digital kita juga dimungkinkan untuk membangun model regresi antar citra. Biasanya model regresi seperti ini dibangun ketika kita melakukan *change detection*, atau deteksi perubahan fitur di atas citra. Regresi antar citra juga dapat diterapkan di dalam transformasi temporal, yaitu ketika kita menambal awan sebagaimana sudah dibahas pada bagian terdahulu. Dimana menambal awan dengan transformasi temporal model regresi secara teoritis akan lebih akurat dibandingkan dengan normalisasi citra multitemporal pada contoh sebelumnya.

Regresi antar citra juga sering dilakukan ketika kita memprediksi kehadiran sebuah fitur di masa yang akan datang, atau intensitas suatu fenomena yang akan terjadi di waktu yang akan datang. Sebagaimana hasil riset kami sebelumnya (Ali et al., 2022) ketika memprediksi intensitas kejadian kebakaran hutan dan lahan yang akan terjadi berbasis informasi biofisik vegetasi saat ini. Lebih lanjut, regresi bahkan dapat diterapkan antar *layer* geofisik yang berbeda jenis. Misalnya antara NDVI dan *Land Surface Temperature* (LST), atau antara kerapatan kanopi pohon dan LST.

### Regresi antara Data Lapangan dan Citra

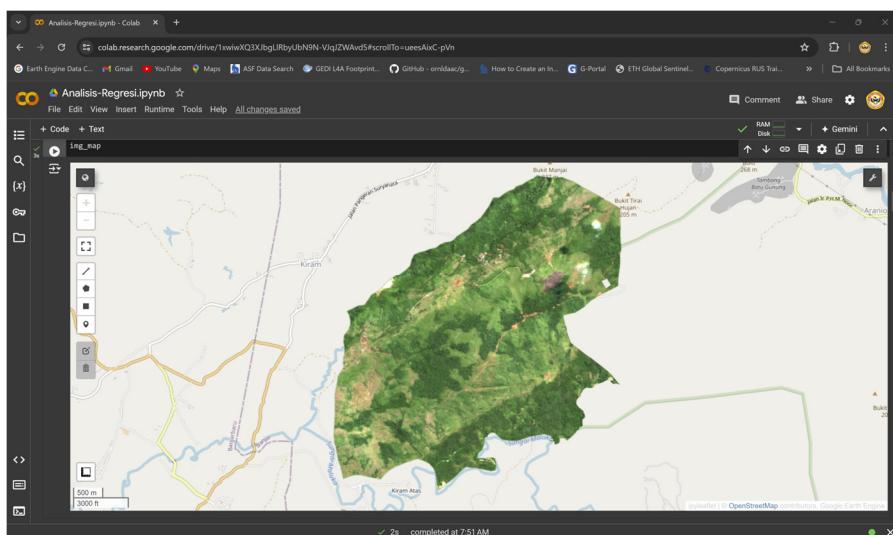
Di dalam buku ini disajikan sebuah contoh proses konstruksi model regresi antara NDVI dan biomassa pohon di atas permukaan tanah (*Above Ground Biomass (AGB)*). Dimana sampel data AGB pohon lapangan yang digunakan hanya data simulasi, bukan data hasil survey lapangan biomassa vegetasi yang sebenarnya.

### Bab III Google Earth Engine

Silahkan Anda buat sebuah notebook baru di dalam Google Colab. Beri sebuah nama yang informatif, misalnya **Analisis-Regresi.ipynb**. Kemudian jalankan kode-kode berikut:

```
import ee, geemap  
  
ee.Authenticate()  
  
ee.Initialize(project='ee-geospatialulm')  
  
# Mencari informasi Image ID Sentinel-2 MSI TOC  
s2_col = (  
    ee.ImageCollection("COPERNICUS/S2_SR_HARMONIZED")  
    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 10))  
)  
  
info_citra = (  
    ee.Image(s2_col.filterDate('2023-05-01', '2023-08-31')  
        .filterMetadata('MGRS_TILE', 'equals', '50MKB'))  
)  
info_citra  
  
# Mengakses Citra Sentinel-2 MSI TOC  
s2_image = (  
    ee.Image('COPERNICUS/S2_SR_HARMONIZED/20230808T022539_20230808T023805_T50MKB')  
    .select(['B2', 'B3', 'B4', 'B8'])  
    .divide(10000)  
)  
  
# Membuat path untuk akses ke data di dalam Google Drive  
path = '/content/gdrive/MyDrive/geebook'  
  
# Membuka shapefile  
import geopandas as gpd  
from google.colab import drive  
drive.mount('/content/gdrive')  
babaris_shp = gpd.read_file(path + '/vector/Babarism.shp')  
  
# Konversi shapefile ke Geometry GEE  
import json  
babaris_js = json.loads(babarism_shp.to_json())  
babaris_fc = ee.FeatureCollection(babarism_js)  
babaris = ee.Geometry(babarism_fc.geometry())  
  
# Memotong citra  
citra_khdtk_ulm = s2_image.clip(babarism)  
rgb_vis = {'min': 0,  
           'max': 0.1,  
           'bands': ['B4', 'B3', 'B2']}  
img_map = geemap.Map()  
img_map.centerObject(babarism, 14)  
img_map.add_layer(citra_khdtk_ulm, rgb_vis, 'Sentinel-2 KHDTK ULM')  
img_map
```

Output:



Lanjutkan dengan mengetikkan dan menjalankan kode-kode berikut:

```
# Menghitung NDVI
ndvi = citra_khdtk.ulm.normalizedDifference(['B8','B4']).rename('NDVI')
geemap.image_stats(ndvi,scale=10)
```

Output:

```
▼ Object (5 properties)
  ▼ max: Object (1 property)
    nd: 0.932088285229202
  ▼ mean: Object (1 property)
    nd: 0.7791842200503062
  ▼ min: Object (1 property)
    nd: 0.051507537688442205
  ► std: Object (1 property)
  ► sum: Object (1 property)
```

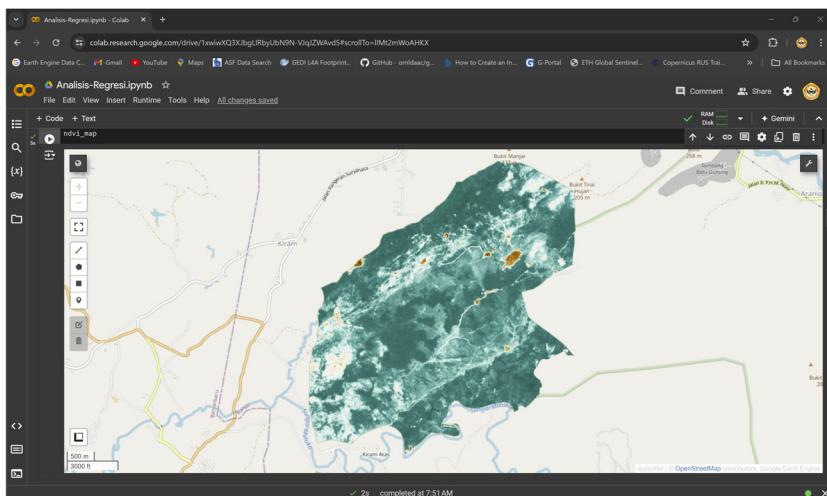
Dari output kode di atas terlihat bahwa nilai minimum NDVI adalah sekitar 0,05, dan nilai maksimumnya sekitar 0,93. Nilai-nilai ini akan menjadi referensi di dalam menentukan nilai 'min' dan 'max' untuk visualisasi citra. Perhatikan dan jalankan kode lanjutan berikut:

```
# visualisasi NDVI
ndvi_vis = {'min': 0,
            'max': 1,
            'palette': 'BrBG'}

ndvi_map = geemap.Map()
ndvi_map.centerObject(babar, 14)
ndvi_map.add_layer(ndvi, ndvi_vis, 'NDVI KHDTK ULM')
ndvi_map
```

### Bab III Google Earth Engine

Output:



Selanjutnya, ketikkan dan jalankan kode berikut:

```
# Membuka tabel data biomassa vegetasi hasil survey lapangan
import pandas as pd
ground_samples = (
    pd.read_csv(path + '/table/ground_samples.csv')
)
ground_samples
```

Output:

	PLOT_ID	CATEGORY	LAT	LONG	BIOMASS	
0	Plot 01	Validation Plot	-3.51182	114.940	96.637	
1	Plot 02	Training Plot	-3.51071	114.940	129.327	
2	Plot 03	Training Plot	-3.50907	114.942	274.912	
3	Plot 04	Training Plot	-3.50960	114.943	268.307	
4	Plot 05	Validation Plot	-3.51043	114.943	74.497	
5	Plot 06	Training Plot	-3.51066	114.944	235.203	

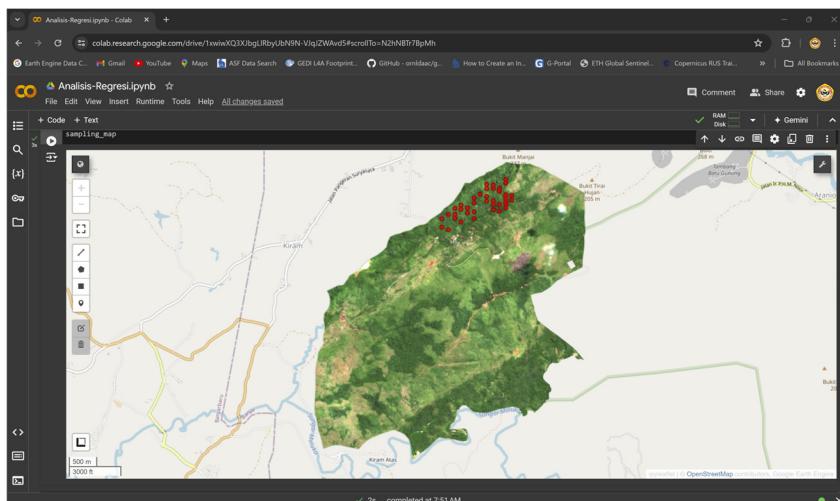
Kode di atas digunakan untuk mengakses tabel data hasil survey lapangan dalam format CSV, yaitu **ground\_samples.csv**. Selanjutnya ketikkan dan jalankan kode-kode berikut:

```
# Konversi data tabel biomassa vegetasi lapangan menjadi data titik
sampling_points = (
    gpd.GeoDataFrame(ground_samples,
                     geometry=gpd.points_from_xy(ground_samples.LONG,
                                                 ground_samples.LAT))
)
```

```
# Konversi titik sampel lapangan ke Geometry GEE
sampling_js = json.loads(sampling_points.to_json())
sampling_fc = ee.FeatureCollection(sampling_js)
sampling = ee.Geometry(sampling_fc.geometry())

# Visualisasi data titik survey lapangan
rgb_vis = {'min': 0,
            'max': 0.1,
            'bands': ['B4', 'B3', 'B2']}
sampling_vis = {
    'color': 'ff0000',
    'width': 2,
    'lineType': 'solid',
    'fillColor': '000000',
}
sampling_map = geemap.Map()
sampling_map.centerobject(babarism, 14)
sampling_map.add_layer(citra_khdtk_ulm, rgb_vis, 'sentinel-2 KHDTK ULM')
sampling_map.add_layer(sampling_fc, sampling_vis, 'Titik Sampel Lapangan')
sampling_map
```

Output:



Kode di atas digunakan untuk mengkonversi data titik sampel lapangan dalam format Pandas DataFrame menjadi geometri GEE, berdasarkan kolom **LAT** dan **LONG**. Kemudian sebaran titik-titiknya divisualisasikan bersama dengan komposit RGB Citra Sentinel-2 MSI TOC sebelumnya. Lanjutkan dengan menjalankan kode berikut:

```
# Mengekstrak nilai pixel NDVI per titik sampel
import os
out_csv = os.path.join(path + '/output/sample_analysis.csv')
geemap.extract_values_to_points(sampling_fc, ndvi, out_csv)
```

Kode di atas akan mengekstrak nilai pixel NDVI berdasarkan lokasi titik-titik sampel. Hasil ekstraksinya kemudian disimpan ke dalam bentuk file CSV (**sample\_analysis.csv**). Lanjutkan dengan menjalankan kode berikut:

### Bab III Google Earth Engine

```
# Membuka data hasil ekstraksi nilai NDVI  
sample_data = pd.read_csv(path + '/output/sample_analysis.csv')  
sample_data
```

Output:

	first	BIOMASS	CATEGORY	PLOT_ID	LAT	LONG	system:index	
0	0.871245	96.637	Validation Plot	Plot 01	-3.51182	114.940	0	
1	0.879286	129.327	Training Plot	Plot 02	-3.51071	114.940	1	
2	0.901929	274.912	Training Plot	Plot 03	-3.50907	114.942	2	
3	0.900055	268.307	Training Plot	Plot 04	-3.50960	114.943	3	
4	0.865676	74.497	Validation Plot	Plot 05	-3.51043	114.943	4	
5	0.892078	235.203	Training Plot	Plot 06	-3.51066	114.944	5	

Kode di atas berfungsi untuk mengakses data hasil ekstraksi nilai NDVI sebelumnya, yaitu file **sample\_analysis.csv**, dari dalam Google Drive. Pada tabel di atas terlihat sebuah kolom baru, yaitu kolom **first**. Kolom **first** merupakan nilai NDVI untuk setiap titik sampel. Selanjutnya, ketikkan dan jalankan kode-kode berikut:

```
# Memisahkan training samples  
train_data = (  
    sample_data[sample_data.CATEGORY.str.contains('Training Plot')])  
train_data
```

```
# Memisahkan validation samples  
val_data = (  
    sample_data[sample_data.CATEGORY.str.contains('Validation Plot')])  
val_data
```

Kode-kode di atas berfungsi untuk memisahkan antara **Training Plot** dan **Validation Plot**, berdasarkan entri-entri yang terdapat di kolom **CATEGORY**. Dimana data *training* dan *validation* ini ditentukan secara *purposive* pada saat pengambilan data lapangan. Lanjutkan dengan mengetikkan dan menjalankan kode-kode berikut:

```
# Mengambil data x dan y dari tabel  
x_train = train_data.loc[:,['first']]  
y_train = train_data.loc[:,['BIOMASS']]  
  
x_val = val_data.loc[:,['first']]  
y_val = val_data.loc[:,['BIOMASS']]
```

Kode di atas berfungsi untuk memisahkan variabel *x* dan *y*, baik untuk *training data* maupun *validation data*. Training data merupakan data yang akan digunakan untuk konstruksi model regresi, sementara validation data merupakan data yang akan digunakan untuk uji akurasi. Instruksi `x_train = train_data.loc[:,['first']]` berarti kita akan mengambil data dari kolom **first** untuk dijadikan sebagai variabel *x*. Dimana **first** sendiri merupakan nilai NDVI, sebagaimana sudah dijelaskan sebelumnya. Instruksi `y_train = train_data.loc[:,['BIOMASS']]` berarti kita akan mengambil data dari kolom **BIOMASS** untuk dijadikan sebagai variabel *y*.

## Regresi Linier Menggunakan Scikit-Learn

Masih melanjutkan kode-kode sebelumnya, silahkan jalankan kode berikut:

```
# Analisis Regresi Linier menggunakan Scikit-Learn
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import mean_absolute_percentage_error as mape

reg = LinearRegression()
reg.fit(x_train,y_train)

intercept = round(reg.intercept_[0],3)
coef = round(reg.coef_[0][0],3)

print(f'Intersep regresi: {intercept}')
print(f'Koefisien regresi: {coef}')

y_train_pred = reg.predict(x_train)
y_val_pred = reg.predict(x_val)
r2 = round(r2_score(y_train, y_train_pred),4)

print(f'Koefisien determinasi (R2): {r2}')
```

Output:

```
Intersep regresi: -2938.334
Koefisien regresi: 3524.322
Koefisien determinasi (R2): 0.7826
```

Kode di atas merupakan implementasi regresi linier tunggal menggunakan Scikit-Learn (<https://scikit-learn.org>). Scikit-Learn (Pedregosa et al., 2011) merupakan libari machine learning yang paling populer di lingkungan Python. Regresi linier Scikit-Learn akan memberikan informasi intersep, koefisien, dan koefisien determinasi ( $R^2$ ). Sebagaimana terlihat pada output di atas.

## Uji Akurasi Menggunakan MAPE dan RMSE

Model regresi biasanya diuji akurasinya menggunakan *Mean Absolute Error* (MAE), *Mean Absolute Percentage Error* (MAPE), atau *Root Mean Square Error* (RMSE). Dimana MAE, MAPE, dan RMSE akan memberikan informasi besarnya error dari model regresi yang diuji. Tentu saja, semakin kecil error-nya (semakin mendekati 0), semakin akurat modelnya. Silahkan ketikkan dan jalankan kode-kode berikut:

```
# Uji akurasi menggunakan MAPE dan RMSE
from math import sqrt

mape = round(mape(y_val, y_val_pred)*100,2)
rmse = round(sqrt(mse(y_val, y_val_pred)),2)

print(f'MAPE: {mape}%')
print(f'RMSE: {rmse} ton/ha')
```

Output:

```
MAPE: 16.6%
RMSE: 27.72 ton/ha
```

### Bab III Google Earth Engine

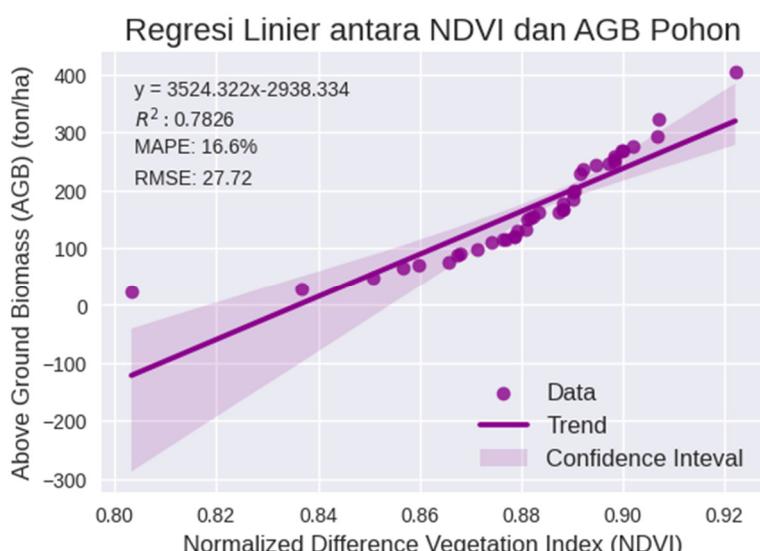
MAPE memberikan informasi error dalam satuan persen, RMSE memberikan informasi error dalam satuan yang sama dengan satuan data yang diuji. MAPE pada umumnya relatif lebih mudah dimengerti dari pada RMSE. Nilai MAPE akan 100% jika besarnya kesalahan hasil estimasi sama dengan besarnya data faktual hasil pengukuran lapangan. Secara definitif, RMSE memberikan informasi selisih antara estimasi AGB menggunakan model regresi, terhadap fakta AGB dari hasil survei lapangan. Pada output di atas terlihat bahwa selisih antara estimasi AGB dan fakta AGB di lapangan adalah sebesar 27,72 ton/hektare. Untuk dapat menilai besar kecilnya RMSE, pada umumnya nilai RMSE akan dibandingkan dengan rerata data hasil survei lapangan.

### Visualisasi Grafik Regresi

Masih melanjutkan kode di atas, silahkan ketikkan dan jalankan kode berikut:

```
# Visualisasi data hasil analisis regresi
import matplotlib.pyplot as plt
import seaborn as sns
plt.style.use('seaborn-v0_8')
fig, ax = plt.subplots(figsize=(6,4))
sns.regplot(data=sample_data, x='first', y='BIOMASS', ax=ax,
color='darkmagenta')
ax.set_xlabel('Normalized Difference Vegetation Index (NDVI)', fontsize=12)
ax.set_ylabel('Above Ground Biomass (AGB) (ton/ha)', fontsize=12)
ax.set_title('Regresi Linier antara NDVI dan AGB Pohon', fontsize=16)
ax.legend(labels=['Data', 'Trend', 'Confidence Inteval'], loc='lower right',
fontsize=12)
ax.annotate('y = ' + str(coef) + 'x' + str(intercept), xy=(0.05, 0.90),
xycoords='axes fraction', fontsize=10)
ax.annotate('$R^2$: ' + str(r2), xy=(0.05, 0.83), xycoords='axes fraction',
fontsize=10)
ax.annotate('MAPE: ' + str(mape) + '%', xy=(0.05, 0.77), xycoords='axes
fraction', fontsize=10)
ax.annotate('RMSE: ' + str(rmse), xy=(0.05, 0.70), xycoords='axes fraction',
fontsize=10)
plt.savefig(path + '/output/model_regresi.png')
```

Output:



Kode di atas merupakan visualisasi data hasil survey lapangan dan model regresi, berikut nilai  $R^2$ , MAPE, dan RMSE-nya. Ada instruksi yang baru pada kode di atas, yaitu `sns.regplot`. `sns.regplot` digunakan untuk memplotkan data AGB hasil survey lapangan ke dalam bentuk grafik xy regresi, dengan menggunakan librari Seaborn (Waskom, 2021). Instruksi `ax.annotate` digunakan untuk menambahkan teks pada plot. Teks-teks yang kita tambahkan pada plot adalah persamaan regresi,  $R^2$ , MAPE, dan RMSE. Posisi teks di dalam plot diatur menggunakan instruksi `xy=(0.05, 0.90)` dan `xycoords='axes fraction'`. Karena acuan posisi yang dipilih adalah `axes fraction`, posisi `x 0.05` berarti lokasi teks 5% dari tepi kiri plot, dan posisi `y 0.90` berarti lokasi teks 90% dari tepi bawah plot. Perhatikan juga entri '`$R^2: '$`, instruksi seperti ini digunakan untuk membuat notasi matematika, seperti  $R^2$ . Instruksi `plt.savefig(path + '/output/model_regresi.png')` digunakan untuk menyimpan plot ke dalam Google Drive dalam format PNG.

## Implementasi Model Regresi

Untuk mengimplementasikan model regresi yang sudah dihasilkan ke dalam estimasi AGB pada seluruh wilayah yang dipetakan, silahkan ketikkan dan jalankan kode berikut:

```
# Pemetaan Above Ground Biomass (AGB) Pohon
agb = (ndvi.multiply(3524.322)).subtract(2938.334).rename('AGB')
agb = agb.updateMask(agb.gt(0))
geemap.image_stats(agb, scale=10)
```

Output:

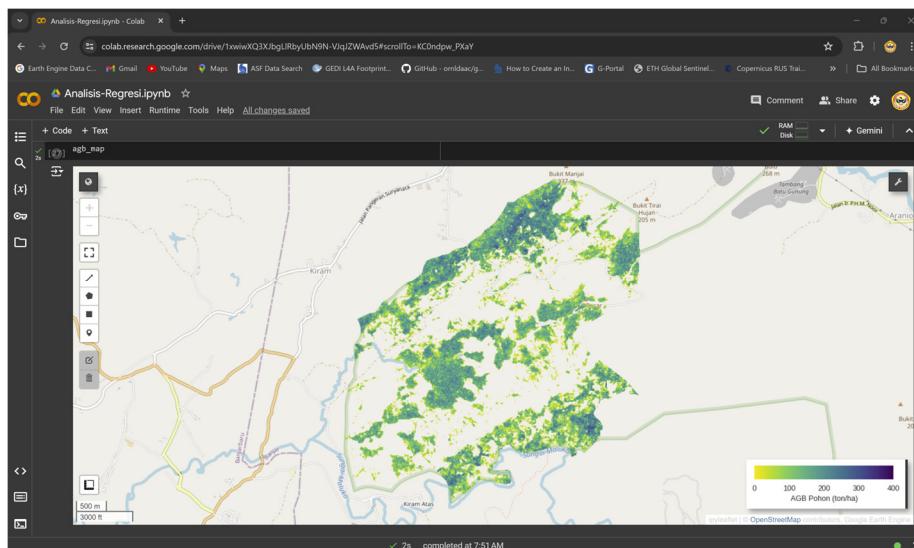
```
▼ Object (5 properties)
  ▼ max: Object (1 property)
    nd: 346.6452495755516
  ▼ mean: Object (1 property)
    nd: 126.74968435340044
  ▼ min: Object (1 property)
    nd: 0.003486867809897376
  ► std: Object (1 property)
  ► sum: Object (1 property)
```

Instruksi `agb = agb.updateMask(agb.gt(0))` digunakan untuk membuang nilai AGB negatif hasil kalkulasi model regresi. Nilai AGB negatif akan muncul pada fitur-fitur selain vegetasi pohon, seperti tubuh air, tanah terbuka, dan sebagainya. Sehingga dengan instruksi ini, nantinya hanya AGB pohon yang akan tampil di atas peta. Instruksi `geemap.image_stats(agb, scale=10)` digunakan menghitung statistik AGB hasil estimasi menggunakan regresi. Hal ini penting untuk dilakukan sebab nantinya nilai-nilai seperti minimum dan maksimum akan digunakan di dalam nilai '`min`' dan '`max`', sebagaimana kode lanjutannya berikut:

```
# Visualisasi AGB Pohon
agb_vis = {'min': 0, 'max': 400, 'palette': 'viridis_r'}
agb_map = geemap.Map()
agb_map.centerObject(babar, 14)
agb_map.add_layer(agb, agb_vis, 'AGB KHDTK ULM')
agb_map.add_colorbar(agb_vis, label='AGB Pohon (ton/ha)', layer_name='AGB KHDTK ULM')
agb_map
```

### Bab III Google Earth Engine

Output:



Kode di atas merupakan visualisasi AGB hasil estimasi menggunakan model regresi linier yang dihasilkan. Sati-satunya yang baru pada kode di atas adalah `agb_map.add_colorbar`. Dimana instruksi ini digunakan untuk menambahkan *color bar* pada visualisasi.

### Regresi antar Citra

Pada contoh kasus regresi antar citra ini kita akan memprediksi intensitas kejadian kebakaran hutan dan lahan yang akan terjadi (*fire severity prediction*). Variabel dependen (y) adalah *fire severity* atau tingkat keparahan kebakaran yang akan terjadi. Dalam hal ini, *fire severity* diwakili oleh parameter *Normalized Burn Ratio Thermal* (NBRT). Variabel independen (x) adalah NDVI, LST, dan *Normalize Difference Moisture Index* (NDMI). Seluruh informasi ini akan diekstrak menggunakan Citra Landsat-9 OLI-2/TIRS-2, dengan mengambil kasus kejadian kebakaran hutan dan lahan di Kota Banjarbaru, Kalimantan Selatan. Konstruksi model dibuat berdasarkan kejadian kebakaran hutan dan lahan pada tahun 2023. Kemudian model akan digunakan untuk memprediksi intensitas kebakaran hutan dan lahan pada tahun 2024.

NDMI diformulasikan sebagai berikut (Gao, 1996):

$$\text{NDMI} = \frac{\text{NIR} - \text{SWIR1}}{\text{NIR} + \text{SWIR1}}$$

NBRT diformulasikan sebagai berikut (Holden et al, 2005):

$$\text{NBRT} = \frac{\text{NIR} - \text{SWIR2} \left( \frac{\text{Thermal}}{1000} \right)}{\text{NIR} + \text{SWIR2} \left( \frac{\text{Thermal}}{1000} \right)}$$

*Fire severity* diformulasikan sebagai berikut:

$$\text{dNBRT} = \text{Prefire NBRT} - \text{Postfire NBRT}$$

Pada umumnya, klasifikasi *fire severity* atau *burn severity* mengacu kepada standar *United State Geological Survey* (USGS) sebagaimana Tabel 3.2. Akan tetapi, berdasarkan pengalaman, nilai interval masing-masing kelas pada standar USGS ini tidak selalu sesuai untuk semua wilayah, khususnya untuk kebakaran hutan dan lahan yang terjadi di lahan basah seperti di Kalimantan Selatan. Sehingga nilai interval kelas ini harus disesuaikan atau dimodifikasi sesuai kondisi wilayah. Kita dapat melakukan inspeksi visual di atas citra untuk mengamati area-area yang benar-benar terbakar, sebagai acuan untuk nilai interval kelas.

Tabel 3.2. *USGS burn severity level classification*

Severity Level	dNBR Range (scaled by $10^3$ )	dNBR Range (not scaled)
Enhanced Regrowth, high (post-fire)	-500 to -251	-0.500 to -0.251
Enhanced Regrowth, low (post-fire)	-250 to -101	-0.250 to -0.101
Unburned	-100 to +99	-0.100 to +0.99
Low Severity	+100 to +269	+0.100 to +0.269
Moderate-low Severity	+270 to +439	+0.270 to +0.439
Moderate-high Severity	+440 to +659	+0.440 to +0.659
High Severity	+660 to +1300	+0.660 to +1.300

Silahkan buat sebuah notebook baru di dalam Google Colab. Kemudian beri nama **Fire-Severity-Prediction.ipynb** misalnya. Selanjutnya, ketikkan dan jalankan kode-kode berikut:

```
import ee, geemap

ee.Authenticate()

ee.Initialize(project='ee-geospatialulm')

# Mengakses Google Drive dan membuka shapefile
import geopandas as gpd
from google.colab import drive

drive.mount('/content/gdrive')
path = '/content/gdrive/MyDrive/geebook'

kalsel_shp = gpd.read_file(path + '/vector/Kalsel.shp')
```

Output:

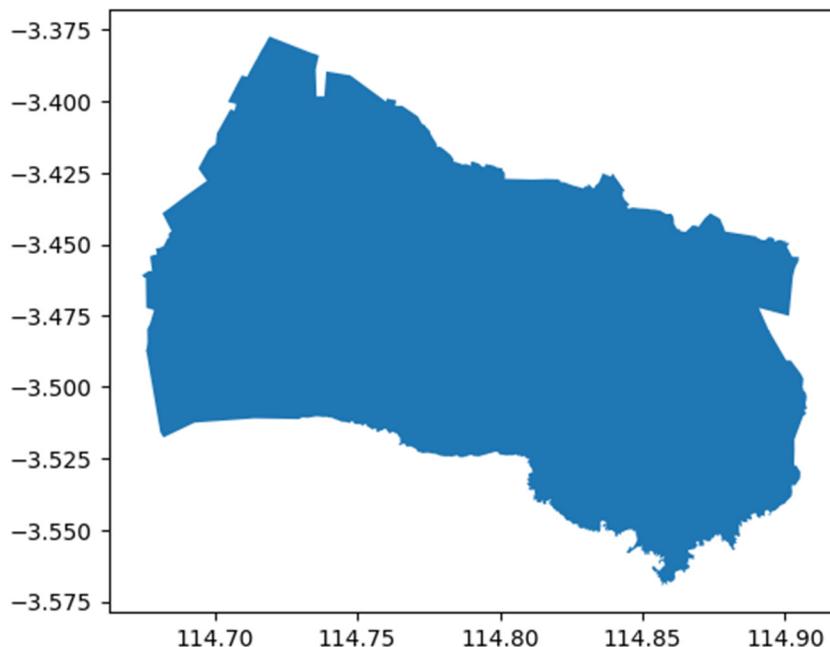
	KAB_KOTA	LUAS_KM2	geometry
0	Kabupaten Balangan	1828.1200	POLYGON ((115.69329 -2.04040, 115.69475 -2.040...
1	Kabupaten Banjar	4589.9800	POLYGON ((115.58690 -2.88613, 115.59487 -2.889...
2	Kota Banjarbaru	305.3640	POLYGON ((114.73548 -3.38905, 114.73581 -3.398...
3	Kota Banjarmasin	98.4678	POLYGON ((114.57695 -3.27186, 114.57680 -3.273...
4	Kabupaten Barito Kuala	2428.9600	POLYGON ((114.85950 -2.52651, 114.85977 -2.526...
5	Kabupaten Hulu Sungai Selatan	1697.5600	POLYGON ((115.02592 -2.50912, 115.05299 -2.518...
6	Kabupaten Hulu Sungai Tengah	1573.4100	POLYGON ((115.41692 -2.45891, 115.41928 -2.460...
7	Kabupaten Hulu Sungai Utara	907.7180	POLYGON ((115.13743 -2.29622, 115.15417 -2.303...
8	Kabupaten Kotabaru	9350.8200	MULTIPOLYGON (((115.65358 -4.94127, 115.65362 ...
9	Kabupaten Tabalong	3472.7000	POLYGON ((115.75800 -1.31491, 115.75791 -1.315...
10	Kabupaten Tanah Bumbu	4884.8700	MULTIPOLYGON (((115.58825 -3.01652, 115.58971 ...
11	Kabupaten Tanah Laut	3843.8600	MULTIPOLYGON (((114.73817 -3.51064, 114.73859 ...
12	Kabupaten Tapin	2156.7500	POLYGON ((115.00466 -2.76343, 115.00475 -2.763...

### Bab III Google Earth Engine

Kode di atas digunakan untuk mengakses wilayah administrasi Kalimantan Selatan (**Kalsel.shp**). Karena kita hanya akan memilih Kota Banjarbaru, jalankan kode berikut:

```
# Mengambil batas Kota Banjarbaru
bjb_shp = kalsel_shp.loc[(kalsel_shp['KAB_KOTA'] == 'Kota Banjarbaru')]
bjb_shp.plot()
```

Output:



Pastikan tidak ada kesalahan dalam pemilihan wilayah administrasi. Selanjutnya, ketikkan dan jalankan kode-kode berikut:

```
# Konversi shapefile ke Geometry GEE
bjb = geemap.geopandas_to_ee(bjb_shp).geometry()
```

```
# Prefire timeline
prefire_start_date = '2023-06-01'
prefire_end_date = '2023-07-31'

# Postfire timeline
postfire_start_date = '2023-09-15'
postfire_end_date = '2023-10-31'
```

Pada kode di atas, prefire merupakan saat sebelum kejadian kebakaran. Biasanya akan diambil waktu akhir musim hujan atau awal musim kemarau. Sementara postfire adalah setelah terbakar. Biasanya akan diambil waktu akhir musim kemarau. Terkait dengan penentuan tanggal ini akan sangat bergantung dengan kondisi wilayah yang dipetakan. Di wilayah Kota Banjarbaru, dan Kalimantan Selatan pada umumnya, kemarau dimulai pada Juni, Juli, atau Agustus dan akan berakhir pada awal atau pertengahan Oktober. Wilayah lain kemungkinan akan berbeda waktu.

Selanjutnya, ketikkan dan jalankan kode-kode berikut:

```
def apply_scale_factors(image):
    optical_bands = image.select('SR_B_').multiply(0.0000275).add(-0.2)
    thermal_bands = image.select('ST_B_*').multiply(0.00341802).add(149.0)

    image = (
        image.addBands(optical_bands, None, True)
        .addBands(thermal_bands, None, True)
    )

    return image
```

Kode-kode di atas digunakan untuk konversi nilai-nilai pixel Landsat-9 menjadi *surface reflectance* dan *brightness temperature*. Penjelasan selengkapnya lihat Kembali halaman 194. Selanjutnya, ketikkan dan jalankan kode-kode berikut:

```
def mask_clouds(image):
    qa = image.select('QA_PIXEL')

    cirrus_bit_mask = 1 << 2
    cloud_bit_mask = 1 << 3
    shadow_bit_mask = 1 << 4

    mask = (
        qa.bitwiseAnd(cirrus_bit_mask).eq(0)
        .And(qa.bitwiseAnd(cloud_bit_mask).eq(0))
        .And(qa.bitwiseAnd(shadow_bit_mask).eq(0))
    )

    return image.updateMask(mask)
```

Kode-kode di atas digunakan untuk masking awan Landsat-9. Penjelasan selengkapnya lihat kembali halaman 195. Selanjutnya, ketikkan dan jalankan kode-kode berikut:

```
# Prefire Landsat-9 OLI-2/TIRS-2

prefire_19_col = (
    ee.ImageCollection("LANDSAT/LC09/C02/T1_L2")
    .filterDate(prefire_start_date,prefire_end_date)
    .filterBounds(bjb)
    .filter(ee.Filter.lte('CLOUD_COVER',50))
    .map(apply_scale_factors)
    .map(mask_clouds)
)

prefire_19 = prefire_19_col.median()
prefire_19_bjb = prefire_19.clip(bjb)
```

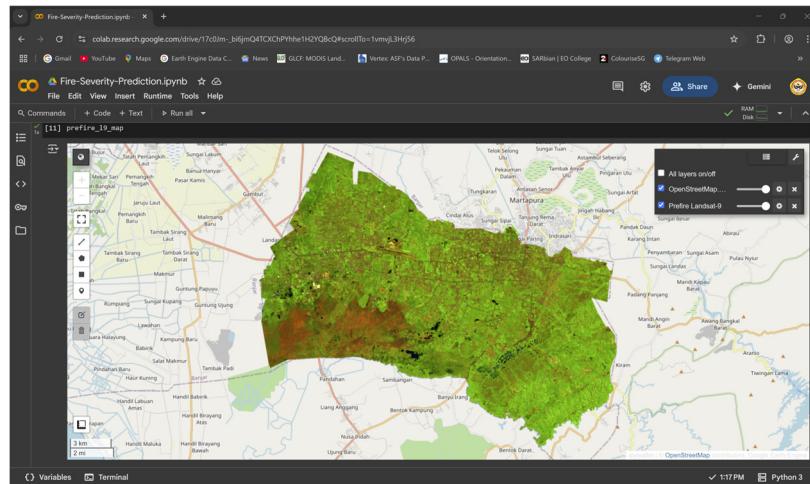
```
# Visualisasi prefire Landsat-9

rgb_vis = {'min': 0, 'max': 0.5, 'bands': ['SR_B6', 'SR_B5', 'SR_B2']}

prefire_19_map = geemap.Map()
prefire_19_map.centerObject(bjb, 12)
prefire_19_map.add_layer(prefire_19_bjb, rgb_vis, 'Prefire Landsat-9')
prefire_19_map
```

Kode-kode di atas digunakan untuk mengakses dan memvisualisasikan Citra Landsat-9 sebelum kejadian kebakaran. Berikut adalah outputnya:

### Bab III Google Earth Engine



```
# Postfire Landsat-9 OLI-2/TIRS-2

postfire_19_col = (
    ee.ImageCollection("LANDSAT/LC09/C02/T1_L2")
        .filterDate(postfire_start_date,postfire_end_date)
        .filterBounds(bjb)
        .filter(ee.Filter.lt('CLOUD_COVER', 50))
        .map(apply_scale_factors)
        .map(mask_clouds)
)

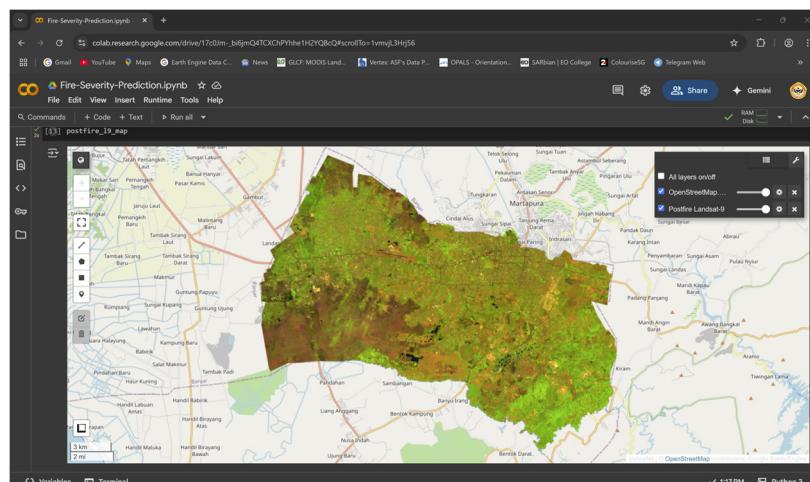
postfire_19 = postfire_19_col.median()
postfire_19_bjb = postfire_19.clip(bjb)
```

```
# visualisasi postfire Landsat-9

rgb_vis = {'min': 0, 'max': 0.5, 'bands': ['SR_B6', 'SR_B5', 'SR_B2']}

postfire_19_map = geemap.Map()
postfire_19_map.centerobject(bjb, 12)
postfire_19_map.addlayer(postfire_19_bjb, rgb_vis, 'Postfire Landsat-9')
postfire_19_map
```

Kode-kode di atas digunakan untuk mengakses dan memvisualisasikan Citra Landsat-9 setelah kejadian kebakaran. Berikut adalah outputnya:



Lanjutkan dengan mengetikkan dan menjalankan kode-kode berikut:

```
# Mengambil data band-band yang diperlukan

# Prefire bands
prefire_red = prefire_19_bjb.select('SR_B4')
prefire_nir = prefire_19_bjb.select('SR_B5')
prefire_swirl1 = prefire_19_bjb.select('SR_B6')
prefire_swirl2 = prefire_19_bjb.select('SR_B7')
prefire_tir = prefire_19_bjb.select('ST_B10')

# Postfire bands
postfire_nir = postfire_19_bjb.select('SR_B5')
postfire_swirl2 = postfire_19_bjb.select('SR_B7')
postfire_tir = postfire_19_bjb.select('ST_B10')
```

```
# Kalkulasi NDVI, NDMI, dan NBRT

# Parameter prefire
prefire_ndvi = (
    prefire_nir.subtract(prefire_red)
    .divide(prefire_nir.add(prefire_red))
).rename('PreNDVI')

prefire_ndmi = (
    prefire_nir.subtract(prefire_swirl1)
    .divide(prefire_nir.add(prefire_swirl1))
).rename('PreNDMI')

prefire_nbrt = (
    prefire_nir.subtract(prefire_swirl2)
    .multiply(prefire_tir.divide(1000))
    .divide(prefire_nir.add(prefire_swirl2)
            .multiply(prefire_tir.divide(1000)))
).rename('PreNBRT')

# Parameter postfire
postfire_nbrt = (
    postfire_nir.subtract(postfire_swirl2)
    .multiply(postfire_tir.divide(1000))
    .divide(postfire_nir.add(postfire_swirl2)
            .multiply(postfire_tir.divide(1000)))
).rename('PostNBRT')
```

```
# Kalkulasi dNBRT

dnbrt = prefire_nbrt.subtract(postfire_nbrt).rename('dNBRT')
dnbrt = dnbrt.updateMask(dnbrt.gte(0.27))
dnbrt = dnbrt.updateMask(dnbrt.lte(1.3))

geemap.image_stats(dnbrt, scale=30)
```

Kode-kode di atas diterapkan untuk kalkulasi NDVI, NDMI, NBRT, dan DNBRT, sekaligus kalkulasi nilai-nilai statistiknya. Berikut adalah output nilai-nilai statistik DNBRT:

```
▼ Object (5 properties)
  ▼ max: Object (1 property)
    DNBRT: 0.7193764150130513
  ▶ mean: Object (1 property)
  ▼ min: Object (1 property)
    DNBRT: 0.2700020985536137
  ▶ std: Object (1 property)
  ▶ sum: Object (1 property)
```

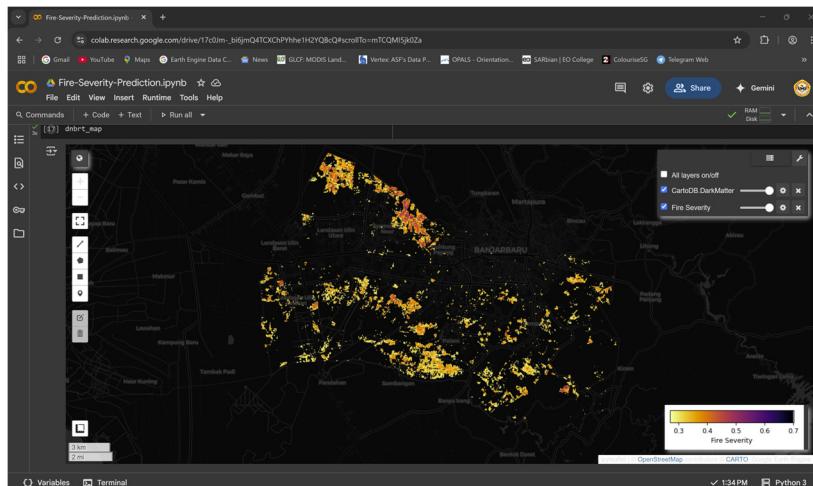
Untuk memvisualisasikan tingkat keparahan kebakaran, ketikkan dan jalankan kode-kode berikut:

### Bab III Google Earth Engine

```
# Visualisasi Fire Severity
dnbrt_vis = {'min': 0.27, 'max': 0.7, 'palette': 'inferno_r'}

dnbrt_map = geemap.Map()
dnbrt_map.centerobject(bjb, 12)
dnbrt_map.add_basemap('CartoDB.DarkMatter')
dnbrt_map.add_layer(dnbrt, dnbrt_vis, 'Fire Severity')
dnbrt_map.add_colorbar(dnbrt_vis, label='Fire Severity', layer_name='Fire Severity')
dnbrt_map
```

Berikut adalah outputnya:



Selanjutnya, untuk implementasi model regresi linier, ketikkan dan jalankan kode-kode berikut:

```
# Analisis Regresi Linier

# Masking semua variabel menggunakan NBRT
prefire_ndvi_masked = prefire_ndvi.updateMask(dnprt.gt(0))
prefire_ndmi_masked = prefire_ndmi.updateMask(dnprt.gt(0))
prefire_tir_masked = prefire_tir.updateMask(dnprt.gt(0))

# Penggabungan semua variabel
constant = ee.Image(1)
img_reg = ee.Image.cat(constant, prefire_ndvi_masked, prefire_ndmi_masked,
                      prefire_tir_masked, dnprt)

# Kalkulasi regresi
regresi_linier = img_reg.reduceRegion(
    reducer=ee.Reducer.linearRegression(numX=4, numY=1),
    scale=30,
    geometry=bjb
)
regresi_linier
```

constant = ee.Image(1) digunakan untuk membuat sebuah image dengan nilai pixel konstan yaitu 1. Image ini wajib dibuat di dalam analisis regresi linier menggunakan GEE. Image ini akan memfasilitasi kalkulasi intersep regresi. img\_reg = ee.Image.cat(constant, prefire\_ndvi\_masked, prefire\_ndmi\_masked, prefire\_tir\_masked, dnprt) digunakan untuk menggabungkan semua variabel GEE image ke dalam analisis regresi linier, termasuk variabel  $y$  (dnprt). ee.Reducer.linearRegression(numX=4, numY=1) merupakan implementasi regresi linier pada GEE image. Perhatikan (numX=4, numY=1), yang berarti terdapat 4 variabel  $x$  (termasuk constant) dan 1 variabel  $y$ .

Berikut adalah output dari kode-kode di atas:

```
▼ Object (2 properties)
  ▼ coefficients: List (4 elements)
    ▼ 0: [0.5992533178215936]
      0: 0.5992533178215936
    ▼ 1: [0.12405004467750591]
      0: 0.12405004467750591
    ▼ 2: [0.06382403667850554]
      0: 0.06382403667850554
    ▼ 3: [-0.0011934320308578162]
      0: -0.0011934320308578162
  ▼ residuals: [0.048678226519370364]
    0: 0.048678226519370364
```

Ingin kembali instruksi `constant = ee.Image(1)` sebelumnya, dengan adanya instruksi ini, maka koefisien 0 pada output di atas, yaitu `0.5992533178215936` sesungguhnya adalah intersep dari model regresi linier yang dihasilkan. Sementara sisanya merupakan koefisien dari masing-masing variabel, sesuai dengan urutan variabel di dalam instruksi penggabungan variabel, yaitu `img_reg = ee.Image.cat(constant, prefire_ndvi_masked, prefire_ndmi_masked, prefire_tir_masked, dnbrt)`. Sehingga urutan variabelnya adalah konstanta (intersep), koefisien NDVI, koefisien NDMI, dan koefisien LST. Untuk `dnbrt` merupakan variabel dependen atau variabel  $y$ -nya, sehingga tentu saja tidak memiliki koefisien. Di dalam aturan regresi GEE image, variabel yang di urutan terakhir sebagaimana `dnbrt`, selalu variabel  $y$  (variabel terikat). Selanjutnya, output model regresi seperti pada gambar di atas dapat dikonversi dan ditampilkan ke dalam bentuk teks dengan kode-kode berikut:

```
# Mengambil parameter-parameter regresi
coef_list = ee.Array(regresi_linier.get('coefficients')).toList()

intercept = round(ee.List(coef_list.get(0)).get(0). getInfo(), 6)
coef_ndvi = round(ee.List(coef_list.get(1)).get(0). getInfo(), 6)
coef_ndmi = round(ee.List(coef_list.get(2)).get(0). getInfo(), 6)
coef_lst = round(ee.List(coef_list.get(3)).get(0). getInfo(), 6)

residual = (
  round(ee.Array(regresi_linier.get('residuals'))
    .toList().get(0).getInfo(), 3)
)

# Tampilkan parameter-parameter regresi
print(f'Intersep: {intercept}')
print(f'Koefisien NDVI: {coef_ndvi}')
print(f'Koefisien NDMI: {coef_ndmi}')
print(f'Koefisien LST: {coef_lst}')
print(f'Residual error: {residual}')
print(f'Model regresi: y = {intercept} + {coef_ndvi}NDVI + {coef_ndmi}NDMI + {coef_lst}LST')
```

Instruksi `ee.Array(regresi_linier.get('coefficients')).toList()` pada kode di atas digunakan untuk mengkonversi output regresi menjadi array GEE. Perhatikan juga instruksi `round(ee.List(coef_list.get(0)).get(0). getInfo(), 6)`, instruksi ini digunakan untuk mengambil informasi (dan merubahnya menjadi teks) koefisien yang pertama (ke-0), yaitu intersep dari model regresi. Sekaligus membulatkan bilangan desimalnya menjadi 6 angka di belakang koma. Demikian seterusnya dengan instruksi-instruksi yang sama di bawahnya, sesuai urutan variabel. Berikut adalah output selengkapnya:

### Bab III Google Earth Engine

```
Intersep: 0.599253
Koefisien NDVI: 0.12405
Koefisien NDMI: 0.063824
Koefisien LST: -0.001193
Residual error: 0.049
Model regresi: y = 0.599253 + 0.12405NDVI + 0.063824NDMI + -0.001193LST
```

Selanjutnya, dengan menggunakan model regresi yang dihasilkan di atas, kita dapat memprediksi tingkat keparahan kejadian kebakaran hutan dan lahan yang akan terjadi di masa yang akan datang. Misalnya, kita akan memprediksi kejadian kebakaran pada musim kemarau 2024. Sebab model regresi di atas dibangun berdasarkan data historis kejadian kebakaran hutan dan lahan pada tahun 2023. Silahkan ketikkan dan eksekusi kode-kode berikut:

```
# Tanggal citra tahun 2024
start_date = '2024-06-01'
end_date = '2024-07-31'

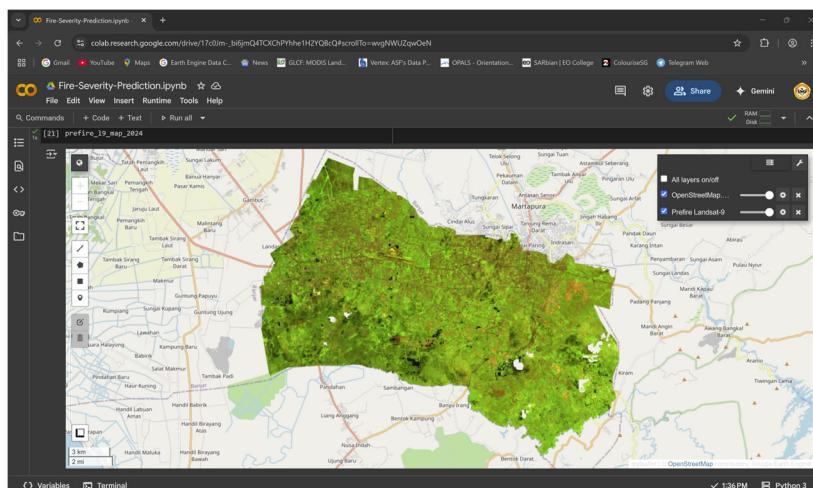
# Prefire Landsat-9 OLI-2/TIRS-2 2024
prefire_19_col_2024 = (
    ee.ImageCollection("LANDSAT/LC09/C02/T1_L2")
    .filterDate(start_date, end_date)
    .filterBounds(bjb)
    .filter(ee.Filter.lte('CLOUD_COVER', 50))
    .map(apply_scale_factors)
    .map(mask_clouds)
)

prefire_19_2024 = prefire_19_col_2024.median()
prefire_19_bjb_2024 = prefire_19_2024.clip(bjb)
```

```
# Visualisasi prefire Landsat-9 2024
rgb_vis = {'min': 0, 'max': 0.5, 'bands': ['SR_B6', 'SR_B5', 'SR_B2']}

prefire_19_map_2024 = geemap.Map()
prefire_19_map_2024.centerobject(bjb, 12)
prefire_19_map_2024.add_layer(prefire_19_bjb_2024, rgb_vis, 'Prefire Landsat-9')
prefire_19_map_2024
```

Output:



Output di atas memperlihatkan kondisi Citra Sentinel-2 menjelang musim kemarau tahun 2024. Berikutnya, ketikkan dan jalankan kode-kode berikut:

```
# Mengambil data band-band yang diperlukan untuk tahun 2024
prefire_red_2024 = prefire_19_bjb_2024.select('SR_B4')
prefire_nir_2024 = prefire_19_bjb_2024.select('SR_B5')
prefire_swir1_2024 = prefire_19_bjb_2024.select('SR_B6')
prefire_swir2_2024 = prefire_19_bjb_2024.select('SR_B7')
prefire_tir_2024 = prefire_19_bjb_2024.select('ST_B10')

# Kalkulasi NDVI dan NDMI tahun 2024
prefire_ndvi_2024 = (
    (prefire_nir_2024.subtract(prefire_red_2024))
    .divide(prefire_nir_2024.add(prefire_red_2024))
).rename('PRENDVI')
prefire_ndmi_2024 = (
    (prefire_nir_2024.subtract(prefire_swir1_2024))
    .divide(prefire_nir_2024.add(prefire_swir1_2024))
).rename('PRENDMI')

# Prediksi Fire Severity tahun 2024 menggunakan intersep dan koefisien regresi
pred_dnbrt_2024 = (
    (prefire_ndvi_2024.multiply(coef_ndvi)
     .add(prefire_ndmi_2024.multiply(coef_ndmi))
     .add(prefire_tir_2024.multiply(coef_tst)))
    .add(intercept)
).rename('DNBRT')
```

Kode-kode di atas merupakan implementasi model regresi pada citra tahun 2024.

```
# Masking Fire Severity tahun 2024
pred_dnbrt_2024 = pred_dnbrt_2024.updateMask(pred_dnbrt_2024.gte(0.27))
pred_dnbrt_2024 = pred_dnbrt_2024.updateMask(pred_dnbrt_2024.lte(1.3))
geemap.image_stats(pred_dnbrt_2024, scale=30)
```

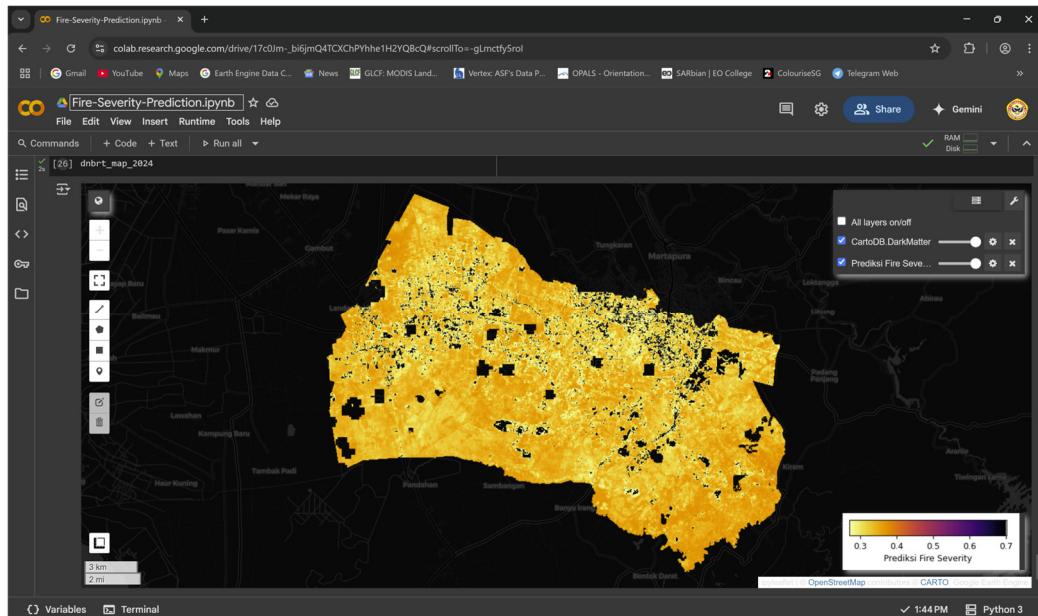
```
▼ Object (5 properties)
  ▼ max: Object (1 property)
    DNBRT: 0.42182775289801266
  ▶ mean: Object (1 property)
  ▼ min: Object (1 property)
    DNBRT: 0.12713621185766444
  ▶ std: Object (1 property)
  ▶ sum: Object (1 property)
```

```
# visualisasi Prediksi Fire Severity tahun 2024
dnbrt_vis_2024 = {'min': 0.27, 'max': 0.7, 'palette': 'inferno_r'}

dnbrt_map_2024 = geemap.Map()
dnbrt_map_2024.centerobject(bjb, 12)
dnbrt_map_2024.add_basemap('CartoDB.DarkMatter')
dnbrt_map_2024.add_layer(pred_dnbrt_2024, dnbrt_vis_2024, 'Prediksi Fire Severity')
dnbrt_map_2024.add_colorbar(dnbrt_vis_2024, label='Prediksi Fire Severity',
                           layer_name='Prediksi Fire Severity')
dnbrt_map_2024
```

Berikut adalah output hasil prediksi tingkat keparahan kebakaran yang akan terjadi tahun 2024:

### Bab III Google Earth Engine



Perhatikan output prediksi fire severity seperti pada gambar di atas. Jika outputnya tidak seperti yang diharapkan, mungkin karena waktu atau kondisi citra yang digunakan untuk membuat prediksi berbeda jauh dengan citra *prefire* yang digunakan untuk konstruksi model. Maksudnya, pada saat membangun model, citra *prefire* diambil pada bulan Juni, pada saat prediksi, citra diambil di waktu yang lebih awal, misalnya April. Perbedaan bulan/musim akan menyebabkan perbedaan kondisi yang jauh pada citra, sebab kondisi cuaca pada umumnya akan berbeda. Idealnya, prediksi dilakukan pada saat awal-awal musim kemarau di wilayah tersebut. Untuk wilayah Kalimantan Selatan biasanya awal musim kemarau adalah Juni, Juli, atau Agustus.

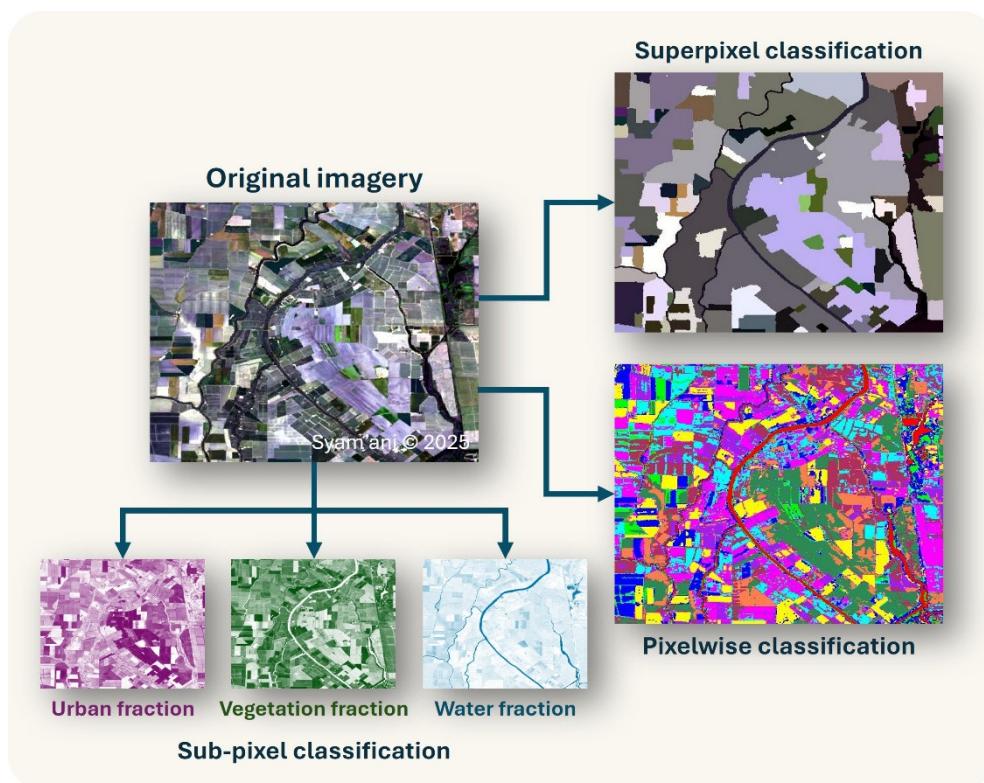
Pada kenyataannya, model yang dikembangkan seperti pada contoh di atas digunakan untuk memprediksi kejadian kebakaran hutan dan lahan beberapa hari yang akan datang. Sehingga direkomendasikan prediksi dilakukan secara *realtime* dan *up to date* ketika tersedia akuisisi citra terbaru. Misalnya, kita sudah melakukan prediksi pada awal Juni, kemudian diperbaharui lagi pada pertengahan Juni, kemudian diperbaharui lagi pada awal Juli, demikian seterusnya setiap ada akuisisi citra terbaru, hingga berakhirnya musim kemarau di tahun tersebut. Tujuannya agar selalu diperoleh informasi terkini prediksi kebakaran hutan dan lahan yang akan terjadi beberapa hari ke depan. Hal ini juga mengingat bahwa kondisi cuaca akan selalu berubah. Jika hari ini turun hujan misalnya, kemungkinan beberapa hari ke depan tidak akan terjadi kebakaran, sebab *vegetation moisture* akan naik dan LST akan turun. Sebaliknya, jika beberapa minggu ke depan semakin panas dan tidak ada turun hujan sama sekali, kelembaban vegetasi akan turun dan LST akan naik. Dampaknya memperbesar potensi kejadian kebakaran hutan dan lahan.

Tentu saja, parameter NDVI, NDMI, dan LST yang digunakan di dalam kode di atas hanya sebagai contoh, dan bukan berdasarkan hasil riset. Untuk model dan hasil yang lebih realistik, Anda dapat mengacu kepada hasil riset yang kredibel dan sesuai dengan kondisi wilayah setempat. Bagi Anda para pakar yang berkompetensi di bidang kebakaran hutan dan lahan, Anda dapat menambahkan parameter-parameter biofisik lain yang dianggap memiliki kontribusi terhadap kebakaran hutan dan lahan. Misalnya curah hujan, temperatur udara, topografi, kedekatan dengan aksesibilitas manusia, penggunaan lahan, data historis kejadian kebakaran hutan dan lahan, dan sebagainya.

## E. Klasifikasi Multispektral

Klasifikasi multispektral tergolong klasifikasi berbasis pixel (*pixelwise classification*), atau kadang juga disebut sebagai *hard classification* (Campbell and Wynne, 2011). Hal ini karena setiap pixel pada citra akan ditunjuk ke dalam kelas tertentu. Selain *pixelwise classification* juga ada yang namanya *superpixel classification* dan *sub-pixel classification*. Superpixel adalah segmen kecil dalam citra yang terdiri dari pixel-pixel dengan karakteristik serupa (warna, tekstur, lokasi, dan sebagainya). *Superpixel classification* disebut juga sebagai *Object Based Image Analysis* (OBIA) atau *image segmentation*, yang akan dibahas pada bagian selanjutnya di dalam buku ini.

*Sub-pixel classification* yang tergolong sebagai *soft classification* (Campbell and Wynne, 2011), nantinya juga akan dibahas tersendiri pada bagian akhir buku ini. *Sub-pixel classification* berasumsi bahwa setiap pixel dapat diklasifikasikan ke dalam beberapa fitur yang dikenal sebagai *endmember*. *Endmember* akan memiliki rasio pada setiap pixel, yang dikenal sebagai fraksi (*fraction*), sesuai proporsi kehadiran setiap *endmember* di dalam setiap pixel. Untuk lebih jelasnya terkait perbedaan antara *pixelwise classification*, *superpixel classification*, dan *sub-pixel classification*, dapat dilihat pada Gambar 3.2.



Gambar 3.2. Jenis klasifikasi digital citra penginderaan jauh menurut perspektif pixel

Secara sederhana, *superpixel classification* berarti menggabungkan beberapa pixel ke dalam satu kelas. *Pixelwise classification* adalah mengklasifikasikan setiap pixel hanya ke dalam satu kelas. Sedangkan *sub-pixel classification* adalah “mengurai” atau “memecah” setiap pixel, sehingga setiap pixel dapat dimasukkan ke dalam lebih dari satu kelas. Pemanfaatan ketiga jenis klasifikasi ini akan sangat bergantung kepada keperluan kita nantinya.

## Klasifikasi Multispektral Terselia

Klasifikasi multispektral terselia atau terbimbing (*supervised multispectral classification*) merupakan teknik *pixelwise classification* dengan mengacu kepada sampel-sampel objek yang sudah dibuat terlebih dahulu. Konsekuensinya, kita harus memiliki pengetahuan tentang interpretasi objek di atas citra. Hal ini untuk pembuatan sampel-sampel (*training area*) objek seperti penutupan lahan. Interpretasi atau pengenalan objek di atas citra dapat dibantu dengan citra yang memiliki resolusi spasial yang lebih tinggi, atau Google Maps. Teknik yang lebih akurat lagi tentu saja adalah dengan survey lapangan secara langsung. Sampel-sampel klasifikasi multispektral biasanya dibuat dalam bentuk poligon.

Sebagai latihan, buat sebuah notebook Google Colab baru, beri nama misalnya **Multispectral-Classification.ipynb**. Kemudian ketikkan dan eksekusi kode-kode berikut:

```
import ee, geemap  
  
ee.Authenticate()  
  
ee.Initialize(project='ee-geospatialulm')  
  
import geopandas as gpd  
import json  
from google.colab import drive  
  
# Mengakses Google Drive  
drive.mount('/content/gdrive', force_remount=True)  
  
# Membuka shapefile  
path = '/content/gdrive/MyDrive/Geopython/vector/'  
tambang_ulang = gpd.read_file(path + 'Tambang_Ulang.shp')  
  
# Konversi shapefile menjadi Earth Engine geometry  
tambang_ulang_js = json.loads(tambang_ulang.to_json())  
tambang_ulang_fc = ee.FeatureCollection(tambang_ulang_js)  
tambang_ulang_region = ee.Geometry(tambang_ulang_fc.geometry())  
  
# Mengakses Sentinel-2 image collection  
imageCollection = (  
    ee.ImageCollection('COPERNICUS/S2_HARMONIZED')  
    .filterDate('2019-06-01', '2019-07-31')  
    .filter(ee.Filter.lte('CLOUDY_PIXEL_PERCENTAGE', 5))  
)  
  
# Reduksi image collection menjadi image  
image = (  
    imageCollection.median()  
    .select(['B2','B3','B4','B8','B11','B12'])  
)  
  
# Memotong Sentinel-2 menggunakan wilayah observasi  
s2_image = image.clip(tambang_ulang_region)
```

Di dalam latihan ini, data training area sudah disediakan dalam bentuk shapefile, yang merupakan hasil digitasi menggunakan perangkat lunak Sistem Informasi Geografis (SIG). Tentu saja dalam aplikasi di dunia nyata nantinya, Anda harus menyediakan training area sendiri. Silahkan ketikkan dan jalankan kode-kode berikut untuk membuka shapefile training area:

```
# Membuka shaefile validation area
validation_area = gpd.read_file(path + 'validation_area.shp')
validation_area
```

	<b>Id</b>	<b>Landcover</b>	<b>geometry</b>
0	1	Urban	POLYGON ((114.70675 -3.66147, 114.7071 -3.6623...
1	1	Urban	POLYGON ((114.68704 -3.66786, 114.68766 -3.668...
2	1	Urban	POLYGON ((114.68832 -3.66803, 114.68912 -3.668...
3	1	Urban	POLYGON ((114.69572 -3.67282, 114.69661 -3.673...
4	1	Urban	POLYGON ((114.69992 -3.65772, 114.70096 -3.657...
5	2	Barelands	POLYGON ((114.69292 -3.65979, 114.69452 -3.660...

Perhatikan atribut shapefile training area di atas. Terdapat kolom **Id** dan **Landcover**, yang masing-masing berisi kode-kode dan kelas-kelas penutupan lahan. Jika diperlukan, kita juga dapat menyediakan sampel-sampel objek untuk validation area. Jika training area merupakan sampel untuk proses klasifikasi, validation area adalah sampel untuk uji akurasi hasil klasifikasi. Silahkan ketikkan dan jalankan kode-kode berikut untuk membuka shaefile validation area:

```
# Membuka shaefile validation area
validation_area = gpd.read_file(path + 'validation_area.shp')
validation_area
```

	<b>Id</b>	<b>Landcover</b>	<b>geometry</b>
0	1	Urban	POLYGON ((114.69838 -3.65768, 114.69937 -3.657...
1	1	Urban	POLYGON ((114.69795 -3.6599, 114.69896 -3.6601...
2	1	Urban	POLYGON ((114.70559 -3.66302, 114.70601 -3.663...
3	1	Urban	POLYGON ((114.69168 -3.66921, 114.69224 -3.669...
4	1	Urban	POLYGON ((114.68771 -3.66569, 114.68827 -3.665...
5	2	Barelands	POLYGON ((114.6717 -3.67489, 114.67241 -3.6761...

Baik training area maupun validation area, keduanya dalam bentuk shapefile polygon, dan memiliki sistem koordinat geografis (EPSG:4326). Selanjutnya, shapefile harus dikonversi menjadi feature collection GEE. Silahkan ketikkan dan jalankan kode-kode berikut untuk mengkonversi shapefile training area dan validation area menjadi feature collection:

```
# Konversi shapefile training area menjadi feature collection
training_js = json.loads(training_area.to_json())
training_fc = ee.FeatureCollection(training_js)
```

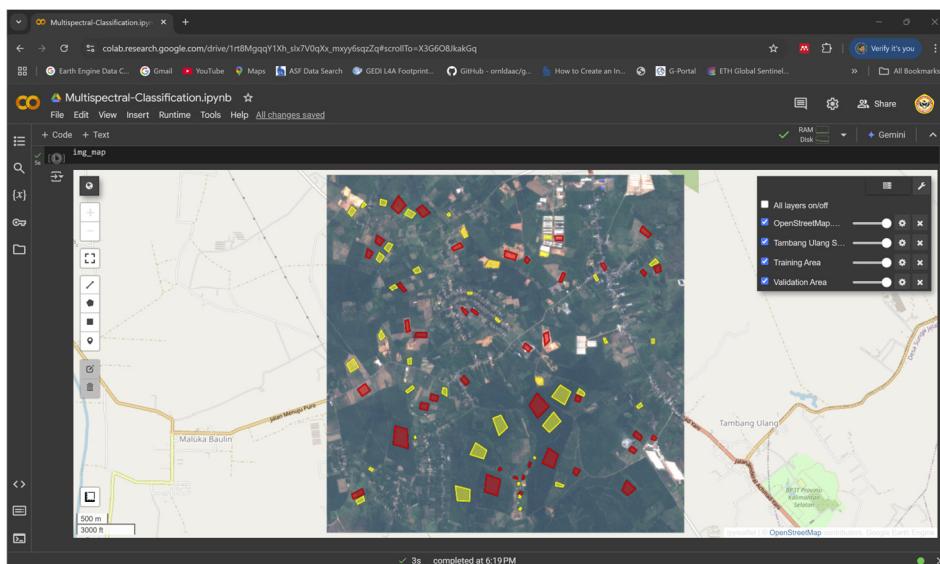
### Bab III Google Earth Engine

```
# Koversi shapefile validation area menjadi feature collection
validation_js = json.loads(validation_area.to_json())
validation_fc = ee.FeatureCollection(validation_js)
```

Jika diperlukan, kita juga dapat memvisualisasikan training area dan validation area ke atas citra yang akan diklasifikasikan, hal ini untuk evaluasi lokasi-lokasi sampel.

```
# Visualisasi Citra Sentinel-2 RGB
rgb_vis = {'min': 0, 'max': 2000, 'bands': ['B4','B3','B2']}
img_map = geemap.Map()
img_map.centerObject(tambang_ulang_region, 14)
img_map.add_layer(s2_image, rgb_vis, 'Tambang Ulang Sentinel-2 MSI RGB')
img_map.add_layer(training_fc, {'color': 'ff0000'}, 'Training Area')
img_map.add_layer(validation_fc, {'color': 'ffff00'}, 'Validation Area')
img_map
```

Berikut adalah output visualisasi training area dan validation area:



Langkah berikutnya adalah seleksi band-band Sentinel-2 yang akan dilibatkan di dalam klasifikasi.

```
# Seleksi band untuk klasifikasi
bands = ['B2','B3','B4','B8','B11','B12']
```

```
# Pembuatan training set
training_set = s2_image.select(bands).sampleRegions(**{
    'collection': training_fc,
    'properties': ['Id'],
    'scale': 10
})
```

Perhatikan kode di atas, `training_fc` Adalah shapefile yang sudah dikonversi menjadi feature collection, `scale` menunjukkan resolusi spasial. Kemudian `properties` yang akan dijadikan sebagai dasar klasifikasi adalah kolom `Id`, bukan `Landcover`. Aturannya adalah, bahwa properti untuk klasifikasi multispektral harus bilangan integer (bilangan bulat). Sehingga

kelas-kelas penutupan lahan (atau apapun objek yang akan kita klasifikasikan), harus diberi kode-kode unik di kolom tersendiri, untuk masing-masing kelas dengan menggunakan bilangan integer. Bilangannya wajib unik untuk setiap kelas, tidak harus berurutan, dan tidak harus dimulai dari 0 atau 1. Dan nama kolomnya juga tidak harus Id, Anda boleh menggunakan nama kolom lain, misalnya Kode.

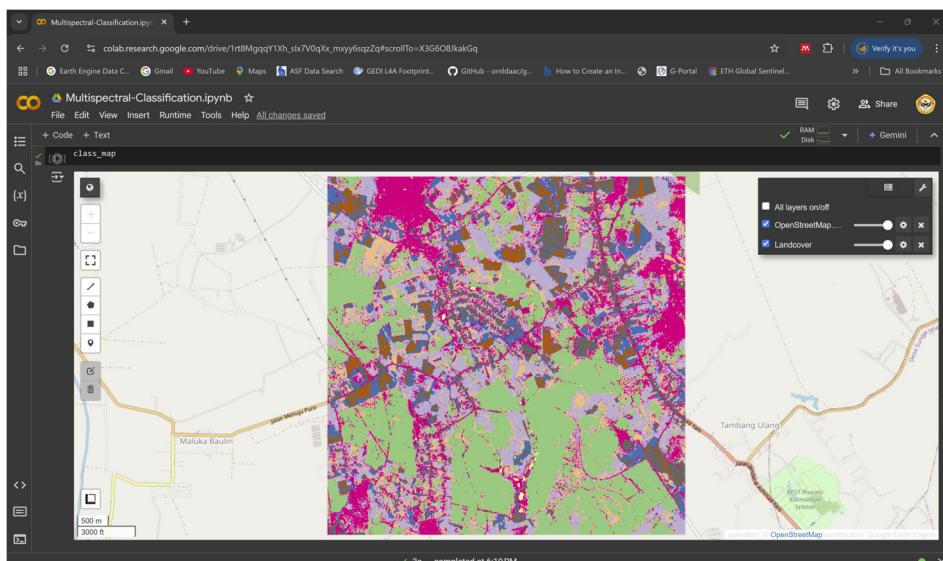
Langkah berikutnya adalah implementasi klasifikasi multispektral terselia. Ada banyak metode atau algoritma klasifikasi multispektral terselia. Di dalam pelatihan ini, kita akan menggunakan algoritma *machine learning* yang cukup populer, yaitu Random Forest (Ho, 1995; Breiman, 2001). Untuk implementasi Random Forest silahkan ketikkan dan eksekusi kode-kode berikut:

```
# Implementasi algoritma klasifikasi Random Forest
classifier = (
    ee.Classifier.smileRandomForest(256)
        .train(training_set, 'Id', bands)
)
class_image = s2_image.select(bands).classify(classifier)
```

Pada kode di atas, 256 adalah *number of trees* di dalam Random Forest, penjelasannya silahkan baca di <https://developers.google.com/earth-engine/apidocs/ee-classifier-smilerandomforest>. `.train(training_set, 'Id', bands)` merupakan instruksi untuk mengambil training set yang sudah dibuat pada kode-kode sebelumnya. Langkah berikutnya adalah visualisasi citra hasil klasifikasi dengan kode-kode berikut:

```
# Visualisasi citra hasil klasifikasi multispektral
class_vis = {'min': 1, 'max': 8, 'palette': 'Accent_r'}
class_map = geemap.Map()
class_map.centerObject(tambang_ulang_region, 14)
class_map.add_layer(class_image, class_vis, 'Landcover')
class_map
```

Berikut adalah outputnya:



### Bab III Google Earth Engine

Klasifikasi multispektral di lingkungan GEE akan menghasilkan data dalam bentuk GEE image, dengan nilai-nilai pixelnya adalah kode-kode unik dari masing-masing kelas penutupan lahan. Untuk mengkonversi image ini menjadi shapefile, silahkan lihat pembahasan Analisis Citra Berbasis Objek pada halaman 274. Langkah praktis, Anda juga dapat mengekspor image hasil klasifikasi tersebut ke dalam format GeoTIFF, dimana tekniknya sudah dibahas pada bagian terdahulu. Nantinya file GeoTIFF ini dapat Anda konversi menjadi shapefile atau pun format lainnya, dengan menggunakan perangkat lunak SIG.

### Uji Akurasi Menggunakan Confusion Matrix

Pada level praktis, uji akurasi data geospasial pada umumnya lebih bersifat opsional atau tidak diharuskan. Tetapi pada lingkup riset, misalnya skripsi, tesis, disertasi, atau riset-riset lainnya, uji akurasi hasil ekstraksi data geospasial dari citra satelit sangat direkomendasikan. Sebab hasil uji akurasi akan menentukan kredibilitas proses dan hasil ekstraksi data geospasial. Terdapat banyak metode uji akurasi, tentu saja tidak dapat dibahas semuanya di dalam buku ini. Pada bagian terdahulu sudah dibahas tentang MAPE dan RMSE, yang merupakan metode uji akurasi data geospasial bertipe numerik (kuantitatif), seperti biomassa vegetasi, temperatur, presipitasi, kedalaman air, muatan suspensi air, dan sebagainya.

Penutupan lahan atau yang sejenisnya merupakan data kualitatif. Untuk data geospasial kualitatif tidak dapat diuji menggunakan MAPE maupun RMSE. Data kualitatif dengan banyak kelas (*multiclass*) seperti penutupan lahan, jenis tanah, jenis batuan, dan sebagainya, biasanya diuji menggunakan confusion matrix. Terkait dengan teori confusion matrix, berikut interpretasi matrix-nya, silahkan lihat penjelasan confusion matrix pada halaman 270. Langkah pertama di dalam uji akurasi menggunakan confusion matrix adalah membuat validation set, sebagaimana training set sebelumnya. Inputnya tentu saja adalah feature collection dari shapefile validation area. Silahkan ketikkan dan jalankan kode-kode berikut:

```
# Pembuatan validation set
validation_set = s2_image.select(bands).sampleRegions(**{
    'collection': validation_fc,
    'properties': ['Id'],
    'scale': 10
})
```

Berikutnya adalah membuat dan menampilkan confusion matrix. Jalankan kode-kode berikut:

```
# Uji akurasi menggunakan confusion matrix
cmatrix = (
    validation_set.classify(classifier)
    .errorMatrix(actual='Id', predicted='classification')
)
```

```
# Menampilkan confusion matrix
cmatrix.getInfo()
```

```
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 159, 10, 2, 5, 0, 0, 0, 0, 0],
 [0, 77, 422, 4, 74, 0, 0, 0, 0, 0],
 [0, 0, 0, 349, 0, 0, 0, 35, 0],
 [0, 0, 56, 0, 125, 0, 0, 8, 0],
 [0, 0, 0, 0, 36, 0, 0, 1],
 [0, 0, 0, 25, 5, 0, 201, 96, 41],
 [0, 0, 0, 6, 0, 0, 67, 235, 4],
 [0, 0, 0, 2, 0, 0, 0, 0, 2083]]
```

Selain matrix hasil uji akurasi, confusion matrix juga akan menghasilkan informasi seperti *Overall Accuracy* (OA), *Koefisien Kappa*, *Producer's Accuracy* (PA), dan *User's Accuracy* (UA). Terkait parameter-parameter ini, berikut interpretasinya, silahkan baca di halaman 273.

Silahkan ketikkan dan eksekusi kode-kode berikut:

```
# Menampilkan overall accuracy
cmatrix.accuracy()
```

```
0.874515503875969
```

```
# Menampilkan Koefisien Kappa
cmatrix.kappa()
```

```
0.8199045410787515
```

```
# Menampilkan producer's accuracy
cmatrix.producersAccuracy()
```

```
▼ List (1 element)
  ▼ 0: List (9 elements)
    0: 0
    1: 0.673728813559322
    2: 0.8647540983606558
    3: 0.8994845360824743
    4: 0.5980861244019139
    5: 1
    6: 0.75
    7: 0.6283422459893048
    8: 0.9783936120244247
```

```
# Menampilkan user's accuracy
cmatrix.consumersAccuracy()
```

```
▼ List (9 elements)
  ► 0: [0]
  ► 1: [0.9034090909090909]
  ► 2: [0.7313691507798961]
  ► 3: [0.9088541666666666]
  ► 4: [0.6613756613756614]
  ► 5: [0.972972972972973]
  ► 6: [0.5461956521739131]
  ► 7: [0.7532051282051282]
  ► 8: [0.9990407673860912]
```

## Klasifikasi Multispektral Tak Terselia

Berbeda dengan klasifikasi multispektral terselia yang sudah kita bahas sebelumnya, klasifikasi multispektral tak terselia (*unsupervised multispectral classification*), atau dikenal juga sebagai *unsupervised clustering*, sama sekali tidak memerlukan keberadaan sampel atau training area. Yang diperlukan metode ini sebagai input awal biasanya hanya jumlah kelas. Beberapa metode klasifikasi multispektral tak terselia bahkan tidak memerlukan informasi apa-apa sebagai input. Kita cukup hanya mempersiapkan citra satelitnya saja untuk diproses. Sehingga metode ini sangat membantu bagi operator atau analis yang belum familiar dengan interpretasi objek di atas citra.

Kekurangan dari metode ini adalah outputnya tidak memberikan kelas-kelas yang informatif. Melainkan hanya kode-kode saja, biasanya dalam bentuk nomor, dari kelas 1 sampai kelas n. Di dalam aplikasinya, jika kita menggunakan klasifikasi multispektral tak terselia dalam ekstraksi penutupan lahan misalnya, maka setelah kode-kode kelasnya keluar (1 sampai n), langkah berikutnya adalah mengisi data atau mengganti kode-kode kelas dengan data informatif. Misalnya kelas 1 adalah tubuh air, kelas 2 adalah permukiman, begitu seterusnya. Informasi kelas-kelas seperti ini bisa diperoleh dari berbagai sumber, mungkin dari citra resolusi spasial tinggi atau langsung survey lapangan.

Terdapat banyak sekali metode klasifikasi multispektral tak terselia yang sudah dikembangkan. Pada latihan ini, kita akan mencoba metode *unsupervised machine learning* yang cukup terkenal, yaitu *k-Means* (MacQueen, 1967). Metode k-Means bekerja dengan cara mengukur jarak spektral setiap vektor nilai pixel ke vektor nilai pixel rerata masing-masing klaster/kelas. Penunjukkan kelas akan ditentukan dari jarak terdekat suatu vector nilai pixel terhadap vektor nilai pixel rerata suatu klaster. Karakter *k* pada nama k-Means menunjukkan jumlah klaster. Algoritma lain yang mirip dengan k-Means adalah *k-Medoids* (Jin and Han, 2011). Jika k-Means menggunakan menggunakan *centroid* (nilai rerata) sebagai pusat klaster, *k-Medoids* menggunakan *medoid* (median) sebagai pusat klaster.

Langkah pertama di dalam implementasi k-Means menggunakan GEE adalah membuat self-training data. Catatan, kode-kode di bawah merupakan kelanjutan dari klasifikasi multispektral sebelumnya. Jika Anda membangun kode-kode k-Means dari awal, maka variabel `s2_image` dan `bands` harus didefinisikan terlebih dahulu.

Silahkan lanjutkan kode-kode sebelumnya dengan kode-kode berikut:

```
# Membuat self-training data
training_data = s2_image.select(bands).sample(**{
    'region': s2_image.geometry(),
    'scale': 10,
    'numPixels': 1000,
    'tileScale': 8
})
```

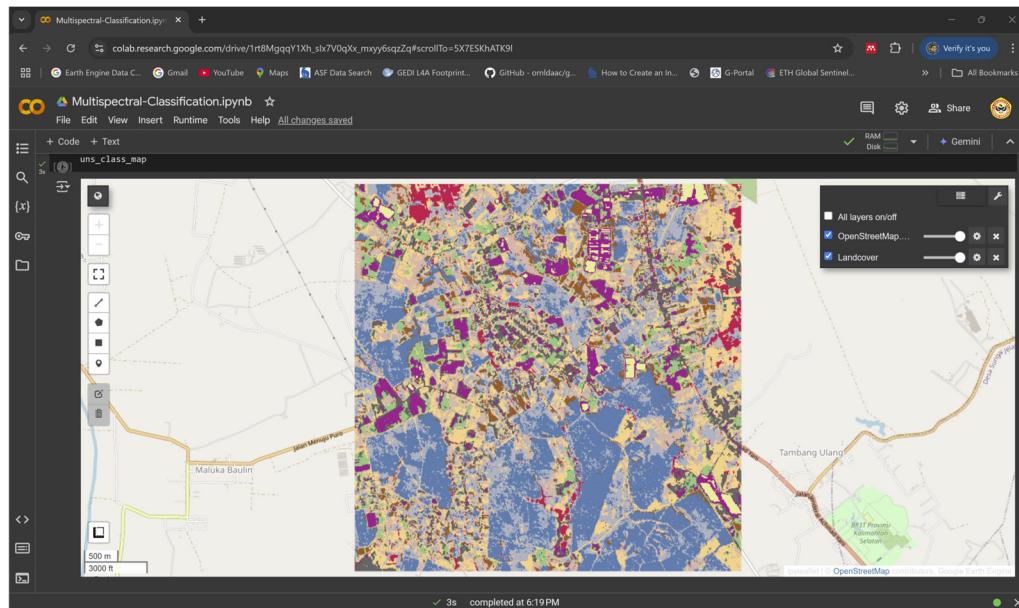
Penjelasan tentang parameter-parameter pada kode-kode di atas selengkapnya dapat dilihat di laman <https://developers.google.com/earth-engine/apidocs/ee-image-sample>. Lanjutkan dengan mengetikkan dan menjalankan kode-kode berikut:

```
# Implementasi algoritma klasifikasi K-Means
clusterer = ee.Clusterer.wekaKMeans(10).train(training_data)
uns_class_image = s2_image.select(bands).cluster(clusterer)
```

Kode `wekakMeans(10)` berarti kita akan membuat 10 kelas atau klaster. Untuk memvisualisasikan hasil k-Means, jalankan kode-kode berikut:

```
# Visualisasi citra hasil klasifikasi multispectral
uns_class_vis = {'min': 0, 'max': 9, 'palette': 'Accent_r'}
uns_class_map = geemap.Map()
uns_class_map.centerObject(tambang_ulang_region, 14)
uns_class_map.add_layer(uns_class_image, uns_class_vis, 'Landcover')
uns_class_map
```

Berikut adalah outputnya:



Langkah berikutnya, kita dapat mengekspor image hasil k-Means di atas menjadi GeoTIFF atau shapefile. Untuk selanjutnya, atributnya nanti akan kita isi dengan informasi dari hasil survei lapangan atau citra resolusi spasial tinggi. Klasifikasi multispektral tak terselai seperti ini juga sering digunakan untuk pengelompokan fitur-fitur seperti kelas kerapatan hutan, kelas kerapatan bangunan, kelas kekeruhan air, dan sebagainya. Termasuk di dalam klasifikasi citra ke dalam dua kelas (*binary classification*), seperti memisahkan vegetasi dan non-vegetasi pada NDVI.

Pada contoh kode-kode di atas, kita mengimplementasikan algoritma k-Means klasik, dimana inputnya adalah jumlah kelas yang tidak dapat berubah (*fixed*). Pada kondisi tertentu, kita terkadang tidak mengetahui dengan pasti berapa jumlah kelas yang paling tepat atau paling optimum. Sehingga hal ini sering menimbulkan pertanyaan atau perdebatan tentang validitas hasil klasifikasi menggunakan k-Means. Pada situasi seperti ini, kita dapat menggunakan variasi lain dari algoritma k-Means, yaitu *x-Means* (Pelleg and Moore, 2000). Berbeda dengan k-Means konvensional yang meminta informasi jumlah kelas yang konstan, x-Means akan meminta informasi berapa jumlah kelas minimum dan berapa jumlah kelas maksimum. Nanti x-Means akan mengestimasi sendiri berapa jumlah kelas yang paling optimum pada rentang nilai minimum dan maksimum yang kita berikan. Silahkan baca penjelasan tentang implementasi x-Means di tautan <https://developers.google.com/earth-engine/apidocs/ee-clusterer-wekaxmeans>.

## F. Analisis Citra Berbasis Objek

Analisis citra berbasis objek atau *Object-Based Image Analysis* (OBIA) (Blaschke, 2010), atau kadang disebut juga *Geographic Object-Based Image Analysis* (GEOBIA) (Blaschke et al., 2014; Chen et al., 2018), didasarkan pada pemikiran bahwa setiap objek yang akan kita ekstrak di atas citra dapat saja terkomposisi dari berbagai fitur yang berbeda. Dimana masing-masing fitur tersebut memiliki karakteristik nilai-nilai spektral (*spectral signature*) yang bervariasi. Sebagai contoh, kelas permukiman di dalam ekstraksi data geospasial penutupan lahan, tentu tidak seluruhnya hanya terdiri atas bangunan atau rumah saja. Melainkan juga pepohonan, rerumputan, tanaman hias, kolam atau sumur, permukaan tanah, permukaan beton, dan sebagainya. Untuk lebih jelasnya dapat dilihat pada Gambar 3.3.



Gambar 3.3. Terminologi objek pada citra penginderaan jauh

Jika kita ingin memetakan penutupan lahan, tentu kelas-kelas yang akan muncul di atas peta adalah permukiman, perkebunan, sawah, hutan, dan sebagainya. Kelas-kelas seperti ini disebut sebagai kelas informasional (*informational class*) (Campbell and Wynne, 2011). Kecil kemungkinan kita akan mengekstrak informasi seperti kolam renang, sumur, pagar rumah, rumput, mobil, dan sebagainya, kecuali mungkin untuk keperluan khusus nantinya. Jika kita menggunakan klasifikasi multispektral pada citra dengan resolusi spasial yang cukup tinggi, yang akan muncul di peta nanti adalah kelas spektral (*spectral class*) (Campbell and Wynne, 2011) seperti kolam renang, rumput, mobil, dan sebagainya.

OBIA akan lebih tepat jika diterapkan dalam pemetaan penutupan lahan, khususnya pada citra satelit dengan resolusi spasial yang cukup tinggi. Sebab OBIA mampu menggabungkan beberapa pixel ke dalam satu kelas. Target dari OBIA tentu saja adalah untuk mengejar kualitas kartografi, agar nantinya tidak terlalu banyak objek yang tidak diperlukan, bahkan hanya satu pixel, yang hadir di atas peta. Serupa dengan klasifikasi multispektral, OBIA juga dapat terselia (*supervised segmentation*) atau tak terselia (*unsupervised segmentation*). Akan tetapi, di dalam latihan ini, kita hanya akan membahas OBIA terselia. OBIA tak terselia lebih mudah diimplementasikan dibandingkan dengan OBIA terselia. Sebab OBIA terselia pastinya memerlukan sampel.

Sebagai latihan, buat sebuah notebook Google Colab baru, kemudian beri nama misalnya **Supervised-OBIA.ipynb**. Selanjutnya ketikkan dan jalankan kode-kode berikut:

```
import ee, geemap

ee.Authenticate()

ee.Initialize(project='ee-geospatialulm')

# Mounting Google Drive
from google.colab import drive

drive.mount('/content/gdrive', force_remount=True)
path = '/content/gdrive/My Drive/geebook/'

# Reading Tambang Ulang administrative area shapefile from Google Drive
import geopandas as gpd

region_shp = gpd.read_file(path + 'vector/Tambang_ulang.shp')
training_shp = gpd.read_file(path + 'vector/training_objects.shp')
validation_shp = gpd.read_file(path + 'vector/validation_objects.shp')
```

Persyaratan *training objects* dan *validation objects* pada OBIA mirip dengan training area dan validation area pada klasifikasi multispectral tersedia sebelumnya. Yaitu terdapat kolom yang berisi keterangan penutupan lahan, misalnya `Landcover`. Dan terdapat kolom yang berisi kode-kode unik dari masing-masing kelas penutupan lahan, misalnya `Id`. Tentu saja nama kolomnya tidak harus `Landcover` dan `Id`, Anda bisa menggunakan nama yang berbeda. Perbedaannya dengan klasifikasi multispectral tersedia Adalah, di dalam OBIA ini kita menggunakan shapefile training dan validation dalam bentuk titik, bukan poligon.

```
# Converting shapefile into Earth Engine geometry

region = geemap.geopandas_to_ee(region_shp).geometry()
training = geemap.geopandas_to_ee(training_shp)
validation = geemap.geopandas_to_ee(validation_shp)

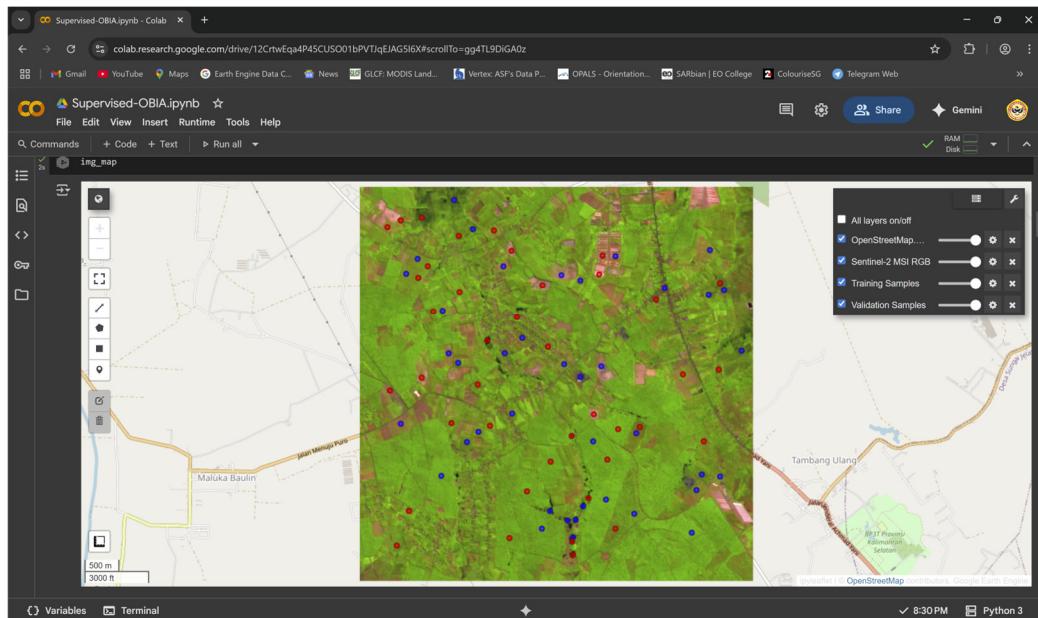
# Getting information about Sentinel-2 image ID
bands = ['B2', 'B3', 'B4', 'B5', 'B6', 'B7', 'B8', 'B8A', 'B11', 'B12']

s2_col = (
    ee.ImageCollection('COPERNICUS/S2_HARMONIZED')
    .filterDate('2019-06-01', '2019-07-31')
    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 5))
)
s2_image = (
    s2_col.median().clip(region)
    .select(bands)
)

# RGB image visualization
rgb_vis = {'min': 0, 'max': 5000, 'bands': ['B11', 'B8', 'B4']}
img_map = geemap.Map()
img_map.centerObject(region, 14)
img_map.add_layer(s2_image, rgb_vis, 'Sentinel-2 MSI RGB')
img_map.add_layer(training, {'color': 'blue'}, 'Training Samples')
img_map.add_layer(validation, {'color': 'red'}, 'Validation Samples')
img_map
```

### Bab III Google Earth Engine

Berikut adalah output visualisasi Citra Sentinel-2, berikut training objects dan validation objects:



Langkah berikutnya adalah membuat *seed pixels* untuk segmentasi citra dengan instruksi berikut:

```
# Select seed pixels for clustering
seeds = ee.Algorithms.Image.Segmentation.seedGrid(10)
```

Seed pixels merupakan penggabungan pixel-pixel yang terdekat lokasinya menjadi superpixel. Pada kode di atas kita membuat seed pixels untuk 10 pixel terdekat, default GEE adalah 5 pixel. Penjelasan selengkapnya dapat dibaca di tautan <https://developers.google.com/earth-engine/apidocs/ee-algorithms-image-segmentation-seedgrid>. Semakin besar ukuran seed pixels, misalnya 50 pixel, maka detail-detail kecil di atas citra akan hilang, dan ini sesuai untuk peta skala kecil, misalnya 1 : 50.000 atau 1 : 250.000. Untuk skala peta yang lebih besar, misalnya 1 : 25.000 atau 1 : 10.000, ukuran seed pixels harus diperkecil agar detail objek hasil OBIA tetap muncul di atas peta.

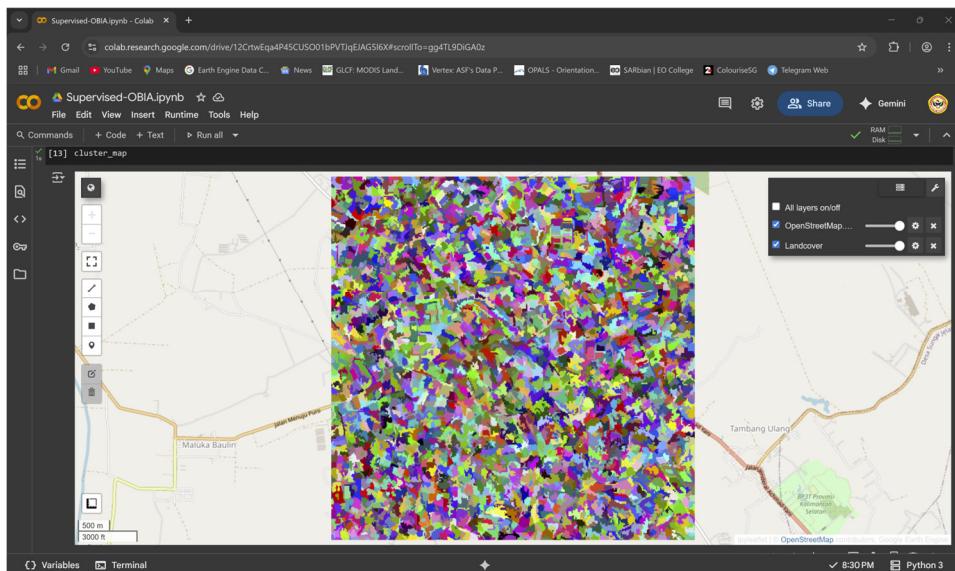
Langkah berikutnya adalah kita akan melakukan *superpixel clustering* berdasarkan seed pixels yang sudah kita buat. Metode yang akan kita gunakan adalah SNIC (*Simple Non-Iterative Clustering*) (Achanta and Süsstrunk, 2017). Penjelasan selengkapnya dapat dilihat di tautan <https://developers.google.com/earth-engine/apidocs/ee-algorithms-image-segmentation-snic>. silahkan ketikkan dan eksekusi kode-kode berikut:

```
# Superpixel clustering based on SNIC (Simple Non-Iterative Clustering)
seg_image = ee.Algorithms.Image.Segmentation.SNIC(
    image = s2_image,
    size = 10,
    connectivity = 8,
    seeds = seeds
)
```

```
# Select clusters band
clusters = seg_image.select('clusters')
```

```
# visualize clusters image
cluster_map = geemap.Map()
cluster_map.centerObject(region, 14)
cluster_map.add_layer(clusters.randomvisualizer(), None, 'Landcover')
cluster_map
```

Berikut adalah visualisasi hasil *superpixel clustering*:



Selanjutnya, kita dapat menampilkan informasi citra hasil superpixel segmentation:

```
# Displaying segmentation image information
seg_image
```

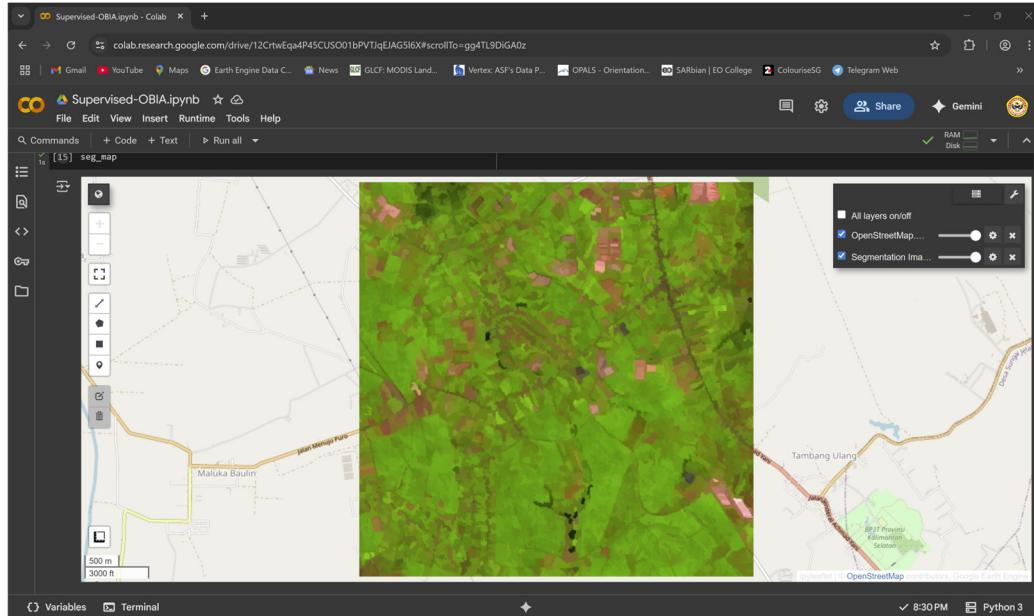
```
▼ Image (11 bands)
  type: Image
  ▼ bands: List (11 elements)
    ► 0: "clusters", signed int32, EPSG:4326, 2x2 px
    ► 1: "B2_mean", float, EPSG:4326, 2x2 px
    ► 2: "B3_mean", float, EPSG:4326, 2x2 px
    ► 3: "B4_mean", float, EPSG:4326, 2x2 px
    ► 4: "B5_mean", float, EPSG:4326, 2x2 px
    ► 5: "B6_mean", float, EPSG:4326, 2x2 px
    ► 6: "B7_mean", float, EPSG:4326, 2x2 px
    ► 7: "B8_mean", float, EPSG:4326, 2x2 px
    ► 8: "B8A_mean", float, EPSG:4326, 2x2 px
    ► 9: "B11_mean", float, EPSG:4326, 2x2 px
    ► 10: "B12_mean", float, EPSG:4326, 2x2 px
  ► properties: Object (1 property)
```

Pada output di atas, hasil segmentasi tersimpan pada band *clusters*. Sementara band-band lainnya merupakan rerata nilai pixel citra untuk setiap segmen (objek). Jika diinginkan, kita juga dapat memvisualisasikan citra komposit RGB band-band rerata tersebut, sebagaimana contoh pada kode-kode berikut:

### Bab III Google Earth Engine

```
# Visualize segmentation image mean RGB
seg_rgb = {'min': 300, 'max': 5000, 'bands': ['B11_mean', 'B8_mean', 'B4_mean']}
seg_map = geemap.Map()
seg_map.centerobject(region, 14)
seg_map.add_layer(seg_image, seg_rgb, 'Segmentation Image Mean RGB')
seg_map
```

Berikut adalah output visualisasinya:



Jika dilihat secara sepintas, memang output citra komposit rerata nilai pixel di atas mirip dengan citra aslinya. Akan tetapi jika citra dizoom, atau segmen (seed pixels)-nya diperbesar, maka segmen-segmennya akan terlihat dengan jelas. Visualisasi rerata nilai pixel per segmen seperti di atas biasanya digunakan untuk evaluasi visual hasil segmentasi, sebelum diproses lebih jauh. Jika ternyata hasil segmentasi OBIA tidak sesuai dengan yang diharapkan, misalnya terlalu banyak detail yang tidak diinginkan, kemungkinan ada parameter-parameter yang perlu dirubah. Misalnya jumlah seed pixels diperbesar jadi 20 atau 30.

Untuk selanjutnya, ketikkan dan jalankan kode-kode berikut untuk memisahkan antara band-band rerata nilai pixel dengan band clusters:

```
# Separates the mean band values from the 'clusters' band in the seg_image
seg_bands = seg_image.bandNames()
mean_bands = seg_bands.removeAll(['clusters'])
cluster_means_image = seg_image.select(mean_bands)
cluster_means_image
```

Kode-kode berikut digunakan untuk mengambil nama-nama band, pada citra khusus rerata nilai pixel per segmen yang sudah dipisahkan pada kode sebelumnya:

```
# Get band names list
training_bands = cluster_means_image.bandNames()
training_bands
```

Langkah berikutnya adalah mengekstrak nilai-nilai pixel citra khusus rerata nilai pixel per segmen, pada lokasi-lokasi training objects:

```
# Extracts pixel values from the specified bands (training_bands)
# at the locations defined by the training geometries
training_data = cluster_means_image.sampleRegions(
    collection = training,
    properties = ['Id'],
    tileScale = 16,
    scale = 10
)
```

Perhatikan kode di atas, bahwa properti yang digunakan pada training objects adalah kolom **Id**. Selanjutnya adalah proses training algoritma klasifikasi, dalam hal ini adalah Random Forest, dengan menggunakan training data hasil proses pada kode-kode di atas.

```
# Trains a Random Forest classifier using the sampled training data
classifier = (
    ee.Classifier.smileRandomForest(numberOfTrees=100)
    .train(training_data, 'Id')
)
```

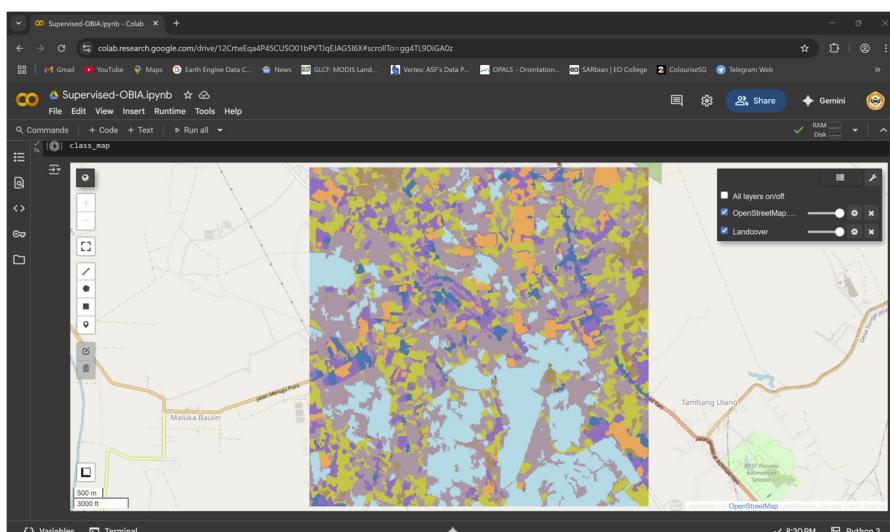
Berikutnya adalah klasifikasi segmen menggunakan citra rerata nilai pixel per segmen, dan algoritma Random Forest yang sudah ditraining sebelumnya:

```
# Applies the trained classifier to the cluster_means_image
# to produce a classified landcover image
class_image = cluster_means_image.classify(classifier)
```

Langkah selanjutnya adalah visualisasi citra hasil klasifikasi OBIA:

```
# Visualize landcover image
class_vis = {'min': 1, 'max': 8, 'palette': 'tab20'}

class_map = geemap.Map()
class_map.centerObject(region, 14)
class_map.add_layer(class_image, class_vis, 'Landcover')
class_map
```



Anda dapat mengevaluasi hasil klasifikasi OBIA di atas secara visual. Jika terdapat hal yang tidak sesuai, misalnya ada detail objek yang hilang, atau sebaliknya terlalu banyak detail, maka parameter-parameter klasifikasi dapat dirubah dan diproses kembali. Bahkan algoritma Random Forest dapat diganti dengan algoritma-algoritma klasifikasi terselain lainnya. Misalnya *k-Nearest Neighbor* (KNN), *Support Vector Machine* (SVM), *Naïve Bayes*, atau *Gradient Tree Boost*.

### Uji Akurasi Menggunakan Confusion Matrix

Sebagaimana klasifikasi multispektral terselain, hasil OBIA terselain juga dapat diuji menggunakan confusion matrix. Confusion matrix atau *error matrix*, atau kadang juga disebut sebagai *contingency matrix*, pada dasarnya merupakan tabel silang (*cross table*) antara data hasil ekstraksi dari citra satelit, seperti data geospasial penutupan lahan, dengan *ground truth* atau data aktual. Data aktual dapat diperoleh secara langsung dari lapangan, atau dengan bantuan citra dengan resolusi spasial yang lebih tinggi, atau dari peta terdahulu yang akurat. Berikut adalah format dasar confusion matrix (Congalton, 1991; Stehman and Czaplewski, 1998):

## Confusion Matrix

Data Geospasial	Data Lapangan				Jumlah Baris	Producer's Accuracy (PA)	User's Accuracy (UA)	Omission Error (OE)	Comission Error (CE)
	1	2	...	i					
1	$x_{11}$	$x_{12}$	...	$x_{1i}$	$x_{1+}$	$x_{11}/x_{1+}$	$x_{11}/x_{1+}$		
2	$x_{21}$	$x_{22}$	...	$x_{2i}$	$x_{2+}$	$x_{22}/x_{2+}$	$x_{22}/x_{2+}$		
...	...	...	...	...	...	...	...		
i	$x_{i1}$	$x_{i2}$	...	$x_{ii}$	$x_{i+}$	$x_{ii}/x_{i+}$	$x_{ii}/x_{i+}$		
Jumlah Kolom	$x_{+1}$	$x_{+2}$	...	$x_{+i}$	n				

### Parameter Akurasi:

$$\text{Producer's Accuracy} = \frac{x_{ii}}{x_{+i}}$$

$$\text{User's Accuracy} = \frac{x_{ii}}{x_{i+}}$$

$$\text{Omission Error} = 100\% - \text{Producer's Accuracy}$$

$$\text{Comission Error} = 100\% - \text{User's Accuracy}$$

$$\text{Overall Accuracy} = \frac{\sum_{i=1}^k x_{ii}}{n} \times 100\%$$

$$\text{Overall Error} = 100\% - \text{Overall Accuracy}$$

Gambar 3.4. Konsep uji akurasi menggunakan confusion matrix

Penjelasan confusion matrix pada Gambar 3.4:

$x_{ii}$  = Jumlah diagonal matrix

$x_{i+}$  = Jumlah kolom matrix

$x_{+i}$  = Jumlah baris matrix

n = Total sampel

k = Jumlah kelas

Langkah pertama di dalam proses uji akurasi menggunakan confusion matrix, adalah ekstraksi nilai-nilai pixel citra nilai rerata pixel per segmen (*cluster\_means\_image*) berdasarkan lokasi-lokasi validation objects. Silahkan ketikkan dan jalankan kode-kode berikut:

```
# Extracts the pixel values from the specified bands (training_bands)
# at the validation locations
validation_data = cluster_means_image.sampleRegions(
    collection = validation,
    properties = ['Id'],
    tileSize = 16,
    scale = 10
)
```

Berikutnya, kita akan menerapkan Random Forest yang sudah ditraining pada saat klasifikasi sebelumnya, tetapi kali ini terhadap validation data. Sebab pada prinsipnya, di dalam uji akurasi kita akan membandingkan antara algoritma Random Forest yang sudah ditraining menggunakan training objects, dengan data aktual dari validation data. Silahkan ketikkan dan jalankan kode-kode berikut:

```
# Applies the trained classifier (classifier) to the validation_data
classification_accuracy = validation_data.classify(classifier)
```

Langkah berikutnya adalah pembuatan confusion matrix dengan kode-kode berikut:

```
# Calculates the error matrix based on the validation data
accuracy_assessment = classification_accuracy.errorMatrix('Id', 'classification')
```

Pada bagian klasifikasi multispektral tersedia sebelumnya, kita menampilkan confusion matrix dalam bentuk teks biasa. Hal tersebut juga dapat kita lakukan pada confusion matrix hasil uji akurasi OBIA, yaitu dengan instruksi `accuracy_assessment.getInfo()`. Akan tetapi, di sini kita akan memvisualisasikan confusion matrix OBIA dengan format yang berbeda, yaitu dengan menggunakan *heatmap*. Silahkan ketikkan dan jalankan kode-kode berikut:

```
# Visualize confusion matrix using heatmap
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd

# Convert the Earth Engine ErrorMatrix to a list
conf_matrix_list = accuracy_assessment.array(). getInfo()

# Access the matrix without the first row and first column
# The first row and column typically represent the class labels
accuracy_matrix_values = [row[1:] for row in conf_matrix_list[1:]]

# Convert matrix list into Pandas DataFrame
conf_matrix_df = pd.DataFrame(accuracy_matrix_values).T

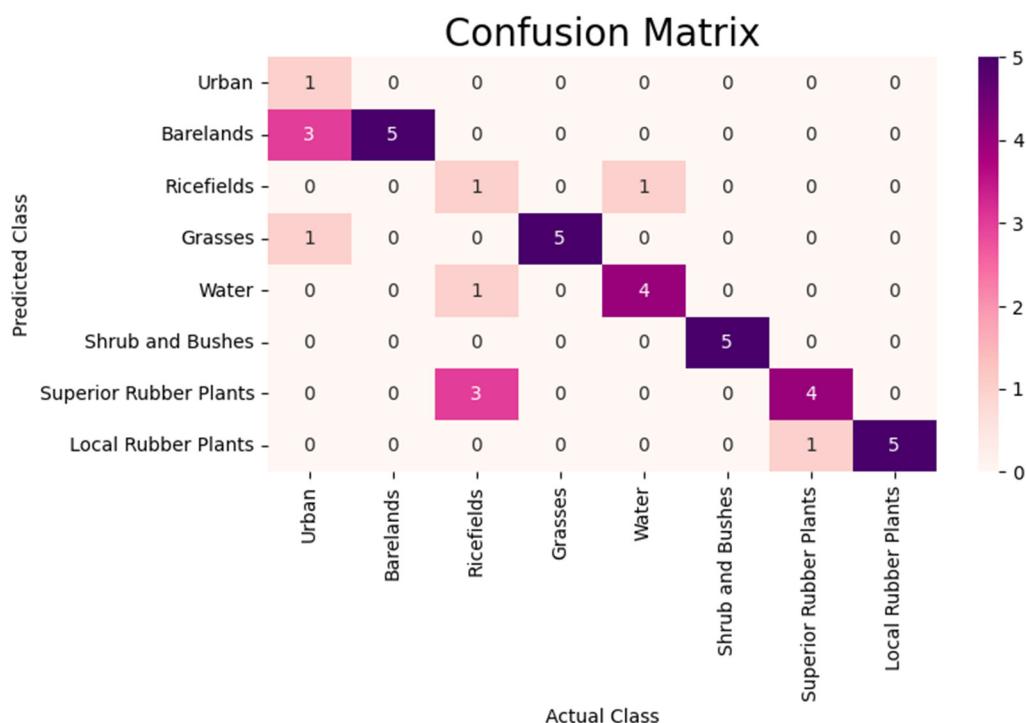
# Get class names from training data
class_labels = list(training_shp['Landcover'].drop_duplicates())

plt.figure(figsize=(8, 4))
sns.heatmap(conf_matrix_df, annot=True, fmt='d', cmap='RdPu',
            xticklabels=class_labels, yticklabels=class_labels)
plt.xlabel('Actual Class')
plt.ylabel('Predicted Class')
plt.title('Confusion Matrix', fontsize=20)
plt.show()
```

Perhatikan kode-kode di atas, heatmap disediakan oleh paket Seaborn, sehingga kita harus mengimport Seaborn terlebih dahulu. Selanjutnya, confusion matrix dikonversi ke dalam bentuk array dan teks dengan instruksi `accuracy_assessment.array().getInfo()`. Perhatikan visualisasi confusion matrix pada klasifikasi multispektral tersedia sebelumnya pada halaman 260. Dimana baris pertama dan kolom pertama berisi nilai 0 semuanya. Hal ini karena

baris dan kolom pertama diperuntukkan untuk label-label kelas. Sehingga kita harus mengekstrak sebuah matriks baru tanpa baris dan kolom pertama, yaitu dengan instruksi `accuracy_matrix_values = [row[1:] for row in conf_matrix_list[1:]]`. Selanjutnya, teks confusion matrix harus dikonversi menjadi Pandas dataframe dengan instruksi `pd.DataFrame(accuracy_matrix_values).T`. Dimana atribut `T` pada instruksi ini merupakan *transpose* atau memutar tabel dataframe. Sebenarnya tidak mengapa jika tidak ditranspose, akan tetapi confusion matrix yang dihasilkan nanti pada barisnya adalah data aktual, sedangkan pada kolomnya adalah data hasil ekstraksi. Meskipun bentuk confusion matrix seperti ini tetap benar, akan tetapi matrix-nya berbeda format dengan Gambar 3.4. Dimana pada Gambar 3.4, barisnya adalah data geospasial hasil ekstraksi, dan kolomnya adalah data aktual lapangan. Agar tidak membingungkan pembaca, maka confusion matrix yang dibuat di dalam kode-kode di atas disesuaikan formatnya dengan konsep pada Gambar 3.4.

Label-label confusion matrix diambil dari kolom '`Landcover`' shapefile training objects, yaitu dengan instruksi `list(training_shp['Landcover'].drop_duplicates())`. Fungsi `drop_duplicates()` harus dipanggil untuk menghilangkan duplikasi label untuk setiap kelas penutupan lahan. Hal ini karena pada shapefile training objects, terdapat lebih dari satu titik sampel untuk setiap kelas penutupan lahan. `sns.heatmap...` adalah instruksi pembuatan heatmap menggunakan Seaborn. Penjelasan yang lebih lengkap ada di tautan <https://seaborn.pydata.org/generated/seaborn.heatmap.html>. Berikut adalah confusion matrix dalam bentuk heatmap yang dihasilkan dari kode-kode di atas:



Angka-angka pada confusion matrix menunjukkan jumlah titik sampel uji akurasi atau titik validasi. Pada confusion matrix, angka yang terdapat pada diagonal matrix menunjukkan kuantitas klasifikasi yang benar. Sedangkan angka-angka di luar diagonal matrix menunjukkan kuantitas klasifikasi yang salah, dalam arti suatu objek terkласifikasikan sebagai kelas lain. Pada confusion matrix di atas, Ricefields yang benar-benar terkласifikasikan sebagai Ricefields hanya 1

titik, sementara Ricefields yang terkelaskan sebagai Water ada 1 titik, dan Ricefields yang terkelaskan sebagai Superior Rubber Plants ada 3 titik. Hal ini akan berpengaruh pada *Producer's Accuracy* (PA). Sebaliknya, ada Water yang terpetakan sebagai Ricefields 1 titik. Hal ini akan berpengaruh pada *User's Accuracy* (UA). Untuk lebih jelasnya, mari kita kalkulasi parameter-parameter akurasi yang diturunkan dari confusion matrix. Silahkan eksekusi kode-kode berikut:

```
# Calculates several common accuracy metrics from the error matrix
```

```
oa = accuracy_assessment.accuracy()
pa = accuracy_assessment.producersAccuracy()
ua = accuracy_assessment.consumersAccuracy()
kappa = accuracy_assessment.kappa()
```

```
# Prints the calculated accuracy metrics
```

```
print('Overall Accuracy: ', oa.getInfo())
print('Producers Accuracy: ', pa.getInfo())
print('Consumers Accuracy: ', ua.getInfo())
print('Kappa: ', kappa.getInfo())
```

Berikut adalah outputnya:

```
Overall Accuracy: 0.75
Producers Accuracy: [[0], [0.2], [1], [0.2], [1], [0.8], [1], [0.8], [1]]
Consumers Accuracy: [[0, 1, 0.625, 0.5, 0.833, 0.8, 1, 0.571, 0.833]]
Kappa: 0.714
```

*Overall Accuracy* (OA) menunjukkan akurasi keseluruhan data geospasial penutupan lahan yang dihasilkan dari OBIA. Biasanya akurasi keseluruhan ini sering menjadi bahan pertanyaan dosen penguji pada saat sidang skripsi/tesis/disertasi, "Seberapa akurat peta yang Anda sajikan?". Jawaban praktisnya adalah dari nilai OA. Sebagaimana konsep pada Gambar 3.4, OA 0,75 atau 75% berarti *Overall Error*-nya adalah 0,25 atau 25%. Untuk PA dan UA disajikan per kelas penutupan lahan. Perhatikan output di atas, angka pertama pada PA dan UA, yaitu 0, diperuntukkan untuk label, jadi angka pertama ini diabaikan. Selanjutnya, nilai-nilai PA dan UA akan berurutan mengikuti urutan label-label penutupan lahan pada confusion matrix. Yaitu dimulai dari Urban, Barelands, Ricefields, Grasses, dan seterusnya.

Jika dibaca dari output, maka PA Ricefields adalah 0,2 atau 20%, sementara UA Ricefields adalah 0,5 atau 50%. PA atau kadang disebut juga sebagai *Classification Accuracy* menunjukkan seberapa sukses proses identifikasi atau ekstraksi objek di atas citra menggunakan suatu metode. PA Ricefields 20% berarti dari seluruh Ricefields yang ada di citra (atau di lapangan), hanya 20% yang benar-benar terklasifikasikan sebagai Ricefields. Sedangkan sisanya yang 0,8 atau 80% merupakan *Omission Error* (OE). Jika OE Ricefields adalah 80%, maka 80% Ricefields yang ada di citra (atau di lapangan) hilang dari peta, sebab terkelaskan ke kelas lain.

UA atau *Consumer's Accuracy* atau disebut juga *Mapping Accuracy* menggambarkan kebenaran di atas peta. UA Ricefields 50% artinya dari seluruh Ricefields yang ada di atas peta, hanya 50% yang benar-benar Ricefields. Sisanya yang 50% merupakan *Comission Error* (CE), atau objek lain yang terpetakan sebagai Ricefields. Dari hasil evaluasi confusion matrix, PA, dan UA, maka Urban dan Ricefields merupakan dua kelas yang paling bermasalah akurasinya, sehingga mungkin harus diperbaiki titik-titik sampel training objects untuk kedua kelas penutupan lahan ini.

**SEBAGAI LATIHAN**, coba Anda visualisasikan nilai-nilai PA dan UA untuk seluruh kelas ke dalam sebuah Pandas dataframe yang menarik (berwarna), lengkap dengan label kelas-kelasnya.

### Bab III Google Earth Engine

Koefisien Kappa atau *Cohen's Kappa* atau dikenal juga sebagai *Khat coefficient* adalah metrik statistik yang digunakan untuk mengukur akurasi klasifikasi dengan mempertimbangkan kesepakatan acak. Berbeda dengan OA yang hanya menghitung proporsi klasifikasi yang benar, Kappa memberikan gambaran yang lebih adil tentang kualitas klasifikasi. Koefisien Kappa sebesar 0,714 dalam konteks confusion matrix untuk uji akurasi pemetaan penutupan lahan menunjukkan bahwa hasil klasifikasi memiliki tingkat kesesuaian yang tinggi dibandingkan dengan data referensi, setelah memperhitungkan kemungkinan kesesuaian yang terjadi secara acak. Koefisien Kappa diformulasikan oleh Profesor Jacob Cohen, seorang pakar psikologi dan statistika dari New York University, USA, sebagai berikut (Cohen, 1960; Congalton, 1991):

$$Kappa = \frac{n \sum_{i=1}^k x_{ii} - \sum_{i=1}^k x_{i+}x_{+i}}{n^2 - \sum_{i=1}^k x_{i+}x_{+i}}$$

Dimana untuk penjelasan parameter-parameter Koefisien Kappa pada formula di atas silahkan lihat konsep confusion matrix pada Gambar 3.4.

Tabel 3.3. Interpretasi umum nilai Kappa

Nilai Kappa	Tingkat Kesepakatan	Interpretasi Umum
< 0.00	Buruk	Tidak ada kesepakatan
0.00–0.20	Rendah	Sangat lemah
0.21–0.40	Cukup	Lemah
0.41–0.60	Sedang	Moderat
0.61–0.80	Tinggi	Baik
0.81–1.00	Sangat tinggi	Sangat baik / hampir sempurna

Sumber: Landis and Koch (1977)

Jadi, nilai Kappa 0,714 termasuk dalam kategori "tinggi", yang berarti bahwa klasifikasi penutupan lahan yang dilakukan memiliki akurasi yang baik dan dapat diandalkan, meskipun belum mencapai kesempurnaan. Dalam ekstrasi data geospasial penutupan lahan, nilai Kappa sebesar 0,714 menunjukkan bahwa metode klasifikasi (misalnya OBIA) mampu membedakan kelas-kelas tutupan lahan dengan cukup akurat, dan hasilnya layak digunakan untuk analisis lanjutan atau pengambilan keputusan.

### Konversi Hasil Segmentasi ke Vektor

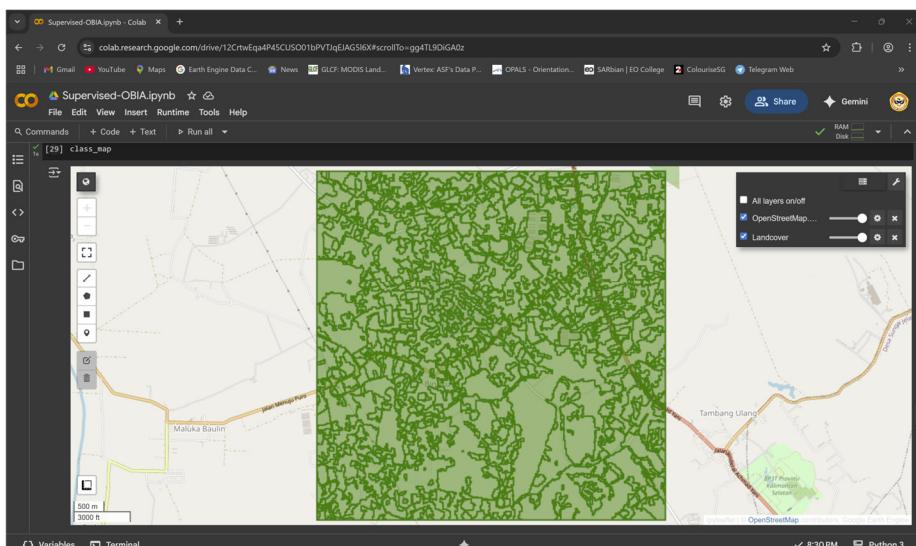
Langkah terakhir di dalam proses OBIA biasanya adalah konversi hasil segmentasi ke dalam format vektor, misalnya shapefile. Hal ini untuk memudahkan kalkulasi luas masing-masing kelas penutupan lahan, membuat layout peta, berbagi pakai data, atau pun analisis-analisis atau pemodelan-pemodelan geospasial yang lebih lanjut. Untuk mengkonversi citra hasil OBIA menjadi shapefile silahkan ketikkan dan jalankan kode-kode berikut:

```
# Converts the classified image into a vector (polygons)
class_fc = class_image.reduceToVectors(
    geometry=region,
    crs=s2_image.projection(),
    scale=10,
    geometryType='polygon',
    eightConnected=False,
    labelProperty='classification',
    maxPixels=1e13
)
```

Kode-kode di atas digunakan untuk mengkonversi citra hasil klasifikasi menjadi feature collection dalam bentuk poligon. Penjelasan selengkapnya terkait Teknik ini dapat dibaca di tautan <https://developers.google.com/earth-engine/apidocs/ee-image-reducetovectors>. Selanjutnya kita dapat memvisualisasikan feature collection tersebut dengan kode-kode seperti berikut:

```
# visualize the vectorized landcover classification
class_map = geemap.Map()
class_map.centerObject(region, 14)
class_map.add_layer(class_fc, {'color': 'green'}, 'Landcover')
class_map
```

Berikut adalah outputnya:



Karena vektor hasil OBIA belum memuat informasi nama-nama kelas, melainkan hanya kode-kode kelas yang berasal dari kolom **Id**, maka perlu diambil nama-nama kelas (kolom **Landcover**) dari shapefile training objek. Tentu saja yang diperlukan adalah nama-nama kelas unik seperti pembuatan label confusion matrix sebelumnya. Jalankan kode-kode berikut:

```
# Create a Pandas DataFrame containing the unique landcover class IDs
# and their corresponding names from the training_shp
class_name = training_shp[['Id', 'Landcover']].drop_duplicates()
class_name
```

Instruksi `training_shp[['Id', 'Landcover']].drop_duplicates()` digunakan untuk mengambil kode-kode dan nama-nama kelas penutupan lahan unik dari kolom **Id** dan **Landcover**. Berikut adalah outputnya:

	<b>Id</b>	<b>Landcover</b>
0	1	Urban
5	2	Barelands
10	3	Ricefields
15	4	Grasses
20	5	Water
25	6	Shrub and Bushes
30	7	Superior Rubber Plants
35	8	Local Rubber Plants

Langkah berikutnya adalah menambahkan properti **Landcover** ke masing-masing kelas penutupan lahan. Silahkan ketikkan dan jalankan kode-kode berikut:

```
# Adds a 'Landcover' property to each polygon in the classified vector data
class_dict = dict(zip(class_name['Id'], class_name['Landcover']))

def add_landcover(feature):
    id_value = feature.get('classification')
    landcover_value = ee.String(ee.Dictionary(class_dict).get(id_value))
    return feature.set('Landcover', landcover_value)

class_fc = class_fc.map(add_landcover)
```

Fungsi **zip()** pada kode di atas adalah fungsi bawaan Python yang digunakan untuk menggabungkan elemen dari beberapa iterable (seperti list, tupel, atau string) menjadi satu iterable tupel. Pada kode di atas, kita membuat sebuah fungsi **add\_landcover** untuk menambahkan nama-nama kelas penutupan lahan ke dalam feature collection hasil OBIA. Langkah selanjutnya tentu saja adalah mengekspor data vektor menjadi shapefile. Silahkan ketikkan dan jalankan kode-kode berikut:

```
# Exports the classified landcover polygons to a shapefile
geemap.ee_export_vector(
    class_fc,
    filename=path + 'output/classification.shp',
)
```

Sampai pada langkah-langkah di atas, sebenarnya shapefile kita sudah tersimpan di dalam Google Drive atau local drive komputer kita, dengan nama **classification.shp**. Selanjutnya jika diperlukan, kita dapat membuka shapefile yang sudah diekspor tersebut ke dalam Google Colab dengan menggunakan GeoPandas.

```
# Open the classified landcover polygon shapefile
class_shp = gpd.read_file(path + 'output/classification.shp')
class_shp
```

Berikut adalah tampilan data atribut shapefilenya:

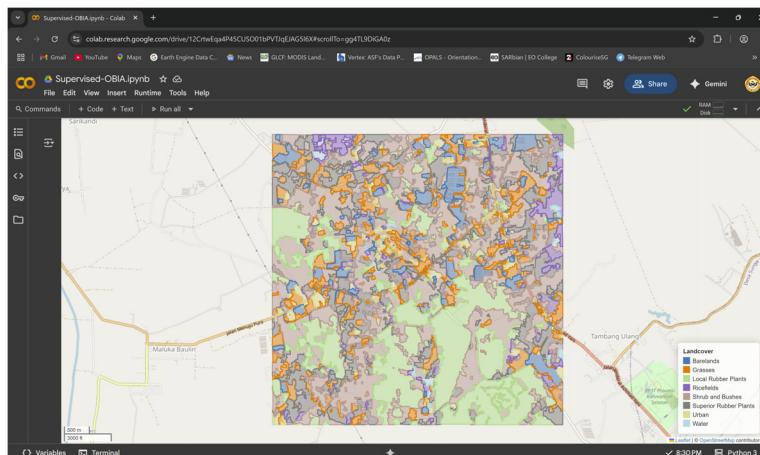
	Landcover	count	classification	geometry
0	Grasses	37	4	POLYGON ((114.66797 -3.65542, 114.66806 -3.655...
1	Shrub and Bushes	10	6	POLYGON ((114.66797 -3.69324, 114.66797 -3.692...
2	Local Rubber Plants	4	8	POLYGON ((114.66797 -3.6936, 114.66797 -3.6932...
3	Shrub and Bushes	48	6	POLYGON ((114.66797 -3.68148, 114.66824 -3.681...
4	Grasses	88	4	POLYGON ((114.66815 -3.65363, 114.66833 -3.653...
...	...	...	...	...

Kita juga dapat memvisualisasikan shapefile secara interaktif dengan menggunakan MapClassify (<https://pysal.org/mapclassify/>). Silahkan ketikkan dan jalankan kode-kode berikut:

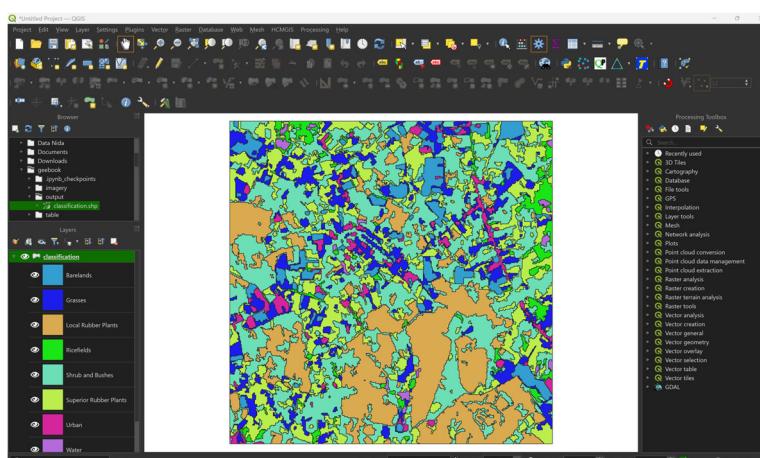
```
!pip install mapclassify
```

```
# visualize the classified landcover polygon shapefile
class_shp.explore(column='Landcover', cmap='tab20')
```

Berikut adalah outputnya:



Dan berikut adalah shapefile hasil ekspor yang dibuka menggunakan QGIS:



## G. Analisis Multitemporal

Analisis multitemporal di dalam konteks penginderaan jauh adalah analisis yang melibatkan beberapa citra dengan waktu akuisisi yang berbeda, biasanya minimal dua waktu akuisisi. Aplikasi analisis multitemporal sangat luas, di dalam buku ini hanya disajikan beberapa di antaranya yang memang cukup sering diterapkan.

Jenis-jenis aplikasi analisis multitemporal yang cukup sering diimplementasikan pada citra penginderaan jauh adalah:

1. Ekstraksi informasi tertentu yang memang tidak dapat atau sulit dilakukan pada citra akuisisi tunggal. Seperti deforestasi, emisi dan sekuestrasi karbon, genangan banjir, area terbakar, deformasi lahan, dan sebagainya.
2. Observasi perubahan atau dinamika objek menurut waktu. Seperti pertumbuhan tanaman, perkembangan permukiman, dinamika habitat satwa liar, perubahan iklim, dan sebagainya.
3. Koreksi, imputasi, atau augmentasi citra. Seperti tambal-sulam awan, koreksi *noise*, interpolasi temporal terhadap pixel-pixel yang hilang, dan sebagainya.
4. Estimasi atau prediksi lokasi sumber dari suatu fenomena. Misalnya dari mana kebakaran hutan dan lahan itu berasal. Hal ini dapat dipantau dari asal-muasal kemunculan titik api atau pergerakan titik api dari waktu ke waktu.
5. Evaluasi, estimasi, atau prediksi penyebab atau dampak suatu fenomena. Seperti estimasi penyebab deforestasi, estimasi penyebab kebakaran hutan dan lahan, evaluasi dan prediksi dampak pembangunan infrastruktur, evaluasi dan prediksi dampak perubahan tata ruang wilayah, dan sebagainya.

### Ekstraksi Informasi Geospasial Sawah

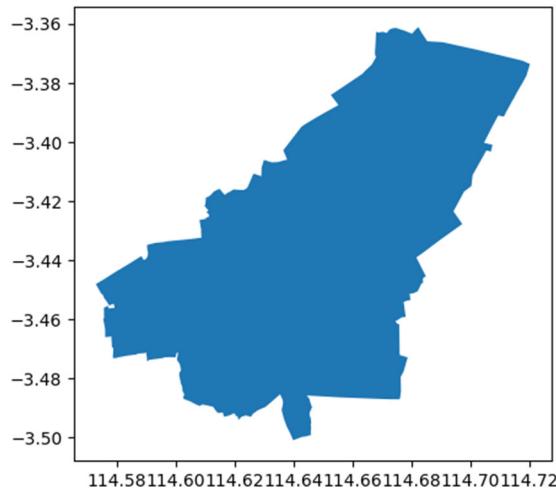
Beberapa informasi geospasial tidak dapat diekstrak secara langsung menggunakan citra akuisisi satu waktu, misalnya deforestasi. Jika kita mau menghitung luas deforestasi selama lima tahun misalnya, setidaknya kita harus menyediakan citra multitemporal dua waktu (bitemporal) dengan interval waktu selama lima tahun. Beberapa informasi geospasial dapat diekstrak langsung dengan citra akuisisi tunggal, sebagaimana penutupan lahan. Akan tetapi, ada juga objek yang dapat diekstrak dengan citra akuisisi tunggal, tetapi ada kemungkinan akan lebih efektif jika diekstrak menggunakan citra multitemporal. Contohnya adalah ekstraksi informasi geospasial sawah tada hujan, yang akan kita praktikkan pada latihan ini.

Silahkan buat sebuah notebook Google Colab baru, beri nama misalnya **Pemetaan-Sawah.ipynb**. Kemudian ketikkan dan jalankan kode-kode berikut:

```
import ee, geemap  
  
ee.Authenticate()  
  
ee.Initialize(project='ee-geospatialulm')  
  
# Mengakses Google Drive  
from google.colab import drive  
drive.mount('/content/drive', force_remount=True)
```

```
# Membuka shapefile wilayah observasi
import geopandas as gpd

region_shp = gpd.read_file('/content/drive/My Drive/geebook/vector/Gambut.shp')
region_shp.plot()
```



```
# Konversi shapefile menjadi Earth Engine geometry
region = geemap.geopandas_to_ee(region_shp).geometry()
```

Kode-kode berikutnya adalah penentuan waktu akuisisi citra multitemporal. Dalam hal ini, kita akan menggunakan tiga waktu akuisisi. Yaitu awal musim hujan (sebelum tanam padi), akhir musim hujan (saat tanaman padi sudah dewasa), dan musim kemarau (setelah padi dipanen). Hal ini karena pada sawah tadah hujan, pada saat awal musim hujan ( $t_1$ ) sawah akan berupa hamparan rawa. Sehingga pada momen ini fitur air sangat dominan, dan mudah diekstrak menggunakan MNDWI. Pada saat akhir musim hujan ( $t_2$ ), padi sudah dewasa dan hijau (konsentrasi klorofil tinggi), sehingga mudah diekstrak menggunakan NDVI. Sementara pada saat pasca panen ( $t_3$ ), padi mengering dan kehilangan klorofil, akibatnya nilai NDVI sangat rendah. Dengan mengeksplorasi *temporal behavior* tanaman padi pada sawah tadah hujan seperti ini, kita dapat membedakan padi dari tanaman lainnya. Silahkan jalankan kode-kode berikut:

```
# Penentuan tanggal untuk 3 waktu akuisisi citra
# 1. Awal musim hujan (sebelum tanam padi)
start_date1 = '2019-02-01'
end_date1 = '2019-02-28'

# 2. Akhir musim hujan (padi dewasa)
start_date2 = '2019-06-01'
end_date2 = '2019-07-15'

# 3. Musim kemarau (pasca panen)
start_date3 = '2019-09-01'
end_date3 = '2019-09-30'
```

Pengetahuan lokal tentang masa tanam dan masa panen padi di lokasi observasi mutlak diperlukan, hal ini untuk menjadi dasar bagi penentuan waktu akuisisi citra yang akan digunakan. Termasuk pengetahuan lapangan tentang kapan sekiranya padi mulai menguning atau dedaunannya mulai kehilangan pigmen klorofil. Silahkan lanjutkan dengan mengetikkan dan menjalankan kode-kode berikut:

### Bab III Google Earth Engine

```
# Akses Sentinel-2 image collection untuk 3 waktu akuisisi
s2_col1 = (
    ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
    .filterDate(start_date1, end_date1)
    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20))
)

s2_col2 = (
    ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
    .filterDate(start_date2, end_date2)
    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20))
)

s2_col3 = (
    ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')
    .filterDate(start_date3, end_date3)
    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20))
)
```

```
# Reduksi Sentinel-2 image collection menjadi image
s2_image1 = s2_col1.median().clip(region).divide(10000)
s2_image1 = s2_image1.select(['B2', 'B3', 'B4', 'B8', 'B11', 'B12'])

s2_image2 = s2_col2.median().clip(region).divide(10000)
s2_image2 = s2_image2.select(['B2', 'B3', 'B4', 'B8', 'B11', 'B12'])

s2_image3 = s2_col3.median().clip(region).divide(10000)
s2_image3 = s2_image3.select(['B2', 'B3', 'B4', 'B8', 'B11', 'B12'])
```

Untuk menguji akurasi informasi geospasial sawah hasil ekstraksi dari citra multitemporal, kita memerlukan sejumlah sampel. Yaitu sampel-sampel objek sawah dan objek selain sawah. Sampel-sampel ini didigitasi menggunakan perangkat lunak SIG dalam bentuk polygon. Digitasi bisa dilakukan pada citra dengan resolusi spasial tinggi, Google Map, atau mengacu ke titik-titik sampel hasil survey lapangan. Di dalam Latihan ini, sampel-sampel sawah dan non-sawah sudah disediakan secara terpadu dalam bentuk shapefile. Silahkan jalankan kode-kode berikut:

```
# Membuka shapefile sampel sawah dan non-sawah
samples_shp = '/content/drive/My Drive/geebook/vector/Rice_Field_Samples.shp'
rf_samples_shp = gpd.read_file(samples_shp)
rf_samples_shp
```

	<b>Id</b>	<b>Cover</b>	<b>geometry</b>
0	1	Ride Fields	POLYGON ((114.63726 -3.41228, 114.6447 -3.4114...
1	1	Ride Fields	POLYGON ((114.64221 -3.40544, 114.64597 -3.405...
2	1	Ride Fields	POLYGON ((114.65487 -3.40583, 114.6573 -3.4039...
3	1	Ride Fields	POLYGON ((114.65133 -3.39409, 114.65297 -3.396...
4	1	Ride Fields	POLYGON ((114.67664 -3.38038, 114.67914 -3.381...
5	1	Ride Fields	POLYGON ((114.68513 -3.36678, 114.68603 -3.368...
6	1	Ride Fields	POLYGON ((114.65706 -3.42095, 114.65796 -3.422...
7	1	Ride Fields	POLYGON ((114.63916 -3.41795, 114.64321 -3.417...
8	1	Ride Fields	POLYGON ((114.60914 -3.43842, 114.61155 -3.437...
9	1	Ride Fields	POLYGON ((114.5841 -3.46364, 114.58924 -3.4635...
10	1	Ride Fields	POLYGON ((114.61722 -3.46419, 114.62047 -3.463...
11	1	Ride Fields	POLYGON ((114.64604 -3.45768, 114.64861 -3.457...
12	1	Ride Fields	POLYGON ((114.60117 -3.45434, 114.60269 -3.454...
13	1	Ride Fields	POLYGON ((114.63108 -3.43743, 114.63301 -3.437...
14	1	Ride Fields	POLYGON ((114.66913 -3.39314, 114.67235 -3.390...
15	0	Non Rice Fields	POLYGON ((114.69702 -3.37185, 114.70052 -3.372...
16	0	Non Rice Fields	POLYGON ((114.69519 -3.37397, 114.69731 -3.374...
17	0	Non Rice Fields	POLYGON ((114.69472 -3.37813, 114.69747 -3.379...
18	0	Non Rice Fields	POLYGON ((114.70923 -3.37607, 114.71229 -3.374...

Selanjutnya, kita akan memisahkan sampel sawah dan non-sawah, sekaligus mengkonversinya menjadi feature collection. Hal ini karena sampel-sampel sawah dan non-sawah kita terpadu dalam satu shapefile. Silahkan ketikkan dan jalankan kode-kode berikut:

```
# Memisahkan sampel sawah dan non-sawah dan konversi menjadi GEE geometry
rice_fields_shp = rf_samples_shp[rf_samples_shp['Id'] == 1]
non_rice_fields_shp = rf_samples_shp[rf_samples_shp['Id'] == 0]

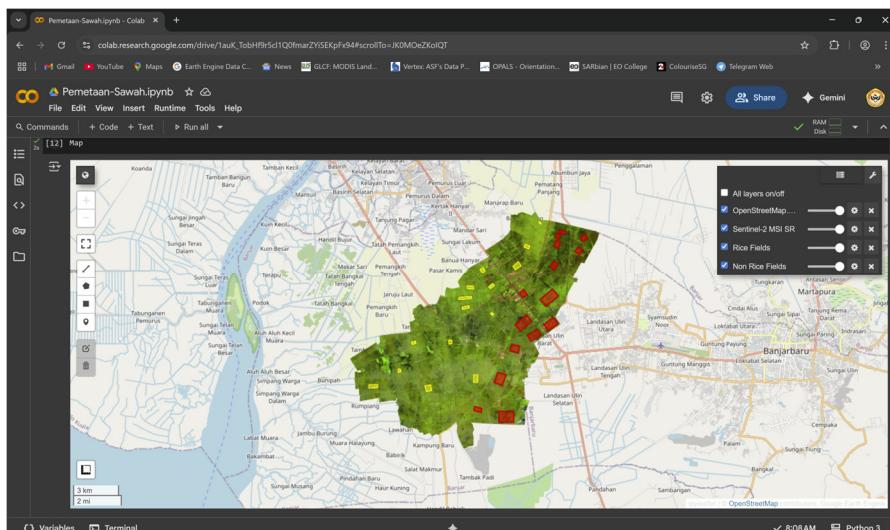
rice_fields = geemap.geopandas_to_ee(rice_fields_shp)
non_rice_fields = geemap.geopandas_to_ee(non_rice_fields_shp)
```

Berikutnya, jika diperlukan untuk evaluasi, sampel-sampel sawah dan non-sawah dapat divisualisasikan ke atas citra. Dimana Citra Sentinel-2 yang dipilih adalah citra *t2*, yaitu akhir musim hujan pada saat padi sudah dewasa.

```
# visualisasi Citra Sentinel-2 RGB dan sampel sawah dan non-sawah
rgb_vis = {'min': 0.0, 'max': 0.5, 'bands': ['B11', 'B8', 'B2']}

Map = geemap.Map()
Map.centerObject(region, 12)
Map.addLayer(s2_image2, rgb_vis, 'sentinel-2 MSI SR')
Map.addLayer(rice_fields, {'color': 'ffff00'}, 'Rice Fields')
Map.addLayer(non_rice_fields, {'color': 'ff0000'}, 'Non Rice Fields')
Map
```

Berikut adalah outputnya:



Selanjutnya, kita akan melakukan transformasi MNDWI pada citra *t1*, dan transformasi NDVI pada citra *t2* dan *t3*.

```
# Mengambil band-band yang diperlukan untuk transformasi citra
green1 = s2_image1.select('B3')
swir11 = s2_image1.select('B11')

red2 = s2_image2.select('B4')
nir2 = s2_image2.select('B8')

red3 = s2_image3.select('B4')
nir3 = s2_image3.select('B8')
```

### Bab III Google Earth Engine

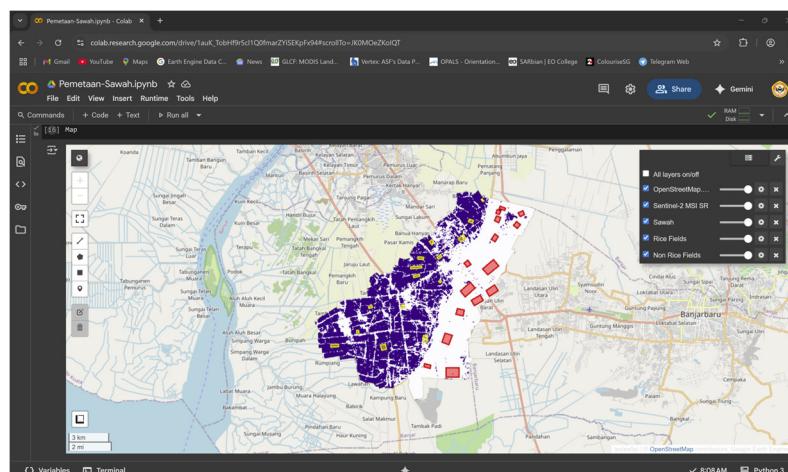
```
# Transformasi MNDWI dan NDVI  
mndwi1 = (green1.subtract(swir11)).divide(green1.add(swir11)).rename('mndwi')  
ndvi2 = (nir2.subtract(red2)).divide(nir2.add(red2)).rename('ndvi')  
ndvi3 = (nir3.subtract(red3)).divide(nir3.add(red3)).rename('ndvi')
```

Langkah berikutnya, kita akan melakukan *thresholding* pada indeks-indeks spectral untuk mengekstrak data sebaran sawah. Dalam hal ini, jika nilai MNDWI pada saat  $t1$  lebih dari -0,25, nilai NDVI pada saat  $t2$  lebih dari atau sama dengan 0,25, dan nilai NDVI pada saat  $t3$  kurang dari 0,5, maka ditetapkan bahwa itu adalah objek tanaman padi atau sawah. Berikut adalah kode-kode implementasi dari ekspresi logika tersebut:

```
# Ekstraksi sawah menggunakan thresholding MNDWI dan NDVI  
rice_field_class = (  
    mndwi1.gt(-0.25).And(ndvi2.gte(0.25))  
    .And(ndvi3.lt(0.5)).rename('Rice Fields')  
)
```

Pada latihan ini, nilai-nilai threshold MNDWI dan NDVI ditentukan secara subjektif untuk kepraktisan. Idealnya, nilai-nilai threshold ini dapat mengacu ke literatur, atau bisa juga didapatkan secara empiris dengan menggunakan sampel-sampel sawah yang ada. Tentu saja prosesnya lebih kompleks, dan tidak dibahas di dalam latihan ini. Anda dapat mengembangkan kode-kode sendiri jika ingin mengekstrak nilai-nilai threshold MNDWI dan NDVI secara langsung dari sampel-sampel dan citra-citra yang digunakan. Selanjutnya, kita akan memvisualisasikan informasi geospasial sawah yang sudah kita ekstrak dengan kode-kode berikut:

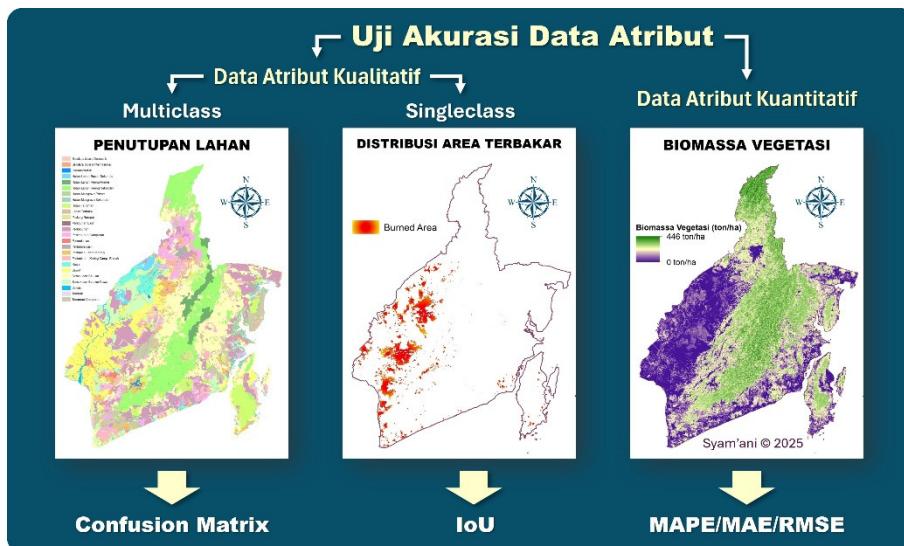
```
# Visualisasi hasil ekstraksi sawah dan sampel sawah dan non-sawah  
rf_vis = {'min': 0, 'max': 1, 'palette': 'Purples'}  
  
Map = geemap.Map()  
Map.centerObject(region, 12)  
Map.addLayer(s2_image2, rgb_vis, 'Sentinel-2 MSI SR')  
Map.addLayer(rice_field_class, rf_vis, 'sawah')  
Map.addLayer(rice_fields, {'color': 'ffff00'}, 'Rice Fields')  
Map.addLayer(non_rice_fields, {'color': 'ff0000'}, 'Non Rice Fields')  
Map
```



Hasil thresholding akan menghasilkan sebuah *binary image*, dimana nilai pixel 1 adalah sawah, dan nilai pixel 0 adalah non-sawah.

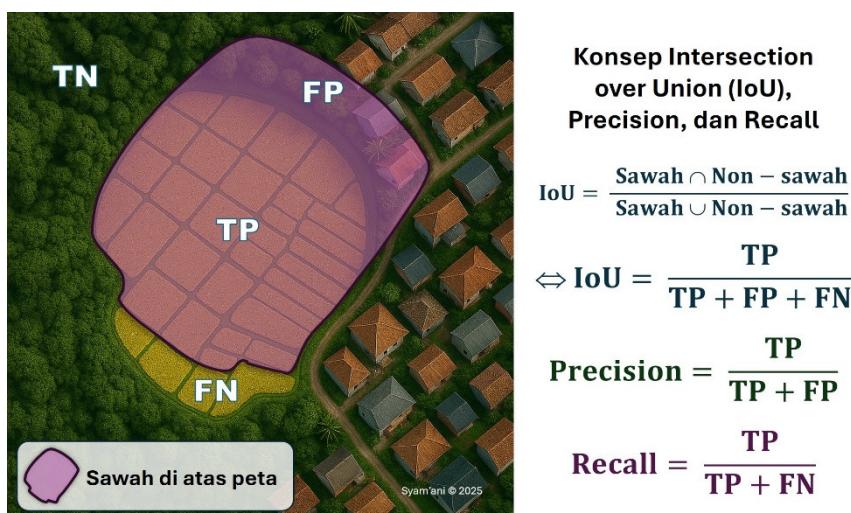
### Uji Akurasi Menggunakan Intersection Over Union (IoU)

Jika diperlukan, kita dapat menguji akurasi informasi geospasial sawah yang sudah kita ekstrak menggunakan analisis multitemporal. Objek sawah merupakan data *singleclass*, bukan data *multiclass* sebagaimana penutupan lahan. Sehingga informasi geospasial sawah tidak dapat diuji akurasinya menggunakan confusion matrix. Sebab untuk konstruksi matrix minimal diperlukan dua objek. Perhatikan skema uji akurasi data atribut pada Gambar 3.5 berikut:



Gambar 3.5. Uji akurasi data atribut informasi geospasial

Untuk informasi geospasial kualitatif *singleclass* seperti sebaran sawah, sebaran kebun karet, sebaran kebun sawit, genangan banjir, area terbakar, deforestasi, dan yang sejenisnya, kita dapat menguji akurasinya menggunakan *Intersection over Union* (IoU), atau dikenal juga sebagai *Jaccard Index* (Jaccard, 1901). Konsep dasar IoU sendiri dijelaskan pada Gambar 3.6 berikut:

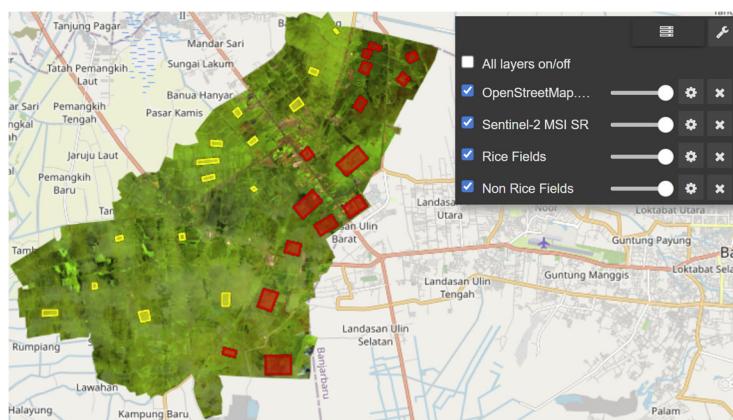


Gambar 3.6. Konsep dasar IoU, Precision, dan Recall (Copilot, 2025a)

#### Keterangan Gambar 3.6:

- True Positive (TP) : Terpetakan sebagai sawah, dan di lapangan sebenarnya memang sawah*  
*False Negative (FN) : Tidak terpetakan sebagai sawah, tetapi di lapangan sebenarnya sawah*  
*True Negative (TN) : Tidak terpetakan sebagai sawah, dan di lapangan memang bukan sawah*  
*False Positive (FP) : Terpetakan sebagai sawah, tetapi di lapangan sebenarnya bukan sawah*

Secara teknis kita memerlukan keberadaan sampel-sampel sawah dan non-sawah, sebagaimana sudah dijelaskan sebelumnya. Sampel-sampel ini didigitasi menggunakan perangkat lunak SIG. Untuk lebih detailnya dapat dilihat pada Gambar 3.7. Poligon-poligon yang berwarna kuning pada Gambar 3.7 merupakan sampel-sampel sawah. Sementara poligon-poligon yang berwarna merah pada Gambar 3.7 merupakan sampel-sampel non-sawah.



Gambar 3.7. Implementasi teknis uji akurasi menggunakan IoU

Silahkan lanjutkan dengan mengetikkan dan menjalankan kode-kode berikut:

```
# Konversi sampel sawah dan non-sawah menjadi image
# untuk keperluan uji akurasi

rf_sample_image = rice_fields.reduceToImage(
    properties=['Id'],
    reducer=ee.Reducer.first()
)

non_rf_sample_image = non_rice_fields.reduceToImage(
    properties=['Id'],
    reducer=ee.Reducer.first()
)
```

Berikutnya adalah pembuatan dan visualisasi area TP, FN, TN, dan FP dengan kode-kode berikut:

```
# Pembuatan area TP, FN, TN, dan FP
# menggunakan sampel sawah dan non-sawah

tp_area = rice_field_class.eq(1).And(rf_sample_image.eq(1)).rename('tp')
fn_area = rice_field_class.eq(0).And(rf_sample_image.eq(1)).rename('fn')
tn_area = rice_field_class.eq(0).And(non_rf_sample_image.eq(0)).rename('tn')
fp_area = rice_field_class.eq(1).And(non_rf_sample_image.eq(0)).rename('fp')

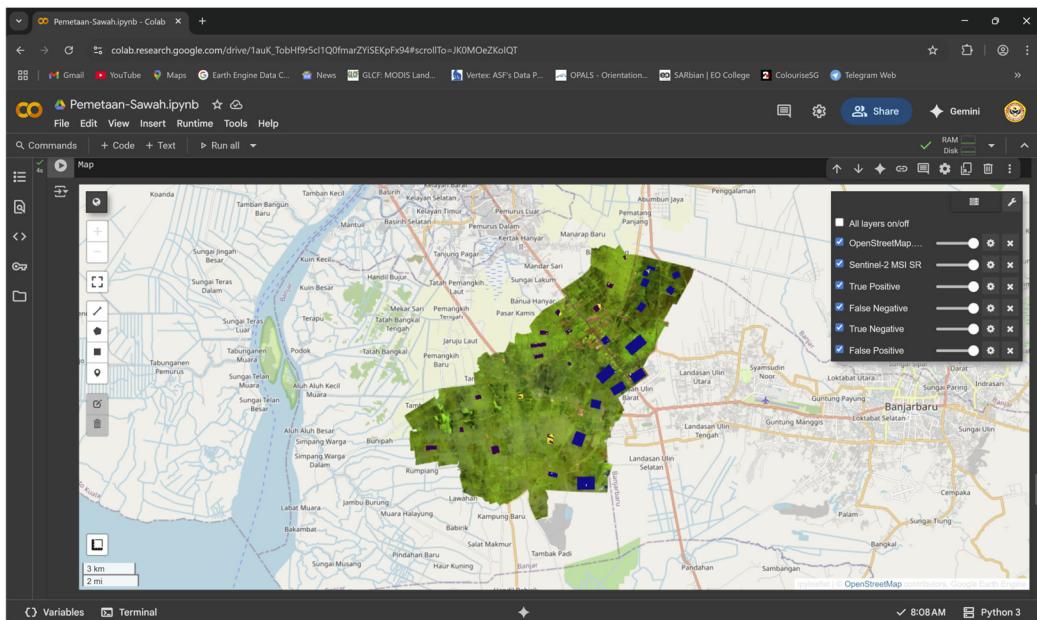
tp_area = tp_area.updateMask(tp_area.neq(0))
fn_area = fn_area.updateMask(fn_area.neq(0))
tn_area = tn_area.updateMask(tn_area.neq(0))
fp_area = fp_area.updateMask(fp_area.neq(0))
```

```
# visualisasi area TP, FN, TN, dan FP

tp_vis = {'min': 0, 'max': 1, 'palette': 'viridis_r'}
fn_vis = {'min': 0, 'max': 1, 'palette': 'viridis_r'}
tn_vis = {'min': 0, 'max': 1, 'palette': 'plasma_r'}
fp_vis = {'min': 0, 'max': 1, 'palette': 'white'}

Map = geemap.Map()
Map.centerObject(region, 12)
Map.addLayer(s2_image2, rgb_vis, 'Sentinel-2 MSI SR')
Map.addLayer(tp_area, tp_vis, 'True Positive')
Map.addLayer(fn_area, fn_vis, 'False Negative')
Map.addLayer(tn_area, tn_vis, 'True Negative')
Map.addLayer(fp_area, fp_vis, 'False Positive')
Map
```

Berikut adalah outputnya:



Perhatikan gambar output di atas, berikut adalah penjelasan masing-masing warnanya:

- Ungu** : TP (*Faktual di lapangan adalah sawah, dan terpetakan sebagai sawah*)
- Kuning** : FN (*Faktual di lapangan adalah sawah, tetapi terpetakan sebagai bukan sawah*)
- Biru** : TN (*Faktual di lapangan adalah bukan sawah, dan terpetakan sebagai bukan sawah*)
- Putih** : FP (*Faktual di lapangan adalah bukan sawah, tetapi terpetakan sebagai sawah*)

Berikutnya kita akan mengkalkulasikan dan menampilkan jumlah pixel pada masing-masing area TP, FN, TN, dan FP, dengan kode-kode berikut:

```
# Kalkulasi jumlah pixel masing-masing area TP, FN, TN, dan FP

tp_count = tp_area.reduceRegion(
    reducer=ee.Reducer.count(),
    geometry=region,
    scale=10
)

fn_count = fn_area.reduceRegion(
```

### Bab III Google Earth Engine

```
reducer=ee.Reducer.count(),
geometry=region,
scale=10
)

tn_count = tn_area.reduceRegion(
    reducer=ee.Reducer.count(),
    geometry=region,
    scale=10
)

fp_count = fp_area.reduceRegion(
    reducer=ee.Reducer.count(),
    geometry=region,
    scale=10
)
```

```
# Mendapatkan informasi hasil kalkulasi jumlah pixel TP, FN, TN, dan FP

tp = tp_count.get('tp'). getInfo()
fn = fn_count.get('fn'). getInfo()
tn = tn_count.get('tn'). getInfo()
fp = fp_count.get('fp'). getInfo()
```

```
# Menampilkan informasi jumlah pixel TP, FN, TN, dan FP

print(f'True Positive: {tp} pixels')
print(f'False Negative: {fn} pixels')
print(f'True Negative: {tn} pixels')
print(f'False Positive: {fp} pixels')
```

```
True Positive: 10866 pixels
False Negative: 2200 pixels
True Negative: 41395 pixels
False Positive: 287 pixels
```

Luas area IoU, Precision, dan Recall, dapat berupa jumlah pixel, sebagaimana contoh di atas, dapat juga berupa luasan dalam hektar misalnya, atau proporsi dalam desimal atau persen. Hal ini akan tergantung pada teknis uji akurasi IoU yang kita terapkan. Langkah terakhir adalah proses kalkulasi nilai-nilai IoU, Precision, dan Recall dengan kode-kode berikut:

```
# Kalkulasi IoU

iou = tp / (tp + fn + fp)
print(f'IoU: {iou}')
```

```
IoU: 0.8137497191642328
```

```
# Kalkulasi Precision

precision = tp / (tp + fp)
print(f'Precision: {precision}')
```

```
Precision: 0.9742670133596342
```

```
# Kalkulasi Recall

recall = tp / (tp + fn)
print(f'Recall: {recall}')
```

```
Recall: 0.8316240624521659
```

IoU 0,8137 berarti akurasi dari informasi geospasial sawah yang kita ekstrak menggunakan analisis multitemporal adalah 81,37%, atau error-nya lebih dari 18%. Precision identik dengan UA pada confusion matrix, sementara Recall identik dengan PA pada confusion matrix. Sehingga interpretasi kedua parameter ini pun sama. Precision 0,9743 berarti dari seluruh sawah yang sudah berhasil kita petakan, hanya 97,43% yang benar-benar sawah. Sisanya yang sekitar 2% lebih adalah objek bukan sawah yang teridentifikasi secara tidak sengaja sebagai sawah. Recall 0,8316 berarti, dari seluruh sawah yang ada di wilayah yang dipetakan, hanya 83,16% yang berhasil terpetakan sebagai sawah. Sisanya, hampir 17% sawah hilang dari peta.

Jika disimpulkan secara singkat, peta sebaran sawah yang kita hasilkan cenderung *under estimate*. Sebab sebaran sawah di atas peta cukup akurat, akan tetapi cukup banyak sawah yang ada di wilayah observasi yang tidak terpetakan. Potensi error utama dari metode yang kita gunakan untuk mengekstrak objek sawah ini adalah, jika ada objek-objek lain, tetumbuhan rawa misalnya, yang memiliki *spectral behavior* dan *temporal behavior* sama persis dengan sawah.

### Multiclass IoU

Faktanya, IoU juga dapat diterapkan pada data tematik *multiclass*, sebagaimana informasi geospasial penutupan lahan. Hal ini karena di dalam Confusion Matrix, *Precision* akan sama dengan *User's Accuracy* (UA), dan *Recall* akan sama dengan *Producer's Accuracy* (PA). Tentu saja, nilai IoU-nya nanti akan didapatkan per kelas, bukan untuk semua kelas sebagaimana OA. Silahkan cermati kembali formula IoU berikut (Jaccard, 1901):

$$IoU = \frac{TP}{TP + FP + FN}, \quad \forall IoU \in \mathbb{R}, 0 \leq IoU \leq 1$$

Notasi  $\forall IoU \in \mathbb{R}$  menunjukkan bahwa semua nilai IoU merupakan anggota dari bilangan nyata (*real numbers*). Dan notasi  $0 \leq IoU \leq 1$  menunjukkan bahwa IoU memiliki rentang nilai dari 0 hingga 1. Kalkulasi IoU memerlukan parameter TP, FP, dan FN. Untuk parameter FP dapat kita peroleh dengan merotasi persamaan Precision sebagai berikut:

$$\begin{aligned} Precision &= \frac{TP}{TP + FP} \\ \Leftrightarrow TP &= Precision(TP + FP) \\ \Leftrightarrow TP &= Precision \cdot TP + Precision \cdot FP \\ \Leftrightarrow Precision \cdot FP &= TP - Precision \cdot TP \\ \Leftrightarrow FP &= \frac{TP - Precision \cdot TP}{Precision} \\ \Leftrightarrow FP &= \frac{TP}{Precision} - TP \end{aligned}$$

Dan untuk parameter FN dapat kita peroleh dengan merotasi persamaan Recall sebagai berikut:

$$Recall = \frac{TP}{TP + FN}$$

$$\Leftrightarrow TP = Recall(TP + FN)$$

$$\Leftrightarrow TP = Recall \cdot TP + Recall \cdot FN$$

$$\Leftrightarrow Recall \cdot FN = TP - Recall \cdot TP$$

$$\Leftrightarrow FN = \frac{TP - Recall \cdot TP}{Recall}$$

$$\Leftrightarrow FN = \frac{TP}{Recall} - TP$$

Substitusi FP dan FN ke formula IoU sebagai berikut:

$$IoU = \frac{TP}{TP + \left( \frac{TP}{Precision} - TP \right) + \left( \frac{TP}{Recall} - TP \right)}$$

$$\Leftrightarrow IoU = \frac{TP}{\frac{TP}{Precision} + \frac{TP}{Recall} - TP}$$

Karena Precision ekivalen dengan UA, dan Recall ekivalen dengan PA, maka:

$$IoU = \frac{TP}{\frac{TP}{UA} + \frac{TP}{PA} - TP}$$

Di dalam confusion matrix, TP merupakan nilai-nilai yang terdapat pada diagonal matriks untuk setiap kelas. Sehingga kita dapat secara langsung mengambil nilai-nilai pada diagonal Confusion Matrix dalam kalkulasi IoU untuk setiap kelas. Akan tetapi, persamaan akhir di atas masih dapat disederhanakan. Yaitu dengan memfaktorkan TP sebagai berikut:

$$IoU = \frac{TP}{TP \left( \frac{1}{UA} + \frac{1}{PA} - 1 \right)}$$

$$\Leftrightarrow IoU = \frac{1}{\frac{1}{UA} + \frac{1}{PA} - 1}$$

Dengan formula yang lebih simpel di atas, IoU dapat dihitung untuk setiap kelas di dalam Confusion Matrix. Yaitu cukup dengan menggunakan nilai-nilai UA dan PA untuk setiap kelas, tanpa perlu mengambil nilai-nilai pada diagonal Confusion Matrix.

---

*Dalam konteks ekstraksi informasi geospasial, implementasi IoU pada data multiclass kurang umum, hal ini dikarenakan kurang praktis. Sebaiknya untuk data geospasial multiclass, sebagaimana penutupan lahan, cukup menggunakan Confusion Matrix saja. Terapkan IoU hanya pada data geospasial singleclass.*

---

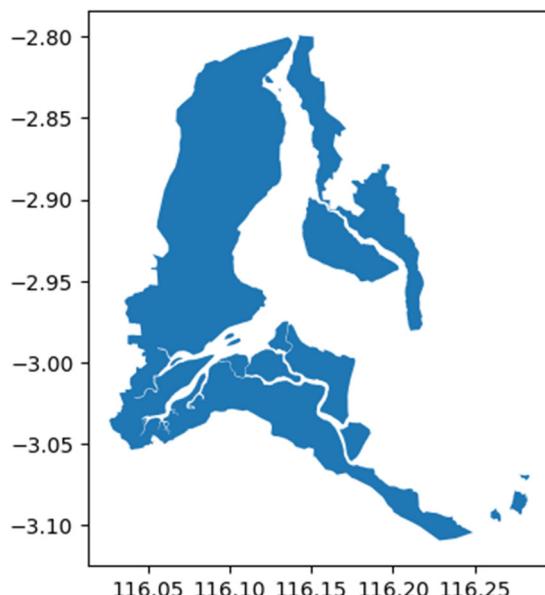
## Memantau Dinamika Hutan Mangrove

Sebagaimana sudah diulas pada bagian terdahulu, bahwa salah satu aplikasi analisis multitemporal yang umum dilakukan adalah memantau dinamika objek di atas permukaan bumi. Pada latihan kali ini, kita akan memantau dinamika hutan mangrove pada Cagar Alam Teluk Kelumpang, Kabupaten Kotabaru, Kalimantan Selatan. Silahkan buat sebuah notebook Google Colab baru, kemudian beri nama misalnya **Mangrove\_Monitoring\_Project.ipynb**. Kemudian ketikkan dan jalankan kode-kode berikut:

```
import ee
ee.Authenticate()
# Initialize your Earth Engine cloud project
ee.Initialize(project='ee-geospatialulm')

# Access google drive and open a shapefile
import geopandas as gpd
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
path = '/content/drive/My Drive/geebook/'
shapefile_path = path + 'vector/Kelumpang.shp'

# open shapefile using Geopandas
try:
    gdf = gpd.read_file(shapefile_path)
    gdf.plot()
except FileNotFoundError:
    print(f"Error: Shapefile not found at {shapefile_path}")
except Exception as e:
    print(f"An error occurred: {e}")
```



### Bab III Google Earth Engine

Pada kode-kode di atas, kita mengakses shapefile menggunakan *try and exception*. Hal ini untuk menghindari *crash* pada kode program ketika shapefile tidak ditemukan, atau terjadi kesalahan di dalam penulisan path atau nama file dari shapefile. Lihat kembali pembahasan tentang penanganan kesalahan dan pengecualian sebelumnya pada halaman 100.

```
# Converting shapefile into Earth Engine geometry
import json
```

```
region_js = json.loads(gdf.to_json())
region_fc = ee.FeatureCollection(region_js)
region = ee.Geometry(region_fc.geometry())
```

```
# Region center
region_center = region.centroid().coordinates().getInfo()
print(region_center)
```

```
[116.1186604102572, -2.9461974517515026]
```

Kode-kode di atas digunakan untuk mengkonversi shapefile menjadi geometri GEE, sekaligus mencari titik tengah wilayah (*region center*) Teluk Kelumpang. Berikutnya kita akan menentukan beberapa parameter, yang meliputi jumlah tahun pemantauan dan akhir tahun pemantauan. Dalam hal ini kita tentukan pemantauan dinamika hutan mangrove selama enam tahun sampai tahun 2024. Yang berarti dimulai dari tahun 2019. Silahkan jalankan kode-kode berikut:

```
# Determining parameters
num_years = 6
last_year = 2024

start_date = []
end_date = []
years = []

for i in range(num_years):
    year = last_year - num_years + i + 1
    start_date.append(f'{year}-01-01')
    end_date.append(f'{year}-12-31')
    years.append(year)

print(start_date)
print(end_date)
print(years)
```

```
['2019-01-01', '2020-01-01', '2021-01-01', '2022-01-01', '2023-01-01', '2024-01-01']
['2019-12-31', '2020-12-31', '2021-12-31', '2022-12-31', '2023-12-31', '2024-12-31']
[2019, 2020, 2021, 2022, 2023, 2024]
```

Berikutnya kita akan mengakses Citra Sentinel-2 multitemporal dengan kode-kode berikut:

```
# Access multitemporal Sentinel-2 surface reflectance
s2_col = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')

s2_image_list = []

for i in range(num_years):
    dataset = s2_col.filterBounds(region) \
        .filterDate(start_date[i], end_date[i]) \
        .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20))
    s2_image_list.append(dataset.median())
```

Di dalam latihan ini, kita akan menggunakan transformasi *Mangrove Vegetation Index* (MVI) untuk memetakan sebaran hutan mangrove. Formula MVI untuk Citra Sentinel-2 MSI adalah sebagai berikut (Baloloy et al., 2020):

$$MVI = \frac{NIR - Green}{SWIR1 - Green}$$

Menurut Baloloy et al. (2020), pada Citra Sentinel-2, mangrove akan memiliki rentang nilai MVI dari 4,5 hingga 16,5. Di luar rentang nilai ini berarti bukan mangrove. Berikut adalah kode-kode implementasi MVI sekaligus thresholding nilai-nilai MVI untuk menghasilkan data sebaran hutan mangrove multitemporal:

```
# Calculate multitemporal Mangrove Vegetation Index (MVI)
# Creating mangrove extent image

mvi_list = []
mgrv_list = []

for i in range(num_years):
    green = s2_image_list[i].select('B3')
    nir = s2_image_list[i].select('B8')
    swir1 = s2_image_list[i].select('B11')
    mvi = (nir.subtract(green)).divide(swir1.subtract(green)).rename('MVI')
    mask = mvi.gte(4.5).And(mvi.lte(16.5)).rename('Mangrove')
    mvi = mvi.updateMask(mask)
    mvi = mvi.clip(region)
    mvi_list.append(mvi)
    mgrv_list.append(mask)
```

Pada kode-kode di atas, data MVI multitemporal disimpan ke dalam list `mvi_list` dan data sebaran hutan mangrove multitemporal disimpan ke dalam list `mgrv_list`. Langkah berikutnya, kita dapat memvisualisasikan MVI multitemporal.

```
# Create geemap MVI timeseries

import geemap

# visualization parameters
vis_params = {
    'min': 4.5,
    'max': 16.5,
    'palette': ['seagreen', 'darkgreen', 'darkblue']
}

# Create the MVI time series visualization using geemap
Map = geemap.Map(basemap='Esri.WorldTopoMap')
Map.centerobject(region, 11)

# Create image collection from image list
mvi_col = ee.ImageCollection.fromImages(mvi_list)

# Create a list of year labels corresponding to the MVI images
labels = [f'Tahun {year}' for year in years]

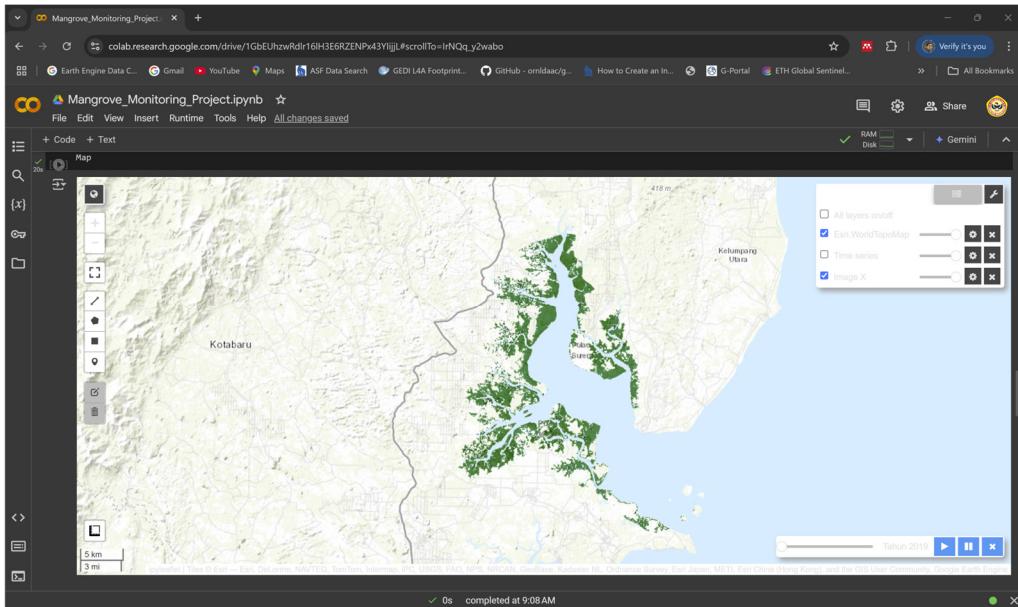
# Add a layer control panel
Map.add_time_slider(mvi_col, vis_params, labels=labels, time_interval=1)

# Display the map
Map
```

Pada kode-kode di atas, terlebih dahulu `mvi_list` dikonversi menjadi image collection, yaitu dengan instruksi `mvi_col = ee.ImageCollection.fromImages(mvi_list)`. MVI

### Bab III Google Earth Engine

multitemporal divisualisasikan ke dalam bentuk *time slider* (`Map.add_time_slider...`) agar kita dapat mengamati dengan cara menggeser-geser tampilan MVI antar tahun. Perhatikan outputnya seperti pada gambar di bawah:



Langkah terakhir adalah kalkulasi dan menampilkan informasi luasan hutan mangrove setiap tahun. Silahkan ketikkan dan jalankan kode-kode berikut:

```
# Mangrove area calculation
for i in range(num_years):
    area = mgrv_list[i].reduceRegion(
        reducer=ee.Reducer.sum(),
        geometry=region,
        scale=10,
        maxPixels=1e13
    ).get('Mangrove'). getInfo()
    print(f'Mangrove area in {years[i]}: {round(area/100,2)} hectares')
```

```
Mangrove area in 2019: 12498.7 hectares
Mangrove area in 2020: 8851.7 hectares
Mangrove area in 2021: 12442.26 hectares
Mangrove area in 2022: 11725.49 hectares
Mangrove area in 2023: 7952.13 hectares
Mangrove area in 2024: 8460.97 hectares
```

### Membuat Timelapse Animation Hutan Mangrove

Selain observasi dinamika hutan mangrove dari tahun ke tahun dalam format statik, berikut kalkulasi luasannya, kita juga dapat memvisualisasikan sebaran hutan mangrove hasil ekstraksi menggunakan MVI ke dalam bentuk animasi atau gambar bergerak. Visualisasi dalam bentuk animasi tentu saja akan lebih menarik, dan dalam konteks tertentu akan lebih informatif. Misalnya kita dapat melihat secara langsung di lokasi-lokasi mana saja terjadi pengurangan atau penambahan luasan hutan mangrove. Kita masih melanjutkan kode-kode di atas, silahkan lanjutkan dengan mengetikkan dan menjalankan kode-kode berikut:

```
# Determining animation properties and export animation into gif file
video_args = {
    'dimensions': 768,
    'region': region,
    'framesPerSecond': 1,
    'crs': 'EPSG:4326',
    'min': 4.5,
    'max': 16.5,
    'palette': ['seagreen', 'darkgreen', 'darkblue'],
}
saved_gif = path + 'output/mangrove_timelapse.gif'
geemap.download_ee_video(mvi_col, video_args, saved_gif)
```

Data yang akan menjadi input di dalam pembuatan animasi adalah image collection, dalam hal ini adalah `mvi_col`. Kode-kode di atas digunakan dalam penentuan property-properti animasi, sekaligus mengekspor animasi ke dalam format file GIF. '`dimensions`' : `768` adalah ukuran lebar animasi dalam satuan pixel. Disarankan kita menginput lebarnya saja, sebab nanti tinggi gambar animasinya akan menyesuaikan secara proporsional terhadap pixel-pixel citra. `geemap.download_ee_video(mvi_col, video_args, saved_gif)` adalah instruksi untuk mengekspor animasi `mvi_col` ke dalam bentuk GIF. File akan tersimpan dengan `path` dan nama dari `saved_gif = path + 'output/mangrove_timelapse.gif'`.

Berikutnya, kita dapat menambahkan teks-teks tertentu, bahkan gambar atau logo ke dalam animasi. Sebagai contoh, kita akan menambahkan teks statik berupa judul animasi, dan teks dinamik yang menampilkan tahun. Silahkan ketikkan dan jalankan kode-kode berikut:

```
# Add title to gif animation
title = 'Dinamika Mangrove Teluk Kelumpang'
geemap.add_text_to_gif(
    saved_gif,
    saved_gif,
    xy=('55%', '2%'),
    text_sequence=title,
    font_size=16,
    font_color='#ffffff',
    add_progress_bar=False,
)
```

```
# Add dynamic text to gif animation and display gif animation
geemap.add_text_to_gif(
    saved_gif,
    saved_gif,
    xy=('81%', '5%'),
    text_sequence=labels,
    font_size=20,
    font_color='#ffffcc',
    add_progress_bar=False,
    duration=1000,
)
geemap.show_image(saved_gif)
```

Pada kode-kode di atas, `saved_gif` yang pertama adalah mengakses animasi GIF yang sudah disimpan sebelumnya. Sementara `saved_gif` yang kedua adalah tujuan penyimpanan kembali animasi setelah ditambahkan teks. Karena `saved_gif` diulang dua kali, artinya kita akan mengganti atau menimpa file GIF animasi yang lama dengan yang baru (setelah ditambahkan teks). Jika diinginkan, tentu saja Anda dapat membuat `path` atau `nama file` yang berbeda. `xy='55%', '2%'` adalah posisi awal (bagian kiri) teks yang ditambahkan. Posisi dihitung

### Bab III Google Earth Engine

dari sudut kanan-atas. `xy=('55%', '2%')` berarti teks Dinamika Mangrove Teluk Kelumpang (seperti gambar di bawah) dimulai dari posisi 55% dari sebelah kiri, dan 2% dari atas. Dalam hal ini, posisi 50% berarti tepat di tengah-tengah gambar animasi. Dan untuk mendapatkan posisi yang terbaik, faktanya kita harus *trial and error*. `text_sequence=labels` merupakan instruksi untuk menambahkan label tahun dinamik, dengan menggunakan instruksi `labels = [f'Tahun {year}' for year in years]` pada kode terdahulu.

Parameter `duration=1000` merupakan durasi animasi per frame dalam satuan milidetik. Default durasinya adalah 100 milidetik per frame. Dalam konteks animasi per tahun, berarti satu frame identik dengan satu tahun. Sehingga `duration=1000` berarti satu tahun akan ditampilkan selama 1.000 milidetik, alias 1 detik. Sehingga animasi selama 6 tahun akan ekivalen dengan 6 detik. Terkait dengan hal ini, penambahan parameter `duration` direkomendasikan untuk dilakukan pada penambahan elemen terakhir pada animasi. Perhatikan pada kode-kode sebelumnya, dimana pada penambahan judul animasi tidak ada parameter `duration`. `duration` baru ditambahkan ketika menambahkan teks tahun, dan setelah penambahan teks tahun tidak ada lagi penambahan elemen pada animasi.

Berikut adalah output animasinya yang tersimpan di dalam Google Drive dalam bentuk GIF:



Karena file animasinya dalam format GIF, tentu saja sifatnya sangat portabel. Animasi dapat ditampilkan di laman web, atau dikirim ke media sosial.

**TUGAS**, coba Anda buat animasi dinamika garis pantai di suatu wilayah yang Anda inginkan, dengan menggunakan MNDWI atau pun indeks-indeks spektral sejenis.

## Pixelwise Change Detections

Pixelwise change detections (Singh, 1989) atau deteksi perubahan berbasis pixel merupakan metode untuk mengidentifikasi perubahan intensitas (nilai spektral) citra pixel per pixel menurut rentang waktu tertentu. Potensi aplikasi metode ini cukup banyak, misalnya observasi perubahan vegetasi, pemantauan perkembangan permukiman, hingga ekstraksi area terbakar dan pemetaan banjir. Terdapat cukup banyak teknik deteksi perubahan berbasis pixel, diantaranya adalah *image differencing*, *image rationing*, *image regression*, *Change Vector Analysis* (CVA), median filter, pixelwise fuzzy, dan sebagainya. Teknik yang akan dibahas di dalam buku ini hanya ada empat, yaitu image differencing, image rationing, image regression, dan CVA. Silahkan buat sebuah notebook Google Colab baru, kemudian ketikkan dan jalankan kode-kode berikut:

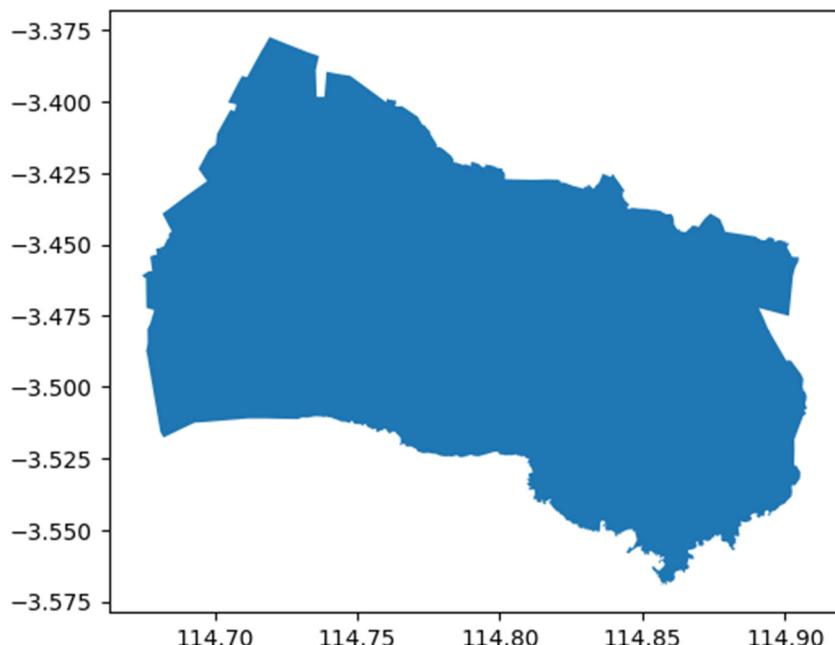
```
import ee, geemap

ee.Authenticate()

ee.Initialize(project='ee-geospatialulm')

# Mengakses Google Drive
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

# Membuka shapefile wilayah observasi
import geopandas as gpd
path = '/content/drive/My Drive/geebook/vector/'
region_shp = gpd.read_file(path + 'Banjarbaru.shp')
region_shp.plot()
```



### Bab III Google Earth Engine

```
# Konversi shapefile menjadi Earth Engine geometry  
region = geemap.geopandas_to_ee(region_shp).geometry()
```

```
# Penentuan tanggal akuisisi citra  
# Waktu akuisisi 1 (t1)  
start_date_image1 = '2019-01-01'  
end_date_image1 = '2019-12-31'  
  
# Waktu akuisisi 2 (t2)  
start_date_image2 = '2024-01-01'  
end_date_image2 = '2024-12-31'
```

```
# Akses Sentinel-2 image collection t1 dan t2
```

```
s2_col1 = (  
    ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')  
    .filterDate(start_date_image1, end_date_image1)  
    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 50))  
)  
  
s2_col2 = (  
    ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED')  
    .filterDate(start_date_image2, end_date_image2)  
    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 50))  
)
```

```
# Reduksi image collection menjadi image dan memotong citra
```

```
s2_image1 = s2_col1.median().clip(region).divide(10000)  
s2_image2 = s2_col2.median().clip(region).divide(10000)
```

```
# Mengambil band-band yang diperlukan
```

```
red1 = s2_image1.select('B4').rename('red')  
nir1 = s2_image1.select('B8').rename('nir')  
swir21 = s2_image1.select('B12').rename('swir')  
  
red2 = s2_image2.select('B4').rename('red')  
nir2 = s2_image2.select('B8').rename('nir')  
swir22 = s2_image2.select('B12').rename('swir')
```

```
# Transformasi NDVI untuk t1 dan t2
```

```
ndvi1 = nir1.subtract(red1).divide(nir1.add(red1)).rename('NDVI')  
ndvi2 = nir2.subtract(red2).divide(nir2.add(red2)).rename('NDVI')
```

### Image Differencing

Prinsip dasar image differencing cukup sederhana, yaitu kita sambil satu band dari citra bitemporal, misalnya citra akuisisi tahun 2019 ( $t_1$ ) dan citra akuisisi tahun 2024 ( $t_2$ ). Band yang diambil bisa berupa band original citra, misalnya NIR, maupun band hasil transformasi seperti NDVI. Kemudian kita kurangkan antara band pada ( $t_2$ ) dan band pada ( $t_1$ ). Formula dasarnya adalah sebagai berikut (Ilsever and Unsalan, 2012):

$$I_d(x,y) = I_{t2}(x,y) - I_{t1}(x,y)$$

Dimana:

- $I_{t1}(x,y)$  : Intensitas suatu band dari citra waktu akuisisi  $t_1$  untuk setiap lokasi pixel  $(x,y)$
- $I_{t2}(x,y)$  : Intensitas suatu band dari citra waktu akuisisi  $t_2$  untuk setiap lokasi pixel  $(x,y)$
- $I_d(x,y)$  : Intensitas citra hasil image differencing untuk setiap lokasi pixel  $(x,y)$

Akan tetapi, formula dasar image differencing seperti di atas hanya akan memberikan hasil yang akurat di bawah kondisi yang ideal. Kondisi yang ideal yang dimaksud adalah, pertama, kalibrasi dan koreksi citra dilakukan dengan benar. Baik koreksi geometrik (tidak ada pergeseran pixel antar waktu), termasuk koreksi radiometrik dan atmosferik, sehingga nilai spektral citra benar-benar nilai *surface reflectance* absolut objek. Kedua, pixel-pixel yang tidak berubah di atas citra lainnya harus sama persis pada kedua citra, sehingga ketika dikurangkan hasilnya 0. Faktanya, kondisi ideal ini mungkin sulit untuk dipenuhi. Sebab pencitraan permukaan bumi pada waktu yang berbeda akan memiliki kondisi yang berbeda juga. Baik karena faktor lingkungan, seperti intensitas dan sudut peninjauan matahari, maupun faktor objeknya sendiri, seperti dedaunan atau tajuk pohon yang berubah posisi karena tertutup angin. Sehingga salah satu citra harus disesuaikan atau dinormalisasi intensitasnya. Dalam hal ini, kita akan menormalisasi citra  $t_1$  dengan formula berikut (dimodifikasi dari Ilsever and Unsalan, 2012):

$$\tilde{I}_{t1}(x, y) = \frac{\sigma_{t2}}{\sigma_{t1}} (I_{t1}(x, y) - \mu_{t2}) + \mu_{t1}$$

Dimana:

- $\tilde{I}_{t1}(x, y)$  : Intensitas hasil normalisasi suatu band dari citra  $t_1$  untuk setiap lokasi pixel  $(x, y)$
- $\sigma_{t1}$  : Standar deviasi nilai spektral suatu band dari citra  $t_1$
- $\sigma_{t2}$  : Standar deviasi nilai spektral suatu band dari citra  $t_2$
- $\mu_{t1}$  : Rerata nilai spektral suatu band dari citra  $t_1$
- $\mu_{t2}$  : Rerata nilai spektral suatu band dari citra  $t_2$

Formula di atas sebenarnya merupakan formula yang kita gunakan di dalam transformasi temporal citra pada bagian terdahulu (lihat halaman 218). Selanjutnya, citra  $t_1$  yang sudah ternormalisasi, yaitu  $\tilde{I}_{t1}(x, y)$ , akan digunakan untuk mengganti  $I_{t1}(x, y)$  (citra  $t_1$  original) di dalam formula image differencing (Ilsever and Unsalan, 2012):

$$I_d(x, y) = I_{t2}(x, y) - \tilde{I}_{t1}(x, y)$$

Berikut adalah kode-kode implementasi image differencing:

```
# Kalkulasi parameter-parameter image differencing
ndvi1_mean = ndvi1.reduceRegion(
    reducer=ee.Reducer.mean(),
    geometry=region,
    scale=10
).get('NDVI'). getInfo()

ndvi2_mean = ndvi2.reduceRegion(
    reducer=ee.Reducer.mean(),
    geometry=region,
    scale=10
).get('NDVI'). getInfo()

ndvi1_sd = ndvi1.reduceRegion(
    reducer=ee.Reducer.stdDev(),
    geometry=region,
    scale=10
).get('NDVI'). getInfo()

ndvi2_sd = ndvi2.reduceRegion(
    reducer=ee.Reducer.stdDev(),
    geometry=region,
    scale=10
).get('NDVI'). getInfo()
```

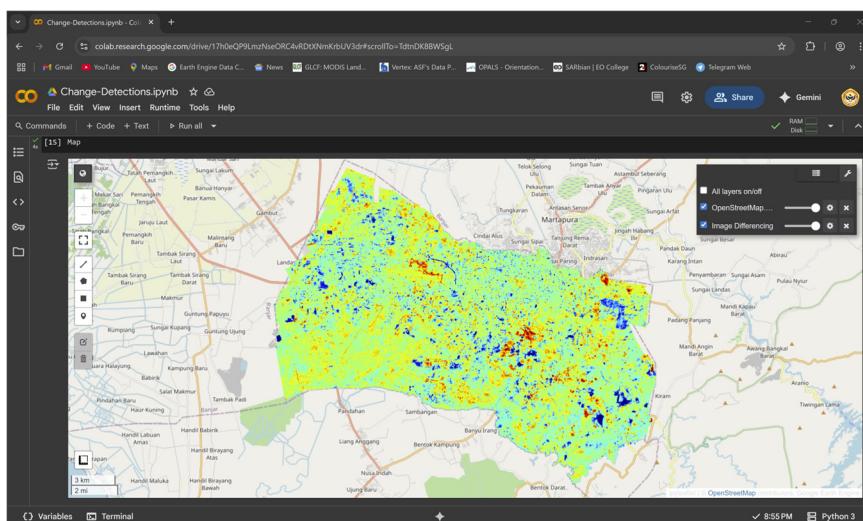
### Bab III Google Earth Engine

```
# Normalisasi NDVI t1  
ndvi1_norm = (  
    ((ndvi1.subtract(ndvi1_mean))  
     .multiply(ndvi2_sd / ndvi1_sd))  
     .add(ndvi2_mean)  
)
```

```
# Kalkulasi image differencing untuk NDVI  
image_diff = ndvi2.subtract(ndvi1_norm)
```

```
# Visualisasi citra hasil image differencing  
id_vis = {'min': -0.5, 'max': 0.5, 'palette': 'jet'}  
  
Map = geemap.Map()  
Map.centerObject(region, zoom=12)  
Map.addLayer(image_diff, id_vis, 'Image Differencing')  
Map
```

Berikut adalah output citra hasil image differencing:



Langkah berikutnya adalah menentukan pixel-pixel yang mengalami perubahan. Jika NDVI yang kita kurangkan memiliki nilai dari -1 sampai 1, maka secara teoritis citra hasil image differencing  $I_d(x, y)$  akan memiliki rentang nilai pixel dari -2 hingga 2. Hal ini merupakan satu kelebihan image differencing, sebab teknik ini dapat menangkap perubahan negatif maupun positif, misalnya vegetasi semakin bertambah atau berkurang. Idealnya, pixel-pixel yang tidak mengalami perubahan dari  $t_1$  ke  $t_2$  akan menghasilkan pixel bernilai 0 pada  $I_d(x, y)$ . Akan tetapi, di dunia nyata sepertinya hal ini sulit untuk dipenuhi, dengan sebab-sebab yang sudah dijelaskan sebelumnya, yaitu faktor lingkungan dan kondisi objek yang ada di dalam pixel. Sehingga untuk menentukan pixel-pixel yang mengalami perubahan, kita harus menetapkan nilai pembatas atau threshold ( $\tau$ ) tertentu. Karena image differencing dapat menangkap perubahan negatif dan positif, maka nilai threshold-nya dapat diasumsikan dengan  $-\tau$  dan  $\tau$ .

Nilai  $\tau$  dapat ditentukan secara empiris, baik dengan observasi citra, atau dengan sampel survey lapangan. Nilai  $\tau$  juga dapat ditentukan secara statistik, misalnya dengan menggunakan *Otsu thresholding* (Otsu, 1979). Lebih lanjut, nilai  $\tau$  juga dapat berbeda untuk perubahan negatif dan perubahan positif, sehingga dapat dinotasikan sebagai  $\tau_n$  dan  $\tau_p$ . Sehingga logika thresholding citra hasil image differencing dapat diformulasikan dengan fungsi kondisional sebagai berikut:

$$I_c(x, y) = \begin{cases} 1, & \text{if } I_d(x, y) \geq \tau_p \\ -1, & \text{if } I_d(x, y) \leq \tau_n \\ 0, & \text{Otherwise} \end{cases}$$

Dimana:

- $I_c(x, y)$  : Intensitas citra hasil thresholding (change detection) untuk setiap lokasi pixel  $(x, y)$
- $\tau_n$  : Nilai threshold untuk perubahan negatif
- $\tau_p$  : Nilai threshold untuk perubahan positif

Pada fungsi kondisional thresholding di atas, perubahan positif akan diberi nilai 1, perubahan negatif akan diberi nilai -1, dan pixel-pixel yang tidak mengalami perubahan akan diberi nilai 0. Karena metode Otsu thresholding hanya bisa mencari satu nilai  $\tau$  untuk sekali kalkulasi, maka proses kalkulasi  $\tau_n$  dan  $\tau_p$  harus dilakukan secara terpisah atau dua kali proses. Sebelum proses thresholding, direkomendasikan untuk menampilkan atau mengevaluasi secara visual histogram nilai-nilai pixel citra yang akan dithresholding. Silahkan ketikkan dan jalankan kode-kode berikut:

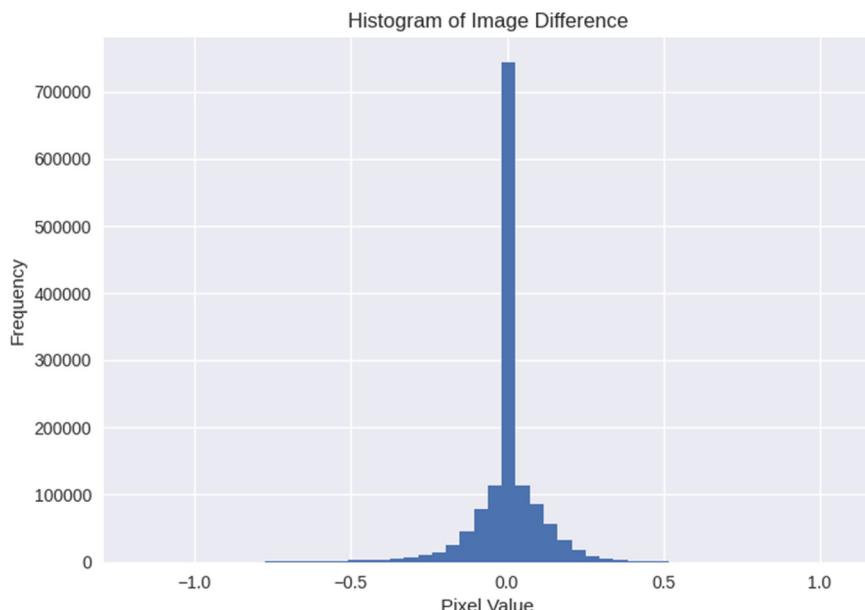
```
# Pembuatan histogram citra hasil image differencing
import matplotlib.pyplot as plt
import numpy as np

plt.style.use('seaborn-v0_8')

# Konversi GEE image ke NumPy array
image_diff_np = geemap.ee_to_numpy(image_diff, region=region, scale=20)

# Menghapus potensi nilai-nilai Nan
image_diff_np = image_diff_np[~np.isnan(image_diff_np)]

# Menampilkan histogram
plt.hist(image_diff_np, bins=50)
plt.title('Histogram of Image Difference')
plt.xlabel('Pixel value')
plt.ylabel('Frequency')
plt.show()
```



### Bab III Google Earth Engine

Perangkat lunak pemrosesan citra digital pada umumnya akan menampilkan histogram sebagaimana gambar di atas pada saat proses thresholding. Terkait penjelasan pembuatan histogram menggunakan Matplotlib, selengkapnya dapat dibaca di tautan [https://matplotlib.org/stable/api/\\_as\\_gen/matplotlib.pyplot.hist.html](https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html). Sebelum pembuatan histogram, citra hasil image differencing harus dikonversi terlebih dahulu ke NumPy array dengan instruksi `geemap.ee_to_numpy(image_diff, region=region, scale=20)`. Jika ada potensi nilai-nilai pixel yang tidak terdefinisi, terlebih dahulu harus dibuang dengan instruksi `image_diff_np[~np.isnan(image_diff_np)]`. Pada gambar di atas dapat terlihat bahwa puncak histogram berada di angka 0. Angka 0 dan area di sekitarnya merupakan pixel-pixel yang tidak mengalami perubahan. Karena proses thresholding dilakukan dua kali, yaitu threshold negatif dan threshold positif, maka langkah berikutnya adalah kita membagi dua data, yaitu nilai-nilai pixel yang kurang dari nol dan nilai-nilai pixel yang lebih dari nol.

```
# Partisi data menjadi perubahan negatif dan perubahan positif
image_diff_np_lt0 = image_diff_np[image_diff_np < 0]
image_diff_np_gt0 = image_diff_np[image_diff_np > 0]
```

Setelah partisi data menjadi dua bagian, Otsu threshold dijalankan pada masing-masing partisi dengan menggunakan Scikit-Image (Van der Walt et al., 2014). Sebagaimana kode-kode berikut:

```
# Otsu thresholding menggunakan Scikit-Image
from skimage.filters import threshold_otsu

# Otsu thresholding untuk perubahan negatif
otsu_neg = threshold_otsu(image_diff_np_lt0)
print(f"otsu threshold negatif: {otsu_neg}")

# Otsu thresholding untuk perubahan positif
otsu_pos = threshold_otsu(image_diff_np_gt0)
print(f"otsu threshold positif: {otsu_pos}")
```

Output:

```
otsu threshold negatif: -0.22339550780342105
otsu threshold positif: 0.1339247616581788
```

Setelah nilai-nilai Otsu threshold negatif dan Otsu threshold positif di temukan, selanjutnya fungsi kondisional dijalankan untuk menghasilkan citra hasil deteksi perubahan.

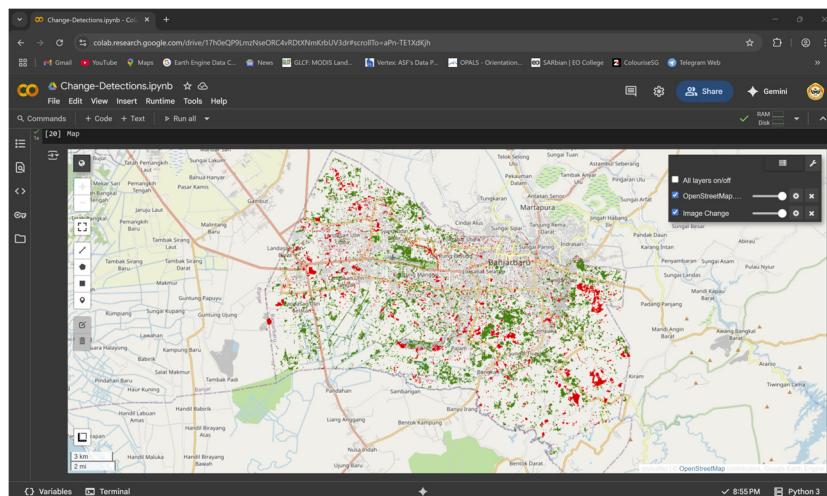
```
# Thresholding citra hasil image differencing
image_change = (
    image_diff.lte(otsu_neg).multiply(-1)
    .add(image_diff.gte(otsu_pos)).multiply(1)
)

mask = image_change.neq(0)
image_change = image_change.updateMask(mask)
```

Langkah terakhir adalah visualisasi citra hasil deteksi perubahan:

```
# Visualisasi citra hasil thresholding image differencing
ch_vis = {'min': -1, 'max': 1, 'palette': ['red','green']}
Map = geemap.Map()
Map.centerObject(region, zoom=12)
Map.addLayer(image_change, ch_vis, 'Image change')
Map
```

Berikut adalah output visualisasi citra hasil deteksi perubahan menggunakan image differencing:



Warna hijau pada gambar di atas merupakan perubahan positif. Karena citra yang diimage differencing adalah NDVI, maka perubahan positif berarti penambahan vegetasi atau reforestasi. Sedangkan warna merah merupakan perubahan negatif, yang kalau diterjemahkan berarti pengurangan vegetasi atau deforestasi.

### Image Rationing

Image rationing memiliki parameter-parameter yang sama dengan image differencing. Kecuali formulanya yang berbeda, sebagaimana persamaan berikut (Ilsever and Unsalan, 2012):

$$I_r(x, y) = \arctan\left(\frac{I_{t2}(x, y)}{I_{t1}(x, y)}\right) - \frac{\pi}{4}$$

Dimana:

$I_r(x, y)$  : Intensitas citra hasil image rationing untuk setiap lokasi pixel  $(x, y)$

Formula di atas merupakan bentuk normalisasi dari persamaan pembagian biasa  $\left(\frac{I_{t2}(x, y)}{I_{t1}(x, y)}\right)$ . Sebab pembagian atau rasio biasa akan menghasilkan rentang bilangan dari  $-\infty$  hingga  $\infty$ . Dengan mentransformasi persamaan rasio menjadi bentuk trigonometri  $\left(\arctan\left(\frac{I_{t2}(x, y)}{I_{t1}(x, y)}\right) - \frac{\pi}{4}\right)$ , rentang bilangannya akan menjadi  $-\frac{\pi}{4}$  hingga  $\frac{\pi}{4}$ . Masih melanjutkan kode-kode image differencing sebelumnya, silahkan lanjutkan dengan kode-kode berikut:

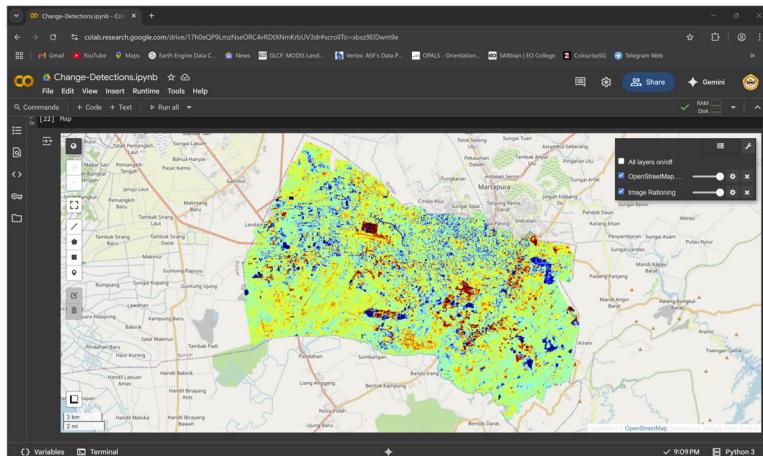
```
# Kalkulasi image rationing untuk NDVI
import math

image_ratio = (
    (ndvi2.divide(ndvi1_norm)
     .rename('image_ratio')).atan()
    .subtract(math.pi/4)
)
```

### Bab III Google Earth Engine

```
# Visualisasi citra hasil image rationing
ir_vis = {'min': -math.pi/8, 'max': math.pi/8, 'palette': 'jet'}
Map = geemap.Map()
Map.centerObject(region, zoom=12)
Map.addLayer(image_ratio, ir_vis, 'Image Rationing')
Map
```

Berikut adalah output citra hasil image rationing:

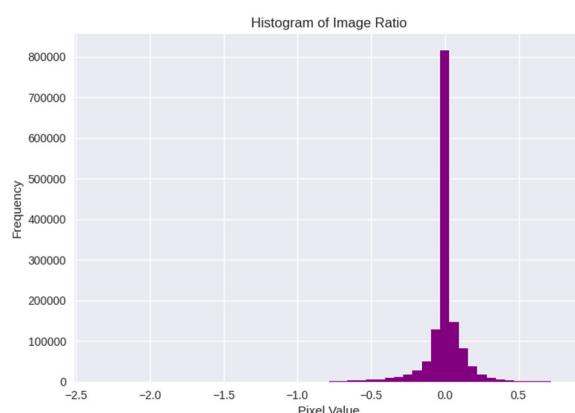


Sama seperti image differencing, pada image rationing kita juga akan membuat histogram dan melakukan Otsu thresholding untuk perubahan negatif dan perubahan positif.

```
# Pembuatan histogram citra hasil image rationing
# Konversi GEE image ke NumPy array
image_ratio_np = geemap.ee_to_numpy(image_ratio, region=region, scale=20)

# Menghapus potensi nilai-nilai NaN
image_ratio_np = image_ratio_np[~np.isnan(image_ratio_np)]

# Menampilkan histogram
plt.hist(image_ratio_np, bins=50, color='purple')
plt.title('Histogram of Image Ratio')
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')
plt.show()
```



```
# Partisi data menjadi perubahan negatif dan perubahan positif
image_ratio_np_lt0 = image_ratio_np[image_ratio_np < 0]
image_ratio_np_gt0 = image_ratio_np[image_ratio_np > 0]

# Otsu thresholding untuk perubahan negatif
otsu_neg = threshold_otsu(image_ratio_np_lt0)
print(f"Otsu threshold negatif: {otsu_neg}")

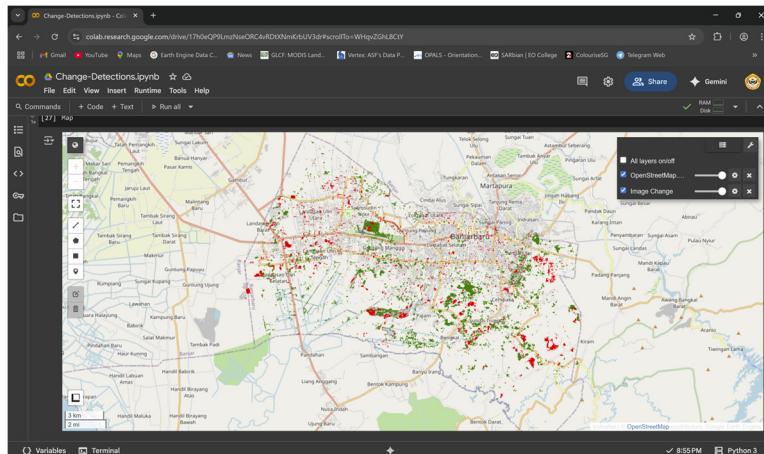
# Otsu thresholding untuk perubahan positif
otsu_pos = threshold_otsu(image_ratio_np_gt0)
print(f"Otsu threshold positif: {otsu_pos}")

# Thresholding citra hasil image rationing
threshold = 0.2
image_change = (
    image_ratio.lt(otsu_neg).multiply(-1)
    .add(image_ratio.gte(otsu_pos)).multiply(1)
)

mask = image_change.neq(0)
image_change = image_change.updateMask(mask)

# visualisasi citra hasil thresholding image rationing
ch_vis = {'min': -1, 'max': 1, 'palette': ['red', 'green']}
Map = geemap.Map()
Map.centerobject(region, zoom=12)
Map.addLayer(image_change, ch_vis, 'Image Change')
Map
```

Berikut adalah output visualisasi citra hasil deteksi perubahan menggunakan image rationing:



## Image Regression

Pada dasarnya, image regression merupakan bentuk lain dari image differencing. Formulanya sama persis dengan image differencing, yaitu citra  $t_2$  dikurangkan dengan citra  $t_1$ . Perbedaannya, jika pada image differencing normalisasi citra  $t_1$  menggunakan parameter-parameter statistik seperti nilai rerata dan standar deviasi, maka pada image regression, normalisasi citra  $t_1$  dilakukan menggunakan regresi linier. Hal ini didasarkan pada asumsi bahwa terdapat korelasi linier antara citra  $t_1$  dan  $t_2$ . Berikut adalah model image regression (Ilsever and Unsalan, 2012):

$$I_{reg}(x, y) = I_{t2}(x, y) - \hat{I}_{t1}(x, y)$$

Dimana:

$$\hat{I}_{t1}(x, y) = aI_{t1}(x, y) + b + \varepsilon$$

Dan:

- $I_{reg}(x, y)$  : Intensitas citra hasil image regression untuk setiap lokasi pixel  $(x, y)$   
 $\hat{I}_{t1}(x, y)$  : Intensitas suatu band dari citra  $t_1$  hasil regresi linier untuk setiap lokasi pixel  $(x, y)$   
 $a$  : Koefisien persamaan regresi linier untuk suatu band  
 $b$  : Intersep persamaan regresi linier untuk suatu band  
 $\varepsilon$  : Residu persamaan regresi linier untuk suatu band

Terkait implementasi regresi linier pada GEE image sudah pernah dibahas sebelumnya, silahkan lihat kembali halaman 250 sampai 252. Masih melanjutkan kode-kode image differencing dan image rationing sebelumnya, silahkan ketikkan dan jalankan kode-kode berikut:

```
# Persiapan parameter-parameter image regression
constant = ee.Image(1)
reg_params = ee.Image.cat(constant, ndvi1, ndvi2)
```

Perhatikan kode-kode di atas, `ee.Image.cat(constant, ndvi1, ndvi2)` berarti NDVI  $t_1$  dijadikan sebagai variabel  $x$ , sementara NDVI  $t_2$  akan dijadikan sebagai variabel  $y$ . Jika diterjemahkan, NDVI  $t_1$  akan dinormalkan menggunakan NDVI  $t_2$ .

```
# Kalkulasi image linear regression untuk NDVI
image_reg = reg_params.reduceRegion(
    reducer=ee.Reducer.linearRegression(numX=2, numY=1),
    scale=10,
    geometry=region
)
```

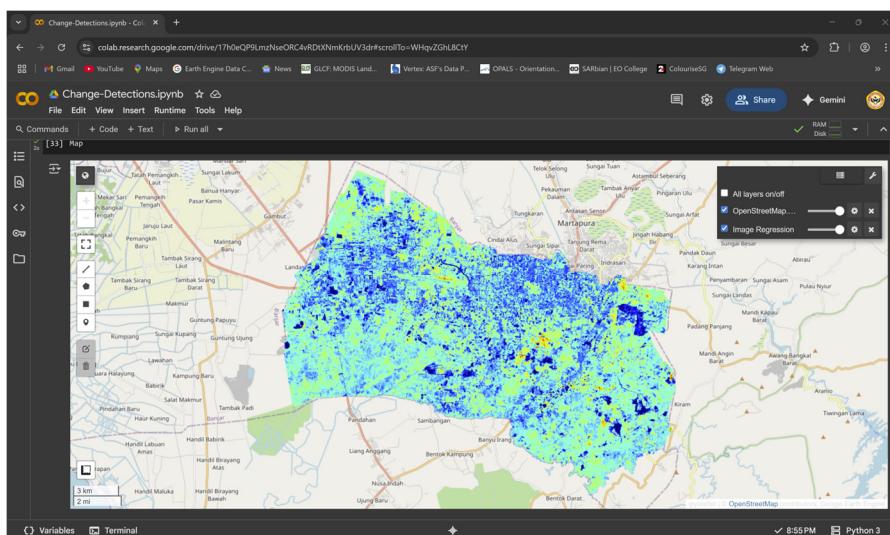
```
# Mendapatkan informasi intersep, koefisien, dan residu image regression
intercept = image_reg.get('coefficients'). getInfo()[0][0]
slope = image_reg.get('coefficients').getInfo()[1][0]
residual = image_reg.get('residuals').getInfo()[0]
```

```
# Membuat sebuah citra prediktif NDVI t1 menggunakan image regression
ndvi1_reg = ndvi1.multiply(slope).add(intercept).add(residual)
```

```
# Deteksi perubahan menggunakan image regression
image_regression = ndvi2.subtract(ndvi1_reg)
```

```
# Visualisasi citra hasil image regression
ireg_vis = {'min': -0.5, 'max': 0.5, 'palette': 'jet'}
Map = geemap.Map()
Map.centerObject(region, zoom=12)
Map.addLayer(image_regression, ireg_vis, 'Image Regression')
Map
```

Berikut adalah output citra hasil image regression:

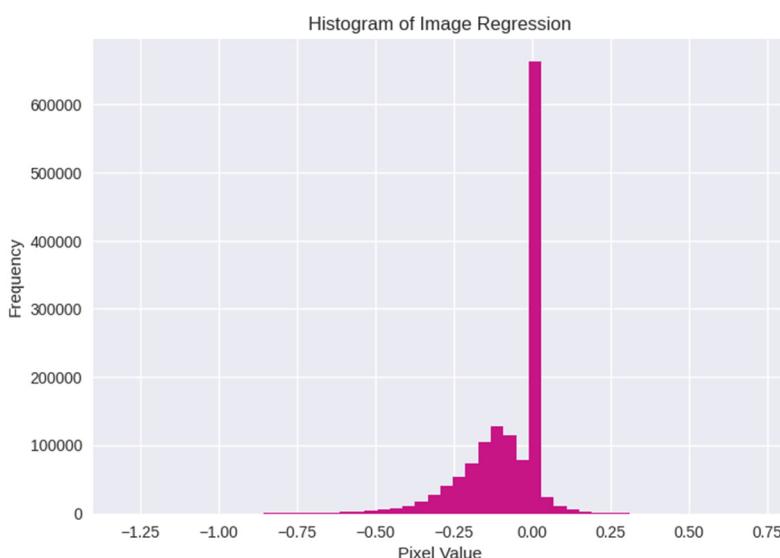


Mirip seperti image differencing dan image rationing sebelumnya, kita juga akan membuat histogram citra dan melakukan Otsu thresholding pada perubahan negatif dan perubahan positif.

```
# Pembuatan histogram citra hasil image regression
# Konversi GEE image ke NumPy array
image_reg_np = geemap.ee_to_numpy(image_regression, region=region, scale=20)

# Menghapus potensi nilai-nilai NaN
image_reg_np = image_reg_np[~np.isnan(image_reg_np)]

# Menampilkan histogram
plt.hist(image_reg_np, bins=50, color='mediumvioletred')
plt.title('Histogram of Image Regression')
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')
plt.show()
```



### Bab III Google Earth Engine

```
# Partisi data menjadi perubahan negatif dan perubahan positif
image_reg_np_lt0 = image_reg_np[image_reg_np < 0]
image_reg_np_gt0 = image_reg_np[image_reg_np > 0]

# Otsu thresholding untuk perubahan negatif
otsu_neg = threshold_otsu(image_reg_np_lt0)
print(f"Otsu threshold negatif: {otsu_neg}")

# Otsu thresholding untuk perubahan positif
otsu_pos = threshold_otsu(image_reg_np_gt0)
print(f"Otsu threshold positif: {otsu_pos}")

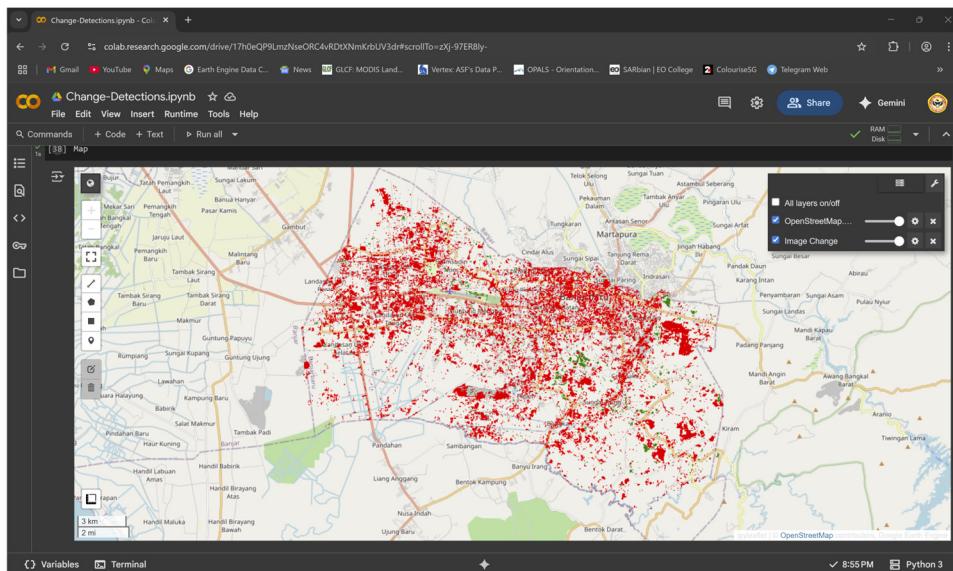
# Thresholding citra hasil image regression
threshold = 0.2
image_change = (
    image_regression.lte(otsu_neg).multiply(-1)
    .add(image_regression.gte(otsu_pos)).multiply(1)
)

mask = image_change.neq(0)
image_change = image_change.updateMask(mask)

# Visualisasi citra hasil thresholding image regression
ch_vis = {'min': -1, 'max': 1, 'palette': ['red','green']}

Map = geemap.Map()
Map.centerObject(region, zoom=12)
Map.addLayer(image_change, ch_vis, 'Image Change')
Map
```

Berikut adalah output visualisasi citra hasil deteksi perubahan menggunakan image regression:



Jika gambar di atas kita cermati, terdapat perbedaan yang kontras antara hasil deteksi perubahan menggunakan image regression, dengan image differencing dan image rationing sebelumnya. Hal yang sama juga dapat dilihat dari perbedaan kenampakan histogram citranya. Hal ini sebenarnya cukup menarik, sebab dapat menjadi peluang riset. Misalnya komparasi image differencing, image rationing, dan image regression dalam mengekstrak informasi geospasial deforestasi.

## Change Vector Analysis

Change Vector Analysis (CVA) (Malila, 1980) memiliki konsep yang sedikit berbeda dengan metode-metode sebelumnya. Kelebihan utama CVA adalah mampu mengakomodir multi saluran. Tidak seperti image differencing, image rationing, dan image regression, yang hanya mengakomodir deteksi perubahan untuk citra saluran tunggal. Kelebihan ini memungkinkan CVA untuk mendeteksi perubahan multi fitur sekaligus. Seperti perubahan antara vegetasi dan urban dalam analisis deforestasi, atau perubahan antara air dan urban dalam analisis konversi lahan rawa menjadi permukiman. CVA memiliki dua metode perhitungan, yaitu *CVA magnitude* yang digunakan untuk mengukur kuantitas atau besarnya perubahan, sebagaimana image differencing dan sejenisnya. Dan *CVA direction*, yang digunakan untuk mengukur arah atau azimut perubahan.

Meskipun CVA magnitude memiliki kemiripan dengan image differencing, image rationing, dan image regression, CVA magnitude memiliki satu kekurangan utama, yaitu nilai-nilai pixel hasil deteksi perubahannya selalu positif. Sehingga secara langsung CVA magnitude tidak dapat membedakan antara perubahan positif (seperti reforestasi) dan perubahan negatif (seperti deforestasi). Akan tetapi, hal ini nantinya dapat diatasi dengan kalkulasi CVA direction, dan integrasi antara CVA magnitude dan CVA direction.

### CVA Magnitude

CVA magnitude diformulasikan sebagai berikut (Ilsever and Unsalan, 2012):

$$|I(x,y)| = \sqrt{\sum_{i=1}^n (I_{i(x,y)t2} - I_{i(x,y)t1})^2}$$

Dimana:

- $|I(x,y)|$  : Intensitas citra CVA magnitude untuk setiap lokasi pixel  $(x,y)$
- $I_{i(x,y)t1}$  : Intensitas band ke- $i$  citra  $t_1$  untuk setiap lokasi pixel  $(x,y)$
- $I_{i(x,y)t2}$  : Intensitas band ke- $i$  citra  $t_2$  untuk setiap lokasi pixel  $(x,y)$

Sebagaimana sudah dijelaskan, bahwa secara teoritis CVA magnitude dapat mengkalkulasi perubahan dengan jumlah band yang banyak sekaligus. Misalnya seluruh band Citra Sentinel-2. Akan tetapi, pada contoh kasus CVA magnitude ini, kita hanya akan menggunakan dua citra hasil transformasi indeks nilai spektral, yaitu NDVI dan *Urban Index* (UI). Sehingga kita dapat melihat hasil deteksi perubahan fitur vegetasi dan perubahan fitur urban (lahan terbuka dan permukiman) sekaligus. UI diformulasikan sebagai berikut (Kawamura et al., 1996):

$$UI = \frac{SWIR2 - NIR}{SWIR2 + NIR}$$

Masih melanjutkan kode-kode image differencing, image rationing, dan image regression sebelumnya, silahkan jalankan kode-kode implementasi dan visualisasi CVA magnitude berikut:

```
# Transformasi UI untuk t1 dan t2
ui1 = swir21.subtract(nir1).divide(swir21.add(nir1)).rename('UI')
ui2 = swir22.subtract(nir2).divide(swir22.add(nir2)).rename('UI')
```

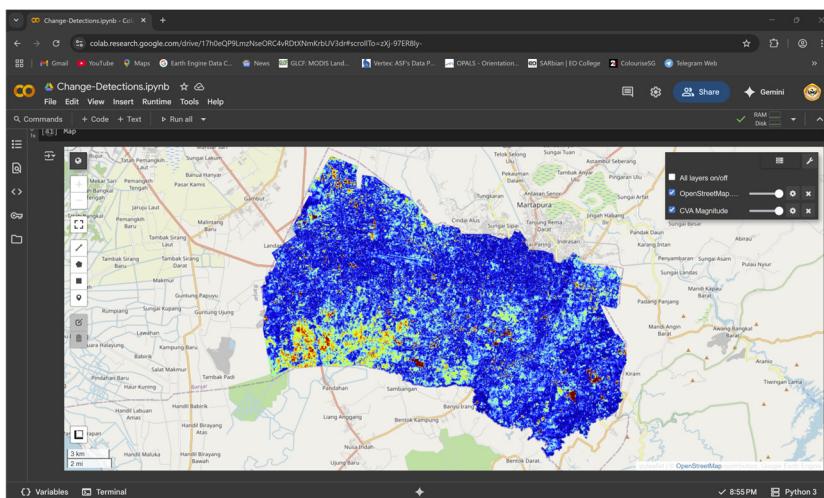
### Bab III Google Earth Engine

```
# Kalkulasi CVA magnitude untuk NDVI dan UI
cva_mag = (
    ((ndvi2.subtract(ndvi1)).pow(2))
    .add(ui2.subtract(ui1)).pow(2))
    .sqrt().rename('CVA Magnitude')
)

# Visualisasi citra CVA magnitude
cva_vis = {'min': 0, 'max': 0.5, 'palette': 'jet'}

Map = geemap.Map()
Map.centerObject(region, zoom=12)
Map.addLayer(cva_mag, cva_vis, 'CVA Magnitude')
Map
```

Berikut adalah citra hasil CVA magnitude:



Karena CVA magnitude hanya akan memberikan nilai-nilai positif, maka hanya ada satu nilai Otsu threshold. Dimana fungsi kondisional thresholding-nya adalah sebagai berikut:

$$I_c(x, y) = \begin{cases} 1, & \text{if } I_d(x, y) \geq \tau \\ 0, & \text{Otherwise} \end{cases}$$

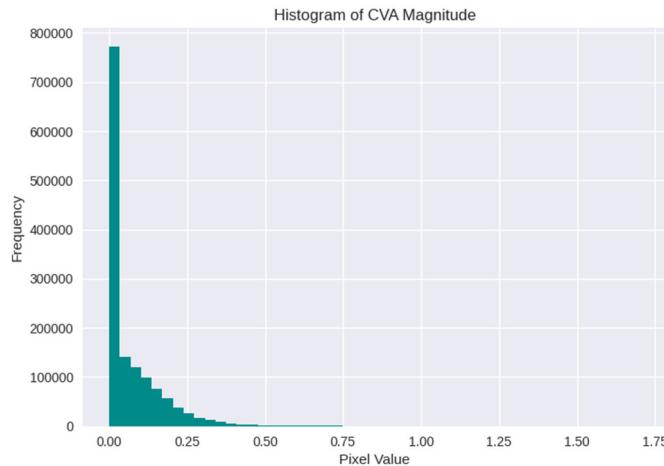
Dimana:

$\tau$  : Nilai threshold

Sebelumnya, kita juga akan membuat dan menampilkan histogram citra hasil CVA magnitude.

```
# Pembuatan histogram citra CVA magnitude
# Konversi GEE image ke NumPy array
cva_mag_np = geemap.ee_to_numpy(cva_mag, region=region, scale=20)
# Menghapus potensi nilai-nilai NaN
cva_mag_np = cva_mag_np[~np.isnan(cva_mag_np)]
```

```
# Menampilkan histogram
plt.hist(cva_mag_np, bins=50, color='darkcyan')
plt.title('Histogram of CVA Magnitude')
plt.xlabel('Pixel value')
plt.ylabel('Frequency')
plt.show()
```

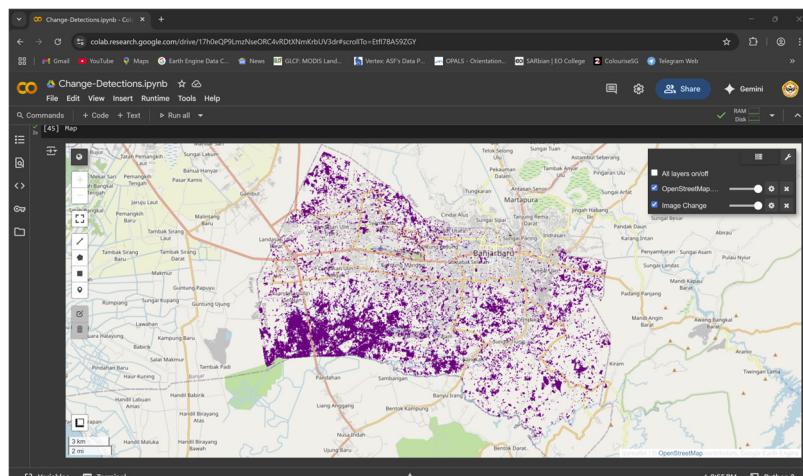


```
# Otsu thresholding CVA magnitude
otsu_thr = threshold_otsu(cva_mag_np[cva_mag_np > 0])
print(f'Otsu threshold: {otsu_thr}')
# Thresholding citra hasil CVA magnitude

image_change = cva_mag.gte(otsu_thr)
mask = image_change.neq(0)
image_change = image_change.updateMask(mask)
```

```
# Visualisasi citra hasil thresholding CVA magnitude
ch_vis = {'max': 1, 'palette': ['purple']}
Map = geemap.Map()
Map.centerobject(region, zoom=12)
Map.addLayer(image_change, ch_vis, 'Image Change')
Map
```

Berikut adalah output visualisasi citra hasil deteksi perubahan menggunakan CVA magnitude:



### Bab III Google Earth Engine

Output hasil deteksi perubahan menggunakan CVA magnitude adalah *binary image*. Yaitu nilai pixel 1 (terjadi perubahan), dan nilai pixel 0 (tidak terjadi perubahan). Normalnya, CVA magnitude menggunakan Euclidean Distance untuk mengukur besarnya perubahan yang terjadi. Akan tetapi, formula-formula pengukur jarak lainnya tentu saja dapat diterapkan juga. Seperti Manhattan Distance, Chebyshev Distance, atau yang lainnya. Perhatikan juga gambar histogram citra hasil CVA magnitude. Dimana puncak histogram berada di nilai 0 dan sekitarnya. Sebab nilai 0 dan sekitarnya merupakan fitur-fitur yang tidak mengalami perubahan dari  $t_1$  ke  $t_2$ .

#### CVA Direction

CVA direction memungkinkan kita untuk mengukur arah perubahan. CVA direction diformulasikan sebagai berikut (Ilsever and Unsalan, 2012):

$$\overrightarrow{I(x,y)} = \arctan \left( \frac{I_{1(x,y)t2} - I_{1(x,y)t1}}{I_{2(x,y)t2} - I_{2(x,y)t1}} \right)$$

Dimana:

- $\overrightarrow{I(x,y)}$  : CVA direction dalam satuan radian untuk setiap lokasi pixel  $(x, y)$   
 $I_{1(x,y)t1}$  : Intensitas band ke-1 citra  $t_1$  untuk setiap lokasi pixel  $(x, y)$   
 $I_{1(x,y)t2}$  : Intensitas band ke-1 citra  $t_2$  untuk setiap lokasi pixel  $(x, y)$   
 $I_{2(x,y)t1}$  : Intensitas band ke-2 citra  $t_1$  untuk setiap lokasi pixel  $(x, y)$   
 $I_{2(x,y)t2}$  : Intensitas band ke-2 citra  $t_2$  untuk setiap lokasi pixel  $(x, y)$

Formula di atas akan menghasilkan nilai arah dalam satuan radian, dengan rentang nilai  $-\frac{\pi}{2}$  hingga  $\frac{\pi}{2}$ , atau jika dikonversi ke dalam satuan derajat, dari  $-90^\circ$  sampai  $90^\circ$ . Tidak seperti CVA magnitude yang dapat mengukur besarnya perubahan secara langsung dengan banyak band atau citra, CVA direction hanya dapat mengukur arah dengan dua band atau citra. Hal ini terkait dengan formula trigonometri yang hanya dapat mengukur sudut secara 2-Dimensi. Pada contoh kasus ini, kita akan mengekstrak CVA direction dari tahun 2019 hingga 2024, dengan menggunakan NDVI dan UI. Masih melanjutkan kode-kode CVA magnitude sebelumnya, silahkan ketikkan dan jalankan kode-kode berikut:

```
# Kalkulasi CVA direction untuk NDVI (x-axis) dan UI (y-axis)
ui_ch = ui2.subtract(ui1)
ndvi_ch = ndvi2.subtract(ndvi1)

cva_dir = ((ui_ch).divide(ndvi_ch)).atan().rename('CVA Direction')
```

Pada sistem koordinat kartesius, jika  $\Delta NDVI$  adalah sumbu  $x$  dan  $\Delta UI$  adalah sumbu  $y$ , maka formula CVA direction pada kode-kode di atas adalah:

$$\overrightarrow{I(x,y)} = \arctan \left( \frac{\Delta UI}{\Delta NDVI} \right) = \arctan \left( \frac{UI_{t2} - UI_{t1}}{NDVI_{t2} - NDVI_{t1}} \right)$$

Permasalahannya adalah, formula CVA direction di atas hanya akan menghasilkan sudut dari  $-\frac{\pi}{2}$  hingga  $\frac{\pi}{2}$ . Padahal sistem koordinat kartesius 4 kuadran memiliki rentang sudut dari 0 hingga  $2\pi$ , atau  $0^\circ$  hingga  $360^\circ$ . Untuk mengkonversi nilai CVA direction menjadi 0 hingga  $2\pi$ , digunakan fungsi kondisional sebagai berikut:

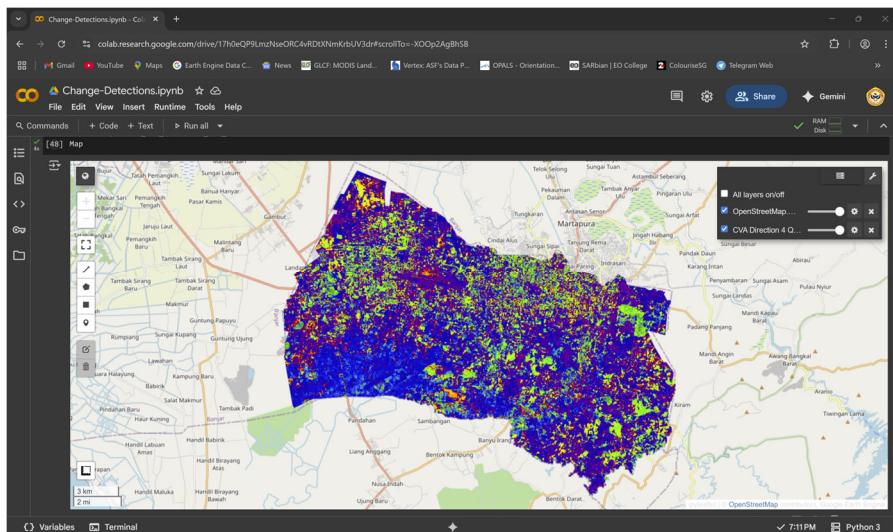
$$\overrightarrow{I(x,y)}_{\oplus} = \begin{cases} \overrightarrow{|I(x,y)|}, & \text{if } (I_{1(x,y)t_2} - I_{1(x,y)t_1}) \geq 0 \cap (I_{2(x,y)t_2} - I_{2(x,y)t_1}) \geq 0 \\ \pi - \overrightarrow{|I(x,y)|}, & \text{if } (I_{1(x,y)t_2} - I_{1(x,y)t_1}) \geq 0 \cap (I_{2(x,y)t_2} - I_{2(x,y)t_1}) < 0 \\ \pi + \overrightarrow{|I(x,y)|}, & \text{if } (I_{1(x,y)t_2} - I_{1(x,y)t_1}) < 0 \cap (I_{2(x,y)t_2} - I_{2(x,y)t_1}) < 0 \\ 2\pi - \overrightarrow{|I(x,y)|}, & \text{if } (I_{1(x,y)t_2} - I_{1(x,y)t_1}) < 0 \cap (I_{2(x,y)t_2} - I_{2(x,y)t_1}) \geq 0 \end{cases}$$

Dimana  $\overrightarrow{I(x,y)}_{\oplus}$  adalah CVA direction 4 kuadran. Perhatikan  $|\overrightarrow{I(x,y)}|$  yang mengindikasikan nilai absolut (positif) dari CVA direction. Berikut adalah implementasi CVA direction 4 kuadran:

```
# Kalkulasi CVA direction untuk 4 kuadran
cva_dir_quad = (
    (ui_ch.gte(0).And(ndvi_ch.gte(0))).multiply(cva_dir.abs())
    .add((ui_ch.gte(0).And(ndvi_ch.lt(0)))
    .multiply(ee.Image(math.pi).subtract(cva_dir.abs())))
    .add((ui_ch.lt(0).And(ndvi_ch.lt(0)))
    .multiply(ee.Image(math.pi).add(cva_dir.abs())))
    .add((ui_ch.lt(0).And(ndvi_ch.gte(0)))
    .multiply(ee.Image(2*math.pi).subtract(cva_dir.abs())))
).rename('CVA Direction')
```

Dan berikut adalah visualisasi citra CVA direction 4 kuadran:

```
# Visualisasi citra hasil CVA direction untuk 4 kuadran
cva_dir_vis = {'min': 0, 'max': 2*math.pi, 'palette': 'nipy_spectral_r'}
Map = geemap.Map()
Map.centerObject(region, zoom=12)
Map.addLayer(cva_dir_quad, cva_dir_vis, 'CVA Direction 4 quadrants')
Map
```



Untuk lebih memahami posisi 4 kuadran sekaligus mengamati sebaran arah-arah perubahannya menurut NDVI dan UI, kita dapat membuat scatter plot CVA direction 4 kuadran dengan kode-kode berikut. Dimana perubahan NDVI ( $\Delta NDVI$ ) ditempatkan sebagai sumbu x dan perubahan UI  $\Delta UI$  ditempatkan sebagai sumbu y.

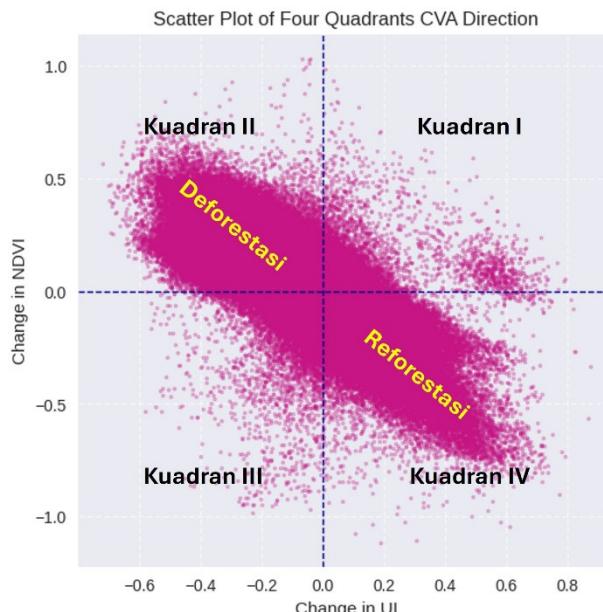
### Bab III Google Earth Engine

```
# Pembuatan scatter plot untuk CVA direction 4 kuadran
# Konversi GEE images of ui_ch and NDVI_ch to NumPy arrays
ui_ch_np = geemap.ee_to_numpy(ui_ch, region=region, scale=20)
ndvi_ch_np = geemap.ee_to_numpy(ndvi_ch, region=region, scale=20)

# Menghapus potensi nilai-nilai NaN dari kedua array secara bersamaan
valid_indices = ~np.isnan(ui_ch_np) & ~np.isnan(ndvi_ch_np)
ui_ch_np_valid = ui_ch_np[valid_indices]
ndvi_ch_np_valid = ndvi_ch_np[valid_indices]

# Menampilkan scatter plot
plt.figure(figsize=(6, 6))
plt.scatter(ui_ch_np_valid, ndvi_ch_np_valid, s=5, c='mediumvioletred',
alpha=0.3)
plt.title('Scatter Plot of Four Quadrants CVA Direction')
plt.xlabel('Change in NDVI')
plt.ylabel('Change in UI')
plt.axhline(0, color='darkblue', linestyle='--', linewidth=1)
plt.axvline(0, color='darkblue', linestyle='--', linewidth=1)
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```

Berikut adalah output scatter plot CVA direction 4 kuadran:

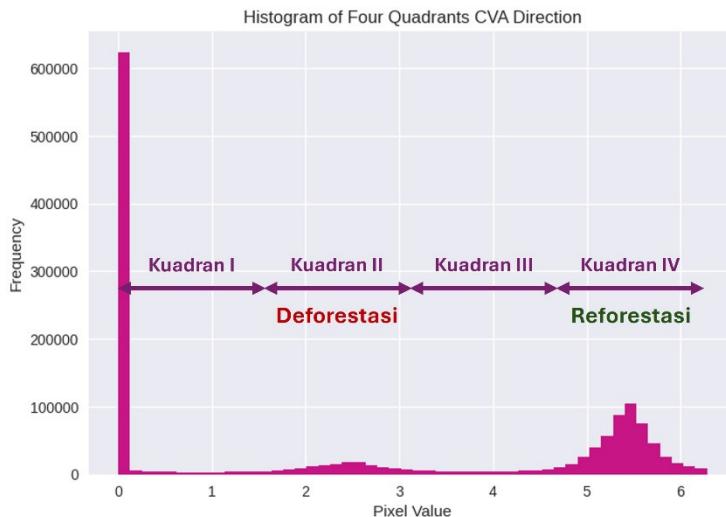


Sebagaimana CVA magnitude, kita juga akan membuat histogram CVA direction 4 kuadran:

```
# Pembuatan histogram citra CVA direction 4 kuadran
# Konversi GEE image ke NumPy array
cva_dir_quad_np = geemap.ee_to_numpy(cva_dir_quad, region=region, scale=20)

# Menghapus potensi nilai-nilai NaN
cva_dir_quad_np = cva_dir_quad_np[~np.isnan(cva_dir_quad_np)]

# Menampilkan histogram
plt.hist(cva_dir_quad_np, bins=50, color='mediumvioletred')
plt.title('Histogram of Four Quadrants CVA Direction')
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')
plt.show()
```



Histogram di atas memperlihatkan sudut perubahan dari 0 hingga  $2\pi$ . Terlihat bahwa terdapat 3 puncak histogram dan 3 konsentrasi arah. Arah atau sudut di sekitar 0 merupakan fitur-fitur yang tidak berubah. Perhatikan gambar scatter plot sebelumnya, area dari  $\frac{\pi}{2}$  hingga  $\pi$ , atau kuadran II, merupakan area deforestasi. Hal ini karena pada kuadran II, perubahan NDVI bernilai negatif, dan perubahan UI bernilai positif. Sementara area dari  $\frac{3\pi}{2}$  hingga  $2\pi$  atau kuadran IV merupakan area reforestasi. Sebab pada kuadran IV, perubahan NDVI bernilai positif dan perubahan UI bernilai negatif. **PERHATIKAN**, bahwa interpretasi seperti ini hanya berlaku jika indeks-indeks nilai spektral yang digunakan adalah NDVI dan UI. Jika indeks-indeks nilai spektral yang digunakan berbeda, maka penafsirannya juga akan berbeda dan harus disesuaikan.

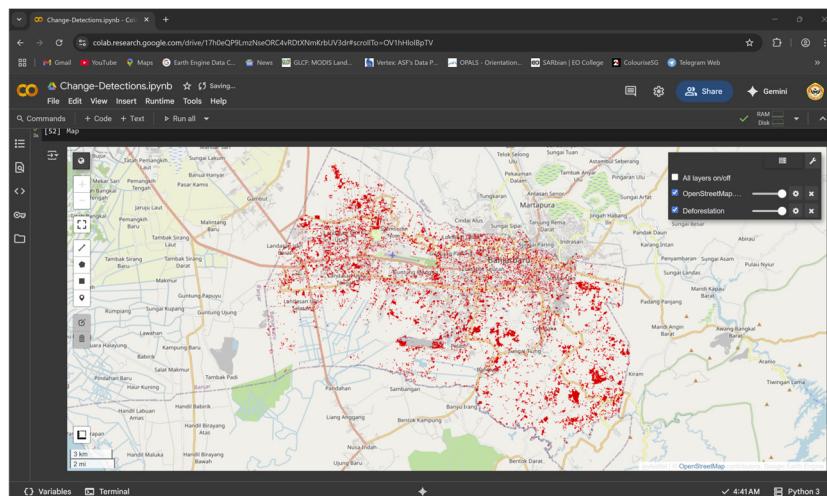
Jika  $\pi$  adalah 3,14, pada gambar histogram CVA direction di atas,  $\frac{\pi}{2}$  sampai  $\pi$  ekivalen dengan 1,57 sampai 3,14 (rentang nilai deforestasi). Dan  $\frac{3\pi}{2}$  sampai  $2\pi$  ekivalen dengan 4,71 sampai 6,28 (rentang nilai reforestasi). Agar lebih efektif (perhatikan konsentrasi sebaran data pada scatter plot dan puncak histogram), kita dapat mempersempit rentang nilai threshold deforestasi dan reforestasi. Misalnya  $25\% \frac{\pi}{4}$  ke kiri dan ke kanan, dari sudut tengah kuadran II ( $\frac{3\pi}{4}$ ) dan dari sudut tengah kuadran IV ( $\frac{7\pi}{4}$ ). Sebagai opsi, jika Anda lebih familiar dengan sudut dalam satuan derajat dibanding radian, Anda dapat mengkonversi CVA direction 4 kuadran ke dalam satuan derajat dengan formula  $\overrightarrow{I(x,y)}_0 = \overrightarrow{I(x,y)}_{\oplus} \frac{180^0}{\pi}$ . Dimana  $\overrightarrow{I(x,y)}_{\oplus}$  adalah CVA direction 4 kuadran dalam satuan derajat, dengan rentang nilai  $0^0$  sampai  $360^0$ . Berikut adalah kode-kode untuk identifikasi deforestasi dan reforestasi menggunakan CVA direction 4 kuadran (dalam satuan radian):

```
# Identifikasi deforestasi (kuadran II)
def_min_th, def_max_th = 3*math.pi/4 - math.pi/8, 3*math.pi/4 + math.pi/8
mask = cva_dir_quad.gte(def_min_th).And(cva_dir_quad.lte(def_max_th))
deforest = mask.updateMask(mask)
```

```
# Visualisasi citra deforestasi
def_vis = {'max': 1, 'palette': ['red']}
Map = geemap.Map()
Map.centerObject(region, zoom=12)
Map.addLayer(deforest, def_vis, 'Deforestation')
```

### Bab III Google Earth Engine

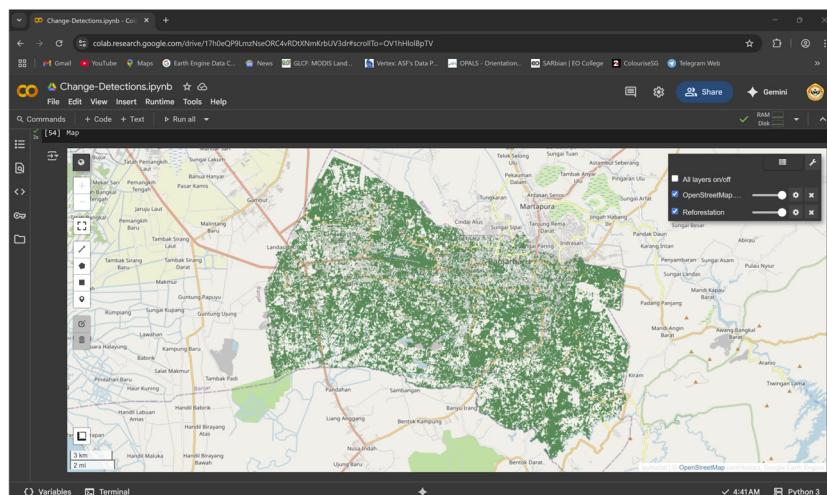
Berikut adalah output visualisasi potensi area-area deforestasi:



```
# Identifikasi reforestasi (kuadran IV)
ref_min_th, ref_max_th = 7*math.pi/4 - math.pi/8, 7*math.pi/4 + math.pi/8
mask = cva_dir_quad.gte(ref_min_th).And(cva_dir_quad.lte(ref_max_th))
reforest = mask.updateMask(mask)
```

```
# Visualisasi citra reforestasi
ref_vis = {'max': 1, 'palette': ['seagreen']}
Map = geemap.Map()
Map.centerObject(region, zoom=12)
Map.addLayer(reforest, ref_vis, 'Reforestation')
Map
```

Berikut adalah output visualisasi potensi area-area reforestasi:



**SEBAGAI LATIHAN**, untuk meningkatkan akurasi hasil ekstraksi informasi deforestasi dan reforestasi, coba Anda integrasikan antara hasil threshold perubahan dari CVA magnitude sebelumnya, dengan CVA direction 4 kuadran.

## H. Analisis dan Prediksi Deret Waktu

Salah satu keuntungan menggunakan GEE adalah kita dapat mengakses data dalam kuantitas yang sangat besar (*big data*). Baik besar secara spasial, dalam arti luasan wilayah yang besar, maupun besar secara temporal, dalam arti rentang waktu yang sangat panjang. Dimana hal seperti ini hampir mustahil jika harus dilakukan dengan komputasi konvensional. Dengan komputasi awan GEE, kita dapat memantau perubahan objek-objek di atas permukaan bumi selama rentang waktu 30 tahun misalnya. Observasi jangka panjang terhadap dinamika objek-objek di permukaan bumi, akan memungkinkan kita untuk memahami lebih komprehensif fenomena-fenomena yang telah, sedang, dan akan terjadi. Termasuk fenomena-fenomena lain yang terkait dengan fenomena yang sedang kita observasi, misalnya korelasi antara penambahan luas area yang terdeforestasi di suatu DAS, dengan peningkatan muatan suspensi air sungai.

Di dalam buku ini, yang dimaksud dengan analisis dan prediksi deret waktu (*time series analysis and prediction*) adalah, metode dan teknik untuk menganalisis kecenderungan sekaligus memprediksi ke masa yang akan datang terhadap kondisi suatu fenomena di permukaan bumi. Sebagai contoh, kecenderungan dan prediksi perkembangan permukiman di suatu wilayah, kecenderungan dan prediksi deforestasi, kecenderungan dan prediksi emisi atau sekuesterasi karbon, kecenderungan dan prediksi kualitas atau pencemaran air, dan sebagainya.

Terinspirasi dari riset Kwong et al. (2022), yang memantau kualitas air laut di perairan pesisir Hong Kong dengan menggunakan Sentinel-2 *time series*, pada latihan ini kita akan mengamati kecenderungan sekaligus memprediksi kekeruhan air muara Sungai Barito, Kalimantan Selatan. Sungai Barito merupakan salah satu sungai terpanjang di Indonesia, dengan kondisi alur sungai yang cukup memprihatinkan. Dimana muatan suspensi dan sedimentasi yang tinggi menyebabkan alur sungai ini harus dikeruk terus-menerus sejak 2003. Silahkan buat sebuah notebook Google Colab baru, beri nama misalnya **Longterm-Monthly-Turbidity-Monitoring.ipynb**. Kemudian ketikkan dan jalankan kode-kode berikut:

```
import ee, geemap

ee.Authenticate()

ee.Initialize(project='ee-geospatialulm')

# Create a straight line from the two point coordinates
line = ee.Geometry.LineString([[114.511, -3.450347],[114.4901, -3.537548]])
```

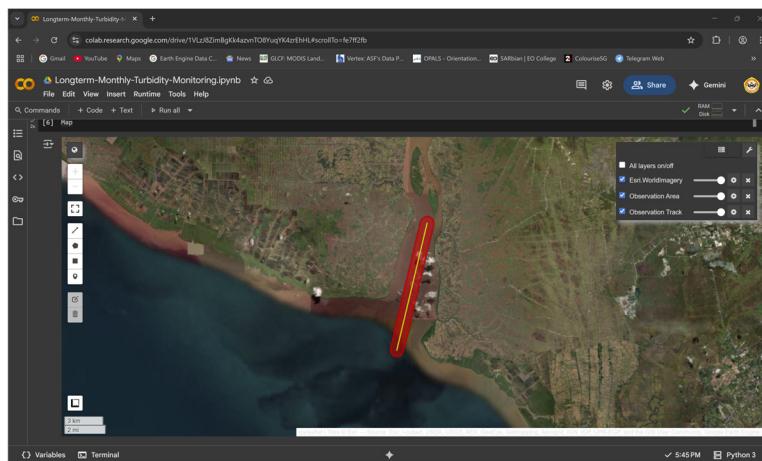
Kode-kode di atas digunakan untuk membuat geometri garis sepanjang ± 10 km di tengah-tengah muara barito. Garis ini akan menjadi transek atau jalur pengamatan kekeruhan air. Selanjutnya, sekeliling transek ini akan dibuat wilayah observasi dengan teknik *buffer* selebar 500 meter.

```
# Create area of interest using buffer of line coordinates for 500 meters
area = line.buffer(500)

# Visualize area of interest
Map = geemap.Map(basemap='SATELLITE')
Map.centerobject(area, 12)
Map.addLayer(area, {'color': 'red'}, 'Observation Area')
Map.addLayer(line, {'color': 'yellow'}, 'Observation Track')
```

### Bab III Google Earth Engine

Berikut adalah visualisasi transek dan wilayah observasi:



Silahkan lanjutkan dengan mengetikkan dan mengeksekusi kode-kode berikut:

```
# Applying scaling factors.  
def apply_scale_factors(image):
```

```
    optical_bands = image.select('SR_B.*').multiply(0.0000275).add(-0.2)  
    thermal_bands = image.select('ST_B.*').multiply(0.00341802).add(149.0)  
  
    image = (  
        image.addBands(optical_bands, None, True)  
        .addBands(thermal_bands, None, True)  
    )  
  
    return image
```

```
# Masking clouds and shadows  
def mask_clouds(image):
```

```
    qa = image.select('QA_PIXEL')  
  
    dilated_bit_mask = 1 << 1  
    cirrus_bit_mask = 1 << 2  
    cloud_bit_mask = 1 << 3  
    shadow_bit_mask = 1 << 4  
  
    mask = (  
        qa.bitwiseAnd(dilated_bit_mask).eq(0)  
        .And(qa.bitwiseAnd(cirrus_bit_mask).eq(0))  
        .And(qa.bitwiseAnd(cloud_bit_mask).eq(0))  
        .And(qa.bitwiseAnd(shadow_bit_mask).eq(0))  
    )  
  
    return image.updateMask(mask)
```

Observasi jangka panjang dilakukan selama 35 tahun, yaitu dari tahun 1990 hingga 2024. Sehingga kita akan menggunakan Seri Landsat lintas generasi, dari Landsat 4 hingga Landsat 9.

```
# Load Landsat 4 Collection  
l4_col = (  
    ee.ImageCollection('LANDSAT/LT04/C02/T1_L2')  
    .filterDate('1990-01-01', '1992-12-31')  
    .map(apply_scale_factors)  
    .map(mask_clouds)  
)
```

```
# Load Landsat 5 collection
15_col = (
    ee.ImageCollection('LANDSAT/LT05/C02/T1_L2')
    .filterDate('1990-01-01', '2011-12-31')
    .map(apply_scale_factors)
    .map(mask_clouds)
)
```

```
# Load Landsat 7 collection
17_col = (
    ee.ImageCollection('LANDSAT/LE07/C02/T1_L2')
    .filterDate('2000-01-01', '2023-12-31')
    .map(apply_scale_factors)
    .map(mask_clouds)
)
```

```
# Load Landsat 8 collection
18_col = (
    ee.ImageCollection('LANDSAT/LC08/C02/T1_L2')
    .filterDate('2014-01-01', '2024-12-31')
    .map(apply_scale_factors)
    .map(mask_clouds)
)
```

```
# Load Landsat 9 collection
19_col = (
    ee.ImageCollection('LANDSAT/LC09/C02/T1_L2')
    .filterDate('2022-01-01', '2024-12-31')
    .map(apply_scale_factors)
    .map(mask_clouds)
)
```

Informasi kekeruhan air yang digunakan adalah Normalized Difference Turbidity Index (NDTI). Silahkan lihat kembali formula NDTI di bagian sebelumnya pada halaman 203.

```
# Normalized Difference Turbidity Index (NDTI)
# from Landsat 4, 5, and 7
def ndti_1457(image):
    green = image.select('SR_B2')
    red = image.select('SR_B3')
    swirl1 = image.select('SR_B5')

    mndwi = green.subtract(swirl1).divide(green.add(swirl1))
    ndti = red.subtract(green).divide(red.add(green))

    water_mask = mndwi.gt(0.1)
    ndti_masked = ndti.updateMask(water_mask).rename('NDTI')

    return image.addBands(ndti_masked)
```

```
# Normalized Difference Turbidity Index (NDTI)
# from Landsat 8 and 9
def ndti_189(image):
    green = image.select('SR_B3')
    red = image.select('SR_B4')
    swirl1 = image.select('SR_B6')

    mndwi = green.subtract(swirl1).divide(green.add(swirl1))
    ndti = red.subtract(green).divide(red.add(green))

    water_mask = mndwi.gt(0.1)
    ndti_masked = ndti.updateMask(water_mask).rename('NDTI')

    return image.addBands(ndti_masked)
```

### Bab III Google Earth Engine

Posisi band-band Landsat 4, 5, dan 7 berbeda dengan Landsat 8 dan 9. Sehingga diperlukan dua fungsi yang berbeda untuk kalkulasi NDTI, sebagaimana kode-kode di atas. Langkah berikutnya adalah kalkulasi NDTI pada semua image collection, dari Landsat 5 hingga Landsat 9:

```
# Calculate NDTI for each Landsat collection
ndti4_col = 14_col.map(ndti_1457)
ndti5_col = 15_col.map(ndti_1457)
ndti7_col = 17_col.map(ndti_1457)
ndti8_col = 18_col.map(ndti_189)
ndti9_col = 19_col.map(ndti_189)
```

Selanjutnya, semua NDTI dari seluruh Seri Landsat digabungkan menjadi satu image collection.

```
# Merge all NDTI collections
ndti_col = (
    ndti4_col.merge(ndti5_col)
    .merge(ndti7_col).merge(ndti8_col)
    .merge(ndti9_col)
)
```

Pada studi kasus ini, kita akan memantau fluktuasi atau kecenderungan NDTI rerata bulanan. Ketikkan dan jalankan kode-kode berikut untuk membuat sebuah image collection NDTI rerata bulanan dari tahun 1990 hingga 2024:

```
# Create Monthly Mean NDTI composites for each year
from tqdm import tqdm

monthly_ndti_images = []
years = ee.List.sequence(1990, 2024)
months = ee.List.sequence(1, 12)

years_list = years.getInfo()
months_list = months.getInfo()

for y in tqdm(years_list):
    for m in months_list:
        # Filter images for the specific year and month
        start_date = f"{y}-{m:02d}-01"
        if m == 12:
            end_date = f"{y+1}-01-01"
        else:
            end_date = f"{y}-{m+1:02d}-01"
        monthly_img = (
            ndti_col.select('NDTI')
            .filterDate(start_date, end_date)
            .mean()
            .set({'year': y, 'month': m})
            .rename('NDTI')
        )
        monthly_ndti_images.append(monthly_img)
monthly_ndti = ee.ImageCollection(monthly_ndti_images)
```

Instruksi `tqdm` pada kode-kode di atas digunakan untuk memvisualisasikan progres iterasi `for`. Iterasi `for` ini akan menghasilkan sebuah GEE list NDTI rerata bulanan, lihat instruksi `monthly_ndti_images.append(monthly_img)`. GEE list ini harus dikonversi menjadi image collection dengan instruksi `ee.ImageCollection(monthly_ndti_images)`. Data bulanan NDTI-nya sendiri difilter dengan `.filterDate(start_date, end_date)`. `start_date` pasti dimulai dari tanggal 1 setiap bulan (`f'{y}-{m:02d}-01'`), `end_date` ditentukan dari tanggal 1 bulan berikutnya (`m+1`), lihat kode `f'{y}-{m+1:02d}-01'`.

Kecuali untuk bulan Desember, lihat ekspresi `if m == 12, end_date harus diambil dari tanggal 1 bulan 1 tahun berikutnya (y+1), lihat kode f"\{y+1\}-01-01". Untuk memahami hal ini, ingat kembali konsep inklusif dan ekslusif pada list (lihat halaman 73). f"\{y\}-{m:02d}-01" merupakan f-string Python, dimana hal ini sudah dibahas dan banyak dipakai pada bagian terdahulu (lihat penjelasan halaman 59). y adalah tahun, 01 adalah tanggal 1 (untuk setiap bulan), dan m:02d adalah kode bulan sebagai bilangan bulat dua digit. m:02d maksudnya jika bulan 3 akan ditulis 03, tetapi jika bulan 11 tetap ditulis 11.`

Berikutnya, kita juga akan membuat rerata nilai NDTI untuk seluruh wilayah observasi (area buffer 500 meter sekeliling garis transek), untuk setiap bulan. Dengan demikian, nanti untuk setiap bulan hanya ada satu data NDTI, yang merupakan rerata NDTI bulanan untuk seluruh wilayah obervasi. Silahkan ketikkan dan jalankan kode-kode berikut:

```
# Calculate Monthly Mean NDTI values for each year from 1990 to 2024
monthly_mean_ndti = []
monthly_labels = []

for y in tqdm(years_list):
    for m in months_list:
        # Filter the image for the specific year and month
        img = (
            monthly_ndti.filter(ee.Filter.eq('year', y))
            .filter(ee.Filter.eq('month', m)).first()
        )
        if img is not None:
            stats = img.reduceRegion(
                reducer=ee.Reducer.mean(),
                geometry=area,
                scale=30,
                maxPixels=1e13
            ). getInfo()
            ndti_mean = stats.get('NDTI', None)
        else:
            ndti_mean = None
        monthly_mean_ndti.append({
            'year': y,
            'month': m,
            'ndti_mean': ndti_mean
        })
    monthly_labels.append(f"\{y\}-{m:02d}\")
```

Selain menghitung rerata NDTI bulanan untuk seluruh wilayah observasi, kode-kode di atas juga akan menghasilkan label bulan untuk list data NDTI. Selanjutnya, list NDTI rerata bulanan ini dikonversi menjadi NumPy array.

```
# Extract only the ndti_mean values from the list of dicts
import numpy as np

monthly_ndti_original = np.array([d['ndti_mean'] for d in monthly_mean_ndti],
                                 dtype=np.float32)
```

Untuk melihat pola dan kecenderungan data NDTI rerata bulanan, kita akan menjalankan regresi linier menggunakan Scikit-Learn.

```
# Perform linear regression on the original Monthly NDTI values
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

# Convert to numpy arrays for plotting
monthly_ndti_values = np.array(monthly_ndti_original, dtype=np.float32)
x = np.arange(len(monthly_ndti_values))
```

### Bab III Google Earth Engine

```
# Prepare data for regression (exclude NaNs)
mask = ~np.isnan(monthly_ndti_values)
x = x[mask].reshape(-1, 1)
y_vals = monthly_ndti_values[mask]

# Fit linear regression
reg = LinearRegression().fit(x, y_vals)
slope = reg.coef_[0]
intercept = reg.intercept_
r2 = r2_score(y_vals, reg.predict(x))
```

Data time series NDTI rerata bulanan, bersama hasil-hasil analisis regresi linier di atas kemudian diplotkan menggunakan Matplotlib Pyplot. Silahkan ketikkan atau *copy-paste* kemudian eksekusi kode-kode berikut:

```
# Plot time series with linear regression and confidence interval
import matplotlib.pyplot as plt
from scipy.stats import t

plt.style.use('seaborn-v0_8')
plt.figure(figsize=(16, 5))
plt.plot(monthly_labels, monthly_ndti_original, marker='o', label='Monthly Mean NDTI')
plt.plot(monthly_labels, reg.predict(x.reshape(-1, 1)), "r--", label='Trend Line')

# Calculate 95% confidence interval for regression line
y_pred = reg.predict(x.reshape(-1, 1))
n = len(y_vals)
se = np.sqrt(np.sum((y_vals - reg.predict(x)) ** 2) / (n - 2))
mean_x = np.mean(x)
t_val = t.ppf(0.975, df=n-2)
conf_int = t_val * se * np.sqrt(1/n + (x - mean_x)**2 / np.sum((x - mean_x)**2))
plt.fill_between(
    monthly_labels,
    y_pred - conf_int,
    y_pred + conf_int,
    color='red',
    alpha=0.2,
    label='95% Confidence Interval'
)

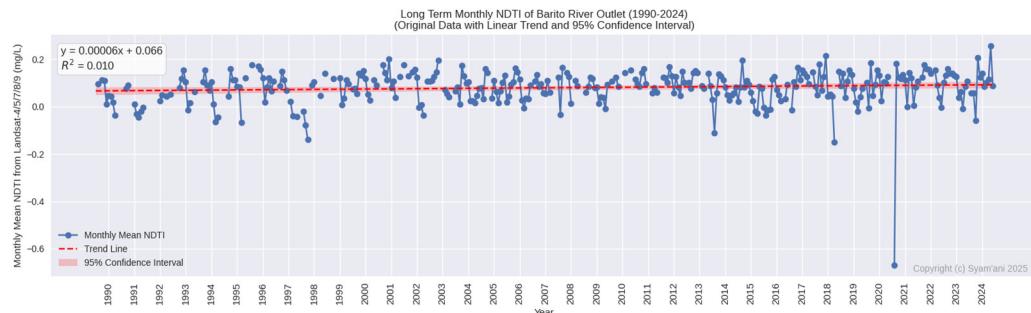
# Annotate equation and R^2
eq_text = f'y = {slope:.5f}x + {intercept:.3f}\nR^2 = {r2:.3f}'
plt.text(0.01, 0.95, eq_text, transform=plt.gca().transAxes,
         fontsize=12, verticalalignment='top',
         bbox=dict(boxstyle="round", fc="w", ec="0.5", alpha=0.8))

plt.text(0.99, 0.01, "copyright (c) Syam'ani 2025",
         transform=plt.gca().transAxes, fontsize=10,
         color='gray', ha='right', va='bottom', alpha=0.7)

# Set x-ticks to show only one label per year
year_ticks = list(range(0, len(monthly_labels), 12))
year_labels = [str(y) for y in years_list]
year_ticks = [i + 6 for i in year_ticks]
plt.xticks(year_ticks, year_labels, rotation=90, fontsize=11)
plt.xlabel('Year')
plt.ylabel('Monthly Mean NDTI from Landsat-4/5/7/8/9 (mg/L)')
plt.title('Long Term Monthly NDTI of Barito River outlet (1990-2024) \
\\n(Original Data with Linear Trend and 95% Confidence Interval)')
plt.legend()
plt.tight_layout()
plt.show()
```

Selain plot data NDTI dan hasil-hasil analisis regresi linier, kode-kode di atas juga menghitung selang kepercayaan (*confidence interval*) 95% untuk garis *trend* regresi, dengan menggunakan SciPy *t-Student's*. Meskipun data yang akan kita plotkan merupakan data NDTI bulanan, kode-kode di atas akan memberikan label tahunan pada sumbu *x*, agar plot lebih mudah diamati.

Berikut adalah outputnya:



Setidaknya ada dua masalah data yang terlihat pada plot di atas. Pertama, terdapat beberapa bulan yang tidak memiliki data atau kehilangan data. Kedua, terdapat data yang sangat ekstrem. Data yang hilang pada bulan-bulan tertentu, kemungkinan dikarenakan pada bulan-bulan tersebut dan di wilayah tersebut tidak ditemukan akuisisi citra yang bebas awan. Untuk kasus data yang hilang ini nantinya kita akan melakukan interpolasi data. Sementara untuk data yang sangat ekstrem, atau terlalu besar perbedaan dengan data lainnya merupakan pencilan (*outliers*). Dalam analisis statistik, pencilan dapat dieliminasi dengan metode tertentu untuk menjaga integritas data. Kita akan menggunakan metode *Interquartile Range* (IQR) untuk menghilangkan data pencilan. IQR diformulasikan sebagai berikut (Clark-Carter, 2005):

$$IQR = Q_3 - Q_1$$

$$Lower Bound = Q_1 - 1,5IQR$$

$$Upper Bound = Q_3 + 1,5IQR$$

$Q_1$  dan  $Q_3$  masing-masing adalah kuartil pertama dan kuartil ketiga dari data. Pencilan adalah data yang kurang dari *Lower Bound* atau data yang lebih dari *Upper Bound*. Jika kita ingin membuang data yang sangat ekstrem,  $1,5IQR$  dapat dirubah menjadi  $3IQR$ .

```
# Remove outliers using Interquartile Range (IQR)
Q1 = np.nanpercentile(monthly_ndti_original, 25)
Q3 = np.nanpercentile(monthly_ndti_original, 75)
IQR = Q3 - Q1

# Define lower and upper bounds
lower_bound = Q1 - 3 * IQR
upper_bound = Q3 + 3 * IQR

print(f"Lower Bound: {lower_bound}")
print(f"Upper Bound: {upper_bound}")
```

Output batas-batas pencilan:

```
Lower Bound: -0.2025848627090454
Upper Bound: 0.37084344029426575
```

Untuk selanjutnya, data NDTI yang dianggap valid dan akan diambil untuk analisis lebih lanjut adalah  $NDTI \geq -0.2025848627090454$  dan  $NDTI \leq 0.37084344029426575$ . Tentu saja, pencilan-pencilan yang dieliminasi akan menyebabkan data hilang. Untuk selanjutnya, data NDTI yang hilang, baik hilang karena eliminasi pencilan, atau data aktual memang tidak ada karena kekosongan akuisisi citra, seluruhnya akan diinterpolasi temporal dengan kode-kode berikut:

```
# Interpolate ndti missing data in Monthly NDTI values
from scipy.interpolate import interp1d

# Copy the original values to a new array for interpolation
monthly_ndti_values = monthly_ndti_original.copy()

# Build anomaly mask: True for valid, False for missing/anomaly
anomaly_mask = np.array([
    False if (v is None or np.isnan(v) or np.isinf(v))
    or v < lower_bound or v > upper_bound
    else True for v in monthly_ndti_values
])

# Find indices where anomaly_mask is False (i.e., anomalies)
anomaly_indices = np.where(~anomaly_mask)[0]

# Interpolate only over valid (non-anomaly) values
valid_idx = np.where(anomaly_mask)[0]
valid_x = valid_idx
valid_y = monthly_ndti_values[valid_idx]

# Create interpolation function
f_interp = interp1d(valid_x, valid_y, kind='linear',
                    fill_value='extrapolate')

# Fill anomaly positions with interpolated values
interp_values = f_interp(anomaly_indices)
for idx, val in zip(anomaly_indices, interp_values):
    monthly_ndti_values[idx] = float(val)
```

Proses interpolasi temporal dilakukan menggunakan SciPy. Untuk lebih jelas, silahkan baca di tautan <https://docs.scipy.org/doc/scipy/reference/generated/scipy.interpolate.interp1d.html>. Pada kode-kode di atas, data yang diinterpolasi adalah data yang tidak ada/hilang (`None`), data yang tidak terdefinisi/`NaN` (`np.isnan(v)`), data tak terhingga/`Infinity` (`np.isinf(v)`), dan pencikan (`v < lower_bound or v > upper_bound`). Metode interpolasi yang digunakan adalah interpolasi linier (`kind='linear'`, `fill_value='extrapolate'`) digunakan untuk mengekstrapolasi nilai-nilai yang berada di luar rentang data.

## Kecenderungan Linier

Kecenderungan linier berasumsi bahwa data normal mengikuti atau mendekati pola garis lurus. Untuk kasus data NDTI multitemporal, berarti kita berasumsi bahwa kenaikan atau penurunan NDTI menurut waktu polanya mendekati garis lurus. Kecenderungan linier dapat dianalisis menggunakan regresi linier. Dalam hal ini, waktu adalah variable  $x$  dan nilai-nilai NDTI adalah variable  $y$ . Untuk data NDTI rerata bulanan periode 1990-2024, maka bulan pertama, yaitu Januari 1990, akan dinotasikan sebagai  $x = 0$ , demikian seterusnya secara berurutan hingga bulan terakhir, yaitu Desember 2024 yang merupakan bulan ke-420 ( $x = 419$ ).

```
# Perform linear regression on the Monthly NDTI values

# Convert to numpy arrays for plotting
monthly_ndti_values = np.array(monthly_ndti_values, dtype=np.float32)
x = np.arange(len(monthly_ndti_values))

# Prepare data for regression (exclude NaNs)
mask = ~np.isnan(monthly_ndti_values)
x = x[mask].reshape(-1, 1)
y_vals = monthly_ndti_values[mask]

# Fit linear regression
reg = LinearRegression().fit(x, y_vals)
slope = reg.coef_[0]
intercept = reg.intercept_
r2 = r2_score(y_vals, reg.predict(x))
```

Selanjutnya, kita akan memplotkan hasil analisis regresi linier:

```
# Plot time series with linear regression and confidence interval
plt.figure(figsize=(16, 5))
plt.plot(monthly_labels, monthly_ndti_values, marker='o', label='Monthly Mean NDTI')
plt.plot(monthly_labels, reg.predict(x.reshape(-1, 1)), "r--", label='Trend Line')

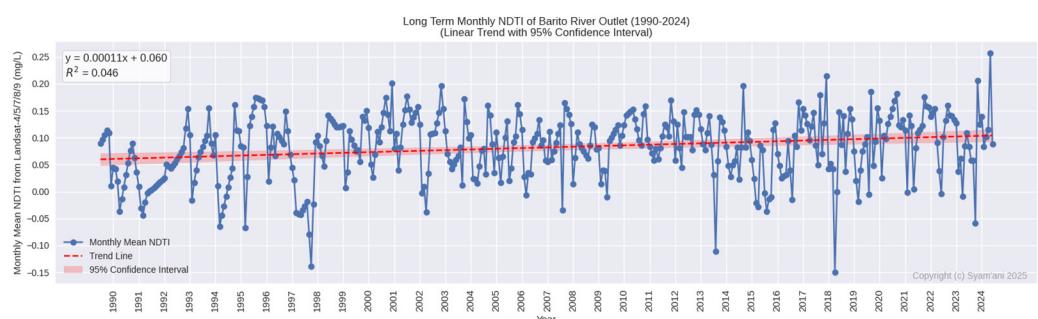
# Calculate 95% confidence interval for regression line
y_pred = reg.predict(x.reshape(-1, 1))
n = len(y_vals)
se = np.sqrt(np.sum((y_vals - reg.predict(x)) ** 2) / (n - 2))
mean_x = np.mean(x)
t_val = t.ppf(0.975, df=n-2)
conf_int = t_val * se * np.sqrt(1/n + (x - mean_x)**2 / np.sum((x - mean_x)**2))
plt.fill_between(
    monthly_labels,
    y_pred - conf_int,
    y_pred + conf_int,
    color='red',
    alpha=0.2,
    label='95% Confidence Interval'
)

# Annotate equation and R^2
eq_text = f"y = {slope:.5f}x + {intercept:.3f}\nR^2 = {r2:.3f}"
plt.text(0.01, 0.95, eq_text, transform=plt.gca().transAxes,
         fontsize=12, verticalalignment='top',
         bbox=dict(boxstyle="round", fc="w", ec="0.5", alpha=0.8))

plt.text(0.99, 0.01, "Copyright (c) Syam'ani 2025",
         transform=plt.gca().transAxes, fontsize=10,
         color='gray', ha='right', va='bottom', alpha=0.7)

# Set x-ticks to show only one label per year
year_ticks = list(range(0, len(monthly_labels), 12))
year_labels = [str(y) for y in years_list]
year_ticks = [i + 6 for i in year_ticks]
plt.xticks(year_ticks, year_labels, rotation=90, fontsize=11)
plt.xlabel('Year')
plt.ylabel('Monthly Mean NDTI from Landsat-4/5/7/8/9 (mg/L)')
plt.title('Long Term Monthly NDTI of Barito River outlet (1990-2024) \n(Linear Trend with 95% Confidence Interval)')
plt.legend()
plt.tight_layout()
plt.show()
```

Berikut adalah outputnya:



Jika kita mengamati output plot regresi linier di atas, maka akan terlihat bahwa koefisien determinasi ( $R^2$ -nya sangat rendah, yaitu 0,046. Dari sini dapat disimpulkan bahwa kecenderungan fluktuasi kekeruhan muara Sungai Barito dari waktu ke waktu sangat jauh dari linier. Untuk dapat dikatakan linier, setidaknya nilai  $R^2$  adalah 0,6.

## Kecenderungan Non-Linier

Berbeda dengan kecenderungan linier, kecenderungan non-linier berasumsi bahwa kenaikan atau penurunan data normal tidak mengikuti pola garis lurus. Kecenderungan non-linier dapat dianalisis menggunakan regresi non-linier. Terdapat banyak jenis regresi non-linier, yang cukup populer antara lain adalah regresi polinomial. Regresi polinomial merupakan bentuk perluasan dari regresi linier. Dimana pada regresi linier hanya ada variabel  $x$ , sementara pada regresi polinomial variabel  $x$  diperluas menjadi  $x^2, x^3, x^4, \dots x^n$ . Pada latihan kecenderungan non-linier ini, kita hanya akan membatasi sampai  $x^2$  (polinomial orde 2 atau *quadratic regression*) dan  $x^3$  (polinomial orde 3 atau *cubic regression*). Silahkan ketikkan dan jalankan kode-kode berikut untuk mengimplementasikan regresi polinomial orde 2:

```
# Perform Polynomial (order 2) regression on the Monthly NDTI values
from numpy.polynomial import Polynomial

# Fit polynomial of degree 2
coefs = np.polyfit(x[mask], y_vals, 2)
p = np.poly1d(coefs)
y_poly = p(x)

# Calculate R^2 for polynomial fit
r2_poly = r2_score(y_vals, p(x[mask]))
```

Pada kode-kode di atas, terlihat bahwa proses kalkulasi regresi polinomial dilakukan menggunakan NumPy. `np.polyfit(x[mask], y_vals, 2)` merupakan instruksi untuk menerapkan regresi polinomial orde 2. Sebagaimana regresi linier sebelumnya, selanjutnya kita dapat memplotkan hasil analisis regresi polinomial orde 2 dengan kode-kode berikut:

```
# Plot time series with polynomial (order 2) regression and confidence interval
plt.figure(figsize=(16, 5))
plt.plot(monthly_labels, monthly_ndti_values, marker='o', label='Monthly Mean NDTI')
plt.plot(monthly_labels, y_poly, "g--", label='Poly2 Trend')

# Calculate 95% confidence interval for polynomial regression
n = len(y_vals)
p_order = 2
y_poly_masked = p(x[mask])
residuals = y_vals - y_poly_masked
se = np.sqrt(np.sum(residuals ** 2) / (n - (p_order + 1)))
t_val = t.ppf(0.975, df=n - (p_order + 1))
mean_x = np.mean(x[mask])
Sxx = np.sum((x[mask] - mean_x) ** 2)
conf_int = t_val * se * np.sqrt(
    1/n + (x - mean_x) ** 2 / Sxx
)
plt.fill_between(
    monthly_labels,
    y_poly - conf_int,
    y_poly + conf_int,
    color='green',
    alpha=0.2,
    label='95% Confidence Interval'
)

# Annotate equation and R^2
eq_poly = f'y = {coefs[0]:.3e}x^2 + {coefs[1]:.3e}x + {coefs[2]:.2f}\n$R^2$ = {r2_poly:.3f}'
plt.text(0.01, 0.95, eq_poly, transform=plt.gca().transAxes,
         fontsize=12, verticalalignment='top',
         bbox=dict(boxstyle="round", fc="w", ec="0.5", alpha=0.8))

plt.text(0.99, 0.01, "copyright (c) Syam'ani 2025",
         transform=plt.gca().transAxes, fontsize=10,
```

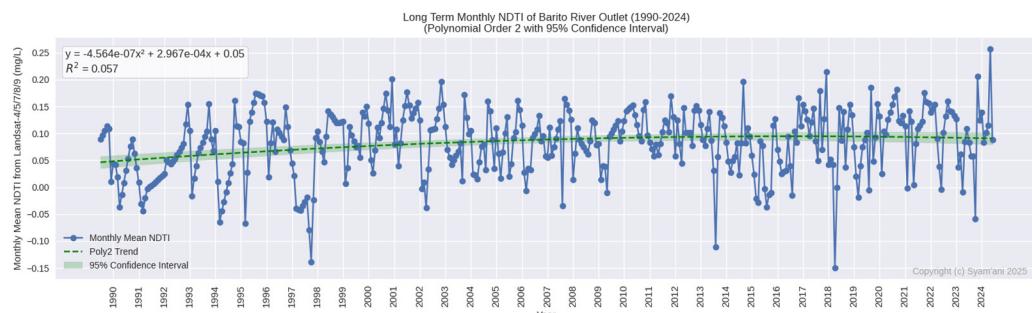
```

color='gray', ha='right', va='bottom', alpha=0.7)

# Set x-ticks to show only one label per year
plt.xticks(year_ticks, year_labels, rotation=90, fontsize=11)
plt.xlabel('Year')
plt.ylabel('Monthly Mean NDTI from Landsat-4/5/7/8/9 (mg/L)')
plt.title('Long Term Monthly NDTI of Barito River Outlet (1990-2024)' \
'\n(Polynomial order 2 with 95% Confidence Interval)')
plt.legend()
plt.tight_layout()
plt.show()

```

Berikut adalah outputnya:



Pada gambar plot di atas, regresi polinomial orde 2 juga memiliki koefisien determinasi rendah. Selanjutnya, silahkan ketikkan dan jalankan kode-kode berikut untuk mengimplementasikan dan memvisualisasikan regresi polinomial orde 3:

```

# Perform Polynomial (order 3) regression on the Monthly NDTI values

# Fit polynomial of degree 3
coefs3 = np.polyfit(x[mask], y_vals, 3)
p3 = np.poly1d(coefs3)
y_poly3 = p3(x)

# Calculate R^2 for polynomial fit (order 3)
r2_poly3 = r2_score(y_vals, p3(x[mask]))

```

```

# Plot time series with polynomial (order 3) regression and confidence interval
plt.figure(figsize=(16, 5))
plt.plot(monthly_labels, monthly_ndti_values, marker='o', label='Monthly Mean NDTI')
plt.plot(monthly_labels, y_poly3, "m--", label='Poly3 Trend')

# Calculate 95% confidence interval for polynomial regression (order 3)
n = len(y_vals)
p_order3 = 3
y_poly3_masked = p3(x[mask])
residuals3 = y_vals - y_poly3_masked
se3 = np.sqrt(np.sum(residuals3 ** 2) / (n - (p_order3 + 1)))
t_val3 = t.ppf(0.975, df=n - (p_order3 + 1))
mean_x3 = np.mean(x[mask])
Sxx3 = np.sum((x[mask] - mean_x3) ** 2)
conf_int3 = t_val3 * se3 * np.sqrt(
    1/n + (x - mean_x3) ** 2 / Sxx3
)
plt.fill_between(
    monthly_labels,
    y_poly3 - conf_int3,
    y_poly3 + conf_int3,
    color='magenta',
    alpha=0.2,
    label='95% Confidence Interval'
)

```

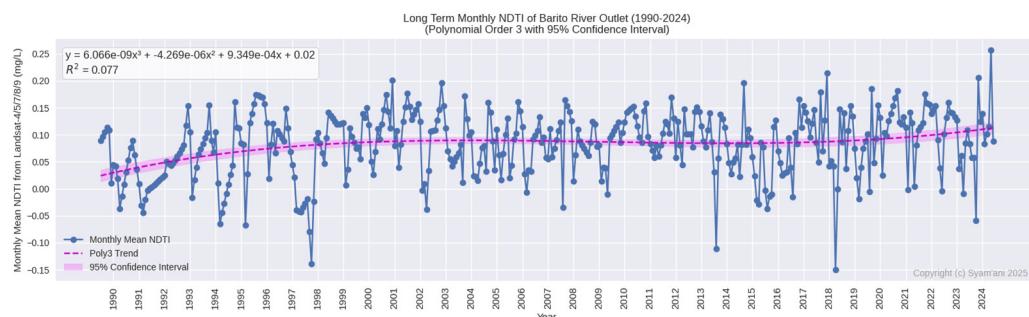
### Bab III Google Earth Engine

```
# Annotate equation and R^2
eq_poly3 =
    f"y = {coefs3[0]:.3e}x³ + {coefs3[1]:.3e}x² + {coefs3[2]:.3e}x +
    {coefs3[3]:.2f}\n"
    f"${R^2}$ = {r2_poly3:.3f}"
)
plt.text(0.01, 0.95, eq_poly3, transform=plt.gca().transAxes,
    fontsize=12, verticalalignment='top',
    bbox=dict(boxstyle="round", fc="w", ec="0.5", alpha=0.8))

plt.text(0.99, 0.01, "Copyright (c) Syam'ani 2025",
    transform=plt.gca().transAxes, fontsize=10,
    color='gray', ha='right', va='bottom', alpha=0.7)

# Set x-ticks to show only one label per year
plt.xticks(year_ticks, year_labels, rotation=90, fontsize=11)
plt.xlabel('Year')
plt.ylabel('Monthly Mean NDTI from Landsat-4/5/7/8/9 (mg/L)')
plt.title('Long Term Monthly NDTI of Barito River Outlet (1990-2024) \
    \n(Polynomial Order 3 with 95% Confidence Interval)')
plt.legend()
plt.tight_layout()
plt.show()
```

Berikut adalah outputnya:



Dari gambar plot di atas terlihat, bahwa meskipun lebih tinggi dibandingkan dengan regresi linier dan polinomial orde 2, regresi polinomial orde 3 juga belum memberikan nilai  $R^2$  yang kuat.

### Kecenderungan Harmonik

Kecenderungan harmonik berasumsi bahwa data normal mengikuti pola fluktuatif atau naik turun secara reguler. Kecenderungan harmonik dapat dianalisis dengan fungsi sinusoidal. Fungsi sinusoidal adalah fungsi yang menggambarkan gerakan periodik berulang, seperti gelombang sinus atau kosinus. Fungsi ini memiliki bentuk gelombang yang halus dan teratur, berosilasi antara nilai maksimum dan minimum. Silahkan ketikkan dan jalankan kode-kode berikut untuk mengimplementasikan dan memvisualisasikan fungsi sinusoidal pada data NDTI rerata bulanan:

```
# Fit a harmonic (sinusoidal) trend to the Monthly NDTI data
from scipy.optimize import curve_fit

# Define a harmonic (sinusoidal) trend function
def harmonic_trend(x, a0, a1, w, phi):
    return a0 + a1 * np.sin(w * x + phi)

# Fit the harmonic model to the data
popt, pcov = curve_fit(
    harmonic_trend,
    x[mask],
    y_vals,
    p0=[np.mean(y_vals), np.std(y_vals), 2 * np.pi / 12, 0], # initial guess
```

```

    maxfev=10000
)

# Generate fitted values
y_harmonic = harmonic_trend(x, *popt)

# Calculate R^2 for the harmonic fit
ss_res = np.sum((y_vals - harmonic_trend(x[mask], *popt)) ** 2)
ss_tot = np.sum((y_vals - np.mean(y_vals)) ** 2)
r2_harmonic = 1 - ss_res / ss_tot

```

---

```

# Plot time series with harmonic (sinusoidal) regression and confidence interval
plt.figure(figsize=(16, 5))
plt.plot(monthly_labels, monthly_ndti_values, marker='o', label='Monthly Mean NDTI')
plt.plot(monthly_labels, y_harmonic, "r--", label='Harmonic Trend')

# calculate 95% confidence interval for harmonic regression
n = len(y_vals)
p_order_harmonic = 4 # Number of parameters in harmonic_trend
residuals_harmonic = y_vals - harmonic_trend(x[mask], *popt)
se_harmonic = np.sqrt(np.sum(residuals_harmonic ** 2) / (n - p_order_harmonic))
t_val_harmonic = t.ppf(0.975, df=n - p_order_harmonic)
conf_int_harmonic = t_val_harmonic * se_harmonic

plt.fill_between(
    monthly_labels,
    y_harmonic - conf_int_harmonic,
    y_harmonic + conf_int_harmonic,
    color='orange',
    alpha=0.2,
    label='95% Confidence Interval'
)

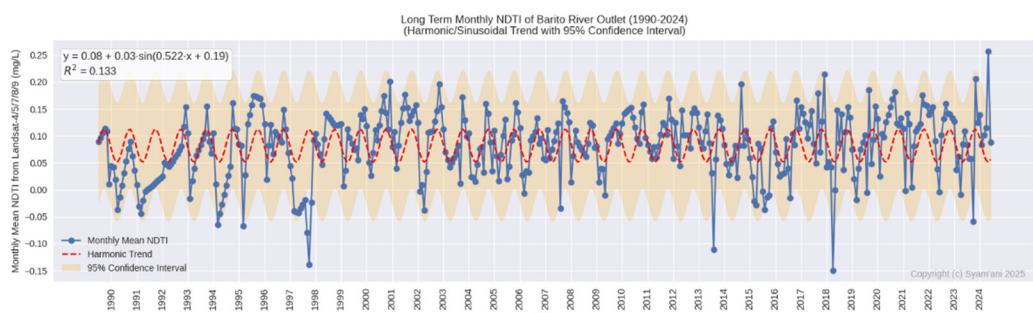
# Annotate equation and R^2
eq_harmonic = (
    f"y = {popt[0]:.2f} + {popt[1]:.2f}·sin({popt[2]:.3f}·x + {popt[3]:.2f})\n"
    f"$R^2$ = {r2_harmonic:.3f}"
)
plt.text(0.01, 0.95, eq_harmonic, transform=plt.gca().transAxes,
         fontsize=12, verticalalignment='top',
         bbox=dict(boxstyle="round", fc="w", ec="0.5", alpha=0.8))

plt.text(0.99, 0.01, "Copyright (c) Syam'ani 2025",
         transform=plt.gca().transAxes, fontsize=10,
         color='gray', ha='right', va='bottom', alpha=0.7)

# Set x-ticks to show only one label per year
plt.xticks(year_ticks, year_labels, rotation=90, fontsize=11)
plt.xlabel('Year')
plt.ylabel('Monthly Mean NDTI from Landsat-4/5/7/8/9 (mg/L)')
plt.title('Long Term Monthly NDTI of Barito River Outlet (1990-2024) '\
          '\n(Harmonic/Sinusoidal Trend with 95% Confidence Interval)')
plt.legend()
plt.tight_layout()
plt.show()

```

Berikut adalah outputnya:



Di antara semua model kecenderungan yang kita implementasikan, ternyata tidak ada yang memberikan koefisien determinasi yang cukup kuat. Meskipun kecenderungan harmonik memiliki nilai  $R^2$  tertinggi. Dari kondisi ini dapat disimpulkan bahwa fluktuasi kekeruhan muara Sungai Barito lebih bersifat acak dan sulit untuk diprediksi bagaimana kondisinya ke depan. Memang yang namanya kekeruhan, yaitu NDTI, sifatnya sangat dinamik, dapat berubah dalam hitungan jam, bahkan menit. Jika Anda mengujinya pada fitur-fitur lain, misalnya vegetasi dengan menggunakan NDVI, kemungkinan polanya akan lebih beraturan.

### Prediksi Deret Waktu

Meskipun pada contoh kasus NDTI muara Sungai Barito di atas, tidak ditemukan model yang dapat mendeskripsikan dengan baik kecenderungan kekeruhan muara Barito dari waktu ke waktu, tetapi sebagai latihan, kita akan mencoba untuk memprediksi kekeruhan muara Barito ke masa yang akan datang. Untuk keperluan uji akurasi, maka masa yang akan datang ini sebenarnya sudah terjadi, yaitu Januari sampai Juni 2025. Nanti akan kita bandingkan antara prediksi NDTI Januari sampai Juni 2025, dengan NDTI aktual Januari sampai Juni 2025.

```
# Predict NDTI for January to June 2025 using all regression models
import math
month = [420, 421, 422, 423, 424, 425] # January to June 2025
lin_ndti = reg.predict(np.array(month).reshape(-1, 1))
poly2_ndti = p(np.array(month))
poly3_ndti = p3(np.array(month))
harmonic_ndti = harmonic_trend(np.array(month), *popt)
# Print predicted NDTI values for January to June 2025
print(f"Predicted NDTI for January to June 2025:")
print(lin_ndti)
print(poly2_ndti)
print(poly3_ndti)
print(harmonic_ndti)
```

Pada kode-kode di atas, perhatikan `month = [420, 421, 422, 423, 424, 425]`. Ingat kembali penjelasan sebelumnya pada halaman 322, bahwa bulan terakhir dalam analisis kecenderungan adalah Desember 2024, atau bulan ke-420, dimana menurut aturan list merupakan indeks ke-419. Sehingga Januari 2025 adalah bulan ke-421 atau  $x = 420$ , begitu seterusnya, sampai Juni 2025 yang merupakan bulan ke-426 atau  $x = 425$ . Tentu saja, untuk memprediksi ke masa depan yang lebih jauh, Anda harus menyesuaikan hitungan bulannya, dimulai dari  $x = 0$  pada bulan Januari 1990.

Berikut adalah output prediksi NDTI rerata bulanan, secara berurutan dari Januari ke Juni 2025, untuk semua model, dari linier, polinomial orde 2, polinomial orde 3, dan harmonik.

```
Predicted NDTI for January to June 2025:
[0.10405617 0.10416162 0.10426707 0.10437252 0.10447798 0.10458343]
[0.09054078 0.09045361 0.09036553 0.09027654 0.09018663 0.09009582]
[0.11333462 0.11389716 0.11446648 0.11504263 0.11562563 0.11621553]
[0.06801796 0.08345274 0.09845339 0.10902408 0.11234901 0.1075425 ]
```

Selanjutnya, kita akan mengambil data Landsat 8 dan 9 dari Januari hingga Juni 2025, untuk menghitung NDTI rerata bulanan aktual. Dimana data NDTI rerata bulanan aktual ini nantinya akan digunakan sebagai pembanding terhadap data NDTI rerata bulanan hasil prediksi dari model-model regresi yang kita bangun sebelumnya.

```
# Load Landsat 8 Collection for January to June 2025
18_2025_col = (
    ee.ImageCollection('LANDSAT/LC08/C02/T1_L2')
    .filterDate('2025-01-01', '2025-07-01')
    .filter(ee.Filter.lt('CLOUD_COVER', 70))
    .map(apply_scale_factors)
    .map(mask_clouds)
)
```

```
# Load Landsat 9 Collection for January to June 2025
19_2025_col = (
    ee.ImageCollection('LANDSAT/LC09/C02/T1_L2')
    .filterDate('2025-01-01', '2025-07-01')
    .filter(ee.Filter.lt('CLOUD_COVER', 70))
    .map(apply_scale_factors)
    .map(mask_clouds)
)
```

```
# Calculate actual ndti for January to June 2025
ndti8_2025 = (
    18_2025_col.map(ndti_189)
    .select('NDTI')
)

ndti9_2025 = (
    19_2025_col.map(ndti_189)
    .select('NDTI')
)
```

```
# Merge Landsat 8 and 9 NDTI collections for January to June 2025
ndti_2025_col = ndti8_2025.merge(ndti9_2025)
```

```
# Calculate monthly mean NDTI for January to June 2025
# using actual Landsat 8/9 data

monthly_ndti_2025 = []
monthly_labels_2025 = []

for m in tqdm(range(1, 7)): # January to June
    start_date = f"2025-{m:02d}-01"
    if m == 6:
        end_date = "2025-07-01"
    else:
        end_date = f"2025-{m+1:02d}-01"
    # Filter images for the specific month
    ndti_month = (
        ndti_2025_col
        .filterDate(start_date, end_date)
        .mean()
        .rename('NDTI')
    )
    # Calculate mean NDTI over the area
    stats = ndti_month.reduceRegion(
        reducer=ee.Reducer.mean(),
        geometry=area,
        scale=30,
        maxPixels=1e13
    ). getInfo()
    ndti_mean = stats.get('NDTI', None)
    monthly_ndti_2025.append(ndti_mean)
    monthly_labels_2025.append(f"2025-{m:02d}")

print(monthly_ndti_2025)
```

Output:

```
[None, 0.1389704864501598, 0.10538566716431891, 0.7766597458592215,
0.009888250258468104, 0.13180278729667272]
```

### Bab III Google Earth Engine

Proses kalkulasi NDTI Januari-Juni 2025 di atas sama dengan proses kalkulasi NDTI sebelumnya. Dan kita dapat melihat bahwa data NDTI Januari 2025 ternyata kosong, kemungkinan disebabkan karena Citra Landsat 8 dan 9 akuisisi Januari 2025 penuh dengan awan. Konsekuensinya, kita harus menerapkan interpolasi temporal. Untuk menghindari nilai-nilai ekstrem, penciran pada data aktual Januari – Juni 2025 juga akan dieliminasi, dengan menggunakan data hasil IQR sebelumnya. Silahkan ketikkan atau copas dan jalankan kode-kode berikut:

```
# Interpolate missing data and anomalies in monthly_ndti_2025
# Convert to numpy array
monthly_ndti_2025_arr = np.array(monthly_ndti_2025, dtype=np.float32)

# Build anomaly mask: True for valid, False for missing/anomaly
anomaly_mask_2025 = np.array([
    False if (v is None or np.isnan(v) or np.isinf(v)
              or v < lower_bound or v > upper_bound)
    else True for v in monthly_ndti_2025_arr
])

# Find indices where anomaly_mask is False (i.e., anomalies)
anomaly_indices_2025 = np.where(~anomaly_mask_2025)[0]

# Interpolate only over valid (non-anomaly) values
valid_idx_2025 = np.where(anomaly_mask_2025)[0]
valid_x_2025 = valid_idx_2025
valid_y_2025 = monthly_ndti_2025_arr[valid_idx_2025]

# Create interpolation function if there are at least two valid points
if len(valid_x_2025) > 1:
    f_interp_2025 = interp1d(valid_x_2025, valid_y_2025, kind='linear',
                             fill_value='extrapolate')
    interp_values_2025 = f_interp_2025(anomaly_indices_2025)
    for idx, val in zip(anomaly_indices_2025, interp_values_2025):
        monthly_ndti_2025_arr[idx] = float(val)
elif len(valid_x_2025) == 1:
    # Only one valid value, fill all with that value
    monthly_ndti_2025_arr[anomaly_indices_2025] = valid_y_2025[0]
# else: all values are invalid, do nothing

# Update the list with interpolated values
monthly_ndti_2025 = monthly_ndti_2025_arr.tolist()
print(monthly_ndti_2025)
```

Output:

```
[0.17255529761314392, 0.1389704793691635, 0.1053856685757637,
 0.057636961340904236, 0.00988825038075447, 0.1318027824163437]
```

Langkah berikutnya, kita akan melakukan uji akurasi hasil prediksi NDTI masa depan dengan menggunakan *Root Mean Square Difference* (RMSD). Dimana konsep RMSD sama dengan RMSE yang sudah pernah kita bahas sebelumnya, lihat kembali halaman 242.

```
# Calculate Root Mean Square Difference (RMSD)
# between predicted and actual NDTI for all regression models
from sklearn.metrics import mean_squared_error

# Ensure monthly_ndti_2025 and predictions are numpy arrays
actual_2025 = np.array(monthly_ndti_2025, dtype=np.float32)
```

```
# Compute RMSD for each model
rmsd_lin = np.sqrt(mean_squared_error(actual_2025, lin_ndti))
rmsd_poly2 = np.sqrt(mean_squared_error(actual_2025, poly2_ndti))
rmsd_poly3 = np.sqrt(mean_squared_error(actual_2025, poly3_ndti))
```

```
rmsd_harmonic = np.sqrt(mean_squared_error(actual_2025, harmonic_ndti))

print("RMSD for January-June 2025:")
print(f"Linear Regression: {rmsd_lin:.5f}")
print(f"Polynomial (Order 2): {rmsd_poly2:.5f}")
print(f"Polynomial (Order 3): {rmsd_poly3:.5f}")
print(f"Harmonic (sinusoidal): {rmsd_harmonic:.5f}")
```

Output RMSD:

```
RMSD for January-June 2025:
Linear Regression: 0.05443
Polynomial (Order 2): 0.05562
Polynomial (Order 3): 0.05618
Harmonic (sinusoidal): 0.06805
```

Cukup mengejutkan, dari output RMSD di atas, terlihat bahwa nilai RMSD terkecil atau model yang paling akurat dalam memprediksi NDTI masa depan adalah regresi linier. Kita dapat memvisualisasikan perbandingan NDTI prediksi dan NDTI aktual ke dalam bentuk plot.

```
# visualize actual NDTI 2025 and predicted NDTI 2025 for all regression models
plt.style.use('seaborn-v0_8')
plt.figure(figsize=(8, 4))

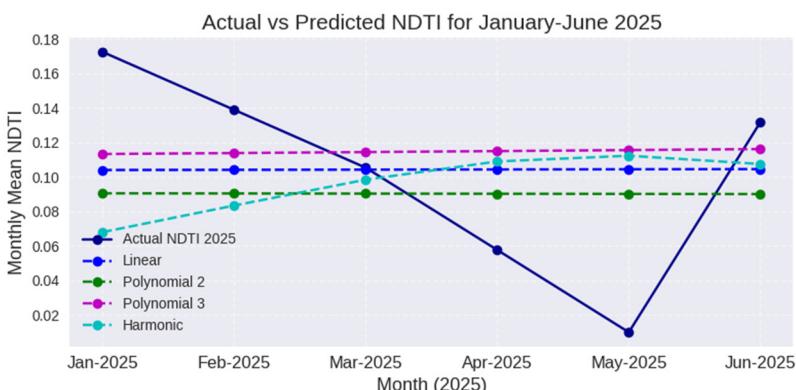
# X-axis: months Jan-Jun 2025
months_2025 = ['Jan-2025', 'Feb-2025', 'Mar-2025', 'Apr-2025', 'May-2025', 'Jun-2025']
x_2025 = np.arange(6)

# Plot actual NDTI
plt.plot(x_2025, monthly_ndti_2025, 'darkblue', marker='o', label='Actual NDTI 2025')

# Plot predictions from all models
plt.plot(x_2025, lin_ndti, 'b--o', label='Linear')
plt.plot(x_2025, poly2_ndti, 'g--o', label='Polynomial 2')
plt.plot(x_2025, poly3_ndti, 'm--o', label='Polynomial 3')
plt.plot(x_2025, harmonic_ndti, 'c--o', label='Harmonic')

plt.xticks(x_2025, months_2025, fontsize=12)
plt.xlabel('Month (2025)', fontsize=13)
plt.ylabel('Monthly Mean NDTI', fontsize=13)
plt.title('Actual vs Predicted NDTI for January-June 2025', fontsize=15)
plt.legend()
plt.grid(True, linestyle='--', alpha=0.5)
plt.tight_layout()
plt.show()
```

Output:



### Bab III Google Earth Engine

Kita juga dapat mencari kondisi kekeruhan terparah muara Sungai Barito sepanjang 1990-2024:

```
# Find the highest NDTI value and its corresponding year and month
max_ndti = np.nanmax(y_vals)
max_idx = np.nanargmax(y_vals)

# calculate year and month from index
year = years_list[max_idx // 12]
month = (max_idx % 12) + 1

print(f"Highest NDTI: {max_ndti:.5f} at {year}-{month:02d}")
```

Output:

```
Highest NDTI: 0.25756 at 2024-11
```

Ternyata NDTI rerata bulanan tertinggi muara Sungai Barito terjadi pada November 2024. Sekarang kita akan mencoba melihat bagaimana kondisi terparah pada November 2024 tersebut.

```
# Visualize the highest NDTI at 2024-11

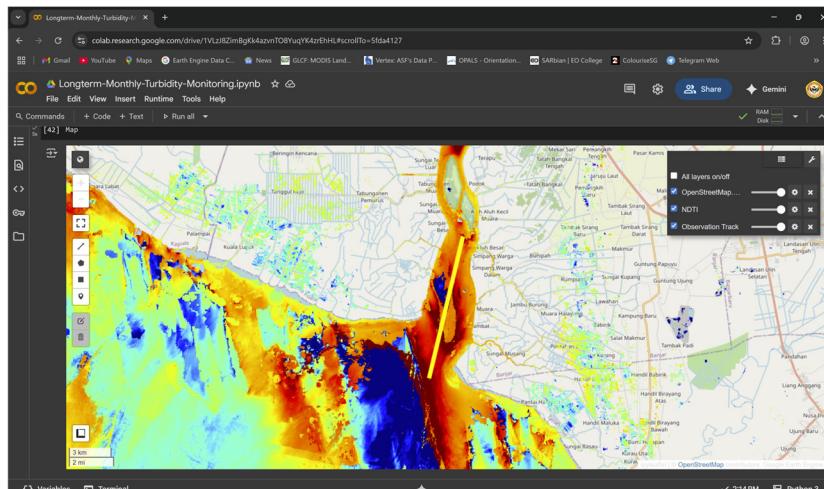
# Set year and month
vis_year = 2024
vis_month = 11

# Filter the monthly_ndti ImageCollection for the specified year and month
img_ndti_2024_11 = (
    monthly_ndti
    .filter(ee.Filter.eq('year', vis_year))
    .filter(ee.Filter.eq('month', vis_month))
    .first()
)

# Visualization parameters for NDTI
ndti_vis = {'min': -0.25, 'max': 0.25, 'palette': 'jet'}

Map = geemap.Map()
Map.centerObject(area, 12)
Map.addLayer(img_ndti_2024_11, ndti_vis, 'NDTI')
Map.addLayer(line, {'color': 'yellow', 'width': 8}, 'Observation Track')
Map
```

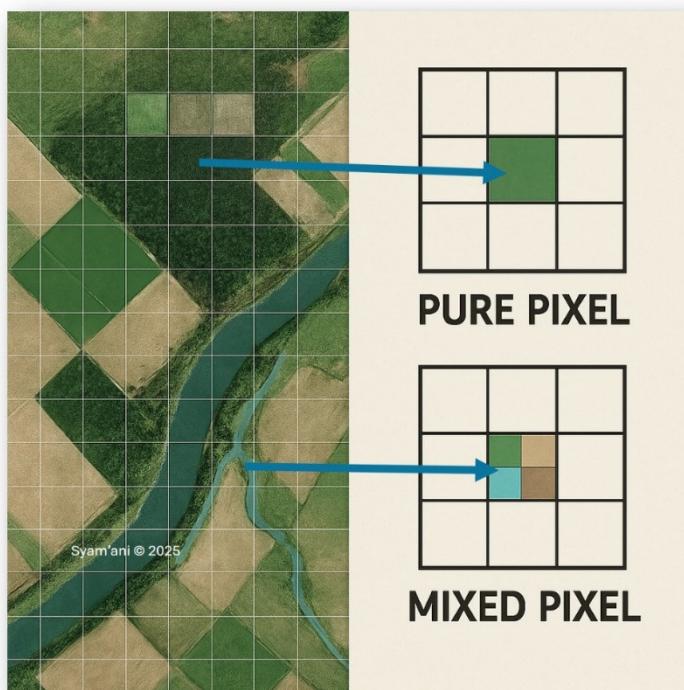
Output:



## I. Analisis Sub-Pixel

Superpixel dan pixelwise classification memiliki satu kelemahan utama, yaitu menganggap bahwa setiap pixel hanya terdiri atas satu objek. Untuk citra-citra satelit resolusi spasial medium seperti Landsat hal ini dapat menjadi masalah. Sebab satu pixel Landsat yang berukuran 30 x 30 meter di lapangan dapat saja terkomposisi atas banyak fitur di dalamnya. Meskipun terdapat juga kemungkinan pixel-pixel yang hanya memuat satu fitur di dalamnya. Pixel-pixel yang hanya memuat satu fitur di dalamnya (misalnya hanya air) dikenal sebagai *pure pixel*. Sedangkan pixel-pixel dengan lebih dari satu jenis fitur di dalamnya dikenal sebagai *mixed pixel* (mixel).

Pure pixel pada umumnya akan mudah ditemukan pada objek-objek yang homogen, seperti tubuh perairan jernih, hutan yang sangat lebat dan luas, lahan pertanian yang seragam, permukaan aspal atau beton yang luas, permukaan batuan yang luas, gurun pasir, dan sebagainya. Di daerah-daerah yang kenampakannya relatif heterogen, seperti permukiman, mixed pixel biasanya akan lebih mendominasi. Ilustrasi perbedaan antara pure pixel dan mixed pixel dapat dilihat pada Gambar 3.7.

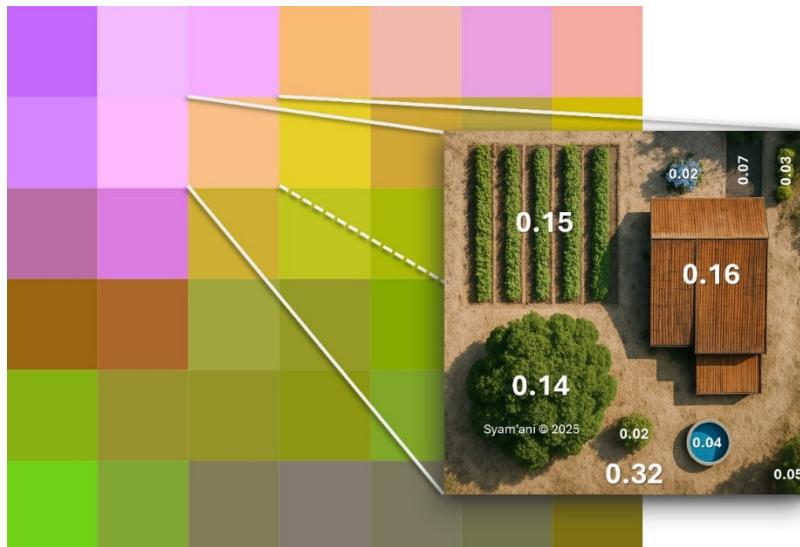


Gambar 3.8. Ilustrasi pure pixel dan mixed pixel (Copilot, 2025b)

Pada bagian terdahulu (halaman 255), sudah diulas tentang klasifikasi sub-pixel, atau disebut juga sebagai analisis sub-pixel. Analisis sub-pixel sebenarnya mencakup beberapa metode. Metode yang cukup populer adalah *spectral unmixing*, yang digunakan untuk mengekstrak informasi dari campuran kelas dalam satu pixel (Keshava and Mustard, 2002). Spectral unmixing pertama kali dikembangkan pada pertengahan 1980-an. Yaitu oleh John B. Adams, Milton O. Smith, dan Paul E. Johnson, ketika menganalisis tipe tanah dan batuan pada lokasi pendaratan Viking Lander 1, di dataran Chryse Planitia, Planet Mars (Adams et al., 1986). Jadi spectral unmixing pertama kali dikembangkan justru untuk observasi ekstra-terrestrial, bukan observasi bumi.

### Bab III Google Earth Engine

Di dalam mixed pixel, setiap fitur yang ada di dalamnya, seperti bangunan, pohon, air, lahan terbuka, dan sebagainya, disebut sebagai endmember (Jensen, 2014). Setiap endmember memiliki rasio atau proporsi masing-masing, sesuai persentasi nilai spektralnya masing-masing di dalam suatu pixel. Proporsi ini dikenal sebagai fraksi endmember atau sub-pixel (Atkinson, 1997). Untuk lebih jelasnya silahkan lihat ilustrasi mixed pixel pada Gambar 3.9 berikut:



Gambar 3.9. Fraksi-fraksi endmember dalam satu mixed pixel (Copilot, 2025c)

Sebagaimana sudah dibahas pada bagian sebelumnya, bahwa untuk mengurai fraksi-fraksi endmember seperti pada Gambar 3.9 adalah menggunakan spectral unmixing. Versi yang paling populer dari spectral unmixing adalah *Linear Spectral Unmixing* (LSU), atau kadang disebut juga sebagai *Linear Spectral Mixture Analysis* (LSMA). LSU memiliki model matematika yang disebut *Linear Mixture Model* (LMM) sebagai berikut (Adams et al., 1993):

$$\rho_b = \sum_{i=1}^n f_i \rho_{i,b} + \varepsilon_b, \quad \forall f_i \in \mathbb{R}, 0 \leq f_i \leq 1, \sum_{i=1}^n f_i = 1$$

Dimana:

- $\rho_b$  : Reflectance pada band b
- $f_i$  : Fraksi endmember ke-i,  $f_i$  adalah bilangan nyata dengan nilai dari 0 sampai 1, dan jumlah  $f_i$  untuk setiap pixel adalah sama dengan 1
- $\rho_{i,b}$  : Sub-reflectance, yaitu reflectance untuk endmember ke-i pada band b
- $\varepsilon_b$  : Error pada band b

Sebagai contoh, pada satu pixel band NIR terdapat 42% mangrove dengan reflectance 0,6, 20% vegetasi non-mangrove dengan reflectance 0,47, 12% air dengan reflectance 0,15, dan 26% lahan terbuka dengan reflectance 0,25. Dengan asumsi error sama dengan 0, maka:

$$\begin{aligned} \rho_{nir} &= (0,42 * 0,6) + (0,2 * 0,47) + (0,12 * 0,15) + (0,26 * 0,25) \\ \Leftrightarrow \rho_{nir} &= 0,43 \end{aligned}$$

Dengan demikian, nilai reflectance pixel tersebut pada band NIR adalah 0,43. Akan tetapi, di dalam konsep “pengurai pixel” LSU, konsepnya dibalik. Sebab yang kita ketahui secara langsung dari citra satelit tentu saja adalah nilai reflectance ( $\rho_b$ ) untuk setiap pixel. Sehingga yang akan dicari adalah fraksinya, yaitu nilai  $f_i$  untuk setiap endmember. Jika nilai  $f_i$  sudah ditemukan, untuk selanjutnya, jika diperlukan nilai-nilai sub-reflectance ( $\rho_{i,b}$ ) dapat dicari menggunakan nilai fraksi ( $f_i$ ), reflectance ( $\rho_b$ ), dan error ( $\varepsilon_b$ ) untuk setiap pixel.

Terdapat banyak pendekatan yang sudah dikembangkan untuk menghitung fraksi endmember, termasuk menentukan endmembernya sendiri. Untuk penentuan endmembernya sendiri, yaitu fitur-fitur yang akan diekstrak untuk setiap pixel, dapat dilakukan secara subjektif. Misalnya dengan pendekatan pengetahuan terdahulu atau pengetahuan lokal. Hal ini yang akan kita kerjakan di dalam tutorial ini, dimana ada empat endmember yang akan kita ekstrak. Yaitu mangrove, vegetasi non-mangrove, permukiman/lahan terbuka, dan tubuh air. Syarat jumlah endmember adalah maksimum  $n - 1$ , dimana  $n$  adalah jumlah band citra yang digunakan untuk mengekstrak endmember (Keshava, 2003). Dengan demikian, jika citra yang digunakan memiliki 10 band, maksimum hanya dapat mengekstrak 9 endmember. Jika kita ingin mengekstrak endmember dalam jumlah besar, misalnya sampai klasifikasi spesies pohon, konsekuensinya kita harus menggunakan citra dengan banyak band, seperti citra hiperspektral.

Fokus kita sekarang adalah memilih metode untuk komputasi fraksi-fraksi endmember. Hal pertama yang harus dilakukan untuk menghitung fraksi-fraksi endmember adalah menemukan pure pixels. Untuk menemukan pure pixels tekniknya juga beragam. Baik secara manual, yaitu dengan *plotting* pixel ke citra yang memiliki resolusi spasial tinggi, atau bahkan *plotting* pixel secara langsung ke lapangan. Maupun dengan cara otomatis, misalnya menggunakan *Pixel Purity Index* (PPI). Metode PPI ini sangat direkomendasikan di dalam penunjukan *pure pixels*. Akan tetapi, sampai dengan saat buku ini ditulis, GEE belum mengimplementasikan algoritma PPI. Sehingga jika kita mau mengekstrak endmember di GEE dengan pendekatan PPI, komputasi PPI harus dilakukan di perangkat lunak lain, atau jika diinginkan algoritma PPI dapat kita implementasikan sendiri dengan kode-kode Python. Tentu saja, hal ini cukup rumit.

Teknik lainnya yang potensial untuk dapat diterapkan dalam penunjukan *pure pixels* adalah dengan pendekatan indeks-indeks nilai spektral. Misalnya pure pixel untuk vegetasi dengan menggunakan NDVI. Teknik ini didasarkan pada asumsi bahwa semakin tinggi nilai indeks spektral suatu pixel maka semakin murni pixel tersebut. Misalnya, semakin nilai NDVI mendekati 1, semakin tinggi kemurnian vegetasi di dalam pixel tersebut. Semakin tinggi nilai MNDWI, semakin tinggi kemurnian air di dalam vegetasi tersebut. Teknik praktis dengan pendekatan indeks-indeks nilai spektral ini yang akan kita terapkan di dalam tutorial analisis sub-pixel ini.

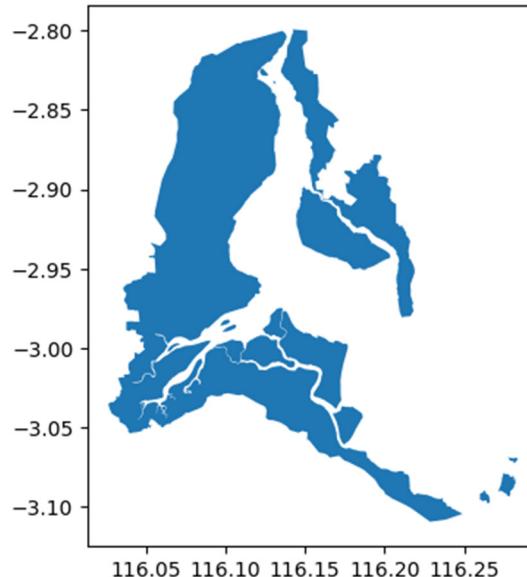
Sebagai contoh kasus, kita akan menggunakan metode LSU untuk klasifikasi kerapatan hutan mangrove di Cagar Alam Teluk Kelumpang, Kalimantan Selatan. Sekaligus menganalisis degradasi hutan mangrove dengan mendeteksi perubahan fraksi-fraksi mangrove antar waktu. Silahkan buat sebuah notebook Google Colab baru, kemudian beri nama misalnya **Analisis-Sub-Pixel.ipynb**. Selanjutnya silahkan ketikkan dan ekskusi kode-kode berikut:

```
import ee, geemap
ee.Authenticate()
ee.Initialize(project='ee-geospatialulm')
```

### Bab III Google Earth Engine

```
# Mengakses Google Drive
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
```

```
# Membuka shapefile
import geopandas as gpd
path = '/content/drive/My Drive/geebook/vector/'
kelumpang_shp = gpd.read_file(path + 'Kelumpang.shp')
kelumpang_shp.plot()
```



```
# Konversi shapefile ke GEE geometry dan membuat kotak batas wilayah
kelumpang_ee = geemap.geopandas_to_ee(kelumpang_shp)
kelumpang = kelumpang_ee.geometry().bounds()
```

```
# Penentuan tanggal akuisisi citra
start_date_2019 = '2019-04-01'
end_date_2019 = '2019-12-31'

start_date_2024 = '2024-04-01'
end_date_2024 = '2024-12-31'
```

Citra yang akan kita gunakan dalam analisis LSU adalah Sentinel-2 MSI *surface reflectance* multitemporal, yaitu akuisisi tahun 2019 dan tahun 2024. Sehingga degradasi mangrove yang terekstrak nantinya adalah degradasi mangrove dalam kurun waktu selama lima tahun.

```
# Masking awan
def maskS2clouds(image):
    qa = image.select('QA60')
    cloudBitMask = 1 << 10
    cirrusBitMask = 1 << 11
    mask = qa.bitwiseAnd(cloudBitMask).eq(0).And(
        qa.bitwiseAnd(cirrusBitMask).eq(0))

    return image.updateMask(mask).divide(10000)
```

```
# Mengakses Sentinel-2 image collection
s2_col_2019 = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED') \
    .filterBounds(kelumpang).filterDate(start_date_2019, end_date_2019) \
    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 50)).map(masks2clouds)

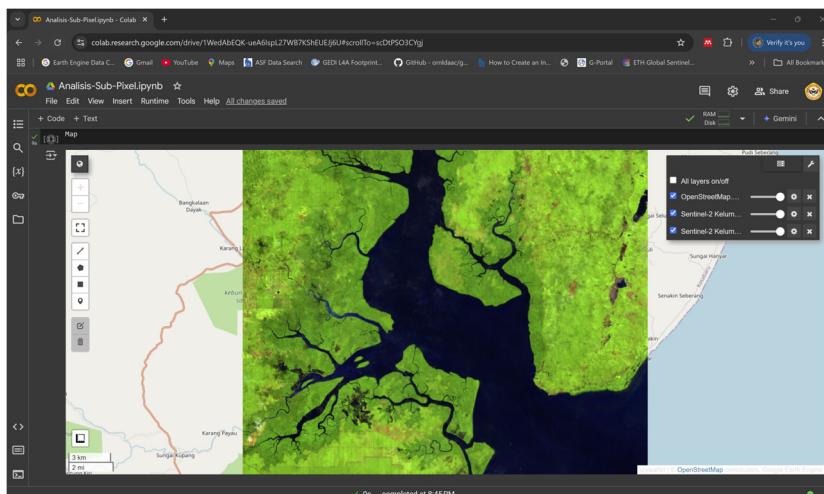
s2_col_2024 = ee.ImageCollection('COPERNICUS/S2_SR_HARMONIZED') \
    .filterBounds(kelumpang).filterDate(start_date_2024, end_date_2024) \
    .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 50)).map(masks2clouds)
```

```
# Penentuan band dan reduksi image collection menjadi image
s2_band_list = ['B2','B3','B4','B5','B6','B7','B8','B8A','B11','B12']

s2_image_2019 = s2_col_2019.median().clip(kelumpang).select(s2_band_list)
s2_image_2024 = s2_col_2024.median().clip(kelumpang).select(s2_band_list)
```

```
# Visualisasi Sentinel-2 RGB
rgb_vis = {'min': 0, 'max': 0.5, 'bands': ['B11','B8','B3']}

Map = geemap.Map()
Map.centerobject(kelumpang, zoom=12)
Map.addLayer(s2_image_2019, rgb_vis, 'Sentinel-2 Kelumpang 2019')
Map.addLayer(s2_image_2024, rgb_vis, 'Sentinel-2 Kelumpang 2024')
Map
```



```
# Seleksi band-band yang diperlukan untuk transformasi citra
green_2019 = s2_image_2019.select('B3')
red_2019 = s2_image_2019.select('B4')
nir_2019 = s2_image_2019.select('B8')
swirl1_2019 = s2_image_2019.select('B11')
swirl2_2019 = s2_image_2019.select('B12')

green_2024 = s2_image_2024.select('B3')
red_2024 = s2_image_2024.select('B4')
nir_2024 = s2_image_2024.select('B8')
swirl1_2024 = s2_image_2024.select('B11')
swirl2_2024 = s2_image_2024.select('B12')
```

Sebagaimana sudah dijelaskan sebelumnya, bahwa kita akan menggunakan indeks-indeks spektral untuk mengidentifikasi pure pixel. Untuk ekstraksi pure pixel mangrove kita akan menerapkan MVI , untuk ekstraksi pure pixel vegetasi non-mangrove kita akan menggunakan NDVI, untuk ekstraksi pure pixel air kita akan menggunakan MNDWI, dan untuk ekstraksi pure pixel urban (termasuk permukiman dan lahan terbuka) kita akan menggunakan UI. Formula keempat indeks nilai spektral ini sudah diulas pada bagian terdahulu. Di dalam ekstraksi pure pixel, nantinya akan dipilih nilai-nilai tertinggi untuk masing-masing indeks nilai spektral.

### Bab III Google Earth Engine

```
# Transformasi Mangrove Vegetation Index (MVI)
mvi_2019 = (
    (nir_2019.subtract(green_2019))
    .divide(swirl_2019.subtract(green_2019))
    .rename('MVI')
)

mvi_2024 = (
    (nir_2024.subtract(green_2024))
    .divide(swirl_2024.subtract(green_2024))
    .rename('MVI')
)
```

Berikutnya adalah ekstraksi sebaran mangrove dengan menggunakan nilai-nilai threshold mangrove pada MVI. Dimana pada Citra Sentinel-2, mangrove akan memiliki rentang nilai MVI dari 4,5 hingga 16,5 (Baloloy et al., 2020).

```
# Ekstraksi sebaran mangrove menggunakan thresholding MVI
mangrove_mask_2019 = (mvi_2019.gte(4.5)).And(mvi_2019.lte(16.5))
mangrove_mask_2024 = (mvi_2024.gte(4.5)).And(mvi_2024.lte(16.5))

non_mangrove_veg_2019 = (mvi_2019.lt(4.5)).Or(mvi_2019.gt(16.5))
non_mangrove_veg_2024 = (mvi_2024.lt(4.5)).Or(mvi_2024.gt(16.5))

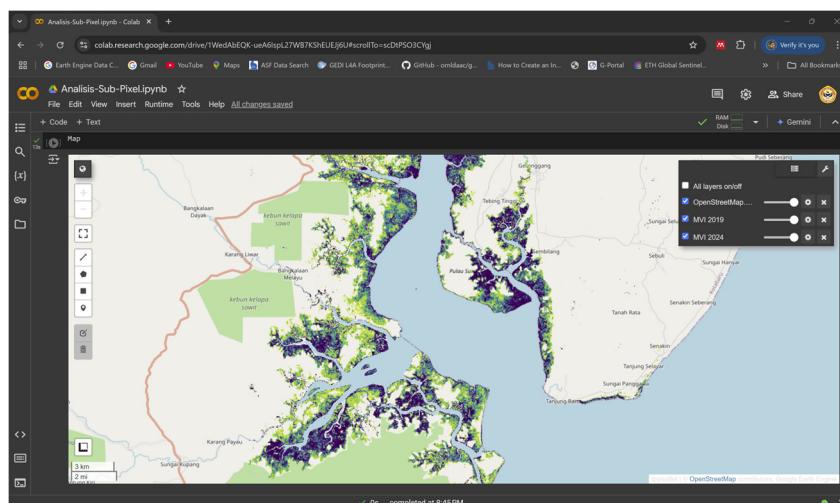
non_mangrove_mask_2019 = non_mangrove_veg_2019.updateMask(non_mangrove_veg_2019)
non_mangrove_mask_2024 = non_mangrove_veg_2024.updateMask(non_mangrove_veg_2024)

mvi_2019 = mvi_2019.updateMask(mangrove_mask_2019)
mvi_2024 = mvi_2024.updateMask(mangrove_mask_2024)
```

```
# Visualisasi MVI
mvi_vis = {'min': 4.5, 'max': 7, 'palette': 'viridis_r'}

Map = geemap.Map()
Map.centerObject(kelumpang, 12)
Map.addLayer(mvi_2019, mvi_vis, 'MVI 2019')
Map.addLayer(mvi_2024, mvi_vis, 'MVI 2024')
Map
```

Berikut adalah output visualisasi nilai-nilai MVI untuk mangrove pada tahun 2019 dan 2024:



Berikutnya adalah transformasi indeks-indeks spektral lainnya, yaitu NDVI, MNDWI, dan UI. Silahkan ketikkan dan jalankan kode-kode berikut:

```
# Transformasi Normalized Difference Vegetation Index (NDVI),
# Modified Normalized Difference Water Index (MNDWI), dan Urban Index (UI)
ndvi_2019 = (
    .nir_2019.subtract(red_2019))
    .divide(nir_2019.add(red_2019))
    .rename('NDVI')
).updateMask(non_mangrove_mask_2019)

ndvi_2024 = (
    .nir_2024.subtract(red_2024))
    .divide(nir_2024.add(red_2024))
    .rename('NDVI')
).updateMask(non_mangrove_mask_2024)

mndwi_2019 = (
    .green_2019.subtract(swirl_2019))
    .divide(green_2019.add(swirl_2019))
    .rename('MNDWI')
)

mndwi_2024 = (
    .green_2024.subtract(swirl_2024))
    .divide(green_2024.add(swirl_2024))
    .rename('MNDWI')
)

ui_2019 = (
    .swirl2_2019.subtract(nir_2019))
    .divide(swirl2_2019.add(nir_2019))
    .rename('UI')
)

ui_2024 = (
    .swirl2_2024.subtract(nir_2024))
    .divide(swirl2_2024.add(nir_2024))
    .rename('UI')
)
```

Langkah berikutnya adalah identifikasi pure pixel untuk seluruh endmember. Dalam hal ini, akan diambil masing 1.000 nilai-nilai pixel tertinggi untuk MVI dan NDVI, 500 nilai-nilai pixel tertinggi untuk MNDWI, dan 100 nilai-nilai pixel tertinggi untuk UI. Sebagaimana kode-kode berikut:

```
# Penentuan pure pixels 4 endmember menggunakan nilai indeks spektral tertinggi:
# Mangrove (MVI), Vegetasi Non-Mangrove (NDVI), Air (MNDWI), dan Urban (UI)

import numpy as np

# Jumlah pure pixels yang diinginkan untuk setiap endmember
mangrove_samples = 1000
non_mangrove_samples = 1000
water_samples = 500
urban_samples = 100

top_mvi_2019 = mvi_2019.reduceRegion(
    reducer=ee.Reducer.toList(),
    geometry=kelumpang,
    scale=10
)
purest_pixels = top_mvi_2019.get('MVI'). getInfo()
mangrove_2019 = np.unique(purest_pixels)[-mangrove_samples: ].tolist()

top_mvi_2024 = mvi_2024.reduceRegion(
    reducer=ee.Reducer.toList(),
    geometry=kelumpang,
    scale=10
)
purest_pixels = top_mvi_2024.get('MVI'). getInfo()
mangrove_2024 = np.unique(purest_pixels)[-mangrove_samples: ].tolist()

top_ndvi_2019 = ndvi_2019.reduceRegion(
    reducer=ee.Reducer.toList(),
    geometry=kelumpang,
```

### Bab III Google Earth Engine

```
scale=10
)
purest_pixels = top_ndvi_2019.getInfo()
non_mangrove_veg_2019 = (
    np.unique(purest_pixels)[-non_mangrove_samples:]
    .tolist()
)

top_ndvi_2024 = ndvi_2024.reduceRegion(
    reducer=ee.Reducer.toList(),
    geometry=kelumpang,
    scale=10
)
purest_pixels = top_ndvi_2024.getInfo()
non_mangrove_veg_2024 = (
    np.unique(purest_pixels)[-non_mangrove_samples:]
    .tolist()
)

top_mndwi_2019 = mndwi_2019.reduceRegion(
    reducer=ee.Reducer.toList(),
    geometry=kelumpang,
    scale=10
)
purest_pixels = top_mndwi_2019.getInfo()
water_2019 = np.unique(purest_pixels)[-water_samples:].tolist()

top_mndwi_2024 = mndwi_2024.reduceRegion(
    reducer=ee.Reducer.toList(),
    geometry=kelumpang,
    scale=10
)
purest_pixels = top_mndwi_2024.getInfo()
water_2024 = np.unique(purest_pixels)[-water_samples:].tolist()

top_ui_2019 = ui_2019.reduceRegion(
    reducer=ee.Reducer.toList(),
    geometry=kelumpang,
    scale=10
)
purest_pixels = top_ui_2019.getInfo()
urban_2019 = np.unique(purest_pixels)[-urban_samples:].tolist()

top_ui_2024 = ui_2024.reduceRegion(
    reducer=ee.Reducer.toList(),
    geometry=kelumpang,
    scale=10
)
purest_pixels = top_ui_2024.getInfo()
urban_2024 = np.unique(purest_pixels)[-urban_samples:].tolist()
```

Pada kode-kode di atas, setelah nilai-nilai tertinggi untuk masing-masing indeks spektral ditemukan, akan diambil nilai-nilai unik dari nilai-nilai tertinggi tersebut. Perhatikan contohnya pada kode `np.unique(purest_pixels)[-urban_samples: ].tolist()`. Instruksi `[-urban_samples: ]` merupakan teknik *array slicing*, artinya ambil `urban_samples` elemen terakhir dari array, selanjutnya NumPy array dikonversi ke list dengan fungsi `tolist()`. Langkah berikutnya, jika diperlukan, kita dapat menampilkan nilai-nilai unik tertinggi untuk masing-masing indeks spektral tersebut. Silahkan ketikkan dan jalankan kode-kode berikut:

```
# Menampilkan nilai-nilai pure pixels untuk masing-masing endmember
print(f'Mangrove 2019: {mangrove_2019}')
print(f'Mangrove 2024: {mangrove_2024}')
print(f'Non Mangrove Veg 2019: {non_mangrove_veg_2019}')
print(f'Non Mangrove Veg 2024: {non_mangrove_veg_2024}')
print(f'Water 2019: {water_2019}')
print(f'Water 2024: {water_2024}')
print(f'Urban 2019: {urban_2019}')
print(f'Urban 2024: {urban_2024}')
```

Berikut adalah outputnya:

```
Mangrove 2019: [16.18977928161621, 16.19025421142578, 16.190488815307617, 16.19068717956543, 1
Mangrove 2024: [16.11764907836914, 16.118282318115234, 16.11830711364746, 16.11880874633789, 1
Non Mangrove Veg 2019: [0.8750562071800232, 0.8750617504119873, 0.8750725388526917, 0.87507355
Non Mangrove Veg 2024: [0.9152892827987671, 0.915296733379364, 0.9152977466583252, 0.915298163
Water 2019: [0.8535714149475098, 0.8535730242729187, 0.8535745143890381, 0.8535773754119873, 0
Water 2024: [0.8516801595687866, 0.8516948819160461, 0.851703405380249, 0.8517241477966309, 0.
Urban 2019: [0.37813910841941833, 0.37857145071029663, 0.37886789441108704, 0.3790307343006134
Urban 2024: [0.44235923886299133, 0.4425634741783142, 0.4429979920387268, 0.44301632046699524,
```

Selanjutnya, dengan menggunakan nilai-nilai tertinggi unik di atas, kita akan menunjuk Lokasi-lokasi pure pixel tersebut di atas masing-masing citra.

```
# Penunjukan Lokasi-lokasi pure pixels di atas citra
# menggunakan nilai-nilai pixels untuk masing-masing endmember
mvi_2019_pure_mask = (
    mvi_2019.select('MVI')
    .gte(min(mangrove_2019))
    .And(mvi_2019.select('MVI'))
    .lte(max(mangrove_2019)))
)

mvi_2024_pure_mask = (
    mvi_2024.select('MVI')
    .gte(min(mangrove_2024))
    .And(mvi_2024.select('MVI'))
    .lte(max(mangrove_2024)))
)

ndvi_2019_pure_mask = (
    ndvi_2019.select('NDVI')
    .gte(min(non_mangrove_veg_2019))
    .And(ndvi_2019.select('NDVI'))
    .lte(max(non_mangrove_veg_2019)))
)

ndvi_2024_pure_mask = (
    ndvi_2024.select('NDVI')
    .gte(min(non_mangrove_veg_2024))
    .And(ndvi_2024.select('NDVI'))
    .lte(max(non_mangrove_veg_2024)))
)

mndwi_2019_pure_mask = (
    mndwi_2019.select('MNDWI')
    .gte(min(water_2019))
    .And(mndwi_2019.select('MNDWI'))
    .lte(max(water_2019)))
)

mndwi_2024_pure_mask = (
    mndwi_2024.select('MNDWI')
    .gte(min(water_2024))
    .And(mndwi_2024.select('MNDWI'))
    .lte(max(water_2024)))
)

ui_2019_pure_mask = (
    ui_2019.select('UI')
    .gte(min(urban_2019))
    .And(ui_2019.select('UI'))
    .lte(max(urban_2019)))
)

ui_2024_pure_mask = (
    ui_2024.select('UI')
    .gte(min(urban_2024))
    .And(ui_2024.select('UI'))
    .lte(max(urban_2024)))
)
```

```
mvi_2019_pure_mask = mvi_2019_pure_mask.updateMask(mvi_2019_pure_mask)
mvi_2024_pure_mask = mvi_2024_pure_mask.updateMask(mvi_2024_pure_mask)
ndvi_2019_pure_mask = ndvi_2019_pure_mask.updateMask(ndvi_2019_pure_mask)
ndvi_2024_pure_mask = ndvi_2024_pure_mask.updateMask(ndvi_2024_pure_mask)
mndwi_2019_pure_mask = mndwi_2019_pure_mask.updateMask(mndwi_2019_pure_mask)
mndwi_2024_pure_mask = mndwi_2024_pure_mask.updateMask(mndwi_2024_pure_mask)
ui_2019_pure_mask = ui_2019_pure_mask.updateMask(ui_2019_pure_mask)
ui_2024_pure_mask = ui_2024_pure_mask.updateMask(ui_2024_pure_mask)
```

Hasil penempatan Lokasi-lokasi pure pixel pada kode-kode di atas akan menghasilkan data pure pixel dalam bentuk GEE image. Selanjutnya, sebagai data input LSU di dalam GEE, kita disyaratkan untuk mengekstrak nilai-nilai spektral (*surface reflectance*) Sentinel-2 per band berdasarkan lokasi-lokasi pure pixel. Untuk keperluan seperti ini, GEE image pure pixel harus dikonversi menjadi feature collection terlebih dahulu. Silahkan ketikkan dan jalankan kode-kode berikut:

```
# Konversi pure pixels citra menjadi vector titik

mvi_2019_pure_mask_points = mvi_2019_pure_mask.reduceToVectors(
    geometry=kelumpang,
    crs=s2_image_2019.projection(),
    scale=10,
    geometryType='centroid',
    eightConnected=False,
    maxPixels=1e13
)

mvi_2024_pure_mask_points = mvi_2024_pure_mask.reduceToVectors(
    geometry=kelumpang,
    crs=s2_image_2024.projection(),
    scale=10,
    geometryType='centroid',
    eightConnected=False,
    maxPixels=1e13
)

ndvi_2019_pure_mask_points = ndvi_2019_pure_mask.reduceToVectors(
    geometry=kelumpang,
    crs=s2_image_2019.projection(),
    scale=10,
    geometryType='centroid',
    eightConnected=False,
    maxPixels=1e13
)

ndvi_2024_pure_mask_points = ndvi_2024_pure_mask.reduceToVectors(
    geometry=kelumpang,
    crs=s2_image_2024.projection(),
    scale=10,
    geometryType='centroid',
    eightConnected=False,
    maxPixels=1e13
)

mndwi_2019_pure_mask_points = mndwi_2019_pure_mask.reduceToVectors(
    geometry=kelumpang,
    crs=s2_image_2019.projection(),
    scale=10,
    geometryType='centroid',
    eightConnected=False,
    maxPixels=1e13
)

mndwi_2024_pure_mask_points = mndwi_2024_pure_mask.reduceToVectors(
    geometry=kelumpang,
    crs=s2_image_2024.projection(),
    scale=10,
    geometryType='centroid',
    eightConnected=False,
    maxPixels=1e13
)
```

```

ui_2019_pure_mask_points = ui_2019_pure_mask.reduceToVectors(
    geometry=kelumpang,
    crs=s2_image_2019.projection(),
    scale=10,
    geometryType='centroid',
    eightConnected=False,
    maxPixels=1e13
)

ui_2024_pure_mask_points = ui_2024_pure_mask.reduceToVectors(
    geometry=kelumpang,
    crs=s2_image_2024.projection(),
    scale=10,
    geometryType='centroid',
    eightConnected=False,
    maxPixels=1e13
)

```

Meskipun tidak disyaratkan, tetapi jika diperlukan, kita dapat memvisualisasikan data vektor pure pixel ke atas Citra Sentinel-2. Mungkin suatu saat nanti akan kita perlukan untuk kepentingan evaluasi secara visual.

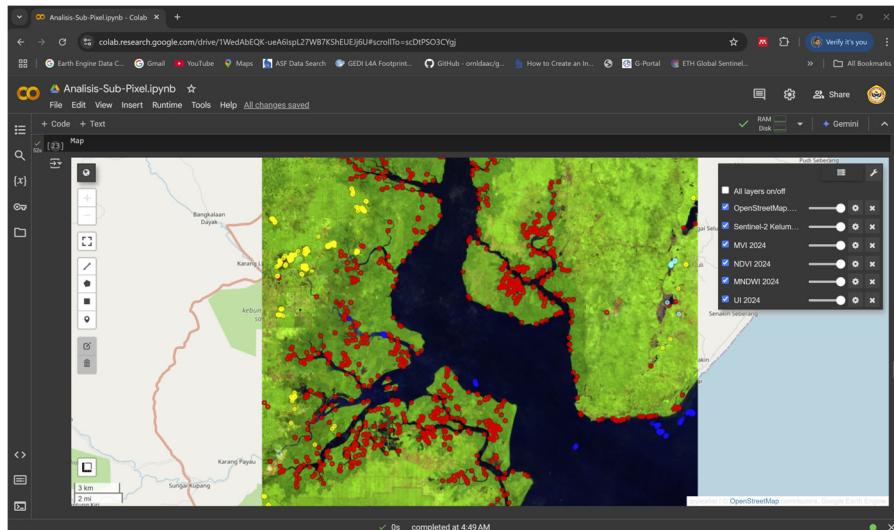
```

# Visualisasi titik-titik pure pixels untuk masing-masing endmember

Map = geemap.Map()
Map.centerObject(kelumpang, 12)
Map.addLayer(s2_image_2024, rgb_vis, 'Sentinel-2 Kelumpang 2024')
Map.addLayer(mvi_2024_pure_mask_points, {'color': 'red'}, 'MVI 2024')
Map.addLayer(ndvi_2024_pure_mask_points, {'color': 'yellow'}, 'NDVI 2024')
Map.addLayer(mndwi_2024_pure_mask_points, {'color': 'blue'}, 'MNDWI 2024')
Map.addLayer(ui_2024_pure_mask_points, {'color': 'cyan'}, 'UI 2024')
Map

```

Berikut adalah output visualisasinya:



Pada output di atas, merah merupakan pure pixel mangrove, kuning merupakan pure pixel vegetasi non-mangrove, biru merupakan pure pixel air, dan cyan merupakan pure pixel urban (lahan terbuka dan permukiman).

Berikutnya adalah proses ekstraksi nilai-nilai spektral rerata per endmember untuk setiap band Sentinel-2, dengan menggunakan data vektor pure pixel di atas.

### Bab III Google Earth Engine

```
# Fungsi untuk mengekstrak nilai-nilai pure pixel rerata per endmember
# pada setiap band Sentinel-2 yang sudah dipilih sebelumnya

def extract_pixel_mean(image, points, band_list):
    pixel_values = image.sampleRegions(
        collection=points,
        properties=[],
        scale=10
    )
    band_means = []
    for band in band_list:
        band_means.append(pixel_values.aggregate_mean(band).getInfo())
    return band_means
```

```
# Ekstraksi nilai-nilai pure pixel rerata
# per endmember untuk setiap band Sentinel-2

mangrove_2019_pure_mean = extract_pixel_mean(s2_image_2019,
mv1_2019_pure_mask_points, s2_band_list)
mangrove_2024_pure_mean = extract_pixel_mean(s2_image_2024,
mv1_2024_pure_mask_points, s2_band_list)
non_mangrove_veg_2019_pure_mean = extract_pixel_mean(s2_image_2019,
ndvi_2019_pure_mask_points, s2_band_list)
non_mangrove_veg_2024_pure_mean = extract_pixel_mean(s2_image_2024,
ndvi_2024_pure_mask_points, s2_band_list)
water_2019_pure_mean = extract_pixel_mean(s2_image_2019,
mndwi_2019_pure_mask_points, s2_band_list)
water_2024_pure_mean = extract_pixel_mean(s2_image_2024,
mndwi_2024_pure_mask_points, s2_band_list)
urban_2019_pure_mean = extract_pixel_mean(s2_image_2019,
ui_2019_pure_mask_points, s2_band_list)
urban_2024_pure_mean = extract_pixel_mean(s2_image_2024,
ui_2024_pure_mask_points, s2_band_list)
```

Sebagai langkah persiapan LSU, nilai-nilai spektral pure pixel rerata per band Sentinel-2 digabungkan untuk seluruh endmember, untuk setiap tahun akuisisi Sentinel-2.

```
# Penggabungan nilai-nilai pure pixel rerata setiap band Sentinel-2
# untuk semua endmember

endmember_2019 = (
    ee.List(mangrove_2019_pure_mean),
    ee.List(non_mangrove_veg_2019_pure_mean),
    ee.List(water_2019_pure_mean),
    ee.List(urban_2019_pure_mean)
)

endmember_2024 = (
    ee.List(mangrove_2024_pure_mean),
    ee.List(non_mangrove_veg_2024_pure_mean),
    ee.List(water_2024_pure_mean),
    ee.List(urban_2024_pure_mean)
)
```

Langkah berikutnya adalah implementasi LSU pada Citra Sentinel-2, dengan menggunakan data endmember yang sudah digabungkan sebelumnya. Terkait implementasi LSU pada GEE, silahkan baca dokumentasinya di tautan <https://developers.google.com/earth-engine/apidocs/ee-image-unmix>. Pada kode-kode di bawah, parameter nonNegative diset **True** agar di dalam output nanti tidak dihasilkan fraksi-fraksi endmember yang bernilai negatif. Parameter sumToOne juga diset **True** agar jumlah seluruh fraksi endmember untuk setiap pixel sama dengan 1 ( $\sum_{i=1}^n f_i = 1$ ). Silahkan lihat kembali konsep fraksi-fraksi endmember pada Gambar 3.9 dan LMM pada halaman 334 sebelumnya.

```
# Unmixing pixel-pixel Citra Sentinel-2 menggunakan endmember sebelumnya
sub_pixel_2019 = (
    s2_image_2019.unmix(endmember_2019, sumToOne=True, nonNegative=True)
    .rename(['Mangrove', 'Non-Mangrove', 'Water', 'Urban'])
)
sub_pixel_2024 = (
    s2_image_2024.unmix(endmember_2024, sumToOne=True, nonNegative=True)
    .rename(['Mangrove', 'Non-Mangrove', 'Water', 'Urban'])
)
```

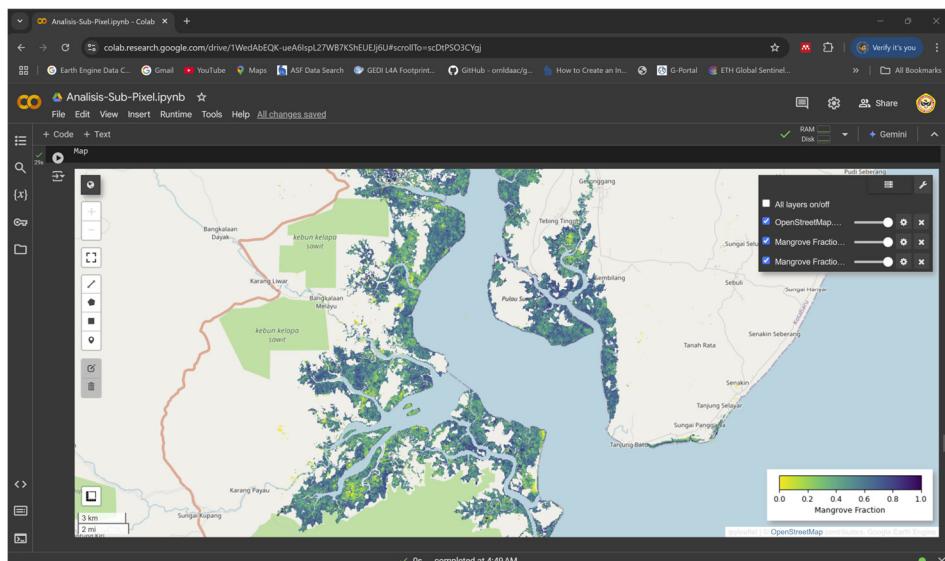
Karena analisis kita hanya berfokus pada degradasi hutan mangrove, maka kita hanya akan mengambil fraksi endmember mangrove. Kemudian fraksi mangrove ini akan kita masking menggunakan data sebaran mangrove yang kita ekstrak menggunakan thresholding MVI sebelumnya. Silahkan ketikkan dan jalankan kode-kode berikut:

```
# Seleksi fraksi mangrove pada citra hasil unmixing
mangrove_fraction_2019 = (
    sub_pixel_2019.select('Mangrove')
    .updateMask(mangrove_mask_2019)
)
mangrove_fraction_2024 = (
    sub_pixel_2024.select('Mangrove')
    .updateMask(mangrove_mask_2024)
)
```

Langkah berikutnya, kita dapat memvisualisasikan fraksi mangrove untuk tahun 2019 dan 2024.

```
# visualisasi fraksi mangrove tahun 2019 dan 2024
sub_vis = {'min': 0, 'max': 1, 'palette': 'viridis_r'}

Map = geemap.Map()
Map.centerobject(kelumpang, 12)
Map.addLayer(mangrove_fraction_2019, sub_vis, 'Mangrove Fraction 2019')
Map.addLayer(mangrove_fraction_2024, sub_vis, 'Mangrove Fraction 2024')
Map.add_colorbar(sub_vis, label='Mangrove Fraction')
Map
```



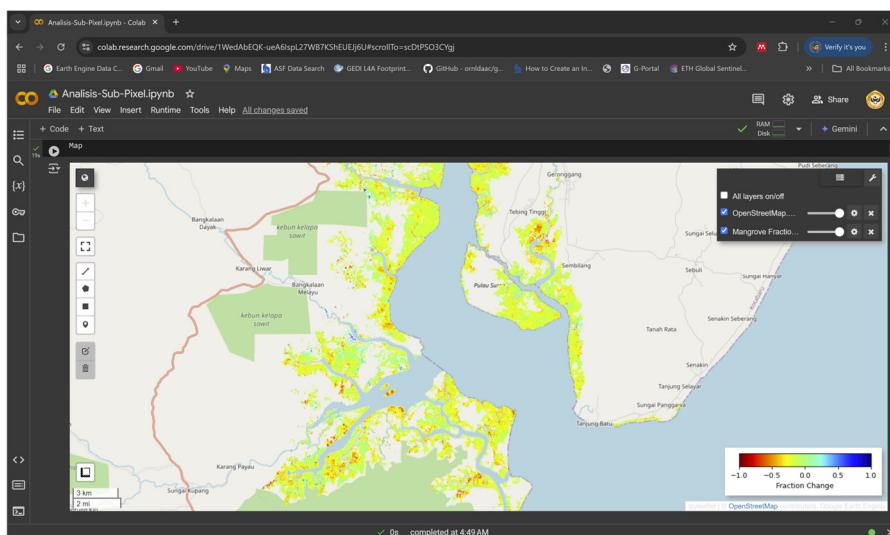
### Bab III Google Earth Engine

Ingat kembali, menurut konsep LMM (halaman 334), nilai fraksi setiap endmember pada setiap pixel akan berkisar dari 0 sampai 1. Jika suatu endmember memiliki nilai fraksi 0, berarti endmember tersebut tidak hadir di dalam pixel tersebut. Jika suatu endmember memiliki nilai fraksi 1, berarti pixel itu merupakan pure pixel yang berisi 100% endmember tersebut. Kita dapat secara langsung mengekstrak perubahan fraksi mangrove dari tahun 2019 ke tahun 2014 dengan metode subtraksi sederhana. Sebagaimana kode-kode berikut:

```
# Ekstraksi perubahan fraksi mangrove dari 2019 ke 2024
fraction_change = (
    mangrove_fraction_2024.subtract(mangrove_fraction_2019)
    .rename('Fraction Change')
)
```

```
# Visualisasi perubahan fraksi mangrove 2019-2024
deg_vis = {'min': -1, 'max': 1, 'palette': 'jet_r'}

Map = geemap.Map()
Map.centerObject(kelumpang, 12)
Map.addLayer(fraction_change, deg_vis, 'Mangrove Fraction Change')
Map.add_colorbar(deg_vis, label='Fraction Change')
Map
```



Karena fraksi mangrove memiliki rentang nilai 0 sampai 1, maka proses pengurangan fraksi mangrove bitemporal akan menghasilkan sebuah citra bernilai -1 sampai 1. Nilai-nilai ini proporsional dengan perubahan nilai-nilai fraksi mangrove secara proporsional.

Jika kita berasumsi bahwa nilai-nilai fraksi mangrove untuk setiap pixel ekivalen dengan persentase penutupan tajuk atau kanopi hutan mangrove, maka kita dapat mengklasifikasikan kerapatan mangrove menurut SNI 7717:2020, tentang Spesifikasi informasi geospasial – Mangrove skala 1:25.000 dan 1:50.000 (Badan Standardisasi Nasional, 2020). Dimana menurut SNI 7717:2020, kerapatan mangrove diklasifikasikan menjadi 3, yaitu Mangrove jarang (kerapatan tajuk < 30%), mangrove sedang (kerapatan tajuk 30–70%), dan mangrove lebat (kerapatan tajuk > 70%) (Badan Standardisasi Nasional, 2020). Dalam hal ini, persentase penutupan tajuk hutan mangrove akan kita terjemahkan dengan nilai-nilai fraksi mangrove, yaitu < 0,3, 0,3 – 0,7, dan > 0,7. Berikut adalah kode-kode implementasi klasifikasi kerapatan hutan mangrove sekaligus visualisasinya:

```
# Klasifikasi kerapatan mangrove berdasarkan nilai-nilai fraksi mangrove,
# mengacu ke SNI 7717:2020 tentang Spesifikasi Informasi Geospatial Mangrove
# Skala 1 : 25.000 dan 1 : 50.000

mangrove_2019_class = (
    (mangrove_fraction_2019.select('Mangrove').gt(0.7).multiply(3))
    .add(mangrove_fraction_2019.select('Mangrove').gte(0.3)
    .And(mangrove_fraction_2019.lte(0.7)).multiply(2))
    .add(mangrove_fraction_2019.select('Mangrove').lt(0.3).multiply(1))
    .rename('Class')
)

mangrove_2024_class = (
    (mangrove_fraction_2024.select('Mangrove').gt(0.7).multiply(3))
    .add(mangrove_fraction_2024.select('Mangrove').gte(0.3)
    .And(mangrove_fraction_2024.lte(0.7)).multiply(2))
    .add(mangrove_fraction_2024.select('Mangrove').lt(0.3).multiply(1))
    .rename('Class')
)
```

Pada klasifikasi kerapatan hutan mangrove di atas, Mangrove jarang diberi kode 1, Mangrove sedang diberi kode 2, dan Mangrove lebat diberi kode 3.

```
# Penentuan warna masing-masing kelas kerapatan mangrove

class_colors = ['#818c3c', '#25591f', '#19270d']

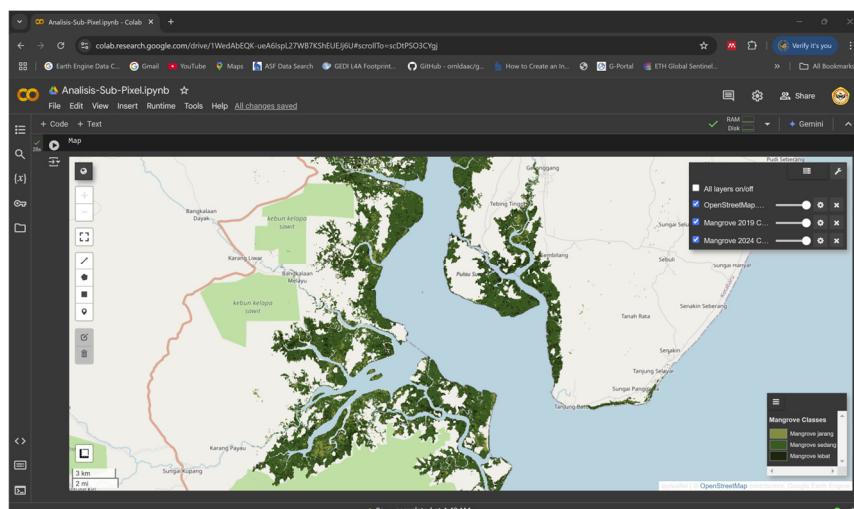
mangrove_legend = {
    'Mangrove jarang': class_colors[0],
    'Mangrove sedang': class_colors[1],
    'Mangrove lebat': class_colors[2]
}

# visualisasi kelas kerapatan mangrove tahun 2019 dan 2024

class_vis = {'min': 1, 'max': 3, 'palette': class_colors}

Map = geemap.Map()
Map.centerobject(kelumpang, 12)
Map.addLayer(mangrove_2019_class, class_vis, 'Mangrove 2019 Classes')
Map.addLayer(mangrove_2024_class, class_vis, 'Mangrove 2024 Classes')
Map.add_legend(title='Mangrove Classes', legend_dict=mangrove_legend)
Map
```

Berikut adalah output visualisasi kerapatan hutan mangrove tahun 2019 dan 2024:



### Bab III Google Earth Engine

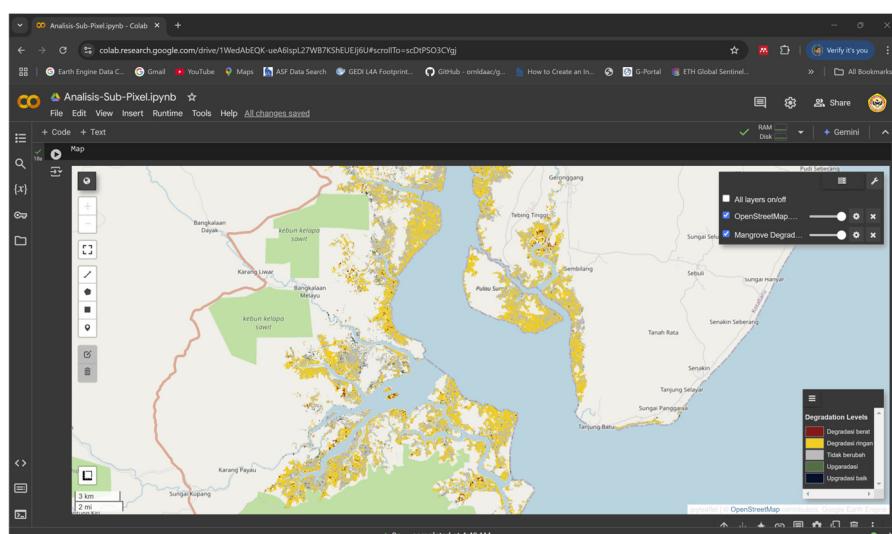
Langkah berikutnya adalah ekstraksi informasi degradasi hutan mangrove 2019-2024. Metodenya juga menggunakan subtraksi sederhana, yaitu kelas kerapatan mangrove 2024 dikurangkan dengan kelas kerapatan mangrove 2019.

```
# Ekstraksi informasi degradasi mangrove 2019-2024  
mangrove_degradation = (  
    mangrove_2024_class.subtract(mangrove_2019_class)  
    .rename('Mangrove Degradation')  
)
```

Karena kode-kode kelas kerapatan mangrove adalah 1 sampai 3, maka hasil subtraksi kelas kerapatan mangrove bitemporal akan menghasilkan citra dengan nilai pixel -2 sampai 2. Selanjutnya, kelas-kelas degradasi mangrove akan ditentukan dari nilai-nilai pixel hasil subtraksi ini. Jika nilainya -2 maka Degradasi berat, jika nilainya -1 maka Degradasi ringan, jika nilainya 0 maka Tidak berubah, jika nilainya 1 maka Upgradaasi, dan jika nilainya 2 maka Upgradasi baik.

```
# Penentuan warna masing-masing kelas degradasi mangrove  
deg_colors = ['#9a1c1c', '#ffcf17', '#bbbbbb', '#446d44', '#000f2c']  
deg_legend = {  
    'Degradasi berat': deg_colors[0],  
    'Degradasi ringan': deg_colors[1],  
    'Tidak berubah': deg_colors[2],  
    'Upgaradasi': deg_colors[3],  
    'Upgradasi baik': deg_colors[4]  
}  
  
# Visualisasi degradasi mangrove 2019-2024  
deg_vis = {'min': -2, 'max': 2, 'palette': deg_colors}  
Map = geemap.Map()  
Map.centerObject(kelumpang, 12)  
Map.addLayer(mangrove_degradation, deg_vis, 'Mangrove Degradations')  
Map.add_legend(title='Degradation Levels', legend_dict=deg_legend)  
Map
```

Berikut adalah output visualisasi degradasi hutan mangrove periode 2019-2024:



Langkah terakhir adalah kalkulasi luas masing-masing kelas degradasi mangrove. Silahkan ketikkan dan jalankan kode-kode berikut:

```
# Kalkulasi luas masing-masing kelas degradasi mangrove
area_berat = mangrove_degradation.eq(-2).reduceRegion(
    reducer=ee.Reducer.sum(),
    geometry=kelumpang,
    scale=100,
    maxPixels=1e13
).get('Mangrove Degradation'). getInfo()
area_ringan = mangrove_degradation.eq(-1).reduceRegion(
    reducer=ee.Reducer.sum(),
    geometry=kelumpang,
    scale=100,
    maxPixels=1e13
).get('Mangrove Degradation'). getInfo()
area_aman = mangrove_degradation.gte(0).reduceRegion(
    reducer=ee.Reducer.sum(),
    geometry=kelumpang,
    scale=100,
    maxPixels=1e13
).get('Mangrove Degradation'). getInfo()
print(f'Luas mangrove degradasi berat: {round(area_berat,2)} ha.')
print(f'Luas mangrove degradasi ringan: {round(area_ringan,2)} ha.')
print(f'Luas mangrove tidak terdegradasi: {round(area_aman,2)} ha.'')
```

Berikut adalah output luasannya:

```
Luas mangrove degradasi berat: 132.25 ha.
Luas mangrove degradasi ringan: 7041.62 ha.
Luas mangrove tidak terdegradasi: 5742.71 ha.
```

Karena fokus kita hanya pada degradasi hutan mangrove, maka kode-kode di atas hanya akan mengekstrak informasi detail luasan kelas-kelas degradasi, yaitu degradasi berat dan degradasi ringan. Sementara kelas tidak berubah, upgradasi, dan gradasi baik, ditotalkan luasannya menjadi tidak terdegradasi. Penjumlahan area-area yang tidak terdegradasi ini diimplementasikan dengan kode `area_aman = mangrove_degradation.gte(0).reduceRegion`. Tentu saja, jika Anda memerlukan informasi detail area yang terupgradasi dan terupgradasi baik, Anda harus memodifikasi kode-kode di atas. Instruksi `scale=100` pada kode-kode di atas berarti di dalam proses kalkulasi luas, dilakukan resampling resolusi pixel Sentinel-2 menjadi 100 meter. Hal ini sebenarnya kurang akurat, tetapi akan mempercepat proses kalkulasi dan mencegah isu limit memori. Untuk ketelitian dan akurasi informasi nantinya, sebaiknya Anda mengatur `scale=10`.

---

**PENAFIAN:** Metode ekstraksi pure pixel berbasis indeks-indeks spektral seperti yang diterapkan di dalam buku ini, disajikan hanya untuk tujuan tutorial, dan tidak mengacu ke literatur mana pun, sehingga belum teruji akurasinya. Untuk aplikasi praktis atau riset di dunia nyata, Anda tetap direkomendasikan untuk menggunakan PPI standar atau metode-metode lainnya yang sudah teruji di dalam identifikasi pure pixel.

---

## J. Visualisasi Data

Visualisasi atau penyajian informasi geospasial dalam bentuk visual biasanya merupakan tahap paling akhir dari analisis geospasial. Sebab dengan teknik visualisasi ini informasi geospasial didistribusikan dalam bentuk yang lebih informatif dan lebih mudah difahami oleh publik. Berbagai teknik visualisasi dapat diterapkan pada informasi geospasial, diantaranya adalah layout, plot, grafik, animasi, bentuk 3-Dimensi, dan sebagainya. Terkait dengan visualisasi informasi geospasial, memang GEE atau Google Colab tidak seinteraktif perangkat lunak SIG seperti ArcGIS Pro atau QGIS, setidaknya sampai saat buku ini ditulis. Sehingga kalau targetnya adalah kemudahan di dalam penyajian informasi geospasial, tentu saja Anda tetap direkomendasikan menggunakan perangkat lunak SIG.

Akan tetapi, visualisasi informasi geospasial secara langsung menggunakan GEE bukan tanpa kelebihan. Dengan mempertimbangkan bahwa kita tidak perlu mengunduh data, tidak perlu instalasi perangkat lunak di komputer sendiri, dan seluruh proses bergantung pada komputasi awan, maka hal ini adalah satu kelebihan visualisasi langsung menggunakan GEE. Termasuk ketika informasi yang harus kita sajikan cukup banyak. Sebagai contoh, jika Anda adalah seorang analis yang diminta untuk menyajikan layout presipitasi rerata tahunan selama 30 tahun terakhir, per kabupaten dalam satu provinsi. Dengan menggunakan GEE, Anda tidak perlu mengunduh dan memproses data di laptop Anda. Kemudian dengan teknik iterasi (**for** misalnya) Anda hanya cukup membuat satu set kode untuk menghasilkan banyak layout sekaligus. Bayangkan, akses data dalam jumlah besar, berbagai teknik analisis yang rumit, dan banyak layout, bila perlu ada tambahan grafik dan animasi, terpaket dalam satu kesatuan kode Python. Dan dengan sekali eksekusi, informasi geospasial langsung tersaji dan siap dipublikasikan. Terlebih kode-kode yang sudah kita bangun dapat dipakai ulang atau dimodifikasi di waktu yang akan datang.

### Timelapse Animation

Pada bagian terdahulu, kita sudah membuat timelapse animation hutan mangrove dengan menggunakan MVI. Agak sedikit berbeda, pada latihan ini kita akan membuat timelapse animation secara langsung dari Citra Sentinel-2 komposit RGB. Teknik ini umumnya dilakukan ketika kita ingin mengamati dinamika bentang lahan secara umum, tidak fokus ke fitur-fitur tertentu, seperti mangrove sebagaimana tutorial pada bagian terdahulu. Silahkan buat sebuah notebook Google Colab baru, kemudian beri nama misalnya **Sentinel2-Timelapse.ipynb**. Kemudian ketikkan dan eksekusi kode-kode berikut:

```
import ee, geemap  
  
ee.Authenticate()  
  
ee.Initialize(project='ee-geospatialulm')  
  
# Mengakses Google Drive untuk menyimpan animasi gif  
from google.colab import drive  
drive.mount('/content/drive', force_remount=True)  
path = '/content/drive/MyDrive/geebook/'  
  
# Menentukan titik pusat observasi  
point = ee.Geometry.Point([115.528648, -2.203888])
```

```
# Menentukan radius 8.000 meter dari titik pusat observasi
buffer_radius = 8000 # meters
```

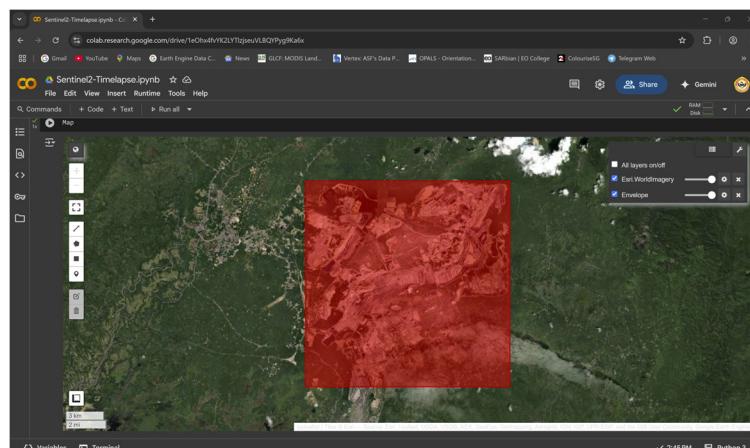
```
# Membuat buffer lingkaran dengan radius 8.000 meter dari titik
buffered_point = point.buffer(buffer_radius)

# Merubah buffer lingkaran menjadi kotak
envelope = buffered_point.bounds()
```

Pada kode-kode di atas kita membuat sebuah geometri titik dengan data koordinat bujur-lintang. Informasi koordinat ini didapatkan dari Google Maps. Jika diinginkan, Anda dapat merubah koordinat ini sesuai keperluan Anda. Dari titik koordinat ini, kemudian dibuat *buffer* dengan radius 8.000 meter. Untuk timelapse animation, disarankan wilayah jangan terlalu luas. Sebab nanti pemrosesan akan lama, bahkan berpotensi gagal karena melampaui kapasitas memori. Untuk evaluasi, kita dapat memvisualisasikan wilayah observasi kita.

```
# Visualisasi wilayah observasi
Map = geemap.Map(baseemap='SATELLITE')
Map.centerobject(point, 12)
Map.addLayer(envelope, {'color': 'red'}, 'Envelope')
Map
```

Output:



Selanjutnya, kita akan menentukan rentang waktu timelapse animation yang diinginkan.

```
# Menentukan waktu akuisisi citra
num_years = 10
last_year = 2025

start_date = []
end_date = []
years = []

for i in range(num_years):
    year = last_year - num_years + i + 1
    start_date.append(f'{year}-01-01')
    end_date.append(f'{year}-12-31')
    years.append(year)

print(start_date)
print(end_date)
print(years)
```

### Bab III Google Earth Engine

Output:

```
['2016-01-01', '2017-01-01', '2018-01-01', '2019-01-01', '2020-01-01', '2021-01-01', '2022-01-01', '2023-01-01', '2024-01-01', '2025-01-01']  
[['2016-12-31', '2017-12-31', '2018-12-31', '2019-12-31', '2020-12-31', '2021-12-31', '2022-12-31', '2023-12-31', '2024-12-31', '2025-12-31']]  
[2016, 2017, 2018, 2019, 2020, 2021, 2022, 2023, 2024, 2025]
```

Perhatikan kode-kode di atas, kita membuat rentang waktu 10 tahun (num\_years = 10), dengan tahun terakhir adalah 2025 (last\_year = 2025). Jika dihitung mundur, maka tahun awalnya adalah 2016, sebagaimana output di atas. Pastikan awal waktunya tidak lebih kurang dari 2016. Sebab meskipun Satelit Sentinel-2A sudah diorbitkan pada pertengahan tahun 2015, untuk wilayah Kalimantan Selatan Citra Sentinel-2 yang bersih baru tersedia mulai 2016.

Iterasi **for** pada kode-kode di atas digunakan untuk membuat tiga buah list yang berisi teks tanggal awal (start\_date), misalnya '2019-01-01', tanggal akhir (end\_date), misalnya '2019-12-31', dan tahun (years), misalnya '2019', untuk setiap tahun. start\_date dan end\_date nantinya akan digunakan untuk memfilter Citra Sentinel-2 setiap tahun. Sementara years akan digunakan untuk label dinamik citra dalam animasi.

```
# Masking awan  
def mask_s2_clouds(image):  
    qa = image.select('QA60')  
    cloud_bit_mask = 1 << 10  
    cirrus_bit_mask = 1 << 11  
  
    mask = (  
        qa.bitwiseAnd(cloud_bit_mask)  
        .eq(0)  
        .And(qa.bitwiseAnd(cirrus_bit_mask).eq(0))  
    )  
  
    return image.updateMask(mask)
```

```
# Mengakses Sentinel-2 multitemporal  
# dan membuat list Citra Sentinel-2 untuk setiap tahun  
  
s2_col = ee.ImageCollection('COPERNICUS/S2_HARMONIZED')  
s2_image_list = []  
  
for i in range(num_years):  
    dataset = (  
        s2_col.filterDate(start_date[i], end_date[i])  
        .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE', 20))  
        .map(mask_s2_clouds)  
    )  
  
    s2_image_list.append(dataset.median().clip(envelope))
```

Perhatikan kode-kode di atas, Dimana kita mengakses image collection Sentinel-2 dengan menggunakan iterasi **for** menurut jumlah tahun yang kita tentukan (num\_years = 10). Hal ini bertujuan untuk mengakses image collection Sentinel-2 per tahun. perhatikan instruksi filterDate(start\_date[i], end\_date[i]). Selanjutnya image collection per tahun ini kemudian direduksi menjadi image dan dipotong mengikuti batas wilayah administrasi (dataset.median().clip(envelope)). Median image Sentinel-2 per tahun ini kemudian disimpan ke dalam list s2\_image\_list. Selanjutnya, kita dapat memvisualisasikan Citra Sentinel-2 per tahun dengan kode-kode berikut:

```
# Visualisasi Citra Sentinel-2 untuk setiap tahun

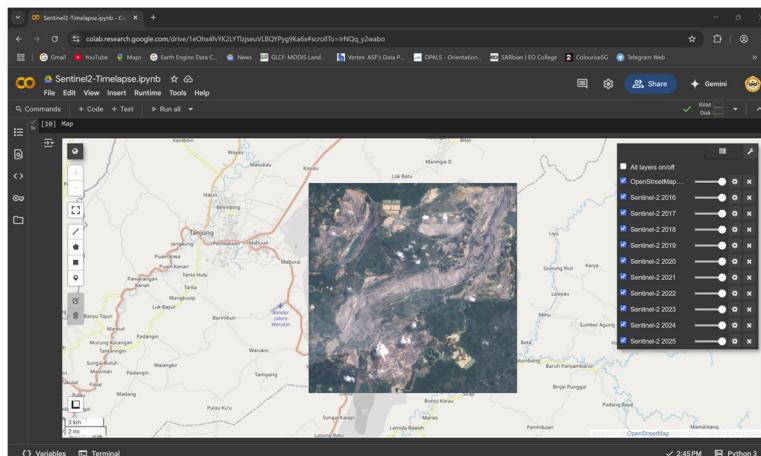
vis_params = {
    'bands': ['B4', 'B3', 'B2'],
    'min': 0,
    'max': 2500
}

Map = geemap.Map()

for i, image in enumerate(s2_image_list):
    Map.addLayer(image, vis_params, f'Sentinel-2 {years[i]}')

Map.centerobject(point, 12)
Map
```

Output:



Perhatikan iterasi `for i, image in enumerate(s2_image_list)` pada kode-kode di atas, yang digunakan untuk menampilkan citra (`Map.addLayer`) sebanyak citra yang tersimpan di dalam list `s2_image_list`. Langkah berikutnya adalah konversi list `s2_image_list` menjadi image collection `sentinel2_col`. Sebagaimana sudah dijelaskan pada halaman 293, bahwa data yang akan dijadikan input di dalam pembuatan timelapse animation adalah image collection.

```
# Membuat koleksi citra multitemporal
sentinel2_col = ee.ImageCollection.fromImages(s2_image_list)
```

Selanjutnya, kita juga perlu untuk membuat sebuah list label tahun dalam format seperti ini, **Tahun 2019**, agar animasi lebih informatif.

```
# Membuat label tahun untuk animasi
labels = [f'Tahun {year}' for year in years]
print(labels)
```

Output:

```
['Tahun 2016', 'Tahun 2017', 'Tahun 2018', 'Tahun 2019', 'Tahun 2020', 'Tahun 2021', 'Tahun 2022', 'Tahun 2023', 'Tahun 2024', 'Tahun 2025']
```

### Bab III Google Earth Engine

Langkah berikutnya, kita akan menentukan beberapa parameter animasi. Seperti dimensi animasi, batas wilayah, *Frame Per Second* (FPS), sistem koordinat, batasan nilai-nilai pixel, output nama dan lokasi penyimpanan file animasi gif (path + 'output/S2\_Timelapse.gif'), dan sebagainya. Silahkan lihat kembali penjelasan tentang parameter-parameter animasi ini pada halaman 293 dan 294 sebelumnya.

```
# Menentukan beberapa parameter animasi
# dan menyimpan animasi gif ke Google Drive

video_args = {
    'dimensions': 768,
    'region': envelope,
    'framesPerSecond': 1,
    'crs': 'EPSG:4326',
    'min': 0,
    'max': 2500,
    'bands': ['B4', 'B3', 'B2']
}

s2_saved_gif = path + 'output/s2_Timelapse.gif'
geemap.download_ee_video(sentinel2_col, video_args, s2_saved_gif)
```

Hal yang berbeda pada parameter-parameter di atas, dibandingkan dengan animasi mangrove sebelumnya pada halaman 293, adalah 'bands': ['B4', 'B3', 'B2']. Hal ini karena kita akan membuat animasi Citra Sentinel-2 komposit RGB. Bukan citra tunggal seperti MVI sebelumnya yang menggunakan 'palette'.

```
# Menambahkan judul pada animasi

title = f'Dinamika Tambang Adaro dari citra sentinel-2 ({years[0]} - {years[-1]})'

geemap.add_text_to_gif(
    s2_saved_gif,
    s2_saved_gif,
    xy=('10%', '93%'),
    text_sequence=title,
    font_size=24,
    font_color="#ffffff",
    add_progress_bar=False,
)
```

```
# Menambahkan logo pada animasi

logo = path + 'logo/logo_uIm.png'

geemap.add_image_to_gif(
    s2_saved_gif,
    s2_saved_gif,
    in_image=logo,
    xy=('12%', '5%'),
    image_size=(85, 85)
)
```

Hal baru pada timelapse animation pada kode-kode di atas, yang belum pernah dibahas pada bagian terdahulu adalah penambahan logo animasi. Kita dapat menambahkan logo atau gambar, dalam format PNG misalnya, ke dalam animasi kita. Parameter `image_size=(85, 85)` merupakan ukuran logo yang ditambahkan ke animasi dalam satuan pixel. Parameter-parameter lainnya, seperti posisi gambar, dan sebagainya, aturannya sama persis dengan penambahan teks pada animasi. Berikutnya, kita dapat menambahkan nama institusi, atau nama kita sendiri, dan teks dinamik tahun ke dalam animasi.

```
# Menambahkan nama institusi pada animasi
geemap.add_text_to_gif(
    s2_saved_gif,
    s2_saved_gif,
    xy=('5%', '20%'),
    text_sequence='PPIIG ULM 2025',
    font_size=24,
    font_color='#ffffff',
    add_progress_bar=False,
)
```

```
# Menambahkan teks dinamik (label tahun) pada animasi
geemap.add_text_to_gif(
    s2_saved_gif,
    s2_saved_gif,
    xy=('70%', '85%'),
    text_sequence=labels,
    font_size=36,
    font_color='#ffffcc',
    add_progress_bar=False,
    duration=1000*len(years),
)
# Menampilkan animasi
geemap.show_image(s2_saved_gif)
```

Teks dinamik yang kita tambahkan ke dalam animasi adalah label tahun (`labels`) yang sudah kita buat pada kode-kode sebelumnya. `duration=1000*len(years)` berarti kita akan menampilkan animasi satu tahun per detik, silahkan lihat kembali penjelasan di halaman 294. Berikut adalah output timelapse animation-nya:



## Membuat Layout

Terdapat beberapa paket Python maupun GEE yang dapat diimplementasikan untuk membuat layout di lingkungan Google Colab/GEE. Pada latihan ini kita akan menggunakan Cartopy (Met Office, 2010 – 2013) dan Cartoee (Markert, 2019). Jika Anda menggunakannya secara lokal, misalnya dengan JupyterLab, VS Code, atau yang sejenisnya, maka Anda harus menginstal kedua paket ini terlebih dahulu. Termasuk paket-paket pendukung, yaitu Mapclassify (Slocum et al., 2009; Jiang, 2012; Rey et al., 2016), Matplotlib-Scalebar, Contextily, dan Plotly. Sebagai studi kasus, kita akan membuat layout Presipitasi Rerata Tahunan Kabupaten Tanah Laut (1994–2023). Dimana data presipitasi atau curah hujannya sendiri akan diambil dari dataset TerraClimate (<https://www.climatologylab.org/terraclimat.html>) (Abatzoglou et al., 2018). Silahkan buat sebuah notebook Google Colab baru, beri nama misalnya **Map-Layout.ipynb**. Kemudian ketikkan dan jalankan kode-kode berikut ini:

```
# Instalasi paket-paket yang diperlukan
!pip install cartopy matplotlib-scalebar contextily mapclassify -U plotly

import ee, geemap

ee.Authenticate()

ee.Initialize(project='ee-geospatialulm')

# Mengakses Google Drive
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

# Membuka shapefile wilayah observasi
import geopandas as gpd
shapefile = '/content/drive/My Drive/geebook/vector/Tala_Polygon.shp'
region_shp = gpd.read_file(shapefile)
region_shp.plot()

region = geemap.geopandas_to_ee(region_shp).geometry()
```

Data TerraClimate memiliki resolusi spasial 4.638,3 meter, untuk lebih jelasnya baca di tautan [https://developers.google.com/earth-engine/datasets/catalog/IDAHO\\_EPSCOR\\_TERRACLIMATE](https://developers.google.com/earth-engine/datasets/catalog/IDAHO_EPSCOR_TERRACLIMATE). Untuk pemetaan pada wilayah yang tidak terlalu luas, misalnya satu kabupaten, resolusi spasial seperti ini terlalu kasar. Sehingga kita harus memperhalus atau meningkatkan resolusi spasialnya (*downscaling*), misalnya menjadi 100 meter, atau lebih tinggi. Kode-kode berikut merupakan sebuah fungsi untuk melakukan reproyeksi, sekaligus meningkatkan resolusi spasial data TerraClimate dengan metode resampling.

```
# Reproyeksi image dan resampling resolusi spasial
def reproject_image(image):
    original_image = image.setDefaultProjection('EPSG:4326', None, 4638.3)
    reprojected_image = (
        original_image.resample('bicubic')
        .reproject(crs='EPSG:4326', scale=100).clip(region)
    )
    return reprojected_image
```

Perhatikan fungsi di atas, sebelum resolusi spasial sebuah GEE image atau GEE image collection diresampling, terlebih dahulu harus diatur proyeksi bawaannya, perhatikan instruksi `image.setDefaultProjection('EPSG:4326', None, 4638.3)`. Untuk selanjutnya, resampling dilakukan dengan metode `bicubic` (`resample('bicubic')`) menjadi 100 meter (`scale=100`). Selain `bicubic`, metode resampling lainnya yang dapat diterapkan adalah `bilinear`. Selanjutnya, kita akan mengakses image collection TerraClimate selama 30 tahun (1994-2023).

```
# Mengakses data TerraClimate image collection 30 tahun
tc_col = (
    ee.ImageCollection('IDAHO_EPSCOR/TERRACLIMATE')
        .filter(ee.Filter.date('1994-01-01', '2024-01-01'))
)
```

Data TerraClimate yang kita gunakan merupakan dataset iklim rerata bulanan, sehingga instruksi `ee.Filter.date('1994-01-01', '2024-01-01')` pada kode-kode di atas berarti mengakses data TerraClimate dari Januari 1994 hingga Desember 2023. Ingat kembali konsep inklusif dan eksklusif pada elemen list. Berikutnya, kita akan memilih parameter presipitasi dari dataset TerraClimate, sekaligus menerapkan reprojeksi dan resampling pada data presipitasi.

```
# Seleksi dan reprojeksi data presipitasi dari TerraClimate image collection
precipitation = tc_col.select('pr').map(reproject_image)
```

Pada kode-kode di atas, `select('pr')` merupakan seleksi terhadap data presipitasi di dalam dataset TerraClimate. Untuk lebih jelasnya lihat di laman [https://developers.google.com/earth-engine/datasets/catalog/IDAHO\\_EPSCOR\\_TERRACLIMATE#bands](https://developers.google.com/earth-engine/datasets/catalog/IDAHO_EPSCOR_TERRACLIMATE#bands).

```
# Kalkulasi presipitasi rerata tahunan selama 30 tahun
precipitation_30_years = precipitation.sum().divide(30)
```

Kode-kode di atas merupakan kalkulasi rerata presipitasi tahunan selama 30 tahun. Teknisnya adalah seluruh data presipitasi 1994-2023 dijumlahkan (`precipitation.sum()`), kemudian hasil penjumlahannya dibagi 30 tahun (`divide(30)`). Selanjutnya, untuk keperluan visualisasi, kita perlu untuk menghitung statistik presipitasi rerata tahunan, terutama `min` dan `max`-nya.

```
# Menghitung statistik presipitasi rerata untuk keperluan visualisasi
geemap.image_stats(precipitation_30_years.clip(region), scale=100)
```

```
▼ Object (5 properties)
  ▼ max: Object (1 property)
    pr: 2626.033333333333
  ▶ mean: Object (1 property)
  ▼ min: Object (1 property)
    pr: 2290.6
  ▶ std: Object (1 property)
  ▶ sum: Object (1 property)
```

Berikutnya kita dapat memvisualisasikan presipitasi rerata tahunan:

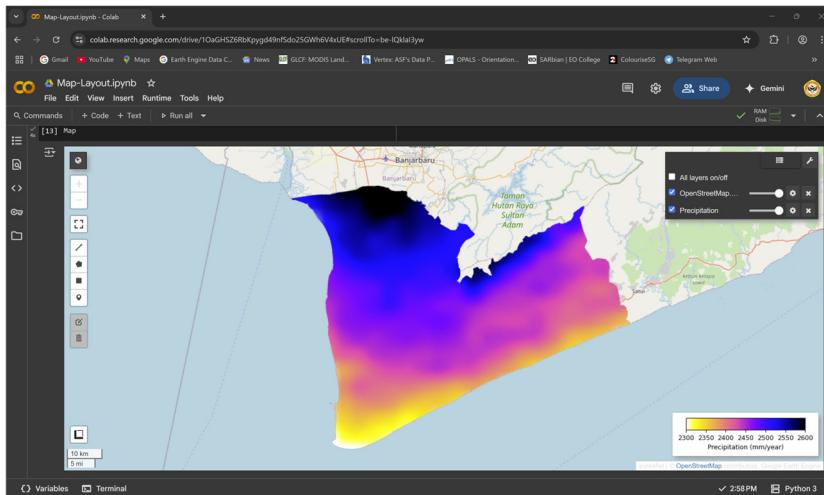
### Bab III Google Earth Engine

```
# Visualisasi presipitasi rerata tahunan

visualization = {
  'min': 2300,
  'max': 2600,
  'palette': 'gnuplot2_r'
}

Map = geemap.Map()
Map.centerObject(region, 10)
Map.addLayer(precipitation_30_years, visualization, 'Precipitation')
Map.add_colorbar(visualization, label='Precipitation (mm/year)',
  layer_name='Precipitation')
Map
```

Berikut adalah outputnya:



Untuk membuat layout menggunakan Cartoee, kita harus mengkonversi Matplotlib colormaps (misalnya `gnuplot2_r`) menjadi kode-kode warna heksadesimal.

```
# Konversi Matplotlib colormaps menjadi kode warna heksadesimal

import matplotlib as mpl
import matplotlib.pyplot as plt

cmap = plt.get_cmap('gnuplot2_r')

color_list = []
color_num = 30

for i in range(color_num):
    color = cmap(i/color_num)
    hex_color = mpl.colors.to_hex(color)
    color_list.append(hex_color)

print(color_list)
```

Output:

```
['#ffffff', '#fffff9b', '#fffff2a', '#fff609', '#ffe41b', '#ffd42b', '#ffc23d',
 '#ffb24d', '#ffa05f', '#ff906f', '#ff7e81', '#ff6e91', '#ff5ca3', '#fe4cb3',
 '#e23ac5', '#c628d7', '#ad18e7', '#9106f9', '#7800ff', '#5b00ff', '#4200ff',
 '#2600ff', '#0d00ff', '#0000ec', '#0000cc', '#0000a8', '#000088', '#000064',
 '#000044', '#000020']
```

Perhatikan kode-kode di atas, pertama kita harus mengambil skema warna kontinyu Matplotlib colormaps, yaitu `cmap = plt.get_cmap('gnuplot2_r')`. Kemudian kita tentukan `gnuplot2_r` akan diekstrak menjadi berapa kode warna diskret heksadesimal, misalnya menjadi 30 warna (`color_num = 30`). 30 kode warna tersebut kemudian disimpan ke dalam list `color_list`. Langkah selanjutnya, kita akan menentukan parameter visualisasi presipitasi, dengan menggunakan kode-kode warna heksadesimal yang sudah kita buat sebelumnya.

```
# Penentuan parameter visualisasi presipitasi untuk layout
vis_params = {
    'min': 2300,
    'max': 2600,
    'palette': color_list
}
precipitation_viz = precipitation_30_years.visualize(**vis_params)
```

Kita juga diharuskan untuk menentukan batas-batas wilayah yang akan dilayout. Dalam hal ini, kita dapat mengambil batas terluar wilayah yang dipetakan, yaitu `region`.

```
# Menentukan koordinat batas-batas layout
coordinates = region.bounds().getInfo()['coordinates'][0]
bbox = [coordinates[1][0], coordinates[0][1], coordinates[0][0], coordinates[2][1]]
```

Pada kode-kode di atas, `region.bounds().getInfo()['coordinates'][0]` digunakan untuk mengekstrak koordinat batas-batas `region`. Akan tetapi, urutan koordinat-koordinat batas wilayahnya di dalam list tidak sesuai dengan sistem Cartoee. Sehingga harus dirubah order atau urutannya dengan instruksi `bbox = ....` Perhatikan dengan cermat penempatan indeks-indeks list-nya. Anda dapat menambahkan instruksi `print(bbox)` pada bagian akhir kode-kode di atas, untuk menampilkan koordinat batas-batas wilayah `bbox`. Langkah berikutnya, kita akan membuat layout presipitasi rerata tahunan, dengan menggunakan parameter-parameter yang sudah kita atur sebelumnya.

```
# Membuat layout presipitasi rerata tahunan
# menggunakan Cartopy, Cartoee, dan Matplotlib
import cartopy.crs as ccrs
from geemap import cartoee
from matplotlib_scalebar.scalebar import ScaleBar

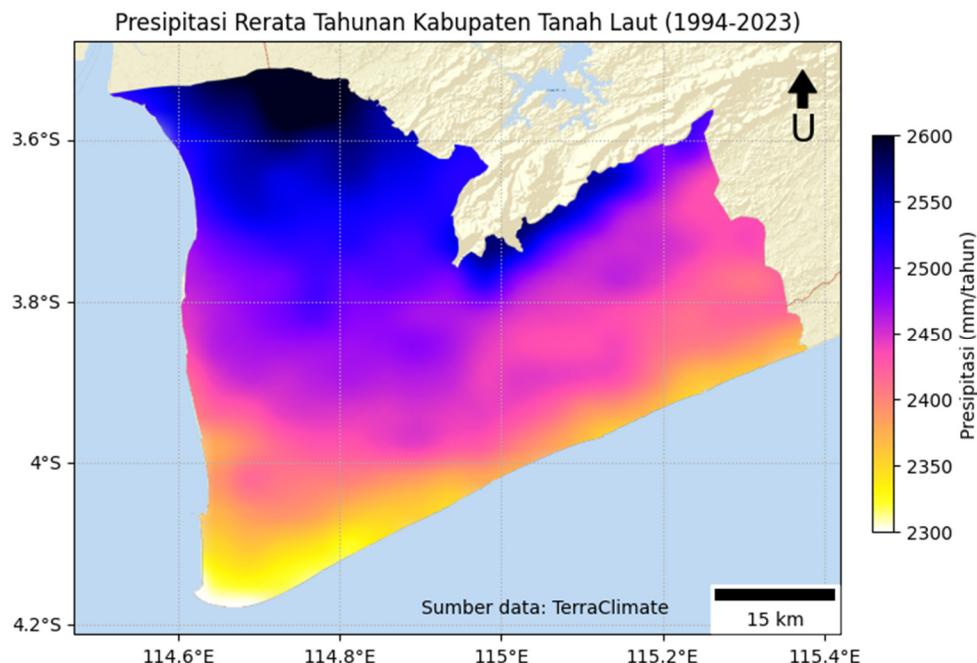
scalebar = ScaleBar(dx=100, units='km', location='lower right',
length_fraction=0.2, height_fraction=0.02)

fig = plt.figure(figsize=(8, 6))

ax = cartoee.get_map(precipitation_viz, basemap='Roadmap', zoom_level=12,
region=bbox)
cartoee.pad_view(ax)
ax.set_title(label='Presipitasi Rerata Tahunan Kabupaten Tanah Laut (1994-2024)', fontsize=12)
cb = cartoee.add_colorbar(ax, vis_params=vis_params, loc='right',
label='Presipitasi (mm/tahun)')
cartoee.add_gridlines(ax, interval=0.2, xtick_rotation=0, linestyle=':')
ax.add_artist(scalebar)
x, y, arrow_length = 0.95, 0.95, 0.1
ax.annotate('U', xy=(x, y), xytext=(x, y-arrow_length),
arrowprops=dict(facecolor='black', width=5, headwidth=15),
ha='center', va='center', fontsize=20, xycoords=ax.transAxes)
ax.annotate('Sumber data: TerraClimate', xy=(0.4, 0.08),
xycoords='figure fraction', ha='left', va='bottom', fontsize=10)
plt.show()
```

### Bab III Google Earth Engine

Berikut adalah output layoutnya:



Pada kode-kode di atas, `scalebar` merupakan pengaturan scalebar pada layout. Untuk lebih jelasnya, silahkan baca di laman <https://github.com/ppinard/matplotlib-scalebar>. Instruksi `cartoee.add_gridlines(ax, interval=0.2, ...)` adalah penambahan garis-garis grid pada layout, hapus baris ini jika Anda tidak memerlukan grid. `x, y, arrow_length = 0.95, 0.95, 0.1` merupakan penambahan tanda panah untuk menunjukkan orientasi/arah Utara, dimana `0.95` merupakan posisi koordinat `x` dan `y` dari tanda panah dalam satuan rasio/persen terhadap area layout, dan `0.1` adalah panjang panah. `ax.annotate('U', xy=(x, y) ...)` adalah penambahan karakter `U` di bawah tanda panah.

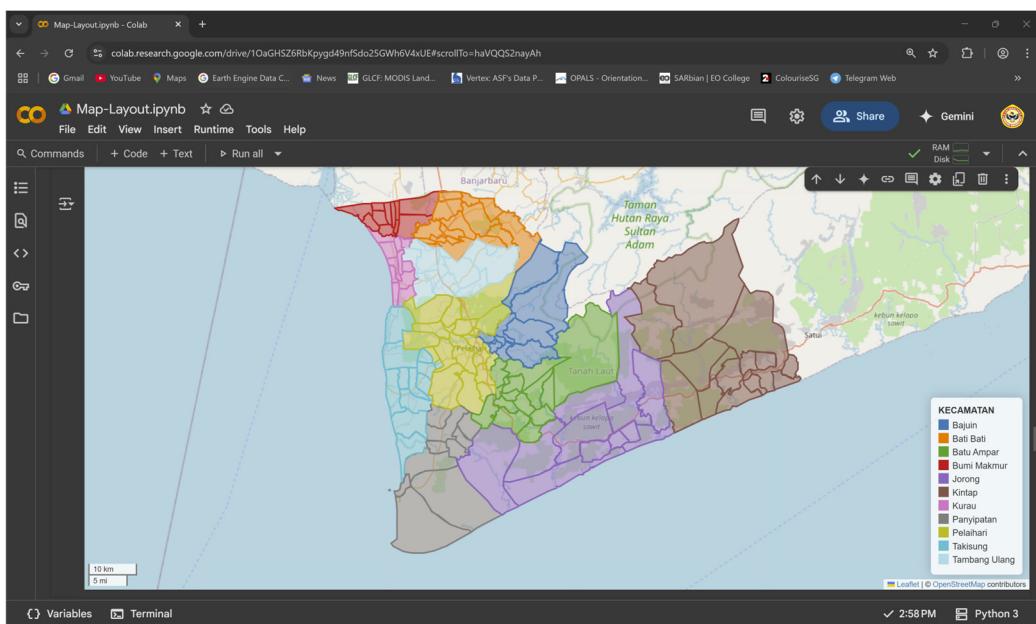
Berikutnya, kita akan menganalisis dan memvisualisasikan presipitasi rerata tahunan per satuan wilayah administrasi, dalam hal ini adalah per desa/kelurahan. Terlebih dahulu kita harus membuka shapefile wilayah administrasi yang diperlukan.

```
# Membuka shapefile wilayah administrasi
admin_tala = '/content/drive/My Drive/geebook/vector/Tanah_Laut.shp'
admin_shp = gpd.read_file(admin_tala)
```

Pada contoh kasus di atas, shapefilenya memiliki sistem koordinat UTM Zone 50S WGS 1984 (EPSG: 32750). Sehingga perlu untuk dikonversi menjadi sistem koordinat geografis WGS 1984 (EPSG: 4326) terlebih dahulu.

```
# Konversi sistem koordinat wilayah administrasi ke sistem Koordinat Geografis
# dan visualisasi wilayah administrasi
admin_shp_geo = admin_shp.to_crs(epsg=4326)
admin_shp_geo.explore(column='KECAMATAN', popup=True)
```

Berikut adalah output visualisasi wilayah administrasinya:



Pada kode-kode di atas, kita memvisualisasikan wilayah administrasi desa/kelurahan dengan menggunakan kolom **KECAMATAN** yang ada di dalam shapefile sebagai dasar warna unik. Visualisasi `explore(column='KECAMATAN', popup=True)` ini memerlukan instalasi paket Mapclassify, sebagaimana sudah dijelaskan sebelumnya. Selanjutnya, kita harus merubah shapefile (yang sudah dalam sistem koordinat geografis) menjadi GEE feature collection.

```
# Konversi shapefile menjadi feature collection
admin_geo_fc = geemap.geopandas_to_ee(admin_shp_geo)
```

Selanjutnya, kita akan melakukan kalkulasi *zonal statistic*, sebagaimana yang bias kita lakukan dengan menggunakan perangkat lunak SIG, untuk membuat data presipitasi rerata per wilayah administrasi. Dalam hal ini, tentu saja per desa/kelurahan.

```
# Kalkulasi zonal statistic untuk menghitung presipitasi rerata
# per wilayah administrasi, dan menyimpannya hasil kalkulasi ke dalam file CSV
out_data_csv = '/content/drive/My Drive/geebook/output/Presipitasi_Tala.csv'
geemap.zonal_stats(
    precipitation_30_years, admin_geo_fc, out_data_csv,
    statistics_type='MEAN', scale=100, return_fc=False
)
```

Pada kode-kode di atas, hasil analisis *zonal statistic* akan disimpan ke dalam Google Drive dalam format CSV (lihat isi variable `out_data_csv`). Data yang menjadi input di dalam analisis *zonal statistic* adalah GEE image presipitasi rerata tahunan (`precipitation_30_years`) dan feature collection wilayah administrasi (`admin_geo_fc`). Selanjutnya, kita buka kembali tabel CSV hasil analisis *zonal statistic* dengan menggunakan Pandas. Kemudian data CSV ini kita konversi menjadi GeoDataFrame dengan menggunakan GeoPandas.

### Bab III Google Earth Engine

```
# Mengimpor file CSV presipitasi rerata per wilayah administrasi
import pandas as pd
presipitasi_df = pd.read_csv(out_data_csv)
presipitasi_gdf = gpd.GeoDataFrame(presipitasi_df,
                                    geometry=admin_shp_geo.geometry)
presipitasi_gdf
```

Data CSV yang berisi presipitasi rerata per desa/kelurahan tidak memiliki informasi geometri yang diwajibkan untuk konversi ke dalam format GeoDataFrame, sebagaimana terlihat pada gambar di bawah. Sehingga pada kode-kode di atas, data geometri harus diambil dari shapefile wilayah administrasi (lihat `geometry=admin_shp_geo.geometry`). Di dalam file CSV, data presipitasi rerata per desa/kelurahan tersimpan di kolom **mean**. Perhatikan gambar berikut:

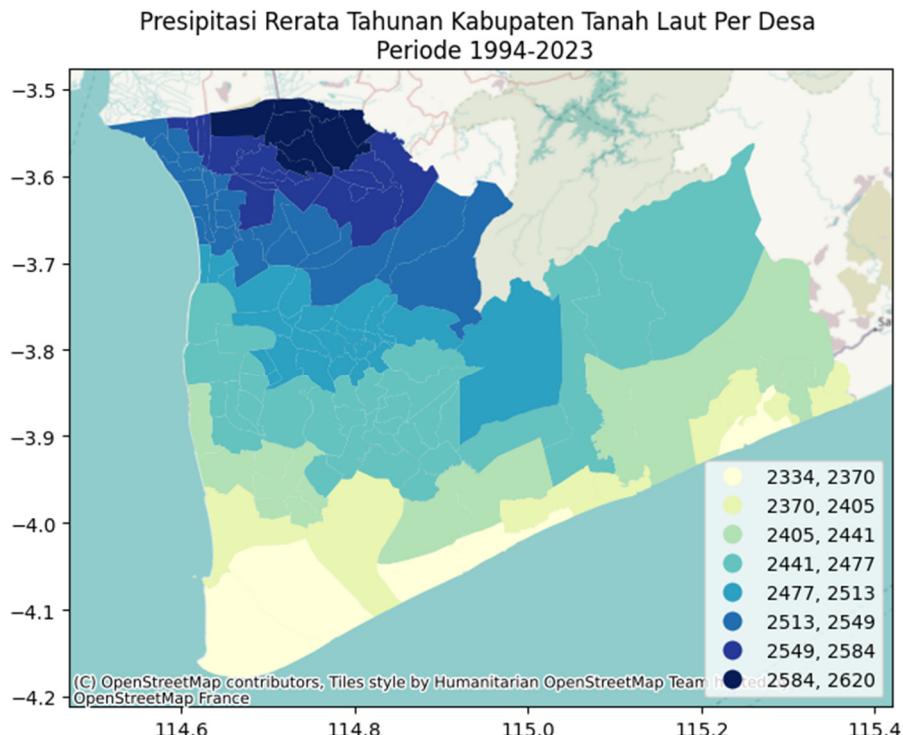
	A	B	C	D	E	F	G	H
1	mean	system:index	DESA	KECAMATAN	KABUPATEN	PENDUDUK	PROVINSI	LUAS_KM2
2	2413.03153	0	Alur	Jorong	Tanah Laut	2372	Kalimantan Selatan	78.8047
3	2451.379712	1	Ambawang	Batu Ampar	Tanah Laut	1978	Kalimantan Selatan	14.5934
4	2522.253503	2	Ambungan	Pelaihari	Tanah Laut	2154	Kalimantan Selatan	53.2773
5	2503.373934	3	Angsau	Pelaihari	Tanah Laut	13024	Kalimantan Selatan	9.28098
6	2461.192726	4	Asam-Asam	Jorong	Tanah Laut	5753	Kalimantan Selatan	125.399
7	2399.21182	5	Asam Jaya	Jorong	Tanah Laut	2006	Kalimantan Selatan	9.81268
8	2398.460407	6	Asri Mulya	Jorong	Tanah Laut	1601	Kalimantan Selatan	11.1503
9	2494.116583	7	Atu-Atu	Pelaihari	Tanah Laut	3446	Kalimantan Selatan	2.4099
10	2498.881606	8	Bajuin	Bajuin	Tanah Laut	2106	Kalimantan Selatan	11.0435
11	2464.535513	9	Banua Lawas	Taksung	Tanah Laut	2188	Kalimantan Selatan	10.6542
12	2461.573742	10	Banua Tengah	Taksung	Tanah Laut	4042	Kalimantan Selatan	11.8934
13	2601.685087	11	Banyu Irang	Bati Bati	Tanah Laut	3076	Kalimantan Selatan	12.1228
14	2333.707737	12	Batakan	Panyipatan	Tanah Laut	4948	Kalimantan Selatan	117.006
15	2444.125449	13	Batalang	Jorong	Tanah Laut	1006	Kalimantan Selatan	60.1602
16	2563.312988	14	Bati-Bati	Bati Bati	Tanah Laut	5501	Kalimantan Selatan	21.7434

Selanjutnya, kita akan membuat *choropleth*, yaitu peta tematik kuantitatif dengan gradasi warna, dengan menggunakan data `presipitasi_gdf`. Dimana `presipitasi_gdf` merupakan hasil konversi CSV presipitasi rerata per desa/kelurahan sebelumnya. Tentu saja, kolom yang akan menjadi dasar gradasi warna adalah **mean**, seperti terlihat pada gambar di atas.

```
# Membuat layout choropleth presipitasi rerata per wilayah administrasi
import contextily as cx
layout = presipitasi_gdf.plot(figsize=(8, 6), column='mean', legend=True,
                               legend_kwds={'loc': 'lower right',
                                            'fmt': '{:.0f}', k=8, cmap='YlGnBu',
                                            'scheme='equalinterval'})
layout.set_title('Presipitasi Rerata Tahunan Kabupaten Tanah Laut Per Desa \n
Periode 1994-2024')
cx.add_basemap(layout, crs=presipitasi_gdf.crs)
```

Pada kode-kode di atas, `column='mean'` merupakan pengambilan kolom **mean** sebagai dasar gradasi warna, `'loc': 'lower right'` adalah lokasi legenda, `'fmt': '{:.0f}'` adalah pengaturan data numerik presipitasi pada legenda, yaitu nol angka di belakang koma, dan `scheme='equalinterval'` merupakan metode klasifikasi data presipitasi.

Berikut adalah output choropleth-nya:



Terkait dengan teori-teori dan teknik-teknik layout peta tematik, *choropleth*, skema warna (*equal interval*, *geometric interval*, *natural breaks*, dan sebagainya), Anda direkomendasikan untuk membaca buku *Cartography: Thematic Map Design (6<sup>th</sup> edition)* (Dent et al., 2008). Tentu saja, untuk menghasilkan sebuah layout dengan kualitas kartografi, diperlukan pengembangan kode-kode Python yang lebih kompleks, tidak sesederhana contoh-contoh di atas.

## Web Map Interaktif

Selain layout statik, kita juga dapat memvisualisasikan data presipitasi rerata tahunan per desa/kelurahan yang kita buat sebelumnya, ke dalam bentuk web map interaktif atau file *HyperText Markup Language* (HTML). Sehingga nanti dapat diunggah ke laman web kita. Terdapat banyak paket Python atau GEE yang dapat diterapkan untuk membuat web map interaktif, pada tutorial ini, kita akan menggunakan Plotly (Plotly Technologies Inc., 2015). Masih melanjutkan kode-kode sebelumnya, silahkan ketikkan dan jalankan kode-kode berikut:

```
# Mengambil data koordinat titik pusat wilayah observasi
region_center = region.centroid().getInfo()['coordinates']
lon, lat = region_center[0], region_center[1]
```

```
# Persiapan data geospasial untuk layout web map interaktif
presipitasi_gdf['wilayah'] = 'Kec. ' + presipitasi_gdf['KECAMATAN'] + \
', Desa ' + presipitasi_gdf['DESA']
presipitasi_gdf['mean'] = presipitasi_gdf['mean'].round(0)
presipitasi_gdf.set_index('wilayah', inplace=True)
```

### Bab III Google Earth Engine

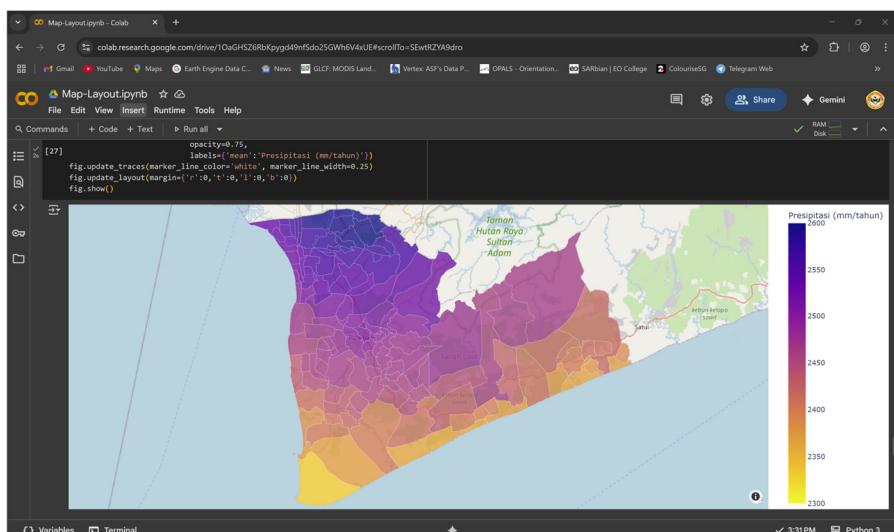
Kode-kode di atas merupakan persiapan pembuatan web map interaktif. Langkah-langkahnya meliputi pengambilan informasi koordinat titik pusat wilayah yang dipetakan, yang nantinya akan menjadi titik fokus laman web. Penambahan kolom baru, yaitu `wilayah`, dan kolom `wilayah` ini akan diisi dengan informasi gabungan kolom `KECAMATAN` dan `Desa` dengan instruksi `presipitasi_gdf['wilayah']='Kec. '+presipitasi_gdf['KECAMATAN']+', Desa '+ presipitasi_gdf['DESA']`. Terkait instruksi ini, lihat kembali pembahasan tentang kalkulasi di dalam dataframe pada halaman 121. Kolom `wilayah` nantinya akan ditampilkan sebagai informasi *pop-up* bersama dengan informasi presipitasi rerata per desa.

Instruksi `presipitasi_gdf['mean'] = presipitasi_gdf['mean'].round()` digunakan untuk membulatkan kolom `mean` yang berisi data presipitasi rerata per desa menjadi nol angka di belakang koma, agar informasi *pop-up* nantinya tidak terlalu rumit. `presipitasi_gdf.set_index('wilayah', inplace=True)` diterapkan agar kolom `wilayah` yang berisi wilayah administrasi menggantikan indeks default pada `presipitasi_gdf`, yang berupa angka berurutan dimulai dari 0. Lihat kembali pembahasan tentang hal ini pada halaman 118. Penggantian indeks default dengan `wilayah` ini bertujuan agar tampilan informasi *pop-up* nantinya adalah informasi `wilayah`, bukan angka indeks. Langkah berikutnya, kita akan membuat layout web map choropleth menggunakan Plotly:

```
# Membuat layout web map choropleth interaktif menggunakan Plotly
import plotly.express as px

fig = px.choropleth_mapbox(presipitasi_gdf,
                           geojson=presipitasi_gdf.geometry.__geo_interface__,
                           locations=presipitasi_gdf.index,
                           color='mean',
                           color_continuous_scale='plasma_r',
                           range_color=(2300, 2600),
                           mapbox_style='open-street-map',
                           zoom=9,
                           center = {'lat': lat, 'lon': lon},
                           opacity=0.75,
                           labels={'mean':'Presipitasi (mm/tahun)'})
fig.update_traces(marker_line_color='white', marker_line_width=0.25)
fig.update_layout(margin={'r':0, 't':0, 'l':0, 'b':0})
fig.show()
```

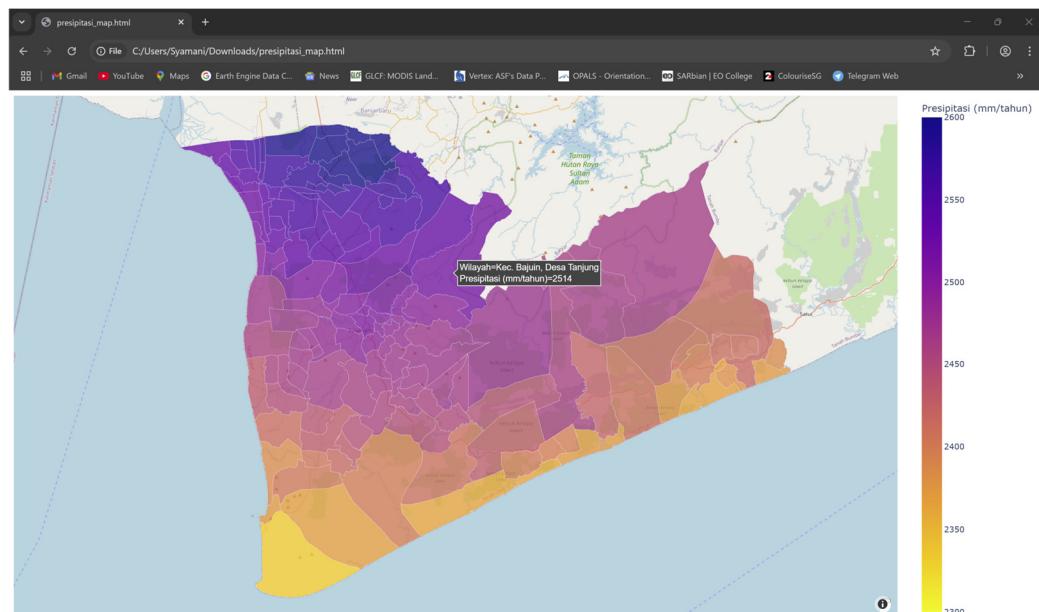
Berikut adalah outputnya:



Penjelasan tentang `choropleth_mapbox` pada kode-kode di atas dapat dibaca di laman [https://plotly.github.io/plotly.py-docs/generated/plotly.express.choropleth\\_mapbox.html](https://plotly.github.io/plotly.py-docs/generated/plotly.express.choropleth_mapbox.html). Instruksi `update_traces(marker_line_color='white', marker_line_width=0.25)` digunakan untuk merubah warna dan lebar garis (*outline*) wilayah administrasi. Tanpa instruksi ini, biasanya garis-garis batas wilayah administrasi akan berwarna hitam. `update_layout(margin={'r':0,'t':0,'l':0,'b':0})` adalah margin atau tepi-tepi layout (`r`: right, `t`: top, `l`: left, `b`: bottom). Langkah selanjutnya, kita akan mengekspor layout yang sudah kita buat ke dalam laman HTML:

```
# Export layout choropleth ke dalam format HTML
html_output = '/content/drive/My Drive/geebook/output/presipitasi_map.html'
fig.write_html(html_output, include_plotlyjs='cdn')
```

Berikut adalah outputnya ketika dibuka menggunakan web browser:



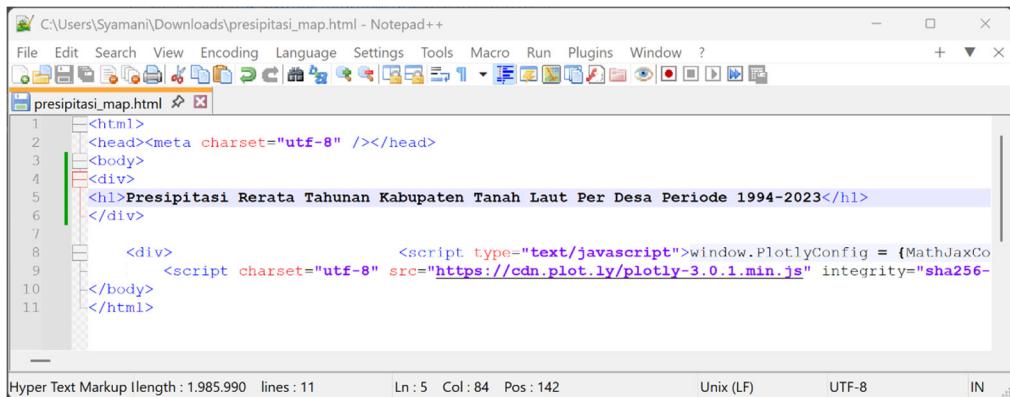
Pada kode-kode di atas, jika `include_plotlyjs='cdn'` maka file HTML yang dihasilkan lebih kecil. Akan tetapi, file HTML memerlukan koneksi internet aktif untuk memuat pustaka plotly.js. Penjelasan tentang plotly.js dapat dibaca di laman <https://plotly.com/javascript/>. Jika Anda ingin file HTML menyimpan langsung plotly.js di dalamnya, sehingga file HTML dapat dijalankan tanpa koneksi internet, atur `include_plotlyjs=True`. Konsekuensinya, kapasitas file HTML yang dihasilkan akan lebih besar.

Anda dapat menambahkan teks-teks atau bahkan objek-objek lain ke dalam file HTML web map. Dalam hal ini, diperlukan sedikit pengetahuan tentang HTML. Sebagai contoh, kita akan menambahkan teks berikut ke dalam laman web map interaktif kita:

```
<div>
<h1>Presipitasi Rerata Tahunan Kabupaten Tanah Laut Per Desa Periode 1994-2023</h1>
</div>
```

### Bab III Google Earth Engine

Silahkan gunakan teks editor seperti Notepad++ atau yang lainnya untuk mengedit file HTML, sebagaimana gambar berikut:

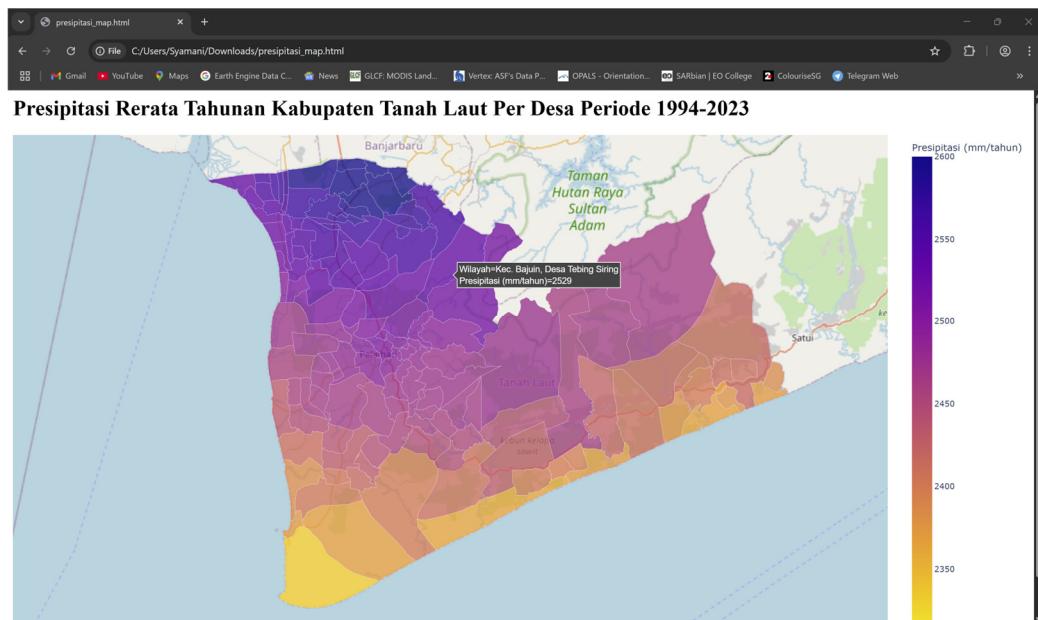


The screenshot shows a Notepad++ window with the file 'presipitasi\_map.html'. The code is as follows:

```
<html>
<head><meta charset="utf-8" /></head>
<body>
<div>
<h1>Presipitasi Rerata Tahunan Kabupaten Tanah Laut Per Desa Periode 1994-2023</h1>
</div>
<div>
<script type="text/javascript">window.PlotlyConfig = {MathJaxConfig: "local", scrollZoom: true};</script>
<script charset="utf-8" src="https://cdn.plot.ly/plotly-3.0.1.min.js" integrity="sha256-...</script>
</div>
</body>
</html>
```

Below the code, status bar information includes: Hyper Text Markup llength : 1.985.990 lines : 11 Ln : 5 Col : 84 Pos : 142 Unix (LF) UTF-8 IN.

Simpan kembali file HTML setelah menambahkan kode-kode di atas. Kemudian coba buka kembali file HTML web map-nya. Berikut adalah outputnya:



Arahkan panah mouse ke atas salah satu wilayah administrasi, maka Anda akan mendapatkan informasi *pop-up* yang berisi wilayah dan presipitasi. Sebagaimana terlihat pada gambar di atas.

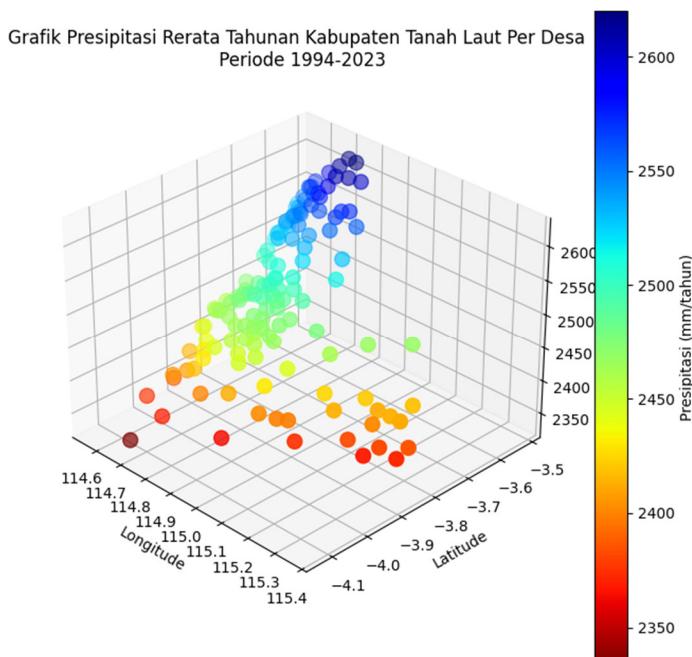
### Grafik 3 Dimensi dan Animasi Grafik

Grafik-grafik statik 2-Dimensi sudah banyak kita sajikan sebelumnya dengan menggunakan Matplotlib. Sekarang kita akan membuat grafik 3-Dimensi, sekaligus akan membuat grafik 3-Dimensi tersebut berputar (animasi). Hal ini karena grafik 3-Dimensi yang disajikan statik di atas bidang 2-Dimensi kadang dapat menimbulkan salah persepsi, akibat kurangnya detail informasi. Sehingga akan lebih menarik dan informatif jika disajikan dalam bentuk animasi. Masih melanjutkan kode-kode sebelumnya, silahkan ketikkan dan eksekusi kode-kode berikut:

```
# Visualisasi grafik 3-Dimensi presipitasi per wilayah administrasi
x = presipitasi_gdf.geometry.centroid.x.values
y = presipitasi_gdf.geometry.centroid.y.values
z = presipitasi_gdf['mean'].values

fig = plt.figure(figsize=(8, 8))
ax = fig.add_subplot(111, projection='3d')
img = ax.scatter(x, y, z, c=z, cmap='jet_r', s=100)
fig.colorbar(img, ax=ax, label='Presipitasi (mm/tahun)')
ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_zlabel('Presipitasi (mm/tahun)')
ax.set_title('Grafik Presipitasi Rerata Tahunan Kabupaten Tanah Laut Per Desa \nPeriode 1994-2024')
ax.view_init(elev=30, azim=-45)
plt.show()
```

Berikut adalah output grafik 3D-nya:



Grafik 3-Dimensi sedikitnya memerlukan 3 data numerik, dalam hal ini adalah koordinat  $x$  dan  $y$ , dan nilai  $z$  yaitu presipitasi rerata per desa. Untuk koordinat  $x$  dan  $y$  akan diambil dari titik tengah (`geometry.centroid.x.values` dan `geometry.centroid.y.values`) masing-masing desa. Sementara nilai  $z$  adalah nilai-nilai kolom `mean`, yaitu data presipitasi rerata per desa pada `presipitasi_gdf`. Ada yang baru pada kode-kode di atas, yaitu `fig.add_subplot(111, projection='3d')`. Yang pasti instruksi ini adalah untuk membuat grafik 3D, dimana angka `111` berarti: 1 baris, 1 kolom, dan ini adalah subplot ke-1. Kode `projection='3d'` akan mengaktifkan fitur 3D dari Matplotlib melalui Axes3D. Jika fitur Axes3D ini tidak jalan, maka pada kode-kode di atas harus ditambahkan instruksi `from mpl_toolkits.mplot3d import Axes3D` sebelum instruksi `fig.add_subplot(111, projection='3d')`.

### Bab III Google Earth Engine

`ax.scatter(x, y, z, c=z, cmap='jet_r', s=100)` adalah instruksi untuk membuat scatter plot. Penjelasan selengkapnya tentang scatter plot dapat dilihat langsung di tautan [https://matplotlib.org/stable/api/\\_as\\_gen/matplotlib.axes.Axes.scatter.html](https://matplotlib.org/stable/api/_as_gen/matplotlib.axes.Axes.scatter.html). Instruksi `view_init(elev=30, azim=-45)` merupakan perspektif atau sudut pandang kita ketika melihat grafik 3D, seolah-olah mata kita memandang dari elevasi  $30^{\circ}$  dan azimut  $45^{\circ}$ .

Selanjutnya, kita akan membuat grafik 3D di atas berputar dengan kode-kode berikut:

```
# Membuat animasi grafik 3-Dimensi presipitasi per wilayah administrasi
import matplotlib.animation as animation

def animate(i):
    ax.view_init(azim=-45 + i)
    return fig

ani = animation.FuncAnimation(fig, animate, frames=360, interval=20)
out_gif = '/content/drive/My Drive/geebook/output/precipitation_animation.gif'
ani.save(out_gif, writer='ffmpeg', fps=30)
```

Instruksi untuk membuat animasi menggunakan Matplotlib selengkapnya dapat dibaca di tautan [https://matplotlib.org/stable/api/animation\\_api.html](https://matplotlib.org/stable/api/animation_api.html). Modul `animation` dari Matplotlib memungkinkan pembuatan animasi berbasis frame, baik untuk visualisasi interaktif maupun ekspor ke video atau file GIF. Fungsi `animate` akan dipanggil untuk setiap frame nantinya, dimana dengan `view_init(azim=-45 + i)` sudut pandang akan berubah setiap frame, menciptakan efek rotasi horizontal, yang dimulai dari perspektif  $-45^{\circ}$ . `i` adalah indeks frame, yaitu  $0, 1, 2, \dots, 359$ , tergantung pada `frames=360` yang merupakan jumlah frame animasi (berputar  $360^{\circ}$ ). `FuncAnimation(fig, animate, frames=360, interval=20, blit=False)` merupakan fungsi untuk membuat animasi, dimana `interval=20` adalah waktu antar frame dalam milidetik.

`ani.save(out_gif, writer='ffmpeg', fps=30)` akan menyimpan animasi yang sudah dibuat dengan `FuncAnimation` sebelumnya ke dalam file GIF dengan lokasi dan nama yang didefinisikan di dalam variabel `out_gif`. `writer='ffmpeg'` berfungsi untuk menggunakan `FFmpeg` sebagai *backend* untuk *encoding*. Terkait `FFmpeg`, silahkan baca penjelasannya di laman <https://ffmpeg.org/documentation.html>. Sementara `fps=30` artinya animasi akan diputar dengan kecepatan 30 fps. Jika sudah sukses menjalankan kode-kode di atas, silahkan buka file `precipitation_animation.gif` yang dihasilkan, dan amati gerakan animasinya. Jika diperlukan, Anda dapat merubah beberapa parameter, misalnya `fps`-nya.

## Web Map Interaktif Citra Satelit Multitemporal

Pada bagian sebelumnya kita sudah membuat web map interaktif presipitasi Kabupaten Tanah Laut dengan menggunakan Plotly. Sebenarnya kita juga bisa membuat web map interaktif secara langsung dengan menggunakan Geemap. Terutama jika data yang akan kita ekspor menjadi web map formatnya GEE image. Sebagai studi kasus, kita akan membuat sebuah web map interaktif untuk memantau kondisi morfologi muara Sungai Barito, Kalimantan Selatan, setiap tahun, selama periode 1990 hingga 2024 (35 tahun). Dalam hal ini, kita harus menggunakan Seri Landsat lintas generasi, yaitu Landsat 4, Landsat 5, Landsat 7, Landsat 8, dan Landsat 9. Buat sebuah notebook Google Colab baru, kemudian beri nama misalnya `Web-Map-Landsat.ipynb`. Selanjutnya ketikkan dan jalankan kode-kode berikut:

```

!pip install -U geemap

import ee, geemap

ee.Authenticate()

ee.Initialize(project='ee-geospatialulm')

# Membuat area observasi dari satu titik koordinat
# yang dibuffer 20.000 meter
point = ee.Geometry.Point([114.501383,-3.503110])
area = point.buffer(20000)
region = area.bounds()

# Applying scaling factors.
def apply_scale_factors(image):

    optical_bands = image.select('SR_B_').multiply(0.0000275).add(-0.2)
    thermal_bands = image.select('ST_B_*').multiply(0.00341802).add(149.0)

    image = (
        image.addBands(optical_bands, None, True)
        .addBands(thermal_bands, None, True)
    )

    return image

# Masking clouds and shadows
def mask_clouds(image):

    qa = image.select('QA_PIXEL')

    dilated_bit_mask = 1 << 1
    cirrus_bit_mask = 1 << 2
    cloud_bit_mask = 1 << 3
    shadow_bit_mask = 1 << 4

    mask = (
        qa.bitwiseAnd(dilated_bit_mask).eq(0)
        .And(qa.bitwiseAnd(cirrus_bit_mask).eq(0))
        .And(qa.bitwiseAnd(cloud_bit_mask).eq(0))
        .And(qa.bitwiseAnd(shadow_bit_mask).eq(0))
    )

    return image.updateMask(mask)

# Mengakses Landsat 4
14_col = (
    ee.ImageCollection('LANDSAT/LT04/C02/T1_L2')
    .filterDate('1990-01-01', '1992-12-31')
    .map(apply_scale_factors)
    .map(mask_clouds)
)

# Mengakses Landsat 5
15_col = (
    ee.ImageCollection('LANDSAT/LT05/C02/T1_L2')
    .filterDate('1990-01-01', '2011-12-31')
    .map(apply_scale_factors)
    .map(mask_clouds)
)

# Mengakses Landsat 7
17_col = (
    ee.ImageCollection('LANDSAT/LE07/C02/T1_L2')
    .filterDate('2000-01-01', '2023-12-31')
)

```

### Bab III Google Earth Engine

```
.map(apply_scale_factors)
.map(mask_clouds)
)

# Mengakses Landsat 8
18_col = (
    ee.ImageCollection('LANDSAT/LC08/C02/T1_L2')
    .filterDate('2014-01-01', '2024-12-31')
    .map(apply_scale_factors)
    .map(mask_clouds)
)

# Mengakses Landsat 9
19_col = (
    ee.ImageCollection('LANDSAT/LC09/C02/T1_L2')
    .filterDate('2022-01-01', '2024-12-31')
    .map(apply_scale_factors)
    .map(mask_clouds)
)
```

Kita akan menyajikan web interaktif dengan Citra Landsat komposit RGB. Terkait dengan hal ini, ada satu masalah, yaitu terdapat perbedaan urutan band-band antara Landsat 4, 5, dan 7 dengan Landsat 8 dan 9. Silahkan lihat kembali Gambar 1.9 pada halaman 10 dan Tabel 1.3 pada halaman 14. Untuk menyelesaikan permasalahan ini, kita akan mengambil secara langsung 3 band dari seluruh Seri Landsat untuk dijadikan komposit RGB, yaitu SWIR1, NIR, dan Red. Tentu saja, Anda dapat mengganti kombinasi band-band ini jika diperlukan.

```
# Mengambil 3 band untuk seluruh Seri Landsat
# yaitu band-band SWIR1, NIR, Red

def process_1457(image):
    bands = ['SR_B5', 'SR_B4', 'SR_B3']
    return image.select(bands, ['SWIR1', 'NIR', 'Red']).clip(region)

def process_189(image):
    bands = ['SR_B6', 'SR_B5', 'SR_B4']
    return image.select(bands, ['SWIR1', 'NIR', 'Red']).clip(region)

14_rgb = 14_col.map(process_1457)
15_rgb = 15_col.map(process_1457)
17_rgb = 17_col.map(process_1457)
18_rgb = 18_col.map(process_189)
19_rgb = 19_col.map(process_189)
```

Untuk keperluan visualisasi RGB multitemporal, kita harus menggabungkan seluruh image collection dari seluruh Seri Landsat menjadi satu image collection.

```
# Menggabungkan seluruh Seri Landsat menjadi 1 image collection
merge_landsat = 14_rgb.merge(15_rgb).merge(17_rgb).merge(18_rgb).merge(19_rgb)
```

Langkah berikutnya adalah membuat median nilai spektral untuk setiap tahun. Sehingga diperoleh Citra Landsat tahunan, total ada 35 tahun (1990-2024).

```
# Membuat Citra Landsat median tahunan
def annual_median(year):
    start = ee.Date.fromYMD(year, 1, 1)
    end = start.advance(1, 'year')
    median = merge_landsat.filterDate(start, end).median().set('year', year)
    return median

years = ee.List.sequence(1990, 2024)
annual_medians = ee.ImageCollection(years.map(annual_median))
```

Berikutnya, kita wajib memvisualisasikan Citra Landsat tahunan tersebut dengan menggunakan Geemap. Visualisasi Geemap ini nantinya akan menjadi dasar bagi output web map interaktif yang akan kita buat selanjutnya. Sebagaimana kode-kode berikut:

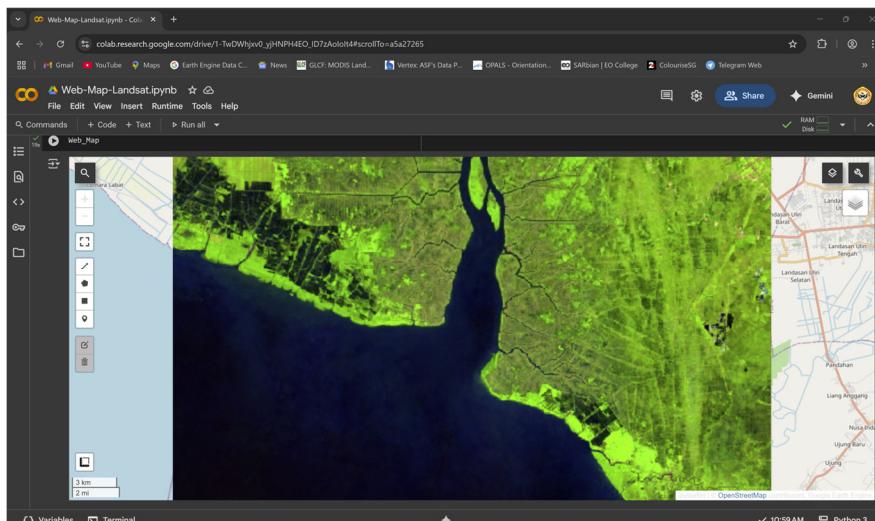
```
# Visualisasi Citra Landsat median tahunan
rgb_vis = {'min': 0, 'max': 0.4, 'bands': ['SWIR1', 'NIR', 'Red']}
Web_Map = geemap.Map()
Web_Map.centerObject(region, 12)

years = list(range(1990, 2025))
labels = [f'Tahun {year}' for year in years]

for year in years:
    start = ee.Date.fromYMD(year, 1, 1)
    end = start.advance(1, 'year')
    median = merge_landsat.filterDate(start, end).median().set('year', year)
    Web_Map.addLayer(median, rgb_vis, f'Tahun {year}')

Web_Map
```

Berikut adalah output visualisasinya:



Pada output di atas, kita dapat mengamati Citra Landsat untuk setiap tahun. Langkah selanjutnya adalah mengekspor output visualisasi Geemap di atas menjadi file HTML ke dalam Google Drive.

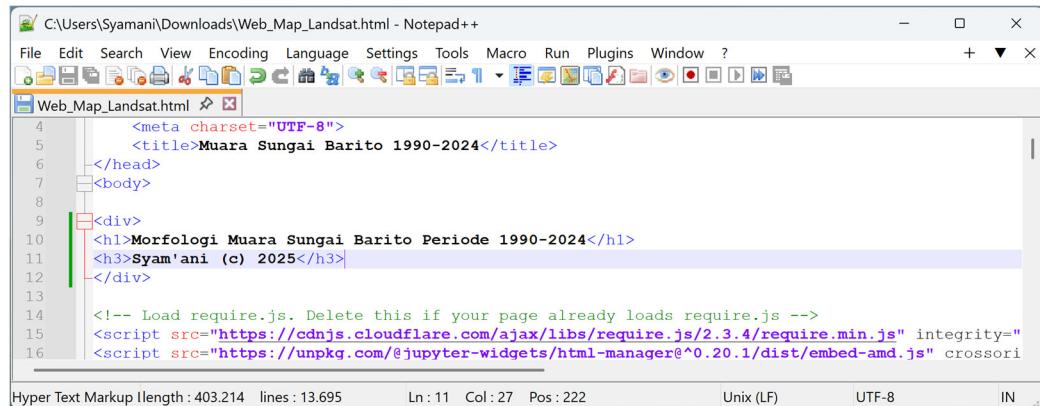
```
# Akses ke Google Drive
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

# Eksport visualisasi ke laman HTML
filename = '/content/drive/MyDrive/geebook/output/web_Map_Landsat.html'
title = 'Muara Sungai Barito 1990-2024'

web_Map.to_html(filename=filename, title=title, width='1024px', height='640px')
```

Setelah file HTML web map kita unduh dari Google Drive, selanjutnya sebagaimana web map interaktif presipitasi Kabupaten Tanah Laut sebelumnya (halaman 366), kita juga dapat menambahkan teks-teks bahkan gambar yang diperlukan ke dalam web map kita.

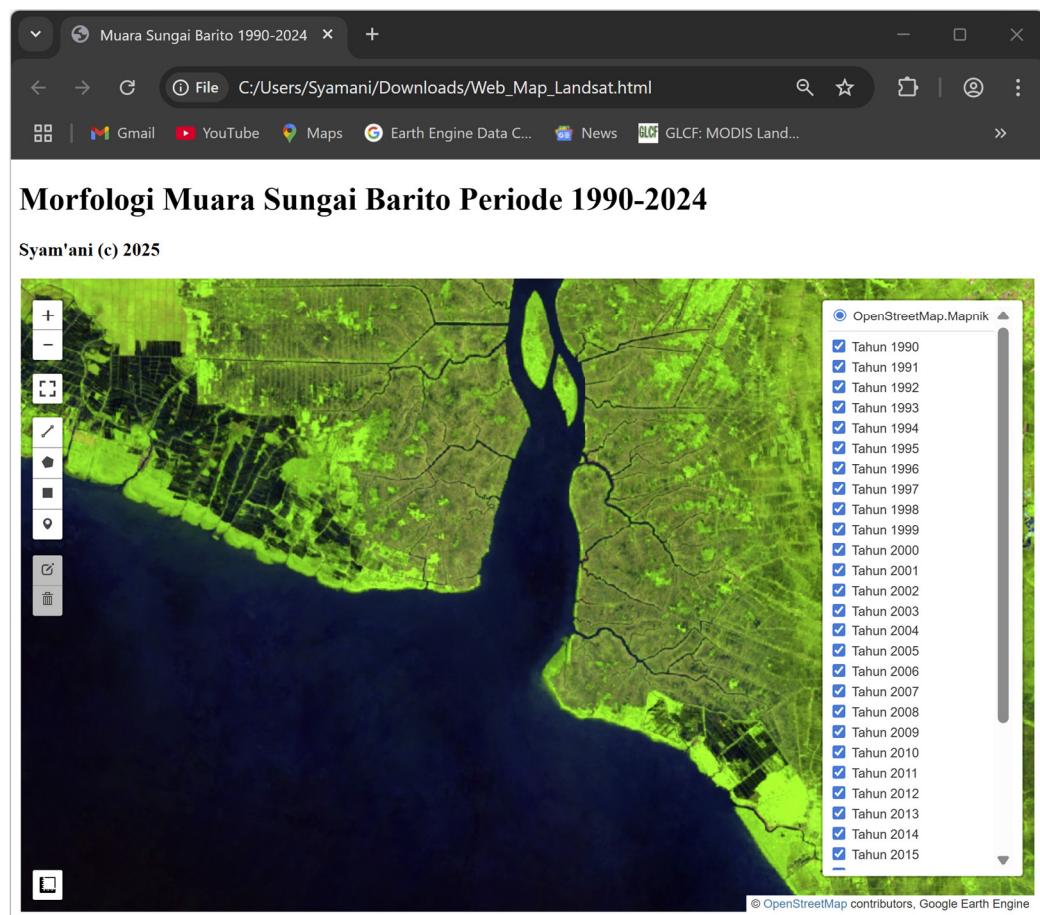
### Bab III Google Earth Engine



```
C:\Users\Syamani\Downloads\Web_Map_Landsat.html - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Web_Map_Landsat.html + ×
4      <meta charset="UTF-8">
5      <title>Muara Sungai Barito 1990-2024</title>
6    </head>
7  <body>
8
9  <div>
10 <h1>Morfologi Muara Sungai Barito Periode 1990-2024</h1>
11 <h3>Syam'ani (c) 2025</h3>
12 </div>
13
14 <!-- Load require.js. Delete this if your page already loads require.js -->
15 <script src="https://cdnjs.cloudflare.com/ajax/libs/require.js/2.3.4/require.min.js" integrity="
16 <script src="https://unpkg.com/@jupyter-widgets/html-manager@0.20.1/dist/embed-amd.js" crossorigin="

Hyper Text Markup Length : 403.214  lines : 13.695     Ln: 11  Col: 27  Pos: 222     Unix (LF)     UTF-8     IN .js
```

Berikut adalah hasil web map interaktif yang sudah kita buat:



Pada web map interaktif di atas, kita dapat memilih Citra Landsat tahun berapa yang mau kita tampilkan untuk kita observasi. Untuk observasi yang lebih detail secara spasial, tentu saja Anda harus menggunakan citra dengan resolusi spasial yang lebih tinggi, Sentinel-2 misalnya. Jika Anda menggunakan Sentinel-2, konsekuensinya rentang waktu observasi tidak sepanjang Seri Landsat. Sebab Satelit Sentinel-2 baru diorbitkan pada pertengahan 2015.

## Permukaan 3 Dimensi dan Animasi Permukaan

Selain grafik, data raster seperti elevasi atau DEM dapat kita visualisasikan secara 3-Dimensi di lingkungan Python/GEE. Sebagai contoh kasus, kita akan membuat visualisasi 3-Dimensi elevasi Gunungapi Merapi. Data yang akan kita gunakan adalah FABDEM (*Forest And Buildings removed Copernicus 30m DEM*) (Hawker et al., 2022). FABDEM merupakan data Copernicus DEM yang memiliki resolusi spasial 30 meter, dimana ketinggian objek-objek seperti pepohonan/hutan dan bangunan sudah dieliminasi. Silahkan buat sebuah notebook Google Colab baru, beri nama misalnya **3D-Surface.ipynb**. Kemudian ketikkan dan jalankan kode-kode berikut:

```
import ee, geemap

ee.Authenticate()

ee.Initialize(project='ee-geospatialulm')

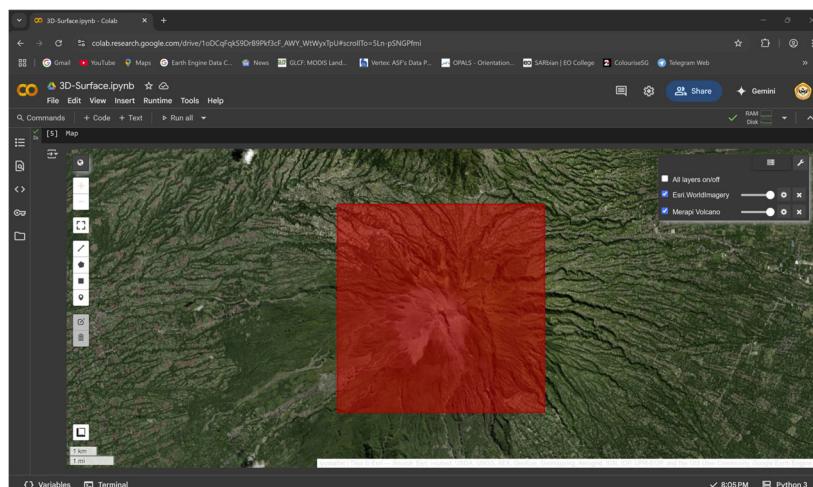
# Menentukan titik pusat observasi (Puncak Gunungapi Merapi)
point = ee.Geometry.Point([110.44598, -7.53863])

# Membuat buffer kotak dari Puncak Merapi dengan jarak 3.700 meter
buffer_radius = 3700
buffered_point = point.buffer(buffer_radius)
region = buffered_point.bounds()
```

Koordinat di atas merupakan titik puncak atau titik tengah kawah Gunungapi Merapi. Dari titik puncak ini, selanjutnya dibuat buffer wilayah dengan radius 3.700 meter.

```
# Visualisasi wilayah buffer
Map = geemap.Map(basemap='SATELLITE')
Map.centerobject(point, 13)
Map.addLayer(region, {'color': 'red'}, 'Merapi Volcano')
Map
```

Output:



### Bab III Google Earth Engine

Langkah berikutnya, kita akan mengakses dataset FABDEM. Untuk lebih jelasnya, silahkan baca di tautan <https://gee-community-catalog.org/projects/fabdem/>.

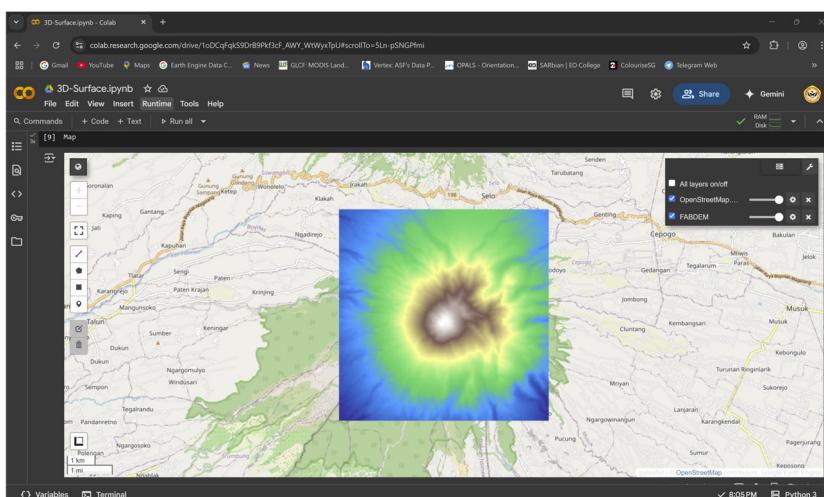
```
# Mengakses FABDEM image collection
fabdem = (
    ee.ImageCollection("projects/sat-io/open-datasets/FABDEM")
    .filterBounds(region)
    .map(lambda image: image.setDefaultProjection('EPSG:4326', None, 30))
```

```
# Membuat DEM image dari FABDEM image collection
dem = fabdem.select('b1').first().clip(region)
geemap.image_stats(dem.clip(region), scale=30)
```

```
# Visualisasi data DEM
dem_vis = {'min': 1100, 'max': 2900, 'palette': 'terrain'}

Map = geemap.Map()
Map.addLayer(dem, dem_vis, 'FABDEM')
Map.centerObject(region, 13)
Map
```

Berikut adalah output visualisasi FABDEM:



Selanjutnya, kita akan membuat visualisasi FABDEM 3-Dimensi dengan kode-kode berikut:

```
# Visualisasi DEM 3-Dimensi
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
from matplotlib.colors import LightSource

region_coords = region.getInfo()['coordinates'][0]
longitude_min, latitude_max = region_coords[0]
longitude_max, latitude_min = region_coords[2]

dem_data = dem.sampleRectangle(region).get('b1').getInfo()
dem_array = np.array(dem_data)
base_elevation = 500
```

```

fill_row = np.full((1, dem_array.shape[1]), base_elevation)
dem_array = np.concatenate((fill_row, dem_array, fill_row), axis=0)

fill_col = np.full((dem_array.shape[0], 1), base_elevation)
dem_array = np.concatenate((fill_col, dem_array, fill_col), axis=1)

x_coords = np.linspace(longitude_min, longitude_max, dem_array.shape[1])
y_coords = np.linspace(latitude_min, latitude_max, dem_array.shape[0])
x, y = np.meshgrid(x_coords, y_coords)

plt.style.use('dark_background')

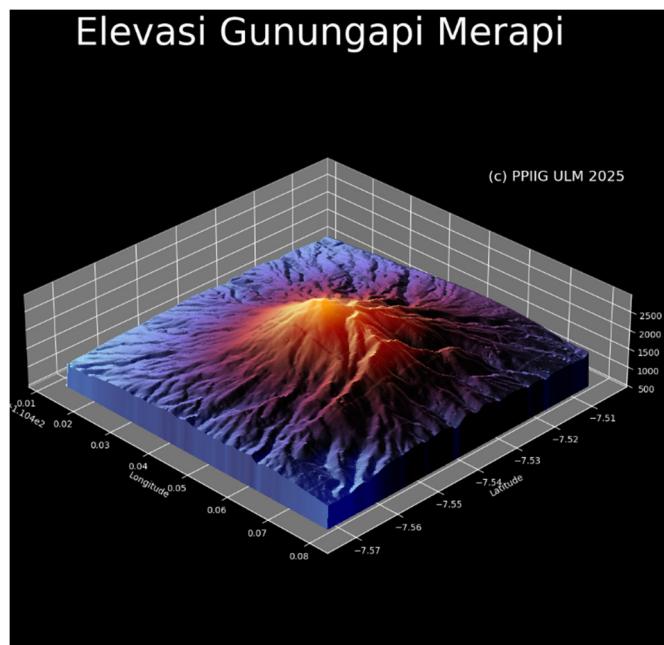
fig = plt.figure(figsize=(12, 12))
ax = fig.add_subplot(111, projection='3d')
ls = LightSource(270, 45)
rgb = ls.shade(dem_array, cmap=cm.managua_r)
surf = ax.plot_surface(x, y, dem_array, facecolors=rgb,
                       lightsource=ls, antialiased=False,
                       rcount = 10000, ccount=10000, shade=False)

ax.set_xlabel('Longitude')
ax.set_ylabel('Latitude')
ax.set_zlabel('Elevasi')
ax.set_title('Elevasi Gunungapi Merapi', fontsize=42)
ax.annotate('(c) PPIIG ULM 2025', xy=(0.05, 0.05), fontsize=16)
ax.view_init(elev=30, azim=-45)
ax.set_box_aspect(aspect = (1,1,0.25))

plt.show()

```

Berikut adalah output visualisasinya:



Visualisasi permukaan 3D di lingkungan Matplotlib memerlukan pengaturan arah dan ketinggian sudut pencahayaan, yaitu menggunakan `LightSource`. Dokumentasinya dapat dibaca di tautan [https://matplotlib.org/stable/api/\\_as\\_gen/matplotlib.colors.LightSource.html](https://matplotlib.org/stable/api/_as_gen/matplotlib.colors.LightSource.html). Karena data DEM biasanya memiliki format raster, maka data DEM harus dikonversi menjadi NumPy array (`dem_array = np.array(dem_data)`). Setelah pixel-pixel DEM diekstrak menjadi array (`dem.sampleRectangle(region).get('b1'). getInfo()`) terlebih dahulu.

`base_elevation = 500` merupakan elevasi terendah dari visualisasi 3D, jika Anda ingin visualisasi 3D dimulai dari titik 0 meter di atas permukaan laut, atur `base_elevation = 0`. `fill_row` dan `fill_col` merupakan instruksi untuk mengisi base\_elevation ke sekeliling DEM, yaitu Utara, Selatan, Barat, Timur, masing-masing sebanyak 1 baris pixel. `fill_row` dan `fill_col` kemudian digabungkan dengan `dem_array` menggunakan instruksi `dem_array = np.concatenate....` Tanpa instruksi-instruksi ini, visualisasi elevasi 3D-nya akan seperti lembaran kertas melayang, tidak ada penutup seperti dinding/permukaan tegak di sekeliling DEM. Sehingga visualisasi 3D-nya menjadi kurang estetik.

`x, y = np.meshgrid(x_coords, y_coords)` adalah instruksi untuk membuat 2 buah array 2D yang berisi koordinat *x* (*longitude*) dan *y* (*latitude*). Dimana data koordinat `x_coords` dan `y_coords` didapatkan dari instruksi `np.linspace....`. `LightSource(270, 45)` digunakan untuk menempatkan sudut pencahayaan pada azimut  $270^{\circ}$  dan elevasi  $45^{\circ}$ , sehingga seolah-olah matahari yang menyinari permukaan 3D berada pada posisi azimut dan elevasi ini. Pengaturan sudut pencahayaan ini kemudian diterapkan pada data dengan instruksi `ls.shade(dem_array, cmap=cm.managua_r)`. `ax.plot_surface...` merupakan kode untuk memplotkan permukaan 3D. Untuk informasi selengkapnya silahkan baca di laman [https://matplotlib.org/stable/api/\\_as\\_gen/mpl\\_toolkits.mplot3d.axes3d.Axes3D.plot\\_surface.html](https://matplotlib.org/stable/api/_as_gen/mpl_toolkits.mplot3d.axes3d.Axes3D.plot_surface.html). Dimana `rcount = 10000` dan `ccount=10000` akan menentukan kehalusan permukaan, semakin tinggi nilainya akan semakin halus permukaan. Konsekuensinya, proses *rendering* animasi akan semakin berat, bahkan jika `rcount` dan `ccount` terlalu besar bisa *crash*.

`antialiased=False` akan menonaktifkan *anti-aliasing* agar tepi permukaan tidak dihaluskan, dan `shade=False` akan menonaktifkan shading otomatis. Untuk `view_init(elev=30, azim=-45)` silahkan lihat kembali penjelasan pada halaman 368. `ax.set_box_aspect(aspect = (1,1,0.25))` merupakan instruksi untuk mengatur rasio atau proporsi aspek dari kotak tampilan 3D (bounding box) agar sumbu Z terlihat lebih rata atau terkompresi dibandingkan sumbu *X* dan *Y*. `aspect = (1,1,0.25)` berarti skala nilai *z* (elevasi) hanya  $\frac{1}{4}$  dari koordinat *x* dan *y* (panjang dan lebar permukaan 3D). Langkah terakhir, kita akan menyimpan animasi permukaan 3D ke dalam file GIF:

```
# Mengakses Google Drive untuk menyimpan file animasi gif
from google.colab import drive
drive.mount('/content/drive', force_remount=True)
path = '/content/drive/My Drive/geebook/'
```

```
# Membuat animasi DEM 3-Dimensi
import matplotlib.animation as animation
def animate(i):
    ax.view_init(azim=-45 + i)
    return fig
ani = animation.FuncAnimation(fig, animate, frames=360, interval=20)
out_gif = path + 'output/dem_merapi.gif'
ani.save(out_gif, writer='ffmpeg', fps=30)
```

Instruksi di atas sama dengan instruksi untuk menyimpan animasi grafik presipitasi 3D sebelumnya. Silahkan lihat kembali penjelasan pada halaman 368. Selanjutnya, Anda dapat membuka file `dem_merapi.gif` untuk melihat output animasi permukaan 3D.

## REFERENSI

- Abatzoglou, J.T., Dobrowski, S.Z., Parks, S.A., Hegewisch, K.C., 2018. Terraclimate, a high-resolution global dataset of monthly climate and climatic water balance from 1958–2015. *Scientific Data* 5 (170191). DOI: <https://doi.org/10.1038/sdata.2017.191>.
- Abedini, M., 2000. *Satellite image fusion using Brovey*. Master Thesis, Geomatics engineering faculty, K.N. Toosi University of Technology, Tehran, Iran.
- Adams, J.B., Smith, M.O., Johnson, P.E., 1986. Spectral mixture modeling: A new analysis of rock and soil types at the Viking Lander 1 Site. *Journal of Geophysical Research: Solid Earth* 91 (B8), 8098-8112. DOI: <https://doi.org/10.1029/JB091iB08p08098>.
- Adams, J. B., Smith, M. O., Gillespie, A. R., 1993. Imaging spectroscopy: Interpretation based on spectral mixture analysis. In C.M. Pieters, P.A.J. Englert (Eds.), *Remote Geochemical Analysis: Elemental and Mineralogical Composition*. Cambridge: Cambridge University Press, 145-166. DOI: <https://doi.org/10.1017/S0016756800012681>.
- Ali, S.D., Ridwan, I., Septiana, M., Fithria, A., Rezekiah, A.A., Rahmadi, A., Asyari, M., Rahman, H., Syafarina, G.A., 2022. GeoAI for Disaster Mitigation: Fire Severity Prediction Models using Sentinel-2 and ANN Regression. *2022 IEEE International Conference on Aerospace Electronics and Remote Sensing Technology (ICARES)*, Yogyakarta, Indonesia, 24-25 November 2022, 1-7. DOI: <https://doi.org/10.1109/ICARES56907.2022.9993515>.
- Achanta, R., Süstrunk, S., 2017. Superpixels and Polygons Using Simple Non-iterative Clustering. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, USA, 4895-4904. DOI: <https://doi.org/10.1109/CVPR.2017.520>.
- Anaconda Software Distribution, 2020. *Anaconda Documentation*. Anaconda Inc. URL: <https://docs.anaconda.com>.
- Atkinson, P. M., 1997. Mapping sub-pixel boundaries from remotely sensed images. In Kemp, Z. (Ed.) *Innovations in GIS 4 (Selected Papers from the Fourth National Conference on GIS Research UK (GISRUK))*. Taylor & Francis, London. DOI: <https://doi.org/10.1201/9781482272956>.
- Badan Standardisasi Nasional, 2020. SNI 7717:2020 tentang Spesifikasi informasi geospasial – Mangrove skala 1:25.000 dan 1:50.000. Ditetapkan tanggal 30 December 2020. URL: <https://pesta.bsn.go.id/produk/detail/13110-sni77172020>.
- Baloloy, A.B., Blanco, A.C., Ana, R.R.C.Sta., Nadaoka, K., 2020. Development and application of a new mangrove vegetation index (MVI) for rapid and accurate mangrove mapping. *ISPRS Journal of Photogrammetry and Remote Sensing* 166, 95-117. DOI: <https://doi.org/10.1016/j.isprsjprs.2020.06.001>.
- Blaschke, T., 2010. Object based image analysis for remote sensing. *ISPRS Journal of Photogrammetry and Remote Sensing* 65 (1), 2-16. DOI: <https://doi.org/10.1016/j.isprsjprs.2009.06.004>.
- Blaschke, T., Hay, G.J., Kelly, M., Lang, S., Hofmann, P., Addink,E., Feitosa, R.Q., van der Meer, F., van der Werff, H., van Coillie, F., Tiede, D., 2014. Geographic Object-Based Image Analysis – Towards a new paradigm. *ISPRS Journal of Photogrammetry and Remote Sensing* 87, 180-191. DOI: <https://doi.org/10.1016/j.isprsjprs.2013.09.014>.

## Referensi

---

- Bokeh Development Team, 2018. *Bokeh: Python library for interactive visualization*. URL: <http://www.bokeh.pydata.org>.
- Bray, T., 2014. *The javascript object notation (json) data interchange format*.
- Breiman, L., 2001. Random Forests. *Machine Learning* 45, 5-32. DOI: <https://doi.org/10.1023/A:1010933404324>.
- Campbell, J.B., Wynne, R.H., 2011. *Introduction to Remote Sensing, Fifth Edition*. The Guilford Press, New York.
- Chen, G., Weng, Q., Hay, G.J., He, Y., 2018. Geographic object-based image analysis (GEOBIA): emerging trends and future opportunities. *GIScience & Remote Sensing* 55 (2), 159-182. DOI: <https://doi.org/10.1080/15481603.2018.1426092>.
- Clark-Carter, D., 2005. Interquartile Range. In Everitt, B.S., Howell, D.C. (Eds.) *Encyclopedia of Statistics in Behavioral Science*. John Wiley & Sons, Ltd., Hoboken, NJ. DOI: <https://doi.org/10.1002/0470013192.bsa311>.
- Cohen, J., 1960. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement* 20 (1), 37-46. <https://doi.org/10.1177/001316446002000104>.
- Congalton, R.G., 1991. A review of assessing the accuracy of classifications of remotely sensed data. *Remote Sensing of Environment* 37 (1), 35-46. DOI: [https://doi.org/10.1016/0034-4257\(91\)90048-B](https://doi.org/10.1016/0034-4257(91)90048-B).
- Copilot, 2025a. *Response to Create a high-resolution satellite image depicting a yellowish-green rice field surrounded by forest and settlements*. Microsoft. Retrieved July 13, 2025. URL: <https://copilot.microsoft.com>.
- Copilot, 2025b. *Response to Create an illustration of pure pixels and mixed pixels on top of remote sensing satellite imagery*. Microsoft. Retrieved July 20, 2025. URL: <https://copilot.microsoft.com>.
- Copilot, 2025c. *Response to Change the appearance of this image to make it look natural (an image design copied and pasted into Copilot)*. Microsoft. Retrieved July 20, 2025. URL: <https://copilot.microsoft.com>.
- Crockford, D., 2006. *The application/json media type for javascript object notation (json)*.
- Dent, B.D., Torguson, J.S., Hodler, T.W., 2008. *Cartography: Thematic Map Design (6th Edition)*. McGraw-Hill Education, Columbus.
- Gao, B.C., 1996. NDWI A – Normalized Difference Water Index for Remote Sensing of Vegetation Liquid Water From Space. *Remote Sensing of Environment* 58 (3), 257-266. [https://doi.org/10.1016/S0034-4257\(96\)00067-3](https://doi.org/10.1016/S0034-4257(96)00067-3).
- GDAL/OGR contributors, 2024. *GDAL/OGR Geospatial Data Abstraction software Library*. Open Source Geospatial Foundation. DOI: <https://doi.org/10.5281/zenodo.5884351>.
- Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., Moore, R., 2017. Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment* 202, 18-27. DOI: <https://doi.org/10.1016/j.rse.2017.06.031>.
- Gillies, S. et al., 2019. *Rasterio: geospatial raster I/O for Python programmers*. Mapbox. URL: <https://github.com/rasterio/rasterio>.

- Haralick, R.M., Shanmugam, K., Dinstein, I., 1973. Textural Features for Image Classification. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-3 (6), 610-621. DOI: <https://doi.org/10.1109/TSMC.1973.4309314>.
- Harris, C.R., Millman, K.J., van der Walt, S.J. et al., 2020. Array programming with NumPy. *Nature* 585, 357-362. DOI: <https://doi.org/10.1038/s41586-020-2649-2>.
- Hawker, L., Uhe, P., Paulo, L., Sosa, J., Savage, J., Sampson, C., Neal, J., 2022. A 30 m global map of elevation with forests and buildings removed. *Environmental Research Letters* 17 (2), 024016. DOI: <https://doi.org/10.1088/1748-9326/ac4d4f>.
- Hazrat, R., 2023. *A Course in Python: The Core of the Language*. Springer Nature Switzerland AG, Cham. DOI: <https://doi.org/10.1007/978-3-031-49780-3>.
- Ho, T.K., 1995. Random decision forests. *Proceedings of 3rd International Conference on Document Analysis and Recognition*, 14-16 August 1995, 278-282. DOI: <https://doi.org/10.1109/ICDAR.1995.598994>.
- Holden, Z. A., Smith, A. M. S., Morgan, P., Rollins, M. G., Gessler, P. E., 2005. Evaluation of Novel Thermally Enhanced Spectral Indices for Mapping Fire Perimeters and Comparisons with Fire Atlas Data. *International Journal of Remote Sensing* 26 (21): 4801-4808. DOI: <https://doi.org/10.1080/01431160500239008>.
- Hoyer, S., Hamman, J., 2017. xarray: N-D labeled Arrays and Datasets in Python. *Journal of Open Research Software*. 5c(1), 10. DOI: <https://doi.org/10.5334/jors.148>.
- Hunt, J., 2023. *A Beginners Guide to Python 3 Programming (Second Edition)*. Springer Nature Switzerland AG, Cham. DOI: <https://doi.org/10.1007/978-3-031-35122-8>.
- Hunter, J. D., 2007. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* 9 (3), 90-95. DOI: <https://doi.org/10.1109/MCSE.2007.55>.
- Jiang, B., 2012. Head/Tail Breaks: A New Classification Scheme for Data with a Heavy-Tailed Distribution. *The Professional Geographer* 65 (3), 482-494. DOI: <https://doi.org/10.1080/00330124.2012.700499>.
- Jordahl, K., Van den Bossche, J., Fleischmann, M. et al., 2020. *geopandas/geopandas: v0.8.1*. Zenodo. DOI: <https://doi.org/10.5281/zenodo.3946761>.
- Ilsever, M., Unsalan, C., 2012. *Two-Dimensional Change Detection Methods: Remote Sensing Applications*. Springer, London. DOI: <https://doi.org/10.1007/978-1-4471-4255-3>.
- Jaccard, P., 1901. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin de la Societe Vaudoise des Sciences Naturelles* 37 (142), 547-579. DOI: <http://dx.doi.org/10.5169/seals-266450>.
- Jensen, J.R., 2014. *Remote Sensing of the Environment An Earth Resource Perspective (Second Edition)*. Pearson Education Limited, Harlow, Essex, UK.
- Jensen, J.R., 2015. *Introductory Digital Image Processing: A Remote Sensing Perspective (4th Edition)*. Pearson Series in Geographic Information Science, Pearson Education, Inc., Glenview.
- Jin, X., Han, J., 2011. K-Medoids Clustering. In Sammut, C., Webb, G.I. (Eds.) *Encyclopedia of Machine Learning*. Springer, Boston, MA. DOI: [https://doi.org/10.1007/978-0-387-30164-8\\_426](https://doi.org/10.1007/978-0-387-30164-8_426).

## Referensi

---

- Kawamura, M., Jayamana, S., Tsujiko, Y., 1996. Relation between Social and Environmental Conditions in Colombo Sri Lanka and the Urban Index Estimated by Satellite Remote Sensing Data. *International Archives of the Photogrammetry and Remote Sensing* 31, 321-326. URL: <https://www.isprs.org/proceedings/XXXI/congress/part7/321 XXXI-part7.pdf>.
- Keshava, N., Mustard, J.F., 2002. Spectral unmixing. *IEEE Signal Processing Magazine* 19 (1), 44-57. DOI: <https://doi.org/10.1109/79.974727>.
- Keshava, N., 2003. A Survey of Spectral Unmixing Algorithms. *Lincoln Laboratory Journal* 14 (1), 55-78. URL: [https://archive.ll.mit.edu/publications/journal/pdf/vol14\\_no1/14\\_1survey.pdf](https://archive.ll.mit.edu/publications/journal/pdf/vol14_no1/14_1survey.pdf).
- Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S., Willing, C., 2016. Jupyter Notebooks – a publishing format for reproducible computational workflows. In Loizides, F., Schmidt, B. (Eds.), *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, 87-90. DOI: <http://dx.doi.org/10.3233/978-1-61499-649-1-87>.
- Kwong, I.H.Y., Wong, F.K.K., Fung, T., 2022. Automatic Mapping and Monitoring of Marine Water Quality Parameters in Hong Kong Using Sentinel-2 Image Time-Series and Google Earth Engine Cloud Computing. *Frontiers in Marine Science* 9, 871470. DOI: <https://doi.org/10.3389/fmars.2022.871470>.
- Lacaux, J.P., Tourre, Y., Vignolles, C., Ndione, J.A., Lafaye, M., 2007. Classification of Ponds from High-Spatial Resolution Remote Sensing: Application to Rift Valley Fever Epidemics in Senegal. *Remote Sensing of Environment* 106 (1), 66-74. DOI: <https://doi.org/10.1016/j.rse.2006.07.012>.
- Landis, J.R., Koch, G.G., 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics* 33 (1), 159-174. DOI: <https://doi.org/10.2307/2529310>.
- Liu, H., Li, Q., Shi, T., Hu, S., Wu, G., Zhou, Q., 2017. Application of Sentinel 2 MSI Images to Retrieve Suspended Particulate Matter Concentration in Poyang Lake. *Remote Sensing* 9 (761), 1-19. DOI: <https://doi.org/10.3390/rs9070761>.
- MacQueen, J.B., 1967. Some Methods for Classification and Analysis of Multivariate Observations. *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability*, Berkeley, University of California Press, 1, 281-297. URL: <http://projecteuclid.org/euclid.bsmsp/1200512992>.
- Malila, W.A., 1980. Change Vector Analysis: An Approach for Detecting Forest Changes with Landsat. *Proceedings of the 6th Annual Symposium on Machine Processing of Remotely Sensed Data*, 03-06 June, Purdue University, West Lafayette, Indiana, 326-335. URL: [https://docs.lib.psu.edu/lars\\_symp/385](https://docs.lib.psu.edu/lars_symp/385).
- Markert, K.N., 2019. cartoee: Publication quality maps using Earth Engine. *Journal of Open Source Software* 4 (33), 1207. DOI: <https://doi.org/10.21105/joss.01207>.
- McKinney, W., 2010. Data Structures for Statistical Computing in Python. *Proceedings of the 9th Python in Science Conference*, Austin, 28 June-3 July 2010, 56-61. DOI: <https://doi.org/10.25080/Majora-92bf1922-00a>.
- Met Office, 2010 – 2013. Cartopy: a cartographic python library with matplotlib support. Exeter, Devon. URL: <http://scitools.org.uk/cartopy>.

- Müller, R., Pfeifroth, U., 2022. Remote sensing of solar surface radiation – a reflection of concepts, applications and input data based on experience with the effective cloud albedo. *Atmospheric Measurement Techniques* 15 (5), 1537-1561. DOI: <https://doi.org/10.5194/amt-15-1537-2022>.
- Otsu, N., 1979. A Threshold Selection Method from Gray-level Histograms. *IEEE Transactions on Systems, Man, and Cybernetics* 9 (1), 62-69. DOI: <https://doi.org/10.1109/TSMC.1979.4310076>.
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12, 2825-2830. DOI: <https://doi.org/10.48550/arXiv.1201.0490>.
- Pelleg, D., Moore, A.W., 2000. X-means: Extending K-means with Efficient Estimation of the Number of Clusters. In Langley, P. (Ed.), *Proceedings of the Seventeenth International Conference on Machine Learning (ICML '00)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 727-734. DOI: <https://dl.acm.org/doi/10.5555/645529.657808>.
- Pezoa, F., Reutter, J.L., Suarez, F., Ugarte, M., Vrgoč, D., 2016. Foundations of JSON Schema. *Proceedings of the 25th International Conference on World Wide Web*, Montréal Québec Canada, April 11-15, 2016, 263-273. DOI: <https://doi.org/10.1145/2872427.2883029>.
- Pinty, B., Verstraete, M.M., 1992. GEMI: a non-linear index to monitor global vegetation from satellites. *Vegetatio* 101, 15–20. DOI: <https://doi.org/10.1007/BF00031911>.
- Plaza, A., Martinez, P., Perez, R., Plaza, J., 2002. Spatial/spectral endmember extraction by multidimensional morphological operations. *IEEE Transactions on Geoscience and Remote Sensing* 40 (9), 2025-2041. DOI: <https://doi.org/10.1109/TGRS.2002.802494>.
- Raybaut, P., 2009. *Spyder-documentation*. URL: <https://docs.spyder-ide.org/>.
- Rey, S.J., Stephens, P., Laura, J., 2016. An evaluation of sampling and full enumeration strategies for Fisher Jenks classification in big data settings. *Transactions in GIS* 21 (4), 796-810. DOI: <https://doi.org/10.1111/tgis.12236>.
- Richards, J.A., 2022. *Remote Sensing Digital Image Analysis (Sixth Edition)*. Springer-Verlag, Berlin. <https://doi.org/10.1007/978-3-030-82327-6>.
- Plotly Technologies Inc., 2015. Collaborative data science. URL: <https://plot.ly>.
- Rouse, J.W., Haas, R.H., Schell, J.A., Deering, D. W., 1973. Monitoring vegetation systems in the Great Plains with ERTS. *Third ERTS Symposium*, NASA SP-351 I, 309-317. URL: <https://ntrs.nasa.gov/citations/19740022614>.
- Schowengerdt, R.A., 2007. *Remote Sensing: Models and Methods for Image Processing (Third Edition)*. Academic Press, Burlington. DOI: <https://doi.org/10.1016/B978-0-12-369407-2.X5000-1>.
- Singh, A., 1989. Review Article Digital Change Detection Techniques Using Remotely-Sensed Data. *International Journal of Remote Sensing* 10 (6), 989-1003. DOI: <https://doi.org/10.1080/01431168908903939>.
- Slocum, T.A., McMaster, R.B., Kessler, F.C., Howard, H.H., 2009. *Thematic cartography and geovisualization*. Pearson Prentice Hall, Upper Saddle River.

## Referensi

---

- Stehman, S.V., Czaplewski, R.L., 1998. Design and Analysis for Thematic Map Accuracy Assessment: Fundamental Principles. *Remote Sensing of Environment* 64, 331-344. DOI: [https://doi.org/10.1016/S0034-4257\(98\)00010-8](https://doi.org/10.1016/S0034-4257(98)00010-8).
- Teoh, T.T., Rong, Z., 2022. Python for Artificial Intelligence. In *Artificial Intelligence with Python. Machine Learning: Foundations, Methodologies, and Applications*. Springer, Singapore. DOI: [https://doi.org/10.1007/978-981-16-8615-3\\_1](https://doi.org/10.1007/978-981-16-8615-3_1).
- The pandas development team, 2020. *pandas-dev/pandas: Pandas*. Zenodo. DOI: <https://doi.org/10.5281/zenodo.3509134>.
- Van der Walt, S., Schönberger, J.L., Nunez-Iglesias, J., Boulogne, F., Warner, J.D., Yager, N., Gouillart, E., Yu, T., 2014. scikit-image: image processing in Python. *PeerJ* 2, e453. DOI: <https://doi.org/10.7717/peerj.453>.
- Van Rossum, G., Drake, F. L., 2009. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace. DOI: <https://dl.acm.org/doi/book/10.5555/1593511>.
- Velastegui-Montoya, A., Montalván-Burbano, N., Carrión-Mero, P., Rivera-Torres, H., Sadeck, L., Adami, M., 2023. Google Earth Engine: A Global Analysis and Future Trends. *Remote Sensing* 15 (14), 3675. DOI : <https://doi.org/10.3390/rs15143675>.
- Vijayakumar, S., Saravanakumar, R., Arulanandam, Arulanandam, M., Ilakkiya, S., 2024. Google Earth Engine: empowering developing countries with large-scale geospatial data analysis—a comprehensive review. *Arabian Journal of Geosciences* 17, 139. DOI: <https://doi.org/10.1007/s12517-024-11948-x>.
- Virtanen, P., Gommers, R., Oliphant, T.E. et al., 2020. SciPy 1.0: fundamental algorithms for scientific computing in Python. *Nature Methods* 17, 261-272. <https://doi.org/10.1038/s41592-019-0686-2>.
- Waskom, M.L., 2021. seaborn: statistical data visualization. *Journal of Open Source Software*, 6 (60), 3021, <https://doi.org/10.21105/joss.03021>.
- Wu, J., Li, T., Lin, L., Zeng, C., 2024. Progressive gap-filling in optical remote sensing imagery through a cascade of temporal and spatial reconstruction models. *Remote Sensing of Environment* 311, 114245. DOI: <https://doi.org/10.1016/j.rse.2024.114245>.
- Wu, Q., 2020. geemap: A Python package for interactive mapping with Google Earth Engine. *The Journal of Open Source Software* 5 (51), 2305. DOI: <https://doi.org/10.21105/joss.02305>.
- Wu, Q., Lane, C. R., Li, X., Zhao, K., Zhou, Y., Clinton, N., DeVries, B., Golden, H. E., Lang, M. W. 2019. Integrating LiDAR data and multi-temporal aerial imagery to map wetland inundation dynamics using Google Earth Engine. *Remote Sensing of Environment* 228, 1-13. DOI: <https://doi.org/10.1016/j.rse.2019.04.015>.
- Xiaona, W., Jinyan, T., Xiaojuan, L., Le, W., Huili, G., Beibei, C., Xiangcai, L., Jinghan, G., 2022. Benefits of Google Earth Engine in remote sensing. *National Remote Sensing Bulletin* 26 (2), 299-309. DOI: <https://dx.doi.org/10.11834/jrs.20211317>.
- Xu, H., 2006. Modification of Normalized Difference Water Index (NDWI) to Enhance Open Water Features in Remotely Sensed Imagery. *International Journal of Remote Sensing* 27 (14), 3025–3033. DOI: <https://doi.org/10.1080/01431160600589179>.
- Zeng, C., Shen, H., Zhang, L., 2013. Recovering missing pixels for Landsat ETM + SLC-off imagery using multi-temporal regression analysis and a regularization method. *Remote Sensing of Environment* 131, 182-194. DOI: <https://doi.org/10.1016/j.rse.2012.12.012>.

## INDEKS

---

### A

AI · 141, 142, 145, 146, 149, 151, 152, 153  
Artificial Intelligence · 56, 141

---

### B

bandwidth · 8  
binary · 3, 4, 8, 205, 263, 282, 310  
binary digit · 4, 8  
binary image · 205, 282, 310  
biner · 3, 4, 206  
bit · 4, 8, 14, 18, 62, 195, 228, 233, 247, 316, 352, 369

---

### C

change detection · 235, 299  
ChatGPT · 141, 145, 151  
choropleth · 129, 362, 363, 364, 365  
Comission Error · 273  
Confusion Matrix · 260, 270, 271, 287, 288  
Copilot · 141, 152, 153, 154

---

### D

data science · 56, 158  
deep learning · 5, 47, 56, 107, 140, 158  
drone · 8

---

### E

Earth Movers Distance · 226  
emitance · 6  
endmember · 255, 334, 335, 339, 340, 341, 343, 344, 345, 346  
enhancement · 222  
Euclidean Distance · 310  
exo-atmospheric reflectance · 7

---

### F

FABDEM · 373, 374  
fire severity · 244, 245, 254  
fraksi endmember · 334, 335, 344

---

## G

Gemini · 141, 142, 144, 145, 146, 148, 149, 150, 151, 153  
GEOBIA · 264  
geostasisioner · 8  
GitHub · 20, 42, 152, 153, 154  
GLCM · 224, 225  
Gray Level Co-occurrence Matrix · 224

---

## H

hard classification · 255  
harmonik · 326, 328  
heksadesimal · 3, 4, 202, 358, 359

---

## I

image filtering · 222  
image segmentation · 255  
informational class · 264  
inheritance · 95, 96  
interpreter · 56, 62, 144  
Interquartile Range · 321  
Intersection Over Union · 283  
IoU · 283, 284, 286, 287, 288

---

## J

Jaccard Index · 283  
JavaScript Object Notation · 174  
JSON · 174, 194

---

## K

k-Means · 262, 263  
k-Medoids · 262  
Koefisien Kappa · 261, 274  
kompiler · 62, 144

---

## L

Lambda · 87  
Land Surface Temperature · 6, 15, 235  
Lidar · 6, 19  
Linear Mixture Model · 334  
Linear Spectral Mixture Analysis · 334  
LMM · 334, 344, 346  
Logical Error · 144

---

LSMA · 334  
LSU · 334, 335, 336, 342, 344

---

## M

machine learning · 5, 42, 56, 107, 140, 158, 241, 259, 262  
Mangrove Vegetation Index · 291, 338  
MAPE · 241, 242, 243, 260  
Mean Absolute Percentage Error · 241  
mixed pixel · 333, 334  
Modified Normalized Difference Water Index · 138, 202, 204, 339

---

## N

Normalize Difference Moisture Index · 244  
Normalized Burn Ratio Thermal · 244  
Normalized Difference Turbidity Index · 203, 205, 317  
Normalized Difference Vegetation Index · 114, 198, 242, 339

---

## O

OBIA · 255, 264, 265, 266, 268, 269, 270, 271, 273, 274, 275, 276  
Omission Error · 273  
optik · 6, 7, 9, 13, 15, 194, 213  
Overall Accuracy · 261, 273

---

## P

Pixel Purity Index · 335  
pixelwise classification · 255, 256, 333  
polynomial · 324, 325, 326, 328  
PPI · 335, 349  
Precision · 283, 286, 287, 288  
Producer's Accuracy · 261, 273, 287  
pure pixel · 333, 335, 337, 339, 341, 342, 343, 344, 346, 349

---

## R

Radar · 6, 12  
radiance · 7  
Random Forest · 259, 269, 270, 271  
Recall · 283, 286, 287, 288  
reflectance · 6, 7, 15, 171, 198, 200, 226, 247, 290, 297, 334, 335, 336, 342  
Region of Interest · 207  
Resolusi radiometrik · 8  
Resolusi spasial · 7  
Resolusi spektral · 8  
Resolusi temporal · 8  
revisit time · 8, 15

## Indeks

---

RMSE · 241, 242, 243, 260, 330

ROI · 207, 219, 221

Root Mean Square Error · 241

---

## S

scatter plot · 311, 312, 313, 368

SNIC · 266

soft classification · 255

Spectral Angle Mapper · 226

spectral class · 264

Spectral Information Divergence · 226

spectral signature · 264

spectral unmixing · 333, 334

spektrum · 5, 7, 8

Squared Euclidean Distance · 226

sub-pixel classification · 255

super resolution · 198

superpixel classification · 255

supervised multispectral classification · 256

surface reflectance · 7

Syntax Error · 144

---

## T

termal · 6, 10, 11, 15, 194

TerraClimate · 356, 357, 359

threshold · 72, 202, 282, 298, 299, 300, 303, 306, 308, 309, 313, 314, 338

Thresholding · 202, 205, 300, 303, 306, 309

timelapse animation · 350, 351, 353, 354, 355

Top of Atmosphere · 7

Top of Canopy · 7

Transformasi Brovey · 208, 209, 210, 211, 212

Transformasi Citra · 198

Transformasi Spasial · 208

Transformasi Spektral · 198

Transformasi Temporal · 213

---

## U

United States Geological Survey · 9

Unmanned Aerial Vehicle (UAV) · 8

unsupervised multispectral classification · 262

Urban Index · 307, 339

User's Accuracy · 261, 273, 287

USGS · 9, 11, 245

---

## X

x-Means · 263



*“Citra digital tersusun atas pixel-pixel, dan pixel-pixel sebenarnya hanyalah angka-angka. Sehingga, semua yang dapat Anda lakukan pada angka, pasti dapat Anda lakukan terhadap citra.”*  
(Syam’ani, 2025)

---



**Syam’ani** adalah dosen ahli penginderaan jauh pada Fakultas Kehutanan Universitas Lambung Mangkurat, Banjarbaru, Kalimantan Selatan. Penulis juga merupakan alumni magister Penginderaan Jauh, Fakultas Geografi, Universitas Gadjah Mada. Pada tahun 2008, penulis mengembangkan penelitian tesis *machine learning*, dengan topik algoritma *Bayesian Model Averaging* untuk klasifikasi digital penutupan lahan menggunakan citra penginderaan jauh, di bawah bimbingan Prof. Dr. Hartono, DEA, DESS dan Prof. Drs. Projo Danoedoro, M.Sc., Ph.D.

Di bidang penginderaan jauh sendiri, penulis lebih berfokus pada subjek pengembangan formula, algoritma, dan metodologi pemrosesan citra digital. Di samping juga menekuni topik-topik riset penginderaan jauh dan informasi geospasial terapan untuk bidang ilmu kehutanan, lingkungan, dan bidang-bidang yang terkait. Penulis sudah memiliki sejumlah publikasi ilmiah yang terkait dengan formula atau algoritma pemrosesan citra penginderaan jauh dan penginderaan jauh terapan. Selain berpengalaman menjadi pemakalah di seminar-seminar nasional dan internasional, penulis juga sudah sering menjadi narasumber di berbagai kegiatan pelatihan, bimbingan teknis, workshop, webinar, dan sebagainya.

Buku ini merupakan sebuah respons atas perkembangan teknologi pemrosesan citra penginderaan jauh berbasis komputasi awan (*cloud computing*) yang begitu pesat akhir-akhir ini. Kehadiran platform komputasi awan seperti Google Earth Engine (GEE) mampu membuat analisis citra digital penginderaan jauh menjadi lebih efisien, sebab pemrosesan tidak lagi membebani sumberdaya mesin yang kita gunakan. Sehingga berbagai proyek riset, rutinitas pemantauan permukaan bumi, atau pun penyediaan data geospasial untuk infrastruktur informasi geospasial pemerintah daerah, akan dapat diselenggarakan secara lebih cepat dan murah. Tentu saja, selain akses internet, hal lainnya yang harus dipenuhi adalah sumberdaya manusia yang mengerti menggunakan platform teknologi tersebut.

Buku ini disusun sebagai bahan untuk mempersiapkan sumberdaya manusia di dalam memanfaatkan teknologi komputasi awan, khususnya pemrosesan citra penginderaan jauh berbasis GEE. Sudah dapat dipastikan, bahwa perkembangan teknologi *cloud-based remote sensing* tidak akan pernah berhenti. Sehingga setelah Anda mempelajari isi buku ini, langkah terpenting yang harus dilakukan selanjutnya adalah terus berkembang dan berusaha untuk menggali ilmu-ilmu baru dari berbagai sumber.

---



Jl. Hasan Basri, Kayu Tangi,  
Banjarmasin, 70123  
Telp./Fax. 0511 – 3305195  
ANGGOTA APPTI (004.035.1.03.2018)

