

BUKU AJAR

METODE KOMPUTASI GEOFISIKA

MENGGUNAKAN PYTHON

IKTRI MADRINOVELLA
IDA HERAWATI
AGUS ABDULLAH

BUKU AJAR

Metode Komputasi Geofisika

Menggunakan Python

Iktri Madrinovella

Ida Herawati

Agus Abdullah

Edisi Pertama, 2020

Buku Ajar Metode Komputasi Geofisika Menggunakan Python

Penulis:

Iktri Madrinovella

Ida Herawati

Agus Abdullah

ISBN: 978-623-9403-00-3

Penerbit: Universitas Pertamina

Cetakan Pertama, 2020

Edisi Pertama, 2020

Hak Cipta dilindungi undang-undang

Dilarang memperbanyak karya tulis ini dalam bentuk dan dengan cara apapun tanpa ijin tertulis dari penerbit.

KATA PENGANTAR

Penyusunan Buku Ajar Metode Komputasi Geofisika Menggunakan Python ini diawali sebagai penunjang perkuliahan Metode Komputasi mahasiswa Teknik Geofisika Universitas Pertamina, didukung perkembangan teknologi dan informasi dalam perangkat lunak yang digunakan. Pada buku ini, lebih banyak menjelaskan tentang penggunaan dasar bahasa pemrograman Python, yang kemudian ditambahkan dengan beberapa contoh aplikasi penggunaan Python untuk metode Geofisika dasar.

Buku ini disusun dengan menambahkan informasi dari berbagai sumber. Diantaranya adalah kode pemrograman yang disusun oleh Agus Abdullah, Ph.D. selama kursus *Python for Geoscientists* yang diselenggarakan oleh *Society of Exploration Geophysicists Student Chapter Universitas Pertamina* (SEGSCUP), tugas yang disusun oleh Mahasiswa Teknik Geofisika Universitas Pertamina angkatan 2016 dan 2017 dalam Mata Kuliah Metode Komputasi, serta forum-forum pengguna Python di dunia maya.

Dengan bahasa yang mudah dipahami, kami berharap agar buku ini dapat digunakan bagi mereka yang ingin mempelajari Python (umumnya) dan aplikasi metode Geofisika dengan menggunakan Python (khususnya), meskipun berawal dari ketidakpahaman terhadap bahasa pemrograman.

Jakarta, 15 Juni 2020

Penulis

Daftar Isi

BAB I Pendahuluan Metode Komputasi	1
Berpikir Komputasional.....	5
Bahasa Pemrograman	6
Terminologi	8
BAB II Dasar-dasar Penggunaan Python	11
Pendahuluan	13
New Project.....	15
Hello World	16
Data Types.....	18
Number Systems.....	21
Interactive Helps	21
Flow Chart.....	22
Data Structures	25
Lists	27
Tuples.....	29
Sets.....	30
Dictionaries	31
Statement and Comment.....	32
Operator.....	33
Operator Aritmatika.....	33
Operator Perbandingan	33
Operator Logika	34
Matrix Operation	35
Control Flow.....	42
Conditional Statement.....	42
For Loop	46
While Loop	52
Break, Continue, Pass.....	54
Looping Techniques	57
File I/O.....	60
Exercise	67
Number Systems.....	70
Array Generator	72
2D Array	75

Integral.....	76
Code Optimization	78
Function	81
Visualisasi	87
Statistics	98
Linear Regression.....	99
Class	101
Random Numbers	102
Convolution.....	104
Ricker Wavelet	105
BAB III Penggunaan Python dalam Komputasi Geofisika	107
Metode <i>Least-Square</i> untuk Menentukan <i>b-value</i>	108
Penentuan Lokasi Gempabumi dengan Metode Tiga Lingkaran	113
<i>Forward Modeling</i> untuk Metode Gayaberat.....	118
<i>Forward Modeling</i> untuk Metode Magnetik	123
<i>Forward Modeling</i> untuk Metode Self-Potential.....	130
Model Pembajian Sintetik (<i>Synthetic Wedge Model</i>)	140
<i>First Break Picking</i>	152
Kecepatan dan Ketebalan Lapisan Gelombang Seismik Refraksi.....	155
<i>Fast Fourier Transform</i>	159
Interpolasi <i>Spline</i> pada Data <i>Gamma-Ray</i>	161
<i>Smoothing 1-D</i> dengan <i>Moving Average Data Gamma Ray</i>	164
Visualisasi Data Seismik	166
<i>Inverse Matrix mxn</i> dengan <i>Singular Value Decomposition</i>	168
Konversi Data <i>Binary</i> ke Format IBM dan IEEE	173
Referensi.....	177

Daftar Gambar

Gambar 1 - Abacus (kiri), Analytical Engine (tengah), Colossus (kanan)	2
Gambar 2 - UNIVAC 1004 (kiri), IBM System 1401 (kanan)	3
Gambar 3 – Fungsi Komputer (Sumber: https://www.slideshare.net/rubyrosedancil1/hardware-79133950)	3
Gambar 4 - High and Low Level Programming Languages (Sumber: http://images.slideplayer.com/16/5222418/slides/slide_3.jpg)	6
Gambar 5 - Top 10 Bahasa Pemrograman Menurut challengerocket.com	7
Gambar 6 - Turunan dari "Constant" (Sumber: https://www.sitesbay.com/cprogramming/c-constant)	8
Gambar 7 - Logo Python	13
Gambar 8 - Tampilan saat browsing Python (versi terakhir Python 3.8.1 pada Februari 2020)	13
Gambar 9 - Logo Pycharm.....	13
Gambar 10 - Penambahan Environment Variables pada System Setting	14
Gambar 11 - Tes command python pada Windows Power Shell.....	14
Gambar 12 - Pembuatan Project pada Python	15
Gambar 13 - Tampilan pembuatan Python File.....	16
Gambar 14 - Tampilan command pertama pada Python file dan cara mengeksekusinya	16
Gambar 15 - Tampilan result dari hasil eksekusi perintah print.....	17
Gambar 16 - Hasil print menggunakan berbagai tipe quote marks.....	17
Gambar 17 – Tipe flow chart	22
Gambar 18 - Contoh Flowchart menentukan angka terkecil dari 3 angka (Sumber: https://www.slideshare.net/mukeshnt/flowcharts-24450634).....	23
Gambar 19 – Flow chart untuk menentkan akar persamaan kuadrat.....	23
Gambar 20 - Ilustrasi perkalian matriks.....	38
Gambar 21 - Diagram alir If statement	42
Gambar 22 - Diagram alir If Else statement.....	43
Gambar 23 - Diagram alir If Elif Else statement.....	44
Gambar 24 – Nested IF statement (Sumber: https://exceljet.net/nested-ifs).....	45
Gambar 25 - Diagram alir For Loop.....	46
Gambar 26 - Diagram alir While Loop.....	52
Gambar 27 – Diagram alir “Break” statement.....	54
Gambar 28 – Diagram alir “Continue” statement	55
Gambar 29 – Diagram alir untuk “loop with condition at the top”, “loop with condition in the middle”, “loop with condition at the bottom”	59
Gambar 30 – Byte pada ASCII (Sumber: https://www.pc当地.com/encyclopedia/term/37184/7-bit-ascii)	60
Gambar 31 – Pembuatan file pada Text Editor.....	61
Gambar 32 – Pembuatan file di Ms. Excel	62
Gambar 33 – Pembuatan file di PyCharm.....	62
Gambar 34 - Logo NumPy	72
Gambar 35 - Langkah menambahkan modul (package) NumPy pada Python	73
Gambar 36 - Logo Matplotlib.....	87
Gambar 37 - Hasil plotting sinusoidal dengan Matplotlib	88
Gambar 38 – Gambar plot untuk inverse axis	89
Gambar 39 – Gambar plot untuk overlay dua kurva	89

Gambar 40 – Gambar plot untuk stem	90
Gambar 41 – Gambar plot untuk menambahkan legend	91
Gambar 42 – Gambar plot untuk menunjukkan area.....	91
Gambar 43 – Gambar plot untuk menampilkan histogram.....	92
Gambar 44 – Gambar subplot (1)	93
Gambar 45 – Gambar subplot (2)	94
Gambar 46 – Gambar plot 3D	94
Gambar 47 – Gambar plot 3D scatter	95
Gambar 48 – Gambar plot 3D scatter	96
Gambar 49 – Gambar plot 3D surface	96
Gambar 50 – Gambar plot 3D contour	97
Gambar 51 – Logo Scikit-Learn	99
Gambar 52 – Regresi linear untuk x dan y (garis) dari data yang diketahui (titik)	100
Gambar 53 - Hasil plotting sinusoidal dengan noise bilangan acak.....	103
Gambar 54 - Contoh Ricker Wavelet	105
Gambar 55 - Ricker Wavelet	106
Gambar 56 - Residual (error) dari true value dan predicted value (Sumber: https://www.mathsisfun.com/data/least-squares-regression.html).....	109
Gambar 57 - Contoh grafik hubungan magnitudo dan frekuensi kejadian gempabumi	110
Gambar 58 - Regresi linier menggunakan Least-Square untuk memperoleh a- dan b-value	112
Gambar 59 - Contoh Rekaman Seismogram untuk Tiga Komponen (Sumber: NMSOP)	113
Gambar 60 - Hubungan antara jarak gempa-stasiun dengan interval S-P (Sumber: https://www.tulane.edu/~sanelson/eens1110/earthint.htm)	114
Gambar 61 - Ilustrasi pertemuan titik dua lingkaran.....	115
Gambar 62 - Metode Tiga Lingkaran Menggunakan Python	117
Gambar 63 - Model sphere dan horizontal cylinder (kiri) dan semi-infinite horizontal sheet (kanan) (El-Tokhey et al, 2015)	119
Gambar 64 - Forward modeling dengan geometri sphere	120
Gambar 65 - Forward modeling dengan geometri horizontal cylinder	121
Gambar 66 - Forward modeling dengan geometri semi-infinite horizontal sheet.....	122
Gambar 67 - Medan magnetik bumi.....	123
Gambar 68 - Model anomali magnetik crustal block (Lowrie, 2007).....	124
Gambar 69 - Distribusi Suseptibilitas Batuan (Sumber: https://gpg.geosci.xyz/content/physical_properties/magnetics_susceptibility_duplicate.html)	125
Gambar 70 - Forward modeling magnetik geometri crustal block	127
Gambar 71 - Model anomali magnetik thin sheet (Rao et al, 1985)	127
Gambar 72 - Forward modeling magnetik geometri thin sheet	128
Gambar 73 - Interpretasi Anomali SP (Reynolds, 1997)	130
Gambar 74 - Anomali SP dengan geometri sphere (A), dipping plate (B) dan dipping rod (C) (Reynolds, 1997)	131
Gambar 75 - Contoh penampang dan peta kontur hasil survei SP pada tubuh sulfida (Telford et al, 1990)	132
Gambar 76 - Perhitungan Nilai Potensial di Suatu Titik (Sumber: http://www.pstcc.edu/departments/natural_behavioral_sciences/Web%20Physics/Chapters%2025%20and%2026.htm).....	132
Gambar 77 - Kiri: Hasil survei SP (Telford et al, 1990), Kanan: Geometri lempeng (Yuliananto & Setyawan, 2015).....	133

Gambar 78 – Anomali SP dengan Geometri Lempeng (Fajriani et al, 2004) dengan inklinasi θ searah jarum jam.....	133
Gambar 79 - Forward Modeling SP dengan Geometri Lempeng Menggunakan Python	136
Gambar 80 – Anomali SP dengan Geometri Bola (A), Silinder horizontal (B) dan Silinder vertikal (C) (Essa, 2019)	136
Gambar 81 - Forward Modeling SP dengan Geometri Bola (kiri) dan Silinder (kanan) Menggunakan Python.....	139
Gambar 82 - Model Pembajian dan Seismik Sintetik (Sumber: https://agilescientific.com/blog/2016/9/15/x-lines-of-python-synthetic-wedge-model).....	140
Gambar 83 – Visualisasi koefiesien refleksi setelah dikonvolusikan dengan Ricker wavelet.....	142
Gambar 84 - Model pembajian.....	144
Gambar 85 - Ricker wavelet untuk wedge model.....	146
Gambar 86 - Seismic trace model pembajian	150
Gambar 87 - Nilai amplitudo maximum sebagai tuning thickness	151
Gambar 88 - First Break Picking	154
Gambar 89 - Hasil first break picking	154
Gambar 90 - Tipe perambatan gelombang seismik (Sumber: http://masw.com/Whatisseismicsurvey.html)	155
Gambar 91 - Shot gather dan perambatan gelombang (Sumber: http://www.enviroscan.com/home/seismic-refraction-versus-reflection)	155
Gambar 92 - Hukum Snell	156
Gambar 93 - Ilustrasi seismik refraksi pada dua lapisan.....	156
Gambar 94 - Ilustrasi seismik refraksi pada beberapa lapisan	157
Gambar 95 - Gradien kecepatan pada grafik time vs offset	157
Gambar 96 – Transformasi Fourier dari Time ke Frequency domain	159
Gambar 97 - Ilustrasi FFT	159
Gambar 98 - Hasil FFT Data Seismik.....	160
Gambar 99 - Interpretasi log Gamma Ray	161
Gambar 100 - Hasil interpolasi spline pada data Gamma Ray.....	163
Gambar 101 - Smoothing 1D Moving Average pada Data Gamma Ray.....	165
Gambar 102 - Pemodelan Sintetik Penampang Seismik.....	166
Gambar 103 - Visualisasi Penampang Data Seismik	167
Gambar 104 - Data seismik dalam format IEEE (kiri), disimpan salah dalam format IBM (tengah), dan setelah gain (kanan).....	173



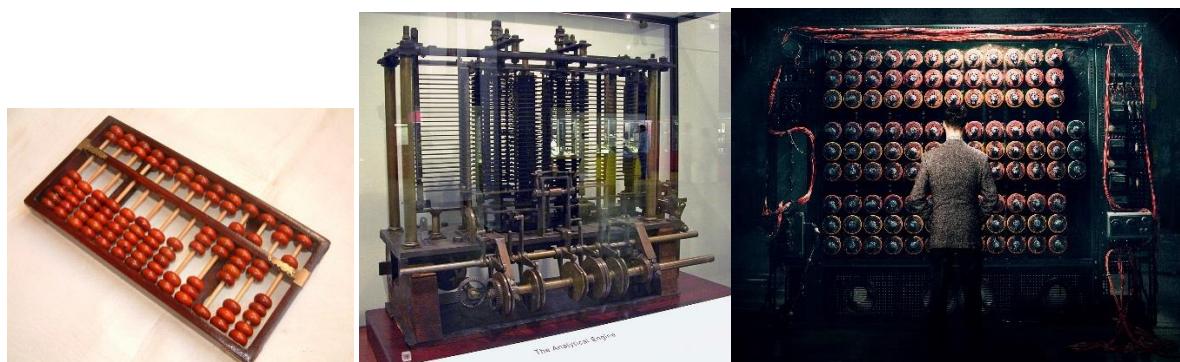
BAB I

Pendahuluan Metode Komputasi

Komputasi merupakan jenis perhitungan dengan menggunakan tahapan aritmetik dan non-aritmetik dan mengikuti suatu model yang sudah didefinisikan, atau algoritma. Studi komputasi berkaitan erat dengan keilmuan komputer.

Komputasi merupakan cara untuk menemukan pemecahan masalah dari data input dengan menggunakan suatu algoritma. Ilmu komputasi adalah bidang ilmu yang mempunyai perhatian pada penyusunan model matematika dan teknik penyelesaian numerik serta penggunaan komputer untuk menganalisis dan memecahkan masalah-masalah ilmu. Dalam penggunaan praktis, biasanya berupa penerapan simulasi komputer atau berbagai bentuk komputasi lainnya untuk menyelesaikan masalah-masalah dalam berbagai bidang keilmuan.

Komputer saat ini sangat dekat dengan manusia, yang idenya sudah berawal dari abad-abad sebelumnya. Awalnya manusia hanya berhitung menggunakan jari dan batu. Kemudian mulai menggunakan **Abacus** sebagai alat hitung, yang digunakan oleh bangsa China dan Babilonia selama ribuan tahun. Pada tahun 1837, Charles Babbage, seorang Matematikawan Inggris mendesain komputer untuk tujuan umum yang disebut dengan **The Analytical Engine**. Kemudian komputer berkembang cepat pada abad 20, diantaranya yang paling dikenal adalah **Colossus** yang dikembangkan oleh Alan Turing pada tahun 1943 (digambarkan dalam film *The Imitation Game* tahun 2014).



Gambar 1 - Abacus (kiri), Analytical Engine (tengah), Colossus (kanan)

Sumber gambar:

Kiri: <https://en.wikipedia.org/wiki/Abacus>

Tengah: https://en.wikipedia.org/wiki/Analytical_Engine

Kanan: <https://www.earthkeptwarm.com/tag/colossus/>

Komputer pertama kali digunakan di Indonesia pada tahun 1956, saat Bank Indonesia menggunakan UNIVAC (*Universal Automatic Computer*) 1004. Pada tahun 1964, TNI Angkatan Darat di Bandung menggunakan IBM System 1401, begitu juga yang digunakan oleh ITB pada tahun 1967.



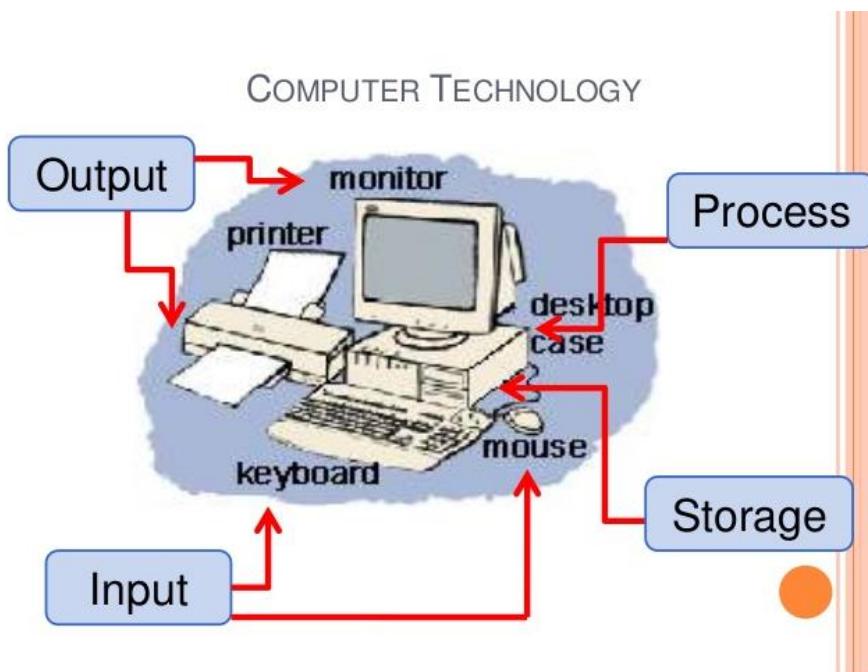
Gambar 2 - UNIVAC 1004 (kiri), IBM System 1401 (kanan)

Sumber gambar:

Kiri: <https://www.ithistory.org/db/hardware/sperry-rand/univac-1004>

Kanan: <https://mikesisco.com/tag/mcb-kaneohe-hawaii/>

Secara garis besar, komputer melakukan 4 (empat) hal, yaitu: (1) menerima input dari dunia luar, (2) memproses informasi, (3) menghasilkan output, (4) menyimpan dan mengambil informasi di tempat penyimpanan (*storage*). Sehingga perangkat keras (*hardware*) komputer pun mengikuti fungsi diatas, diantaranya adalah *input devices*, *output devices*, *microprocessor* atau CPU (*Central Processing Unit*), *Memory* dan *Storage devices*, dan *peripherals*.



Gambar 3 – Fungsi Komputer (Sumber: <https://www.slideshare.net/rubyrosedancil1/hardware-79133950>)

Komputer memiliki berbagai macam tipe, diantaranya:

1. *Personal computer*, dirancang untuk digunakan oleh satu orang dalam satu waktu.
2. *Desktop computer*, memiliki beberapa komponen, yaitu *tower*, *monitor*, *keyboard*, *mouse*, *speaker*.
3. *Workstation*, komputer *desktop high-end* dengan kekuatan komputasi yang besar.
4. *Portable computer*, diantaranya adalah *laptop* atau *notebook* yang didesain *portable*, dan juga *netbook* yang lebih ringan dan untuk keperluan dasar.
5. *Handheld device*, seperti *smartphone*, *personal digital assistants*, *tablet*.
6. *Server*, menyediakan layanan kepada komputer lain yang terkoneksi dalam jaringan. Semua komputer desktop bisa digunakan sebagai server. *Server* mempunyai processor yang lebih cepat, memory yang lebih besar, dan koneksi jaringan yang lebih cepat.
7. *Supercomputer*, umumnya dibangun dari ribuan microprocessor dan menyediakan kemampuan komputasi paling cepat dan dengan kekuatan paling besar.

Berpikir Komputasional

Menurut Jeannette Wing, peneliti komputer dari Columbia University, berpikir komputasional adalah proses pemikiran yang terlibat dalam formulasi masalah, dan mengekspresikan solusinya dalam cara dimana sebuah komputer, manusia atau mesin, bisa menyampaikannya secara efektif.

Berpikir komputasional berbeda dengan *programming*, dimana *programming* merupakan pemikiran yang runut langkah demi langkah, dan menentukan keputusan bila ada dua (atau lebih) kemungkinan yang berbeda. Misalnya, seorang ibu berkata pada anaknya “Nak, tolong pergi ke swalayan dan belikan satu botol susu. Jika dia punya telur, beli enam.” Seorang anak *programmer* akan pulang dengan membawa enam botol susu karena di swalayan dijual telur.

Karakter berpikir komputasional antara lain adalah konseptual, keterampilan fungsional, berpikir sebagaimana manusia berpikir, berpikir **dengan** komputer, bukan berpikir seperti komputer, mengkombinasikan pemikiran matematis dan teknis, dengan produk merupakan ide, bukan artifak.

Hal mendasar dalam komputasi adalah **algoritma**, yaitu suatu set instruksi, yang digunakan untuk menyelesaikan suatu masalah dan melakukan komputasi. Algoritma berasal dari nama Matematikawan Timur Tengah di abad ke-9 Muhammad ibn Musa al-Khwarizmi.

Contoh paling sederhana dalam sebuah algoritma adalah menentukan angka terbesar dari suatu kumpulan angka acak. Deskripsi algoritma dari hal tersebut antara lain:

1. Asumsi angka pertama dalam kumpulan adalah angka terbesar dalam kumpulan tersebut.
2. Setiap data berikutnya, jika angka berikutnya lebih besar dari data sebelumnya, maka angka berikut tersebut menjadi angka terbesar dalam kumpulan.
3. Jika angka berikut lebih kecil, maka angka terbesar tetap angka sebelum.
4. Jika satu kumpulan angka sudah selesai, maka angka terbesar yang tercatat adalah angka yang terbesar dari kumpulan.

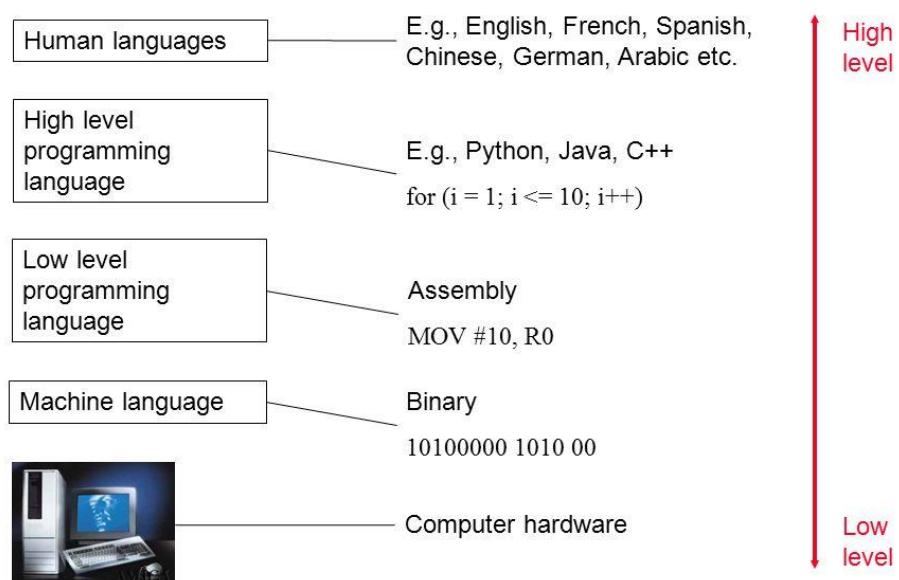
Bahasa Pemrograman

Bahasa pemrograman adalah bahasa formal yang didesain untuk mengekspresikan komputasi. Bahasa formal adalah bahasa yang dirancang untuk tujuan tertentu dan memiliki syntax yang harus diikuti, contohnya adalah bahasa pemrograman, notasi matematika dan kimia.

Ekspresi komputasi dalam bahasa pemrograman terdiri atas:

1. **Input**, yang merupakan data yang diambil dari *file* atau dimasukkan dengan mengetik dengan *keyboard*.
2. **Output**, untuk ditunjukkan pada layar, atau dikirimkan ke *file* atau *device* lain.
3. **Maths**, operasi matematika seperti penambahan dan perkalian.
4. **Conditional logic**, memeriksa kondisi tertentu dan mengeksekusi kode yang pantas.
5. **Repetition**, melakukan tindakan secara berulang, biasanya dengan beberapa variasi.

High Vs. Low Level Languages



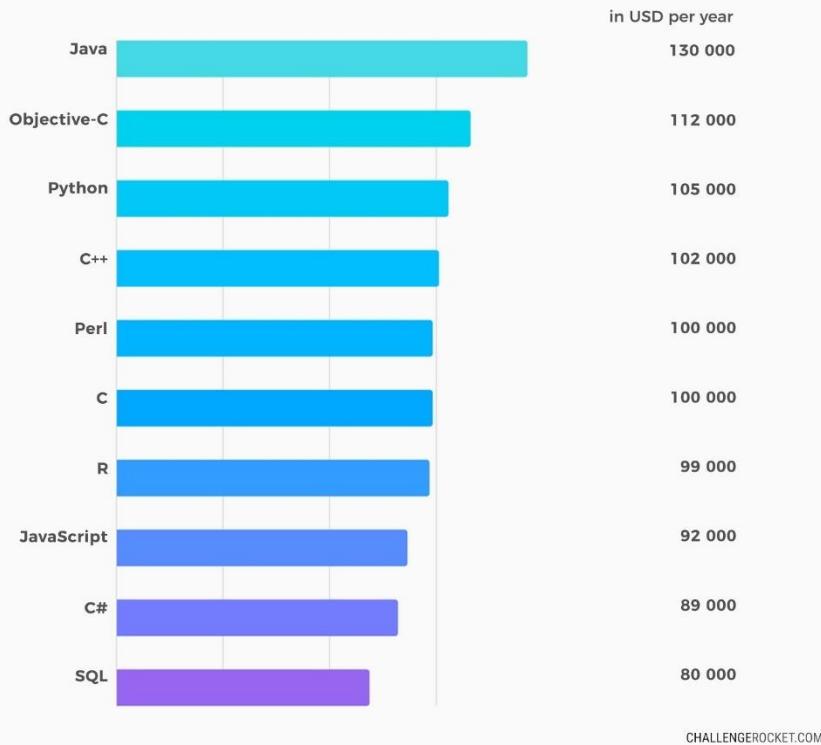
Gambar 4 - High and Low Level Programming Languages (Sumber: http://images.slideplayer.com/16/5222418/slides/slide_3.jpg)

Bahasa pemrograman memiliki level rendah dan tinggi. Bahasa pemrograman yang level rendah adalah dimana syntax yang dimiliki hanya dapat dipahami oleh komputer tertentu, sementara bahasa pemrograman level tinggi memiliki syntax yang *human-friendly*.

Contoh bahasa pemrograman level tinggi adalah Java, C, C++, Python, Perl, Matlab, SQL, Shell, Ruby, Visual Basic, Fortran, Bash Scripting.

TOP 10 CHALLENGEROCKET.COM RANKING

OF PROJECTED EARNINGS IN 2017 BY A PROGRAMMING LANGUAGE



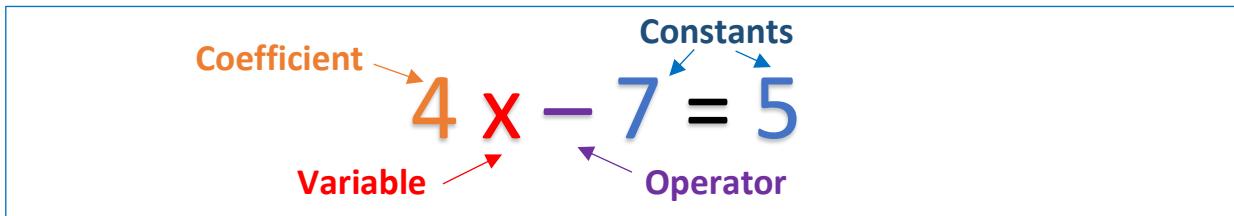
Gambar 5 - Top 10 Bahasa Pemrograman Menurut challengerocket.com

Dalam dunia energi (terutama eksplorasi), data yang dimiliki sangat besar, kemudian kita memerlukan bantuan dalam melakukan *processing*, *modeling*, hingga visualisasi dari data tersebut. Berbagai perangkat lunak (*software*) dikembangkan untuk komputasi Geofisika, diantaranya adalah Omega (Schlumberger), Geovation (CGG), Vista, Seismic Unix, dll.

Perlu untuk memahami perangkat keras yang menyusun sistem kerja komputer, antara lain:

1. *Power Supply*, peralatan elektronik yang menyediakan tenaga listrik. Fungsi utama dari *power supply* adalah untuk mengkonversi arus listrik dari sumber ke beban dengan tegangan, arus dan frekuensi yang tepat.
2. *Motherboard*, menyediakan jalan firman informasi dan tenaga bisa melalui komponen-komponen.
3. *Processor*, sebagai “otak” dari komputer
4. *RAM (Random-Access Memory)*, menyediakan *memory* yang menyimpan sementara instruksi dan data yang diakses oleh *processor*.
5. *Storage Media*, RAM menyimpan sementara, sementara *storage* akan menyimpan selama pekerjaan berlangsung.
6. *Peripherals*, menghubungkan komputer secara eksternal.

Terminologi

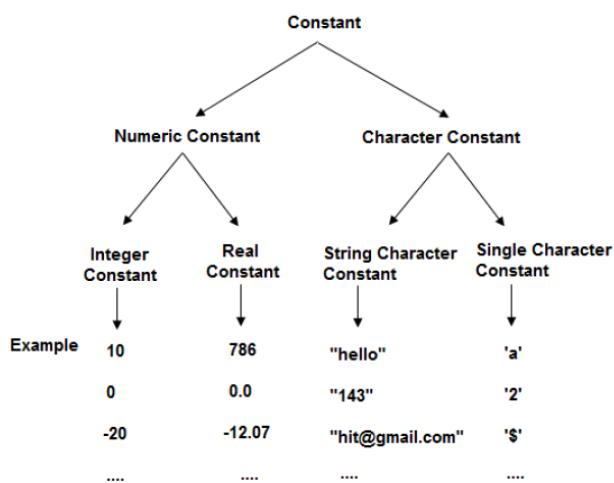


Definisi dari terminologi diatas adalah:

Terminologi	Definisi
<i>Coefficient</i>	Angka yang dikalikan dengan suatu variabel
<i>Variable</i>	Elemen yang dapat diubah dan tidak konsisten
<i>Operator</i>	Karakter yang merepresentasikan suatu tindakan
<i>Constant</i>	Angka yang tidak berubah

Constant terdiri atas:

1. *Numeric constant*, atau angka, yang terdiri atas:
 - a. *Integer constant*, yaitu bulangan bulat seperti 10, 0, -20.
 - b. *Real constant*, yaitu bilangan desimal seperti 786, 0.0, -12.07.
2. *Character constant*, atau huruf, yang terdiri atas:
 - a. *String character constant*, berupa suatu kata atau kalimat, misalnya "hello", "143", hit@gmail.com.
 - b. *Single character constant*, berupa satu huruf saja, misalnya "a", "2", "\$".



Gambar 6 - Turunan dari "Constant" (Sumber: <https://www.sitesbay.com/cprogramming/c-constant>)

Dalam bahasa pemrograman, *Variable* merupakan lokasi memori yang disiapkan untuk menyimpan suatu nilai. Setiap data akan disimpan dalam bentuk tipe data tertentu sesuai dengan yang di-*assign* (tugaskan), apakah *string*, *integer* atau *decimal*.

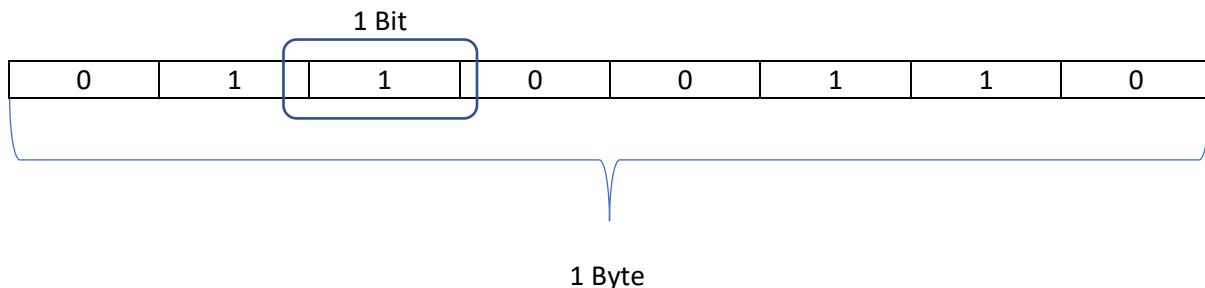
Variable → **a** = Value ← Data type

Ilustrasi diatas menunjukkan bahwa suatu *variable* adalah lokasi memori, dimana **a** merupakan *variable* yang nilainya ditentukan pada ruas kanan persamaan tersebut. Angka **1** bertindak sebagai *value*, sebagai nilai dari *variable a*. Sedangkan *value* tersebut harus berupa data tipe tertentu.

Berikut adalah jenis tipe data (*data type*) dasar dalam bahasa pemrograman:

Data Type	Contoh
<i>integer</i>	1, -30, 0, 205
<i>float</i>	3.98, -2.6788
<i>string</i>	"halo", "baik"
<i>boolean</i>	True, False

Satuan ukuran komputer dinyatakan dalam **Bit** (*Binary digit*) dan juga **Byte**. *Bit* merupakan unit terkecil dari informasi yang dapat diproses komputer, dengan bilangan antara 0 dan 1. Sedangkan *byte* merupakan kumpulan dari 8 bit. 1 kB merupakan 1024 Bytes, 1 MB merupakan 1024 kB, dst.





BAB II

Dasar-dasar Penggunaan Python

Python adalah bahasa pemrograman tingkat tinggi yang mengintegrasikan sistem secara lebih efektif, diciptakan oleh Guido van Rossum (*computer programmer* asal Belanda) dan pertama dirilis pada tahun 1991. Guido van Rossum merupakan penggemar serial komedi BBC 1970-an berjudul “*Monty Python’s Flying Circus*” dan kemudian terinspirasi untuk menamakan bahasa pemrogramannya dengan kata yang singkat, unik, dan sedikit misterius, yang kemudian kita kenal dengan “Python”.

Meskipun sudah beredar selama dua dekade, penggunaan Python pada geosains baru berkembang beberapa tahun terakhir setelah berkembangnya kebutuhan terhadap data yang relatif besar yang berhubungan dengan geosains. Beberapa proyek juga membutuhkan Bahasa pemrograman untuk analisis dan visualisasi, termasuk di dalamnya pengolahan data geofisika.

Python bukanlah satu-satunya Bahasa pemrograman. Kita mengenal beberapa Bahasa pemrograman tingkat tinggi lainnya seperti C++, Visual Basic, Matlab, Fortran, Java, Pascal, dll. Sebagai Bahasa pemrograman tingkat tinggi, Python memiliki kedekatan dengan Bahasa sehari-hari dan dapat membaca data dalam format seperti ASCII (*text*), *binary*, dan XLS/XLSX (Ms. Excel). Salah satu keuntungan menggunakan Python adalah perangkat ini merupakan perangkat *open source (free)*, digunakan dan dikembangkan oleh praktisi sains dan industri, sehingga banyak forum yang membantu pembelajaran Python itu sendiri. Selain alasan *open source*, Python sering digunakan karena membutuhkan waktu yang lebih singkat, memiliki *interconnected tools*, *cross platform* dan *object-oriented programming* (OOP).

Kelebihan dan kekurangan Python lainnya adalah sebagai berikut:

Kelebihan	Kekurangan
<ol style="list-style-type: none">1. Tidak ada deklarasi tipe data yang merumitkan sehingga program menjadi lebih sederhana, singkat, dan fleksibel.2. Tidak ada tahapan kompilasi dan penyambungan (<i>link</i>) sehingga kecepatan perubahan pada masa pembuatan sistem aplikasi meningkat.3. Pemrograman berorientasi objek.4. Manajemen memori otomatis yaitu kumpulan sampah memori sehingga dapat menghindari pencacatan kode.5. Terdapat kelas, modul, eksepsi sehingga terdapat dukungan pemrograman skala besar secara modular.6. Interaktif, dinamis dan alamiah.7. Pemuatan kembali secara dinamis modul phyton seperti memodifikasi aplikasi tanpa menghentikannya.	<ol style="list-style-type: none">1. Beberapa penugasan terdapat diluar dari jangkauan Python, seperti bahasa pemrograman dinamis lainnya, Python tidak secepat atau efisien sebagai statis, tidak seperti bahasa pemrograman kompilasi seperti bahasa C.2. Python tidak dapat digunakan sebagai dasar bahasa pemrograman implementasi untuk beberapa komponen, tetapi dapat bekerja dengan baik sebagai bagian depan skrip antarmuka untuk mereka.3. Disebabkan Python merupakan interpreter, Python bukan merupakan perangkat bantu terbaik untuk pengantar komponen performa kritis.

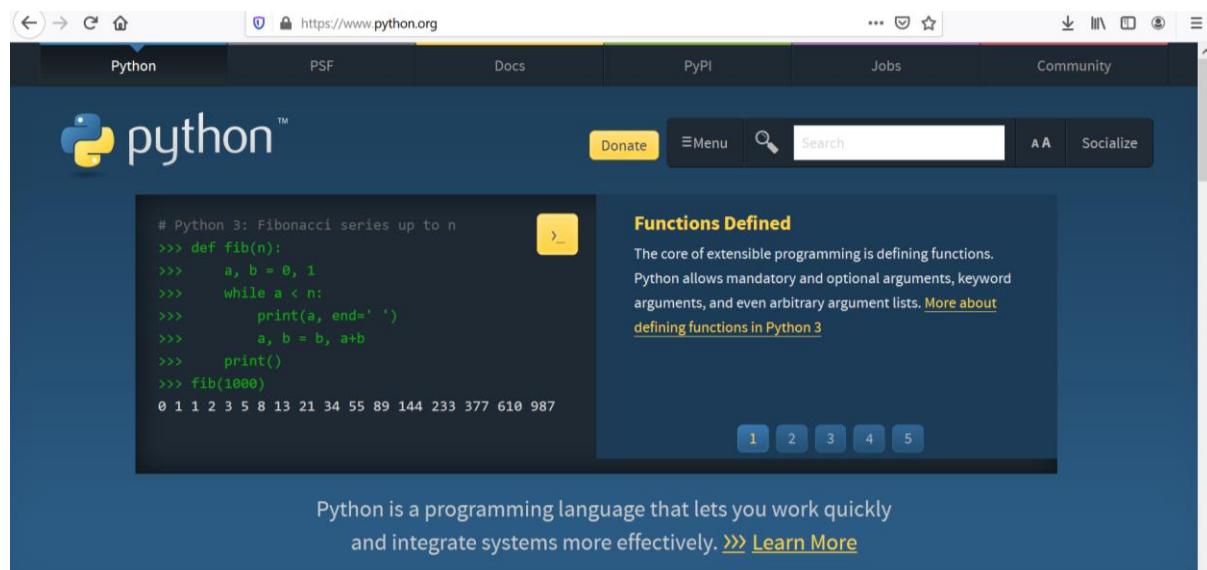


Gambar 7 - Logo Python

Pendahuluan

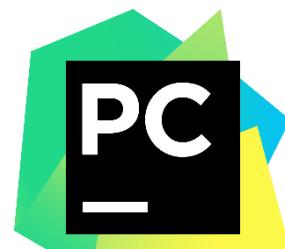
How to install Python on Windows

Python merupakan Bahasa pemrograman yang *open source*, dapat diakses melalui *link* <https://www.python.org>. Python memiliki versi 2.x dan 3.x hingga tahun 2019, dan *compatible* dengan sistem operasi Windows, Linux/UNIX, dan Mac OS X. Adapun versi yang direkomendasikan adalah versi terakhir yang *executable* untuk di-*install*.



Gambar 8 - Tampilan saat browsing Python (versi terakhir Python 3.8.1 pada Februari 2020)

Dalam menjalankan Bahasa pemrograman perlu adanya *Integrated Development Environment* (IDE) untuk memudahkan dalam eksekusi pemrograman. IDE yang dipakai untuk bahasa Python bermacam-macam seperti Spyder, PyCharm, PyDev, Rodeo, Atom, Jupyter Notebook, Eric, dan lain-lain yang juga *cross-platform* dengan Windows, MacOS dan Linux.



Salah satu IDE yang paling sering dipakai untuk Python adalah PyCharm, yang dikembangkan oleh JetBrains, sebuah perusahaan dari Ceko dan dirilis pada tahun 2010. PyCharm memiliki edisi yang gratis yaitu The Community Edition, yang dirilis oleh Apache License, dan juga edisi berbayar (lisensi hak milik) dengan tambahan beberapa fitur yaitu The Professional Edition. PyCharm dapat diakses melalui *link* <https://www.jetbrains.com/pycharm/>. PyCharm Community Edition

Gambar 9 - Logo Pycharm

memiliki fitur-fitur penting seperti Intelligent Editor, Debugger, Refactoring, Inspection, VCS Integration, dan fitur-fitur lainnya.

Berikut adalah beberapa tahapan yang perlu diperhatikan pada saat instalasi Python pada Windows:

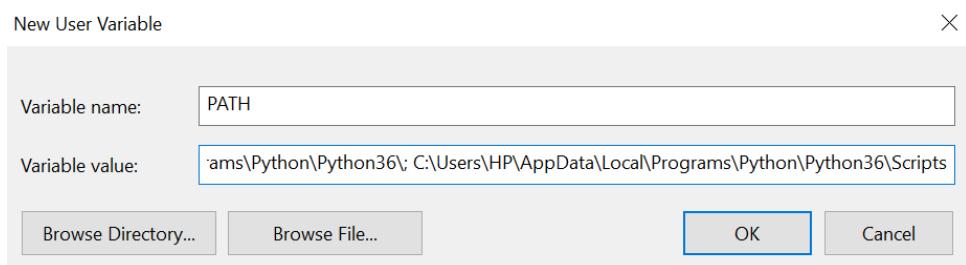
1. ***Environment Variables***

Python dapat diproses di komputer apabila pengaturan Python pada *environment* komputer telah ditambahkan. Pertama, pastikan lokasi dari *executable* program Python, biasanya berada pada **C:\Users*\AppData\Local\Programs\Python\Python36\Scripts**

Kemudian, buka lokasi pengaturan *Environment Variables*, contohnya pada Windows 10 ada pada **Control Panel → System & Security → System → See the name of this computer → Advanced system settings → Advanced → Environment Variables**

Tuliskan:

Variable name: PATH
Variable value: C:\Users*\AppData\Local\Programs\Python\Python36\
C:\Users*\AppData\Local\Programs\Python\Python36\Scripts



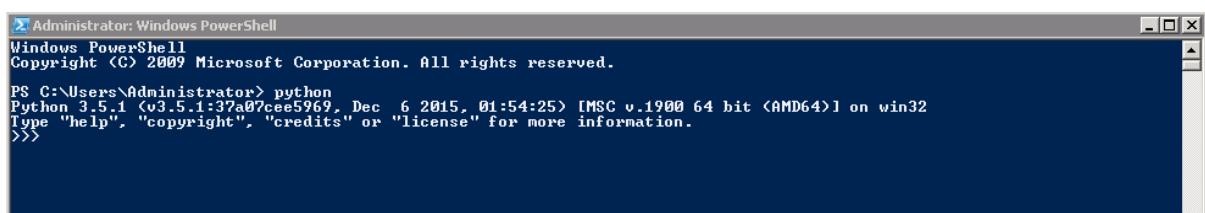
Gambar 10 - Penambahan *Environment Variables* pada System Setting

2. ***Testing***

Sebelum meng-*install* IDE, perlu dipastikan bahwa Python telah ter-*install* dengan baik. Cara memastikannya adalah dengan *testing* pada *command line program* seperti *Command Prompt* atau *Windows Power Shell*. Ketik *Command Prompt* atau *Windows Power Shell* pada *search box* atau klik Windows (⊞)+ R dan ketikkan **cmd**.

Pada *Command Prompt* atau *Windows Power Shell*, ketiklah **python**.

Apabila Python telah ter-*install* dengan baik, maka akan muncul keterangan dari serial Python yang dimiliki.



```
Administrator: Windows PowerShell
Windows PowerShell
Copyright <C> 2009 Microsoft Corporation. All rights reserved.

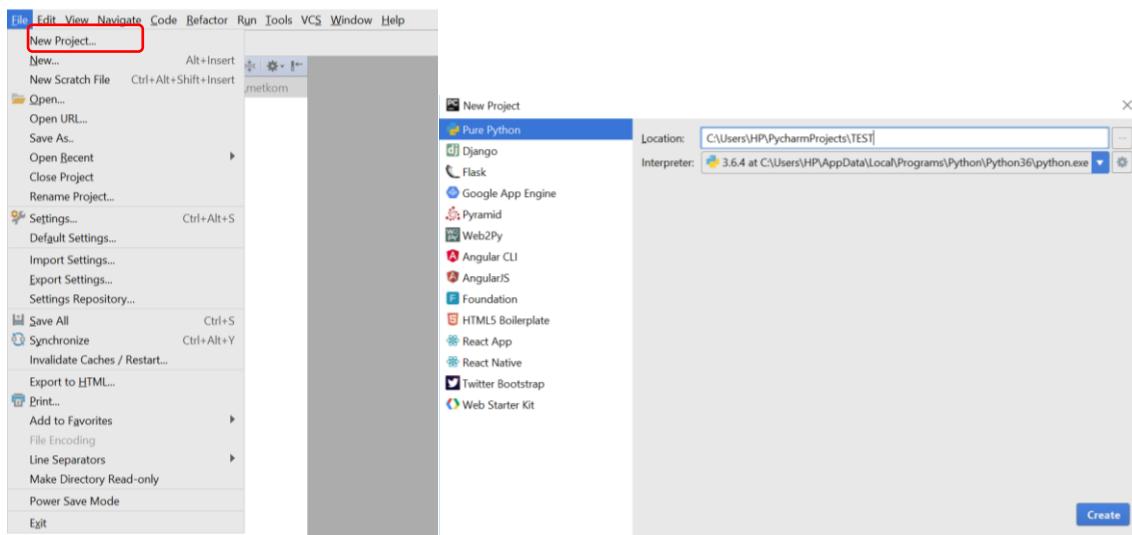
PS C:\Users\Administrator> python
Python 3.5.1 (v3.5.1:37607ce59f9, Dec  6 2015, 01:54:25) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Gambar 11 - Tes command python pada Windows Power Shell

New Project

Setelah meng-*install* PyCharm (Community) sesuai arahan, kemudian muncul *icon* pada *Desktop* sehingga *running program* dapat langsung dilakukan dengan *double click* pada *icon* PyCharm tersebut.

Pertama, buatlah *New Project* pada PyCharm dengan klik **File → New Project** dan kemudian akan muncul *window New Project* yang secara *default* berlokasi di **C:\Users*\PycharmProjects** dan buatlah nama folder *project* yang diinginkan. Misalnya adalah “TEST”. Pastikan *Interpreter* sudah menunjukkan lokasi *executable Python*. Kemudian klik **Create**.



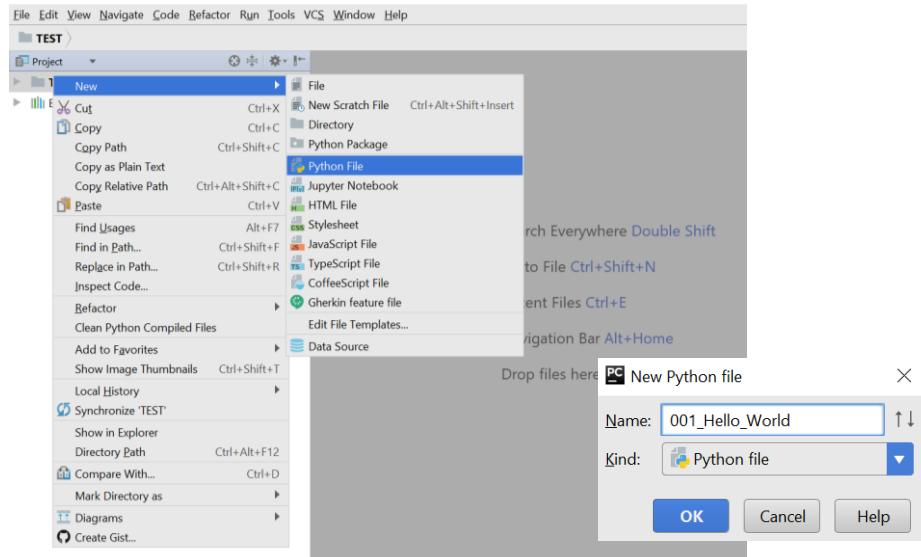
Gambar 12 - Pembuatan Project pada Python

Hello World

Buatlah **Python File** dengan cara Klik kanan pada *Project* kemudian klik **New → Python File**. Buatlah nama *file* sesuai yang diinginkan, dan pastikan tidak ada **spasi** pada nama *file*, gantilah dengan **underscore**.

Kami menyarankan agar lebih ter-*organized* dalam pembuatan *file*, sehingga kita akan membuat *file* secara berurutan dengan memberikan nomor penggerjaan di awal nama *file*.

Misalnya, sebagai *python file* yang pertama, kita akan menamakan *file* **001_hello_world.py**

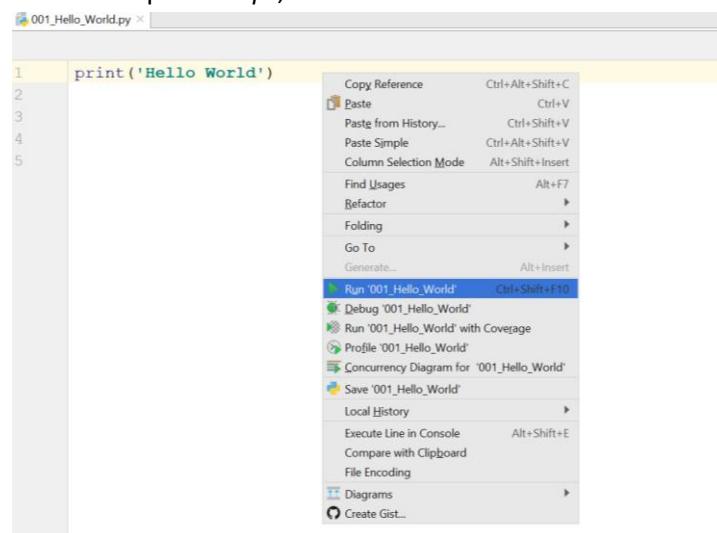


Gambar 13 - Tampilan pembuatan Python File

Kemudian pada *script* tuliskan:

```
print('Hello, world!')
```

Execute dengan cara klik kanan pada *script*, kemudian klik **Run**.



Gambar 14 - Tampilan command pertama pada Python file dan cara mengeksekusinya

The screenshot shows the PyCharm Run tool window. The title bar says "Run 001_Hello_World". The main area displays the command: "C:\Users\HP\AppData\Local\Programs\Python\Python36\python.exe C:/Users/HP/PycharmProjects/TEST/001_Hello_World.py" followed by the output "Hello World". Below the output, it says "Process finished with exit code 0". At the bottom, there is a note: "PEP 8: blank line at end of file".

Gambar 15 - Tampilan result dari hasil eksekusi perintah print

Teks berupa string diikuti dengan tanda kutip satu ' (single quote atau apostrophe) atau tanda kutip dua " (double quote) atau tanda kutip tiga "" (triple quote) tanpa mempengaruhi hasil *printout* dari command **print** tersebut.

The screenshot shows the PyCharm Python Console. It contains the following code and output:

```
>>> print("How're you?")
How're you?
>>> print('''' Susi said:"How're you?"'''')
Susi said:"How're you?"
>>> print('He said: "Hai')
He said: "Hai
>>> print("Levi's")
Levi's
>>> |
```

The console also shows a sidebar titled "Favorites" with a count of 2, and tabs for "TODO" (6 items), "Python Console" (selected), and "Terminal".

Gambar 16 - Hasil print menggunakan berbagai tipe quote marks

Data Types

Tipe data (*data types*) terdiri atas `int`, `float` dan `complex` yang juga disebut sebagai bilangan (*numbers*) dan juga `string`, `boolean`.

<code>string</code>	barisan karakter, ditampilkan dengan perintah <code>print()</code> , diapit oleh <i>single quotation marks</i> ('') atau <i>double quotation marks</i> ("")
<code>int</code>	(integer) merupakan bilangan bulat.
<code>float</code>	merupakan bilangan sebenarnya yang presisi (desimal/pecahan).
<code>complex</code>	merupakan bilangan yang terdiri atas bilangan <i>real</i> dan bilangan <i>imaginary</i> .
<code>boolean</code>	Suatu <i>object</i> diuji apakah betul (True) atau salah (False).

Contoh dari `string`:

```
pesan1 = 'Makan malam yuk.'  
pesan2 = 'Nanti aku tunggu di Restoran ya.'
```

```
pesan1 = 'Makan malam yuk.'  
pesan2 = 'Nanti aku tunggu di restoran ya.'  
print(pesan1)  
print(pesan2)  
  
>>  
Makan malam yuk.  
Nanti aku tunggu di restoran ya.
```

Penggunaan *quotation marks* atau tanda kutip pun mengikuti kaidah yang berlaku. Contoh yang salah adalah sebagai berikut:

```
pesan3 = 'Makannya hari Jum'at aja'  
print(pesan3)
```

Seharusnya dituliskan sebagai berikut:

```
pesan4 = "Makannya hari Jum'at aja"  
print(pesan4)  
  
>>Makannya hari Jum'at aja
```

Berikutnya adalah contoh untuk *numbers*. Misalnya suatu bilangan kompleks (*complex number*) $5.12 + 3.2i$ terdiri atas *real number* 5.12 dan *imaginary number* 3.2. Pada program Python, symbol bilangan imajiner **i** diganti dalam bentuk **j**.

```

a = 5
print('tipe data a:', type(a))
>> tipe data a: <class 'int'>

b = 5.0
print('tipe data b:', type(b))
>> tipe data b: <class 'float'>

c = 2 + 1j
print('tipe data c :', type(c))
>> tipe data c : <class 'complex'>

print('c real :', c.real)
>> c real : 2.0

print('c imag :', c.imag)
>> c imag : 1.0

print('c imag as integer :', int(c.imag))
>> c imag as integer : 1

```

Fungsi matematika pada bilangan yang sederhana, contohnya adalah:

<code>math.ceil</code>	pembulatan ke atas
<code>math.floor</code>	pembulatan ke bawah
<code>round</code>	pembulatan ke integer terdekat

```

import math

print(math.ceil(3.7))
>> 4

print(math.floor(3.7))
>> 3

print(round(3.7))
>> 4

```

Suatu *object* dapat diuji apakah betul atau salah, selain dengan menggunakan `if` dan `while`, dapat juga menggunakan operasi **Boolean**. Syntax dari boolean juga dapat ditulis dengan `bool(value)`. Dimana **False** adalah jika *value* tersebut dihilangkan atau salah, sedangkan **True** adalah jika *value* tersebut benar.

Value yang dianggap **False** pada Python:

- *None*
- *False*
- *Zero of any numeric type. For example, 0, 0.0, 0j*
- *Empty sequence. For example, (), [], ''*

- *Empty mapping. For example, {}*
- *Objects of classes which has `_bool_()` or `_len()` method which returns 0 or False*

Selain `value` diatas dianggap **True**.

```
a = 2
b = 3

print(a != b)
>> True
```

Object diatas menyatakan bahwa `a` tidak sama dengan `b`, sehingga pernyataan `a != b` adalah benar (`True`). Jika dinyatakan dalam bilangan, maka nilai `True = 1`, `False = 0`, yang kemudian dapat dioperasikan dengan bilangan lain.

```
b = 3
c = False
print(c)
>> False

print(int(c))
>> 0

print(b * c)
>> 0
```

Number Systems

Seperti dibahas dalam Bab Pendahuluan sebelumnya, *number systems* terdiri atas *decimal*, *binary*, *octal* dan *hexadecimal*. Berikut adalah contoh *script* mengenai *number systems* tersebut dalam Python:

```
print(0b1101011)
print(0B1101011)
>> 107

print(0o15)
print(0O15)
>> 13

print(0xFB +0b10)
print(0XFB +0B10)
>> 253
```

Perintah diawali dengan angka 0, kemudian dilanjutkan dengan **b** atau **B** untuk *binary*, **o** atau **O** untuk *octal*, dan **x** atau **X** untuk *hexadecimal*. Sedangkan *decimal* adalah angka dengan *base* 10 yang digunakan secara umum, sehingga tidak memerlukan kode untuk memasukkan angka tersebut.

Interactive Helps

Perintah `help()` bersifat interaktif untuk memunculkan keterangan tentang hal-hal yang berhubungan dengan Python.

```
help ('pow')

Help on built-in function pow in module builtins:

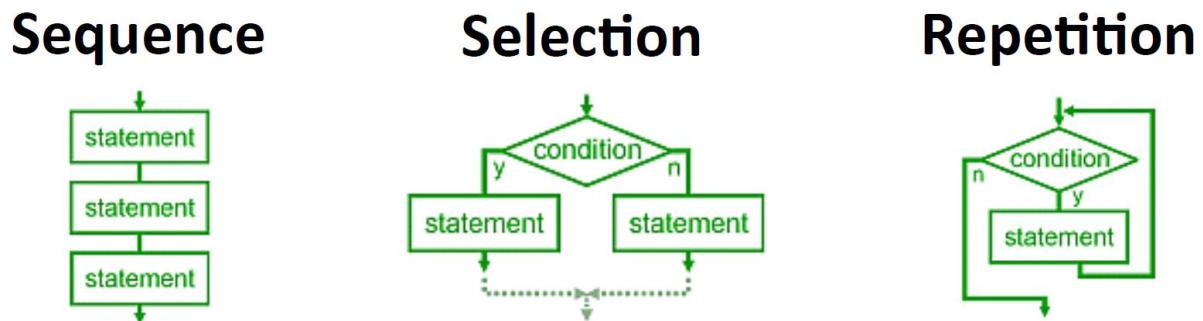
pow(x, y, z=None, /)
    Equivalent to x**y (with two arguments) or x**y % z (with three
    arguments)

    Some types, such as ints, are able to use a more efficient
    algorithm when invoked using the three argument form.
```

Flow Chart

Flow chart atau diagram alir adalah diagram dari suatu urutan pekerjaan seseorang atau sesuatu, yang melibatkan sistem atau aktivitas yang kompleks. Contohnya adalah peta pada Google, yang menunjukkan rute tercepat dengan ETA (*Estimated Time Arrival*) tertentu. Kemudian *Google maps* juga menyediakan rute alternatif lainnya dengan diketahui jarak, rata-rata kecepatan mobil yang disediakan oleh *tracking* pengguna secara *real-time*.

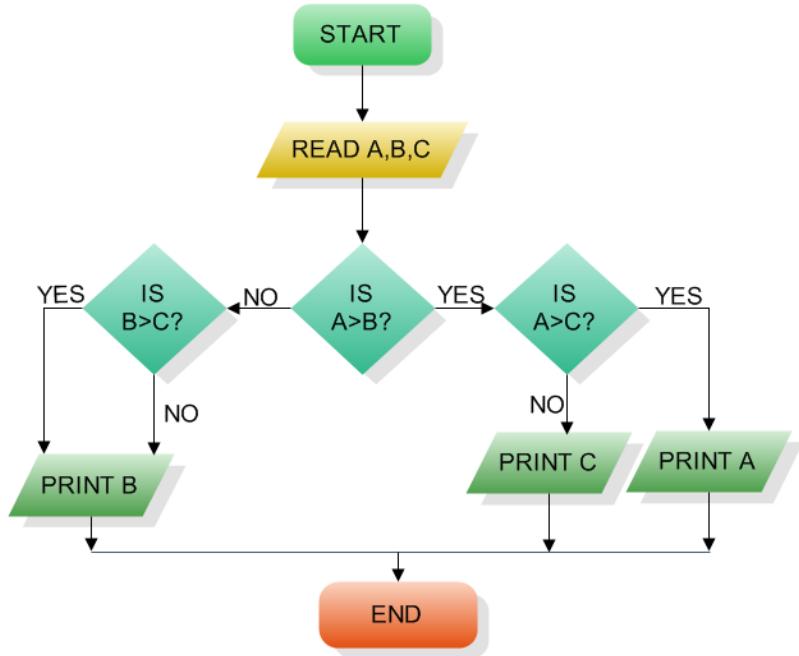
Flow-chart memiliki tiga tipe, yaitu *sequence* atau kejadian secara berurutan, *selection* atau adanya pemilihan karena syarat tertentu, dan *repetition* yaitu perulangan.



Gambar 17 – Tipe flow chart

Diagram alir (*flowchart*) dilambangkan dengan:

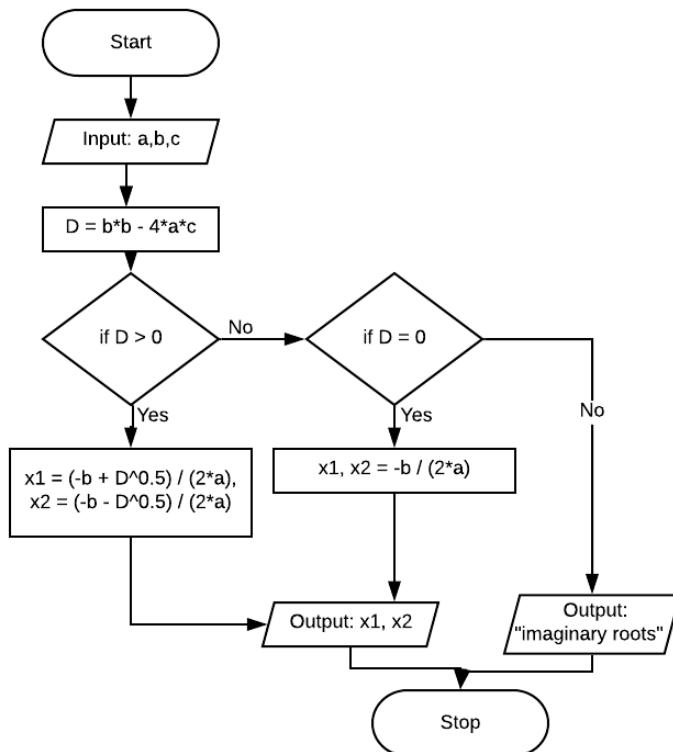
Nama	Simbol	Kegunaan
Start / End	Start / End	Dimana suatu diagram alir dimulai (<i>Start</i>) atau berakhir (<i>End</i>)
Direction/ Flow	→	Menghubungkan simbol, panah menunjukkan arah aliran.
Process	process	Instruksi atau perintah.
Decision	decision	Keputusan, Yes or No.
Input/ Output	Input/output	Input data yang diterima komputer. Output data dikirimkan ke komputer.
Connector		<i>Jump</i> untuk satu titik ke titik lainnya



Gambar 18 - Contoh Flowchart menentukan angka terkecil dari 3 angka (Sumber: <https://www.slideshare.net/mukeshnt/flowcharts-24450634>)

Bagaimana diagram alir untuk menentukan akar persamaan kuadrat?

Jika diketahui persamaan kuadrat $y = ax^2 + bx + c$, maka kita ketahui bahwa hal yang pertama dilakukan adalah menghitung diskriminan dengan persamaan $D = b^2 - 4ac$ sehingga diperoleh akar-akar persamaan kuadrat adalah $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$



Gambar 19 – Flow chart untuk menentkan akar persamaan kuadrat

LATIHAN SOAL.

1. Buatlah *flow chart* untuk menentukan suatu bilangan ganjil atau genap!
2. Buatlah sebuah *flowchart* yang menerima suhu air (dalam derajat *celcius*) dan menuliskan wujud air ke layar sebagai berikut:
 - a. Jika suhu air ≤ 0 derajat, maka tuliskan "beku"
 - b. Jika suhu air > 0 maka tuliskan "cair"
3. Buatlah *flowchart* yang digunakan untuk menuliskan nama-nama bulan dari nomor bulan, yaitu 1 s.d. 12.
 - a. Jika nomor yang diinput bukan antara 1 s.d 12, maka keluarkan pesan "Masukan nomor bulan tidak tepat"
 - b. Setiap angka yang dimasukkan mengeluarkan nama bulan.
4. Buatlah *flowchart* atau diagram alir untuk menentukan jenis segitiga apabila diketahui sisi segitiga berupa a, b, dan c. Jelaskan maksud *flowchart* tersebut!
5. Buatlah *flowchart* atau diagram alir untuk menentukan bilangan ganjil antara 1 sampai 20!

Data Structures

Data structures merupakan cara untuk meng-*organize* atau menyimpan data. Misalkan untuk keterangan berikut:

Fruit = ["apple", "banana", "cherry"]

Fruit merupakan **variable** sebagai nama penyimpan atau yang mendefinisikan suatu nilai (*values*).

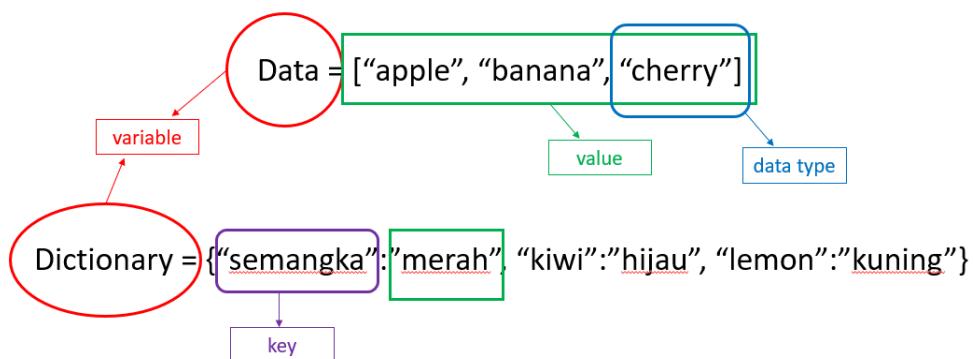
["apple", "banana", "cherry"] merupakan **values** atau makna dari **variable** tersebut.

Isi dari **values** dapat berupa **Data Types** (string, int, float) dan juga *object* lainnya.

Sementara seluruh kesatuan diatas adalah *data structures*.

Data structures pada Python terdiri atas **List**, **Tuple**, **Set** dan **Dictionary** yang akan dijelaskan lebih lanjut. Setiap *data structure* memiliki perbedaan dalam bentuk tanda kurung (*bracket*), apakah elemen di dalam *data structure* tersebut dapat diganti, ditambahkan atau dikurangi isinya (*mutable*), tersusun (*ordered*), ber-index (*indexed*) dan bisa memiliki nilai elemen yang duplikat/sama (*duplicate*).

Array	Bracket	Mutable	Ordered	Indexed	Duplicate
List	[] square	✓	✓	✓	✓
Tuple	() round	-	✓	✓	✓
Set	{ } curly	✓	-	-	-
Dictionary	{ } curly	✓	-	✓	-



Length of array: `len(array)`

```
fruits = ["apple", "mango", "orange"] #list
numbers = (1, 2, 3) #tuple
alphabets = {'a':'apple', 'b':'ball', 'c':'cat'} #dictionary
vowels = {'a', 'e', 'i', 'o', 'u'} #set

print(fruits)
print(numbers)
print(alphabets)
print(vowels)

print(type(fruits))
print(type(numbers))
print(type(alphabets))
print(type(vowels))

>>
['apple', 'mango', 'orange']
(1, 2, 3)
{'a': 'apple', 'b': 'ball', 'c': 'cat'}
{'u', 'e', 'a', 'o', 'i'}
<class 'list'>
<class 'tuple'>
<class 'dict'>
<class 'set'>
```

Lists

Pada prinsipnya, **list** merupakan *array* pada Python. Setiap *array* memiliki dimensi baris dan kolom, yang setiap nilainya dapat ditunjukkan dalam *index*.

Misalkan untuk *array* atau **list** satu dimensi:

Index [1]	Index [2]	Index [3]	Index [4]	Index [5]
--------------	--------------	--------------	--------------	--------------

Namun *index* pertama pada Python adalah 0, bukan 1. Maka pernyataan *array* menjadi:

Index [0]	Index [1]	Index [2]	Index [3]	Index [4]
--------------	--------------	--------------	--------------	--------------

Contohnya untuk *list* berikut:

```
digits = [1, 2, 3]
```

Urutan pertama adalah angka 1, urutan kedua adalah angka 2, dan urutan ketiga adalah angka 3.

Maka penulisan berdasarkan *index* dari setiap elemen *list* diatas adalah:

```
print(digits[0])  
>>1  
  
print(digits[1])  
>>2  
  
print(digits[2])  
>>3
```

Selain dimulai dari 0, *index* juga dapat dibalik dari angka paling terakhir yaitu memiliki *index* [-1] dst.

```
print(digits[-1])  
>>3  
  
print(digits[-2])  
>>2
```

Jika menuliskan dalam *range*, dapat dilakukan dengan cara berikut:

```
print(<variable>[<first_index>:<last_index + 1>])  
  
print(digits[0:2])  
>>[1, 2]
```

Untuk contoh diatas, artinya adalah tuliskan *list* dari index[0] ke index[1]

Namun jika *index* diambil dari angka terakhir, maka cara penulisan menjadi:

```
print(<variable>[<first_index>:<last_index + 1>])
```

```
print(digits[0:-1])
>>[1, 2]
```

Untuk contoh diatas, artinya adalah tuliskan *list* dari index[0] ke index[-2]

Untuk mengetahui jumlah dari elemen di dalam *list*, maka gunakan `len(list)` seperti berikut:

```
print(len(digits))
>>3
```

List mengizinkan terjadi duplikasi nilai elemen, dan juga dapat diubah isinya. Contohnya adalah sebagai berikut:

Contoh untuk duplikasi:

```
thislist = ["apple", "banana", "cherry", "banana"]
print(thislist)
>>['apple', 'banana', 'cherry', 'banana']
```

Contoh untuk perubahan isi:

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"
print(thislist)
>>['apple', 'blackcurrant', 'cherry']

thislist = ["apple", "banana", "cherry"]
thislist.remove("apple")
print(thislist)
>> ['banana', 'cherry']

thislist = ["apple", "banana", "cherry"]
thislist.append('dragonfruit')
print(thislist)
>> ['apple', 'banana', 'cherry', 'dragonfruit']
```

Contoh diatas adalah untuk *array* satu dimensi (*vector*). Untuk *array* dua dimensi (*matrix*) dapat digambarkan sebagai berikut:

	Column 1	Column 2	Column 3
Row 1	arr[0][0]	arr[0][1]	arr[0][2]
Row 2	arr[1][0]	arr[1][1]	arr[1][2]

Tuples

Tuple memiliki kesamaan dengan *list*, kecuali dalam hal bentuk tanda kurung dan, dan *tuple* tidak memiliki kemampuan untuk diubah isinya.

Contoh untuk duplikasi:

```
thistuple = ("apple", "banana", "cherry", "banana")
print(thistuple)
>> ('apple', 'banana', 'cherry', 'banana')
```

Contoh untuk perubahan isi:

```
thistuple = ("apple", "banana", "cherry")
thistuple[1] = "blackcurrant"
print(thistuple)
>> TypeError: 'tuple' object does not support item assignment

thistuple = ("apple", "banana", "cherry")
thistuple.remove("apple")
print(thistuple)
>> 'tuple' object has no attribute 'remove'

thistuple = ("apple", "banana", "cherry")
thistuple.append('dragonfruit')
print(thistuple)
>> 'tuple' object has no attribute 'append'
```

Dalam hal *storage*, *tuple* menyimpan data dalam kapasitas yang lebih sedikit dibandingkan *list*.

```
import sys
a = (1, 3, 4, 5, 6, 6) #tuple
b = [1, 3, 4, 5, 6, 6] #list

print('a:', sys.getsizeof(a))
print('b:', sys.getsizeof(b))
>>
a: 96
b: 112
```

Sets

Set tidak memiliki *order* dan *index*. Sehingga data yang muncul tidak sesuai *order* yang didefinisikan. Isi dari set dapat diubah, namun set hanya memiliki elemen yang unik sehingga tidak menerima duplikasi.

```
iniset = {"satu", "dua", "tiga"}  
print(iniset)  
>> {'satu', 'tiga', 'dua'}
```

Contoh untuk duplikasi:

```
iniset = {"satu", "dua", "tiga", "tiga"}  
print(iniset)  
>> {'tiga', 'dua', 'satu'}
```

Contoh untuk perubahan isi:

```
iniset = {"satu", "dua", "tiga"}  
iniset.remove("satu")  
print(iniset)  
>> {'tiga', 'dua'}  
  
iniset = {"satu", "dua", "tiga"}  
iniset.add("empat")  
print(iniset)  
>> {'dua', 'empat', 'satu', 'tiga'}
```

Note: untuk menambahkan isi dari set tidak menggunakan fungsi *append*, namun dengan fungsi *add*.

Dictionaries

Dictionary merupakan *data structure* yang memiliki *key* dan *value*. Atau sesuatu mendefinisikan sesuatu yang lain. *Dictionary* tidak memiliki urutan, namun memiliki *index* dan dapat diubah. *Dictionary* juga hanya memiliki data yang unik sehingga tidak mengizinkan duplikasi.

```
inidictio = {"semangka": "merah", "kiwi": "hijau", "lemon": "kuning"}
```

Pada *dictionary* diatas berarti adalah, jika disebutkan kata kunci (*key*) semangka, maka nilai (*value*)-nya adalah merah. Jika disebut kiwi, maka nilainya adalah hijau, dan seterusnya. Tidak boleh ada *key* yang sama meskipun ada *value* yang sama.

Berikut adalah contoh penggunaan *dictionary*:

```
inidictio = {"semangka": "merah", "kiwi": "hijau", "lemon": "kuning"}  
print(inidictio["semangka"])  
>>merah
```

Contoh duplikasi *key*:

```
kamus = {"semangka": "merah", "kiwi": "hijau", "lemon": "kuning",  
"semangka": "biru"}  
print(kamus)  
>>{'semangka': 'biru', 'kiwi': 'hijau', 'lemon': 'kuning'}
```

Dictionary akan memunculkan *key* dengan *value* yang terakhir.

Contoh duplikasi *value*:

```
kamus = {"semangka": "kuning", "kiwi": "hijau", "lemon": "kuning"}  
print(kamus)  
>> {'semangka': 'kuning', 'kiwi': 'hijau', 'lemon': 'kuning'}
```

Dictionary tetap memunculkan seluruh elemen meskipun ada *value* yang sama, asalkan *key* berbeda.

Contoh untuk perubahan isi:

```
inidictio = {"semangka": "merah", "kiwi": "hijau", "lemon": "kuning"}  
del inidictio["semangka"]  
print(inidictio)  
>> {'kiwi': 'hijau', 'lemon': 'kuning'}
```



```
inidictio = {"semangka": "merah", "kiwi": "hijau", "lemon": "kuning"}  
inidictio["sirsak"] = "putih"  
print(inidictio)  
>> {'semangka': 'merah', 'kiwi': 'hijau', 'lemon': 'kuning',  
'sirsak': 'putih'}
```

Statement and Comment

Statement merupakan instruksi pada interpreter Python yang bisa dieksekusi.

Assignment statement: `a = 1`

Multi-line statement merupakan *statement* yang di-*extend* menjadi beberapa *line* dengan menggunakan penghubung.

```
x = 1 + 2 + \
    3 + 4
print(x)
```

Line continuation dapat berupa kurung/*parentheses* (), kurung kurawal/*braces* {}, kurung siku/*brackets* []

```
x = (1 + 2 +
      3 + 4)
print(x)
```

Suatu blok kode (*body of function, loop, etc*) dimulai dengan **indentation** dan berakhir dengan dengan suatu *line* tanpa indent (**unindented line**). Jumlah **indentation** tidak terbatas, namun harus konsisten sepanjang blok. Pada umumnya, 4 (empat) **whitespace** dibutuhkan untuk *indentation* dan lebih sering digunakan dibandingkan **tab**.

Python menggunakan **indentation** untuk mengindikasikan suatu blok kode.

```
if 5 > 2:
    print("Five is greater than two!")
```

Python akan mengeluarkan **error** jika tidak menggunakan **indentation**.

```
if 5 > 2:
print("Five is greater than two!")
```

Python memiliki kemampuan untuk komentar, dengan tujuan sebagai dokumentasi di dalam kode.

Comment diawali dengan #, dan seluruh *line* merupakan **comment**.

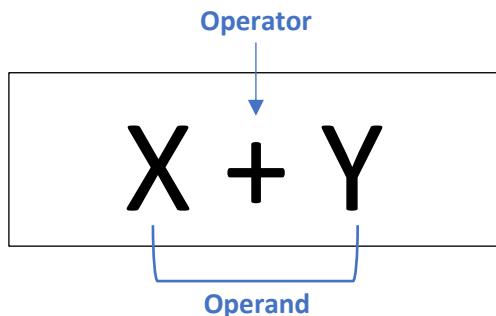
```
#This is a comment
print("Hello World")
```

Multi-line comments memiliki beberapa *line*, dengan menggunakan tagar (#) pada awal setiap *line*.

```
#ini komentar panjang
#dan bersambung
#ke baris berikutnya
```

Operator

Operator merupakan symbol khusus pada Python yang mewakili aritmatika atau komputasi logis. Nilai yang dioperasikan oleh operator disebut **operand**.



Operator Aritmatika

Operator aritmatika digunakan untuk operasi matematika seperti penambahan, pengurangan, perkalian, pembagian, dll.

Operator aritmatika terdiri atas:

Operator	Keterangan	Deskripsi	Contoh (Misalkan a=5, b=3)
+	Addition	Penambahan a dengan b	$a + b = 8$
-	Subtraction	Pengurangan a dengan b	$a - b = 2$
*	Multiplication	Pengalian a dengan b	$a * b = 15$
/	Division	Pembagian a terhadap b	$a / b = 1.667$
**	Power	a pangkat b	$a ** b = 125$
%	Modulus	Sisa dari a dibagi b (dan kelipatan b)	$a \% b = 2$
//	Floor Division	Bilangan bulat terkecil dari a bagi b	$a // b = 1$

Operator Perbandingan

Operator	Keterangan	Deskripsi	Contoh
>	Greater than	<i>True</i> jika operand kiri lebih besar daripada operand kanan	$x > y$
<	Less than	<i>True</i> jika operand kiri lebih kecil daripada operand kanan	$x < y$
==	Equal to	<i>True</i> jika kedua operand bernilai sama	$x == y$
!=	Not equal to	<i>True</i> jika kedua operand tidak bernilai sama	$x != y$
>=	Greater than and equal to	<i>True</i> jika operand kiri lebih besar daripada operand kanan atau bernilai sama	$x >= y$
<=	Less than and equal to	<i>True</i> jika operand kiri lebih kecil daripada operand kanan atau bernilai sama	$x <= y$

```

a = 5.0
b = 3

c = a+b
print(c)

d = a-b
print(d)

e = a*b
print(e)

f = a/b
print(f)

g = a**b
print(g)

h = pow(a,b)
print(h)

i= a%b
print(i)

j= a//b
print(j)

```

```

<<Result>>

8.0

2.0

15.0

1.6666666666666667

125.0

125.0

2.0

1.0

```

Operator Logika

Operator	Deskripsi	Contoh
and	<i>True</i> jika kedua <i>operand</i> benar	x and y
or	<i>True</i> jika salah satu <i>operand</i> benar	x or y
not	<i>True</i> jika <i>operand</i> salah	not x

Matrix Operation

Sebelum masuk ke operasi *matrix*, sebuah *matrix* dapat kita tuliskan dalam ASCII dan dipanggil oleh Python. Misalkan 3 matrix berikut:

$$X = \begin{bmatrix} 1 & 3 \\ 1 & 6 \end{bmatrix} \quad Y = \begin{bmatrix} 2 & 3 \\ 6 & 2 \end{bmatrix} \quad Z = \begin{bmatrix} 8 & 4 \\ 9 & 5 \end{bmatrix}$$

Disimpan dalam ASCII (*Notepad*) dengan nama X adalah matrix1.txt, Y adalah matrix 2.txt, Z adalah matrix3.txt dengan keterangan sebagai berikut:

matrix1.txt	matrix2.txt	matrix3.txt
1 3 1 6	2 3 6 2	8 4 9 5

ASCII akan dipanggil oleh Python dengan perintah `np.genfromtxt()`.

```
import numpy as np
X = np.genfromtxt('matrix1.txt')
Y = np.genfromtxt('matrix2.txt')
Z = np.genfromtxt('matrix3.txt')

print(X)
>>
[[1. 3.]
 [1. 6.]]

print(Y)
>>
[[2. 3.]
 [6. 2.]]

print(Z)
>>
[[8. 4.]
 [9. 5.]]
```

Operasi matrix terdiri atas *scalar product* dan *matrix product*. *Scalar product* merupakan operasi matematika sederhana untuk setiap elemen matrix.

Misalkan untuk matrix X dan Y.

Operation		Scalar Product
Addition	$X + Y = \begin{bmatrix} 1 & 3 \\ 1 & 6 \end{bmatrix} + \begin{bmatrix} 2 & 3 \\ 6 & 2 \end{bmatrix}$	$\begin{bmatrix} 3 & 6 \\ 7 & 8 \end{bmatrix}$
Subtraction	$X - Y = \begin{bmatrix} 1 & 3 \\ 1 & 6 \end{bmatrix} - \begin{bmatrix} 2 & 3 \\ 6 & 2 \end{bmatrix}$	$\begin{bmatrix} -1 & 0 \\ -5 & 4 \end{bmatrix}$
Multiplication	$X * Y = \begin{bmatrix} 1 & 3 \\ 1 & 6 \end{bmatrix} * \begin{bmatrix} 2 & 3 \\ 6 & 2 \end{bmatrix}$	$\begin{bmatrix} 2 & 9 \\ 6 & 12 \end{bmatrix}$
Division	$X / Y = \begin{bmatrix} 1 & 3 \\ 1 & 6 \end{bmatrix} / \begin{bmatrix} 2 & 3 \\ 6 & 2 \end{bmatrix}$	$\begin{bmatrix} 0.5 & 1 \\ 0.16666667 & 3 \end{bmatrix}$

Berikut adalah *script* untuk menghasilkan *scalar product* diatas:

```
print(np.mat(X) + np.mat(Y))
>>
[[3. 6.]
 [7. 8.]]

print(np.mat(X) - np.mat(Y))
>>
[[-1. 0.]
 [-5. 4.]]

print(np.multiply(np.mat(X),np.mat(Y)))
>>
[[ 2. 9.]
 [ 6. 12.]]

print(np.mat(X) / np.mat(Y))
>>
[[0.5 1.]
 [0.16666667 3.]]
```

Contoh lainnya adalah:

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

# scalar product
#matrix operation (addition)
for row in range(0,len(A)):
    for col in range(0, len(A)):
        print(A[row][col] + B[row][col], end=' ')
    print()

#matrix operation (subtraction)
for row in range(0, len(A)):
    for col in range(0, len(A[0])):
        print(A[row][col] - B[row][col], end=' ')
    print()
```

```

# matrix operation (multiplication)
print(len(A))
for row in range(0, len(A)):
    for col in range(0, len(A)):
        print(A[row][col] * B[row][col], end=' ')
    print()

#matrix operation (division)
for row in range(0, len(A)):
    for col in range(0, len(A[0])):
        print(A[row][col] / B[row][col], end=' ')
    print()

#matrix operation (power)
for row in range(0, len(A)):
    for col in range(0, len(A[0])):
        print(A[row][col] ** B[row][col], end=' ')
    print()

#
#matrix operation (modulus)
for row in range(0, len(A)):
    for col in range(0, len(A)):
        print(A[row][col] % B[row][col], end=' ')
    print()

#matrix operation (floor division)
for row in range(0, len(A)):
    for col in range(0, len(A[0])):
        print(A[row][col] // B[row][col], end=' ')
    print()

```

Sementara *matrix product* adalah berupa perkalian matrix, *transpose* dan *inverse*.

Perkalian *matrix* memiliki syarat dimensi Matrix ($u \times v$) dapat dikalikan dengan Matrix ($v \times w$), atau jumlah kolom untuk matrix pertama, sama dengan jumlah baris pada matrix kedua. Maka matrix yang dihasilkan memiliki dimensi Matrix ($u \times w$).

Misalnya, matrix (2x3) dikalikan dengan matrix (3x4) memiliki hasil matrix (2x4).

Sebagai contoh perkalian Matrix M (2x3) dan Matrix A (3x3) berikut:

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \end{bmatrix}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Hasil perkalian dari matrix diatas adalah:

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \end{bmatrix}$$

Dengan penjelasan lebih rinci:

$$c_{11} = (m_{11} \times a_{11}) + (m_{12} \times a_{21}) + (m_{13} \times a_{31})$$

$$c_{12} = (m_{11} \times a_{12}) + (m_{12} \times a_{22}) + (m_{13} \times a_{32})$$

$$c_{13} = (m_{11} \times a_{13}) + (m_{12} \times a_{23}) + (m_{13} \times a_{33})$$

$$c_{21} = (m_{21} \times a_{11}) + (m_{22} \times a_{21}) + (m_{23} \times a_{31})$$

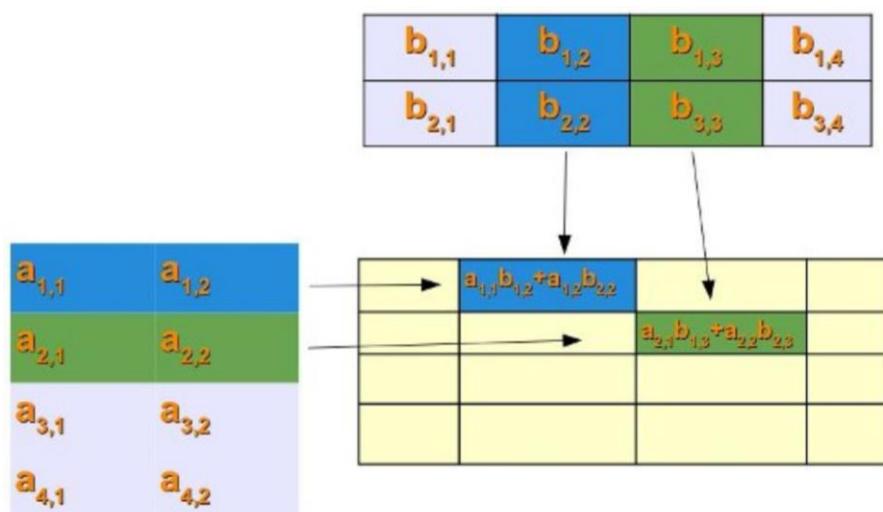
$$c_{22} = (m_{21} \times a_{12}) + (m_{22} \times a_{22}) + (m_{23} \times a_{32})$$

$$c_{23} = (m_{21} \times a_{13}) + (m_{22} \times a_{23}) + (m_{23} \times a_{33})$$

$$c_{ij} = \sum_{k=1}^3 (m_{ik} \times a_{kj})$$

Atau untuk matrix dengan jumlah v = m adalah:

$$c_{ij} = \sum_{k=1}^m (m_{ik} \times a_{kj})$$



Gambar 20 - Ilustrasi perkalian matriks

Operasi perkalian matrix tersebut X dan Y pada Python adalah:

```
import numpy as np
X = np.loadtxt('matrix1.txt')
Y = np.loadtxt('matrix2.txt')

print(X)
>>
[[1. 3.]
 [1. 6.]]

print(Y)
>>
[[2. 3.]
 [6. 2.]]

print(np.mat(X) * np.mat(Y))
>>
[[20. 9.]
 [38. 15.]]
```

Transpose matrix adalah mengubah posisi elemen matrix dari *index* baris menjadi kolom dan sebaliknya. Misalkan pada matrix (3x3) berikut:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Transpose matrix dari A adalah:

$$A^T = \begin{bmatrix} a_{11} & a_{21} & a_{31} \\ a_{12} & a_{22} & a_{32} \\ a_{13} & a_{23} & a_{33} \end{bmatrix}$$

```
import numpy as np
X = np.loadtxt('matrix1.txt')

print(X)
>>
[[1. 3.]
 [1. 6.]]

print(np.mat(np.transpose(X)))
>>
[[1. 1.]
 [3. 6.]]
```

```

import numpy as np
A = [[1,2],[3,4]]

print(A)
At = np.transpose(A)

for row in At:
    for elem in row:
        print(elem, end=' ')
    print()

```

Inverse matrix untuk matrix(2x2) dapat dilakukan dengan langkah berikut:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

$$A^{-1} = \frac{1}{\det(A)} \text{adj}(A)$$

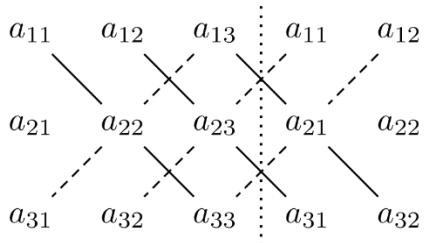
Dengan *determinant* dan *adjoint A* sebagai berikut:

$$\det(A) = ad - bc$$

$$\text{adj}(A) = \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

Namun *determinant A* untuk matrix(3x3) dan seterusnya dilakukan dengan langkah:

$$\det(A) = a_{11}a_{22}a_{33} + a_{12}a_{23}a_{31} + a_{13}a_{21}a_{32} - a_{31}a_{22}a_{13} - a_{32}a_{23}a_{11} - a_{33}a_{21}a_{12}$$



Sarrus's rule: Determinan dari tiga kolom di sebelah kiri adalah penjumlahan dari produk diagonal garis utuh dikurangi produk diagonal garis putus-putus.

Kemudian, sebelum mencari *adjoint (A)*, terlebih dahulu dicari *cofactor* dari A. *Cofactor* diperoleh dari perkalian diagonal antara elemen yang tidak memiliki *index* baris ataupun kolom yang sama dengan *index* elemen *cofactor*. Kemudian tanda positif dan negatif menyesuaikan lokasi *index* dimana berselingan dimulai dari positif dari elemen dengan barisan teratas.

$$\begin{array}{c}
 \left[\begin{array}{ccc} 0 & 0 & 2 \\ 2 & 0 & -2 \\ 0 & 1 & 1 \end{array} \right] \quad 0 \times 1 - (-2) \times 1 = 2 \\
 \left[\begin{array}{ccc} 3 & 0 & 2 \\ 2 & 0 & -2 \\ 0 & 1 & 1 \end{array} \right] \quad 2 \times 1 - (-2) \times 0 = 2 \\
 \dots \\
 \left[\begin{array}{ccc} 3 & 0 & 2 \\ 2 & 0 & -2 \\ 0 & 0 & 1 \end{array} \right] \quad 3 \times -2 - 2 \times 2 = -10 \\
 \left[\begin{array}{ccc} 3 & 0 & 2 \\ 2 & 0 & -2 \\ 0 & 1 & 1 \end{array} \right] \quad 3 \times 0 - 0 \times 2 = 0
 \end{array}$$

$$C = \left(\begin{array}{ccc}
 + \left| \begin{array}{cc} a_{22} & a_{23} \\ a_{32} & a_{33} \end{array} \right| & - \left| \begin{array}{cc} a_{21} & a_{23} \\ a_{31} & a_{33} \end{array} \right| & + \left| \begin{array}{cc} a_{21} & a_{22} \\ a_{31} & a_{32} \end{array} \right| \\
 - \left| \begin{array}{cc} a_{12} & a_{13} \\ a_{32} & a_{33} \end{array} \right| & + \left| \begin{array}{cc} a_{11} & a_{13} \\ a_{31} & a_{33} \end{array} \right| & - \left| \begin{array}{cc} a_{11} & a_{12} \\ a_{31} & a_{32} \end{array} \right| \\
 + \left| \begin{array}{cc} a_{12} & a_{13} \\ a_{22} & a_{23} \end{array} \right| & - \left| \begin{array}{cc} a_{11} & a_{13} \\ a_{21} & a_{23} \end{array} \right| & + \left| \begin{array}{cc} a_{11} & a_{12} \\ a_{21} & a_{22} \end{array} \right|
 \end{array} \right)$$

Adjoint dari A merupakan transpose dari matrix Cofactor (A) tersebut.

$$\text{adj}(A) = C^T = \left(\begin{array}{ccc}
 + \left| \begin{array}{cc} a_{22} & a_{23} \\ a_{32} & a_{33} \end{array} \right| & - \left| \begin{array}{cc} a_{12} & a_{13} \\ a_{32} & a_{33} \end{array} \right| & + \left| \begin{array}{cc} a_{12} & a_{13} \\ a_{22} & a_{23} \end{array} \right| \\
 - \left| \begin{array}{cc} a_{21} & a_{23} \\ a_{31} & a_{33} \end{array} \right| & + \left| \begin{array}{cc} a_{11} & a_{13} \\ a_{31} & a_{33} \end{array} \right| & - \left| \begin{array}{cc} a_{11} & a_{13} \\ a_{21} & a_{23} \end{array} \right| \\
 + \left| \begin{array}{cc} a_{21} & a_{22} \\ a_{31} & a_{32} \end{array} \right| & - \left| \begin{array}{cc} a_{11} & a_{12} \\ a_{31} & a_{32} \end{array} \right| & + \left| \begin{array}{cc} a_{11} & a_{12} \\ a_{21} & a_{22} \end{array} \right|
 \end{array} \right)$$

Setelah memiliki determinant dan adjoint, baru kemudian diperoleh inverse matrix dari matrix A.

$$A^{-1} = \frac{1}{\det(A)} \text{adj}(A)$$

Operasi inverse matrix pada Python menggunakan modul Numpy LinAlg (Linear Algebra) dengan fungsi *inverse* (hanya untuk singular matrix)

```

import numpy as np
X = np.loadtxt('matrix1.txt')

print(X)
>>
[[1. 3.]
 [1. 6.]]

from numpy.linalg import inv
print(inv(np.mat(X)))
>>
[[ 2.        -1.        ]
 [-0.33333333  0.33333333]]

```

	Operation	Matrix Product
Matrix Multiplication	$X * Y = \begin{bmatrix} 1 & 3 \\ 1 & 6 \end{bmatrix} * \begin{bmatrix} 2 & 3 \\ 6 & 2 \end{bmatrix}$	$\begin{bmatrix} 20 & 9 \\ 38 & 15 \end{bmatrix}$
Transpose Matrix	$X = \begin{bmatrix} 1 & 3 \\ 1 & 6 \end{bmatrix}$	$\begin{bmatrix} 1 & 1 \\ 3 & 6 \end{bmatrix}$
Inverse Matrix	$X = \begin{bmatrix} 1 & 3 \\ 1 & 6 \end{bmatrix}$	$\begin{bmatrix} -2.0 & -1.0 \\ -0.333 & 0.333 \end{bmatrix}$

Control Flow

Repetition atau pengulangan berarti melakukan suatu instruksi bahkan aksi secara berulang-ulang. Komputer memiliki performansi yang sama. Manusia punya kecenderungan untuk melakukan kesalahan (karena letih atau bosan). Program dimaksudkan untuk mengulang pernyataan beberapa kali, mengeksusi suatu *set* dan setiap *item*-nya, dan memulai lagi dari awal hingga akhir suatu barisan pernyataan.

Elemen suatu set pemrograman terdiri atas:

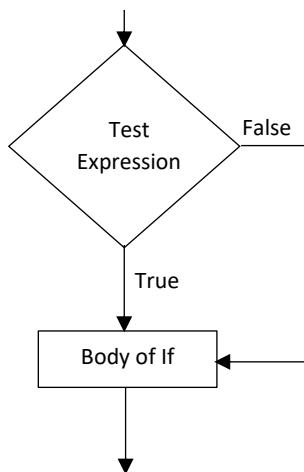
- *Conditional statement*: ekspresi logis untuk memeriksa apakah program dilanjutkan atau tidak.
Pada Python, *conditional statement* dinyatakan dalam **if elif else**
- *Body loop*: blok kode tertentu yang berulang
Pada Python, *loop* terdiri atas **for, while, break, continue, pass**

Conditional Statement

Jika suatu hal membutuhkan syarat tertentu untuk dipenuhi, maka diperlukan **if.. else** dalam pengambilan keputusan tersebut.

Syntax untuk *if statement*

```
if test expression:  
    statement(s)
```



Gambar 21 - Diagram alir If statement

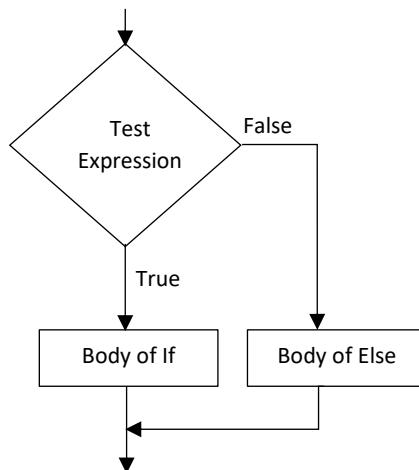
Contoh:

```
num = -2  
if num > 0:  
    print(num, "adalah bilangan positif.")
```

`if.. else` mengevaluasi **test expression** dan hanya mengeksekusi **Body of if** apabila syaratnya terpenuhi (True). Apabila syarat tidak terpenuhi (False) maka yang dieksekusi adalah **Body of else**. *Indentation* digunakan untuk memisahkan setiap *block*.

Syntax untuk *if.. else statement*

```
if test expression:  
    Body of if  
else:  
    Body of else
```



Gambar 22 - Diagram alir If Else statement

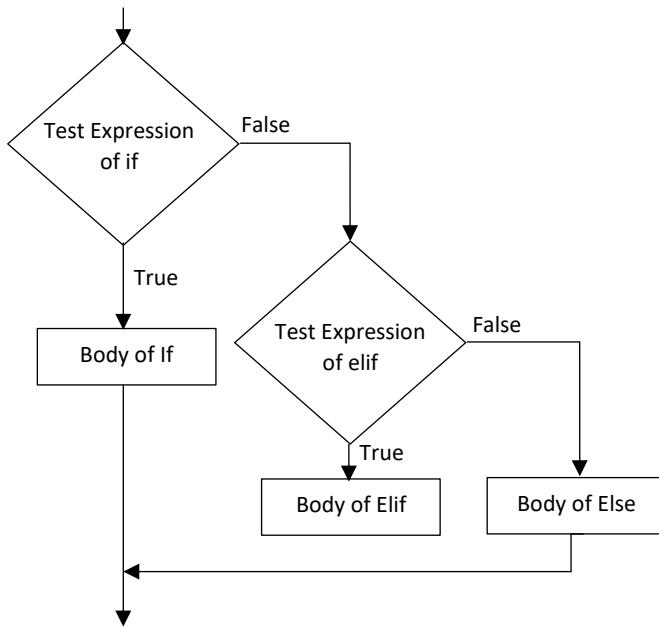
Contoh:

```
num = -1  
if num > 0:  
    print(num, "adalah bilangan positif.")  
else:  
    print(num, "adalah bilangan negatif")
```

Jika ada beberapa syarat, maka gunakan **elif** atau singkatan dari **else if**. Jika suatu syarat untuk *if* adalah *False*, maka gunakan syarat lain dengan *block elif* dan seterusnya. Jika seluruh syarat tidak terpenuhi (*False*), maka **Body of else** yang akan terpenuhi.

Syntax untuk *if.. elif.. else statement*

```
if test expression:  
    Body of if  
elif test expression:  
    Body of elif  
else:  
    Body of else
```



Gambar 23 - Diagram alir If Elif Else statement

Contoh:

```

num = -2
if num > 0:
    print(num, "adalah bilangan positif.")
elif num == 0:
    print(num, "adalah nol.")
else:
    print(num,"adalah bilangan negatif")

```

Apabila *print* berada pada posisi *unindent* setelah if statement, maka perintah tersebut akan dieksekusi sebagaimana else.

```

if test expression:
    Body of if
Body of else

```

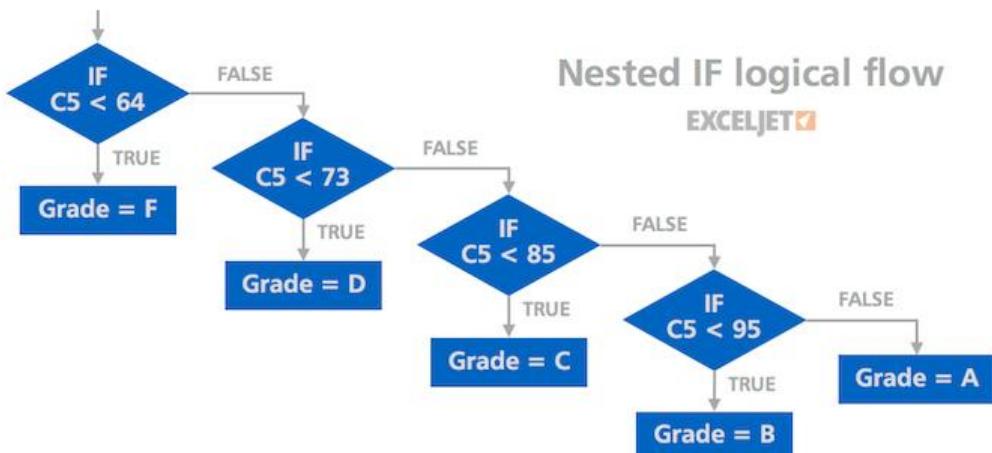
Sebagai contoh:

```

if 'foo' in ['bar', 'bax', 'qux']:
    print('Expression was true')
print('Expression was false')
>>
Expression was false

```

If elif else statement dapat berada di dalam if elif else statement yang lain. Dalam pemrograman komputer, hal ini disebut dengan **nesting**.



Gambar 24 – Nested IF statement (Sumber: <https://exceljet.net/nested-ifs>)

Contoh:

Point	Grade
95 - 100	A
85 – 94	B
74 – 85	C
64 – 73	D
0 – 63	F

```
c5 = 80
if c5 < 64:
    print("F")
elif c5 < 73:
    print("D")
elif c5 < 85:
    print("C")
elif c5 < 95:
    print("B")
else:
    print("A")
```

LATIHAN SOAL.

Buatlah pemrograman Python menggunakan if elif else statement untuk:

- Menentukan tipe segitiga
- Menentukan apakah suatu angka negatif, positif ganjil atau positif genap

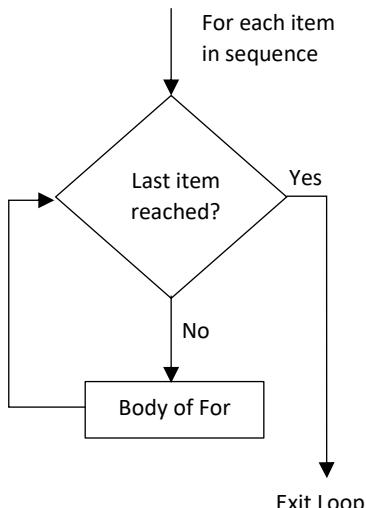
For Loop

For Loop pada Python digunakan untuk mengiterasi suatu *sequence* (urutan) dari data berupa **list**, **tuple**, **string** atau **object** iteratif lainnya.

Syntax untuk *for loop*

```
for val in sequence:  
    Body of for
```

Pada pernyataan diatas, **val** merupakan variabel yang memiliki *value* dalam **sequence** untuk setiap iterasi. *Loop* akan berlanjut hingga mencapai elemen terakhir pada **sequence**. Sementara **Body of for** pada *loop* diletakkan terpisah dengan menggunakan *indentation*.



Gambar 25 - Diagram alir For Loop

Sebuah **sequence** dapat dibuat dengan menggunakan fungsi **range()** dimana **range(10)** akan membuat angka 0 sampai 9. *Range* juga dapat dibuat dengan mendefinisikan dimulai dari angka berapa (**start**), selesai di angka berapa (**stop**), dan berapa angka yang memisahkan setiap urutan (**step**).

Perhatikan bahwa **sequence** akan berhenti pada angka **stop - 1**. Secara *default*, jika tidak diberikan definisi, maka **step = 1**. Fungsi tersebut tidak akan menyimpan *value* pada *memory* sehingga lebih efisien. Untuk memunculkan *output* dari fungsi tersebut, maka gunakan **list()**.

```
for i in range(1,10,1):
```

```
for <variable> in range(<start>,<stop>,<step>):
```

For Loop juga dapat digunakan dengan opsi **else** (jika ada syarat lain dari *object*), **break** (untuk menghentikan *loop* saat bertemu dengan syarat tertentu) dan juga **continue** (untuk meneruskan *loop* jika menemukan syarat tertentu).

```
for i in range(10):
    print(i)

>>
0
1
2
3
4
5
6
7
8
9

for j in range(1,10,1):
    print(j)
>>
1
2
3
4
5
6
7
8
9
```

```
a = [1,2,3,6,7,8]

for i in a:
    print(i*5)
>>
5
10
15
30
35
40
```

Contoh untuk **else**:

```
digits = [0, 1, 5]

for i in digits:
    print(i)
else:
    print("No items left.")
>>
0
1
5
No items left.
```

Note: Unindent perintah `print` merupakan perintah `else: print`.

Nested loop adalah *loop* di dalam *loop*. **Inner loop** akan dieksekusi satu kali setiap iterasi **outer loop**.

```
adj = ["red", "sweet", "round"]
fruits = ["apple", "banana", "cherry"]
for x in adj:
    for y in fruits:
        print(x,y)
```

Pada kasus diatas, *inner loop* y akan dieksekusi untuk setiap *outer loop* x. Maka hasil *output* dari *script* diatas adalah:

```
>>
red apple
red banana
red cherry
sweet apple
sweet banana
sweet cherry
round apple
round banana
round cherry
```

Bagaimana pemrograman pada lagu anak-anak berikut?

Anak ayam turun lah 4, mati satu tinggal lah 3
Anak ayam turun lah 3, mati satu tinggal lah 2
Anak ayam turun lah 2, mati satu tinggal lah 1
Anak ayam turun lah 1, mati satu tinggal induknya
Tekotek koteck koteck, anak ayam turun berkotek
Tekotek koteck koteck, anak ayam turun berkotek

```

for i in range(4,0,-1):
    if i > 1:
        print('Anak ayam turun lah',i,'mati satu tinggal lah',i-1)
    else:
        print('Anak ayam turun lah',i,'mati satu tinggal
induknya')

for i in range(2):
    print('Tekotek kotek kotek, anak ayam turun berkotek')

```

kepala pundak lutut kaki lutut kaki
 kepala pundak lutut kaki lutut kaki
 mata telinga mulut hidung dan pipi
 kepala pundak lutut kaki lutut kaki

```

a = 'kepala'
b = 'pundak'
c = 'lutut'
d = 'kaki'

for i in range(4):
    if i != 2:
        print(a,b,end=' ')
        for j in range(2):
            print(c,d,end=' ')
        print()
    else:
        print('mata telinga mulut hidung dan pipi')

```

LATIHAN SOAL.

Buatlah pemrograman Python untuk lagu anak-anak berikut:

TEN IN A BED	
There were 10 in a bed, and the little one said “Roll over! Roll over!” So they all rolled over and one fell out 9.	There were 5 in a bed, and the little one said “Roll over! Roll over!” So they all rolled over and one fell out 4.
There were 9 in a bed, and the little one said “Roll over! Roll over!” So they all rolled over and one fell out 8.	There were 4 in a bed, and the little one said “Roll over! Roll over!” So they all rolled over and one fell out 3.
There were 8 in a bed, and the little one said “Roll over! Roll over!” So they all rolled over and one fell out 7.	There were 3 in a bed, and the little one said “Roll over! Roll over!” So they all rolled over and one fell out 2.
There were 7 in a bed, and the little one said “Roll over! Roll over!” So they all rolled over and one fell out 6.	There were 2 in a bed, and the little one said “Roll over! Roll over!” So they all rolled over and one fell out 1.
There were 6 in a bed, and the little one said “Roll over! Roll over!” So they all rolled over and one fell out 5.	There was 1 in a bed, and the little one said “I’m sooo sleepy!”

BABY SHARK

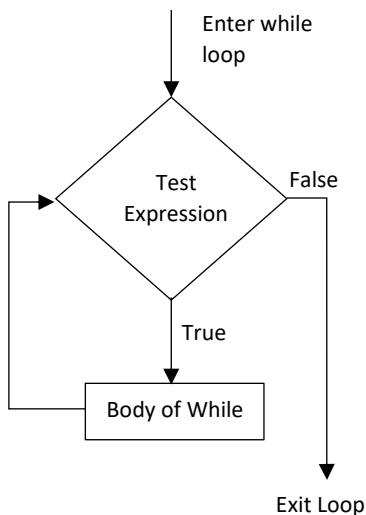
Baby shark doo doo doo doo doo doo Baby shark doo doo doo doo doo doo Baby shark doo doo doo doo doo doo Baby shark	Let's go hunt doo doo doo doo doo doo Let's go hunt doo doo doo doo doo doo Let's go hunt doo doo doo doo doo doo Let's go hunt
Mommy shark doo doo doo doo doo doo Mommy shark doo doo doo doo doo doo Mommy shark doo doo doo doo doo doo Mommy shark	Run away doo doo doo doo doo doo Run away doo doo doo doo doo doo Run away doo doo doo doo doo doo Run away
Daddy shark doo doo doo doo doo doo Daddy shark doo doo doo doo doo doo Daddy shark doo doo doo doo doo doo Daddy shark	Safe at last doo doo doo doo doo doo Safe at last doo doo doo doo doo doo Safe at last doo doo doo doo doo doo Safe at last
Grandma shark doo doo doo doo doo doo Grandma shark doo doo doo doo doo doo Grandma shark doo doo doo doo doo doo Grandma shark	It's the end doo doo doo doo doo doo It's the end doo doo doo doo doo doo It's the end doo doo doo doo doo doo It's the end
Grandpa shark doo doo doo doo doo doo Grandpa shark doo doo doo doo doo doo Grandpa shark doo doo doo doo doo doo Grandpa shark	

While Loop

While Loop pada Python digunakan untuk mengiterasi suatu *block* kode apabila syarat dari suatu *test expression* terpenuhi (True). *While loop* digunakan disaat kita tidak mengetahui berapa jumlah iterasi.

Syntax untuk *for loop*

```
while test expression:  
    Body of while
```



Gambar 26 - Diagram alir While Loop

Seperti *for loop*, *while loop* juga memiliki opsi **else**, **break**, **continue**. Pada *while loop*, suatu *variable* harus didefinisikan terlebih dahulu sebelum *block while loop*, kemudian iterasi mengikuti fungsi yang didefinisikan dari *variable* tersebut.

Sebagai contoh, apabila kita ingin membuat *sequence* 1, 2, 3 pada *for loop* adalah dengan menuliskan

```
for i in range(1, 4, 1):  
  
    for <variable> in range(<start>, <stop>, <step>):
```

Stop adalah angka akhir yang diinginkan **ditambah 1**.

Namun dalam *while loop*, cara penulisannya adalah sebagai berikut:

```
i=0  
while i <=2:  
    i=i+1  
    print(i)
```

<variable>=<start>
`while <variable> <= <stop>:
 i=i+1
 print(i)`

Stop adalah angka akhir yang diinginkan **dikurangi 1**.

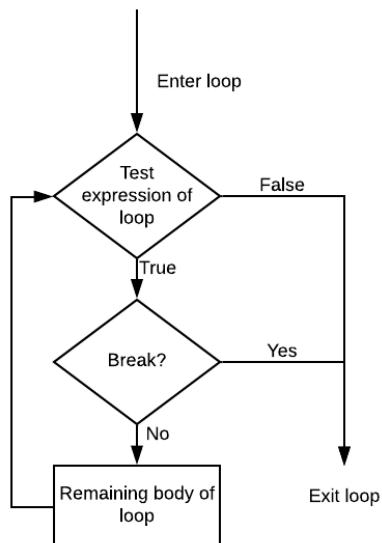
Contoh untuk **else**:

```
n = 3
while n != 0:
    print(n)
    n -= 1
else:
    print("selesai")
>>
3
2
1
selesai
```

Break, Continue, Pass

Pada Python, **break** dan **continue statement** mengubah aliran dari loop normal. Loop mengiterasi suatu blok kode hingga ekspresi tes False, namun kadang pada iterasi tertentu perlu adanya penghentian bahkan pada seluruh loop tanpa memeriksa ekspresi tes. Pada pemrograman Python, **pass** merupakan *null statement* (pembatalan). Perbedaan antara *comment* dan *pass* pada Python, adalah interpreter mengabaikan seluruh *comment*, namun tidak untuk *pass*.

Break statement menghentikan loop. Kontrol program mengalirkan *statement* segera setelah *body* dari *loop*. Jika **break statement** di dalam *nested loop* (loop di dalam loop), **break** akan menghentikan *inner loop* terdalam.



Gambar 27 – Diagram alir “Break” statement

Contoh untuk **break**:

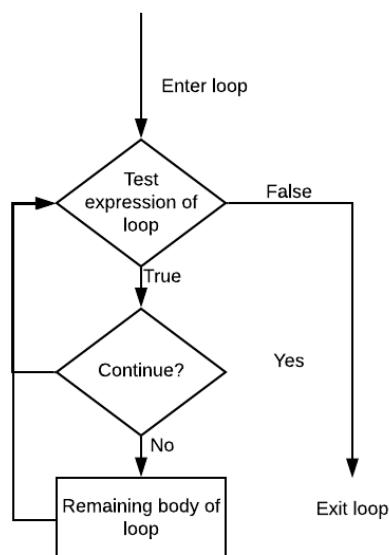
```
for num in range(1,10):
    if num == 5:
        break
    print(num)
>>
1
2
3
4
```

```

val = 0
while val >= 0:
    print(val)
    val = val+1
    if val == 3 :
        break
print(val)
>>
0
1
2
3

```

Continue statement digunakan untuk men-skip kode didalam loop pada iterasi saat itu. Loop tidak berhenti, tapi berlanjut pada iterasi berikutnya.



Gambar 28 – Diagram alir “Continue” statement

Contoh untuk **continue**:

```

for num in range(2, 10):
    if num % 2 == 0:
        print("Found an even number", num)
        continue
    print("Found a number", num)
>>
Found an even number 2
Found a number 3
Found an even number 4
Found a number 5
Found an even number 6
Found a number 7
Found an even number 8
Found a number 9

```

```
num=0
while num <=5:
    num=num+1
    if num%2 == 0:
        continue
    print(num)
>>
1
3
5
```

Sementara **pass statement** digunakan sebagai *placeholder*.

```
for letter in 'Python':
    if letter == 'h':
        pass
        print('This is pass block')
        print ('Current letter:', letter)
>>
This is pass block
Current letter: h
```

Looping Techniques

Infinite loop	Infinite loop dibuat menggunakan while statement . Jika syarat dari while loop adalah True , maka kita akan mendapatkan loop tak hingga.
Loop with condition at the top	While loop normal tanpa break statement . Syarat while loop diletakkan di atas dan loop berhenti saat syarat False.
Loop with condition in the middle	Loop ini dapat diimplementasikan menggunakan <i>infinite loop</i> sepanjang <i>conditional break</i> antara <i>body</i> suatu loop.
Loop with condition at the bottom	Loop ini memastikan <i>body</i> loop dieksekusi setidaknya satu kali. Dapat diimplementasikan menggunakan <i>infinite loop</i> sepanjang <i>conditional break</i> di akhir.

Contoh untuk **infinite loop**:

```
while True:  
    num = int(input("Enter an integer: "))  
    print("The double of", num, "is", 2 * num)  
  
>>  
Enter an integer: 1  
The double of 1 is 2  
Enter an integer: 2  
The double of 2 is 4  
Enter an integer:
```

Untuk menghentikan *infinite loop*, klik Stop pada Python Console.

Contoh untuk **loop with condition at the top**:

```
n = int(input("Enter n: "))  
# initialize sum and counter  
sum = 0  
i = 1  
while i <= n:  
    print('i:', i)  
    sum = sum + i  
    i = i+2  
print("The sum is", sum)  
  
>>  
Enter n: 10  
i: 1  
i: 3  
i: 5  
i: 7  
i: 9  
The sum is 25
```

Contoh untuk **loop with condition in the middle**:

```
vowels = "aeiouAEIOU"
while True:
    v = input("Enter a vowel: ")
    # condition in the middle
    if v in vowels:
        break
    print("That is not a vowel. Try again!")
print("Thank you!")

>>
Enter a vowel: a
Thank you!

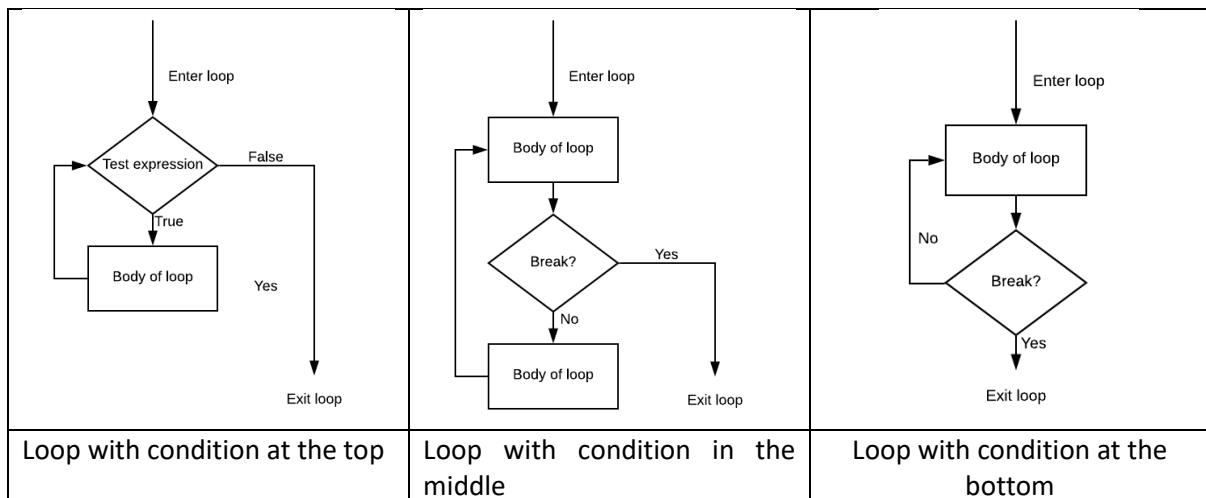
>>
Enter a vowel: b
That is not a vowel. Try again!
Enter a vowel:
```

Contoh untuk **loop with condition at the bottom**:

```
import random as rd
while True:
    input("Press enter to roll the dice")
    # get a number between 1 to 6
    num = rd.randint(1, 6)
    print("You got", num)
    option = input("Roll again? (y/n) ")

    # condition
    if option == 'n':
        break

>>
Press enter to roll the dice
You got 6
Roll again? (y/n)
```



Gambar 29 – Diagram alir untuk “loop with condition at the top”, “loop with condition in the middle”, “loop with condition at the bottom”

LATIHAN SOAL.

1. Gunakan “while” dan “break” untuk menuliskan pemrograman Python, dimana keluarannya adalah 0, 1, 2, 3, 4
2. Gunakan “for” dan “continue”, untuk menuliskan pemrograman Python, dimana keluarannya adalah 1, 3, 5, 7, 9
3. Tulis pemrograman Python dimana keluarannya adalah angka antara 2 dan 10.

File I/O

File adalah sebuah lokasi yang dinamai di dalam disk dan menyimpan informasi tertentu. **File** digunakan untuk menyimpan data secara permanen pada *non-volatile memory* (bukan memori tidak stabil), contohnya *hard disk*.

Operasi untuk *file*:

1. *Open a file*
2. *Read / Write (perform operation)*
3. *Close the file*

File biasanya disimpan dalam format ASCII. **ASCII** (*American Standard Code for Information Interchange*) adalah standar pengkodean karakter untuk komunikasi elektronik. Kode ASCII merepresentasikan teks dalam komputer, perlengkapan telekomunikasi dan perangkat lainnya. Setiap karakter dalam ASCII memiliki *size* sebesar 1 byte, sementara antar *line* memiliki *size* 2 byte.

Contoh:

bertanggungjawab : memiliki 16 karakter (*size*: 16 bytes)
bertanggung jawab : memiliki 17 karakter (*size*: 17 bytes)
saya }
bertanggung jawab : *size* 23 bytes (Enter = 2 bytes)

Setiap byte merupakan 8-bit yang terdiri atas 7-bit Standard ASCII dan 1-bit Extended ASCII.

Standard ASCII The first 32 characters are control codes.		Extended ASCII (DOS)	
0 Null	33 !	81 Q	128 C
1 Start of heading	34 "!	82 R	129 S
2 Start of text	35 #	83 S	130 T
3 End of text	36 \$	84 T	131 U
4 End of transmit	37 %	85 U	132 V
5 Enquiry	38 &	86 U	133 W
6 Acknowledge	39 ^	87 U	134 X
7 Audible bell	40 <	88 X	135 Y
8 Backspace	41 >	89 Y	136 Z
9 Horizontal tab	42 *	90 Z	137 ^
10 Line feed	43 +	91 L	138 <
11 Vertical tab	44 ,	92 V	139 >
12 Form feed	45 -	93 A	140 ^
13 Carriage return	46 .	94 B	141 <
14 Shift out	47 /	95 C	142 >
15 Shift in	48 \	96 D	143 ^
16 Data link escape	49 _	97 a	144 <
17 Device control 1	50 ^	98 b	145 >
18 Device control 2	51 ~	99 c	146 ^
19 Device control 3	52 #	100 d	147 <
20 Device control 4	53 \$	101 e	148 >
21 Neg. acknowledge	54 @	102 f	149 ^
22 Synchronous idle	55 ?	103 g	150 <
23 End trans. block	56 @	104 h	151 >
24 Cancel	57 ^	105 i	152 ^
25 End of medium	58 :	106 j	153 <
26 Substitution	59 ,	107 k	154 >
27 Escape	60 <	108 l	155 ^
28 File separator	61 =	109 m	156 <
29 Group separator	62 >	110 n	157 >
30 Record separator	63 ?	111 o	158 ^
31 Unit separator	64 @	112 p	159 <
32 Blank space	65 A	113 q	160 >
	66 B	114 r	161 ^
	67 C	115 s	162 <
	68 D	116 t	163 >
	69 E	117 u	164 ^
	70 F	118 v	165 <
	71 G	119 w	166 >
	72 H	120 x	167 ^
	73 I	121 y	168 <
	74 J	122 z	169 >
	75 K	123 {	170 ^
	76 L	124 ;	171 <
	77 M	125 >	172 >
	78 N	126 ~	173 ^
	79 O	127 ^	174 <
	80 P		175 >

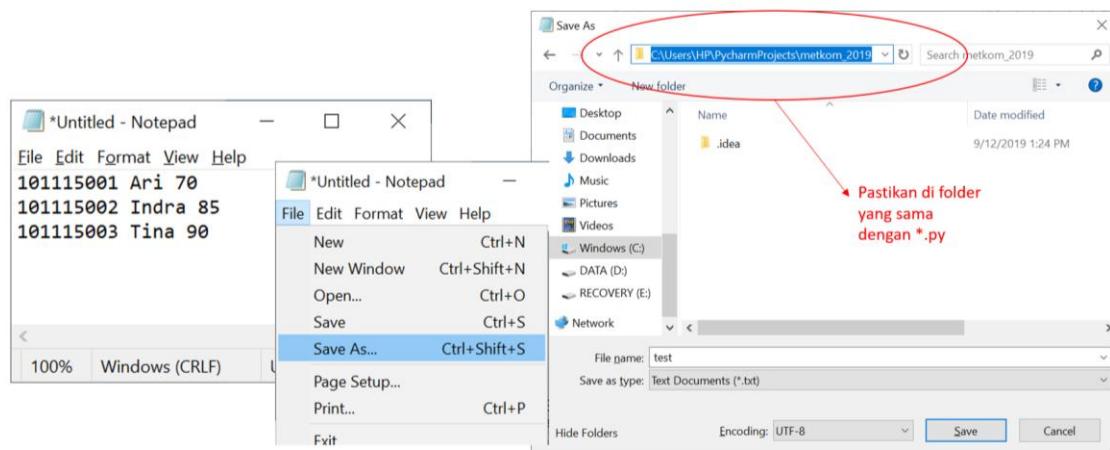
Gambar 30 – Byte pada ASCII (Sumber: <https://www.pcmag.com/encyclopedia/term/37184/7-bit-ascii>)

Contoh:

1	1	0	1	1	0	1
2^6	2^5	2^4	2^3	2^2	2^1	2^0
64	32	0	8	4	0	1

Jumlah total untuk *binary* 1101101 diatas adalah 109, merupakan karakter “m” seperti ditunjukkan gambar di atas.

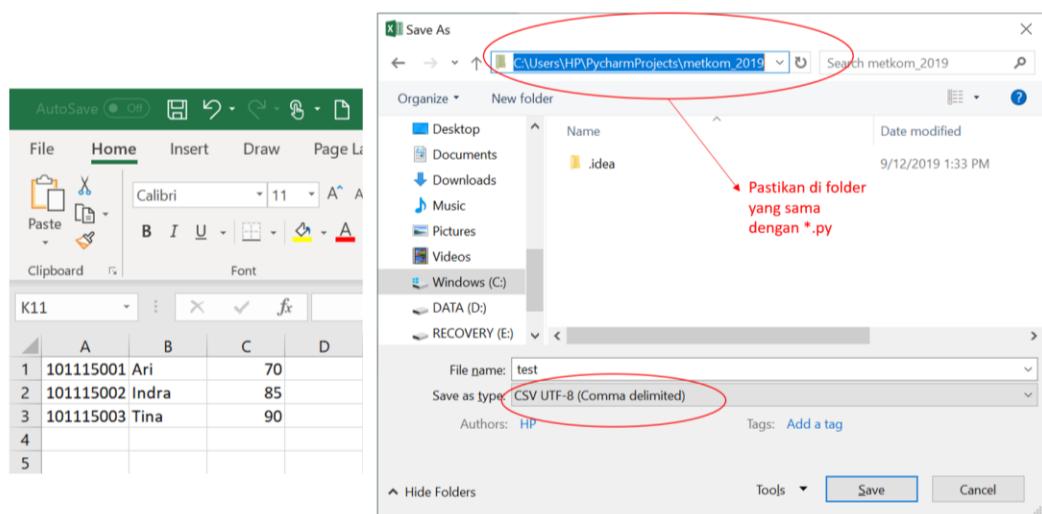
Terdapat beberapa cara membuat *file*, dapat langsung di PyCharm ataupun dari Text Editor (ex. Notepad). Contoh pembuatan *file* di Text Editor:

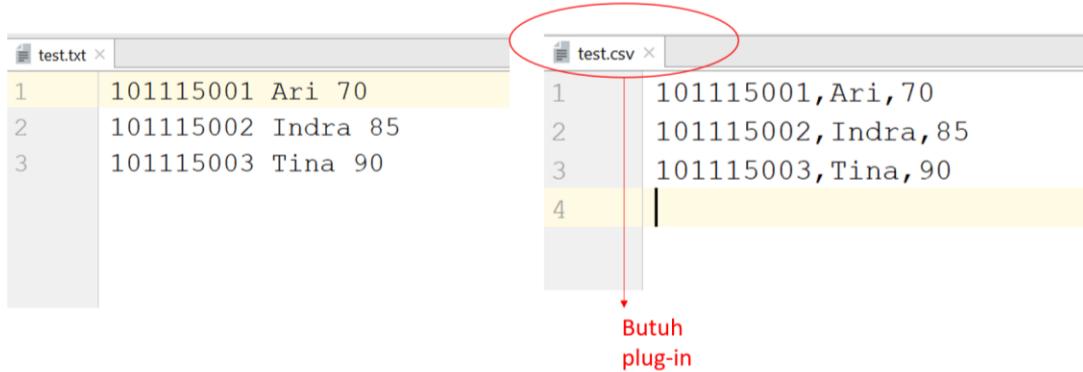


Gambar 31 – Pembuatan *file* pada Text Editor

File akan berada dalam *list folder* dan dapat dilihat di PyCharm.

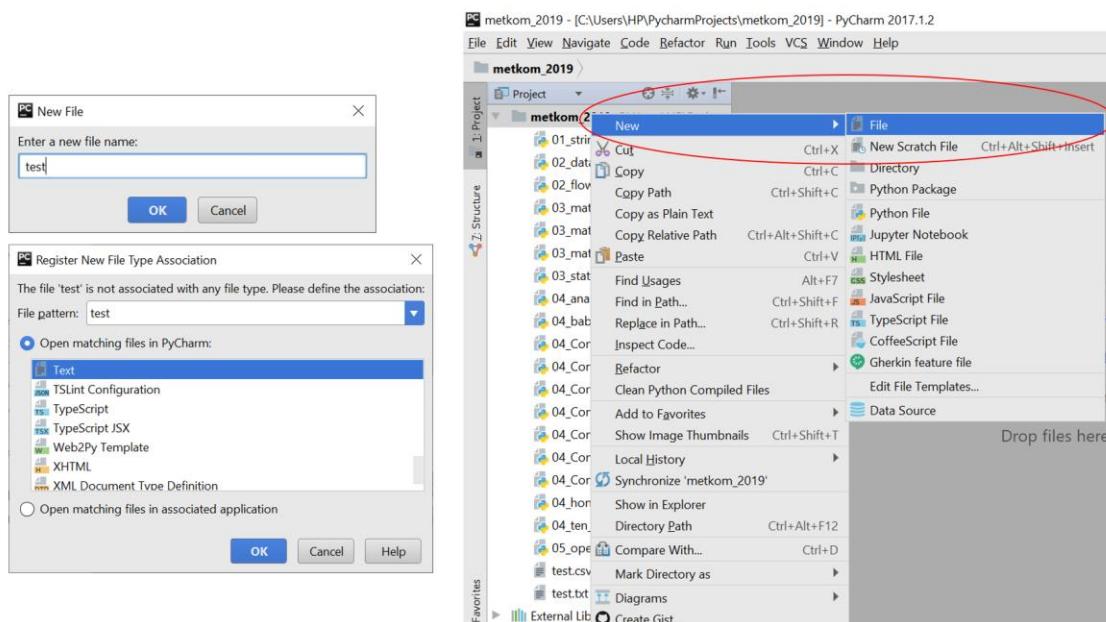
Contoh pembuatan *file* di Ms. Excel:





Gambar 32 – Pembuatan file di Ms. Excel

File dengan format csv (*comma separated value*) memiliki tanda koma (,) sebagai pemisah atau *delimiter* antar kolom. Untuk membaca file csv, maka perlu melakukan **import csv** pada Python.



Gambar 33 – Pembuatan file di PyCharm

Berikut adalah cara untuk mengetahui besar kapasitas file:

```
import os
f = open("test.txt")
print(os.stat("test.txt").st_size)
```

Open a File

Python memiliki fungsi *built-in* untuk membuka suatu *file*, yaitu **open()**. Fungsi ini mengembalikan obyek *file*, disebut juga dengan *handle*, digunakan untuk membaca ataupun memodifikasi *file*.

```
f = open("test.txt")      # open file in current directory
f = open("C:/Python33/README.txt")  # specifying full path
```

Jenis perintah disimbolkan dengan *mode* sebagai berikut:

Mode	Deskripsi
'r'	Membuka <i>file</i> untuk dibaca
'w'	Membuka <i>file</i> untuk ditulis. Membuat <i>file</i> baru jika sebelumnya belum ada, atau menggantikan <i>file</i> yang sudah ada.
'x'	Membuka <i>file</i> untuk kreasi eksklusif. Jika <i>file</i> sudah ada, operasi dibatalkan.
'a'	Membuka untuk menambahkan isi <i>file</i> tanpa menggantikan isi <i>file</i> yang sudah ada. Jika <i>file</i> belum ada, maka <i>file</i> baru dibuat.
't'	Membuka dalam <i>text mode</i> .
'b'	Membuka dalam <i>binary mode</i> .
'+'	Membuka <i>file</i> untuk di-update (<i>reading and writing</i>)

```
f = open("test.txt")      # equivalent to 'r' or 'rt'
f = open("test.txt", 'w')  # write in text mode
```

Close a File

Menutup *file* akan membebaskan dari ikatan *file* yang dilakukan Python, dengan metode **close()**.

```
f = open("test.txt")
# perform file operations
f.close()
```

Metode ini tidak sepenuhnya aman. Jika suatu pengecualian terjadi saat operasi dilakukan, kode akan berhenti tanpa menutup *file*. Cara terbaik untuk hal ini adalah dengan membuat *statement*, yang dapat meyakinkan bahwa *file* sudah tertutup saat keluar dari *block*.

```
with open("test.txt") as f:
# perform file operations
```

Write a File

Untuk menulis suatu *file*, perlu dilakukan *mode* ‘w’, atau *append* ‘a’ atau *exclusive creation* ‘x’. Menulis suatu *string* atau *sequence* dari *bytes* menggunakan metode **write()**. Metode ini mengembalikan sekumpulan angka atau karakter ke dalam *file*.

```
with open("test.txt", 'w') as f:  
    f.write("my first file\n")  
    f.write("This file\n\n")  
    f.write("contains three lines\n")
```

Read a File

Membaca *file* haruslah dalam *text mode*.

```
f = open("angka.txt", 'r')  
print(f.read(4)) # read the first 4 data
```

Posisi *file (cursor)* dapat diganti dengan metode **seek()**. Hal ini serupa dengan metode **tell()** untuk mengembalikan kepada posisi sekarang (dalam angka *bytes*).

```
f.seek()  
f.tell()
```

Save Data

Suatu data yang di-*generate* di Python dapat disimpan ke dalam format ASCII (contoh: *.txt) atau ke dalam format file Numpy (*.npy)

Berikut adalah suatu contoh data yang disimpan ke dalam format ASCII dan format file Numpy:

```
import numpy as np  
x = np.linspace(-4, 4, 81)  
y = np.sinc(x)  
print(x)  
data = np.vstack((x, y)).T  
np.save('./data', data)  
np.savetxt('./data.txt', data)
```

X merupakan suatu nilai dari -4 hingga 4 sebanyak 81 data, sedangkan Y adalah nilai $\text{sinc}(x)$ atau $\frac{\sin(x)}{x}$. Dengan menggunakan modul Numpy dan *command* `np.save` maka data X dan Y akan tersimpan ke dalam format file Numpy dan `np.savetxt` untuk menyimpan ke dalam format ASCII.

Load Data

Data dalam format ASCII dan file Numpy dapat di-load ke dalam program dengan menggunakan perintah sebagai berikut:

```
import numpy as np
data = np.load('./data.npy')
data = np.loadtxt('./data.txt')
```

```
import numpy as np
data = np.genfromtxt('test.txt')
NIM = data[:, 0]
Nama = data[:, 1]
Nilai = data[:, 2]
```

Berikut beberapa cara untuk meng-import file dalam format csv:

```
import csv
with open('test.txt') as csv_file:
    csv_reader = csv.reader(csv_file, delimiter=' ')
    x = []
    y = []
    z = []
    for row in csv_reader:
        x.append(row[0])
        y.append(row[1])
        z.append(row[2])
print(x)
print(y)
print(z)
```

```
import pandas as pd
tes = pd.read_csv('test.txt')
print(tes.head())
print(tes.columns)
```

LATIHAN SOAL.

Intan membeli buah dengan keterangan seperti file “Buah.txt” berikut.

Apel;50000;3
Pisang;30000;2
Jambu;25000;4
Mangga;35000;1

Load data dengan np.genfromtxt atau import csv

Variabel: nama, harga, jumlah

Input Value

Suatu *variable* dapat memiliki value yang ditentukan pada *script*, atau dimasukkan pada saat *running* program.

```
val = float(input("Enter your value: "))
print(val)
print(val/5)

>>
Enter your value: 75
75.0
15.0
```

Input value digunakan untuk kelanjutan operasi pada program, atau *infinite looping*.

```
while True:
    val = float(input("Enter your value: "))
    print("You got", val/5)
    option = input("Again? (y/n) ")
    if option == 'n':
        break

>>
Enter your value: 100
You got 20.0
Again? (y/n)
```

Exercise

"We are what we repeatedly do. Excellence, then, is not an act, but a habit." – Aristotle

Sejenak dari apa yang sudah kita pelajari, mari kita ulang dan kita asah kembali dengan masalah-masalah yang baru.

1. Load data dari file “**Brian.txt**” menggunakan `np.genfromtxt` atau `import csv`.

Barang	Harga
Biskuit	10.000
Deodorant	20.000
Shampoo	30.000
Deterjen	40.000
Mug	50.000
Handuk	60.000
Kopi	70.000
Sarung	80.000
Tumbler	90.000
Panci	100.000

2. Brian hanya membeli satu barang. Jika ia berbelanja menggunakan OVO, maka ia akan mendapatkan cashback sebesar 30% maksimum Rp. 25.000. Buatlah algoritma berapa biaya belanja Brian untuk setiap barang jika ia menggunakan OVO. (Gunakan `for` dan `if else`)
3. Buatlah algoritma untuk menentukan: keliling lingkaran, luas lingkaran, volume bola, luas permukaan bola. Jika diketahui jari-jari suatu lingkaran adalah r .
4. Generasi dengan tahun kelahiran tertentu dikategorikan sebagai berikut:

Tahun Kelahiran	Generasi
1945 – 1960	Baby Boomers
1961 – 1980	X
1981 – 1994	Y
1995 – 2010	Z
2011 – 2019	Alpha

Buatlah algoritma untuk mencari tahu seseorang yang lahir di tahun tertentu merupakan generasi yang mana.

5. Tulislah algoritma untuk menghitung gaji karyawan dengan keterangan:
Gaji bersih = Gaji Pokok + Tunjangan – Pajak
Tunjangan adalah 20% dari Gaji Pokok
Pajak adalah 15% dari Gaji Pokok + Tunjangan
6. Seorang pelari menempuh 10 km dalam 1 jam, 5 menit, 40 detik. Berapa detikkah total waktu yang ditempuh pelari tersebut?

7. Reino dan Syahrini melakukan percakapan di telepon selama 4287 detik. Jika dikonversi ke dalam jam, menit, detik, berapa lamakah waktu percakapan mereka?
8. Jika diketahui nilai koordinat x dan y suatu titik, bagaimana cara mengetahui titik tersebut berada di kuadran yang mana?
9. Buatlah algoritma untuk mencari akar persamaan kuadrat dari persamaan $ax^2 + bx + c = 0$
10. Buatlah algoritma untuk deret Fibonacci!
11. Berapakah tarif percakapan via telepon antara Reino dan Syahrini jika diketahui tarif progresif adalah sebagai berikut?

Lama	Tarif
5 menit pertama	Rp. 5.000
15 menit berikutnya	Rp. 10/ detik
30 menit berikutnya	Rp. 5/ detik
Berikutnya	Rp. 1/detik

12. Karyawan PT. ABC digaji berdasarkan jumlah jam kerjanya selama satu minggu. Upah per jam adalah Rp. 20.000. Bila jam kerja lebih besar dari 40 jam, maka sisanya dianggap sebagai jam lembur. Upah lembur adalah Rp. 30.000/ jam.
Tulislah algoritma yang membaca jumlah jam kerja seorang karyawan selama satu minggu, lalu menentukan upah mingguannya.
13. Upah karyawan PT. ABC ditentukan berdasarkan Golongannya. Berikut adalah tabel golongan dan upah per jam (hitung hanya untuk 40 jam kerja per minggu).

Golongan	Upah Per Jam
A	Rp. 50.000
B	Rp. 40.000
C	Rp. 30.000
D	Rp. 20.000

14. Jika diketahui shio nenek moyang kita adalah sebagai berikut:

Shio	Tahun	Shio	Tahun
Tikus	1924	Kuda	1930
Kerbau	1925	Kambing	1931
Macan	1926	Monyet	1932
Kelinci	1927	Ayam	1933
Naga	1928	Anjing	1934
Ular	1929	Babi	1935

Shio berulang setiap 12 tahun, dan cari tahulah seseorang yang lahir di tahun tertentu memiliki shio apa.

15. Buatlah file Data_den.txt dengan isi seperti berikut. Kolom pertama adalah Titik Pengukuran, kolom kedua adalah Data Pengamatan, dan kolom ketiga adalah Data Hasil Perhitungan. Buatlah *script* untuk memanggil input data dari file tersebut dan mengetahui isi dari setiap kolom tersebut. Silakan gunakan import numpy as np (np.genfromtxt). Buatlah algoritma untuk mengetahui titik pengukuran yang mempunyai galat kurang dari 0.015 dengan Galat = Data hasil pengukuran – Data pengamatan. (Gunakan *for* atau *while*, *if else*)

Contoh output yang diminta:

Titik 1 mempunyai galat > 0.015 yaitu

.....

Titik Mempunya galat < 0.015 yaitu

Titik	Pengamatan	Pengukuran
1	2.768	2.813
2	2.756	2.793
3	2.763	2.798
4	2.755	2.785
5	2.766	2.792
6	2.753	2.777
7	2.748	2.764
8	2.762	2.775
9	2.755	2.765
10	2.781	2.789

Number Systems

Dalam Matematika, suatu **base** merupakan angka dari digit berbeda atau kombinasi digit dan huruf dengan menggunakan suatu sistem untuk menghitung untuk merepresentasikan angka.

$$\log_a b = c$$

$$b = a^c$$

Sistem penomoran berdasarkan basis logaritma (a) untuk menentukan nilai b.

Misalnya,

1. *Base 10 (Decimal)* untuk merepresentasikan angka menggunakan 10 digit [0-9]
2. *Base 2 (Binary)* untuk merepresentasikan angka menggunakan 2 digit [0-1]
3. *Base 8 (Octal)* untuk merepresentasikan angka menggunakan 8 digit [0-7]
4. *Base 16 (Hexadecimal)* untuk merepresentasikan angka menggunakan 10 digit [0-9] dan 6 karakter [A, B, C, D, E, F], dimana [A, B, C, D, E, F] merepresentasikan angka [10, 11, 12, 13, 14, 15]

Berikut adalah cara membaca setiap *number systems* diatas:

Number Systems	Example	How to Read	Result										
Decimal	70526	<table border="1"><tr><td>7</td><td>0</td><td>5</td><td>2</td><td>6</td></tr><tr><td>10^4</td><td>10^3</td><td>10^2</td><td>10^1</td><td>10^0</td></tr></table> $= (7 \times 10^4) + (0 \times 10^3) + (5 \times 10^2) + (2 \times 10^1) + (6 \times 10^0)$ $= 70.000 + 0 + 500 + 20 + 6$	7	0	5	2	6	10^4	10^3	10^2	10^1	10^0	70526
7	0	5	2	6									
10^4	10^3	10^2	10^1	10^0									
Binary	10101	<table border="1"><tr><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td>2^4</td><td>2^3</td><td>2^2</td><td>2^1</td><td>2^0</td></tr></table> $= (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0)$ $= 16 + 0 + 4 + 0 + 1$	1	0	1	0	1	2^4	2^3	2^2	2^1	2^0	21
1	0	1	0	1									
2^4	2^3	2^2	2^1	2^0									
Octal	143	<table border="1"><tr><td>1</td><td>4</td><td>3</td></tr><tr><td>8^2</td><td>8^1</td><td>8^0</td></tr></table> $= (1 \times 8^2) + (4 \times 8^1) + 3 \times 8^0$ $= 64 + 32 + 3$	1	4	3	8^2	8^1	8^0	99				
1	4	3											
8^2	8^1	8^0											
Hexadecimal	AF7	<table border="1"><tr><td>10</td><td>15</td><td>7</td></tr><tr><td>16^2</td><td>16^1</td><td>16^0</td></tr></table> $= (10 \times 16^2) + (15 \times 16^1) + 7 \times 16^0$ $= 2560 + 240 + 7$	10	15	7	16^2	16^1	16^0	2807				
10	15	7											
16^2	16^1	16^0											

```
print(0b10101)
print(0B10101)
print(0o143)
print(0O143)
print(0xAF7)
print(0XAf7)
print(bin(79))
print(oct(79))
print(hex(79))
print(0b1110101)
print(0o3421)
print(0x2C8)
print(0x1DE + 0o112)
```

Random Numbers

Tanpa data *real* di lapangan, data sintetik dapat digunakan untuk komputasi. Data sintetik dapat diperoleh dengan *generating* suatu *array* atau *random number*.

Berikut adalah perintah Python untuk men-generate angka acak antara 0.0 – 1.0 sebanyak 10 kali.

```
import random as rd
for i in range(10):
    print(rd.random())
```

Sorting Numbers

Berikut adalah perintah Python untuk mengurutkan angka pada list.

```
a = [25, 10, 7.5, 308, 411, 0.9]
print(sorted(a))
print(sorted(a, reverse=True))
```

Array Generator

Array dua dimensi memiliki jumlah baris dan kolom (*matrix*), maka *index* pun merupakan identitas baris dan kolom pada suatu *array*.

	Kolom 1	Kolom 2	Kolom 3
Baris 1	Index [0][0]	Index [0][1]	Index [0][2]
Baris 2	Index [1][0]	Index [1][1]	Index [1][2]

Contoh *matrix* (2x2) sebagai berikut:

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Nilai berdasarkan *index* adalah:

$$A[0][0] = 1$$

$$A[0][1] = 2$$

$$A[1][0] = 3$$

$$A[1][1] = 4$$

Penulisan pada *list* adalah sebagai berikut:

```
A = [[1, 2], [3, 4]]
```

Setiap elemen dalam satu tanda kurung [] merupakan satu baris, yang kemudian diurutkan berdasarkan kolom. Jika *index* hanya satu angka, maka angka tersebut mengindikasikan barisnya saja.

```
print(A[0])
>> [1, 2]
Untuk array yang besar, maka penggunaan list tidak lagi efisien. Paket NumPy dibutuhkan untuk hal tersebut. NumPy merupakan package dasar untuk komputasi saintifik dengan Python yang memiliki keunggulan berikut:
```

- *Object* berupa *N-dimensional array* yang kuat
- Fungsi yang canggih
- *Tools* untuk mengintegrasikan C/C++ dan ode Fortran
- *Linear algebra, Fourier transform* dan kapabilitas bilangan acak
- Sebagai *container* multi-dimensi yang efisien untuk data generik
- Berintegrasi dengan bermacam *database*



Gambar 34 - Logo NumPy

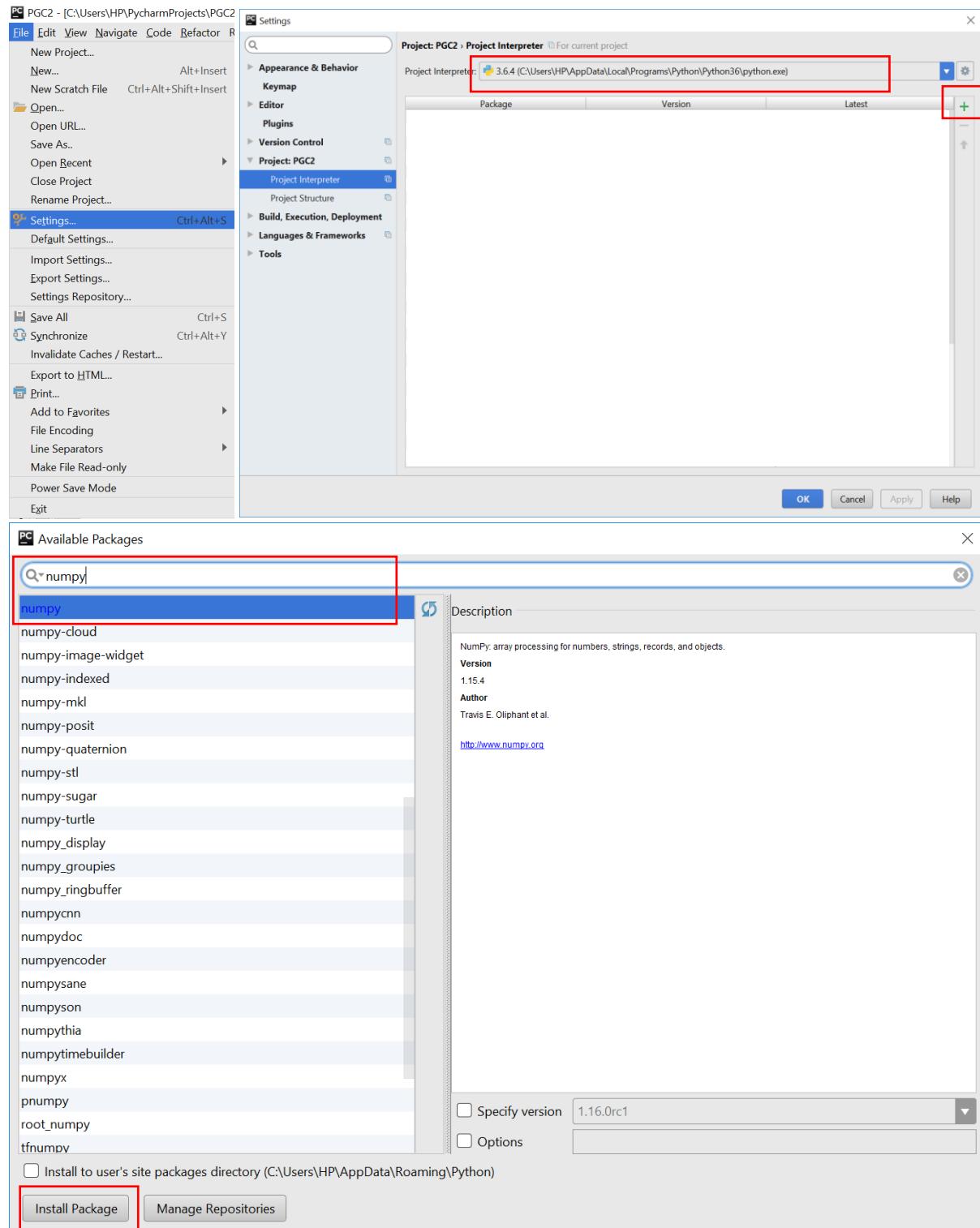
Berikut adalah tahapan menambahkan package NumPy ke Python:

File → Settings → Project Interpreter

Pastikan *Project Interpreter* sudah menunjukkan lokasi *python.exe*

Klik tanda + di sebelah kanan atas *window*.

Ketik “numpy” pada *search box*, lalu klik *install package*.



Gambar 35 - Langkah menambahkan modul (package) NumPy pada Python

Cara membuat suatu *array* jika diketahui batas minimum, maksimum dan *increment*:

```
import numpy as np
x = np.arange(0,361,10) #begin,end+1,inc
print(x)

>>
[ 0  10  20  30  40  50  60  70  80  90 100 110 120 130 140 150
160 170
180 190 200 210 220 230 240 250 260 270 280 290 300 310 320 330
340 350
360]
```

Note: Penggunaan `as np` pada saat `import numpy` adalah untuk memudahkan penulisan pada *script* agar cukup menuliskan `np` sebagai pengganti tulisan `numpy`.

Cara lainnya adalah dengan diketahui minimum, maksimum dan jumlah angka yang keluar.

```
import numpy as np
y = np.linspace(0,360,10) #begin,end,number
print(y)

>>
[ 0.  40.  80. 120. 160. 200. 240. 280. 320. 360.]
```

2D Array

Dalam menyusun *2D array* digunakan *for loop* untuk mengubah dari *list* dengan ketentuan:

```
for row in a:  
    for elem in row:  
        print(elem, end=' ')
```

row sebagai baris dan elem sebagai kolom.

$$a = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$$

```
a = [[1, 2, 3], [4, 5, 6]]  
for row in a:  
    for elem in row:  
        print(elem, end=' ')  
    print()  
>>  
1 2 3  
4 5 6
```

Sebagaimana *list* dapat diubah isinya, maka *2D array* juga *mutable*.

```
a = [[1, 2, 3], [4, 5, 6]]  
a[1][1] = 8  
for row in a:  
    for elem in row:  
        print(elem, end=' ')  
    print()  
>>  
1 2 3  
4 8 6
```

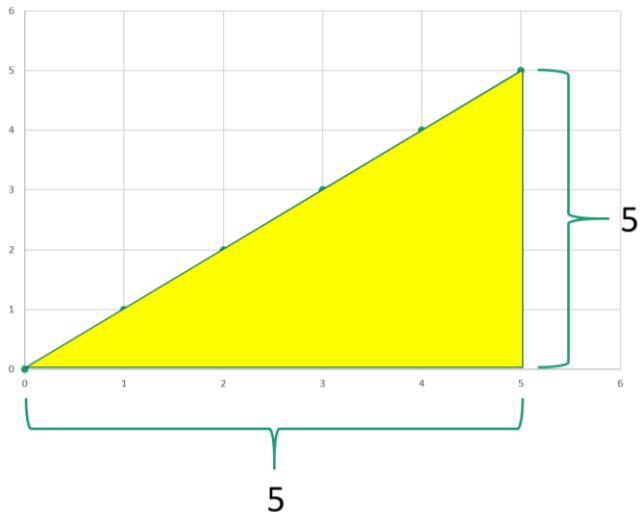
Integral

Integral dapat dihitung dengan mengetahui luas dibawah kurva, atau dengan persamaan berikut:

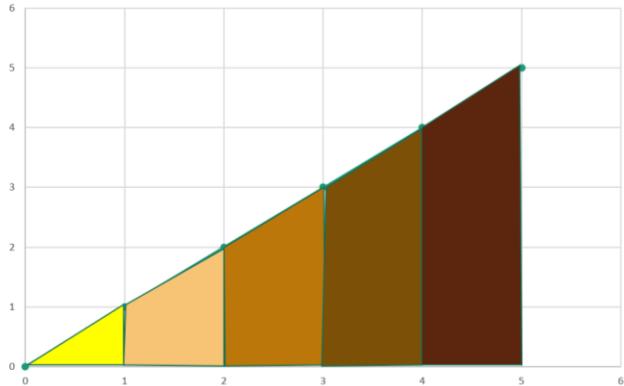
$$\int_a^b x^n dx = \frac{1}{n+1} x^{n+1} \Big|_a^b = \left(\frac{1}{n+1} b^{n+1} \right) - \left(\frac{1}{n+1} a^{n+1} \right)$$

Dimana kurva berada pada batas sumbu x adalah antara a dan b, dengan persamaan x^n .

$$\begin{aligned} \int_0^5 x dx &= ? \\ \frac{1}{2} x^2 \Big|_0^5 &= 12.5 - 0 = 12.5 \\ \frac{5 \times 5}{2} &= 12.5 \end{aligned}$$



$$\begin{aligned} \int_0^5 x dx &= ? \\ a &= \frac{(0+1) \times 1}{2} = 0.5 & d &= \frac{(3+4) \times 1}{2} = 3.5 \\ b &= \frac{(1+2) \times 1}{2} = 1.5 & e &= \frac{(4+5) \times 1}{2} = 4.5 \\ c &= \frac{(2+3) \times 1}{2} = 2.5 & \text{Total } & 12.5 \end{aligned}$$



Berdasarkan *trapezoidal rule*, maka diperoleh:

$$\int_a^b f(x) dx = \sum_{i=a+1}^b \frac{f(x_{i-1}) + f(x_i)}{2} \Delta x_i$$

$$\int_a^b f(x) dx = \frac{\Delta x}{2} (f(a) + 2f(a+1) + 2f(a+2) + \dots + 2f(b-1) + f(b))$$

$$\Delta x = \frac{b-a}{n}$$

$n = \text{jumlah partisi}$

Berikut adalah pemrograman untuk menghitung integral menggunakan *looping* dengan luas daerah dibawah kurva.

```
a = 0
b = 5
d = 1
area = 0
while a < b:
    area = area+((a+(a+1))*d)/2
    a += 1
print(area)

for i in range(a,b):
    area = area + (((i + (i + 1)) * d) / 2)
print(area)
```

LATIHAN SOAL.

Hitunglah luas dibawah kurva untuk

$$\int_0^5 (x^3 + x + 1) dx$$

Code Optimization

Hal yang perlu diperhatikan dalam *coding* adalah *make it run, make it right, make it fast*. Optimisasi merupakan langkah untuk membuat program yang efektif, namun memiliki kode yang tidak umum dan kompleks, serta *programmer* akan lebih banyak meluangkan waktu untuk membuat kode.

Code bisa melambat karena keterbatasan sumber daya komputer, diantaranya:

- **CPU-bound** (ketika CPU sudah mencapai batas kerja maksimum)
- **Memory-bound** (RAM terbatas)
- **IO-bound** (data transfer ke dan dari *hard disk* yang besar)
- **Network-bound** (CPU menunggu data menuju jaringan, atau dari “kelaparan” memory)

Hal pertama yang harus dilakukan dalam *performance optimization* adalah membuat *coding* yang sesuai dengan *style* masing-masing, yang berjalan dan dapat di-submit.

Kesalahan yang paling banyak ditemukan pada *developer Python* adalah:

1. Kesalahan penggunaan ekspresi pada *function*
2. Kesalahan penggunaan *variable*
3. Kesalahan men-spesifikasi parameter untuk pengecualian
4. Kesalahpahaman dalam aturan Python
5. Memodifikasi *list* ketika iterasi
6. Kesalahpahaman dalam hubungan *variable* dalam suatu *closure*
7. Membuat ketergantungan modul yang sirkular
8. Penamaan *clash* atau sama dengan modul *Python Standar Library*
9. Tidak mengetahui perbedaan antara Python 2 dan Python 3
10. Kesalahan penggunaan metode `__del__`

Cara penanganan hal-hal diatas yang biasanya dilakukan”

1. **Cheat:** menggunakan mesin yang lebih baik, atau menyelesaikan permasalahan yang lebih sederhana.
2. Menemukan apa yang memperlambat *code* (**profiling**): menggunakan `timeit`, `time`, `cProfile`, `line_profiler`, `memory_profiler`
3. Menggunakan **algoritma** dan **data structure** yang lebih baik
4. Menggunakan *compiled code* yang ditulis dalam bahasa lain (**C/C++ and Fortran**)
5. **Parallel program:** permasalahan paralel
6. Mengeksusi secara **paralel**: dalam mesin yang *multi-core*, mesin yang banyak (Ipython, Hadoop/SPARK, pada GPU)

Code Optimization pada Python diantaranya adalah:

Menggunakan *Stop-Watch Profiling* dengan modul *Timeit*, yang merekam waktu yang diperlukan untuk mengeksekusi suatu kode dan menghitung waktu yang sudah dilewati dalam *milliseconds*.

Berikut merupakan contoh dari perintah dengan tujuan yang sama, namun dengan kode yang berbeda, memiliki kecepatan waktu eksekusi yang berbeda. Manakah kode yang lebih optimal dan tidak *time consuming*?

```

import time

#Slower
start1 = time.time()
a = range(100000)
b = []
for i in a:
    b.append(i*2)
end1 = time.time()
print(end1-start1)
>>0.016762495040893555

#Faster
start2 = time.time()
a = range(100000)
b = [i*2 for i in a]
end2 = time.time()
print(end2-start2)
>>0.0030241012573242188

```

```

import time
import numpy as np

#Slower
start1 = time.time()
a = range(100000)
b = np.zeros(len(a))
for i in a:
    b[i]=i*2
end1 = time.time()
print(end1-start1)
>>0.03832578659057617

#Faster
start2 = time.time()
a = range(100000)
b = []
for i in a:
    b.append(i*2)
end2 = time.time()
print(end2-start2)
>>0.0289154052734375

```

Contoh lainnya, menukar nilai variabel *x* dan *y* dengan menyisipkan variabel lain. Optimasi yang dilakukan dalam hal ini bermaksud menyingkat jumlah langkah yang dilakukan, antara lain:

```
#slower
x = 2
y = 5
temp = x
x = y
y = temp
print(x,y)

#faster
x, y = 2, 5
x, y = y, x
print(x,y)
```

Atau dengan menyingkat *footprint* seperti contoh berikut:

```
my_var = 'you'
#slow
msg = 'hello ' + my_var + ' world'
print(msg)

#fast
msg = 'hello %s world' % my_var
print(msg)

#faster
msg = 'hello {} world'.format(my_var)
print(msg)
```

```
#slow
msg = 'line1\n'
msg += 'line2\n'
msg += 'line3\n'
print(msg)

#fast
msg = ['line1', 'line2', 'line3']
s = '\n'
print(s.join(msg))
```

Function

Fungsi (*user-defined function*) merupakan satu *block* kode yang tersusun dan dapat digunakan ulang untuk eksekusi perintah lain yang berkaitan dengan fungsi tersebut. *Functions* merupakan sekelompok pernyataan yang berkaitan untuk melakukan suatu tugas tertentu. *Functions* merupakan cara yang praktis untuk memisahkan kode menjadi blok-blok, agar kode lebih mudah untuk dibaca, digunakan kembali dan menghemat waktu. *Functions* merupakan kunci untuk mendefinisikan suatu *interface* agar *programmer* dapat berbagi kode.

Tipe *functions* antara lain:

- **Built-in functions** – fungsi yang sudah di-*built* di Python.
- **User-defined functions** – fungsi yang didefinisikan oleh pengguna.
- **Lambda functions** – fungsi yang tidak membutuhkan *function name*.
- **Recursion functions** – fungsi untuk memanggil dirinya sendiri.

Built-in functions

Interpreter Python memiliki sejumlah fungsi yang tersedia untuk digunakan. Fungsi tersebut dikenal sebagai **built-in functions**. Contohnya, fungsi **print()** untuk menampilkan obyek pada layer atau teks. Pada Python 3.6 terdapat 68 built-in functions. Contoh lainnya adalah **abs()**, untuk memberikan nilai absolut suatu angka. Jika angka adalah bilangan kompleks, maka nilai **abs()** adalah nilai sebenarnya.

User-defined functions

Functions pada Python didefinisikan menggunakan kata kunci **def** pada blok, diikuti nama fungsi sebagai nama blok dan tanda kurung () yang biasanya berisi parameter. Blok kode dimulai dengan *colon* (:) dan *di-indent*. Pernyataan **return [expression]** merupakan akhir dari *function*, mengembalikan ekspresi kepada panggilan. Pernyataan **return** tanpa argument memiliki arti yang sama dengan **return None**.

Fungsi didefinisikan terlebih dahulu menggunakan *block* sebagai berikut:

```
def functionname(parameters) :  
    function_suite  
    return [expression]
```

def	<i>Keyword</i> untuk menjalankan fungsi
functionname	Nama dari fungsi yang dibuat
parameters	Parameter yang akan di- <i>input</i> untuk fungsi tersebut
function_suite	Fungsi (persamaan) yang dibuat
return	Pernyataan untuk menghentikan fungsi

Contoh fungsi:

```

def my_function():
    print("Hello From My Function!")

def my_function_with_args(username, greeting):
    print("Hello, %s , From My Function!, I wish you
%s"%(username, greeting))

def sum_two_numbers(a, b):
    return a + b

```

Hasil dari fungsi dapat ditampilkan dengan memanggil fungsi tersebut, disertai nilai dari variabel yang ditentukan pada fungsi. Sebagai contoh:

```

def sum(a, b):
    print(f"{a}+{b}={a+b}")

sum(2,3)

>>2+3=5

```

```

def my_function(x):
    return 5 * x
print(my_function(3))
print(my_function(5))
print(my_function(9))

>>
15
25
45

```

```

def printinfo( name, age):
    print("Name: ", name)
    print("Age: ", age)
printinfo( age=10, name="miki" )

>>
Name: miki
Age: 10

```

```

def list_benefits():
    return "More organized code", "More readable code"

def build_sentence(benefit):
    return "%s is a benefit of functions!" % benefit

def name_the_benefits_of_functions():
    for benefit in list_benefits():
        print(build_sentence(benefit))
name_the_benefits_of_functions()

>>
More organized code is a benefit of functions!
More readable code is a benefit of functions!

```

Contoh fungsi menggunakan *built-in functions*:

```
def absolute_value(num):
    if num >= 0:
        return num
    else:
        return -num

print(absolute_value(2))
print(absolute_value(-4))

>>
2
4
```

```
def absolute_value2(num):
    num = abs(num)
    return num

print(absolute_value2(2))
print(absolute_value2(-4))

>>
2
4
```

```
# This function adds two numbers
def add(x,y):
    return x + y
# This function subtracts two numbers
def subtract(x, y):
    return x - y
# This function multiplies two numbers
def multiply(x, y):
    return x * y
# This function divides two numbers
def divide(x, y):
    return x / y

print("Select operation.")
print("1.Add")
print("2.Subtract")
print("3.Multiply")
print("4.Divide")

# Take input from the user
choice = input("Enter choice(1/2/3/4):")
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))

if choice == '1':
    print(num1,"+",num2,"=", add(num1,num2))
elif choice == '2':
    print(num1,"-",num2,"=", subtract(num1,num2))
elif choice == '3':
    print(num1,"*",num2,"=", multiply(num1,num2))
elif choice == '4':
    print(num1,"/",num2,"=", divide(num1,num2))
else:
    print("Invalid input")
```

Functions dapat dibuat dalam sebuah file Python, kemudian dipanggil dari *file* yang lain menggunakan **from**. Contoh *script* berikut disimpan dalam *file rumus.py*

```
import numpy as np
def luasling(r):
    L = np.pi*r**2
    return L
```

Pada *python file* yang lain, tuliskan:

```
from rumus import luasling
A = luasling(2)
print(A)
>>12.566370614359172
```

Pastikan bahwa kedua *python file* berada pada folder yang sama.

Fungsi juga dapat digunakan untuk suatu *sequence* dari parameter dengan *range* tertentu.

```
from rumus import luasling
r = range(1,10)
for i in r:
    print('r:',i,'L:',luasling(i))
>>
r: 1 L: 3.141592653589793
r: 2 L: 12.566370614359172
r: 3 L: 28.274333882308138
r: 4 L: 50.26548245743669
r: 5 L: 78.53981633974483
r: 6 L: 113.09733552923255
r: 7 L: 153.93804002589985
r: 8 L: 201.06192982974676
r: 9 L: 254.46900494077323
```

Untuk menyimpan hasil diatas ke dalam *list*, maka lakukan langkah berikut:

```
from rumus import luasling
r = range(1,10)
L=[]
for i in r:
    L.append(luasling(i))
print(L)
>>
[3.141592653589793, 12.566370614359172, 28.274333882308138,
50.26548245743669, 78.53981633974483, 113.09733552923255,
153.93804002589985, 201.06192982974676, 254.46900494077323]
```

Lambda functions

Lambda functions tidak memiliki banyak argumen, cukup dengan satu ekspresi.

```
myvar=lambda a,b: (a*b)+2  
print(myvar(3,5))  
  
>>17
```

Lambda operator digunakan untuk membuat fungsi anonim sederhana untuk menyelesaikan suatu persamaan dengan berbagai parameter.

```
def quad_equation(a,b,c):  
    return lambda x: a*x**2+b*x+c  
  
y = quad_equation(2,3,-5) (3)  
  
print(y)  
>>22
```

Persamaan untuk `quad_equation` diatas adalah:

$$y = ax^2 + bx + c$$

a=2, b=3, c=-5, x=3 sehingga:

$$y = 2 \times (3)^2 + 3 \times (3) + (-5) = 18 + 9 - 5 = 22$$

Recursion functions

Recursion functions merupakan fungsi yang memanggil fungsinya sendiri. Contoh:

```
def facto(n):  
    if n==1:  
        return 1  
    return n*facto(n-1)  
  
print(facto(3))  
  
>>6
```

LATIHAN SOAL.

1. Buatlah fungsi untuk:
 - Bilangan genap atau ganjil
 - Akar kuadrat
 - Tahun kabisat
 - Luas dan keliling lingkaran
 - Volume dan luas permukaan bola
2. Brian ingin membeli Boba Milk Tea seharga Rp. 40.000. Toko tersebut menerima pembayaran dengan aplikasi OVO, DANA dan GOPAY.
 - OVO memberikan *cashback* 30% dengan *cashback* maksimum Rp. 15.000.
 - GOPAY memberikan *cashback* 25% dengan *cashback* maksimum Rp. 12.500.
 - DANA memberikan *cashback* 40% dengan *cashback* maksimum Rp. 20.000.Aplikasi apa yang sebaiknya Brian gunakan jika ia ingin mendapatkan harga termurah?

Visualisasi

Selain NumPy, *package* lain yang dibutuhkan adalah Matplotlib, sebagai *Python 2D plotting library* untuk memproduksi gambar yang berkualitas untuk publikasi dalam berbagai format, dan lingkungan interaktif antar *platform*. Matplotlib biasa digunakan dalam *script* Python, atau juga *toolkit* lainnya. Matplotlib dapat men-generate plot, histogram, power spectra, bar chart, errorchart, scatterplot, dll. Modul pyplot digunakan untuk *plotting* yang sederhana yang mirip dengan MATLAB.



Gambar 36 - Logo Matplotlib

Visualisasi yang dapat ditampilkan antara lain:

- *Line Plots*
- *Pie Charts, Bar Plots, and Histograms*
- *Discrete Data Plots*
- *Polar Plots*
- *Contour Plots*
- *Vector Fields*
- *Surfaces, Volumes, and Polygons*
- *Animation*
- *Images*

Misalkan pada contoh array sinusoidal diatas, dapat dilakukan *plotting* berikut:

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

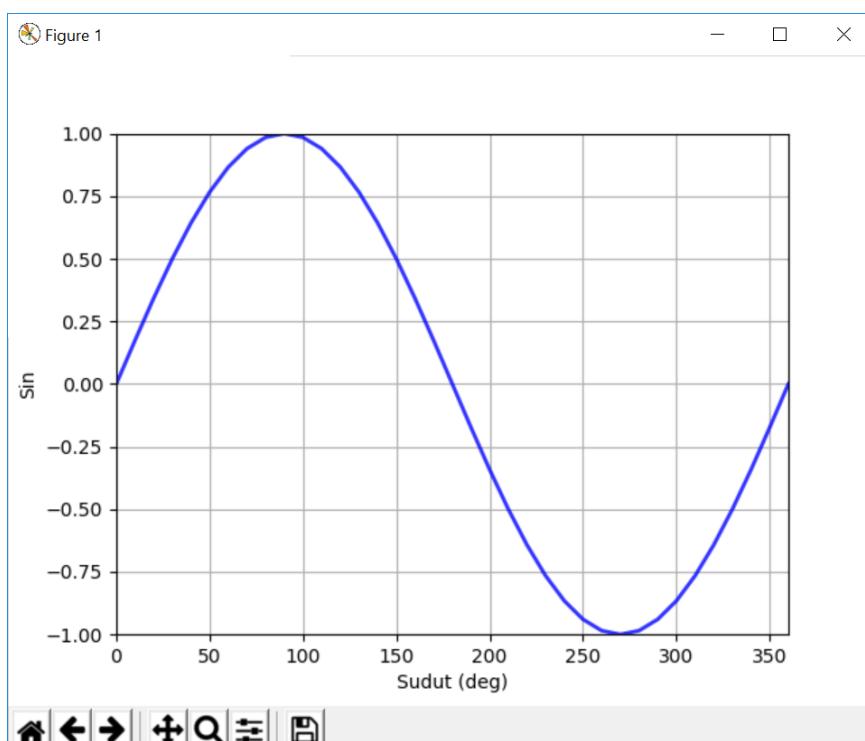
x = np.arange(0,361,10)
y = np.sin(x*np.pi/180)

plt.plot(x,y, color='b', linewidth='2', alpha=0.8)
plt.xlabel('Sudut (deg)')
plt.ylabel('Sin')
plt.xlim(0,360)
plt.ylim(-1,1)
plt.grid('True')
plt.show()
```

Note: plt.plot() dan plt.show() adalah *mandatory*.

Aksesoris lainnya dari *script* diatas:

plt.xlabel()	Memberikan label untuk sumbu x
plt.ylabel()	Memberikan label untuk sumbu y
plt.xlim()	Memberikan batasan koordinat sumbu x
plt.ylim()	Memberikan batasan koordinat sumbu y
plt.grid()	Memberikan <i>grid</i> pada <i>plot area</i>
color	Pewarnaan kurva b: blue g: green r: red c: cyan m: magenta y: yellow k: black w: white
linewidth	Ketebalan garis kurva
alpha	Ketajaman warna pada kurva

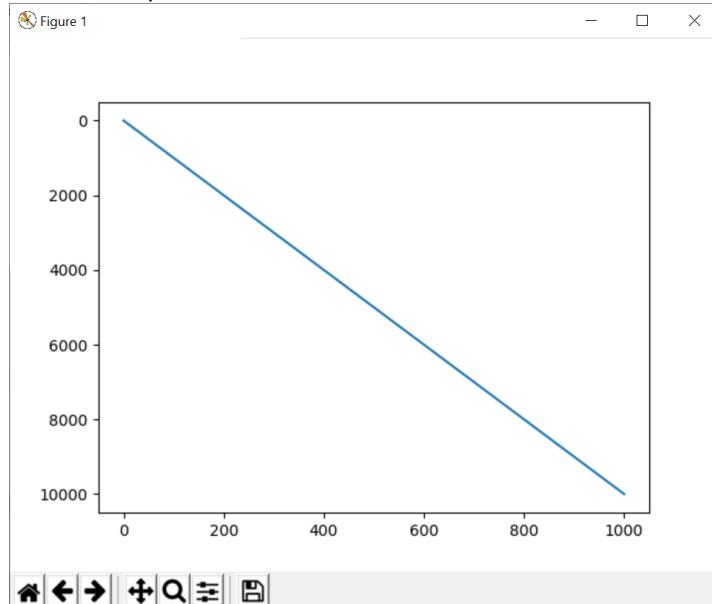


Gambar 37 - Hasil plotting sinusoidal dengan Matplotlib

Contoh *plotting* jika sumbu positif dan negatif dibalik (**inverse axis**):

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(0,1001,10)
z = np.arange(0,10001,100)
plt.plot(x,z)
plt.gca().invert_yaxis()
plt.show()
```

Gambar yang dihasilkan dari *script* diatas adalah:

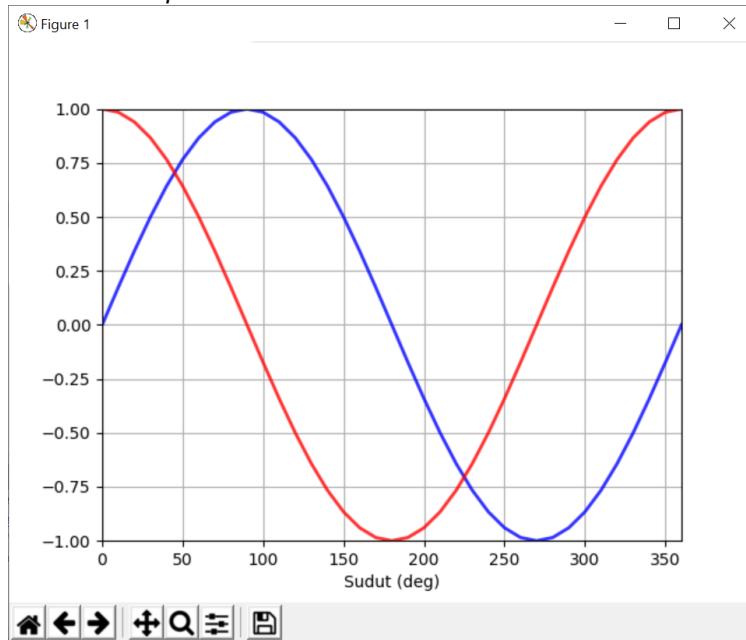


Gambar 38 – Gambar plot untuk inverse axis

Contoh *plotting* untuk meng-*overlay* dua kurva:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(0, 361, 10)
a = np.sin(x*np.pi/180)
b = np.cos(x*np.pi/180)
plt.plot(x,a, '--go', x,b, ':r*')
plt.xlabel('Sudut (deg)')
plt.xlim(0,360)
plt.ylim(-1,1)
plt.show()
```

Gambar yang dihasilkan dari *script* diatas adalah:

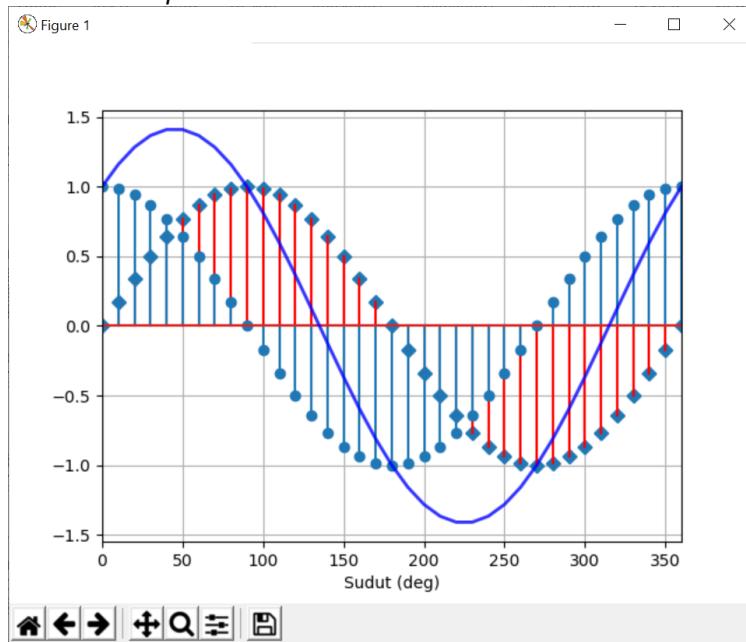


Gambar 39 – Gambar plot untuk overlay dua kurva

Contoh *plotting* untuk **stem** atau plot garis vertikal dari *baseline* ke koordinat-y dan membuat *marker*:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(0,361,10)
a = np.sin(x*np.pi/180)
b = np.cos(x*np.pi/180)
plt.stem(x, a, linefmt='red', markerfmt='D')
plt.stem(x, b)
plt.plot(x,a+b, color='b', linewidth='2', alpha=0.8)
plt.xlabel('Sudut (deg)')
plt.xlim(0,360)
plt.grid('True')
plt.show()
```

Gambar yang dihasilkan dari *script* diatas adalah:

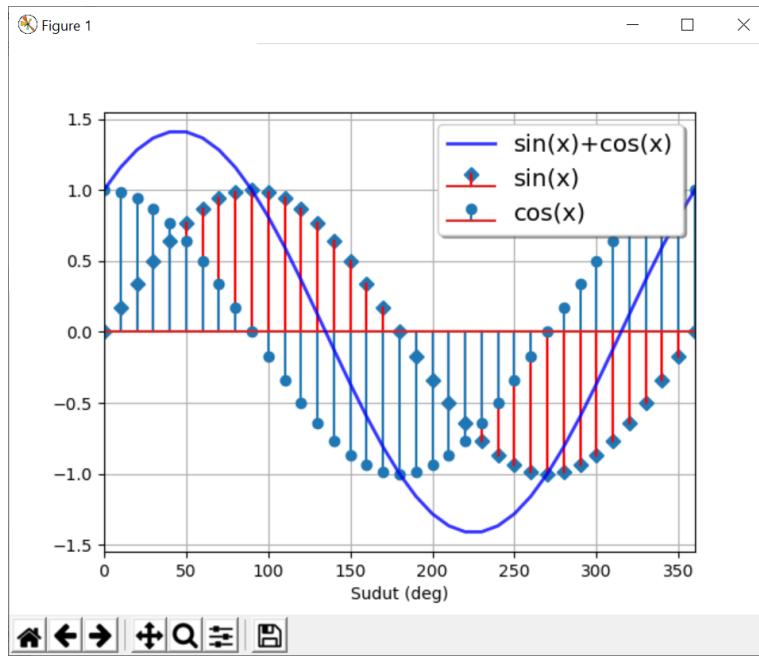


Gambar 40 – Gambar plot untuk stem

Contoh *plotting* untuk menambahkan **legend**:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(0,361,10)
a = np.sin(x*np.pi/180)
b = np.cos(x*np.pi/180)
plt.stem(x, a, linefmt='red', markerfmt='D', label='sin(x)')
plt.stem(x, b, label='cos(x)')
plt.plot(x,a+b, color='b', linewidth='2', alpha=0.8,
label='sin(x)+cos(x)')
plt.legend(loc='upper right', shadow=True, fontsize='x-large')
plt.xlabel('Sudut (deg)')
plt.xlim(0,360)
plt.grid('True')
plt.show()
```

Gambar yang dihasilkan dari *script* diatas adalah:

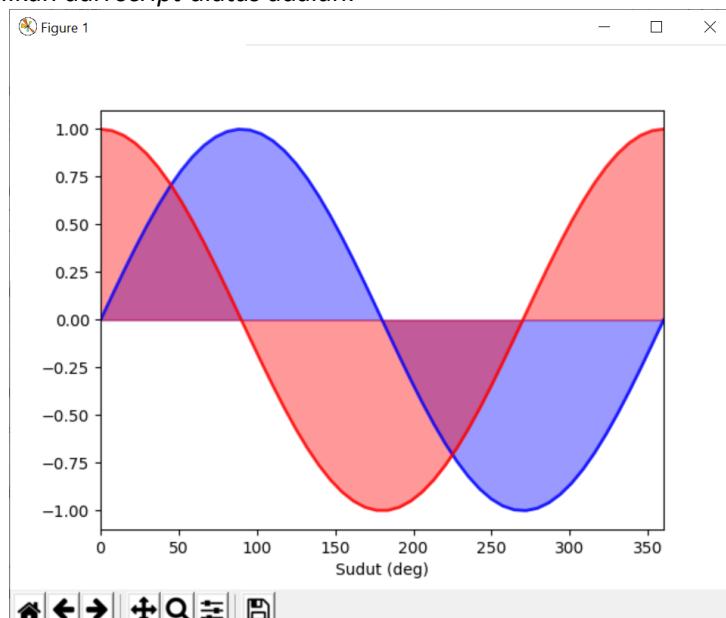


Gambar 41 – Gambar plot untuk menambahkan legend

Contoh plotting untuk menunjukkan area:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0, 360)
a = np.sin(x*np.pi/180)
b = np.cos(x*np.pi/180)
plt.plot(x,a, color='b', linewidth='2', alpha=0.8)
plt.plot(x,b, color='r', linewidth='2', alpha=0.8)
plt.fill_between(x,a, color="b", alpha=0.4)
plt.fill_between(x,b, color="r", alpha=0.4)
plt.xlabel('Sudut (deg)')
plt.xlim(0,360)
plt.show()
```

Gambar yang dihasilkan dari script diatas adalah:



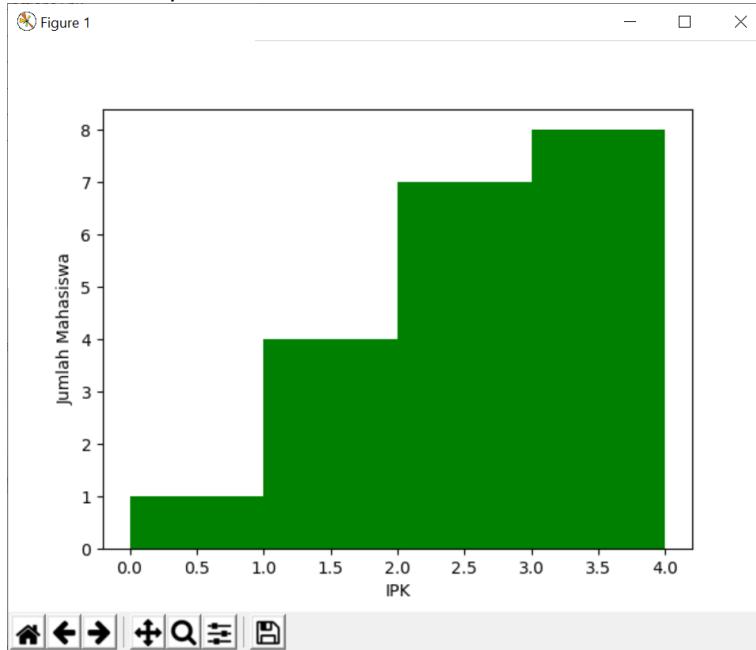
Gambar 42 – Gambar plot untuk menunjukkan area

Contoh *plotting* untuk menampilkan **histogram**:

```
import matplotlib.pyplot as plt

ipk = [3.88, 3.54, 2.87, 3.24, 1.67, 2.11, 3.87, 1.01, 2.08,
       1.88, 3.98, 2.76, 3.50, 4.00, 0.54, 2.89, 3.17, 1.33,
       2.48, 2.22]
plt.hist(ipk, bins=[0,1,2,3,4], facecolor='g')
plt.xlabel('IPK')
plt.ylabel('Jumlah Mahasiswa')
plt.show()
```

Gambar yang dihasilkan dari *script* diatas adalah:



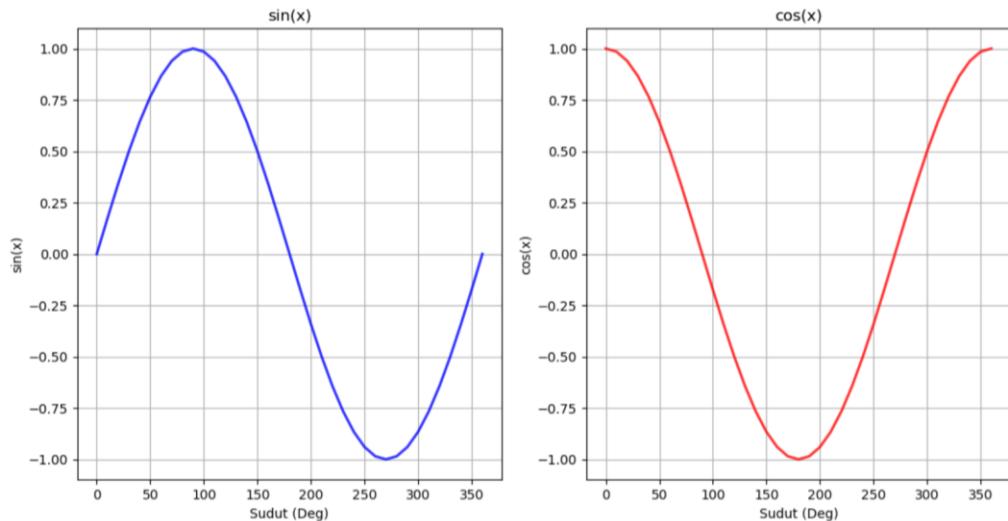
Gambar 43 – Gambar plot untuk menampilkan histogram

Contoh membuat **subplot**:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(0,361,10)
a = np.sin(x*np.pi/180)
b = np.cos(x*np.pi/180)

fig = plt.figure()
ax1 = fig.add_subplot(121)
ax1.plot(x,a, color='b', linewidth='2', alpha=0.8)
ax1.grid('True')
ax1.set_xlabel('Sudut (Deg)')
ax1.set_ylabel('sin(x)')
ax1.set_title('sin(x)')
ax2 = fig.add_subplot(122)
ax2.plot(x,b, color='r', linewidth='2', alpha=0.8)
ax2.grid('True')
ax2.set_xlabel('Sudut (Deg)')
ax2.set_ylabel('cos(x)')
ax2.set_title('cos(x)')
plt.show()
```

Gambar yang dihasilkan dari *script* diatas adalah:



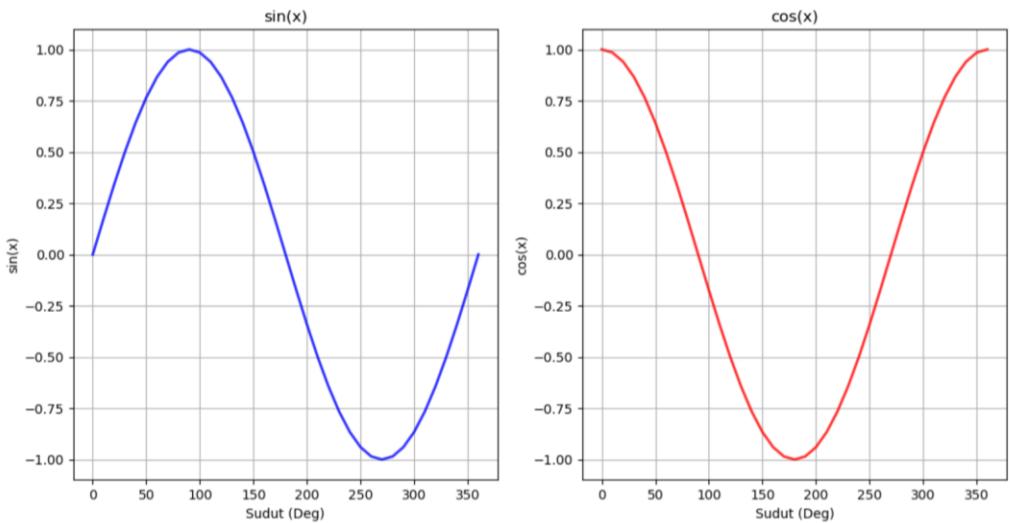
Gambar 44 – Gambar subplot (1)

Contoh membuat **subplot**:

```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(0,361,10)
a = np.sin(x*np.pi/180)
b = np.cos(x*np.pi/180)

plt.subplot(211)
plt.plot(x,a, color='b', linewidth='2', alpha=0.8)
plt.grid('True')
plt.xlabel('Sudut (Deg)')
plt.ylabel('sin(x)')
plt.title('sin(x)')
plt.subplot(212)
plt.plot(x,b, color='r', linewidth='2', alpha=0.8)
plt.grid('True')
plt.xlabel('Sudut (Deg)')
plt.ylabel('cos(x)')
plt.title('cos(x)')
plt.show()
```

Gambar yang dihasilkan dari *script* diatas adalah:



Gambar 45 – Gambar subplot (2)

Untuk visualisasi 3D, dibutuhkan modul **mpl_toolkits.mplot3d** atau **Axes3D**.

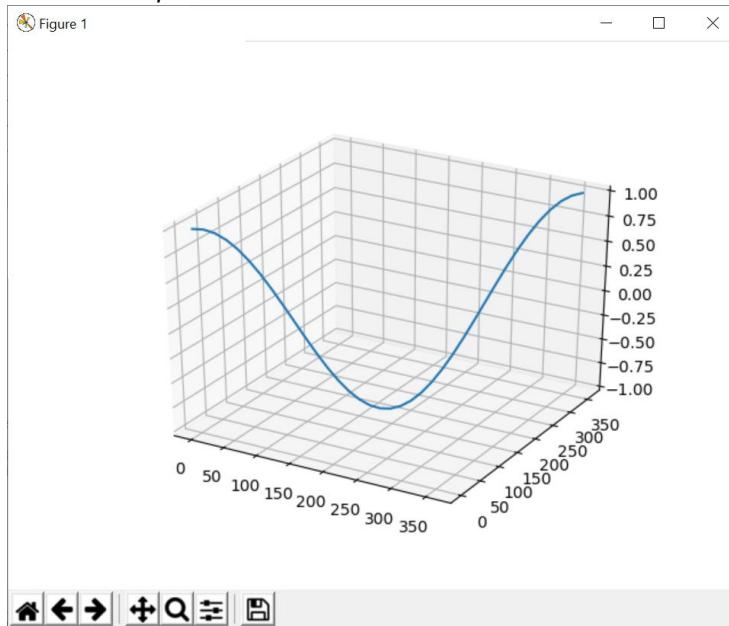
Contoh **plotting 3D**:

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

x = np.arange(0, 361, 10)
y = np.arange(0, 361, 10)
b = np.cos(x*np.pi/180)

fig = plt.figure()
ax = fig.gca(projection='3d')
ax.plot(x, y, b)
plt.show()
```

Gambar yang dihasilkan dari script diatas adalah:

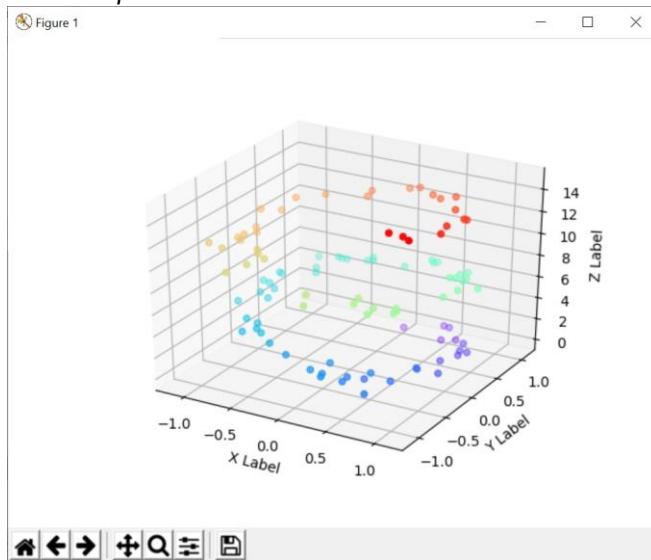


Gambar 46 – Gambar plot 3D

Contoh plotting 3D scatter:

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
zdata = 15 * np.random.random(100)
xdata = np.sin(zdata) + 0.1 * np.random.randn(100)
ydata = np.cos(zdata) + 0.1 * np.random.randn(100)
ax.scatter(xdata, ydata, zdata, c=zdata, cmap='rainbow')
ax.set_xlabel('X Label')
ax.set_ylabel('Y Label')
ax.set_zlabel('Z Label')
plt.show()
```

Gambar yang dihasilkan dari script diatas adalah:



Gambar 47 – Gambar plot 3D scatter

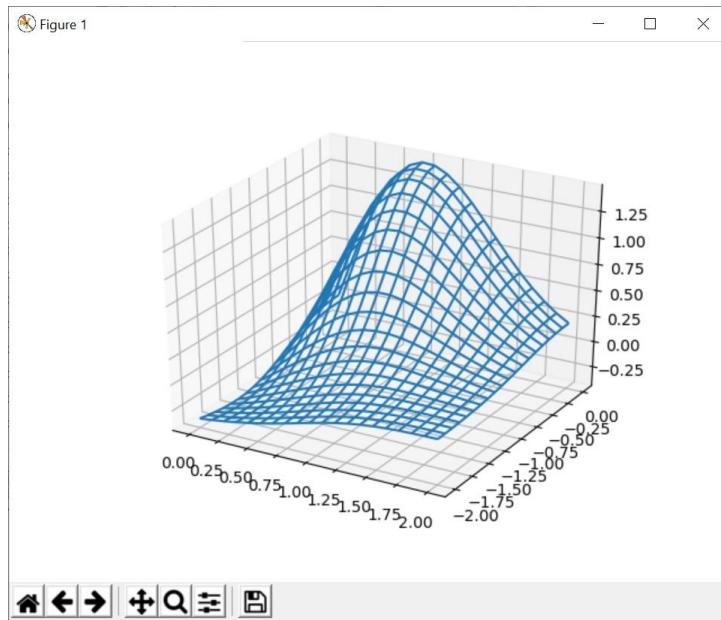
Contoh plotting 3D wireframe:

```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
import numpy as np

points = np.linspace(-2, 0, 20)
[x, y] = np.meshgrid(points, points)
# plt.plot(x, y)
# plt.show()

z = (2 / (np.exp((x-0.5)**2 + (y**2))) -
     (2 / (np.exp((x+0.5)**2 + (y**2))))
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_wireframe(x, y, z)
plt.show()
```

Gambar yang dihasilkan dari script diatas adalah:



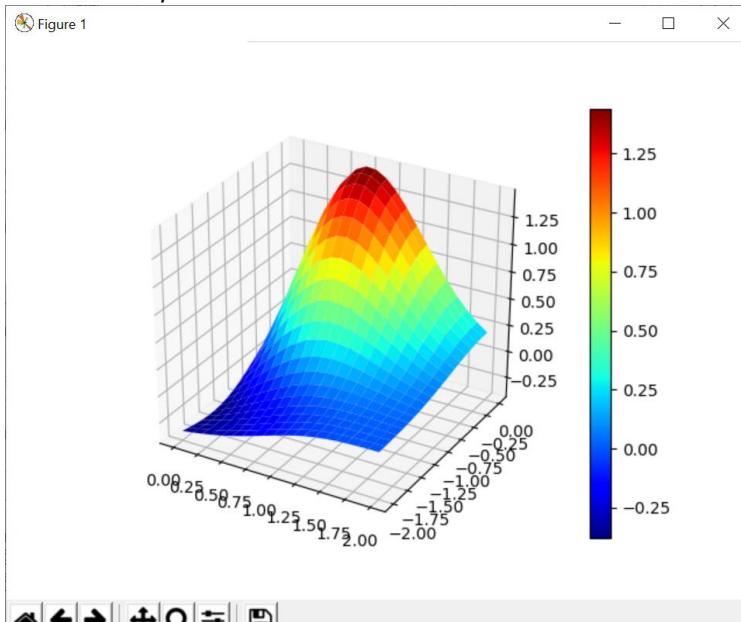
Gambar 48 – Gambar plot 3D scatter

Contoh *plotting 3D surface*:

```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
import numpy as np

points = np.linspace(-2,0,20)
[x,y]=np.meshgrid(-points,points)
z = (2 / (np.exp((x-0.5)**2+(y**2))))-
(2/(np.exp((x+0.5)**2)+(y**2)))
fig = plt.figure()
ax = fig.gca(projection='3d')
cbar=ax.plot_surface(x,y,z, cmap='jet')
fig.colorbar(cbar)
plt.show()
```

Gambar yang dihasilkan dari *script* diatas adalah:



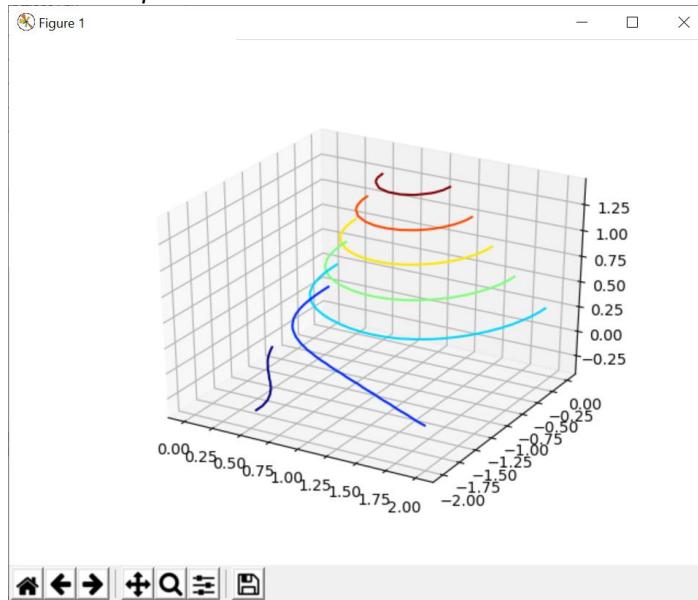
Gambar 49 – Gambar plot 3D surface

Contoh *plotting 3D contour*:

```
from mpl_toolkits.mplot3d import axes3d
import matplotlib.pyplot as plt
import numpy as np

points = np.linspace(-2,0,20)
[x,y]=np.meshgrid(-points,points)
z = (2 / (np.exp((x-0.5)**2+(y**2))))-
(2/(np.exp((x+0.5)**2)+(y**2)))
fig = plt.figure()
ax = fig.gca(projection='3d')
ax.contour(x,y,z, cmap='jet')
plt.show()
```

Gambar yang dihasilkan dari *script* diatas adalah:



Gambar 50 – Gambar plot 3D contour

Statistics

Statistics digunakan untuk mengumpulkan dan menganalisis data. Berikut adalah fungsi statistika menggunakan perintah dalam Python.

mean()	<i>Arithmetic mean</i> atau rata-rata dari sekumpulan data
harmonic_mean()	<i>Harmonic mean</i> atau rata-rata secara Pythagoras
median()	Nilai tengah suatu data
median_low()	Nilai median terkecil
median_high()	Nilai median terbesar
median_grouped()	Nilai median dalam grup ($50^{\text{th}} \text{ percentile}$)
mode()	Nilai yang paling sering muncul
min()	Nilai minimum dari sekumpulan data
max()	Nilai maksimum dari sekumpulan data
pstdev()	Standar deviasi dari suatu data populasi
pvariance()	Variansi dari suatu data populasi
stdev()	Standar deviasi dari suatu data sampel
variance()	Variansi dari suatu data sampel

Berikut adalah perbandingan persamaan *mean*, *standar deviation* dan *variance* pada sampel dan populasi:

	Population	Sample
Mean	$\mu = \frac{\sum_{i=1}^n x_i}{n}$	$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$
Variance	$\sigma^2 = \frac{\sum_{i=1}^n (x_i - \mu)^2}{n}$	$s^2 = \frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}$
Standard Deviation	$\sigma = \sqrt{\sigma^2}$	$s = \sqrt{s^2}$

Pada Python, analisis diatas dipanggil dengan modul *Numpy* atau *Statistics*, seperti contoh dibawah ini:

```
import statistics as sta
import numpy as np

print(np.mean([-1.0, 2.5, 3.25, 5.75]))
print(sta.mean([-1.0, 2.5, 3.25, 5.75]))
print(sta.median([1, 3, 5, 7]))
print(sta.median_low([1, 3, 5, 7]))
print(sta.median_high([1, 3, 5, 7]))
print(sta.mode([1, 1, 2, 3, 3, 3, 3, 4]))
print(np.min([1, 3, 5, 7]))
print(sta.variance([-1.0, 2.5, 3.25, 5.75]))
print(sta.stdev([-1.0, 2.5, 3.25, 5.75]))
```

Linear Regression

Regresi linear merupakan pendekatan secara linear untuk memodelkan hubungan antara suatu variabel dengan variabel lainnya. Persamaan regresi linear adalah $y = a + bx$, dimana a merupakan *intercept* atau perpotongan kurva dengan sumbu y (pada saat $x = 0$), sedangkan b merupakan gradien yang menunjukkan besar hubungan variabel x dan y .

Untuk menghitung nilai a dan b , digunakan persamaan berikut:

$$b = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^n (x_i - \bar{x})^2}$$

$$a = \bar{y} - b \bar{x}$$

\bar{x}, \bar{y} adalah rata-rata x dan y

Contoh perhitungan regresi linear pada Python adalah:

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

x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12, 13])

m_x, m_y = np.mean(x), np.mean(y)

SS_xy = np.sum((x-m_x)*(y-m_y))
SS_xx = np.sum((x-m_x)**2)
SS_yy = np.sum((y-m_y)**2)
b = SS_xy/ SS_xx
a = m_y - b*m_x
print(a,b)
```

Namun kita juga dapat menggunakan modul **Scikit Learn**, yang merupakan *tools* untuk *Machine Learning* pada Python.

Berikut adalah *script* untuk melakukan regresi linear menggunakan Scikit Learn:



Gambar 51 – Logo Scikit-Learn

```

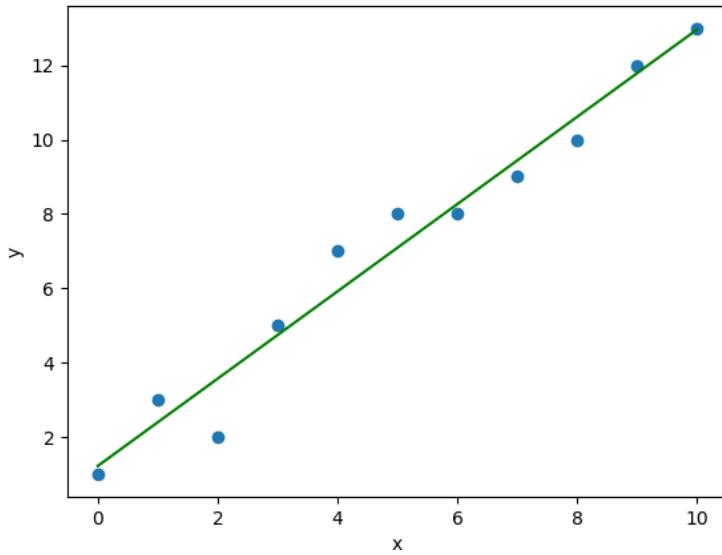
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

x = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]).reshape((-1, 1))
y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12, 13])

model = LinearRegression().fit(x,y)
r_sq = model.score(x,y)
print(r_sq)
print('intercept',model.intercept_)
print('gradient', model.coef_)
y_pred=model.predict(x)

plt.scatter(x,y)
plt.plot(x,y_pred, 'g')
plt.xlabel('x')
plt.ylabel('y')
plt.show()

```



Gambar 52 – Regresi linear untuk x dan y (garis) dari data yang diketahui (titik)

Class

Class digunakan untuk mem-*bundle* beberapa data dan fungsi.

```
class matematika:  
    def __init__(self,a,b):  
        self.x=a  
        self.y=b  
    def tambah(self):  
        return self.x + self.y  
    def kurang(self):  
        return self.x - self.y  
    def pangkat(self):  
        return self.x ** self.y  
    def bagi(self):  
        return self.x / self.y
```

Pada contoh diatas, dibuat kelas (*class*) matematika, dengan fungsi tambah, kurang, pangkat dan bagi antara x dan y. Berikut adalah hasil dari kelas diatas:

```
hasil=matematika(2,3)  
print(hasil.tambah())  
>>5  
  
print(hasil.kurang())  
>>-1  
  
print(hasil.pangkat())  
>>8  
  
print(hasil.bagi())  
>>0.6666666666666666
```

Random Numbers

Python dapat membuat angka acak dengan modul *random*.

```
import random as rd
for i in range(10):
    print(rd.random())
>>
0.525661104277393
0.6821396356035507
0.17892510369884296
0.6566288422784823
0.4785437517779487
0.4544502640164714
0.7920370707561407
0.3188332449133626
0.07782171437706942
0.7655225175250954
```

Pada contoh diatas adalah pembuatan angka acak sebanyak 10 angka (didefinisikan dalam *range*) yang merupakan angka *float* dengan nilai antara 0.0 dan 1.0 dengan *command* *rd.random()* .

Terdapat berbagai jenis modul yang dapat digunakan untuk *random*, salah satu lainnya adalah untuk memilih salah satu elemen dari suatu *sequence*, dengan *command* *rd.choice()* .

```
import random as rd
names = ['a', 'b', 'c']
for i in range(1):
    print(rd.choice(names))
>>b
```

Berikutnya adalah pembuatan *noise* dari bilangan acak 0.0 hingga 0.01 sebanyak 360 (sebagai sudut dalam derajat), yang kemudian ditampilkan dalam kurva sinusoidal $0^\circ - 360^\circ$.

```

import random as rd

noise=[]
for i in range(360):
    noise.append(0.1*rd.random())

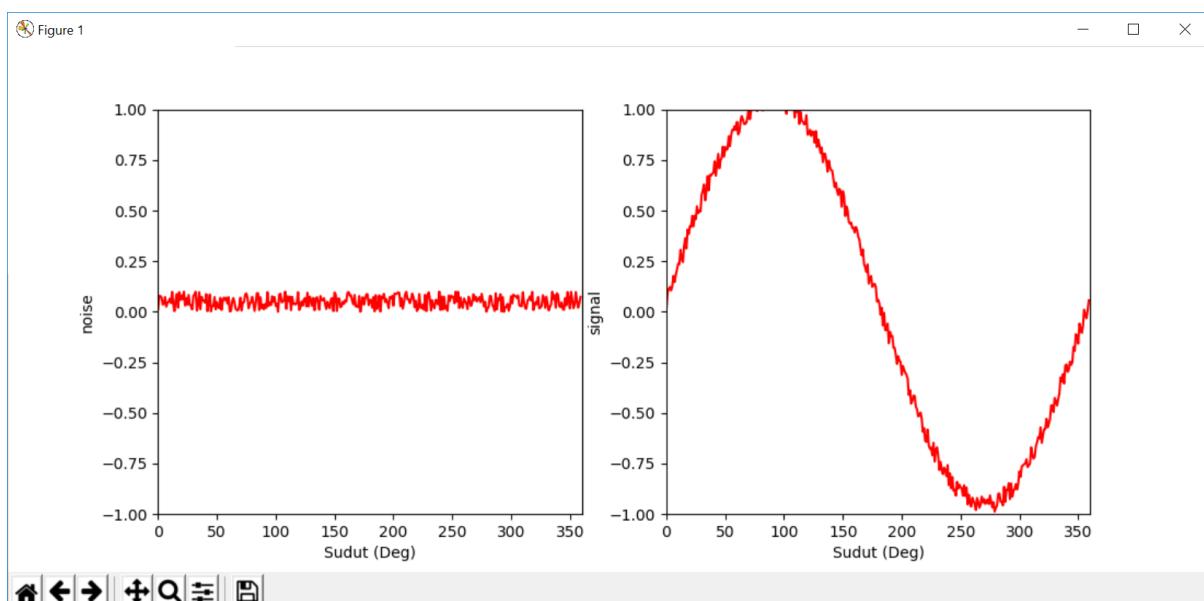
import numpy as np
import matplotlib.pyplot as plt

x = np.arange(0,360)
y = np.sin(x*np.pi/180)

plt.subplot(1,2,1)
plt.plot(x,noise, color='r')
plt.xlabel('Sudut (Deg)')
plt.ylabel('noise')
plt.xlim(0,360)
plt.ylim(-1,1)

plt.subplot(1,2,2)
plt.plot(x,y+noise, color='r')
plt.xlabel('Sudut (Deg)')
plt.ylabel('signal')
plt.xlim(0,360)
plt.ylim(-1,1)
plt.show()
plt.xlim(0,360)
plt.ylim(1,-1)
plt.show()

```



Gambar 53 - Hasil plotting sinusoidal dengan noise bilangan acak

Convolution

Konvolusi adalah operasi matematika pada dua deret angka untuk menghasilkan deret angka ketiga. Secara diskrit, deret ketiga (c) yang dihasilkan dari deret a dan b adalah seperti berikut:

$$c[k] = \sum_{n=0}^{n+k} a[k]b[k-n]$$

Misalkan:

$$a = [1, 2, 3]$$

$$b = [4, 5, 6]$$

Konvolusi dilambangkan dengan *asterisk* (*) maka konvolusi a terhadap b adalah $a * b = c$.

Fungsi b :	4 5 6			
Fungsi a	3 2	1	(4x1)	= 4
dicerminkan:	3	2 1	(4x2) + (5x1)	= 13
		3 2 1	(4x3) + (5x2) + (6x1)	= 28
		3 2 1	(5x3) + (6x2)	= 27
		3 2 1	(6x3)	= 18

Hasil konvolusi $a * b = c$.

$$[1, 2, 3] * [4, 5, 6] = [4, 13, 28, 27, 18]$$

Dengan panjang dimensi a (3) dan b (3) serta c (6), maka dapat disimpulkan bahwa:
panjang dimensi $c = \text{panjang dimensi } a + \text{panjang dimensi } b - 1$

Pada Python, konvolusi dapat dilakukan dengan command `np.convolve(a, b)`

```
import numpy as np
a=[1,2,3]
b=[4,5,6]
c=np.convolve(a,b)

print(c)
>>
[ 4 13 28 27 18]
```

Ricker Wavelet

Ricker wavelet adalah *wavelet zero-phase* sebagai turunan kedua dari fungsi Gaussian dan turunan ketiga dari fungsi densitas probabilitas normal. Berasal dari nama geofisisis asal Amerika, Norman H. Ricker (1896–1980) dan disebut juga dengan *Mexican hat wavelet*.

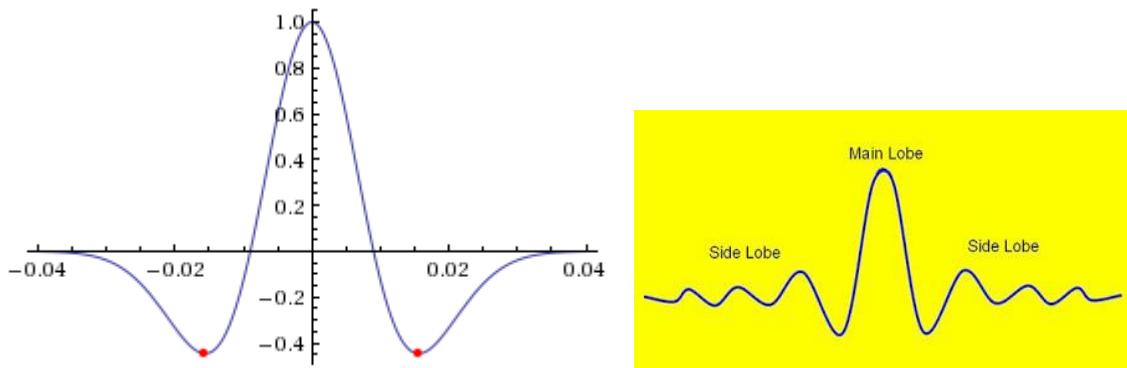
Wavelet sendiri berarti suatu osilasi gelombang pada saat amplitudo berasal dari nol, kemudian bertambah dan kembali ke nol. *Wavelet* merupakan *output* dari operator matematika yang merepresentasikan sinyal yang bervariasi terhadap waktu dan frekuensi.

Besar amplitudo dari *Ricker wavelet* pada frekuensi maksimum dengan waktu tertentu adalah:

$$A = (1 - 2\pi^2 f^2 t^2) e^{-\pi^2 f^2 t^2}$$

Wavelet memiliki bagian utama (*main lobe*) dan bagian samping (*side lobe*). *Wavelet* akan semakin baik jika *sidelobe* minima dan bagian *main lobe* dominan. *Side lobe* memberikan efek *noise* pada rekaman seismik. Pada *Ricker wavelet*, *sidelobe minima* berada pada $\pm \frac{\sqrt{3/2}}{\pi f}$ dengan nilai amplitudo:

$$A_{min} = -\frac{2}{e^{3/2}}$$



Gambar 54 - Contoh Ricker Wavelet

Sumber:

Kiri: http://subsurfwiki.org/wiki/Ricker_wavelet

Kanan: <http://ensiklopediseismik.blogspot.com/2007/09/main-lobeside-lode.html>

```

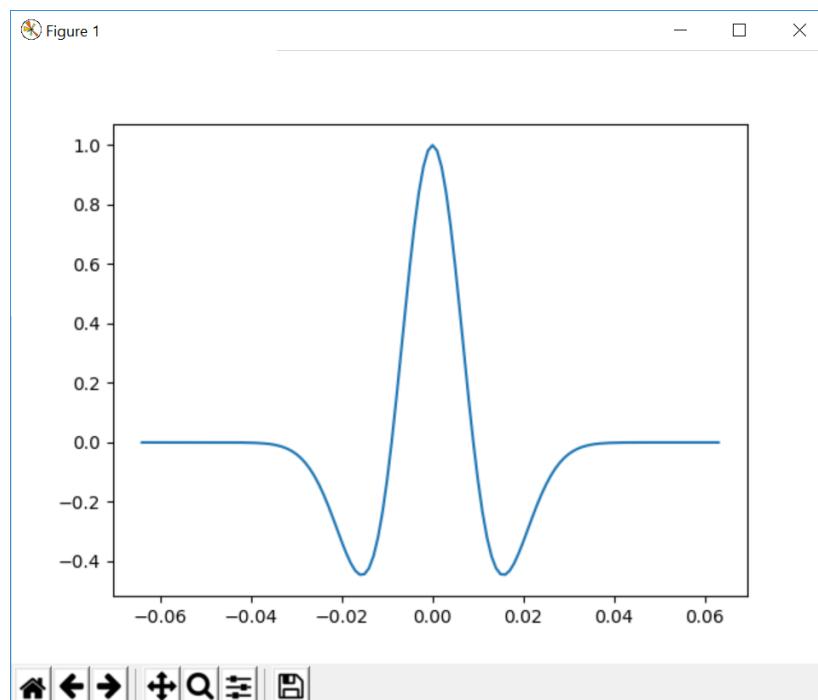
import numpy as np
import matplotlib.pyplot as plt

def ricker(f, length=0.128, dt=0.001):
    t = np.arange(-length / 2, (length - dt) / 2, dt)
    y = (1.0 - 2.0 * (np.pi ** 2) * (f ** 2) * (t ** 2)) *
        np.exp(-(np.pi ** 2) * (f ** 2) * (t ** 2))
    return t, y

f = 25 # A low wavelength of 25 Hz
t, w = ricker(f)

plt.plot(t,w)
plt.show()

```



Gambar 55 - Ricker Wavelet



BAB III

Penggunaan Python dalam Komputasi Geofisika

Metode *Least-Square* untuk Menentukan *b-value*

Jumlah kejadian (frekuensi) gempabumi dan magnitudo dapat dihubungkan dengan suatu gradien logaritmik yang dinamakan dengan *b-value*. Hal ini berdasarkan Hukum Gutenberg-Richter (*GR Law*) yang dipublikasikan oleh Charles F. Richter dan Beno Gutenberg pada tahun 1956, dengan persamaan sebagai berikut:

$$\log_{10} N = a - bM$$
$$N = 10^{a-bM}$$

Eq. 1

Dengan N merupakan jumlah kejadian gempa bumi (frekuensi terjadinya gempabumi) dengan magnitudo $\geq M$. Nilai a dan b merupakan konstanta, dimana nilai b merupakan gradien, sedangkan a merupakan *intercept* dari persamaan diatas.

a-value merupakan seismisitas (aktivitas seismik) total, atau jumlah seluruh kejadian gempa dalam kurun waktu yang ditentukan. Sedangkan jumlah kejadian gempa untuk magnitudo tertentu, dapat dihitung dari persamaan berikut:

$$N_{TOT} = 10^a$$
$$N = N_{TOT} 10^{-bM}$$

Eq. 2

Sementara *b-value* menunjukkan karakter tektonik pada suatu daerah. Nilainya bervariasi tergantung kedalaman fokus (gempabumi), heterogenitas batuan dan distribusi *stress* pada volume batuan sumber gempabumi. Selaku gradien, *b-value* menunjukkan aktivitas tektonik pada daerah tersebut. Semakin tinggi *b-value*, maka semakin sering terjadi gempabumi di daerah tersebut. Hal ini ditunjukkan dengan “kecuraman” gradien kemiringan dengan jumlah kejadian yang banyak pada magnitudo kecil, yang bisa merupakan gempa kecil atau mikroseismik, atau *swarm*.

b-value juga digunakan sebagai *precursor* atau fenomena yang dapat memberikan peringatan akan terjadinya gempabumi. Nilai yang rendah mengindikasikan bahwa di daerah tersebut memiliki *stress* atau akumulasi energi yang tinggi, yang menandakan bahwa di daerah tersebut belum terjadi pelepasan energi melalui gempabumi. Sehingga masyarakat perlu waspada terhadap wilayah-wilayah yang memiliki *b-value* yang rendah.

Dengan mengetahui bahwa nilai a dan b merupakan nilai *intercept* dan gradien dari persamaan linear $\log N = a - bM$, maka nilai a dan b dapat diperoleh dengan regresi linear, salah satunya adalah dengan menggunakan Metode *Least-Square* atau Metode Kuadrat-Terkecil.

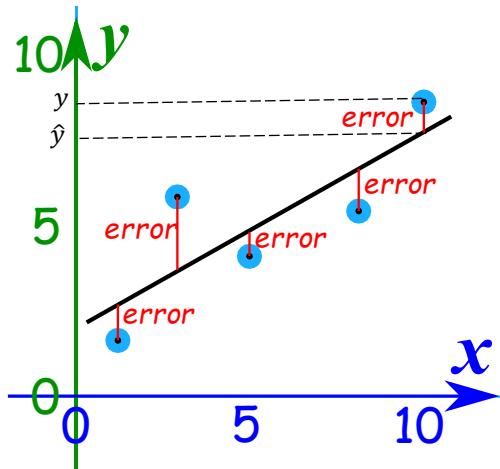
Metode *Least-Square* merupakan solusi umum dengan meminimalisasi jumlah dari kuadrat residual, dengan persamaan umum seperti berikut:

$$S = \sum_{i=1}^n r_i^2$$

Eq. 3

S adalah jumlah dari kuadrat residual.

Residual adalah perbedaan antara nilai sebenarnya dengan nilai prediksi model. Jika suatu model regresi linear menyebutkan nilai dari hasil perhitungan, dan nilainya berbeda dari nilai sebenarnya, maka akan diperoleh residual dengan persamaan berikut:



Gambar 56 - Residual (error) dari true value dan predicted value (Sumber: <https://www.mathsisfun.com/data/least-squares-regression.html>)

$$r = y - \hat{y}$$

Eq. 4

r adalah nilai residual, y adalah nilai sebenarnya (*true value*) dan \hat{y} adalah nilai prediksi model (*predicted value*).

Model yang digunakan memiliki persamaan $\hat{y}_i = mx_i + c$, sehingga persamaan residual menjadi $r_i = \sum_{i=1}^n [y_i - (mx_i + c)]^2$ dan dengan menggunakan turunan parsial r terhadap m dan c maka diperoleh:

$$\begin{bmatrix} n & \sum_{i=1}^n x_i \\ \sum_{i=1}^n x_i & \sum_{i=1}^n x_i^2 \end{bmatrix} \begin{bmatrix} c \\ m \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^n y_i \\ \sum_{i=1}^n x_i y_i \end{bmatrix}$$

Eq. 5

Sehingga dapat dihitung m dan c dengan persamaan berikut:

$$m = \frac{n \sum xy - \sum x \sum y}{n \sum x^2 - (\sum x)^2}$$

$$c = \frac{\sum y - m \sum x}{n}$$

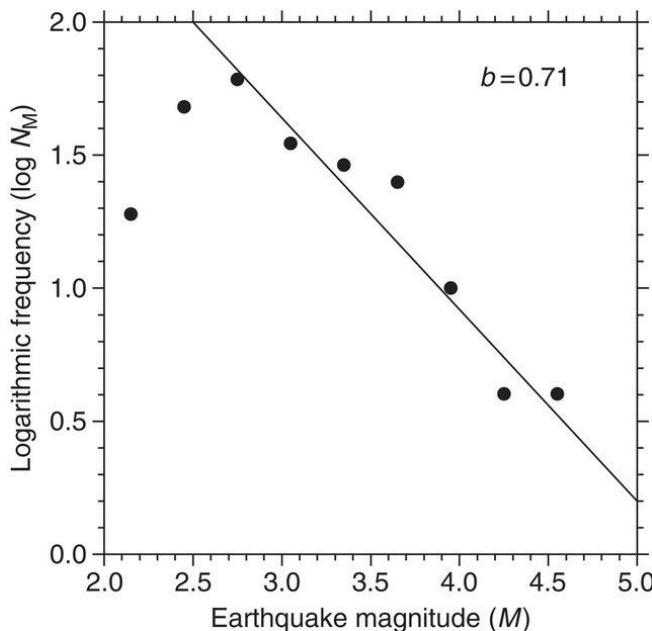
Eq. 6

Dengan keterangan bahwa n adalah jumlah data. Persamaan $y = mx + c$ diatas dapat disesuaikan dengan persamaan perhitungan b -value $\log_{10} N = a - bM$ dengan $\log N = y$ dan $M = x$, kemudian $a = c$ dan $b = -m$, sehingga dapat ditulis perhitungan a -value dan b -value sebagai berikut:

$$b = -\left(\frac{n \sum M \log N - \sum M \sum \log N}{n \sum M^2 - (\sum M)^2} \right)$$

$$a = \frac{\sum \log N + b \sum M}{n}$$

Eq. 7



Gambar 57 - Contoh grafik hubungan magnitudo dan frekuensi kejadian gempabumi

Namun dalam pemrograman Python, perhitungan dengan metode *Least-Square* dapat dengan menggunakan *Scipy* khususnya `scipy.sparse.linalg` (`linalg` = linear algebra) dengan meng-import modul `lsqr`. Selain itu, juga bisa dihitung dengan menggunakan `polyfit` atau `polynomial curve fitting` sebagai proses menemukan *fit* terbaik untuk setiap titik data.

Contoh *script* untuk `lsqr`:

```
import numpy as np
from scipy.sparse.linalg import lsqr

data = np.genfromtxt('data_bvalue.dat')
magnitude = data[:, 0]
frequency = data[:, 1]
M = np.array(magnitude)
logN = np.array(np.log10(frequency))
M = np.column_stack((M, np.ones(np.shape(M)[0])))
grad = lsqr(M, logN)[0]
a = grad[1]
b = grad[0]**-1
print('a-value is: ', '%.2f' %a)
print('b-value is: ', '%.2f' %b)
```

`data_bvalue.dat` merupakan data magnitudo dan jumlah kejadian gempabumi dengan keterangan sebagai berikut:

Magnitudo	Frekuensi Kejadian
3	20
4	10
5	8
6	5
7	2

Dengan menggunakan persamaan $\log_{10} N = a - bM$, kita sudah memiliki data M untuk nilai magnitudo dan N untuk frekuensi kejadian gempa. Metode *Least-Square* pada Python memiliki prinsip $A * x = b$ dimana x merupakan nilai gradien dan *intercept*. Dalam hal ini, A merupakan magnitudo (M) dan b merupakan $\log_{10} N$ dan x merupakan nilai a dan $-b$.

Sehingga jika x diperoleh dari $x = \text{lsqr}(A, b)$, maka dengan persamaan $\log_{10} N = a - bM$ diperoleh dengan $\text{lsqr}(M, \log N)$.

Hasil *output* dari *script* diatas:

```
a-value is: 1.99
b-value is: 0.23
```

Selain dengan menggunakan *lsqr*, nilai a dan b juga dapat diperoleh dengan menggunakan *polyfit* seperti contoh sebagai berikut:

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

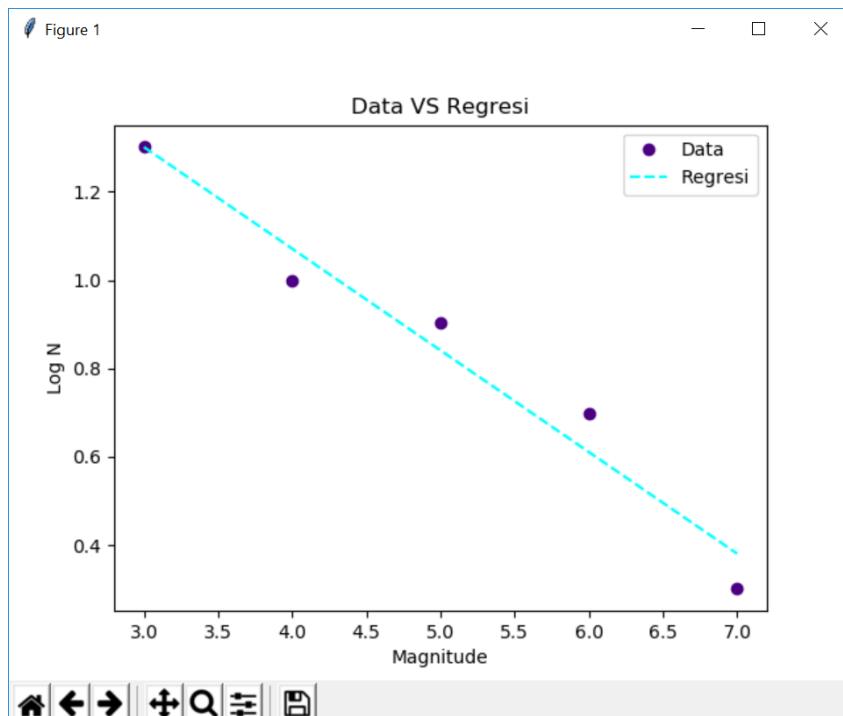
# data awal
data = np.genfromtxt('data_bvalue.dat')
x = data[:, 0]
y = data[:, 1]

# nilai a dan b-value
p = np.polyfit(x, np.log10(y), 1)
print('Least Square : b value: ', '%.2f' % -p[0], 'a value: ',
      '%.2f' % p[1])

# perbandingan data awal vs regresi
y_akhir = p[0]*x+p[1]
plt.plot(x, np.log10(y), 'ko', label='Data', color='indigo')
plt.plot(x, y_akhir, '--', label='Regresi', color='cyan')
plt.xlabel('Magnitude')
plt.ylabel('Log N')
plt.legend()
plt.title('Data VS Regresi')
plt.show()
```

```
Least Square : b value: 0.23 a value: 1.99
```

Figure 1

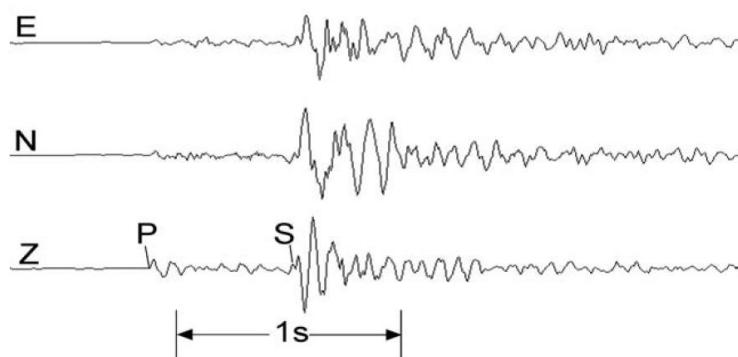


Gambar 58 - Regresi linier menggunakan Least-Square untuk memperoleh a - dan b -value

Penentuan Lokasi Gempabumi dengan Metode Tiga Lingkaran

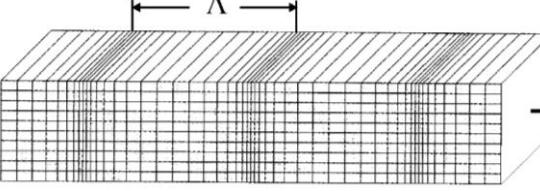
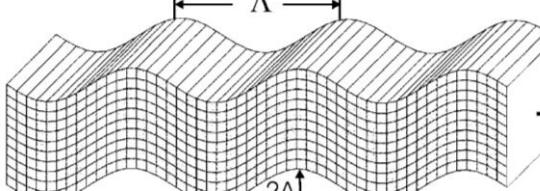
Pada saat terjadi gempabumi, seismograf atau stasiun akan merekam gelombang yang kemudian menunjukkan adanya kenaikan amplitudo yang signifikan yang ditunjukkan oleh seismogram dari 3 (tiga) arah, yaitu vertikal dari sumber gempabumi menuju stasiun (komponen Z), dan horizontal searah Timur-Barat (komponen E-W) dan searah Utara-Selatan (komponen N-S).

Hal yang pertama dilakukan adalah pendekripsi fasa gelombang P dan S, terutama pada komponen Z (vertikal) untuk menentukan lokasi gempabumi. Kemudian dengan menggunakan kurva waktu tempuh (*Travel-Time Table*), maka dapat diperoleh hubungan antara jarak gempa dan stasiun dengan interval antara waktu tiba gelombang P dan S pada stasiun.



Gambar 59 - Contoh Rekaman Seismogram untuk Tiga Komponen (Sumber: NMSOP)

Sedikit pengenalan mengenai gelombang P dan S, sebagai gelombang seismik yang merambat di bumi.

Gelombang P	Gelombang S
	
Gelombang P berasal dari kata <i>Primary</i> , gelombang yang merambat paling cepat sehingga datang lebih dulu ke permukaan bumi. Gelombang ini mirip dengan gelombang suara, merambat secara longitudinal atau compressional , dimana arah gerak partikel sejajar dengan arah perambatan gelombang. Pada kerak bumi, kecepatan gelombang P adalah sekitar 6 km/s.	Gelombang S berasal dari kata <i>Secondary</i> . Gelombang ini memiliki gerakan yang sama dengan gelombang tali, merambat secara transversal atau shear , dimana arah gerak partikel tegak lurus dengan arah perambatan gelombang. Pada kerak bumi, kecepatan gelombang S adalah sekitar 3.5 km/s.

Untuk menentukan lokasi gempabumi, kita tidak dapat menggunakan data jika kurang dari tiga stasiun penerima. Jarak antara gempabumi dan stasiun merupakan suatu radius dimana stasiun sebagai pusat lingkarannya. Jika kita hanya menggunakan satu data, maka kita punya banyak kemungkinan lokasi gempabumi. Jika kita hanya menggunakan dua data, kita menemukan dua kemungkinan lokasi

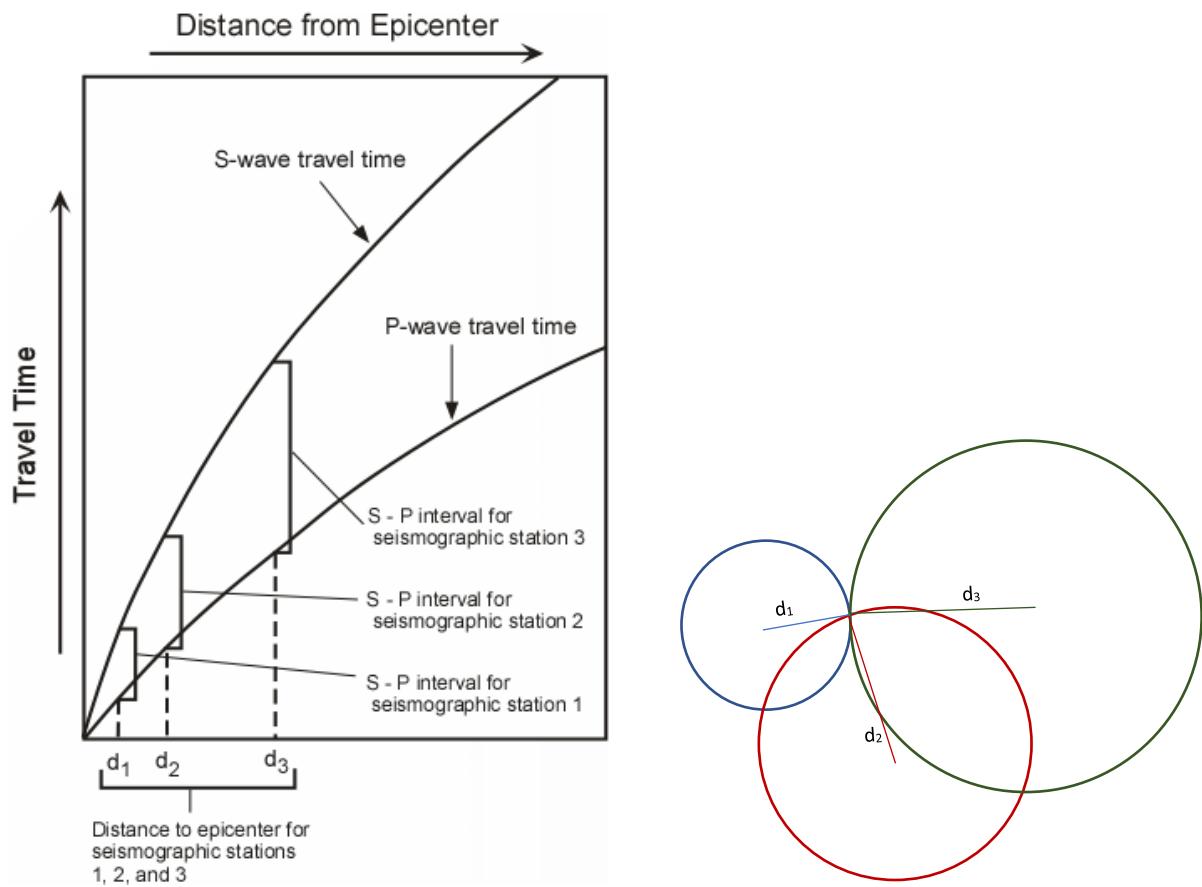
gempabumi, sebagai dua titik potong pertemuan antara dua lingkaran. Namun jika kita menggunakan tiga lingkaran, kita dapat menentukan satu lokasi yang mendekati akurat.

Berikut adalah persamaan perhitungan jarak antara gempabumi dan stasiun:

$$\Delta = (t_s^{arr} - t_p^{arr}) \frac{v_p v_s}{v_p - v_s}$$

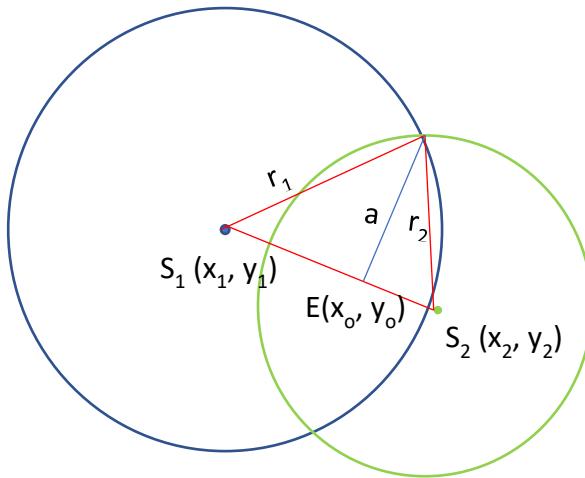
Eq. 8

Δ merupakan jarak antara gempa dan stasiun, t_p^{arr} adalah waktu tiba (*arrival time*) gelombang P dan v_p adalah kecepatan gelombang P, sedangkan t_s^{arr} adalah waktu tiba (*arrival time*) gelombang S dan v_s adalah kecepatan gelombang S. Persamaan $\frac{v_p v_s}{v_p - v_s}$ disebut juga dengan konstanta Omori (K), dimana untuk kerak bumi, K memiliki nilai 8.



Gambar 60 - Hubungan antara jarak gempa-stasiun dengan interval S-P (Sumber:
<https://www.tulane.edu/~sanelson/eens1110/earthint.htm>)

Secara umum, untuk persamaan titik pertemuan antara dua lingkaran, adalah sebagai berikut:



Gambar 61 - Ilustrasi pertemuan titik dua lingkaran

Misalnya diketahui dua lingkaran S_1 dan S_2 dengan jari-jari r_1 dan r_2 , memiliki titik pertemuan E . Jika diketahui bahwa $r_1^2 = a^2 + (\sqrt{(x_1 - x_o)^2 + (y_1 - y_o)^2})^2$ dan $r_2^2 = a^2 + (\sqrt{(x_2 - x_o)^2 + (y_2 - y_o)^2})^2$, Maka lokasi titik temu E dapat diperoleh dari persamaan sebagai berikut:

$$(r_1^2 - r_2^2) - (x_1^2 - x_2^2) - (y_1^2 - y_2^2) = -2x_o(x_1 - x_2) - 2y_o(y_1 - y_2)$$

Eq. 9

Sehingga jika kita memiliki tiga lingkaran, persamaan tersebut akan menjadi:

$$(r_i^2 - r_{i+1}^2) - (x_i^2 - x_{i+1}^2) - (y_i^2 - y_{i+1}^2) = -2x_o(x_i - x_{i+1}) - 2y_o(y_i - y_{i+1})$$

Eq. 10

Jika $i = 1, 2, \dots, n - 1$.

Jika $i = n$, maka $i+1 = 1$.

Kemudian kita ubah ke dalam persamaan linear berikut:

$$b_i = c_i x_o + d_i y_o$$

Eq. 11

Yang kita peroleh dari:

$$b_i = (r_i^2 - r_{i+1}^2) - (x_i^2 - x_{i+1}^2) - (y_i^2 - y_{i+1}^2)$$

$$c_i = -2(x_i - x_{i+1})$$

$$d_i = -2(y_i - y_{i+1})$$

Eq. 12

Maka kita akan mendapatkan lokasi pertemuan ketiga titik di titik (x_o, y_o) , dimana disini merupakan lokasi episenter gempabumi. Metode penentuan (x_o, y_o) dari persamaan diatas, dijelaskan pada bagian "Metode Least-Square untuk menentukan b-value".

Berikut adalah *script* pemrograman pada Python:

```

import numpy as np
import matplotlib.pyplot as plt
from scipy.sparse.linalg import lsqr

Data = np.genfromtxt("datagempa.dat")
STA = ['KRL', 'BUS', 'MRT']
tp = Data[0:, 1]
ts = Data[0:, 2]
x = Data[0:, 3]
y = Data[0:, 4]
K = 8
r = (ts - tp) * K
n = len(tp)
teta = np.linspace(0, 2 * np.pi, 100)
X = []
Y = []
b = []
c = []
for i in range(0, n):
    X = (r[i] * np.cos(teta)) + x[i]
    Y = (r[i] * np.sin(teta)) + y[i]
    # create the figure
    plt.plot(X, Y, label=i+1)
    plt.title('Metode Tiga Lingkaran', color='k')
    plt.xlabel('X (km)', color='k')
    plt.ylabel('Y (km)', color='k')
    plt.tick_params(axis='both', labelcolor='#610023')
    plt.grid()
    plt.axis('equal')

    if i != n-1:
        b.append((r[i] ** 2 - r[i+1]**2) - (x[i]**2 - x[i + 1] ** 2) - (y[i] ** 2 - y[i+1]**2))
        c.append([(x[i] - x[i + 1]) * -2, (y[i] - y[i + 1]) *-2])

    elif i == n-1:
        b.append((r[0] ** 2 -r[i] ** 2) - (x[0] ** 2 - x[i] ** 2) - (y[0] ** 2 - y[i]**2))
        c.append([(x[0] - x[i]) * -2, (y[0] - y[i]) * -2])

b = np.array (b)
c = np.array (c)

# jarak epicenter
A = lsqr(c,b)
A1 = A[0][0]
A2 = A[0][1]
print('Lokasi Episenter adalah (', '%.2f' %A1, ', ', '%.2f' %A2, ') ')
plt.plot(A1,A2,'o', label= 'epicenter' )

```

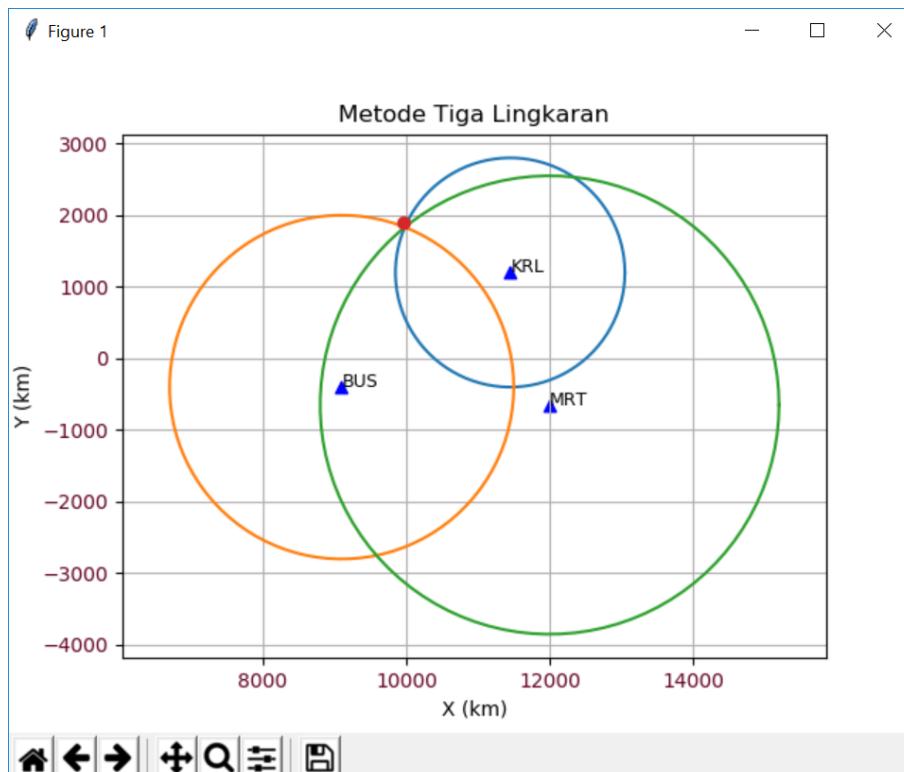
```

for j,stat in enumerate(STA):
    xl = x[j]
    yl = y[j]
    plt.scatter(xl, yl, marker='^', color='blue')
    plt.text(xl+0.3, yl+0.3, stat, fontsize=9)
plt.legend()
plt.show()

```

Misalkan data gempa yang dimiliki yang disimpan dalam **datagempa.dat** berisi informasi berikut:

STA (nama stasiun)	tp (waktu tiba gelombang P)	ts (waktu tiba gelombang S)	Lokasi Stasiun X (km)	Lokasi Stasiun Y (km)
KRL	250	450	11450	1200
BUS	350	650	9100	-400
MRT	450	850	12000	-650



Gambar 62 - Metode Tiga Lingkaran Menggunakan Python

Dari pemrograman diatas, diperoleh hasil lokasi episenter adalah:

```
Lokasi Episenter adalah ( 9962.57 , 1902.95 )
```

Forward Modeling untuk Metode Gayaberat

Prinsip dasar yang digunakan dalam metode gayaberat adalah hukum universal Newton (1687) untuk gravitasi bumi yang menyatakan bahwa gaya tarik menarik massa bumi (M_E) dan massa benda (m) yang terpisah pada jarak tertentu (r) memiliki persamaan:

$$\vec{F} = G \frac{M_E \cdot m}{r^2} \hat{r}$$

Eq. 13

Dimana F adalah gaya antara dua partikel tersebut (gaya gravitasi), G adalah konstanta gravitasi universal ($6,673 \times 10^{-11} m^3 \cdot kg^{-1} \cdot s^{-2}$), M_E adalah massa bumi dan m adalah massa benda m tersebut, r adalah jarak antara pusat bumi dengan massa m , dan \hat{r} adalah satuan vektor bumi ke benda m . Karena $F = m \cdot g$ maka percepatan gravitasi diperoleh sebagai:

$$g = \frac{G \cdot M_E}{r^2}$$

Eq. 14

Metode gayaberat sendiri digunakan untuk mengukur variasi medan gravitasi pada lokasi tertentu di permukaan bumi, untuk menentukan densitas di bawah permukaan. Variasi medan gravitasi dapat dilihat jelas jika suatu zona memiliki densitas yang lebih kecil atau lebih besar dibandingkan material di sekitarnya. *Forward modeling* merupakan bentuk perhitungan anomali gravitasi dari densitas yang diketahui dari bentuk sederhana yang diintegrasikan dengan menghitung volume dari anomali tersebut. Anomali gravitasi atau gayaberat berkaitan langsung dengan kontras densitas. Komputasi berikut merupakan pemodelan dari berbagai bentuk geometri untuk anomali gayaberat, berdasarkan jurnal yang ditulis oleh El-Tokhey et al, 2015 dengan judul "*Gravity and Density Relationship (Forward Modeling)*".

Perhitungan dari percepatan gravitasi menggunakan integrasi untuk gaya tarik komponen vertikal dengan persamaan matematis:

$$g_z = \int^v G \cdot \Delta\rho \cdot \frac{z}{r^3} dv$$

Eq. 15

Dimana g_z merupakan percepatan gravitasi (komponen vertikal), G adalah konstanta gravitasi universal, v adalah volume benda, $\Delta\rho$ adalah kontras densitas, z adalah kedalaman titik massa dan r adalah jarak antara titik massa dengan titik observasi. Secara umum, perhitungan pada komponen vertikal lebih simpel karena sumbu vertikal melewati pusat gravitasi dengan geometri apapun. Konsep geometri yang digunakan untuk perhitungan berikut adalah model bola (*sphere*), silinder horizontal (*horizontal cylinder*), dan lembar horizontal semi-tak hingga (*semi-infinite horizontal sheet*).

Perlu diketahui bahwa percepatan gravitasi dalam satuan *mgal* atau 10^{-5} m/s^2 . Satuan percepatan gravitas lainnya adalah 1 g atau satu kali percepatan gravitasi ($9.8 - 10 \text{ m/s}^2$) sehingga 1 mgal setara dengan 10^{-6} g , dan 1 gal sama dengan 1 cm/s^2 .

Berikut adalah persamaan matematis yang digunakan untuk *forward modeling* anomali gayaberat dengan geometri *sphere*, *horizontal cylinder* dan *semi-infinite horizontal sheet*.

Geometry	Equation
<i>Sphere</i>	$g_z(\text{mgal}) = \frac{G \cdot \Delta\rho \cdot V \cdot z}{(x^2 + z^2)^{\frac{3}{2}}}$
<i>Horizontal cylinder</i>	$g_z(\text{mgal}) = \frac{2 \cdot G \cdot \pi \cdot a^2 \Delta\rho \cdot z}{(x^2 + z^2)^2}$
<i>Semi-infinite horizontal sheet</i>	$g_z(\text{mgal}) = 2 \cdot G \cdot \Delta\rho \cdot T \cdot \left[\frac{\pi}{2} - \tan^{-1} \left(\frac{x}{z} \right) \right]$

Eq. 16

Keterangan:

g_z = percepatan gravitasi (komponen vertikal) (mgal)

G = konstanta gravitasi universal ($6,673 \times 10^{-11} \text{m}^3 \cdot \text{kg}^{-1} \cdot \text{s}^{-2}$)

$\Delta\rho$ = kontras densitas (kg/m^3)

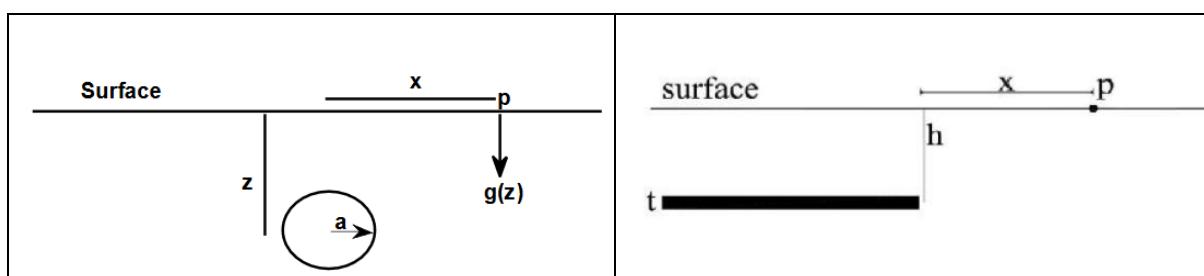
V = volume bola atau *sphere* (m^3)

z = kedalaman pusat bola atau silinder (m)

x = lokasi titik perhitungan (m)

a = radius silinder (m)

T = ketebalan (m)



Gambar 63 - Model *sphere* dan *horizontal cylinder* (kiri) dan *semi-infinite horizontal sheet* (kanan) (El-Tokhey et al, 2015)

Berikut adalah *script* untuk *forward modeling* anomali gayaberat dengan model diatas:

Catatan: parameter yang digunakan pada geometri *sphere* adalah $\Delta\rho = 1000 \text{ kg}/\text{m}^3$, radius bola untuk perhitungan volume = 50 m, $z = 100 \text{ m}$ dan x dari -500 hingga 500 m.

Parameter yang sama digunakan untuk pemodelan geometri *horizontal cylinder* dengan radius silinder = 50 m, dan juga digunakan untuk pemodelan geometri *semi-infinite horizontal sheet* dengan ketebalan 50 m.

```

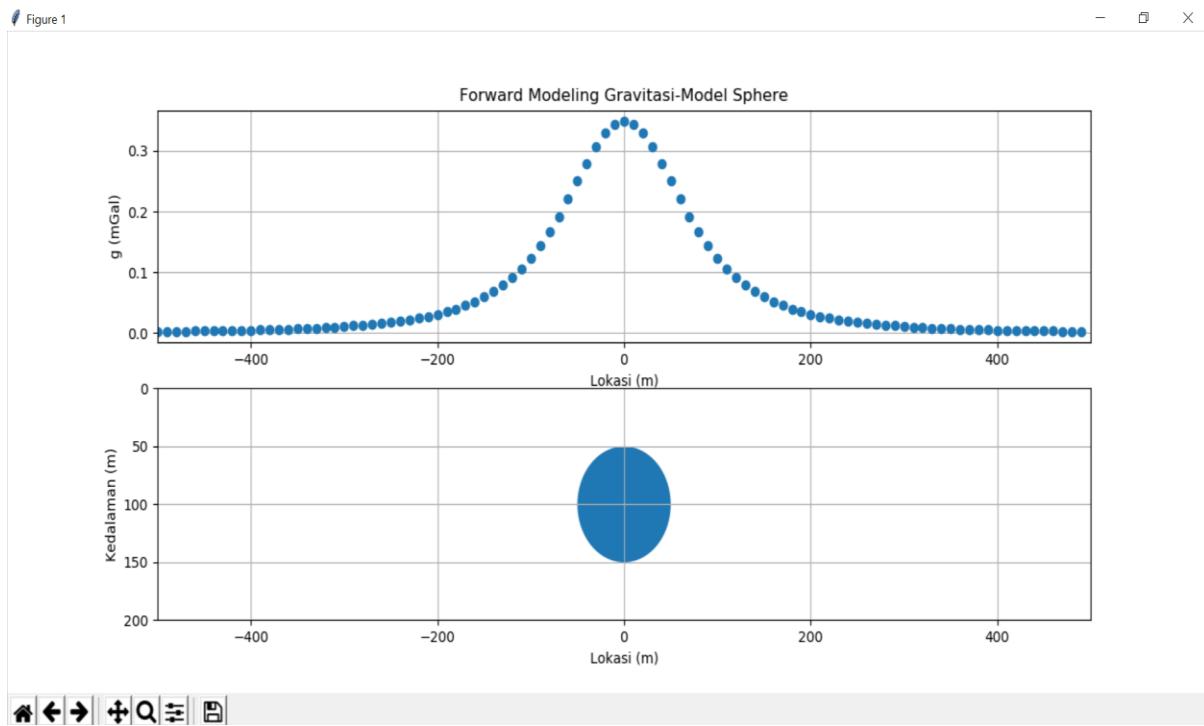
import numpy as np
import matplotlib.pyplot as plt

G = 6.673 * 10e-11
rho = float(input("Kontras densitas (kg/m^3): "))
a = float(input("Radius bola (m): "))
z = float(input("Kedalaman Titik Pusat Massa (m): "))
x = int(input("Lokasi Maksimum di Permukaan (m): "))
v = 4 / 3 * np.pi * a ** 3
p = np.arange(0, 360, 1)
q = np.array(p * np.pi / 180)
i = np.arange(-x, x, 10)
m = i ** 2 + z ** 2
n = np.array(m) ** 1.5
g = np.array(10e3 * G * rho * v * z / n)

plt.subplot(211)
plt.xlim(-x, x, 0.01)
plt.plot(i, g, "o")
plt.grid()
plt.xlabel("Lokasi (m)")
plt.ylabel("g (mGal)")
plt.title("Forward Modeling Gravitasasi-Model Sphere")

plt.subplot(212)
plt.fill(a * np.cos(q), a * np.sin(q) + z, "-")
plt.xlim(-x, x, 0.01)
plt.ylim(0, 2 * z, 0.01)
plt.gca().invert_yaxis()
plt.grid()
plt.xlabel("Lokasi (m)")
plt.ylabel("Kedalaman (m)")
plt.show()

```



Gambar 64 - Forward modeling dengan geometri sphere

```

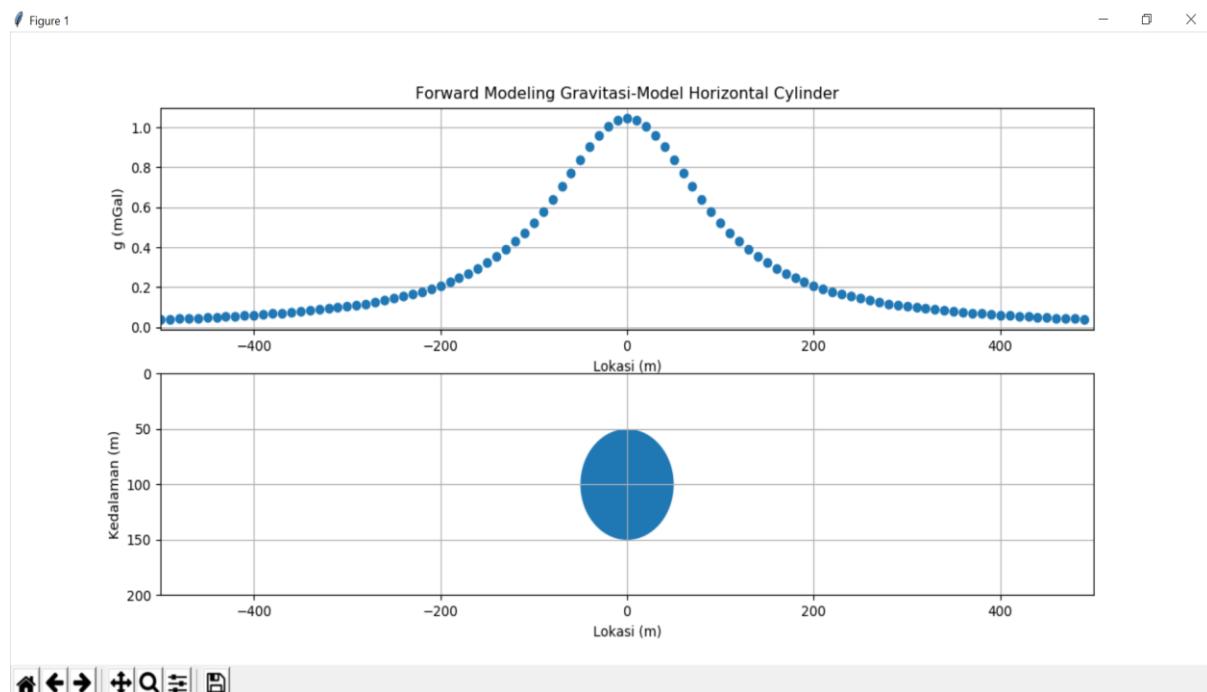
import numpy as np
import matplotlib.pyplot as plt

G = 6.673 * 10e-11
rho = float(input("Kontras densitas (kg/m^3): "))
a = float(input("Radius bola (m): "))
z = float(input("Kedalaman Titik Pusat Massa (m): "))
x = int(input("Lokasi Maksimum di Permukaan (m): "))
p = np.arange(0, 360, 1)
q = np.array(p * np.pi / 180)
i = np.arange(-x, x, 10)
m = i ** 2 + z ** 2
n = np.array(m)
g = np.array(10e3 * 2 * G * np.pi * rho * (a ** 2) * z / n)

plt.subplot(211)
plt.xlim(-x, x, 0.01)
plt.plot(i, g, "o")
plt.grid()
plt.xlabel("Lokasi (m)")
plt.ylabel("g (mGal)")
plt.title("Forward Modeling Gravitasasi-Model Horizontal Cylinder")

plt.subplot(212)
plt.fill(a * np.cos(q), a * np.sin(q) + z, "-")
plt.xlim(-x, x, 0.01)
plt.ylim(0, 2 * z, 0.01)
plt.gca().invert_yaxis()
plt.grid()
plt.xlabel("Lokasi (m)")
plt.ylabel("Kedalaman (m)")
plt.show()

```



Gambar 65 - Forward modeling dengan geometri horizontal cylinder

```

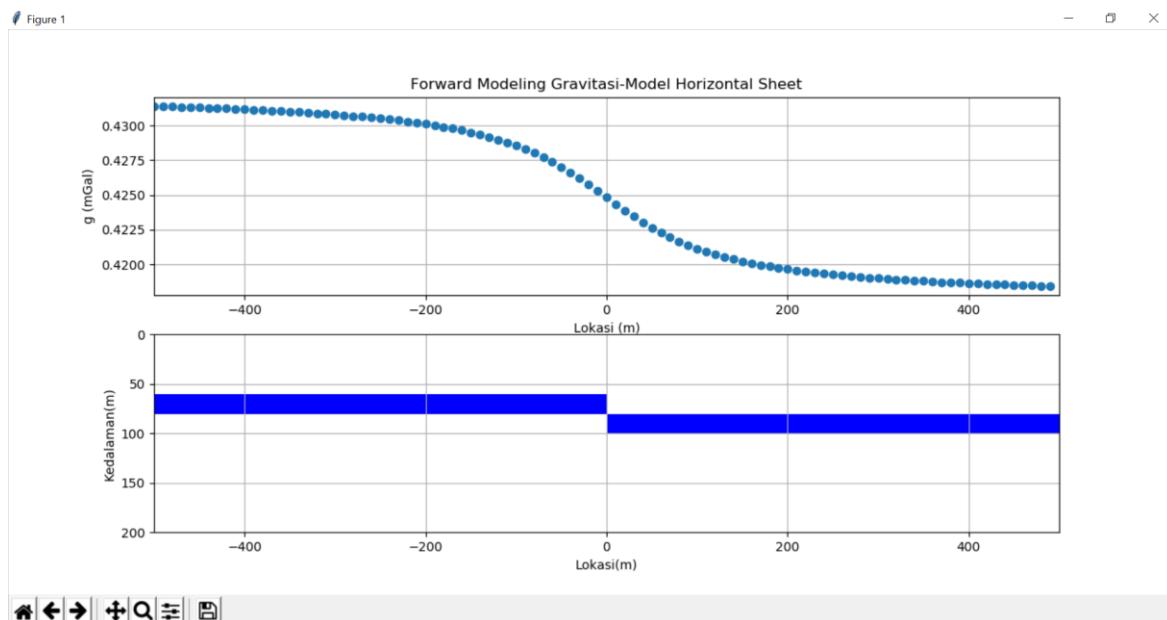
import numpy as np
import matplotlib.pyplot as plt

G = 6.673 * 10e-11
rho = float(input("Kontras densitas (kg/m^3) : "))
t = float(input("Ketebalan anomali (m) : "))
z = float(input("Kedalaman Titik Pusat Massa (m) : "))
x = int(input("Lokasi Maksimum di Permukaan (m) : "))
p = np.arange(0, 360, 1)
q = np.array(p * np.pi / 180)
i = np.arange(-x, x, 10)
m = np.arctan(-i / z) * np.pi / 180
n = np.array(m)
g = np.array(10e3 * 2 * G * rho * t / (np.pi * 0.5 - n))

plt.subplot(211)
plt.xlim(-x, x, 0.01)
plt.plot(i, g, "o")
plt.grid()
plt.xlabel("Lokasi (m)")
plt.ylabel("g (mGal)")
plt.title("Forward Modeling Gravitasasi-Model Horizontal Sheet")

plt.subplot(212)
k1 = [-x, -x, 0, 0]
l1 = [z-z*(1/5), z-z*(2/5), z-z*(2/5), z-z*(1/5)]
plt.fill(k1, l1, "-b")
k2 = [0, 0, x, x]
l2 = [z, z-z*(1/5), z-z*(1/5), z]
plt.fill(k2, l2, "-b")
plt.xlim(-x, x, 0.01)
plt.ylim(0, 2 * z, 0.01)
plt.grid()
plt.gca().invert_yaxis()
plt.xlabel("Lokasi(m)")
plt.ylabel("Kedalaman(m) ")
plt.show()

```



Gambar 66 - Forward modeling dengan geometri semi-infinite horizontal sheet

Forward Modeling untuk Metode Magnetik

Medan magnetik akan terjadi jika muatan listrik bergerak dari satu titik ke titik lainnya, atau merupakan perpindahan muatan listrik yang menimbulkan gaya magnetik.

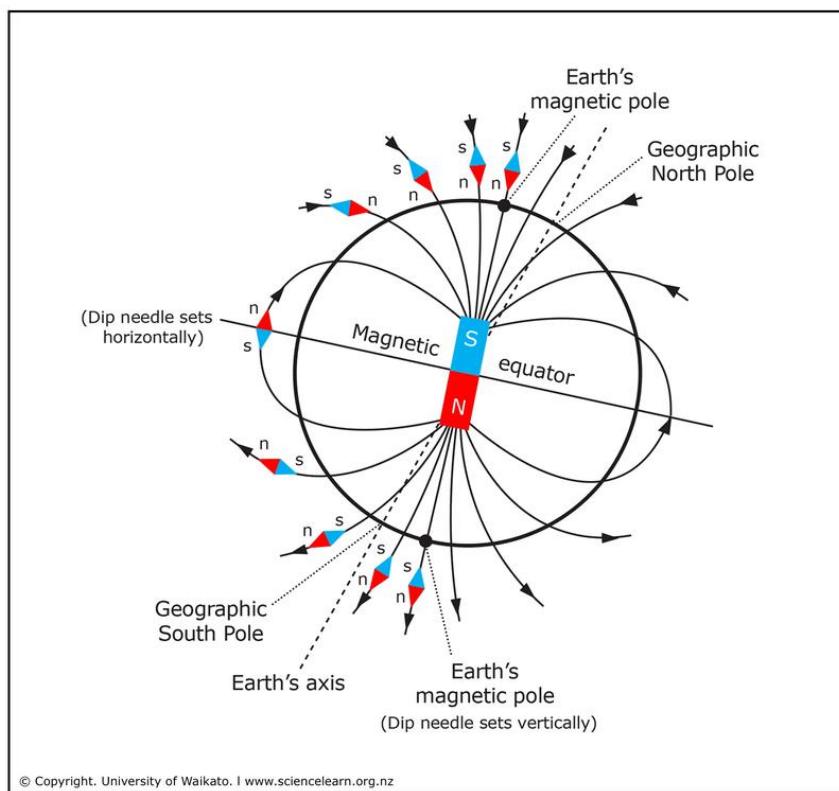
Suatu partikel yang mengandung muatan listrik bergerak melalui suatu medan magnet dan menimbulkan gaya F yang berbanding lurus terhadap muatan dan kecepatannya, ditunjukkan dalam persamaan:

$$F = qvH \sin \theta = q\bar{v} \times \bar{H}$$

Eq. 17

Dimana F adalah vektor gaya yang bekerja pada partikel, H adalah kuat medan magnetik (*magnetic field strength*), v adalah kecepatan partikel bergerak, q adalah muatan listrik dan θ adalah sudut antara medan magnetik dan arah gerak partikel.

Sedangkan bumi memiliki medan magnetik sendiri yang diakibatkan oleh hasil induksi gerakan berputar (rotasi) inti bumi yang berbahan ferromagnetik. Di permukaan bumi, jarum kompas bersifat magnetik akan bergerak bebas mengikuti arah utara (yang disebut kutub utara N) dan selatan (yang disebut kutub selatan S). Berdasarkan hasil observasi, bumi dikelilingi oleh medan magnetik yang bervariasi terhadap waktu dan lokasi.



Gambar 67 - Medan magnetik bumi

Metode magnetik digunakan untuk mengamati medan magnetik bumi pada suatu titik di permukaan bumi. Dalam Geofisika Eksplorasi, metode ini digunakan untuk mengamati anomali medan magnetik yang disebabkan oleh adanya perbedaan kemagnetan batuan dari suatu titik ke titik lainnya.

Mirip dengan gayaberat, gaya magnet memiliki persamaan:

$$\vec{F} = \frac{\mu_0}{4\pi} \frac{q_1 \cdot q_2}{r^2} \hat{r}$$

Eq. 18

Dimana F merupakan gaya magnet, q_1 dan q_2 sebagai muatan kutub magnet (*magnetic pole strength*), \hat{r} adalah vektor satuan dari r (jarak antara q_1 dan q_2), dan μ_0 adalah permeabilitas magnetik di ruang hampa ($4\pi \times 10^{-7}$ Henry/m atau 1.256×10^{-6} T.m/A). Sedangkan kuat medan magnet ditunjukkan dalam persamaan berikut:

$$\vec{B} = \frac{\vec{F}}{q_2} = \frac{\mu_0}{4\pi} \frac{q_1}{r^2} \hat{r}$$

Eq. 19

\vec{B} (kuat medan magnet) memiliki satuan Tesla (T), sedangkan dalam dunia eksplorasi, kuat medan magnet biasa berada pada ukuran nT (nano Tesla).

William Lowrie dalam bukunya yang berjudul "*Fundamentals of Geophysics*" (2007) menyebutkan anomali magnetik dilustrasikan dalam geometri sederhana secara vertikal, salah satunya adalah geometri *crustal block*. Adapun perhitungannya antara lain sebagai berikut:

$$\Delta B_z = \frac{\mu_0 \Delta M_z}{2\pi} [(\alpha_1 - \alpha_2) - (\alpha_3 - \alpha_4)]$$

Eq. 20

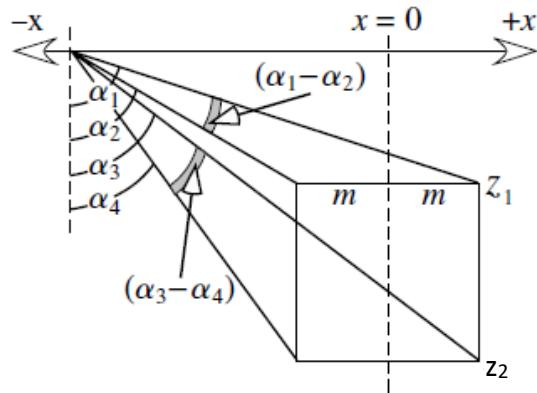
Persamaan diatas memiliki keterangan sebagai berikut:

ΔB_z = anomali medan magnetik (komponen vertikal) (nT)

μ_0 = permeabilitas magnetik di ruang hampa (1.256×10^3 nT.m/A)

ΔM_z = kontras magnetisasi antara batuan dengan medan magnet lainnya (A/m)

$\alpha_1, \alpha_2, \alpha_3, \alpha_4$ merupakan sudut pada setiap tepi *crustal block* seperti tergambar di bawah ini ($^{\circ}$)



Gambar 68 - Model anomali magnetik crustal block (Lowrie, 2007)

$$\alpha_1 = \tan^{-1} \left(\frac{(x-x_0)+m}{z_1} \right); \alpha_2 = \tan^{-1} \left(\frac{(x-x_0)-m}{z_1} \right); \alpha_3 = \tan^{-1} \left(\frac{(x-x_0)+m}{z_2} \right); \alpha_4 = \tan^{-1} \left(\frac{(x-x_0)-m}{z_2} \right)$$

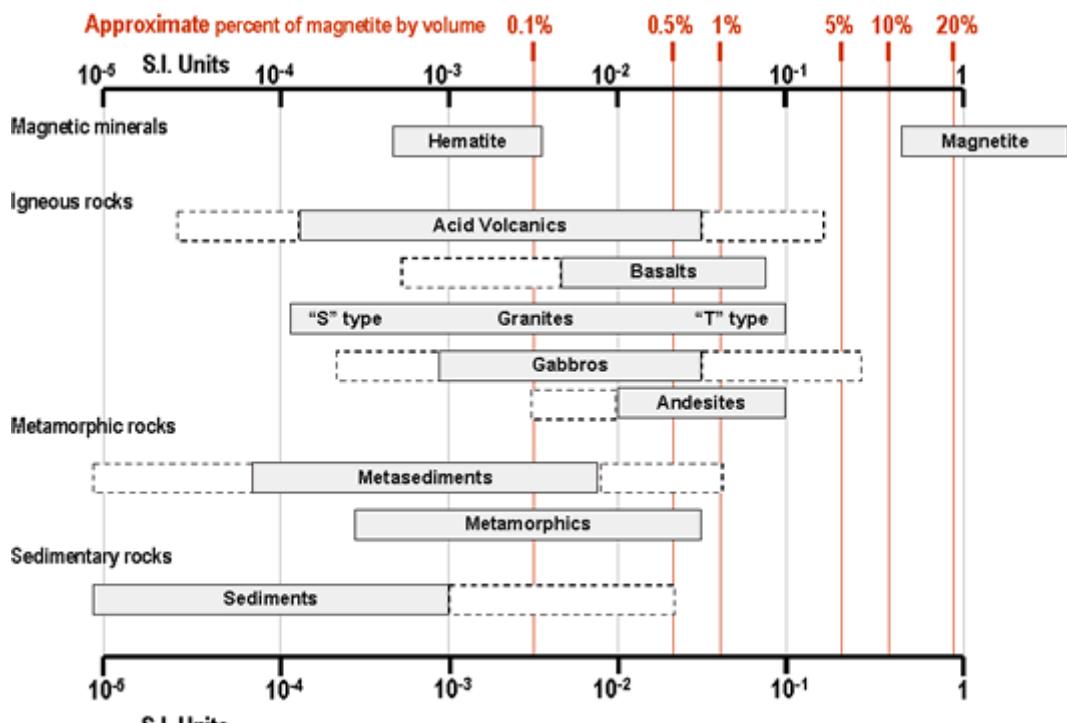
$2m$ merupakan lebar lapisan pada *crustal block*, z_1 adalah kedalaman sisi atas objek dan z_2 adalah kedalaman sisi bawah objek, x sebagai lokasi titik perhitungan (m) dan x_o merupakan lokasi objek.

Kontras magnetisasi memiliki persamaan:

$$\Delta M = (k - k_o)F$$

Eq. 21

Dengan ΔM sebagai kontras magnetisasi, k adalah suseptibilitas (kemampuan batuan termagnetisasi jika dikenai medan magnet) pada anomali dan k_o suseptibilitas batuan di sekitarnya. Sedangkan F menunjukkan kekuatan medan magnet yang diinduksikan. Oleh karena itu, nilai kontras magnetisasi bergantung pada nilai suseptibilitas, sebagai contoh:



Adapted from Clark and Emerson, Exploration Geophysics, 1991.

Gambar 69 - Distribusi Suseptibilitas Batuan (Sumber:

https://apg.geosci.xyz/content/physical_properties/magnetics_susceptibility_duplicate.html

Semakin tinggi suseptibilitas suatu anomali yang dapat ditimbulkan oleh mineral tertentu dibandingkan batuan sekitarnya, maka semakin tinggi nilai kontras magnetisasi yang menyebabkan semakin tinggi anomali medan magnet di lokasi tersebut.

Misalkan di bawah adalah pemodelan dengan menggunakan kontras magnetisasi 1 nT, objek berada pada kedalaman 10 m (batas atas) dan 50 m (batas bawah), dengan lebar lapisan 30 m berlokasi pada $x = 0$ m, dan lokasi pengukuran dari -100 m hingga 100 m.

```

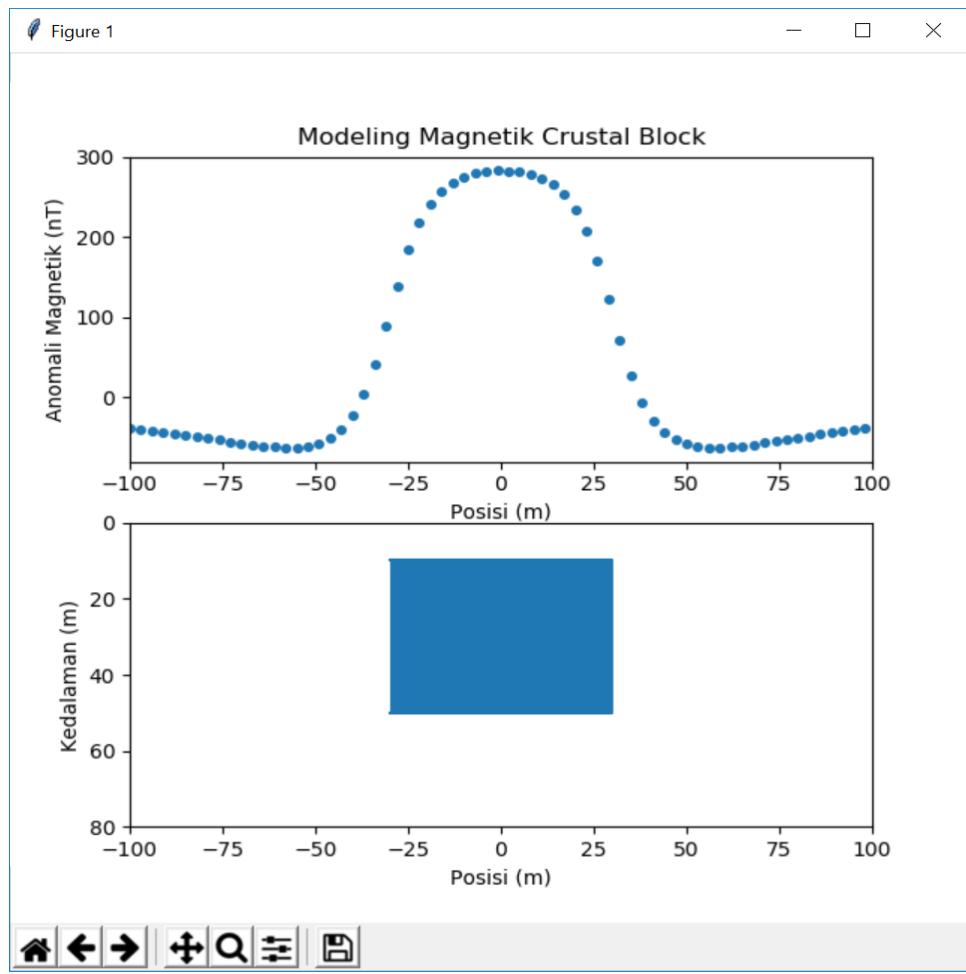
import numpy as np
import matplotlib.pyplot as plt

Uo = 1.256 * (10 ** 3)
Mz = 1
z1 = int(input('Kedalaman batas atas objek(m):'))
z2 = int(input('Kedalaman batas bawah objek(m):'))
m = int(input('Lebar lapisan(m):'))
xo = int(input('Lokasi objek(m):'))
w = int(input('Lokasi pengukuran maksimum(m):'))
x = np.arange(-w, w, 3)
f = []
for i in range(len(x)):
    a1 = np.arctan((x[i] - xo + m) / z1)
    a2 = np.arctan((x[i] - xo - m) / z1)
    a3 = np.arctan((x[i] - xo + m) / z2)
    a4 = np.arctan((x[i] - xo - m) / z2)
    Bz=(Uo * Mz *((a1 - a2) - (a3 - a4)) / (2 * np.pi))
    f.append(Bz)

plt.subplot(211)
plt.plot(x, f, 'o', markersize=4)
plt.xlim(-w, w)
plt.xlabel('Posisi (m)')
plt.ylabel('Anomali Magnetik (nT)')
plt.title('Modeling Magnetik Crustal Block')

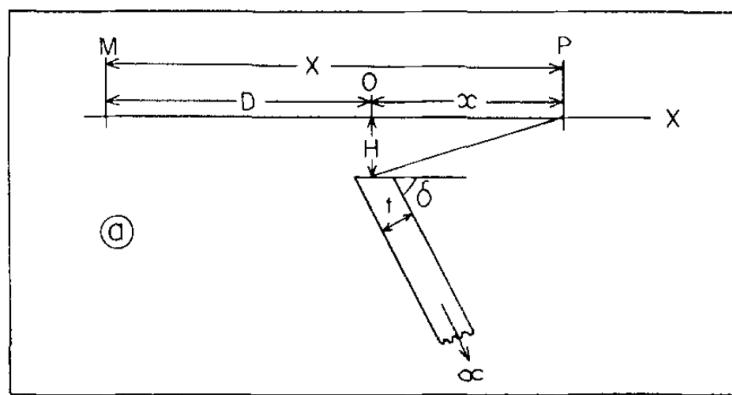
plt.subplot(212)
X = [xo - m, xo + m, xo + m, xo - m]
Y = [z1, z1, z2, z2]
plt.plot(X, Y)
plt.fill(X, Y)
plt.xlim(-w, w)
plt.ylim(0, z2 + 30)
plt.gca().invert_yaxis()
plt.xlabel('Posisi (m)')
plt.ylabel('Kedalaman (m)')
plt.show()

```



Gambar 70 - Forward modeling magnetik geometri crustal block

Selain geometri crustal block, model anomali magnetik lainnya sebagaimana yang disebutkan dalam “*Inversion of gravity and magnetic anomalies over some bodies of simple geometric shape*” (Rao et al, 1985), yaitu model *thin sheet*.



Gambar 71 - Model anomali magnetik thin sheet (Rao et al, 1985)

Respon yang ditampilkan adalah anomali magnetik (kontras magnetisasi) ΔM yang ditunjukkan dengan persamaan:

$$\Delta M(x) = P \frac{x \sin Q + H \cos Q}{x^2 + H^2}$$

Eq. 22

Persamaan diatas memiliki keterangan sebagai berikut:

- ΔM = anomali magnetik (kontras magnetisasi) (A/m)
- x = jarak titik pengukuran ke titik objek (m)
- H = kedalaman batas atas *thin sheet* (m)
- P = koefisien amplitudo
- Q = parameter index ($^{\circ}$)
- t = ketebalan lapisan *thin sheet* (m)
- δ = *dip* atau sudut kemiringan *thin sheet* dan horizontal ($^{\circ}$)

Adapun persamaan untuk koefisien amplitudo dan parameter index adalah:

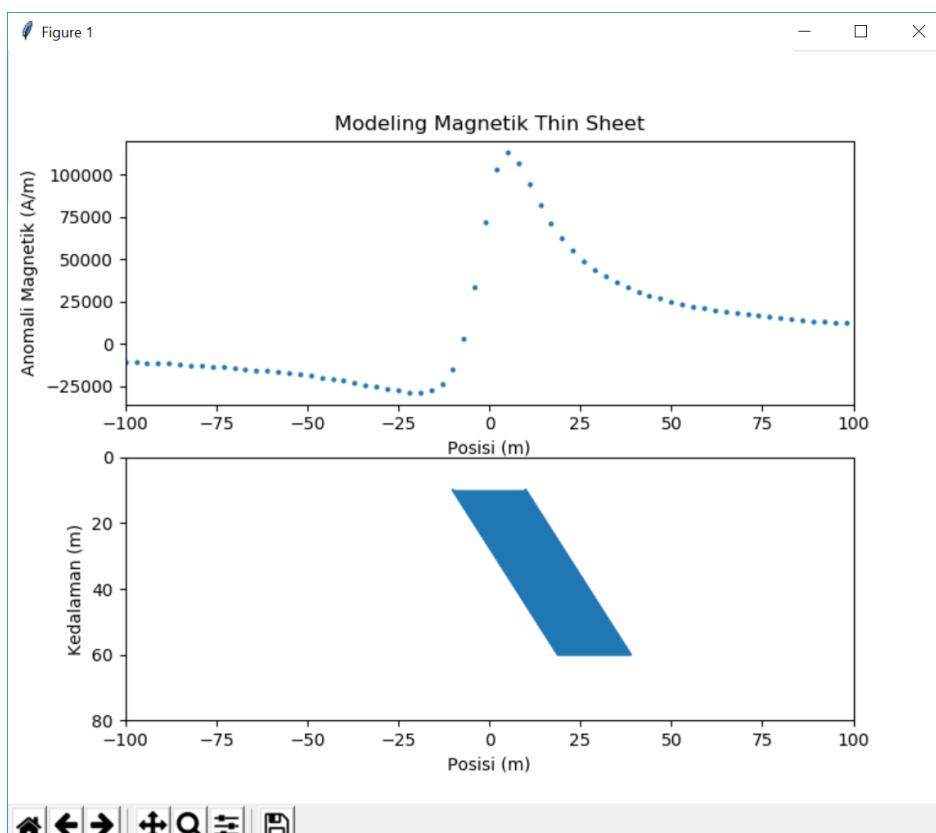
$P = 2kT_o t (1 - \cos^2 I_o - \cos^2 a)^{\frac{1}{2}}$	$Q = I'_o - \delta$ $I'_o = \tan^{-1} \frac{\tan I_o}{\sin a}$
---------------------------------------------------------	----------------------------------------------------------------

Eq. 23

Keterangan:

- k = kontras suseptibilitas
- T_o = intensitas medan magnetik bumi (nT)
- I_o = sudut inklinasi medan magnet bumi ($^{\circ}$)
- a = *strike* atau azimuth sudut objek dari utara ($^{\circ}$)

Misalkan pada pemodelan berikut, diberikan informasi bahwa $k = 1$, $T_o = 50000$ nT, $I_o = 60^{\circ}$, $a = 120^{\circ}$, $t = 20$ m, $\delta = 10^{\circ}$, $H = 10$ m dan pengukuran dilakukan pada -100 m hingga 100 m.



Gambar 72 - Forward modeling magnetik geometri *thin sheet*

```

import numpy as np
import matplotlib.pyplot as plt

#Parameter
k = 1          #kontras suseptibilitas
To = 50000    #intensitas medan magnet bumi
Io = 60        #sudut inklinasi medan magnet bumi
a = 120        #strike
d = 10         #dip
t = 20         #ketebalan thin sheet
H = 10          #kedalaman (batas atas) obyek
w = 100         #maksimum lokasi pengukuran
x = np.arange(-w, w, 3)

s = np.cos(a * np.pi / 180) / H
Iio =
np.degrees(np.arctan(np.tan(np.radians(Io))/np.sin(np.radians(a))))
)
Q = np.radians(Iio - d)
P = 2*k*To*t*((1-(np.cos(np.radians(Io)))**2)-
(np.cos(np.radians(a)))**2)**0.5

M = []
for i in range(len(x)):
    M.append(
        P * ((x[i] * np.sin(Q)) + (H * np.cos(Q))) / ((x[i]**2) + (H**2)))

plt.subplot(211)
plt.plot(x, M, 'o', markersize=2)
plt.xlim(-w, w)
plt.xlabel('Posisi (m)')
plt.ylabel('Anomali Magnetik (A/m)')
plt.title('Modeling Magnetik Thin Sheet')

plt.subplot(212)
X = (t / 2, (50 / -np.tan(a * np.pi / 180)) + t / 2, (50 / -
np.tan(a * np.pi / 180)) - t / 2, -t / 2)
Y = (H, H + 50, H + 50, H)
plt.plot(X, Y)
plt.fill(X, Y)
plt.xlim(-w, w)
plt.ylim(0, H + 70)
plt.gca().invert_yaxis()
plt.xlabel('Posisi (m)')
plt.ylabel('Kedalaman (m)')
plt.show()

```

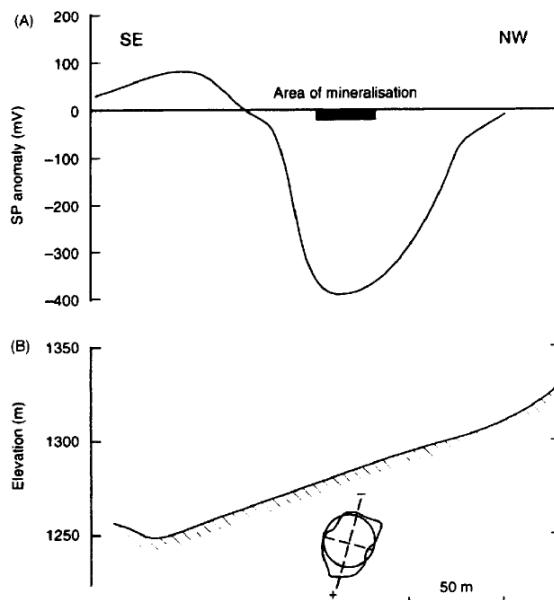
Forward Modeling untuk Metode Self-Potential

Menurut Reynolds (1997) dalam bukunya “*An Introduction to Applied and Environmental Geophysics*”, metode *Self-Potential* atau *Spontaneous Polarisation (SP)* dirancang oleh Robert Fox pada tahun 1830, menggunakan elektroda plat tembaga yang terhubung dengan galvanometer untuk mendeteksi deposit sulfida tembaga di Cornwall, Inggris. Metode ini digunakan sejak 1920 sebagai alat bantu dalam eksplorasi logam dasar. Studi terbaru metode SP adalah untuk menginvestigasi air tanah dan panasbumi, yang juga digunakan untuk membantu pemetaan geologi, misalnya menggambarkan zona geser dan sesar dekat permukaan.

Metode SP sendiri merupakan metode pasif, untuk mengukur perbedaan potensial atau tegangan antara dua titik di permukaan tanah, dalam skala miliVolt hingga 1 Volt. Secara sederhana, teknik pengukuran metode SP adalah dengan menggunakan dua elektroda yang dibuat dari plat tembaga yang berada dalam larutan jenuh sulfat yang dapat berhubungan dengan tanah dan menghasilkan listrik.

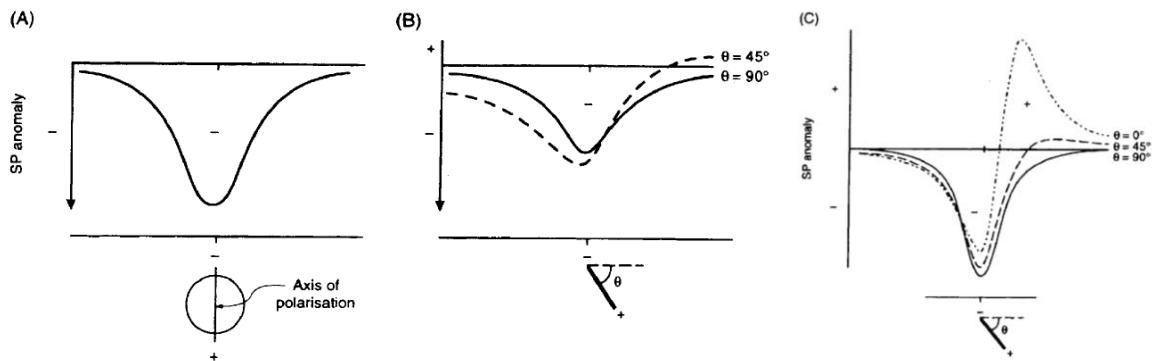
Sedangkan hasil dari survei potensial adalah peta isopotensial, dan interpretasi terhadap daerah anomali menggunakan penampang melintang yang memotong daerah anomali. Anomali SP diinterpretasikan secara kualitatif dari bentuk profil. Bagian atas dari bijih diasumsikan berada persis dibawah potensial minimum.

Sebagai contoh gambar di bawah ini, anomali SP Weiss (A) di Ergani, Turki, dengan bijih kausatif terlihat secara sistematis pada gambar (B). Note: sumbu polarisasi pada inklinasi tanjakan. Sumber Gambar dari Reynolds (1997) yang mengambil dari Yungul (1950).



Gambar 73 - Interpretasi Anomali SP (Reynolds, 1997)

Pendekatan inversi dilakukan untuk memanipulasi anomali yang teramati untuk menghasilkan suatu model. Metode ini digunakan untuk mengestimasi ukuran suatu fitur geologi. Berikut adalah anomali SP yang berkaitan dengan model (asumsi) suatu geometri tertentu.



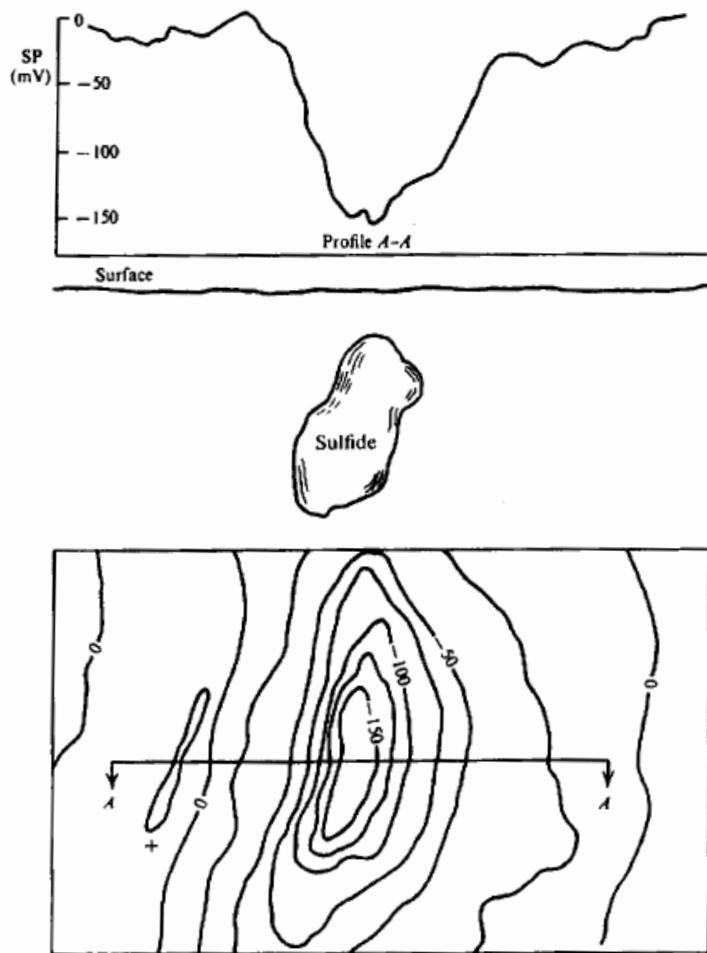
Gambar 74 - Anomali SP dengan geometri sphere (A), dipping plate (B) dan dipping rod (C) (Reynolds, 1997)

Tipe anomali SP dan sumber geologinya:

Sumber	Tipe Anomali
Potensial Mineral <ul style="list-style-type: none"> • Tubuh bijih sulfida (pirit, kalkopirit, spalerit, galena) • Tubuh bijih grafit • Magnetit + mineral penghantar listrik (konduktif) • Batubara • Mangan • Urat kuarsa (<i>quartz vein</i>) • Pegmatit 	<div style="display: flex; align-items: center;"> } Hingga (negatif) ratusan mV </div> <div style="display: flex; align-items: center;"> } Hingga (positif) puluhan mV </div>
Potensial Alami <ul style="list-style-type: none"> • Aliran fluida, reaksi geokimia • Biolistik (pepothonan) • Aliran air tanah • Topografi 	<ul style="list-style-type: none"> • $V < \pm 100$ mV • $V \leq -300$ mV atau lebih • Hingga (positif/ negatif) ratusan mV • Hingga -2V

Forward modeling sendiri merupakan penggunaan suatu model untuk mensimulasikan suatu keluaran (*outcome*). Oleh karena itu, perlu dikenali jenis (bentuk) target anomali yang akan diukur menggunakan metode SP tersebut, yang dijadikan sebagai model awal. Misalnya target (mineral) dimodelkan dalam model bola, silinder, dan lempeng.

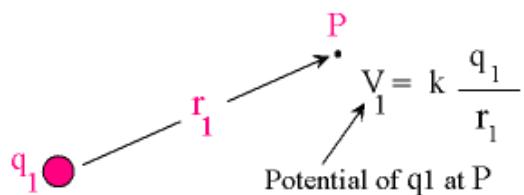
Menurut Telford et al (1990) dalam buku “*Applied Geophysics*”, hasil akhir dari survei SP adalah suatu profil dan peta kontur isopotensial. Nilai SP paling minimum menunjukkan adanya anomali potensial.



Gambar 75 - Contoh penampang dan peta kontur hasil survei SP pada tubuh sulfida (Telford et al, 1990)

Nilai potensial diperoleh jika diketahui muatan listrik, dan jaraknya terhadap suatu titik, seperti ditunjukkan pada gambar berikut:

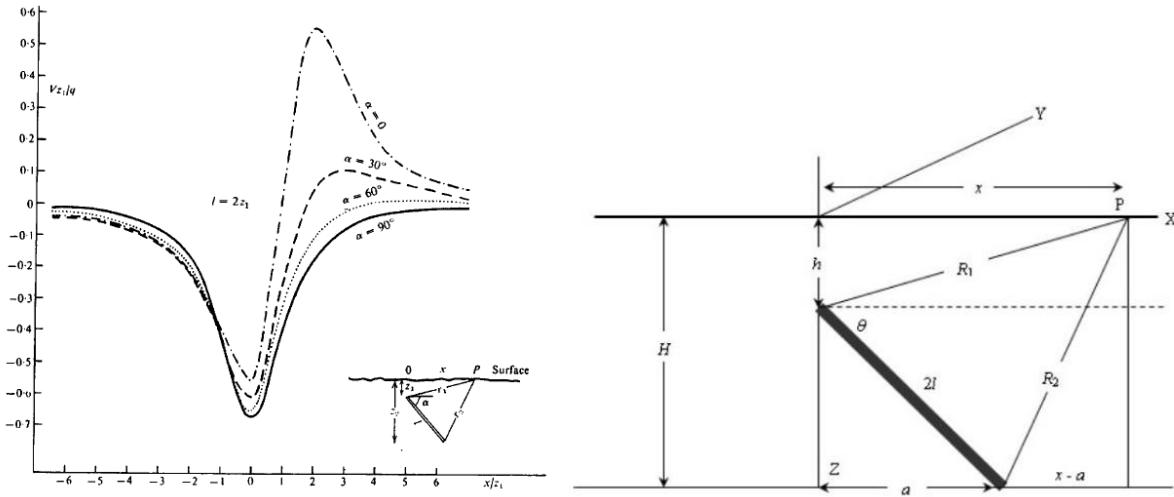
$$V_1 = k \frac{q_1}{r_1} \quad P \text{ is just a point in space.}$$



Gambar 76 - Perhitungan Nilai Potensial di Suatu Titik (Sumber: http://www.pstcc.edu/departments/natural_behavioral_sciences/Web%20Physics/Chapters%2025%20and%2026.htm)

Pada gambar diatas, nilai potensial di titik P ditentukan dengan mengalikan \mathbf{k} (konstanta Coulomb yang setara dengan $9 \times 10^9 \text{ Nm}^2/\text{C}^2$ dengan nilai muatan listrik (q) yang diketahui dalam satuan C (Coulomb), dibagi dengan jarak antara muatan listrik dengan titik P tersebut (r).

Misalkan model mineral adalah seperti lempeng/ tongkat (*plate/ rod*) seperti yang ditunjukkan pada gambar di bawah ini. Perhitungan beda potensial pada titik tersebut akan menjadi:



Gambar 77 - Kiri: Hasil survei SP (Telford et al, 1990), Kanan: Geometri lempeng (Yuliananto & Setyawan, 2015)

Mineral dianggap merupakan sumber anomali dengan kedalaman batas atas h , kedalaman batas bawah H , panjang $2l$, dan sudut inklinasi θ . R_1 dan R_2 merupakan jarak dari batas atas dan batas bawah ke titik P. Misalkan x adalah titik sembarang antara titik awal (x_o) dan titik P, maka beda potensial $V(x)$ dapat dihitung dengan persamaan:

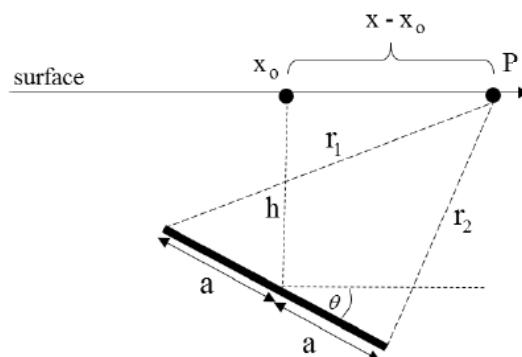
$$V(x) = K \log \frac{R_1^2}{R_2^2} = K \ln \left[\frac{x^2 + h^2}{(x - a)^2 + H^2} \right]$$

Eq. 24

Atau dapat juga diekspresikan dalam bentuk persamaan berikut, dimana panjang lempeng adalah $2a$ dengan h merupakan kedalaman titik tengah lempeng tersebut (Gambar 35).

$$V(x) = K \ln \left[\frac{\{(x - x_o) + a \cos \theta\}^2 + (h - a \sin \theta)^2}{\{(x - x_o) - a \cos \theta\}^2 + (h + a \sin \theta)^2} \right]$$

Eq. 25



Gambar 78 – Anomali SP dengan Geometri Lempeng (Fajriani et al, 2004) dengan inklinasi θ searah jarum jam.

K merupakan suatu konstanta dimana nilainya adalah $K = \frac{I\rho}{2\pi}$ dimana I adalah rapat arus per satuan panjang dan ρ adalah resistivitas medium batuan. Nilai K biasanya berkisar antara 0 – 2000 mV.

Contoh soal:

Pada *forward modeling* metode *Self-Potential*, digunakan model lempeng dengan sudut inklinasi 30° , panjang lempeng adalah 20 m dan kedalaman titik tengah lempeng adalah 40 m. Konstanta K diketahui memiliki nilai 500 mV, titik x_o berada pada koordinat $x = 100$ m. Misalkan suatu titik x berada pada koordinat $x = 200$ m, hitunglah potensial pada titik x tersebut.

Jawaban:

$$\theta = 30^\circ$$

$$2a = 20m$$

$$a = 10m$$

$$h = 20m$$

$$K = 500mV$$

$$x_o = 100m$$

$$x = 200m$$

Maka potensial pada titik x adalah:

$$V(x) = K \ln \left[\frac{\{(x - x_o) + a \cos \theta\}^2 + (h - a \sin \theta)^2}{\{(x - x_o) - a \cos \theta\}^2 + (h + a \sin \theta)^2} \right]$$

$$V(x) = 500 \ln \left[\frac{\{(200 - 100) + 10 \cos 30^\circ\}^2 + (20 - 10 \sin 30^\circ)^2}{\{(200 - 100) - 10 \cos 30^\circ\}^2 + (20 + 10 \sin 30^\circ)^2} \right]$$

$$V(x) = 62.3 mV$$

Berikut adalah *script* pemrograman Python untuk pemodelan (*forward modeling*) metode *Self-Potential* (SP) dengan geometri lempeng seperti yang dijelaskan diatas:

```

import numpy as np
import matplotlib.pyplot as plt

k = int(input('Nilai K = '))
h = int(input('Kedalaman Titik Tengah Anomali (h) = '))
x0 = int(input('Titik awal pengukuran (x0) = '))
i = int(input('Jarak pengukuran (x) = '))
x = np.arange(0, i, 5)
a = int(input('Setengah panjang lempeng (a) = '))
theta = int(input('Sudut inklinasi (theta) = '))
t = theta * (np.pi / 180)
s = []
for i in range(0, len(x)):
    b = (((x[i] - x0) + a * np.cos(t)) ** 2 + (h - a * np.sin(t))
    ** 2)
    c = (((x[i] - x0) - a * np.cos(t)) ** 2 + (h + a * np.sin(t))
    ** 2)
    s.append(np.log(b / c))
v = k * np.array(s)

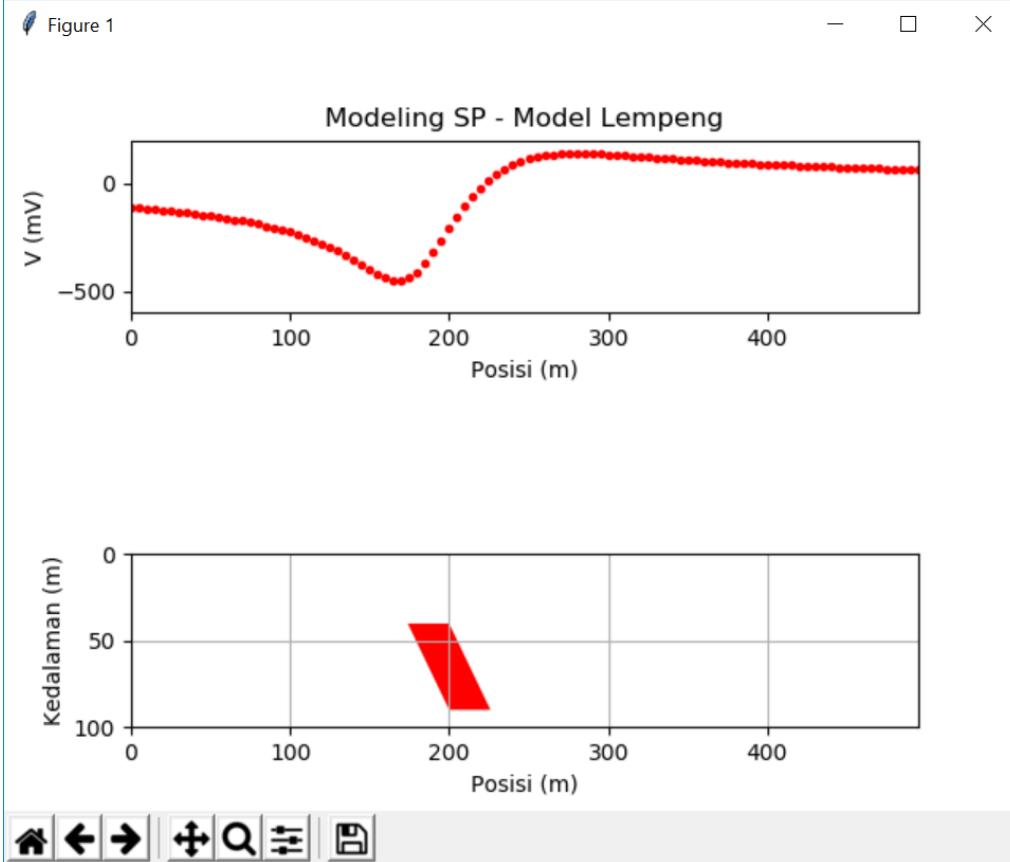
plt.subplot(311)
plt.plot(x, v, '.r')
plt.xlim(min(x), max(x))
plt.ylim(k, -k*3)
plt.gca().invert_yaxis()
plt.xlabel('Posisi (m)')
plt.ylabel('V (mV)')
plt.title('Modeling SP - Model Lempeng')

plt.subplot(313)
k1 = [x0 - a * np.cos(t), x0, x0 + a * np.cos(t), x0]
l1 = [h, h + 50, h + 50, h]
plt.fill(k1, l1, "r")
plt.xlim(min(x), max(x), 0.01)
plt.ylim(0, 2.5 * h, 0.01)
plt.grid()
plt.gca().invert_yaxis()
plt.xlabel("Posisi (m)")
plt.ylabel("Kedalaman (m)")
plt.show()

```

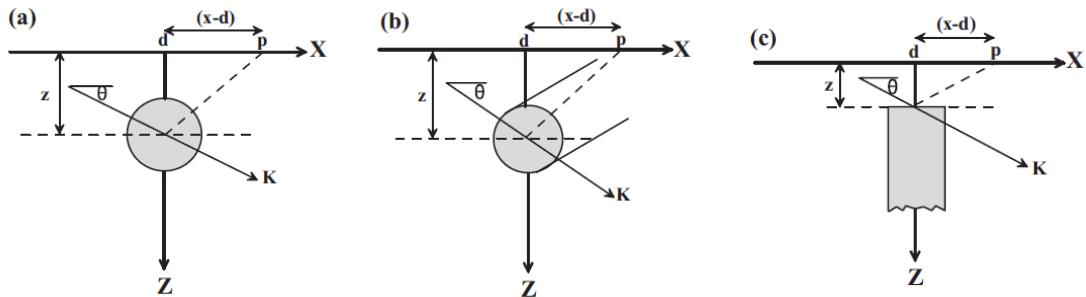
Nilai K = 200
Kedalaman Titik Tengah Anomali (h) = 40
Titik awal pengukuran (x0) = 200
Jarak pengukuran (x) = 500
Setengah panjang lempeng (a) = 30
Sudut inklinasi (theta) = 30

Berikut adalah hasil *forward modeling Self-Potential* dengan anomali berbentuk geometri lempeng (*plate/rod*) menggunakan Eq. 2 dengan parameter diatas:



Gambar 79 - Forward Modeling SP dengan Geometri Lempeng Menggunakan Python

Model lain yang digunakan untuk pemodelan SP adalah dengan geometri bola (*sphere*) dan silinder (*cylinder*), dengan ilustrasi perhitungan sebagai berikut:



Gambar 80 – Anomali SP dengan Geometri Bola (A), Silinder horizontal (B) dan Silinder vertikal (C) (Essa, 2019)

Persamaan umum yang digunakan untuk ketiga geometri adalah:

$$V(x) = K \frac{(x-d) \cos \theta - z \sin \theta}{[(x-d)^2 + z^2]^q}$$

Eq. 26

K disini adalah faktor amplitudo dengan satuan $\text{mV} \cdot \text{m}^{2q-1}$, z adalah kedalaman struktur, d adalah lokasi sumber, dan θ adalah sudut polarisasi. Sementara q adalah parameter bentuk (non-dimensi) dengan keterangan:

Geometry	<i>q</i>	<i>K</i> unit	Equation
<i>Sphere</i>	1.5	mV.m^2	$V(x) = K \frac{(x - d) \cos \theta - z \sin \theta}{[(x - d)^2 + z^2]^{3/2}}$
<i>Horizontal cylinder</i>	1.0	mV.m	$V(x) = K \frac{(x - d) \cos \theta - z \sin \theta}{(x - d)^2 + z^2}$
<i>Vertical cylinder</i>	0.5	mV	$V(x) = K \frac{(x - d) \cos \theta - z \sin \theta}{\sqrt{(x - d)^2 + z^2}}$

Eq. 27

Pada pemrograman Python, dapat dibuat dengan *script* berikut:

```

import numpy as np
import matplotlib.pyplot as plt

k = int(input('Nilai K = '))
z = int(input('Kedalaman Anomali (z) = '))
d = int(input('Titik awal pengukuran (d) = '))
i = int(input('Jarak pengukuran (x) = '))
x = np.arange(-i, i, 5)
theta = int(input('Sudut Polarisasi (theta) = '))
t = theta * (np.pi / 180)

def fungsi_model_bola():
    r = int(input('Jari-jari Model Bola = '))
    th = np.arange(0, 360)
    s = []
    for i in range(0, len(x)):
        s.append(((x[i]) - d) * np.cos(t)) +
               (z * np.sin(t))) / (((x[i] - d) ** 2) + (z **
2)) ** 1.5)
    v = k * np.array(s)
    xunit = r * np.cos(th) + d
    yunit = r * np.sin(th) + z
    plt.subplot(211)
    plt.plot(x, v, '.r')
    plt.xlim(min(x), max(x))
    plt.xlabel('Posisi (m)')
    plt.ylabel('Delta g (mGal)')
    plt.title('Modeling SP - Model Bola')
    plt.subplot(212)
    plt.fill(xunit, yunit, 'r')
    plt.xlim(min(x), max(x))
    plt.gca().set_aspect('equal', adjustable='box')
    plt.gca().invert_yaxis()
    plt.xlabel('Posisi (m)')
    plt.ylabel('Kedalaman (m)')
    plt.show()
    return ()

```

To Be Continued ──

```

def fungsi_model_silinder():
    r = int(input('Jari-jari Model Silinder = '))
    th = np.arange(0, 360)
    s = []
    for i in range(0, len(x)):
        a1 = (((x[i] - d) * np.cos(t)) + (z * np.sin(t)))
        b1 = (((x[i] - d) ** 2) + (z ** 2))
        s.append(a1 / b1)
    v = k * np.array(s)
    xunit = (r * np.cos(th) + d)
    yunit = (r * np.sin(th) + z)
    plt.subplot(211)
    plt.plot(x, v, '.r')
    plt.xlim(min(x), max(x))
    plt.xlabel('Posisi (m)')
    plt.ylabel('Delta g (mGal)')
    plt.title('Modeling SP - Model Silinder')
    plt.subplot(212)
    plt.fill(xunit, yunit, 'r')
    plt.xlim(min(x), max(x))
    plt.gca().set_aspect('equal', adjustable='box')
    plt.gca().invert_yaxis()
    plt.xlabel('Posisi (m)')
    plt.ylabel('Kedalaman (m)')
    plt.show()
    return ()

i1 = 0
while i1 <= 1000000:
    print("Masukkan angka 1 untuk Model Bola dan angka 2 untuk
Model Silinder")
    Variabel = int(input("Masukkan Model Yang Ingin Dituju( 1/2 ):"))
    if Variabel == 1:
        print(fungsi_model_bola())
    elif Variabel == 2:
        print(fungsi_model_silinder())
    else:
        print("Kode Yang Anda Input Error")
    Lanjut = str(input("Perhitungan yang Lain untuk parameter yang
    sama? ( y/n ): "))
    if Lanjut == "y":
        i1 = i1 + 1
    else:
        print("Terima Kasih")
        break

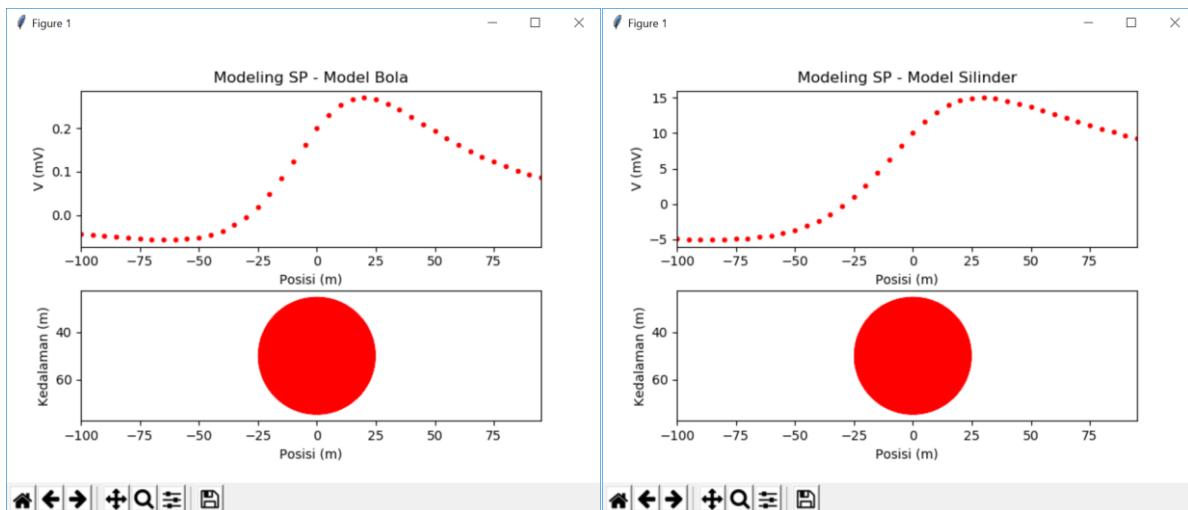
```

Contoh parameter yang di-input:

```
Nilai K = 1000
Kedalaman Anomali (z) = 50
Titik awal pengukuran (d) = 0
Jarak pengukuran (x) = 100
Sudut Polarisasi (theta) = 30
Masukkan angka 1 untuk Model Bola dan angka 2 untuk Model Silinder
Masukkan Model Yang Ingin Dituju( 1/2 ): 1
Jari-jari Model Bola = 25
```

Dengan parameter yang sama, namun model yang berbeda:

```
Perhitungan yang Lain untuk parameter yang sama? ( y/n ): y
Masukkan angka 1 untuk Model Bola dan angka 2 untuk Model Silinder
Masukkan Model Yang Ingin Dituju( 1/2 ): 2
Jari-jari Model Silinder = 25
```



Gambar 81 - Forward Modeling SP dengan Geometri Bola (kiri) dan Silinder (kanan) Menggunakan Python

Karena adanya perbedaan persamaan antara model bola dan silinder, nilai potensial dan model yang dihasilkan pun berbeda.

Model Pembajian Sintetik (*Synthetic Wedge Model*)

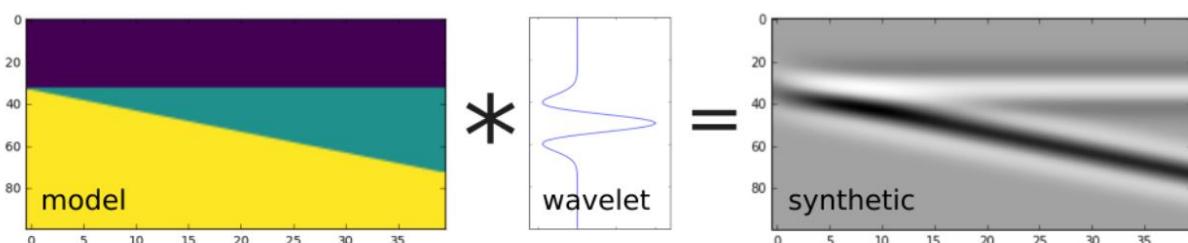
Model pembajian menunjukkan respon seismik terhadap ketebalan lapisan, yang menjelaskan tentang *tuning* atau fenomena lapisan tipis pada data seismik. *Tuning* berkaitan dengan amplitudo seismik yang semakin jelas ataupun tidak, yang disebabkan karena interferensi konstruktif dan destruktif dari reflektor seismik yang *overlapping*, dan refleksi dari beberapa bidang batas tipis yang berdekatan. Ketebalan dimana interferensi mulai terjadi disebut dengan *tuning thickness* dengan ketebalan $\lambda/4$.

Dalam membuat model pembajian sintetik, perlu diketahui bahwa seismik sintetik merupakan konvolusi antara koefisien refleksi dengan *seismic wavelet*. Sedangkan koefisien refleksi menunjukkan nilai batas antara dua lapisan yang diperoleh dari rasio impedansi akustik (AI), atau perkalian antara densitas dan kecepatan batuan.

$$RC = \frac{\rho_2 v_2 - \rho_1 v_1}{\rho_2 v_2 + \rho_1 v_1}$$

Eq. 28

Berikut adalah model pembajian dari menggunakan tiga jenis batuan, yang kemudian dikonvolusikan dengan *wavelet* untuk memperoleh seismik sintetik.



Gambar 82 - Model Pembajian dan Seismik Sintetik (Sumber: <https://agilescientific.com/blog/2016/9/15/x-lines-of-python-synthetic-wedge-model>)

Berikut adalah script *tuning_wedge.py* yang diperoleh dari https://wiki.seg.org/wiki/Thin_beds,_tuning,_and_AVO untuk model pembajian dan *tuning thickness*. Namun script dibagi dalam beberapa partisi untuk menyesuaikan dengan penjelasannya.

```
"""
Python script to generate a zero-offset synthetic from a 3-layer wedge
model.

Created by: Wes Hamlyn
Create Date: 19-Aug-2014
Last Mod: 1-Nov-2014
This script is provided without warranty of any kind.
"""
```

```

import numpy as np
import matplotlib.pyplot as plt

#      DEFINE MODELING PARAMETERS HERE
# 3-Layer Model Parameters [Layer1, Layer2, Layer 3]
vp_mod = [2500.0, 2600.0, 2550.0] # P-wave velocity (m/s)
vs_mod = [1200.0, 1300.0, 1200.0] # S-wave velocity (m/s)
rho_mod = [1.95, 2.0, 1.98] # Density (g/cc)

dz_min = 0.0 # Minimum thickness of Layer 2 (m)
dz_max = 60.0 # Maximum thickness of Layer 2 (m)
dz_step = 1.0 # Thickness step from trace-to-trace (normally 1.0 m)

```

Lapisan memiliki model pembajian dengan ketebalan 0 – 60 m, dengan keterangan berikut:

	Kecepatan gelombang P atau vp (m/s)	Kecepatan gelombang S atau vs (m/s)	Densitas (ρ) (gr/cm ³)
Lapisan 1	2500	1200	1.95
Lapisan 2	2600	1300	2.0
Lapisan 3	2550	1200	1.98

Sebelum membuat model pembajian, berikut adalah pemrograman apabila ketiga lapisan dibatasi pada *time* tertentu.

```

vp = [2500, 2600, 2550]
rho = [1950, 2000, 1980]

rc = []
for i in range(0,2):
    ref = ((rho[i+1]*vp[i+1])-
(rho[i]*vp[i]))/((rho[i+1]*vp[i+1])+(rho[i]*vp[i]))
    rc.append(ref)

import matplotlib.pyplot as plt
import numpy as np

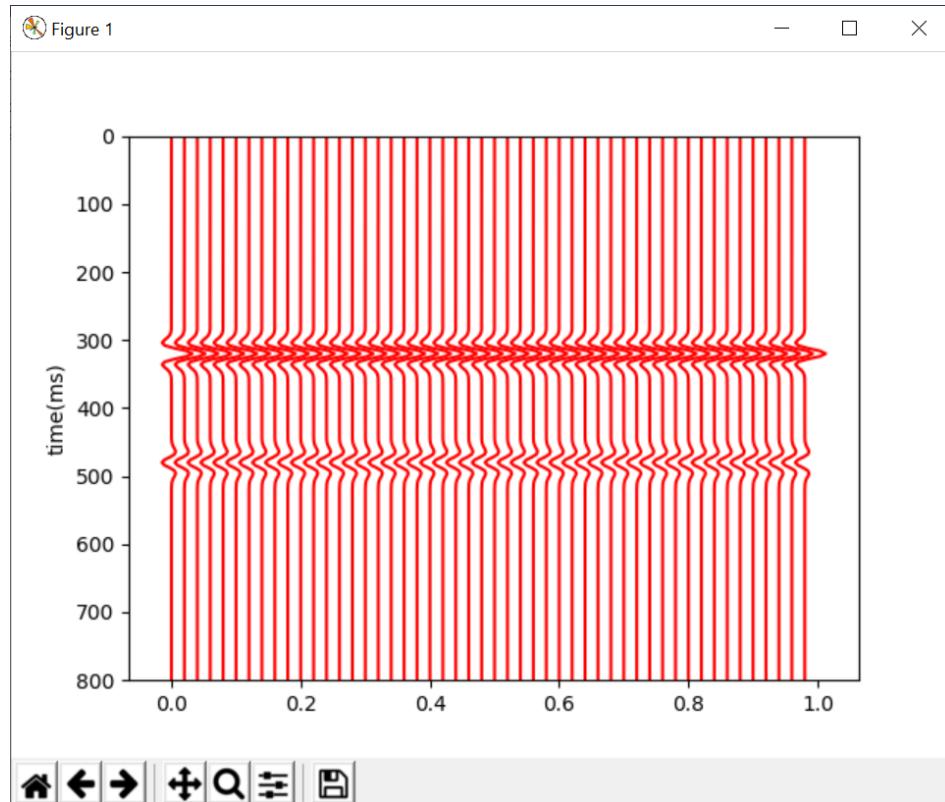
#ricker wavelet
L = 400 # wavelet length (ms)
s = 4 # sampling rate (ms)
f = 25 # frequency (Hz)
t = np.arange(-L/2000,L/2000+s/1000,s/1000) # must be in second
r = (1 - 2 * np.pi**2 * f**2 * t**2) *np.exp(-1*np.pi**2 *
f**2*t**2) #ricker wavelet

plt.plot(t,r)
plt.show()

tr = 0.0*np.arange(0,len(r),1)
tr[30] = rc[0]
tr[70] = rc[1]
time=s*np.arange(0,(len(r)+len(tr)-1),1)

```

```
for i in range(50):
    seismic = np.convolve(r,tr)
    plt.plot(seismic+i*0.02,time,'r-')
    plt.ylim(max(time),min(time))
    plt.ylabel('time(ms)')
plt.show()
```



Gambar 83 – Visualisasi koefiesien refleksi setelah dikonvolusikan dengan Ricker wavelet

```

import matplotlib.pyplot as plt
import numpy as np

# 3-Layer Model Parameters [Layer1, Layer2, Layer 3]
vp_mod = [2500.0, 2600.0, 2550.0] # P-wave velocity (m/s)

dz_min = 0.0 # Minimum thickness of Layer 2 (m)
dz_max = 60.0 # Maximum thickness of Layer 2 (m)
dz_step = 1.0 # Thickness step from trace-to-trace (normally 1.0 m)

# Plot Parameters
min_plot_time = 0.15
max_plot_time = 0.3

def calc_times(z_int, vp_mod):
    """
    t_int = calc_times(z_int, vp_mod)

    nlayers = len(vp_mod)
    nint = nlayers - 1

    t_int = []
    for i in range(0, nint):
        if i == 0:
            tbuf = z_int[i] / vp_mod[i]
            t_int.append(tbuf)
        else:
            zdiff = z_int[i] - z_int[i - 1]
            tbuf = 2 * zdiff / vp_mod[i] + t_int[i - 1]
            t_int.append(tbuf)

    return t_int

nmodel = int((dz_max - dz_min) / dz_step + 1)

lyr_times = []
for model in range(0, nmodel):
    # Calculate interface depths
    z_int = [500.0]
    z_int.append(z_int[0] + dz_min + dz_step * model)

    # Calculate interface times
    t_int = calc_times(z_int, vp_mod)
    lyr_times.append(t_int)

lyr_times = np.array(lyr_times)

plt.plot(lyr_times[:, 0], color='blue', lw=1.5)
plt.plot(lyr_times[:, 1], color='red', lw=1.5)
plt.ylim((min_plot_time, max_plot_time))
plt.gca().invert_yaxis()
plt.xlabel('Thickness (m)')
plt.ylabel('Time (s)')
plt.text(2,
         min_plot_time + (lyr_times[0, 0] - min_plot_time) / 2.,
         'Layer 1',
         fontsize=16)

```

```

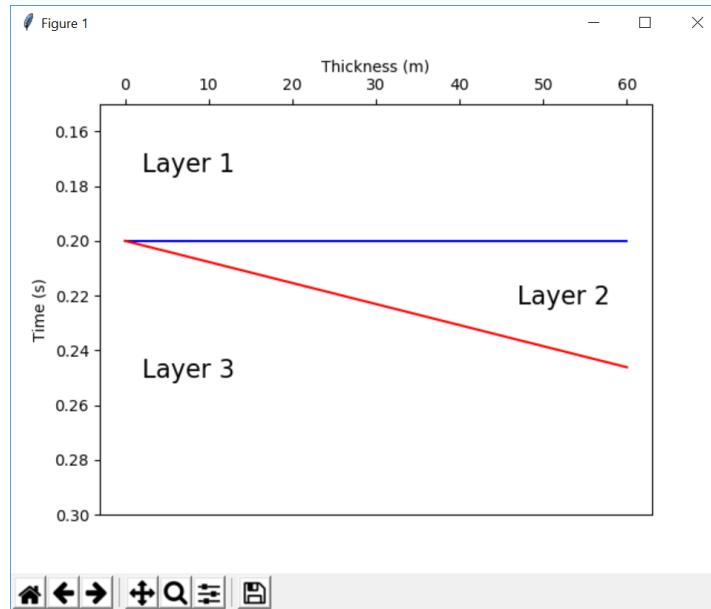
plt.text(dz_max / dz_step - 2,
         lyr_times[-1, 0] + (lyr_times[-1, 1] - lyr_times[-1, 0]) / 2.,
         'Layer 2',
         fontsize=16,
         horizontalalignment='right')
plt.text(2,
         lyr_times[0, 0] + (max_plot_time - lyr_times[0, 0]) / 2.,
         'Layer 3',
         fontsize=16)
plt.gca().xaxis.tick_top()
plt.gca().xaxis.set_label_position('top')
plt.show()

```

Fungsi *calc_times* diatas adalah untuk menghitung *interface time* (*t_int*) atau waktu tempuh setiap batas lapisan sebagai persamaan dari $\frac{z}{v_p}$ dimana nint adalah jumlah *interface* dan *z_int* adalah kedalaman setiap *interface*. Untuk lapisan berikutnya, maka ketebalan lapisan diperoleh dari $z_{diff} = z_{int_i} - z_{int_{i-1}}$ dan waktu tempuh sebagai $\left(\frac{(2 \times z_{diff})}{v_p} + t_{int_{i-1}}\right)$.

Model pembajian dimulai dari kedalaman lapisan kedua yang membaji dari 500 m (*z_int*) hingga 560 m (*z_int* + *dz_min* + (*dz_step* × *model_step*)) dimana ketebalan minimum (*dz_min*) 0 m dan langkah perhitungan antar *trace* (*dz_step*) 1 m. Perhitungan dilakukan sebanyak $\left(int\left(\frac{dz_max-dz_min}{dz_step}\right) + 1\right)$, dalam contoh ini adalah $\left(\frac{60-0}{1} + 1\right)$ atau 61 step. *Interface time* sendiri disimpan dalam array *lyr_times* untuk setiap lapisannya.

Sehingga diperoleh model pembajian sebagai berikut:



Gambar 84 - Model pembajian

Model tersebut digunakan untuk membuat sintetik seismik dengan cara melakukan konvolusi antara batas lapisan (koefisien refleksi) dengan *Ricker wavelet* berdasarkan yang telah dijelaskan sebelumnya. Persamaan amplitudo *Ricker wavelet* adalah sebagai berikut:

$$A = (1 - 2\pi^2 f^2 t^2) e^{-\pi^2 f^2 t^2}$$

Eq. 29

Parameter yang digunakan dalam perhitungan tersebut adalah: *wavelet length* = 0.128 s, *central frequency* (f) = 30 Hz, fasa *wavelet* = 0°, dan *sampling rate* (dt) = 0.0001 s.

Jumlah *sampling rate* (nsamp) adalah $\left(\text{int}\left(\frac{\text{wavelet length}}{\text{sampling rate}}\right) + 1 \right)$.

Ricker wavelet memiliki pusat atau amplitudo maksimum *main lobe* yang berada pada *time* 0 s, kemudian *batas side lobe* pada $\pm \left(\frac{\text{wavelet length}}{2}\right)$, sehingga *time* (t) pada persamaan diatas adalah dari waktu minimum ke maksimum *side lobe* untuk setiap *sampling rate*, atau ditulis

```
t = np.linspace(-wvlt_length / 2, (wvlt_length - dt) / 2, wvlt_length / dt)
```

Jika fasa *wavelet* bukan 0°, maka persamaan *Ricker wavelet* menggunakan *Hilbert transform* sebagai berikut:

$$F(t) = \frac{1}{\pi t} f(t)$$

Eq. 30

```
import matplotlib.pyplot as plt

# Ricker Wavelet Parameters
wvlt_length = 0.128
wvlt_cfreq = 30.0
wvlt_phase = 0.0
dt = 0.0001

def ricker(cf freq, phase, dt, wvlt_length):
    import numpy as np
    import scipy.signal as signal

    nsamp = int(wvlt_length / dt + 1)
    t_max = wvlt_length / 2
    t_min = -t_max

    t = np.arange(t_min, t_max, dt)

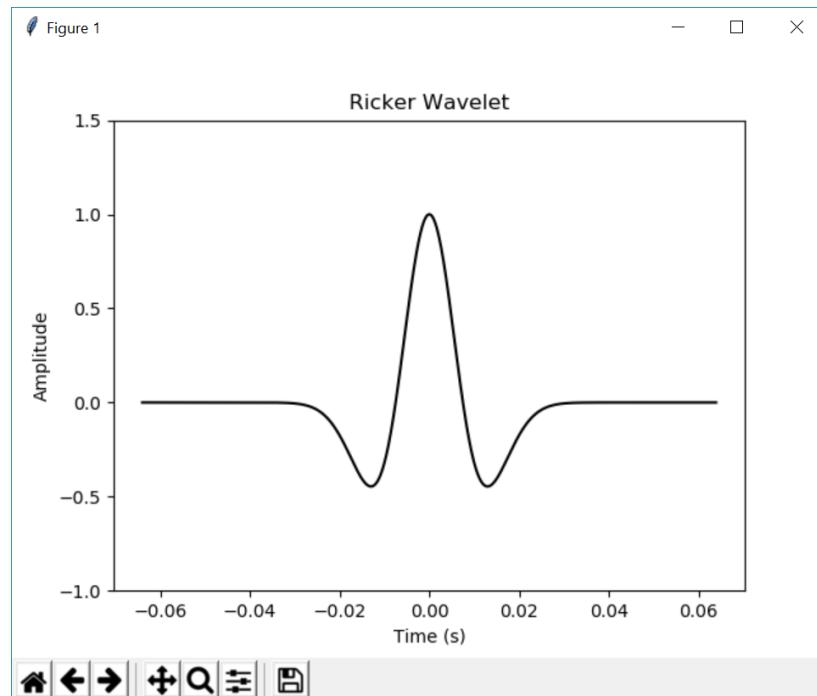
    t = np.linspace(-wvlt_length / 2, (wvlt_length - dt) / 2,
                    wvlt_length / dt)
    wvlt = (1.0 - 2.0 * (np.pi ** 2) * (cfreq ** 2) * (t ** 2)) *
    np.exp(-(np.pi ** 2) * (cfreq ** 2) * (t ** 2))

    if phase != 0:
        phase = phase * np.pi / 180.0
        wvlth = signal.hilbert(wvlt)
        wvlth = np.imag(wvlth)
        wvlt = np.cos(phase) * wvlt - np.sin(phase) * wvlth

    return t, wvlt
```

```
# Generate ricker wavelet
wvlt_t, wvlt_amp = ricker(wvlt_cfreq, wvlt_phase, dt, wvlt_length)

plt.plot(wvlt_t,wvlt_amp, 'k-')
plt.title('Ricker Wavelet')
plt.xlabel('Time (s)')
plt.ylabel('Amplitude')
plt.ylim(-1,1.5)
plt.show()
```



Gambar 85 - Ricker wavelet untuk wedge model

Dengan mengetahui kecepatan dan densitas setiap lapisan, dapat diketahui koefisien refleksi (RC) sebagaimana disebutkan pada persamaan sebelumnya, dengan *script* sebagai berikut:

```

# 3-Layer Model Parameters [Layer1, Layer2, Layer 3]
vp_mod = [2500.0, 2600.0, 2550.0] # P-wave velocity (m/s)
rho_mod = [1.95, 2.0, 1.98] # Density (g/cc)

def calc_rc(vp_mod, rho_mod):
    """
    rc_int = calc_rc(vp_mod, rho_mod)

    nlayers = len(vp_mod)
    nint = nlayers - 1

    rc_int = []
    for i in range(0, nint):
        buf1 = vp_mod[i + 1] * rho_mod[i + 1] - vp_mod[i] * rho_mod[i]
        buf2 = vp_mod[i + 1] * rho_mod[i + 1] + vp_mod[i] * rho_mod[i]
        buf3 = buf1 / buf2
        rc_int.append(buf3)

    return rc_int

# Calculate reflectivities from model parameters
rc_int = calc_rc(vp_mod, rho_mod)

print(rc_int)

```

Sehingga diperoleh koefisien refleksi antara lapisan 1 dan 2 dengan angka 0.032 dan antara lapisan 2 dan 3 dengan angka -0.014. Koefisien refleksi tersebut kemudian di-digitasi dengan menggunakan fungsi berikut (catatan: digitasi terhubung dengan perhitungan t_{int} yang sebelumnya dijalankan oleh fungsi *calc_times*):

```

def digitize_model(rc_int, t_int, t):
    """
    rc = digitize_model(rc, t_int, t)

    rc = reflection coefficients corresponding to interface times
    t_int = interface times
    t = regularly sampled time series defining model sampling
    """

    import numpy as np

    nlayers = len(rc_int)
    nint = nlayers - 1
    nsamp = len(t)

    rc = list(np.zeros(nsamp, dtype='float'))
    lyr = 0

    for i in range(0, nsamp):

        if t[i] >= t_int[lyr]:
            rc[i] = rc_int[lyr]
            lyr = lyr + 1

        if lyr > nint:
            break

    return rc

# Trace Parameters
tmin = 0.0
tmax = 0.5
dt = 0.0001 # changing this from 0.0001 can affect the display quality

rc_zo = []
for model in range(0, nmodel):

    # Calculate interface depths
    z_int = [500.0]
    z_int.append(z_int[0] + dz_min + dz_step * model)

    # Calculate interface times
    t_int = calc_times(z_int, vp_mod)

    # Digitize 3-layer model
    nsamp = int((tmax - tmin) / dt) + 1
    t = []
    for i in range(0, nsamp):
        t.append(i * dt)

    rc = digitize_model(rc_int, t_int, t)
    rc_zo.append(rc)

```

Digitasi dilakukan pada *time* 0 – 0.5 s dengan *sampling rate* 0.0001 s atau 0.1 ms sehingga diperoleh 5001 digit. Sehingga pada ketebalan maksimum, batas lapisan 1 dan 2 akan berada pada digit 2000 atau 0.2 s, pada kedalaman batas lapisan pada 500 m dengan kecepatan gelombang P lapisan pertama adalah 2500 m/s ($t_{int} = \frac{z_{int}}{v_p} = \frac{500}{2500} = 0.2 \text{ s}$).

Sedangkan batas lapisan 2 dan 3 akan dihitung dari $z_{diff} = z_{int_i} - z_{int_{i-1}}$. Misalnya pada batas maksimum di ketebalan 60 m, maka $z_{diff} = 560 - 500 \text{ m}$, kemudian *interface time* berada pada $\left(\frac{(2 \times z_{diff})}{v_p} + t_{int_{i-1}}\right) = \left(\frac{(2 \times 60)}{2600} + 0.2\right) = \left(\frac{120}{2600} + 0.2\right) = 0.0462 + 0.2 = 0.2462$, atau pada digit ke-2462.

Setelah di-digitasi, kemudian dilakukan konvolusi antara koefisien refleksi (rc) dan *Ricker wavelet* (wvlt_amp) dengan *command np.convolve* sebagaimana yang dijelaskan pada bagian *Convolution* sebelumnya. Kemudian disimpan hasil konvolusi dalam *array syn_zo* dengan dimensi (61 x 5001) sebagai koordinat antara *step* untuk ketebalan dan *digit* untuk waktu.

```
syn_zo = []
for model in range(0, nmodel):

    # Convolve wavelet with reflectivities
    syn_buf = np.convolve(rc, wvlt_amp, mode='same')
    syn_buf = list(syn_buf)
    syn_zo.append(syn_buf)

syn_zo = np.array(syn_zo)
t = np.array(t)
```

Fungsi untuk mem-visualisasikan konvolusi tersebut adalah:

```
# Plot Parameters
min_plot_time = 0.15
max_plot_time = 0.3
excursion = 2

[ntrc, nsamp] = syn_zo.shape

def plot_vawig(axhdl, data, t, excursion, highlight=None):
    import numpy as np
    import matplotlib.pyplot as plt

    [ntrc, nsamp] = data.shape

    t = np.hstack([0, t, t.max()])

    for i in range(0, ntrc):
        tbuf = excursion * data[i] / np.max(np.abs(data)) + i

        tbuf = np.hstack([i, tbuf, i])

        if i == highlight:
            lw = 2
        else:
            lw = 0.5

        axhdl.plot(tbuf, t, color='black', linewidth=lw)

        plt.fill_betweenx(t, tbuf, i, where=tbuf > i, facecolor=[0.6,
0.6, 1.0], linewidth=0)
        plt.fill_betweenx(t, tbuf, i, where=tbuf < i, facecolor=[1.0,
0.7, 0.7], linewidth=0)
```

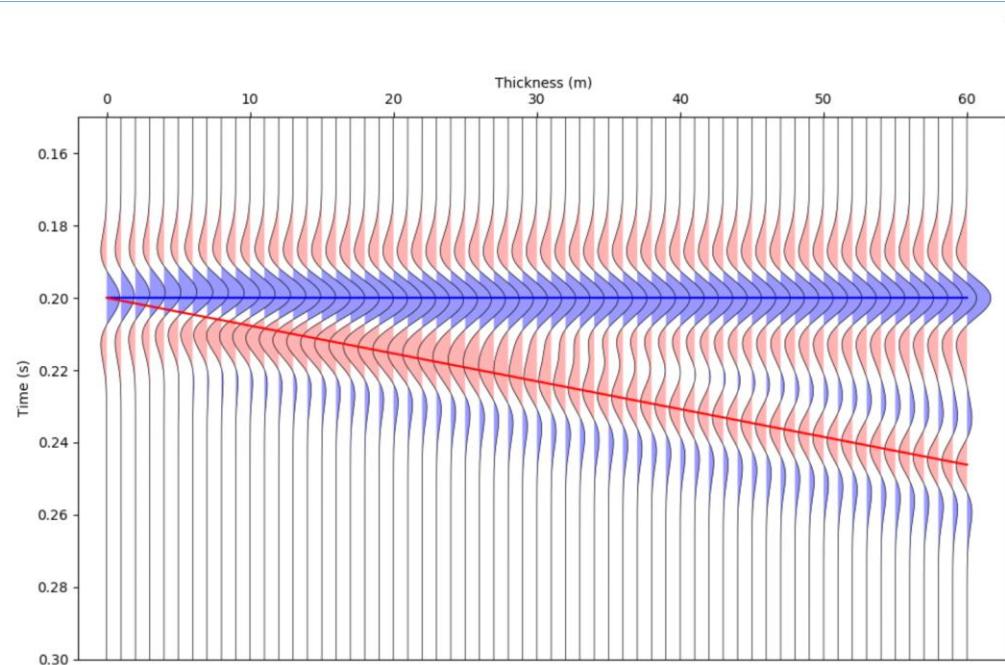
```

axhdl.set_xlim((-excursion, ntrc + excursion))
axhdl.xaxis.tick_top()
axhdl.xaxis.set_label_position('top')
axhdl.invert_yaxis()

fig = plt.figure(figsize=(12, 14))
ax1 = fig.add_subplot(1,1,1)
plot_vawig(ax1, syn_zo, t, excursion)
ax1.plot(lyr_times[:, 0], color='blue', lw=1.5)
ax1.plot(lyr_times[:, 1], color='red', lw=1.5)
ax1.set_ylim((min_plot_time, max_plot_time))
ax1.invert_yaxis()
ax1.set_xlabel('Thickness (m)')
ax1.set_ylabel('Time (s)')
plt.show()

```

Hasil konvolusi tersebut ditunjukkan oleh gambar berikut:



Gambar 86 - Seismic trace model pembajian

Posisi digit *layer* pada setiap *trace* disimpan dalam array *lyr_index*:

```
lyr_idx = np.array(np.round(lyr_times / dt), dtype='int16')
```

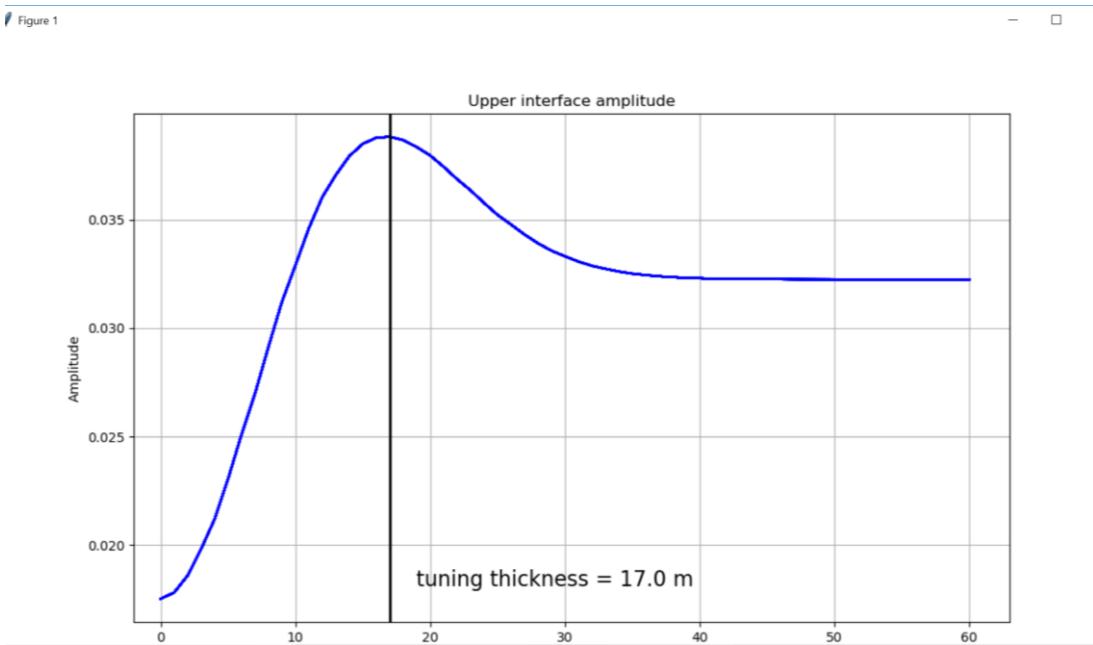
Langkah berikutnya adalah menentukan *tuning thickness*, yaitu dengan mencari posisi *trace* yang memiliki nilai amplitudo tertinggi.

```
tuning_trace = np.argmax(np.abs(syn_zo.T)) % syn_zo.T.shape[1]
tuning_thickness = tuning_trace * dz_step
```

```

plt.plot(syn_zo[:, lyr_idx[:, 0]], color='blue')
plt.xlim((-excursion, ntrc + excursion))
plt.axvline(tuning_trace, color='k', lw=2)
plt.grid()
plt.title('Upper interface amplitude')
plt.xlabel('Thickness (m)')
plt.ylabel('Amplitude')
plt.text(tuning_trace + 2,
         plt.ylim()[0] * 1.1,
         'tuning thickness = {0} m'.format(str(tuning_thickness)),
         fontsize=16)
plt.show()

```



Gambar 87 - Nilai amplitudo maximum sebagai tuning thickness

Silakan akses *full script* pada link berikut:

https://github.com/seg/tutorials-2014/blob/master/1412_Tuning_and_AVO/tuning_wedge.py

Courtesy of Matt Hall, Agile Scientific.

First Break Picking

Dalam pengolahan data seismik, data yang direkam adalah waktu tiba gelombang yang merambat di bawah permukaan yang direkam oleh *receiver*, yang disebut dengan *first arrival time (first break)*. Waktu tiba pertama tersebut ditentukan dengan cara *picking*, yang dapat dilakukan secara manual maupun otomatis.

Sebelum menjalankan program, terlebih dahulu mengunduh *package* PyOSGPUP (yang dibuat oleh Teknik Geofisika Universitas Pertamina) dari <https://sites.google.com/site/metkomup/pyosgpup> untuk membaca SEGY file dalam hal ini. *Copy file* tersebut ke dalam folder interpreter Python, atau PycharmProjects.

Kemudian pada PyCharm, pilih **Terminal** dan pastikan di lokasi yang sama dengan PyOSGPUP tuliskan **pip install PyOSGPUP***

Setelah berhasil, baru kemudian *script* dapat dijalankan, dengan *input* suatu *file* berupa *shotpoint gather* dalam format SEGY.

Pada data dibawah ini, diketahui suatu *shotpoint gather* dengan *sampling interval* = 4 ms (jumlah *sampling* 501), *receiver interval* = 50 m. Namun *picking* yang dilakukan adalah dengan interval 5 *trace* dari *trace* ke-140 hingga *trace* ke-255. *Picking* menggunakan **ginput**, dengan keterangan:

<code>mouse_add=1</code>	klik kiri untuk menambahkan data <i>picking</i>
<code>mouse_pop=3</code>	klik kanan untuk menghapus data <i>picking</i> di <i>trace</i> yang terakhir di-pick
<code>mouse_stop=2</code>	klik tengah untuk menyelesaikan <i>picking</i>

```

import PyOSGPUP.segypy as segypy
import matplotlib.pyplot as plt
import numpy as np

#Read SEGY File
[data, SH, STH] = segypy.readSegy('shot.sgy')
print(SH) #ukuran data (501 x 2930)

if 'time' in SH:
    time = SH['time']           #time
    ntraces = SH['ntraces']     #nomor trace
    ns = SH['ns']               #nomor sampel
else:
    ns = data.shape[0]*4
    time = np.arange(ns)
    ntraces = data.shape[1]
print('number of traces: ', ntraces)
print('record length: ', (ns-1)*4, ' ms')

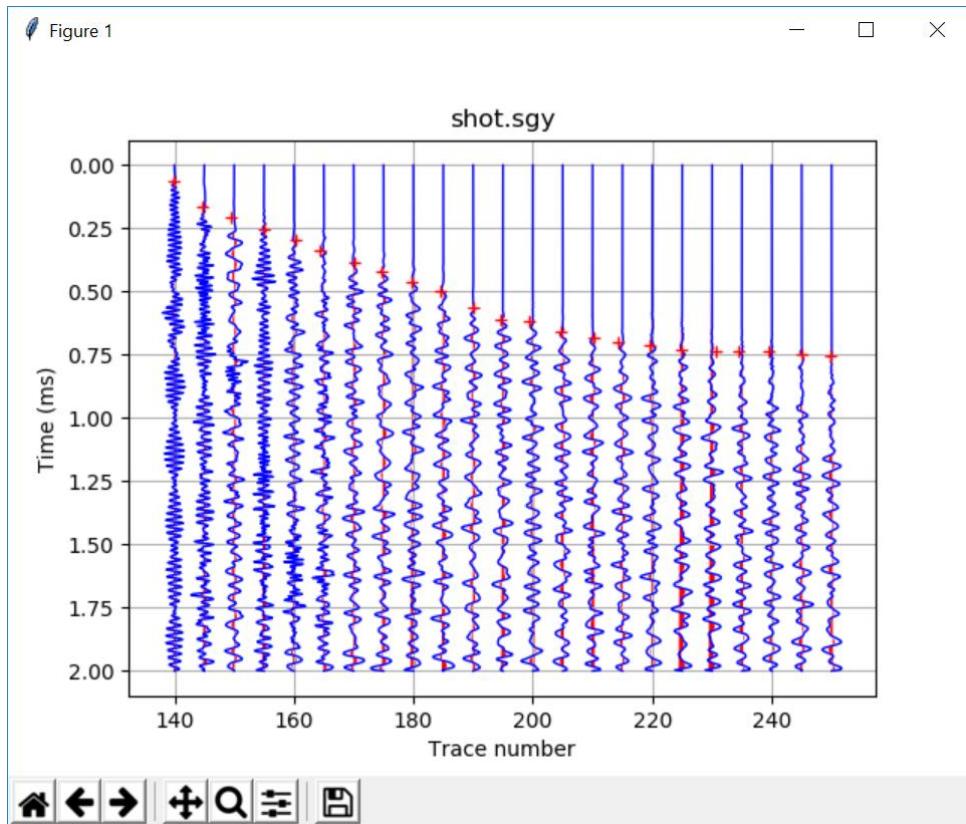
#Plot
maxval = -1
if maxval <= 0:
    Dmax = np.max(data)
    maxval = -1*maxval*Dmax
fig = plt.figure(1)
for i in range(140,255,5):
    trace = data[:,i] * 3
    trace[0] = 0
    trace[ns-1] = 0
    plt.plot(i + trace/maxval, time, color='b', linewidth=1, picker=10)
    for a in range(len(trace)):
        if trace[a] < 0:
            trace[a] = 0
    plt.fill(i + data[:,i]/maxval, time, 'r', linewidth=1)

plt.grid(True)
plt.ylim(-0.1, 2.1)
plt.gca().invert_yaxis()
plt.xlabel('Trace number')
if 'time' in SH:
    plt.ylabel('Time (ms)')
else:
    plt.ylabel('Sample number')
if 'filename' in SH:
    plt.title(SH['filename'])

plt.xlabel('Trace number')
if 'time' in SH:
    plt.ylabel('Time (ms)')
else:
    plt.ylabel('Sample number')
if 'filename' in SH:
    plt.title(SH['filename'])

print("Pick by clicking the first arrival time of every trace")
result = plt.ginput(n=-1, mouse_add=1, mouse_pop=3, mouse_stop=2)
print("Result", result)
plt.show()

```



Gambar 88 - First Break Picking

Sementara untuk menyimpan hasil *picking*, digunakan **savetxt** untuk menyimpan ke dalam file dalam format ASCII. Kolom pertama (x) merupakan nomor *trace* dan kolom kedua (y) merupakan *time* untuk lokasi *first break*.

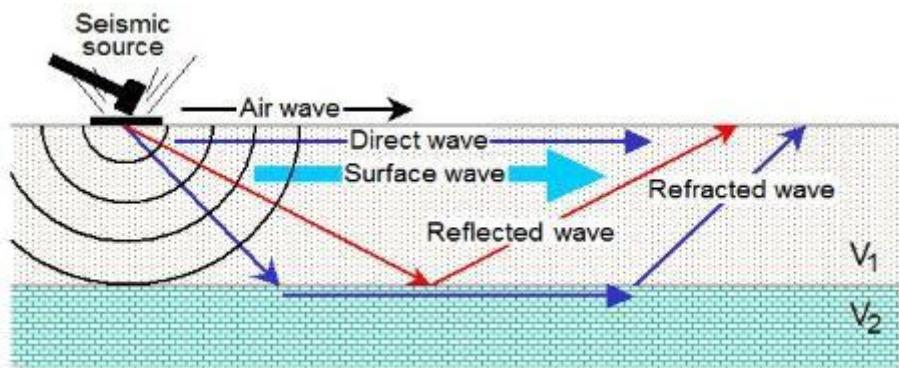
```
np.savetxt('picking.txt', result, fmt='%.4f')
dt = np.genfromtxt('picking.txt')
x = dt[:,0]
y = dt[:,1]
```

1	140.1194	0.0631
2	144.6583	0.1464
3	149.4492	0.1940
4	154.7446	0.2119
5	160.2920	0.2833
6	163.8222	0.3131
7	169.8740	0.3548
8	173.6564	0.3964
9	179.4560	0.4440
10	184.2470	0.4976
11	189.5423	0.5452
12	194.0811	0.5988
13	198.8721	0.6226
14	204.6717	0.6583
15	209.7148	0.6881
16	214.7580	0.7000
17	219.2968	0.7060

Gambar 89 - Hasil first break picking

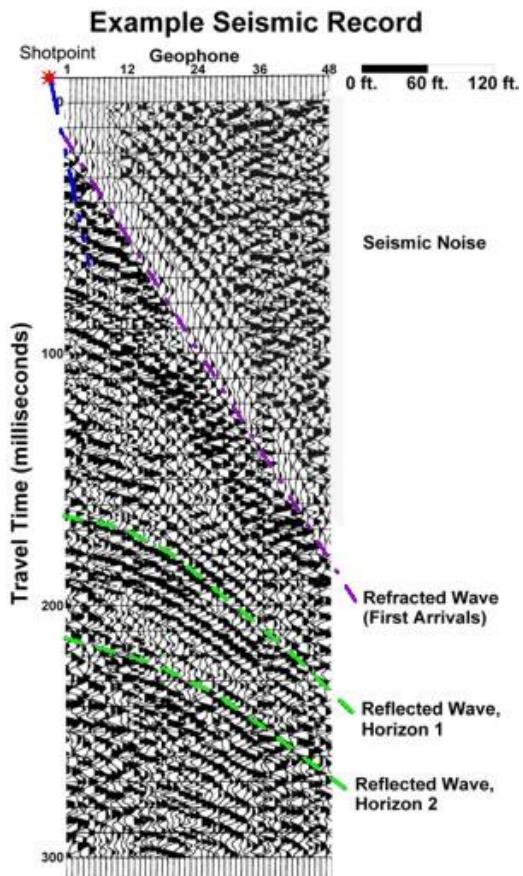
Kecepatan dan Ketebalan Lapisan Gelombang Seismik Refraksi

Sebagaimana kita ketahui, bahwa gelombang seismik di bawah permukaan bumi dapat merambat langsung, direfraksikan dan direfleksikan seperti yang ditunjukkan oleh gambar berikut:



Gambar 90 - Tipe perambatan gelombang seismik (Sumber: <http://masw.com/Whatisseismsurvey.html>)

Jika ditampilkan dalam *shotpoint gather* (kumpulan *trace* seismik yang berasal dari satu *shotpoint* yang sama), pada tampilan *trace* urut berdasarkan lokasi *receiver* atau *offset* (jarak antara *source* dan *receiver*), maka gelombang refraksi dan refleksi dapat dilihat sebagai berikut:

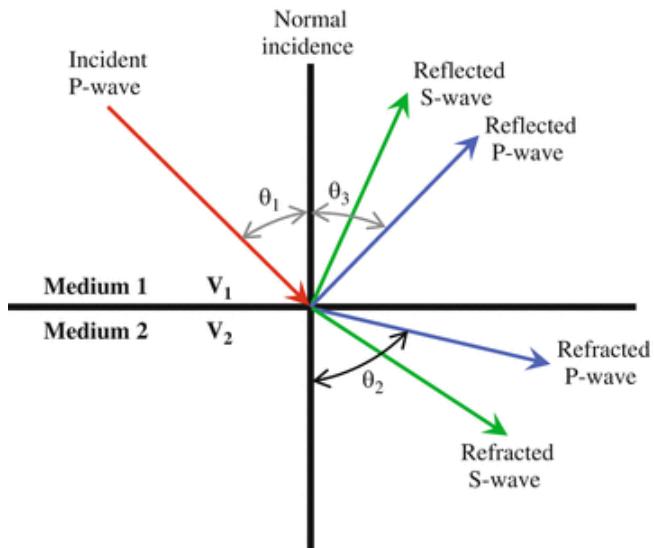


Gambar 91 - Shot gather dan perambatan gelombang (Sumber: <http://www.enviroscan.com/home/seismic-refraction-versus-reflection>)

Refleksi dan refraksi gelombang seismik pada bidang batas lapisan ditunjukkan dalam Hukum Snell sebagai berikut:

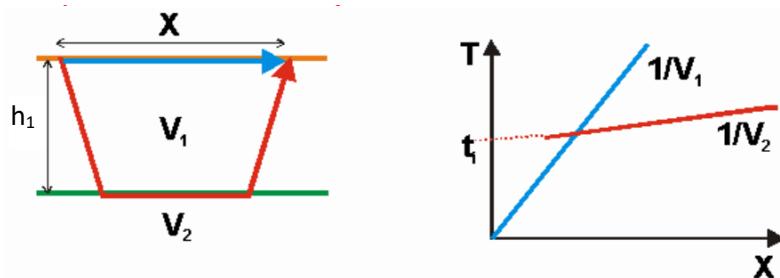
$$\frac{\sin \theta_1}{v_1} = \frac{\sin \theta_2}{v_2}$$

Dimana θ_1 adalah sudut antara garis normal dengan gelombang datang (lapisan 1), θ_2 adalah sudut antara garis normal dengan gelombang refraksi (lapisan 2), v_1 dan v_2 masing-masing adalah kecepatan lapisan 1 dan 2.



Gambar 92 - Hukum Snell

Seperti yang ditunjukkan dalam *shotpoint gather*, kecepatan gelombang seismik dapat dihitung sebagai gradien antara *offset* dan waktu tempuh gelombang seismik. Pada kasus dua lapisan, maka ketebalan lapisan pertama dapat dihitung sebagai berikut:



Gambar 93 - Ilustrasi seismik refraksi pada dua lapisan

Sumber: <http://www.ukm.my/rahim/Seismic%20Refraction%20Surveying.htm>

$$t_i = 2h_1 \sqrt{\frac{1}{v_1^2} - \frac{1}{v_2^2}}$$

Eq. 31

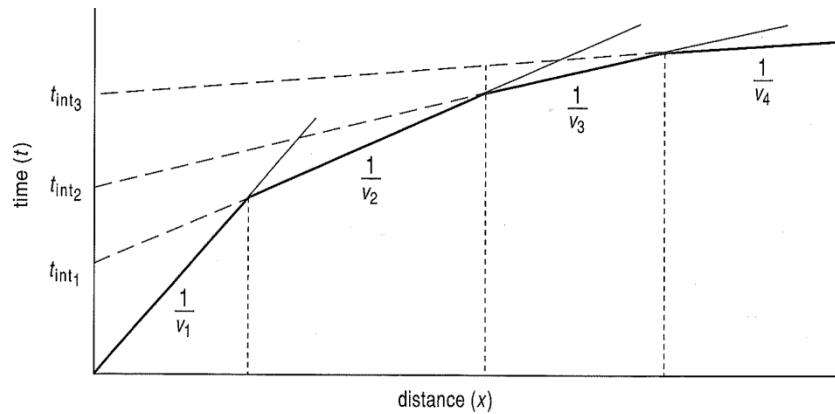
Keterangan:

h_1 = ketebalan lapisan 1

v_1, v_2 = kecepatan lapisan 1 dan 2

t_i = waktu *intercept, time* saat v_2 pada $x = 0$

Berikut adalah persamaan untuk kasus lebih dari dua lapisan:



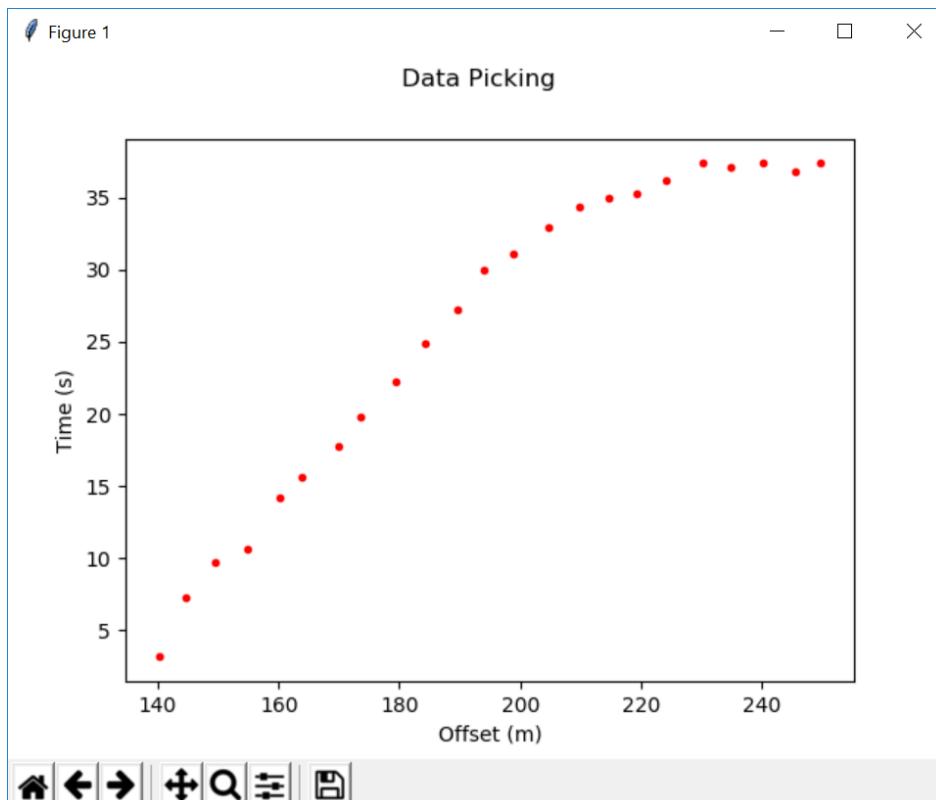
Gambar 94 - Ilustrasi seismik refraksi pada beberapa lapisan

$$t_{i_2} = 2h_1 \sqrt{\frac{1}{v_1^2} - \frac{1}{v_2^2}} + 2h_2 \sqrt{\frac{1}{v_2^2} - \frac{1}{v_3^2}}$$

$$t_{i_n} = \sum_{i=1}^n 2h_i \sqrt{\frac{1}{v_i^2} - \frac{1}{v_{i+1}^2}}$$

Eq. 32

Untuk menghitung kecepatan dan ketebalan lapisan, terlebih dahulu dilakukan *first break picking* seperti yang dijelaskan sebelumnya, sebagai *input* perhitungan.



Gambar 95 - Gradien kecepatan pada grafik time vs offset

```

#script first break picking#

from scipy.sparse.linalg import lsqr

offset = 50 * y
plt.plot(x,offset,'.r')
fig = plt.figure(1)
fig.suptitle('Data Picking')
plt.xlabel('Offset (m)')
plt.ylabel('Time (s)')
print("Select two slope (start-end points)")
slope = plt.ginput(4)
print("Slope", slope)
plt.show()

#Ketebalan lapisan
np.savetxt('slope.txt', result, fmt='%.4f')
dt2 = np.genfromtxt('slope.txt')
offset = dt2[:,0]
time = dt2[:,1]
v1 = (offset[1]-offset[0])/(time[1]-time[0])
v2 = (offset[3]-offset[2])/(time[3]-time[2])
x_lin = np.array([offset[3],offset[2]])
y_lin = np.array([time[2],time[3]])
x_lin = np.column_stack((x_lin, np.ones(np.shape(x_lin)[0])))
grad = lsqr(x_lin, y_lin)[0]
ti = grad[1]

h1 = ti / (2* (((1/v1)**2)-((1/v2)**2))**0.5)
print('Kecepatan lapisan pertama = ', v1, ' m')
print('Kecepatan lapisan kedua = ', v2, ' m')
print('Ketebalan lapisan pertama = ', h1, ' m')

```

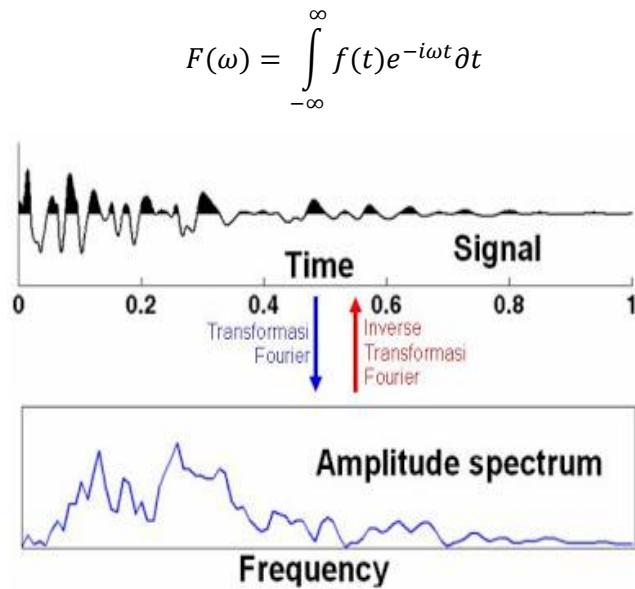
```

Kecepatan lapisan pertama = 49.39645522388057 m
Kecepatan lapisan kedua = 127.29086538461533 m
Ketebalan lapisan pertama = 38.47050045880338 m

```

Fast Fourier Transform

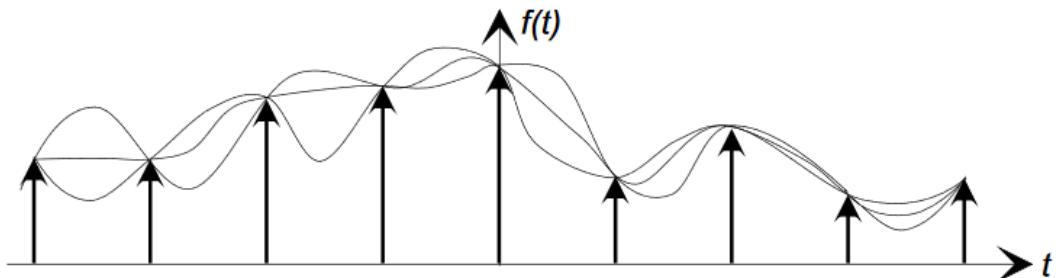
Data seismik merupakan data waktu tempuh vs jarak, namun dapat ditampilkan dalam spektrum frekuensi dengan melakukan transformasi Fourier dengan persamaan sebagai berikut:



Gambar 96 – Transformasi Fourier dari Time ke Frequency domain

Prinsip FFT adalah, setiap frekuensi dihasilkan dari waktu dengan interval yang sama (*sampling rate*) berdasarkan persamaan diatas. Jika diketahui *sampling interval* ΔT , maka *sampling frequency* adalah $F_s = \frac{1}{\Delta T}$ (yang lebih besar daripada $2F_{max}$ (frekuensi maksimum pada satu siklus) yang menyebabkan frekuensi yang lebih besar daripada *sampling frequency* memiliki sifat *aliasing*. Sedangkan setiap sisi lebih kecil dan lebih besar daripada F_s memiliki sifat *folding*, yang dibatasi oleh Frekuensi Nyquist sebesar $F_N = \frac{1}{2}F_s = \frac{1}{2\Delta T}$

Modul yang digunakan untuk FFT pada Python adalah **fft** dari **scipy.fftpack**.



Gambar 97 - Ilustrasi FFT

Sumber: https://ocw.mit.edu/courses/mechanical-engineering/2-161-signal-processing-continuous-and-discrete-fall-2008/lecture-notes/lecture_10.pdf

```

import PyOSGPUP.segypy as segypy
import matplotlib.pyplot as plt
from scipy.fftpack import fft
import numpy as np

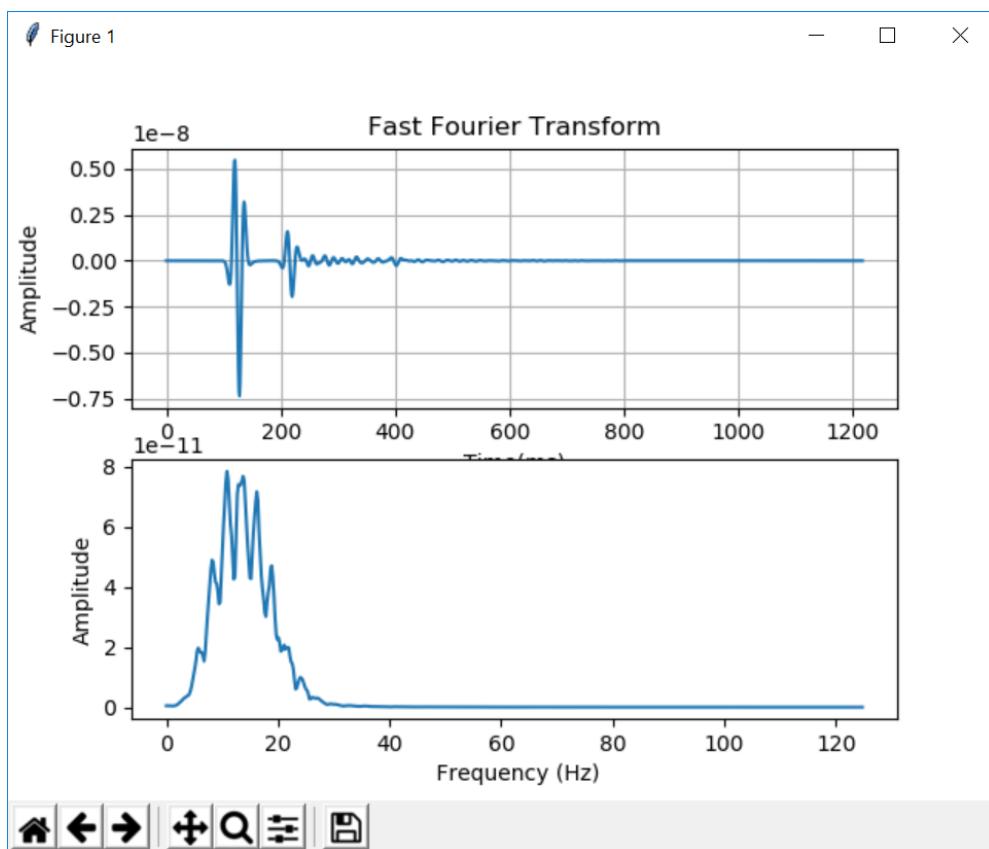
# Read Segy File
[data, SH, STH] = segypy.readSegy('shotgather.sgy')

plt.subplot(211)
seismic = data[:, 7]
plt.plot(seismic)
plt.title("Fast Fourier Transform")
plt.xlabel("Time (ms)")
plt.ylabel("Amplitude")
plt.grid()

#fft
T = 0.004
Fs = 1/T
L = len(seismic)
Y = fft(seismic)
P1 = np.abs(Y/L)
P2 = P1[0:int(L/2)]
f = Fs * np.arange(0, int(L/2))/L

plt.subplot(212)
plt.plot(f, P2)
plt.xlabel("Frequency (Hz)")
plt.ylabel("Amplitude")
plt.show()

```



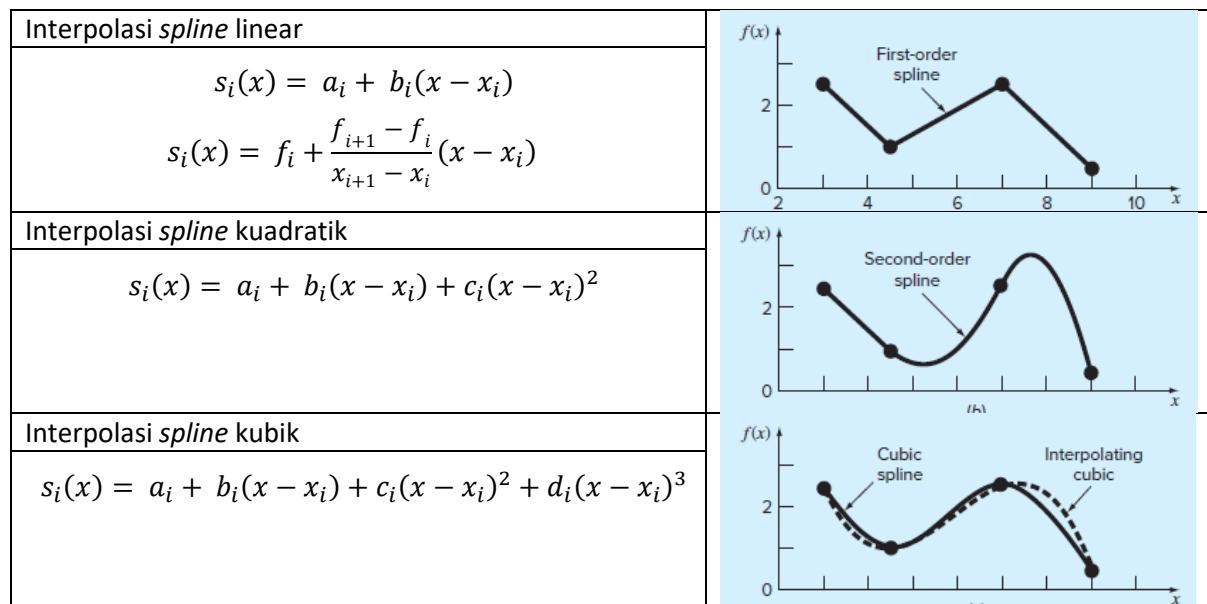
Gambar 98 - Hasil FFT Data Seismik

Interpolasi Spline pada Data Gamma-Ray

Interpolasi merupakan penentuan nilai di suatu titik yang belum diketahui, dengan menggunakan nilai di titik-titik lain di sekitarnya. Metode *spline* adalah metode interpolasi yang menghasilkan *error* yang kecil meskipun dengan derajat polinomial yang kecil.

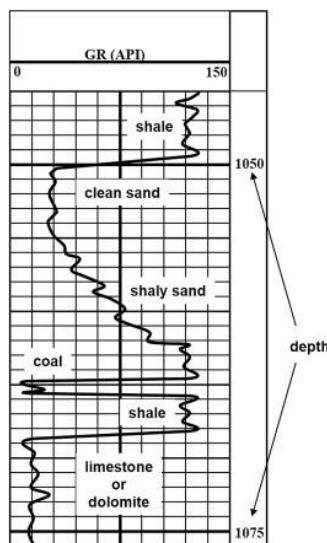
Interpolasi *spline* yang dilakukan adalah interpolasi *spline* linear (polinomial orde-1), kuadratik (polinomial orde-2), dan kubik (polinomial orde-3).

Seperti yang disebutkan dalam tulisan “*Splines and Piecewise Interpolation*” pada “*Applied Numerical methods with MATLAB for Engineers and Scientists*” (Chapra, 2012), interpolasi *spline* memiliki persamaan dan ilustrasi sebagai berikut:



Eq. 33

Data *Gamma Ray* menunjukkan respon batuan terhadap radioaktif, dimana sangat membantu dalam membedakan batupasir (*sand*) dan batulempung (*shale*). Berikut adalah contoh interpretasi log *Gamma Ray*:



Gambar 99 - Interpretasi log Gamma Ray

```

import numpy as np
import scipy.interpolate as interpolasi
import matplotlib.pyplot as plt

import time

start1 = time.time()
data = np.genfromtxt('gammaray.txt')
kedalaman = data[:, 0]
data_Gamma_Ray = data[:, 5]

x = kedalaman
y = data_Gamma_Ray

kedalaman_1 = np.linspace(kedalaman[0], kedalaman[14], num=100)
Gamma_RSp_1 = interpolasi.interp1d(kedalaman[0:15],
data_Gamma_Ray[0:15], kind='slinear')
Gamma_RSp_linier = Gamma_RSp_1(kedalaman_1)
Gamma_RSp_2 = interpolasi.interp1d(kedalaman[0:15],
data_Gamma_Ray[0:15], kind='quadratic')
Gamma_RSp_square = Gamma_RSp_2(kedalaman_1)
Gamma_RSp_3 = interpolasi.interp1d(kedalaman[0:15],
data_Gamma_Ray[0:15], kind='cubic')
Gamma_RSp_cubic = Gamma_RSp_3(kedalaman_1)

plt.subplot(141)
plt.plot(y[0:14], x[0:14], 'ob')
plt.plot(Gamma_RSp_linier, kedalaman_1, 'b')
plt.gca().invert_yaxis()
plt.xlabel("Gamma Ray")
plt.ylabel("Depth (m)")
plt.title("Linear Spline")

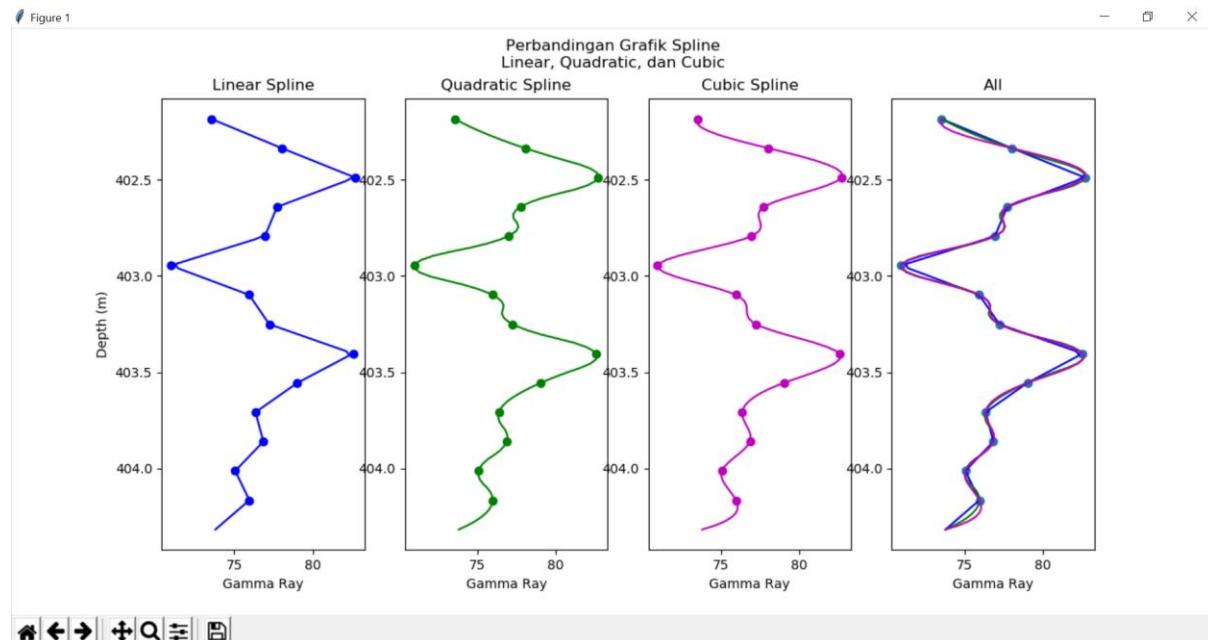
plt.subplot(142)
plt.plot(y[0:14], x[0:14], 'og')
plt.plot(Gamma_RSp_square, kedalaman_1, 'g')
plt.gca().invert_yaxis()
plt.xlabel("Gamma Ray")
plt.title("Quadratic Spline")

plt.subplot(143)
plt.plot(y[0:14], x[0:14], 'om')
plt.plot(Gamma_RSp_cubic, kedalaman_1, 'm')
plt.gca().invert_yaxis()
plt.xlabel("Gamma Ray")
plt.title("Cubic Spline")

plt.subplot(144)
plt.plot(y[0:14], x[0:14], 'o')
plt.plot(Gamma_RSp_linier, kedalaman_1, 'b')
plt.plot(Gamma_RSp_square, kedalaman_1, 'g')
plt.plot(Gamma_RSp_cubic, kedalaman_1, 'm')
plt.gca().invert_yaxis()
plt.xlabel("Gamma Ray")
plt.title("All")
plt.suptitle("Perbandingan Grafik Spline\nLinear, Quadratic, dan Cubic")
plt.show()

```

Data *Gamma Ray* diambil pada setiap sampel satuan waktu, dan kemudian diinterpolasi untuk dapat melihat *log* secara keseluruhan. Script diatas digunakan untuk interpolasi *spline* pada data *Gamma Ray*. Modul yang digunakan pada Python adalah **scipy.interpolate**.



Gambar 100 - Hasil interpolasi spline pada data *Gamma Ray*

Smoothing 1-D dengan Moving Average Data Gamma Ray

Metode *moving average* merupakan metode *forecasting* dengan memanfaatkan nilai rata-rata bergerak yang dibatasi nilai periode atau orde yang ingin diukur. Pemrosesan data bergantung dari nilai *orde* yang dimasukkan dan lebar atau *window* data.

$$F_{t+1} = \frac{\sum Y}{k} = \frac{Y_t + Y_{t-1} + \dots + Y_{t-k+1}}{k}$$

Eq. 34

Dimana:

F_{t+1} = rata- rata
 Y = data ke-t
 k = orde

Berikut adalah ilustrasi perhitungan *Moving Average* dengan 10 data dan periode 3:

Data-1	Data-2	Data-3	Data-4	Data-5	Data-6	Data-7	Data-8	Data-9	Data-10
--------	--------	--------	--------	--------	--------	--------	--------	--------	---------

$$F_1 = \frac{Y_1 + Y_2 + Y_3}{3}$$

$$F_2 = \frac{Y_2 + Y_3 + Y_4}{3}$$

Sehingga kemudian diperoleh model yang memberikan relasi antar data. Berikut adalah *script* yang digunakan untuk *smoothing* secara 1D untuk data Gamma Ray.

```

import numpy as np
import matplotlib.pyplot as plt

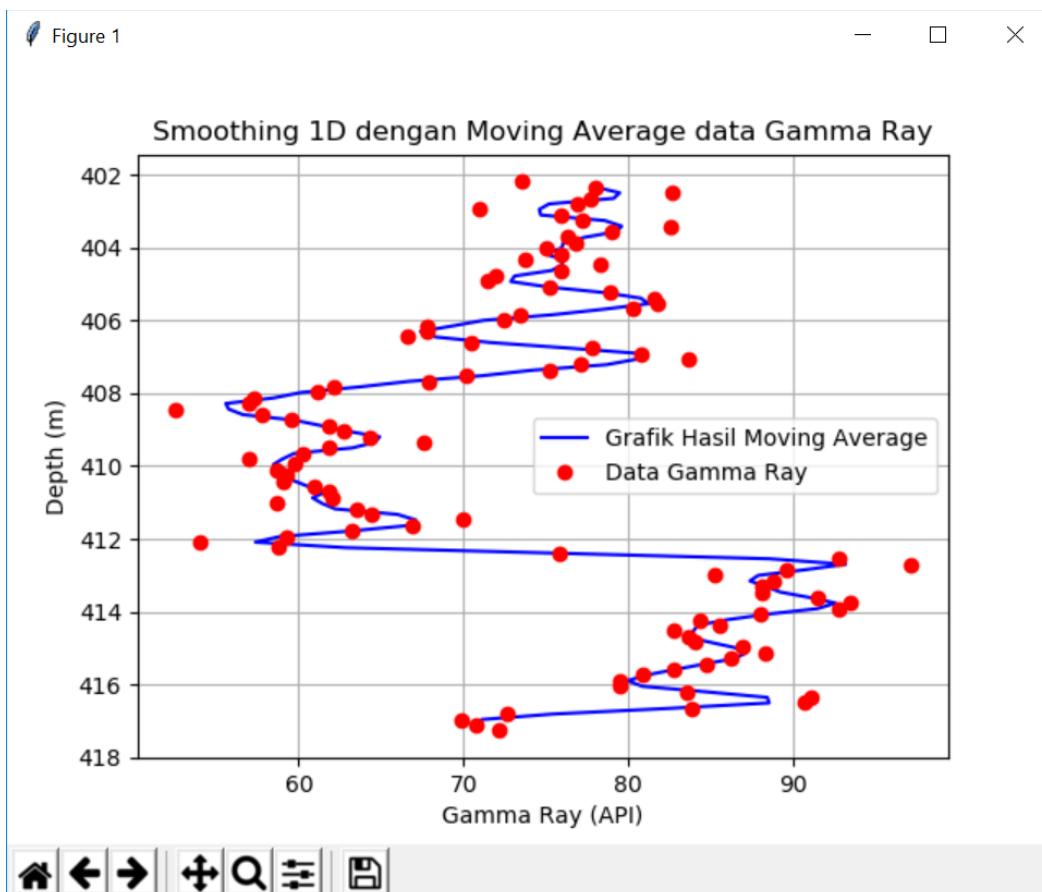
data = np.genfromtxt('gammaray.txt')
depth = data[0:100, 0]
gammaray = data[0:100, 5]
orde = 3

def movingaverage(values, periode): # perhitungan moving average
    mean = np.repeat(1.0, periode) / periode
    sma = np.convolve(values, mean, 'valid')
    return sma

MV_Depth = movingaverage(depth, orde) # Depth MA
MV_Gammaray = movingaverage(gammaray, orde) # Gamma Ray MA

# Plot data asli dan hasil moving average
plt.plot(MV_Gammaray, MV_Depth, color='blue', label='Grafik Hasil Moving Average')
plt.plot(gammaray, depth, 'o', color='red', label='Data Gamma Ray')
plt.gca().invert_yaxis()
plt.ylabel('Depth (m)')
plt.xlabel('Gamma Ray (API)')
plt.legend()
plt.grid()
plt.title('Smoothing 1D dengan Moving Average data Gamma Ray')
plt.show()

```



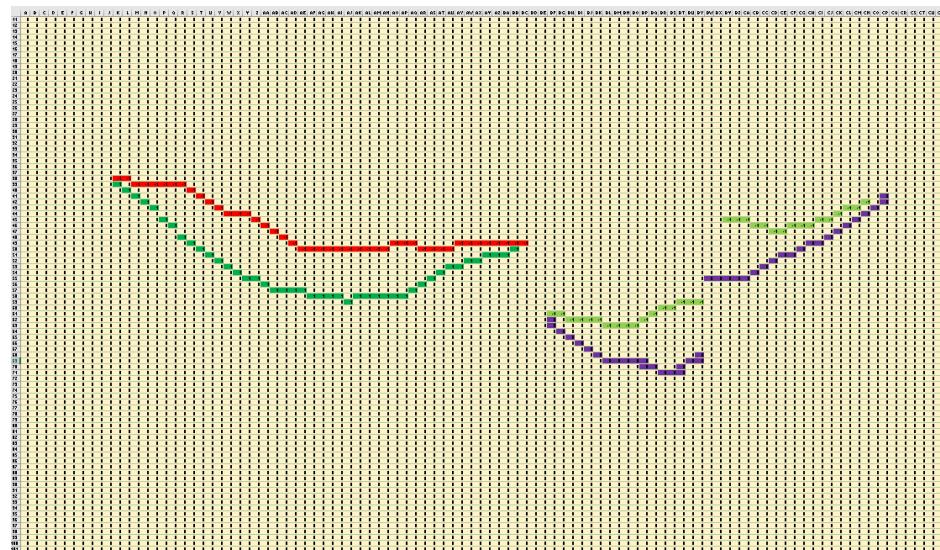
Gambar 101 - Smoothing 1D Moving Average pada Data Gamma Ray

Visualisasi Data Seismik

Data seismik sintetik dapat divisualisasikan dengan mengkonvolusikan antara koefisien refleksi dengan Ricker *wavelet*.

Sebelumnya, suatu data penampang seismik dimodelkan dengan menggunakan Microsoft Office Excel dengan membuat *grid* sebanyak 100×100 dengan seluruh grid memiliki nilai = 0.

Misalkan pada gambar di bawah ini, beberapa grid memiliki nilai (dan warna) yang berbeda, sama dengan nilai koefisien refleksi sebagai ilustrasi dari batas lapisan. Kemudian data tersebut disimpan ke dalam format ASCII (*.txt)



Gambar 102 - Pemodelan Sintetik Penampang Seismik

Pada gambar diatas, warna merah menunjukkan batas lapisan dengan koefisien refleksi (RC) = -0.8, warna hijau tua dengan RC 0.7, warna hijau muda dengan RC -0.5 dan warna ungu dengan RC 0.4. Dengan menyimpan data ke dalam file dengan nama **channel.txt**, berikut adalah *script* yang digunakan untuk mem-visualisasikan penampang data seismik tersebut dengan Python:

```
import numpy as np
import matplotlib.pyplot as plt
cr=np.genfromtxt('channel.txt')
traces=[]

#ricker wavelet
L = 400    # wavelet length (ms)
s = 4      # sampling rate (ms)
f = 25     # frequency (Hz)
t = np.arange(-L/2000,L/2000+s/1000,s/1000) # must be in second
ricker = (1 - 2 * np.pi**2 * f**2 * t**2) *np.exp(-1*np.pi**2 *
f**2*t**2) #ricker wavelet
```

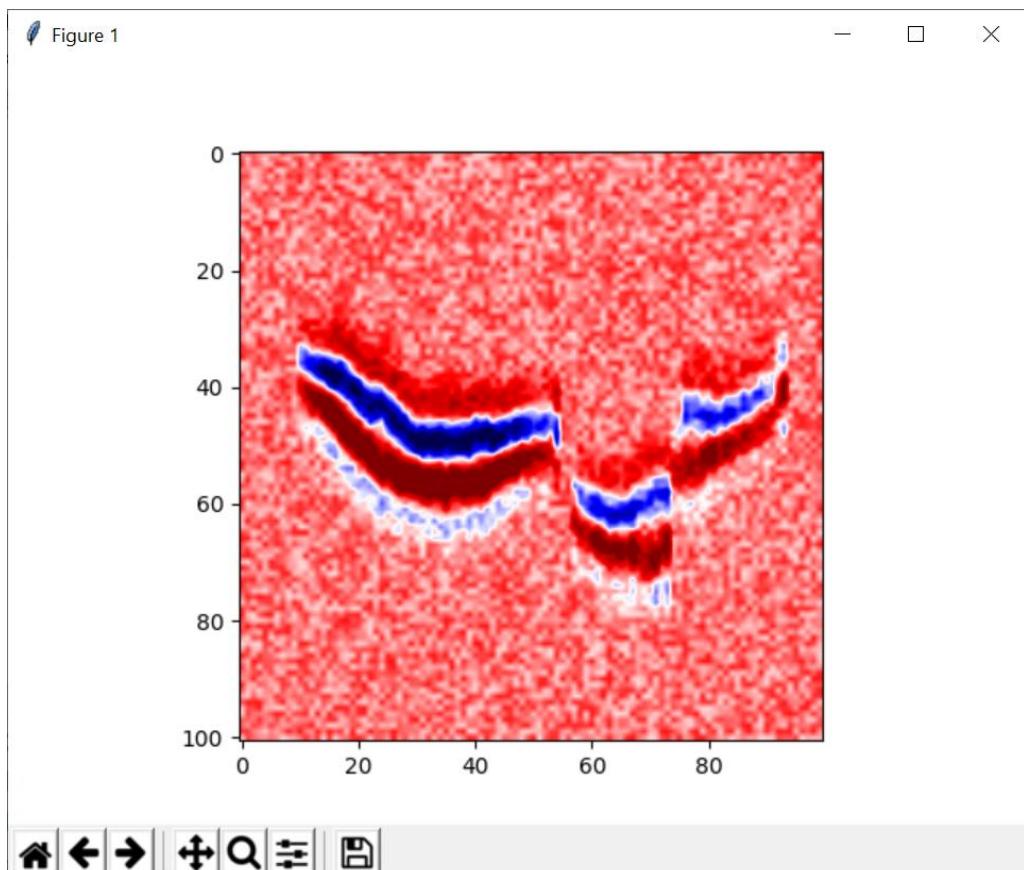
```

for i in range (cr.shape[1]):
    seismic = np.convolve(ricker,cr[:,i], 'same')
    traces.append(seismic)
traces=np.asarray(traces).T

noise=0.3*np.random.random((traces.shape[0],traces.shape[1]))
plt.imshow(traces+noise,cmap='seismic',vmin=-0.8,vmax=0.7,
interpolation='bilinear')
plt.show()

np.save('traces', traces+noise)

```



Gambar 103 - Visualisasi Penampang Data Seismik

Inverse Matrix mxn dengan Singular Value Decomposition

Dekomposisi nilai singular (SVD atau *Singular Value Decomposition*) adalah suatu teknik faktorisasi matriks ukuran apa saja (tidak harus matriks persegi) dengan mengurai suatu matriks ke dalam tiga matriks unit yaitu matriks ortonormal U dan V serta sebuah matriks diagonal S yang berisi faktor skalar yang disebut dengan nilai singular. Metode *inverse matrix* dengan SVD digunakan apabila *determinan* suatu matrix bernilai = 0. Dengan metode ini, sebuah matriks singular A didefinisikan sebagai:

Matrix A	Inverse Matrix A
$A = U S V^T$	$A^{-1} = V S^{-1} U^T$

Eq. 35

Keterangan:

A = matriks singular ukuran $m \times n$

U = vektor singular dari matriks ortonormal $A^T A$ yang berukuran $m \times m$

V = vektor singular dari matriks ortonormal AA^T yang berukuran $n \times n$

S = matriks diagonal berukuran $m \times n$ yang disusun dari akar *eigenvalues* dengan urutan dari yang terbesar ke yang terkecil

Prosedur penentuan SVD:

- Menentukan matriks singular A (mxn) yaitu matriks dengan determinan sama dengan nol.

Misal: $A = \begin{bmatrix} 1 & -2 \\ 1 & -2 \end{bmatrix}$ $A^T = \begin{bmatrix} 1 & 1 \\ -2 & -2 \end{bmatrix}$

- Menentukan matriks $A^T A$ dan AA^T untuk menghitung *eigenvalues* (λ) terlebih dahulu.

$A^T A = \begin{bmatrix} 1 & 1 \\ -2 & -2 \end{bmatrix} \begin{bmatrix} 1 & -2 \\ 1 & -2 \end{bmatrix} = \begin{bmatrix} 2 & -4 \\ -4 & 8 \end{bmatrix}$ <p><i>Eigenvalues</i> matriks $A^T A$:</p> $\det(A^T A - I\lambda) = 0$ $\det(\begin{bmatrix} 2 & -4 \\ -4 & 8 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\lambda) = 0$ $\det(2 - \lambda \quad -4 \\ -4 \quad 8 - \lambda) = 0$ $[(2 - \lambda)(8 - \lambda)] - [(-4)(-4)] = 0$ $(16 - 10\lambda + \lambda^2 - 16) = 0$ $(\lambda^2 - 10\lambda) = 0$ $\lambda(\lambda - 10) = 0$ <p>sehingga $\lambda_1 = 10$ dan $\lambda_2 = 0$</p>	$AA^T = \begin{bmatrix} 1 & -2 \\ 1 & -2 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -2 & -2 \end{bmatrix} = \begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix}$ <p><i>Eigenvalues</i> matriks AA^T :</p> $\det(AA^T - I\lambda) = 0$ $\det(\begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\lambda) = 0$ $\det(5 - \lambda \quad 5 \\ 5 \quad 5 - \lambda) = 0$ $[(5 - \lambda)(5 - \lambda)] - [(5)(5)] = 0$ $(25 - 10\lambda + \lambda^2 - 25) = 0$ $(\lambda^2 - 10\lambda) = 0$ $\lambda(\lambda - 10) = 0$ <p>sehingga $\lambda_1 = 10$ dan $\lambda_2 = 0$</p>
-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

3. Menentukan:

- a. Matriks V yang merupakan *eigenvectors* $A^T A$ yang sudah dinormalisasikan.

Syarat: $(A^T A)e = \lambda e$

$$\begin{bmatrix} 2 & -4 \\ -4 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$2x_1 + (-4x_2) = \lambda x_1 \rightarrow (2-\lambda)x_1 - 4x_2 = 0$$

$$-4x_1 + 8x_2 = \lambda x_2 \rightarrow -4x_1 + (8-\lambda)x_2 = 0$$

$\lambda_1 = 10$ $(2-10)x_1 - 4x_2 = 0$ $-4x_1 + (8-10)x_2 = 0$ $\underline{-8x_1 - 4x_2 = 0}$ $\underline{-4x_1 - 2x_2 = 0} +$ $-12x_1 - 6x_2 = 0$ $-12x_1 = 6x_2$ $x_1 = -0.5x_2$ $x = \begin{bmatrix} -0.5 \\ 1 \end{bmatrix} \sim \hat{e}_1$	$\lambda_2 = 0$ $(2-0)x_1 - 4x_2 = 0$ $-4x_1 + (8-0)x_2 = 0$ $\underline{2x_1 - 4x_2 = 0}$ $\underline{-4x_1 + 8x_2 = 0} +$ $-2x_1 + 4x_2 = 0$ $-2x_1 = -4x_2$ $x_1 = 2x_2$ $x = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \sim \hat{e}_2$
------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Untuk $\lambda_1 = 10$ maka $x = \begin{bmatrix} -0.5 \\ 1 \end{bmatrix} \sim \hat{e}_1$ dan untuk $\lambda_2 = 0$ maka $x = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \sim \hat{e}_2$

Normalisasi *eigenvector*:

$$\bar{e}_1 = \frac{\begin{bmatrix} -0.5 \\ 1 \end{bmatrix}}{\sqrt{(-0.5)^2 + 1^2}} = \frac{\begin{bmatrix} -0.5 \\ 1 \end{bmatrix}}{1.118} = [(-0.447), (0.894)]$$

$$\bar{e}_2 = \frac{\begin{bmatrix} 2 \\ 1 \end{bmatrix}}{\sqrt{2^2 + 1^2}} = \frac{\begin{bmatrix} 2 \\ 1 \end{bmatrix}}{2.236} = [(0.894), (0.447)]$$

$$\text{Sehingga matrix } V = [\bar{e}_1, \bar{e}_2] = \begin{bmatrix} -0.447 & 0.894 \\ 0.894 & -0.447 \end{bmatrix}$$

- b. Matriks U yang merupakan *eigenvectors* AA^T yang sudah dinormalisasikan.

Syarat: $(AA^T)e = \lambda e$

$$\begin{bmatrix} 5 & 5 \\ 5 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$5x_1 + 5x_2 = \lambda x_1 \rightarrow (5-\lambda)x_1 + 5x_2 = 0$$

$$5x_1 + 5x_2 = \lambda x_2 \rightarrow 5x_1 + (5-\lambda)x_2 = 0$$

$\lambda_1 = 10$ $(5-10)x_1 + 5x_2 = 0$ $5x_1 + (5-10)x_2 = 0$ $-5x_1 + 5x_2 = 0$ $5x_1 + (-5x_2) = 0$ $-x_1 = -x_2$ $x = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \sim \hat{e}_1$	$\lambda_2 = 0$ $(5-0)x_1 + 5x_2 = 0$ $5x_1 + (5-0)x_2 = 0$ $5x_1 + 5x_2 = 0$ $\underline{5x_1 + 5x_2 = 0} +$ $10x_1 + 10x_2 = 0$ $10x_1 = -10x_2$ $x_1 = -x_2$ $x = \begin{bmatrix} -1 \\ 1 \end{bmatrix} \sim \hat{e}_2$
--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Untuk $\lambda_1 = 10$ maka $x = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \sim \hat{e}_1$ dan untuk $\lambda_2 = 0$ maka $x = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \sim \hat{e}_2$

Normalisasi *eigenvector*:

$$\bar{e}_1 = \frac{\begin{bmatrix} -1 \\ -1 \end{bmatrix}}{\sqrt{(-1)^2 + (-1)^2}} = \frac{\begin{bmatrix} -1 \\ -1 \end{bmatrix}}{1.414} = [(-0.707), (-0.707)]$$

$$\bar{e}_2 = \frac{\begin{bmatrix} -1 \\ 1 \end{bmatrix}}{\sqrt{(-1)^2 + 1^2}} = \frac{\begin{bmatrix} -1 \\ 1 \end{bmatrix}}{1.414} = [(-0.707), (0.707)]$$

$$\text{Sehingga matriks } U = [\bar{e}_1, \bar{e}_2] = \begin{bmatrix} -0.707 & -0.707 \\ -0.707 & 0.707 \end{bmatrix}$$

4. Menentukan nilai singular S yang merupakan matriks diagonal dari $\sqrt{\lambda}$. S disusun diagonal dengan urutan dari yang terbesar ke yang terkecil.

$$S = \begin{bmatrix} \sqrt{10} & 0 \\ 0 & 0 \end{bmatrix}$$

5. Menentukan inversi matriks.

$$A^{-1} = VS^{-1}U^T$$

$$\text{Dengan } S^{-1} = \frac{1}{\text{elemen } S} = \begin{bmatrix} 0.3162 & 0 \\ 0 & 0 \end{bmatrix}$$

$$\text{Dan } U^T = \begin{bmatrix} 0.707 & 0.707 \\ 0.707 & -0.707 \end{bmatrix}$$

Sehingga

$$A^{-1} = \begin{bmatrix} -0.447 & 0.894 \\ 0.894 & -0.447 \end{bmatrix} \begin{bmatrix} 0.3162 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} -0.707 & -0.707 \\ -0.707 & 0.707 \end{bmatrix}$$

$$A^{-1} = \begin{bmatrix} -0.141 & 0 \\ 0.282 & 0 \end{bmatrix} \begin{bmatrix} -0.707 & -0.707 \\ -0.707 & 0.707 \end{bmatrix}$$

$$A^{-1} = \begin{bmatrix} 0.1 & 0.1 \\ -0.2 & -0.2 \end{bmatrix}$$

Modul yang digunakan pada Python adalah **numpy.linalg.svd**

```

# Singular-value decomposition
import numpy as np
from scipy.linalg import svd

# define a matrix
A = np.array([[1, -2], [1, -2]])

# SVD
U, s, VT = svd(A)

# S nxn diagonal
S = []
for i in range(len(A)):
    b = []
    for j in range(len(A[0])):
        c = 0
        b.append(c)
    S.append(b)
for m in range(len(S)):
    for n in range(len(S[0])):
        if m == n:
            S[m][n] = s[m]

Si = []
for i in range(len(S)):
    b = []
    for j in range(len(S[0])):
        c = 0
        b.append(c)
    Si.append(b)
for i in range(len(S)):
    for j in range(len(S[0])):
        if 0.000001 > S[i][j] >= 0:
            Si[i][j] = 0
        elif 0 >= S[i][j] > (-0.000001):
            Si[i][j] = 0
        else:
            Si[i][j] = 1/S[i][j]

print("\nU:\n", U)
print("\nS:\n", np.array(S))
print("\nVT:\n", VT)
print("\nA:\n", A)
print("\nV:\n", np.transpose(VT))
print("\nSi:\n", np.array(Si))
print("\nUT:\n", np.transpose(U))
Ai = np.transpose(VT) * np.mat(Si) * np.transpose(U)
print("\nAinv:\n", Ai)

```

```

U:
[[ -0.70710678 -0.70710678]
 [ -0.70710678 0.70710678]]

S:
[[ 3.16227766 0.          ]
 [ 0.          0.          ]]

VT:
[[ -0.4472136 0.89442719]
 [ 0.89442719 0.4472136 ]]

A:
[[ 1 -2]
 [ 1 -2]]

V:
[[ -0.4472136 0.89442719]
 [ 0.89442719 0.4472136 ]]

Si:
[[ 0.31622777 0.          ]
 [ 0.          0.          ]]

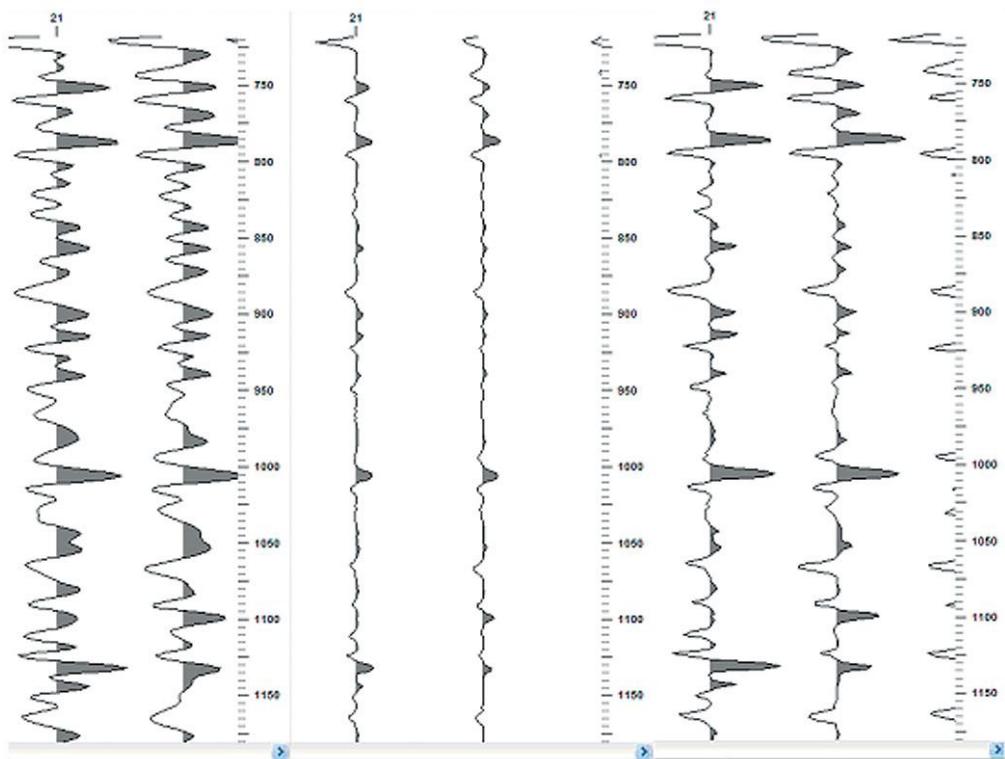
UT:
[[ -0.70710678 -0.70710678]
 [ -0.70710678 0.70710678]]

Ainv:
[[ 0.1 0.1]
 [-0.2 -0.2]]

```

Konversi Data *Binary* ke Format IBM dan IEEE

Data seismik merupakan data *binary* yang disimpan dalam format IBM (*International Business Machines*) dan IEEE (*Institute of Electrical and Electronics Engineers*). Jika salah *input* atau penyimpanan dalam format yang berbeda, maka dapat berakibat pada kalkulasi dan interpretasi.



Gambar 104 - Data seismik dalam format IEEE (kiri), disimpan salah dalam format IBM (tengah), dan setelah gain (kanan)

Sumber: <https://csegrecorder.com/articles/view/segy-floating-point-confusion>

Struktur bilangan 32-bit *binary* dengan format IBM memiliki komposisi sebagai berikut:

1	7	24	width in bits
Sign	Exponent	Fraction	
31	30 – 24	23 – 0	bit index

Cara meng-konversi *binary* 32-bit menjadi desimal dengan format IBM adalah dengan persamaan:

$$V = (-1)^{sign} \times (16)^{e-64} \times fraction$$

Eq. 36

Contoh suatu data *binary* 32-bit memiliki nilai [110000101010001100111110101111](https://www.binaryconvert.com/result_ieee.html?text=110000101010001100111110101111)

Langkah konversi:

- Pengkategorian menjadi $1-1000010-101100011001111110101111$ dengan:
 Sign = 1
 Exponent = 1000010
 Fraction = 101100011001111110101111
 Jika sign = 1, maka angka merupakan angka negatif, namun jika sign = 0 maka angka merupakan angka positif.
- Ubah komponen exponent menjadi suatu angka dengan persamaan konversi *binary* ke desimal:

$$\sum_{i=1}^n (b_i)2^{i-1}$$

Eq. 37

Karena terdapat 7 angka, maka $n = 7$ dengan angka 1000010

1×2^6	0×2^5	0×2^4	0×2^3	0×2^2	1×2^1	0×2^0
64	0	0	0	0	2	0

Maka exponent = 66

- Ubah komponen fraction menjadi suatu angka konversi *binary* ke desimal dengan persamaan:

$$\sum_{i=1}^n (b_i)2^{-i}$$

Eq. 38

101100011001111110101111

1×2^{-1}	0×2^{-2}	1×2^{-3}	1×2^{-4}	0×2^{-5}	0×2^{-6}
0×2^{-7}	1×2^{-8}	1×2^{-9}	0×2^{-10}	0×2^{-11}	1×2^{-12}
1×2^{-13}	1×2^{-14}	1×2^{-15}	1×2^{-16}	1×2^{-17}	0×2^{-18}
1×2^{-19}	0×2^{-20}	1×2^{-21}	1×2^{-22}	1×2^{-23}	1×2^{-24}

Hasilnya adalah: 0.69384282827377319335

- Hasil akhir dari konversi *binary* 32-bit ke format IBM adalah:

$$V = (-1)^{\text{sign}} \times (16)^{e-64} \times \text{fraction} = (-1)^1 \times (16)^{66-64} \times 0.69384$$

$$V = (-1) \times (16)^2 \times 0.69384 = -177.63$$

Struktur bilangan 32-bit *binary* dengan format IEEE memiliki komposisi sebagai berikut:

1	8	23	width in bits
Sign	Exponent	Fraction	
31	30 – 24	23 – 0	bit index

Cara meng-konversi *binary* 32-bit menjadi desimal dengan format IEEE adalah dengan persamaan:

$$V = (-1)^{\text{sign}} \times (1 + \text{fraction}) \times 2^{e-127}$$

Contoh yang sama, suatu data *binary* 32-bit memiliki nilai

1100001010100011001111110101111

Langkah konversi:

- Pengkategorian menjadi 1-10000101-01100011001111110101111 dengan:
Sign = 1
Exponent = 10000101
Fraction = 01100011001111110101111
Jika sign = 1, maka angka merupakan angka negatif, namun jika sign = 0 maka angka merupakan angka positif.
- Ubah komponen exponent menjadi suatu angka dengan persamaan sebelumnya, konversi *binary* ke desimal:

Karena terdapat 8 angka, maka n = 8 dengan angka 1000010

1×2^7	0×2^6	0×2^5	0×2^4	0×2^3	1×2^2	0×2^1	1×2^0
128	0	0	0	0	4	0	1

Maka exponent = 133

- Ubah komponen fraction menjadi suatu angka konversi *binary* ke desimal

01100011001111110101111

0×2^{-1}	1×2^{-2}	1×2^{-3}	0×2^{-4}	0×2^{-5}	0×2^{-6}
1×2^{-7}	1×2^{-8}	0×2^{-9}	0×2^{-10}	1×2^{-11}	1×2^{-12}
1×2^{-13}	1×2^{-14}	1×2^{-15}	1×2^{-16}	0×2^{-17}	1×2^{-18}
0×2^{-19}	1×2^{-20}	1×2^{-21}	1×2^{-22}	1×2^{-23}	

Hasilnya adalah: 0.38768565654754638671

- Hasil akhir dari konversi *binary* 32-bit ke format IEEE adalah:

$$V = (-1)^{\text{sign}} \times (1 + \text{fraction}) \times 2^{e-127} = (-1)^1 \times (1 + 0.38768) \times 2^{133-127}$$

$$V = (-1) \times (1.38768) \times 2^6 = -88.81$$

```

binary = 10100110010011101001001001010111
c = str(binary)
print('Binary = ', binary)

#konversi IBM
sign = int(c[0])
x = 7
y = 0
exp = 0
while x >= 1:
    q = (c[x])
    exp += int(q) * (2 ** y)
    x -= 1
    y += 1
w = 8
r = -1
fr = 0
while w <= 31:
    m = (c[w])
    fr += int(m) * (2**r)
    w += 1
    r -= 1
v = exp - 64
ibm = (-1) ** sign * (16) ** v * (fr)
print('Hasil konversi IBM adalah ', ibm)

#konversi ieee
sign = int(c[0])
x = 8
y = 0
exp = 0
while x >= 1:
    q = (c[x])
    exp += int(q) * (2 ** y)
    x -= 1
    y += 1
w = 9
r = -1
fr = 0
while w <= 31:
    m = (c[w])
    fr += int(m) * (2 ** r)
    w += 1
    r -= 1
z = exp - 127
ieee = (-1) ** sign * (1 + fr) * (2 ** z)
print('Hasil konversi IEEE adalah ', ieee)

```

```

Binary = 10100110010011101001001001010111
Hasil konversi IBM adalah -1.5132347428382578e-32
Hasil konversi IEEE adalah -7.166893468088813e-16

```

Referensi

- Chapra, S.C. 2012. Applied Numerical methods with MATLAB for Engineers and Scientists. McGraw-Hill.
- El-Tokhey, M., Elhabiby, M., Ragheb, A., Shebl, M. Gravity and Density Relationship (Forward Modeling). International Journal of Scientific & Engineering Research, Volume 6, Issue 1, January-2015ISSN 2229-5518, page 1359 -1363.
- Essa, K.S. 2019. A particle swarmoptimization method for interpreting self-potential anomalies. *Journal of Geophysics and Engineering* (2019) 0, 1–15. doi:[10.1093/jge/gxz024](https://doi.org/10.1093/jge/gxz024)
- Fajriani., Srigutomo, W., Pratomo, P.M. 2004. Interpretation of Self-Potential anomalies for investigating fault using the Levenberg-Marquardt method: a study case in Pinggirsari, West Java, Indonesia. Southeast Asian Conference on Geophysics, IOP Conf. Series: Earth and Environmental Science 62 (2017) 012004. doi :[10.1088/1755-1315/62/1/012004](https://doi.org/10.1088/1755-1315/62/1/012004)
- Lowrie, W. 2007. Fundamentals of Geophysics. Cambridge University Press.
- Munir, R. 2016. Algoritma dan Pemrograman Dalam Bahasa Pascal, C, dan C++. Penerbit Informatika.
- Mussett, A.E., & Khan, M.A. 2000. Looking into the Earth: An Introduction to Geological Geophysics. Cambridge University Press.
- Rao, D.A., Babu, H.V.R., Raju, D.C.V., 1985. Inversion of gravity and magnetic anomalies over some bodies of simple geometric shape. *Pageoph* Vol 123 page 239-249.
- Reynolds, J.M. 1997. An Introduction to Applied and Environmental Geophysics. Wiley.
- Telford, W.M., Geldart, L.P., Sheriff, R.E. 1990. Applied Geophysics. Cambridge University Press.
- Yuliananto, Y. & Setyawan, A. 2015. Interpretasi Pola Aliran Fluida Panasbumi Menggunakan Metode *Spontaneous-Potential (Sp)* Dan Suhu Permukaan Dangkal Pada Sistem Panasbumi Paguyangan Kabupaten Brebes. *Youngster Physics Journal ISSN: 2302 - 7371 Vol. 4, No. 1, Januari 2015, Hal 41 – 48.*
- Slide Kuliah Metode Komputasi, Teknik Geofisika Universitas Pertamina, Semester Ganjil 2017/2018 dan 2018/2019.
- Slide Kuliah Pengantar Teknologi Informasi, Ilmu Komputer Universitas Pertamina, Semester Ganjil 2017/2018.
- New Manual of Seismological Observatory Practice (2002), revised version, lectronically published 2009.
- Slide Kuliah Metode Komputasi, Teknik Geofisika Universitas Pertamina, Semester Ganjil 2017/2018 dan 2018/2019.
- Slide Kuliah Pengantar Teknologi Informasi, Ilmu Komputer Universitas Pertamina, Semester Ganjil 2017/2018.

<https://docs.python.org/>

<http://ensiklopediseismik.blogspot.com/>

<https://stackoverflow.com/>

<https://www.youtube.com/user/SocraticaStudios>

http://informatika.stei.itb.ac.id/~rinaldi.munir/PTI/2013-2014/KU1072_EksplorasiDuniaDigital.pdf

BIOGRAFI PENULIS



Iktri Madrinovella merupakan dosen program studi Teknik Geofisika Universitas Pertamina, lulusan Sarjana Teknik Geofisika Institut Teknologi Bandung dan Master Sains Kebumian Institut Teknologi Bandung. Penulis mengawali karirnya sebagai *Junior Seismologist* di PT. Elnusa Tbk sebelum menjadi dosen di Universitas Pertamina. Pada saat ini, penulis menekuni bidang Seismologi gempabumi.



Ida Herawati merupakan dosen program studi Teknik Geofisika Universitas Pertamina, lulusan Sarjana Geofisika Institut Teknologi Bandung, Master Geofisika *Colorado School of Mines* (USA) dan Doktor Geofisika Institut Teknologi Bandung. Penulis mengawali karirnya sebagai *Geophysicist* di Unocal dan Chevron, dan dosen di Universitas Indonesia sebelum menjadi dosen di Universitas Pertamina. Pada saat ini, penulis menekuni bidang seismik eksplorasi seperti *seismic anisotropy*, *seismic interpretation*, *3D velocity modeling*, dll.



Agus Abdullah merupakan dosen program studi Teknik Geofisika Universitas Pertamina, lulusan Sarjana dan Master Teknik Geofisika Institut Teknologi Bandung, dan Doktor di *The Australian National University* (ANU) Australia. Penulis mengawali karirnya sebagai *Geophysicist* di ConocoPhillips, ExxonMobil, dan Kuwait Oil Company sebelum menjadi dosen di Universitas Pertamina. Penulis juga merupakan *founder* Geodwipa *Cloud Computing*. Penulis memiliki spesifikasi di berbagai bidang Geofisika dan komputasi seperti tomografi, inversi, *seismic interpretation*, *seismic visualization*, dll.

Buku Ajar Metode Komputasi Geofisika Menggunakan Python terdiri atas pengenalan bahasa pemrograman Python secara umum mulai dari apa itu bahasa pemrograman, berpikir komputasional, apa itu Python dan bagaimana cara instalasinya, apa saja terminologi dan sistem angka yang digunakan. Kemudian diikuti dengan dasar-dasar penggunaan Python seperti cara membuat *project*, *Hello World*, *number systems*, *data type*, *data structure*, *statement*, *comment*, *operator*, *matrix operation*, *control flow*, *file i/o*, *array generator*, *2D array*, *plotting (2D & 3D)*, *function*, *statistics*, *class*, *lambda operator*, *sorting*, *random numbers*. Sebagian besar dalam buku ini membahas tentang contoh penggunaan Python dalam komputasi Geofisika seperti metode *least-square*, *forward modeling* untuk metode gayaberat dan magnetik, *first-break picking*, *Fast-Fourier Transform (FFT)*, *smoothing 1D* dengan *moving average*, penentuan kecepatan dan ketebalan lapisan gelombang seismik refraksi, interpolasi *spline*, *inverse matrix* dengan *Singular Value Decomposition*, dan konversi data *binary* dari format IBM dan IEEE. Buku ajar ini bersifat bahasa pemrograman dan metode-metode dasar yang dapat dengan mudah dipahami oleh pelajar dan mahasiswa.



Penerbit Universitas Pertamina
Jalan Teuku Nyak Arief Simprug
Kebayoran Lama
Jakarta Selatan 12220
www.universitaspertamina.ac.id

