

MODUL PEMBELAJARAN
PRAKTIKUM PEMROGRAMAN LANJUT
VIK2218



Oleh:
Ratna Lestari Budiani, S.Si., M.EngSc (0027108704)

**DEPARTEMEN TEKNIK ELEKTRO DAN INFORMATIKA
SEKOLAH VOKASI
UNIVERSITAS GADJAH MADA
2021**

HALAMAN PENGESAHAN MODUL PEMBELAJARAN

NAMA MATA KULIAH	:	PRAKTIKUM PEMROGRAMAN LANJUT
KODE MATA KULIAH	:	VIK2218
NAMA PENULIS	:	RATNA LESTARI BUDIANI, S.Si., M.EngSc
NIP/NIDN	:	0027108704
DEPARTEMEN/PRODI	:	DEPARTEMEN TEKNIK ELEKTRO DAN INFORMATIKA / DIV TEKNOLOGI REKAYASA INSTRUMENTASI DAN KONTROL

Mengetahui,
Ketua Program Studi

Yogyakarta, 11 Desember 2021
Penulis,

(Hidayat Nur Isnianto, S.T., M.Eng)
NIP. 197305282002121001

(Ratna Lestari Budiani, S.Si., M.EngSc)
NIKA. 111198710201811202

Mengetahui,
Ketua DepartemenTeknik Elektro dan Informatika,

(Nur Rohman Rosyid, S.T., M.T., D.Eng)
NIP. 111197510201206101

BAB I

PENGENALAN PYTHON

CAPAIAN PEMBELAJARAN (*Learning Out Come*)

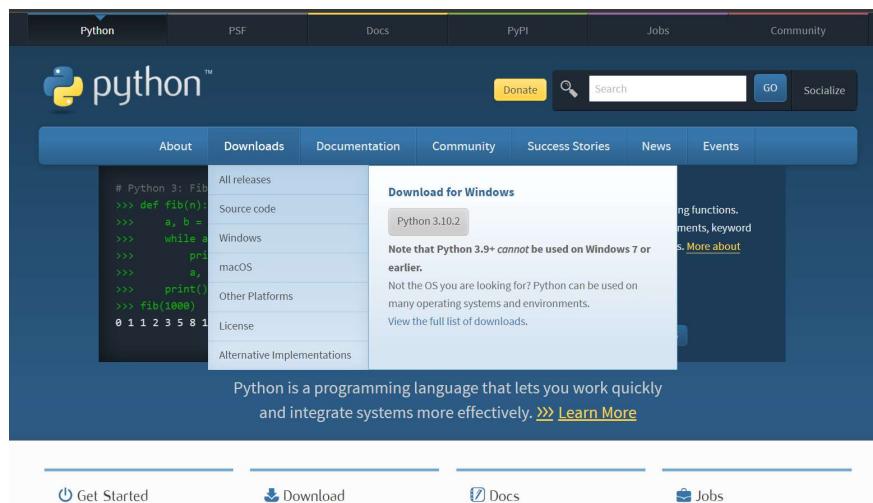
1. Mahasiswa memahami tujuan dari pemrograman
2. Mahasiswa mengenal bahasa pemrograman Python
3. Mahasiswa memahami software yang akan digunakan
4. Mahasiswa dapat membuat program “Hello World!”
5. Mahasiswa memahami special karakter

A. PENDAHULUAN / DESKRIPSI SINGKAT

Python merupakan pemrograman dengan yang berkembang dengan cepat, dan sekarang merupakan pemrograman terbesar. Tidak hanya untuk aplikasi pemrograman dasar, python memiliki segudang *library* yang dapat dimanfaatkan para *developer* untuk membangun sebuah produk. *Library* yang dimiliki dimanfaatkan dan dikelola dengan baik, meskipun bersifat *opensource*. Python dapat digunakan untuk membangun *game*, *website*, sebagai pengolah data atau *data science*, pengolah sinyal, bahkan kecerdasan buatan. Python juga memiliki banyak editor yang dapat dimanfaatkan.

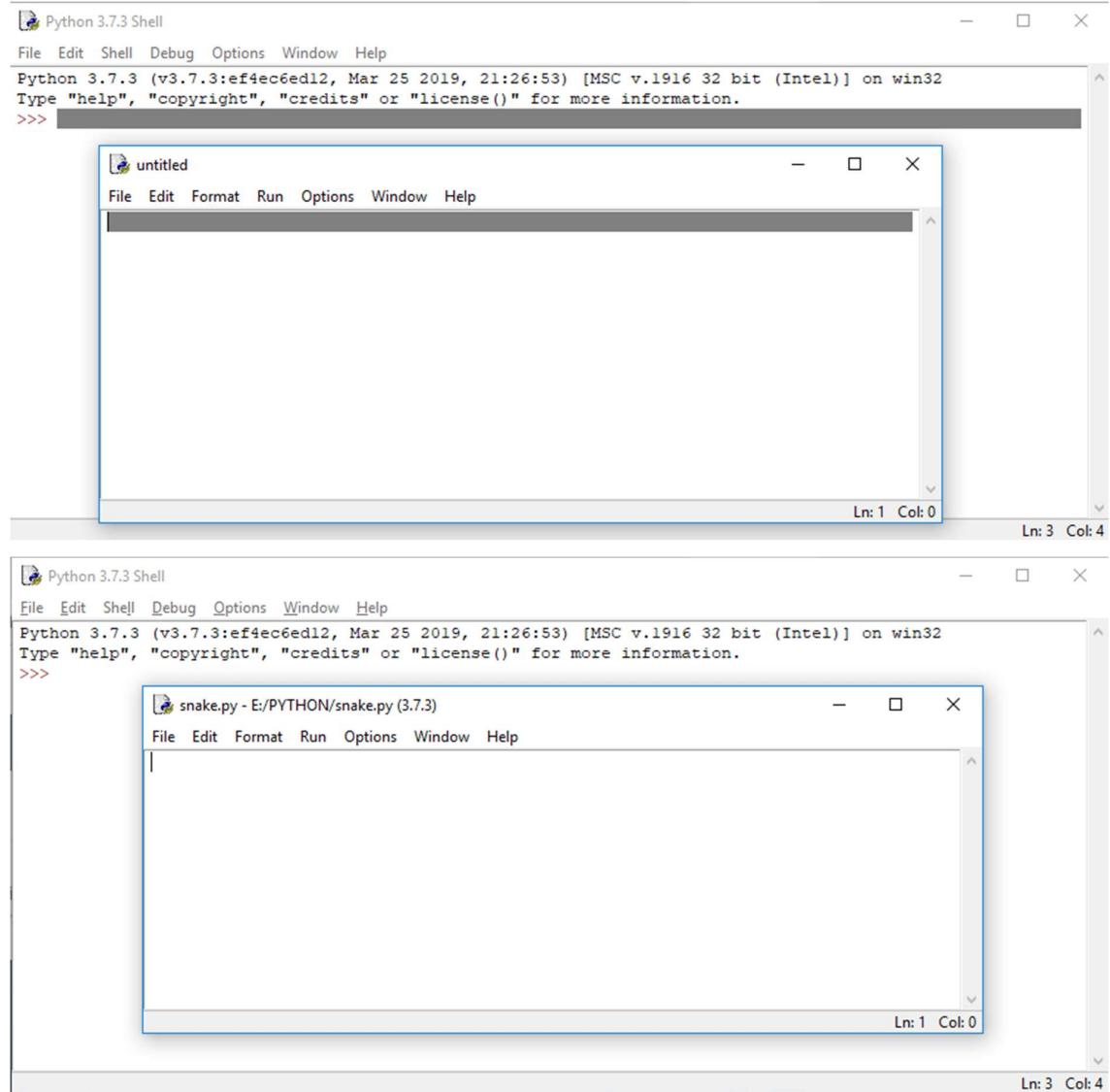
B. EDITOR PYTHON

1. Python IDLE



Python IDLE dapat di download di link berikut dan gratis (<https://www.python.org/>). IDLE merupakan singkatan dari Integrated Development and Learning Environment. IDLE merupakan teks editor bawaan dari Python, ketika kita menginstall python akan terinstall secara otomatis.

Setelah Python pada web telah di download dan diinstall, maka teks editor dapat dibuka pada menu Python IDLE. Klik File>> New File untuk membuat file .py baru. Simpan file dengan nama yang diinginkan dan akan terbentuk dengan format .py. Untuk mengeksekusi program, dapat menekan F5 dan hasil akan terlihat pada command window. Tahapan dapat dilihat pada Gambar dibawah :



Python 3.7.3 Shell

File Edit Shell Debug Options Window Help

Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

>>>

snake.py - E:/PYTHON/snake.py (3.7.3)*

File Edit Format Run Options Window Help

print("Hello World")|

Ln: 1 Col: 20

Python 3.7.3 Shell

File Edit Shell Debug Options Window Help

Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

>>>

===== RESTART: E:/PYTHON/snake.py =====

Hello World

>>>

snake.py - E:/PYTHON/snake.py (3.7.3)

File Edit Format Run Options Window Help

print("Hello World")|

Ln: 1 Col: 20

Python 3.7.3 Shell

File Edit Shell Debug Options Window Help

Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit (Intel)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

>>>

=====

Hello World

>>>

snake.py - E:/PYTHON/snake.py (3.7.3)*

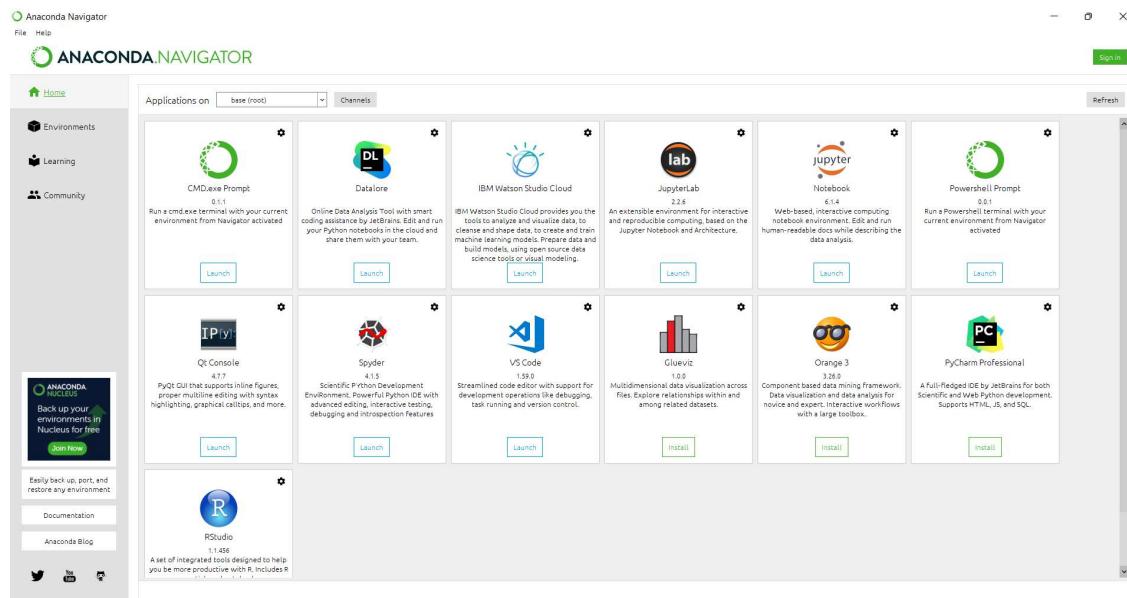
File Edit Format Run Options Window Help

print "Hello World"|

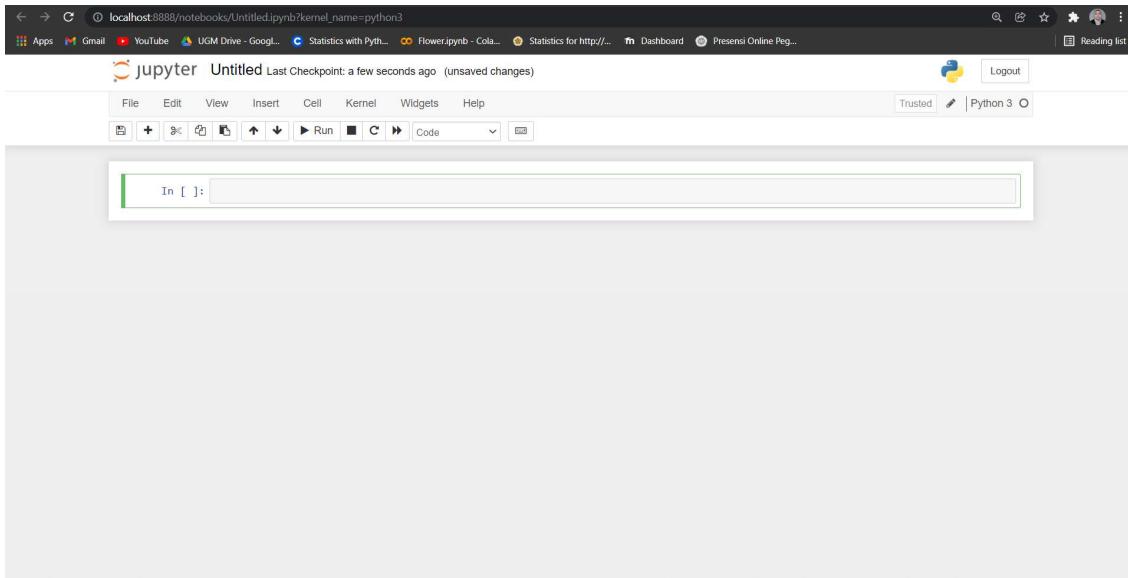
2. Anaconda

Anaconda merupakan paket distribusi dari Python yang sangat powerful. Tidak hanya untuk keperluan pemrograman umum, namun ada beberapa paket tambahan seperti keperluan *data science* maupun matematika yang dirangkum dalam sebuah platform yang *user friendly*. Perlu di catat bahwa Anaconda membutuhkan space yang besar ketika dipasang di perangkat anda. Installer Anaconda dapat di unduh di link ini

<https://docs.anaconda.com/anaconda/install/windows/>



Untuk pemrograman dasar dapat menggunakan Jupyter NoteBook yang akan tersimpan dengan format .ipynb. Ketika Jupyter Notebook di klik maka akan muncul tampilan pada program browser. Klik New>> Python 3 pada tombol ujung kanan atas. Setelah itu akan tampil window sebagaimana dibawah pada browser anda.

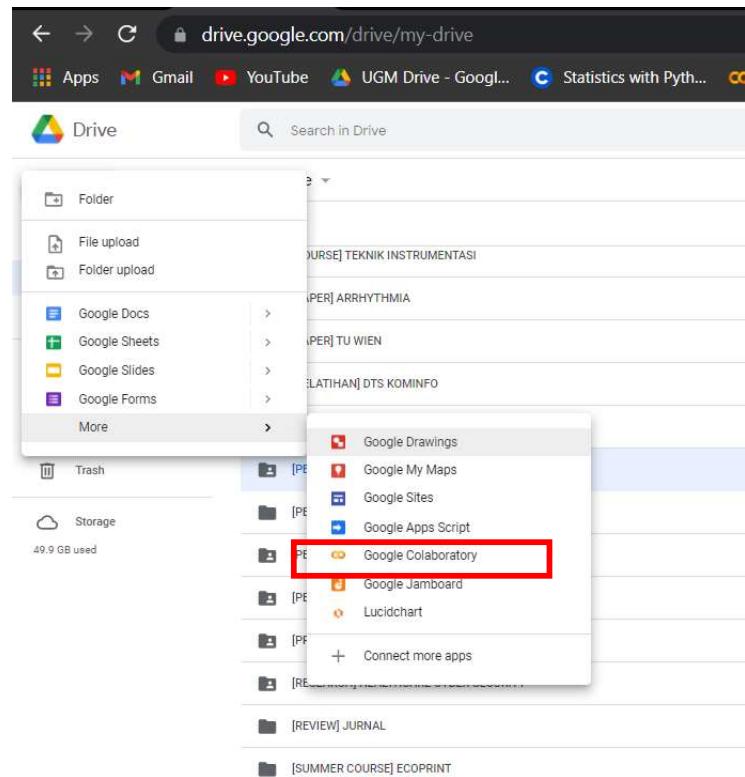


Berbagai kemudahan bisa didapatkan dengan menggunakan Jupyter dan sangat nyaman untuk media pembelajaran. *Code* dan *debugger* ada pada halaman yang bersamaan. Tidak hanya itu, Jupyter Notebook juga dapat digunakan sebagai catatan materi perkuliahan.

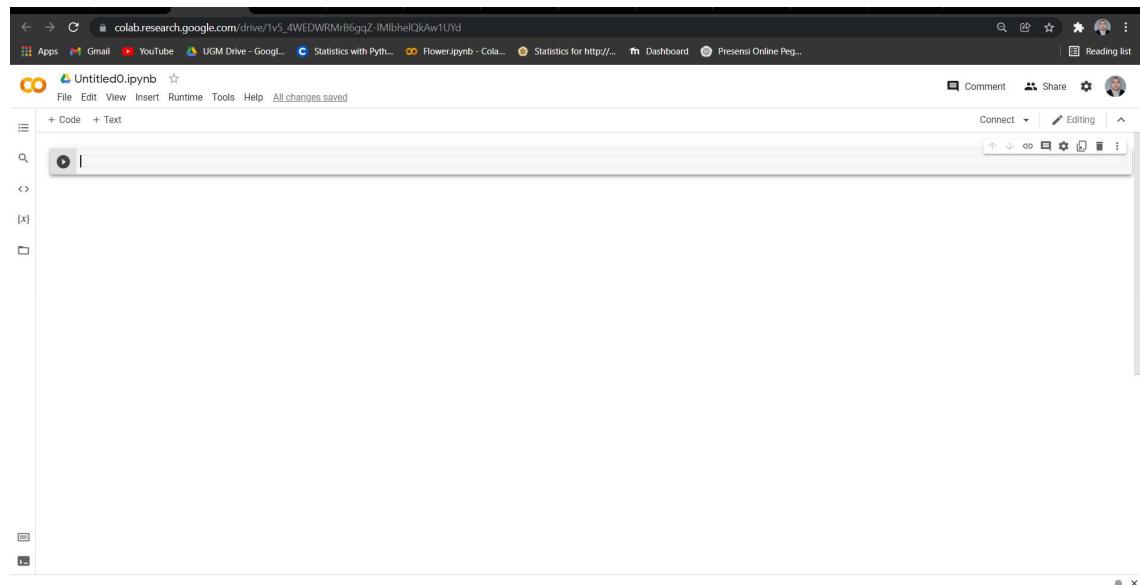
3. Google Colab

Google Colab atau merupakan singkatan dari Google Collaboratory merupakan sebuah *executable document* yang dapat digunakan untuk menyimpan, menulis, serta membagikan program yang telah ditulis. Menggunakan Google Colab akan tersinkronikasi dengan Google Drive anda, sehingga semua program akan tersimpan disana.

Google Colab merupakan aplikasi berbasis *website* yang mana Python terinstall pada *cloud*. Editor ini sangat *powerful* terutama ketika harus mengerjakan program dalam tim. Selain daripada itu pengguna tidak perlu menginstall pada komputer nya. Namun kekurangannya adalah untuk menggunakan program ini harus memanfaatkan internet. Google Colab memiliki *environment* yang serupa dengan Jupyter, dan file yang disimpan juga dalam bentuk .ipynb. Cara mengakses Google Colab adalah pertama-tama dengan membuat folder baru pada gdrive anda. Klik New folder >> More dan Google Colaboratory.

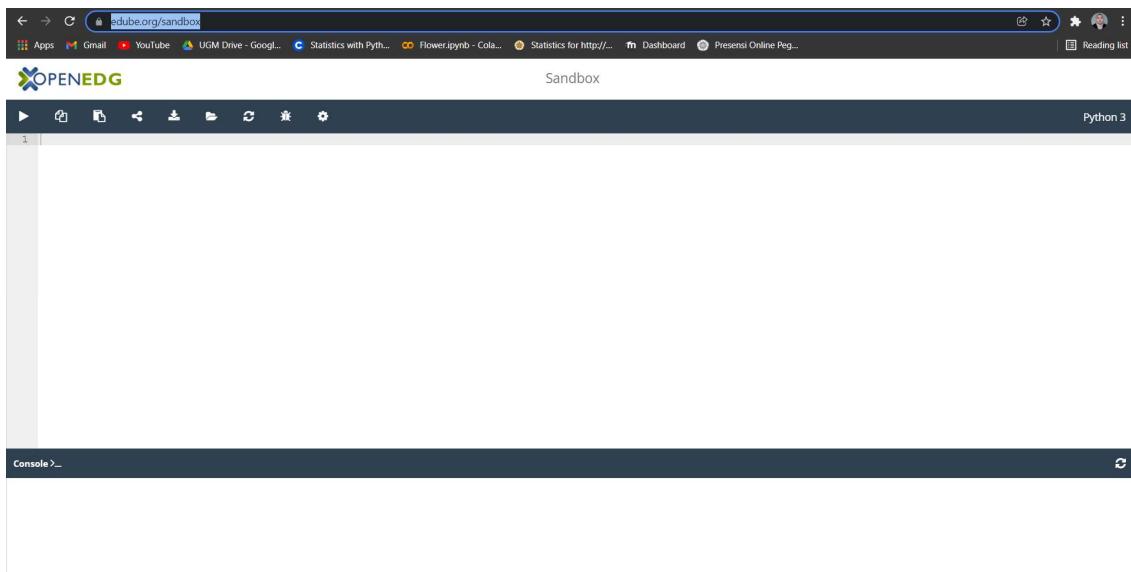


Setelah itu akan tampil windows sebagaimana dibawah, dan anda akan dapat menggunakan Jupyter secara gratis tanpa perlu memakan space komputer anda. Code dapat ditambahkan dengan menekan (+Code) pada ujung kiri atas, dan dapat mer -run dengan menekan tombol *play* pada samping code. kecepatan komputasi akan bergantung pada kecepatan internet yang anda miliki.



4. Edube

Program serupa juga dimiliki oleh Edube yang dapat diakses di <https://edube.org/sandbox>. Namun tidak se *powerful* Google Colab, sandbox hanya dipakai untuk mencoba program dasar. Tampilan akan dapat dilihat sebagaimana dibawah. Kolom atas merupakan tempat menuliskan program, dan kolom *console* untuk melihat hasil dari programnya.



Tidak hanya keempat program diatas yang dapat digunakan sebagai editor Python. Contoh lainnya seperti Sublime Text atau Visual Basic. Gunakan editor sesuai dengan keperluan anda.

C. MEMULAI PEMROGRAMAN

1. Program “Hello World”

Dibawah ini adalah contoh fungsi Python yang digunakan untuk mencetak. Di Python untuk mencetak cukup gunakan fungsi **print()**, dimana sesuatu yang akan dicetak harus diletakkan diantara kurung buka dan kurung tutup, Jika ingin mencetak tipe data String langsung, Anda harus memasukkannya ke dalam tanda kutip terlebih dahulu.

```
print("Hello World")
```

Saat anda menjalankan script diatas, Anda akan melihat output berupa text **Hello World**

Cobalah beberapa baris program dibawah :

```
print("Hello,Python!!")
print("Ratna")
Ratna = 1
print(Ratna)
```

```
print("Ratna")
print(12)
python = "Ratna"
print(python)
```

2. Special character

- `\\` = karakter spesial "escaping nature"
 - `\\n` = untuk enter
 - `\\` = escaping karakter
- print() = berarti line kosong
Tidak boleh ada 2 perintah dalam 1 baris seperti contoh

```
print("nama depan") print ("nama belakang")
```

Cobalah beberapa baris program dibawah :

```
print("Ratna")
print()
print("python")
print("Ratna")print("python")
```

```
print("seketika langit menjadi gelap")
print("kemudian hujan turun dengan deras")
print()
print("seketika langit menjadi gelap \\nkemudian hujan turun dengan deras")
```

```
print("\\")
print("\\")
print("Jum'at")
print('Jum\'at')
```

```
#Dia mengatakan "Halo"
print("Dia mengatakan \'Halo\'")
print('Dia mengatakan "Halo"')
#Katanya "Apa kabar", begitu
print('Katanya "Apa kabar",begitu')
print("Katanya \'Apa kabar\', begitu")
```

D. SOAL LATIHAN/ TUGAS

Pilih salah satu editor yang sesuai dengan kebutuhan anda dan cobalah program “Hello World”. Screen shot hasil dari program “Hello World!”, dan jelaskan keuntungan dan kelemahan dari Editor tersebut.

BAB II

PERCABANGAN PYTHON

CAPAIAN PEMBELAJARAN (*Learning Out Come*)

1. Mahasiswa memahami struktur dari percabangan pada python
2. Mahasiswa memahami struktur dari perulangan pada python

A. STATEMENT IF-ELSE

Cara kerja dari fungsi ini sama saja dengan yang terdapat pada bahasa pemrograman lainnya. Hanya penulisan *syntax* nya cukup berbeda. Perhatikan perbandingan struktur penulisannya pada bahasa C dan Python di bawah ini.

<< Python >>	<< C >>
<pre>if kondisi_1: statement1 elif kondisi_2:</pre>	<pre>if (kondisi_1){ statement1;} else if (kondisi_2){</pre>

berdasarkan perbandingan tersebut, setidaknya ada 3 hal yang wajib dituliskan pada bahasa C yakni tanda kurun, titik koma, dan kurung kurawal yang tidak lagi ditemukan pada Python. Berdasarkan perbandingan tersebut, setidaknya ada 3 hal yang wajib dituliskan pada bahasa C yakni tanda kurun, titik koma, dan kurung kurawal yang tidak lagi ditemukan pada Python.

Perhatikan diatas yang menjelaskan tentang cara kerja fungsi ini untuk mendekripsi genap atau ganjil sebuah bilangan yang merupakan bentuk lain dari contoh dibawah ini.

```
import random  
  
bawah1 = 8; atas1 = 10;  
bawah2 = 8; atas2 = 11;  
bil1 = random.randint(bawah1, atas1)  
bil2 = random.randint(bawah2, atas2)  
print('Bil1:', bil1); print('Bil2:', bil2);  
sisa1 = bil1 % 2; sisa2 = bil2 % 2;  
if (sisa1 != 1) & (sisa2 != 1):  
    print('Bil1 dan Bil2 genap')  
else:  
    print('Bil1 atau Bil2 ganjil')
```

Sementara itu, berikutnya merupakan program untuk *scoring* nilai akhir mahasiswa sebuah program studi di Sekolah Vokasi, Universitas Gadjah Mada. Ketik dan jalankan program tersebut lalu amati hasilnya terkait dengan alur kerja pencabangan.

```
print('***** NILAI MAHASISWA *****')
nama = input('Nama:')
nim = input('NIM:')
nilai = float(input('Nilai: '))
print('*****')

if nilai >= 80:
    skor = 'A'
elif nilai >= 70:
    skor = 'B'
elif nilai >= 60:
    skor = 'C'
elif nilai >= 35:
    skor = 'D'
else:
    skor = 'E'

print('Nilai akhir ' + nim + ' = ' + skor)
```

3. Statemen SWITCH – CASE

Python tidak memiliki statemen SWITCH – CASE. Namun ada alternatif untuk statemen ini di dalam Python yakni dengan menggunakan *dictionary mapping*. Materi ini akan diberikan pada modul “list dan dictionary”.

4. Statemen Pencabangan Bersarang (*nested conditional*)

Secara sederhana dapat dimaknai dengan statemen IF di dalam IF. Dalam banyak masalah pemrograman, *nested IF* sering dipakai. Alur kerjanya sama dengan IF tak bersarang. Perhatikan syntax dibawah ini tentang penggunaan *nested IF*.

```
angka = int(input('Masukkan bilangan bulat bebas = '))
if (angka % 2) == 0:
    GenGan = 'Genap'
    if angka < 0:
        PosorNeg = 'Negatif'
    else:
        PosorNeg = 'Positif'
else:
    GenGan = 'Ganjil'
    if angka < 0:
        PosorNeg = 'Negatif'
    else:
        PosorNeg = 'Positif'
print(angka, 'adalah bilangan', GenGan, PosorNeg)
```

Program diatas berisi program yang mendekripsi apakah bilangan yang dimasukkan oleh *user* berupa bilangan Genap/Ganjil yang bertanda Positif/Negatif. Ketik dan jalankan program tersebut berulang kali agar lebih memahami alur kerjanya sehingga bisa diterapkan untuk persoalan – persoalan lain.

B. SOAL LATIHAN/ TUGAS

1. Buatlah program menggunakan pencabangan IF untuk mengubah suhu/temperature dari dan ke untuk Celcius dan Fahrenheit. Untuk memudahkan anda, gunakanlah angka sebagai penanda pilihan seperti 1-Fahrenheit dan 2-Celcius. Penggunaan string/character untuk logika pencabangan akan diajarkan pada modul perulangan. Masukannya berasal dari *user* saat program dijalankan. Jangan lupa menampilkan satuan yang benar untuk hasilnya. Tampilkan pesan *error* jika ada pilihan konversi suhu bukan antara angka 1 atau 2 dan gunakan pembulatan agar suhu hasil konversi berbilangan bulat.

Note: Anda boleh *browsing* rumus konversi suhu dari dan ke untuk kedua ukuran temperature tersebut.

2. Buatlah sebuah program yang meminta *user* menebak sebuah angka yang digenerate terlebih dahulu oleh *random number generator*. Kesempatan yang diberikan adalah 3 kali. Angka ini berada pada rentang 5 – 8. Jika ada salah satu dari tebakan tersebut benar, maka program akan berhenti dan menyampaikan **pesan bahwa tebakan tepat**. Untuk anda sudah diupload sebuah file program python yang menjadi tempat anda mengerjakan tugas nomor ini, bernama **Tugas2.py**. Anda hanya boleh menambahkan program pada area yang sudah disiapkan dalam program dan tidak boleh mengubah atau menghapus program yang sudah ada. Gunakanlah variabel bernama tebak1, tebak2 dan tebak3 untuk menampung 3 kesempatan tebakan *user*.
3. Buatlah program untuk mengubah nilai detik ke dalam jam, Menit, dan Detik hasil konversi harus bertipe bilangan bulat. Ujilah program anda dengan 3 masukan berikut dan program anda benar jika memberikan hasil yang sama.

Jumlah Detik	Jam, Menit, Detik
3600	1 jam 0 menit 0 detik
400	0 jam 6 menit 40 detik
88888	24 jam 41 menit 28 detik

4. Jika anda diberi tiga buah lidi dengan masing – masing memiliki panjang tertentu, maka tentukanlah apakah mereka bisa membentuk sebuah segitiga atau tidak. Cara yang

paling sederhana adalah dengan menggunakan aturan bahwa **sebuah segitiga hanya dapat terbentuk dari 3 buah sisi dengan ketentuan tidak ada salah satu sisi yang panjangnya melebihi jumlah dari kedua sisi lainnya**. Buatlah program yang menerima masukan dari *user* berupa panjang masing – masing sisi (3 buah sisi) dan menampilkan YA jika kombinasinya dapat membentuk sebuah segitiga, dan TIDAK jika sebaliknya.

5. Buatlah program untuk menentukan hasil dari ujian sebuah mata kuliah! (**SCORE : 10**)

Nilai Akhir (NA) = 70% Nilai UAS + 30% Nilai UTS

Aturan Nilai Indeks :

- Jika $NA \geq 80$, maka “Lulus”
- Jika $80 > NA \geq 70$, maka “Lulus dengan pertimbangan”
- Jika $70 > NA \geq 55$, maka “Lulus dengan tugas”
- Jika $55 > NA \geq 40$, maka “Mengulang”
- Jika $NA < 40$, maka “Gagal”

Input : Nilai UAS dan Nilai UTS

Output : “Lulus” / “Lulus dengan pertimbangan” / “Lulus dengan tugas” /“Mengulang” / “Gagal”

BAB III

PERULANGAN PYTHON

CAPAIAN PEMBELAJARAN (*Learning Out Come*)

1. Mahasiswa memahami logika pemrograman perulangan
2. Mahasiswa memahami penulisan logika perulangan pada Python untuk kasus tertentu

A. PENDAHULUAN / DESKRIPSI SINGKAT

Perulangan atau iterasi adalah bagian pada program yang sangat sering digunakan dalam berbagai penyelesaian masalah. Perulangan ditujukan untuk melakukan operasi atau ekspresi yang sama secara berulang – ulang hingga menemui kondisi penghentian (*stopping condition*). Sama halnya dengan bahasa pemrograman lain, *built-in function* yang digunakan adalah FOR dan WHILE. Perbedaan adalah dalam penulisannya. Biasanya perulangan atau iterasi juga disebut dengan LOOPS.

B. POKOK-POKOK ISI

1. Strings

Dalam bahasa pemrograman C, string dikenal sebagai tipe data yang menampung lebih dari satu karakter. Definisi yang sama juga dapat diberlakukan pada Python. Di dalam modul ini, salah satu fungsi yang digunakan adalah *len* yang akan mengembalikan jumlah karakter dari sebuah *string* ketika digunakan. Perhatikan contoh 3.5 untuk penjelasannya.

2. Perulangan FOR

Perulangan ini memiliki struktur penulisan sebagai berikut

```
for variabel in sebuah_runtun:  
    ekspresi/operasi
```

```
range (start, stop, step)
```

Dalam perulangan FOR, inkrement terjadi secara otomatis. Biasanya statemen pada *sebuah_runtun* diganti dengan *built-in function* bernama Range. Range memiliki bentuk penulisan sebagai berikut

Misalnya, untuk `range(0, 5, 2)` maka hasilnya adalah 0, 2, 4. Jika nilai *step* tidak diberikan maka secara *default* bernilai 1. Perhatikan syntax dibawah ini yang menggambarkan cara kerja perulangan FOR.

```
for i in range(0, 5, 2):
    print (i)
```

Ketika program tersebut dijalankan, maka menghasilkan

```
0
2
4
```

Yang menjelaskan bahwa perulangan dimulai dari angka 0 menuju maksimal 5 dengan nilai langkah (inkremen/*step*) 2. Nilai maksimum yang memungkinkan hanya 4 karena jika perulangan dilanjutkan maka hasil akhirnya adalah 6. Tentu saja hal itu tidak mematuhi syarat perulangan yang sudah ditetapkan dalam fungsi **range**.

Ada beberapa bentuk lain dalam penulisan **range** yang menunjukkan fleksibilitas fungsi ini. Terkadang, nilai yang dituliskan dalam **range** hanya nilai maksimalnya saja. Dengan demikian nilai awal otomatis 0 dan inkremennya 1. Perhatikan syntax dibawah ini yang menggambarkan proses tersebut. Ketiklah program tersebut dan jalankan. Lalu amati dengan seksama hasil dari program.

```
for i in range(0, 5):
    print (i)

for i in range(5):
    print (i)
```

Perulangan ini juga dapat digunakan untuk menghitung penjumlahan sederet angka seperti pada syntax dibawah. Ketik dan jalankan program pada contoh tersebut untuk mendapatkan hasil akhir dari penjumlahan bilangan 1 sampai 100 yang akan sulit dilakukan secara manual.

```
jumlah = 0
for i in range(1, 101):
    jumlah += i

print ("Jumlah bilangan 0 - 100 = ", jumlah)
```

3. Indeks Pada Array

Merupakan posisi dari tiap elemen dalam sebuah array. Secara sederhana, array bisa diartikan sebagai runtun bilangan atau karakter. Namun, jika array sudah dalam 2 dimensi, maka disebut sebagai matriks. Dengan adanya indeks pada array, akan memudahkan mengakses isi dari array tersebut. Biasanya indeks array dituliskan setelah nama variabel di dalam kurung siku “[indeks]”. Perlu diingat bahwa indeks selalu bernilai integer. Demikian juga dengan nilai – nilai pada **range** selalu bernilai integer.

Perhatikan syntax dibawah yang meminta *string* diberikan oleh *user* secara interaktif lalu ditampilkan dalam tunda (*delay*) waktu tertentu.

```
import time
nama = input('Tuliskan sebuah nama: ')
for i in range(len(nama)):
    print (nama[i])
    time.sleep(1)
```

Pada contoh tersebut, sebuah modul atau *package* dipanggil yang bernama *time*. Modul ini mengandung sebuah fungsi bernama *sleep* yang menciptakan *delay* sebesar *t* detik. Ketik dan jalankan program tersebut agar anda mendapatkan gambaran lengkap tentang setiap bagian program.

4. Perulangan WHILE

Perulangan ini serupa dengan perulangan FOR dalam hal fungsi dan cara kerja. Hanya, untuk membuat inkremen, statemen dituliskan di dalam tubuh, bukan bersamaan dengan penulisan WHILE. Struktur penulisannya sebagai berikut

```
while prasyarat:
    statemen/ekspresi/operasi
    inkremen
```

Prasyarat bisa berupa perbandingan nilai variabel terhadap bilangan bulat maupun desimal. Untuk mengakses elemen – elemen dari array, nilai indeks yang berasal dari inkremen harus bernilai bulat atau integer. Perhatikan contoh 3.6 yang menggambarkan secara sederhana cara kerja perulangan ini. Hasil dari menjalankan program tersebut adalah

```
i = 0
while i <= 5:
    print(i)
    i = i + 1
```

```
0
1
2
3
4
5
```

Statemen inkremen dalam perulangan ini bisa digantikan dalam bentuk prasyarat saja sehingga tidak perlu menuliskan perhitungan inkremen dalam tubuh perulangan. Perhatikan syntax berikutnya tentang bagaimana mendapatkan pendekatan akar dari 26 menggunakan metode *Newton-Raphson* dengan prasyarat sebagai pengganti inkremen. Ujilah hasil dari program tersebut secara manual menggunakan kalkulator.

```
epsilon = 0.01 # target error
bil = 26.0
kira2 = bil / 2.0
while abs(kira2 * kira2 - bil) >= epsilon:
    kira2 = kira2 - (((kira2 ** 2) - bil) / (2 * kira2))
print('Akar dari ', bil, ' adalah sekitar ', kira2)
```

5. Perulangan Bersarang (*Nested Loops*)

Perulangan juga bisa digunakan secara berlipat – lipat atau biasa disebut dengan bersarang (*nested*). Perulangan bersarang dapat diaplikasikan pada kombinasi FOR – FOR, WHILE – WHILE, FOR – WHILE atau WHILE – FOR. Bahkan lipatannya bisa lebih dari 2. Untuk array berbentuk matriks, lipatan perulangan ini sebanyak 2 kali.

Perlu diingat bahwa fungsi **print** pada Python secara *default* mengandung komponen *new line* (\n). Hal ini sangat merugikan ketika kita berhadapan dengan matriks, karena peletakan elemen – elemennya berbentuk persegi panjang. Agar dapat menampilkan elemen matriks yang layak, maka di bagian akhir dari tubuh **print** diberikan penanda yang tertulis sebagai *end* menjadi:

```
print ('statemen', end = "");
defaultnya, isi dari END = '\n' yang artinya menyediakan new-line
setiap kali print dipanggil.
```

Perhatikan syntax dibawah yang menampilkan karakter '*' dalam bentuk garis dengan sudut 45° terhadap sumbu – x positif. Dalam program ini, akan ditemukan penggunaan perulangan FOR bersarang yang disertai dengan kondisional IF.

```
N = 6
M = 6
for i in range (N):
    for j in range (M, 0, -1):
        if i == j:
            print ('*', end = "")
        else:
            print(' ', end = "")
print("")
```

Jika program tersebut dijalankan, maka akan dihasilkan

```
*
*
*
*
*
```

Ketik dan jalankanlah program dibawah dan perhatikan dengan seksama hasilnya beserta logika program tersebut.

```
N = 6
M = 6
for i in range (N):
    for j in range (M):
        if i == j:
            print ('*', end = "")
        else:
            print(' ', end = "")
```

C. SOAL LATIHAN/ TUGAS

- Buatlah sebuah program yang keluarannya ditunjukkan oleh gambar dibawah

```
*
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * *
* * * * *
* * * *
* *
*
```

2. Buatlah sebuah program untuk menampilkan secara terbalik (*reversed*) kembali karakter demi karakter tulisan: “*Sekolah Vokasi UGM*” yang bertipe data strings dengan menggunakan FOR/WHILE.
3. Program di bawah ini adalah program untuk LOG IN ke dalam akun email dengan rincian sebagai berikut:
 - Program meminta user memasukkan username berupa NIM masing-masing praktikan
 - Program meminta user memasukkan password yakni *abc1234*
 - Tuliskan pesan **LOG IN BERHASIL** lalu program selesai jika username dan password benar dan jika salah satu salah akan menampilkan **GAGAL, COBA LAGI**
 - Jika user salah dalam memasukkan baik username maupun password, sistem akan *looping* meminta user memasukkan identitas yang benar kecuali user memilih pilihan NO untuk membatalkan proses LOG IN.

Namun program saat ini ketika dijalankan tidak memberikan hasil output yang tepat. Carilah letak kesalahannya lalu perbaiki.

Soal nomor 3

```
while True:  
    # memasukkan username dan password  
    print('***** LOG IN *****')  
    usn = input('Username: ')  
    psw = float(input('Password: '))  
    print('*****')  
  
    # deteksi sukses atau gagal  
    if (usn == 27396) & (psw == '888888'):  
        print('LOG IN BERHASIL')  
        print('SELAMAT BERAKTIFITAS')  
        break  
    else:  
        print('GAGAL, COBA LAGI')  
        # konfirmasi ulang log in atau tidak
```

Cat: Selain angka, karakter/string juga bisa digunakan sebagai syarat pencabangan (IF-ELSE).

4. Buatlah sebuah program untuk menjumlahkan dua buah matriks berbeda (A dan B) berukuran 2×2 dengan terlebih dahulu memasukkan elemen – elemen dari matriks – matriks tersebut. Ikutilah aturan penjumlahan matriks yang berlaku umum. Sebagai bantuan, hasil dari program anda akan menyerupai Gambar 3.3.

```
Tuliskan elemen - elemen matriks A
A[ 0 , 0 ] = 1
A[ 0 , 1 ] = 1
A[ 1 , 0 ] = 1
A[ 1 , 1 ] = 1
Tuliskan elemen - elemen matriks B
B[ 0 , 0 ] = 1
B[ 0 , 1 ] = 1
B[ 1 , 0 ] = 1
B[ 1 , 1 ] = 1
C = A + B
2 2
2 2
```

BAB IV

LIST DAN ARRAY

CAPAIAN PEMBELAJARAN (*Learning Out Come*)

1. Mahasiswa memahami bentuk list
2. Mahasiswa memahami sifat dan bentuk modifikasi pada list

A. PENDAHULUAN / DESKRIPSI SINGKAT

Dalam bahasa pemrograman Python, struktur data yang paling dasar adalah urutan atau lists. Setiap elemen-elemen berurutan akan diberi nomor posisi atau indeksnya. Indeks pertama dalam list adalah nol, indeks kedua adalah satu dan seterusnya.

Bentuk umum untuk membuat list dalam python adalah :

```
nama_list = [nilai1, nilai2, nilai3, ...]
```

List dapat diisi oleh beberapa tipe data yang berbeda, tidak harus sama, seperti contoh berikut :

```
var=[10, 5, 7, 2, 1]  
mylist=[1,"Ratna", 4.2, "Digital Talent"]
```

Ada beberapa hal yang dapat Anda lakukan terhadap data didalam list. Operasi ini meliputi pengindeksan, pengiris, penambahan, perbanyak, dan pengecekan keanggotaan. Selain itu, Python memiliki fungsi built-in untuk menemukan panjang list dan untuk menemukan elemen terbesar dan terkecilnya.

B. POKOK-POKOK ISI

1. Pembuatan List

Membuat list sangat sederhana, tinggal memasukkan berbagai nilai yang dipisahkan koma di antara tanda kurung siku. Dibawah ini adalah contoh sederhana pembuatan list dalam bahasa Python.

```
#Contoh sederhana pembuatan list pada bahasa pemrograman python
list1 = ['kimia', 'fisika', 1993, 2017]
list2 = [1, 2, 3, 4, 5]
list3 = ["a", "b", "c", "d"]
```

2. Menambah Elemen dalam List

Terdapat tiga cara untuk menambahkan elemen baru kedalam sebuah list yaitu :

- Menggunakan metode **append()** : Metode ini digunakan untuk menambah elemen tunggal dibagian akhir list. Perhatikan contoh dibawah :

```
bah = ["durian", "mangga", "apel"]
buah.append("jeruk")
print (buah)
```

Append juga dapat digunakan didalam perulangan untuk mengisi sebuah list kosong :

```
myList = [] # creating an empty list

for i in range(10):
    myList.append(i + 2)

print(myList)
```

- Menggunakan metode **insert()** : Metode ini digunakan untuk menambah elemen tunggal di indeks / posisi tertentu. Perhatikan contoh dibawah :

```
bah = ["durian", "mangga", "apel","melon","anggur"]
buah.insert(2,"kiwi")
print (buah)
```

Perhatikan contoh dibawah ini untuk mengetahui perbedaan penggunaan **insert()** dan **append()** :

```
numbers = [111, 7, 2, 1]
print(len(numbers))
print(numbers)

###
```

```
numbers.append(4)
print(len(numbers))
print(numbers)

###

numbers.insert(0, 222)
print(len(numbers))
print(numbers)
```

- Menggunakan metode **extend()**. Metode ini digunakan untuk menyambung atau menggabungkan suatu list dengan list yang lain. Perhatikan contoh berikut :

```
bah = ["durian", "mangga", "apel"]
b=["jeruk","melon"]
buah.extend(b)
print(bah)
```

3. Menghapus Elemen dari dalam List

Menghapus elemen dapat dilakukan dengan :

```
nama_list.remove(nilai)
```

atau dengan memberikan indeksnya menggunakan instruksi delete :

```
del nama_list[indeks]
```

Perhatikan contoh berikut untuk mengetahui perbedaan penggunaannya :

```
angka = [5,4,3,2,1]

del angka[1]
print("Panjang angka:", len(angka))
print("Isi:", angka)
```

4. List dan Variabel

List merupakan sebuah variable yang berisi beberapa data, sehingga sifatnya akan sama dengan variabel pada umumnya, termasuk bagaimana cara membuat nama variabel. Setiap

variabel nantinya akan menempati sebuah lokasi memori sesuai dengan nama variabelnya, namun berbeda dengan list, nama list adalah nama lokasi memori dimana list tersebut disimpan, sehingga ketika list2 = list1 akan menduplikasi nama dari list namun bukan isinya. Sebagai efeknya, dua list tersebut akan diidentifikasi pada lokasi memori yang sama. Jika salah satu di update, maka list lainnya akan terpengaruh (*vice versa*).

Untuk lebih memahaminya, dapat dicoba dengan syntax berikut :

```
list1 = [1]
list2 = list1
list1[0] = 2
print(list2)
print(list1)
```

5. Operasi Dasar

List Python merespons operator + dan * seperti string; Itu artinya penggabungan dan pengulangan di sini juga berlaku, kecuali hasilnya adalah list baru, bukan sebuah String. Sebenarnya, list merespons semua operasi urutan umum yang kami gunakan pada String di bab sebelumnya. Dibawah ini adalah tabel daftar operasi dasar pada list python.

Python Expression	Hasil	Penjelasan
len([1, 2, 3, 4])	4	Length
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Halo!'] * 4	['Halo!', 'Halo!', 'Halo!', 'Halo!']	Repetition
2 in [1, 2, 3]	True	Membership
for x in [1,2,3] : print (x,end = '')	1 2 3	Iteration

6. Indexing, Slicing dan Operato In/Not In

a. Indexing

Karena list adalah urutan, pengindeksan dan pengiris bekerja dengan cara yang sama untuk list seperti yang mereka lakukan untuk [String](#). Perhatikan contoh berikut :

```
L = ['C++', 'Java', 'Python']
```

Python Expression	Hasil	Penjelasan
L[2]	'Python'	Offset mulai dari nol
L[-2]	'Java'	Negatif: hitung dari kanan
L[1:]	['Java', 'Python']	Slicing mengambil bagian

b. Slicing

Pada umumnya slice ditulis dalam bentuk berikut :

```
myList[start:end]
```

Perhatikan contoh berikut :

```
myList = [10, 8, 6, 4, 2]
newList = myList[1:3]
print(newList)
```

Contoh dibawah ini adalah bagaimana perbedaan mencopy seluruh isi List dan mengambil sebagian saja :

```
# Mengambil seluruh isi list
list1 = [1]
list2 = list1[:]
list1[0] = 2
print(list2)

# Mengambil Sebagian isi list
myList = [10, 8, 6, 4, 2]
newList = myList[1:3]
print(newList)
```

Mengambil isi keseluruhan list juga dapat dilakukan mulai dari index tertentu. Perhatikan contoh berikut :

```
myList = [10, 8, 6, 4, 2]
```

```
newList = myList[3:]
print(newList)
```

Slicing juga bisa dilakukan dengan index negatif. Index negatif dihitung dari list sebelah kanan.

Perhatikan contoh berikut :

```
myList = [10, 8, 6, 4, 2]
newList = myList[1:-1]
print(newList)
```

```
myList = [10, 8, 6, 4, 2]
newList = myList[-1:1]
print(newList)
```

Index negative juga dapat digunakan untuk mengambil keseluruhan isi list. Perhatikan contoh berikut :

```
myList = [10, 8, 6, 4, 2]
newList = myList[::-1]
print(newList)
```

Slicing juga dapat menggunakan function del() untuk menghapus Sebagian isi dari list :

```
#menghapus list pada range tertentu

myList = [10, 8, 6, 4, 2]
del myList[1:3]
print(myList)
```

Instruksi del() dapat menghapus semua isi dari list :

```
# menghapus keseluruhan list

myList = [10, 8, 6, 4, 2]
del myList[:]
print(myList)
```

c. In dan Not In

Pyhton memiliki dua operator yang mampu memeriksa apakah list memiliki sebuah nilai tertentu atau tidak. Operator pertama adalah (**in**) yang memeriksa apakah elemen tersebut

tersimpan dalam list atau tidak. Operator yang kedua adalah (**not in**) yang memeriksa apakah elemen tersebut tidak tersimpan dalam list atau tidak. Kedua operator ini akan mengembalikan nilai Boolean (True/False). Perhatikan contoh berikut :

```
myList = [0, 3, 12, 8, 2]

print(5 in myList)
print(5 not in myList)
print(12 in myList)
```

7. Aplikasi List

Pemrosesan list juga dapat menggunakan percabangan maupun perulangan. Perhatikan contoh berikut :

```
myList = [17, 3, 11, 5, 1, 9, 7, 15, 13]
largest = myList[3]

for i in range(0, len(myList)):
    if myList[i] > largest:
        largest = myList[i]

print(largest)
```

```
myList = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
toFind = 5
found = False

for i in range(len(myList)):
    found = myList[i] == toFind
    if found:
        break

if found:
    print("Element found at index", i)
else:
    print("absent")
```

8. Method dan Fungsi Build-in pada List Python

Python menyertakan fungsi built-in sebagaimana terlihat pada Tabel 1 berikut :

Python Function	Penjelasan
cmp(list1, list2)	# Tidak lagi tersedia dengan Python 3
len(list)	Memberikan total panjang list.
Python Function	Penjelasan
max(list)	Mengembalikan item dari list dengan nilai maks.
min(list)	Mengembalikan item dari list dengan nilai min.
list(seq)	Mengubah tuple menjadi list.

Python menyertakan methods built-in sebagai berikut

Python Method	Penjelasan
list.append(obj)	Menambahkan objek obj ke list
list.count(obj)	Jumlah pengembalian berapa kali obj terjadi dalam list
list.extend(seq)	Tambahkan isi seq ke list
list.index(obj)	Mengembalikan indeks terendah dalam list yang muncul obj
list.insert(index, obj)	Sisipkan objek obj ke dalam list di indeks offset
list.pop(obj = list[-1])	Menghapus dan mengembalikan objek atau obj terakhir dari list
list.remove(obj)	Removes object obj from list
list.reverse()	Membalik list objek di tempat
list.sort([func])	Urutkan objek list, gunakan compare func jika diberikan

BAB IV

PERULANGAN *ADVANCE* (LIST)

CAPAIAN PEMBELAJARAN (*Learning Out Come*)

1. Mahasiswa memahami aplikasi penggunaan perulangan pada data array

A. MATRIKS

Pada contoh – contoh program sebelumnya, telah ditunjukkan bagaimana perulangan baik FOR maupun WHILE bisa digunakan membentuk matriks atau array 2 dimensi (2D). pada contoh 3.11, program ini akan menampilkan sebuah matriks yang nilai elemennya dimasukkan secara interaktif oleh *user*. Kemudian ditampilkan ulang dalam bentuk matriks utuh.

```
N = 2
M = 2
print('Tuliskan elemen - elemen matriks')
# menerima masukan secara interaktif dari user
A = []
for i in range (N):
    d = []
    for j in range (M):
        print('A[',i,',',j,']= ', end = "")
        d.append(int(input()))
    A.append(d)

# menampilkan elemen array dalam bentuk matriks utuh
print('Matriks A = ')
for i in range (N):
    for j in range (M):
        print(A[i][j], end = " ")
    print()
```

Pada contoh program di atas ada beberapa *built-in function* baru dikenalkan dalam modul ini yakni **append**. Kegunaan dari fungsi ini adalah untuk menempatkan data baru tepat berdampingan dengan data lama atau yang lebih dulu dimasukkan.

B. LIST ADVANCE

Pada beberapa kondisi list dapat dimanfaatkan untuk mengisi data dari sebuah perulangan. Perhatikan contoh dibawah :

```
myList = [10,1,8,3,5]
total = 0

for i in myList :
    total +=i
print(total)
```

List yang ada pada myList akan di komputasi dan ditampung kembali ke list yang sama.

Perhatikan contoh lainnya berikut ini :

```
squares = []
for i in range(5):
    squares.append(i**2)
print(squares)
```

Untuk melakukan perulangan pada list, dapat dipersingkat menjadi syntax berikut ini :

```
squares2 = [i ** 2 for i in range(5)]
print (squares2)
```

Kedua program akan menghasilkan hasil yang sama. Perhatikan lagi contoh berikut :

```
#mengambil nilai ganjil dari squares
square = [0, 1, 4, 9, 16]

oddSquares =[]
for i in square :
    if i%2 !=0:
        oddSquares.append(i)
print(oddSquares)
```

Program diatas dapat diringkas menjadi program dibawah :

```
square = [0, 1, 4, 9, 16]
oddSquare = [i for i in square if i%2 !=0]
print(oddSquare)
```

C. LIST DUA DIMENSI

List dapat berisi berbagai tipe data seperti strings, boolean maupun list lainnya. Kita sering sekali menemukan bentuk array seperti ini dikehidupan kita, salah satu contohnya adalah papan catur. Papan catur terdiri dari baris dan kolom, terdapat 8 baris dan juga 8 kolom. Setiap kolom ditandai oleh huruf A sampai dengan H, dan setiap baris ditandai oleh 1 sampai dengan 8.

Perhatikan kode berikut :

```
EMPTY = "-"
board = [[EMPTY for i in range (8)]for i in range (8)]
print(board)
```

List diatas menggambarkan list papan catur yang masih kosong sebagaimana gambar ilustrasi dibawah :

	A	B	C	D	E	F	G	H	
8	[0] [0]	[0] [1]	[0] [2]	[0] [3]	[0] [4]	[0] [5]	[0] [6]	[0] [7]	8
7	[1] [0]	[1] [1]	[1] [2]	[1] [3]	[1] [4]	[1] [5]	[1] [6]	[1] [7]	7
6	[2] [0]	[2] [1]	[2] [2]	[2] [3]	[2] [4]	[2] [5]	[2] [6]	[2] [7]	6
5	[3] [0]	[3] [1]	[3] [2]	[3] [3]	[3] [4]	[3] [5]	[3] [6]	[3] [7]	5
4	[4] [0]	[4] [1]	[4] [2]	[4] [3]	[4] [4]	[4] [5]	[4] [6]	[4] [7]	4
3	[5] [0]	[5] [1]	[5] [2]	[5] [3]	[5] [4]	[5] [5]	[5] [6]	[5] [7]	3
2	[6] [0]	[6] [1]	[6] [2]	[6] [3]	[6] [4]	[6] [5]	[6] [6]	[6] [7]	2
1	[7] [0]	[7] [1]	[7] [2]	[7] [3]	[7] [4]	[7] [5]	[7] [6]	[7] [7]	1
	A	B	C	D	E	F	G	H	

Kita bisa mulai mengisi papan catur dengan beberapa pion dengan contoh program dibawah :

```
EMPTY = "_"
ROOK = "ROOK"
KNIGHT = "KNIGHT"
PAWN = "PAWN"
board = []
for i in range(8):
    row = [EMPTY for i in range(8)]
    board.append(row)
board[0][0] = ROOK
board[0][7] = ROOK
board[7][0] = ROOK
board[7][7] = ROOK

#isi dengan KNIGHT
board[4][2] = KNIGHT

#isi dengan PAWN
board[3][4] = PAWN

print(board)
```

Tugas : Lanjutkan dengan pion lainnya!

D. BREAK DAN CONTINUE

Prasyarat pada perulangan juga bisa berupa kelas *boolean*. Kendalanya adalah untuk penghentian perulangan tersebut. Fungsi *break* dan *continue* pada Python dapat digunakan untuk mengendalikan jalannya perulangan baik FOR maupun WHILE.

Fungsi *break* jika dipanggil, akan menghentikan perulangan terdalam (*inner loop*) yaitu perulangan tempat fungsi ini diletakkan. Sementara fungsi *continue* akan melewatkkan proses pada iterasi saat ini lalu dilanjutkan dengan iterasi berikutnya. Perhatikan contoh dibawah ini yang menggambarkan cara kerja *break* dan *continue*.

Ketik dan jalankan kedua program tersebut lalu bandingkan hasilnya untuk mendapatkan gambaran lengkap tentang perbedaan kedua fungsi tersebut.

```
kata = 'Vokasi'  
for huruf in kata:  
    if huruf == 'a':  
        break  
    print('Huruf: ', huruf)
```

```
kata = 'Vokasi'  
for huruf in kata:  
    if huruf == 'a':  
        continue  
    print('Huruf: ', huruf)
```

E. TUGAS

Sebuah penginapan akan dibuka pada sebuah area pariwisata baru. Penginapan tersebut terdiri dari tiga gedung (1,2,3) yang masing-masing terdiri dari 15 lantai (lantai 1-15). Pada setiap lantai, terdiri dari 20 kamar hotel (kamar 1 -20). Buatlah sebuah program untuk memasukkan booking kamar (gedung, lantai dan nomer kamar yang diinginkan) dan akan memberikan keluaran harga kamar yang perlu dibayar.

List yang semula dalam keadaan available atau “-“ akan berganti menjadi “+” Ketika kamar telah dibooking. Spesifikasi harga dari kamar hotel adalah sebagai berikut :

- a. President suite (jumlah 15 kamar) : Rp. 1.000.000/ malam. Merupakan kamar tengah (nomer 8,9,10,11,12) dilantai paling atas dari setiap gedung
- b. Superior suite (jumlah 60 kamar) : Rp. 850.000/malam. Merupakan seluruh kamar dilantai 14 pada tiap gedung
- c. Deluxe suite (jumlah 12 kamar) : Rp. 700.000 / malam. Merupakan dua kamar paling ujung kanan dan kiri dari lantai teratas dari setiap gedung.
- d. Standard room : Rp. 500.000 / malam. Merupakan kamar selain dari 87 kamar diatas.

Lakukan poin berikut :

- a. Tampilkan list yang telah dibuat (3 gedung, 15 lantai dan 20 kamar) dengan kondisi seluruh kamar available dengan memasukkan “-“ kedalam list.
- b. Buatlah sebuah program yang dapat memasukkan, sekali run adalah satu kali booking:
 - Gedung Hotel : 1/2/3
 - Lantai : 1-15
 - Nomer kamar : 1-20
- c. Dan luaran yang diharapkan adalah :
 - Harga kamar yang harus dibayar
 - Tampilan list hasil dari booking yang dimasukkan dengan kondisi kamar yang di booking berganti menjadi “+”

Lakukan ujicoba dengan memasukkan booking sebagai berikut :

- a. Booking kamar : Gedung 1, Lantai 15, Kamar 8.

OUTPUT : "Anda memilih kamar PRESIDENT dengan harga Rp. 1.000.000"

- b. Booking kamar : Gedung 3, Lantai 7, Kamar 1.

OUTPUT : "Anda memilih kamar STANDARD dengan harga Rp. 500.000"

- c. Booking kamar : Gedung 2, Lantai 14, Kamar 1.

OUTPUT : "Anda memilih kamar DELUXE dengan harga Rp. 700.000"

- d. Booking kamar : Gedung 1, Lantai 14, Kamar 5.

OUTPUT : "Anda memilih kamar SUPERIOR dengan harga Rp. 850.000"

BAB V

DICTIONARY DAN TUPLE

CAPAIAN PEMBELAJARAN (*Learning Out Come*)

1. Mahasiswa memahami bentuk dictionary dan tuple
2. Mahasiswa memahami sifat dan bentuk modifikasi pada dictionary dan tuple

A. PENDAHULUAN

Selain daripada list, Python juga memiliki tipe data lainnya yang akan sangat bermanfaat dalam pemrosesan data. Sifat dari data ada dua, yaitu *mutable* yaitu data yang dapat dirubah sedangkan *immutable* adalah data yang tidak dapat dirubah. Keduanya memiliki keuntungan dan kelemahannya masing-masing, sehingga pilihlah tipe data yang sesuai agar program berjalan dengan baik.

B. TUPLE

Sebuah tupel adalah merupakan sebuah list yang urutannya tidak dapat berubah. List datanya bisa dirubah selama eksekusi program, sifatnya bernama *mutable* sebaliknya tupel tidak bisa, sifatnya dinamakan *immutable*. Persamaannya keduanya sama-sama bisa menyimpan banyak nilai sekaligus dalam satu tipe data (*sequence type*). Tupel menggunakan tanda kurung, sedangkan List Python menggunakan tanda kurung siku.

Bentuk umum untuk membuat list dalam python adalah :

```
nama_tuple = (nilai1, nilai2, nilai3, ...)
```

Perhatikan contoh berikut untuk membedakan penulisan list dan tupel :

```
myTuple = (1, 2, True, "a string", (3, 4), [5, 6], None)
print(myTuple)

myList = [1, 2, True, "a string", (3, 4), [5, 6], None]
print(myList)
```

Penulisan tupel juga dapat dilakukan tanpa menggunakan buka dan tutup kurung. Lakukan percobaan dibawah :

```
tuple1 = (1, 2, 4, 8)
tuple2 = 1., .5, .25, .125

print(tuple1)
print(tuple2)
```

Tupel kosong juga dapat dibuat sebagaimana contoh dibawah. Tupel kosong nantinya akan diisi dengan algoritma yang diinginkan. Lakukan percobaan berikut :

```
emptyTuple = ()
print(type(emptyTuple))
```

Untuk menulis tupel yang berisi satu nilai, anda harus memasukkan koma, meskipun hanya ada satu nilai. Jika tidak maka akan teridentifikasi sebagai variabel dengan satu data. Lakukan percobaan berikut :

```
tup1=(50,)
tup2=(50)

print(type(tup1))
print(type(tup2))
```

1. Akses Nilai pada Tupel

Untuk mengakses nilai sama dengan bagaimana dilakukan pada indexing list, gunakan tanda kurung siku untuk mengiris beserta indeks atau indeks untuk mendapatkan nilai yang tersedia pada indeks tersebut. Lakukan percobaan sebagai berikut :

```
#Cara mengakses nilai tuple
tup1 = ('fisika', 'kimia', 1993, 2017)
tup2 = (1, 2, 3, 4, 5, 6, 7 )
print ("tup1[0]: ", tup1[0])
print ("tup2[1:5]: ", tup2[1:5])
```

2. Update Nilai

Tupel tidak berubah, yang berarti Anda tidak dapat memperbarui atau mengubah nilai elemen tupel. Anda dapat mengambil bagian dari tupel yang ada untuk membuat tupel baru. Lakukan contoh berikut :

```
tup1 = (12, 34.56)
tup2 = ('abc', 'xyz')

tup3 = tup1 + tup2
print (tup3)
```

Selain daripada mengupdate dengan operasi tambah, tupel juga dapat digandakan dengan menggunakan operasi perkalian, Lakukan percobaan berikut :

```
myTuple = (1, 10, 100)
t2 = myTuple * 3
print(t2)
```

Menghapus maupun menambahkan elemen individual pada tupel tidak mungkin dilakukan. Tentu saja, tidak ada yang salah dengan menggabungkan tupel lain dengan unsur-unsur yang tidak diinginkan dibuang. Untuk secara eksplisit menghapus keseluruhan tuple, cukup gunakan del(). Lakukan percobaan dibawah, hasil akan menunjukkan error karena tupel sudah dihapus, bukan hanya isinya namun seluruh variabelnya.

```
tup = ('fisika', 'kimia', 1993, 2017)
print
(tup)
del tup
print ("Setelah menghapus tuple : ",tup)
```

3. Indexing, Slicing dan Matrix

Karena tupel adalah urutan, pengindeksan dan pengiris bekerja dengan cara yang sama untuk tupel seperti pada String, dengan asumsi masukan dibawah, terlihat pada Tabel 2:

```
T = ('C++', 'Java', 'Python')
```

Python Expression	Hasil	Penjelasan
T[2]	'Python'	Offset mulai dari nol
T[-2]	'Java'	Negatif: hitung dari kanan
T[1:]	('Java', 'Python')	Slicing mengambil bagian

4. Fungsi Built-in

Python menyertakan fungsi built-in sebagaimana ditunjukkan pada Tabel dibawah.

Python Function	Penjelasan
cmp(tuple1, tuple2)	# Tidak lagi tersedia dengan Python 3
len(tuple)	Memberikan total panjang tuple.
max(tuple)	Mengembalikan item dari tuple dengan nilai maks.
min(tuple)	Mengembalikan item dari tuple dengan nilai min.
Python Function	Penjelasan
tuple(seq)	Mengubah tuple menjadi tuple.

C. DICTIONARY

Dictionary Python berbeda dengan List ataupun Tuple. Karena setiap urutannya berisi *key* dan *value*. Setiap *key* dipisahkan dari *value*-nya oleh titik dua (:), item dipisahkan oleh koma, dan semuanya tertutup dalam kurung kurawal. Berikut beberapa syarat dari pasangan *key-value* :

- Setiap *key* harus unik. Tidak diperbolehkan didalam satu dictionary memiliki dua *key* yang sama
- *Key* dapat berupa tipe data apapun. *Integer*, *float* maupun string.
- Perbedaan dengan list adalah dictionary memiliki sepasang *key-value* untuk setiap elemennya.
- *Dictionary* merupakan *tool* satu arah, sebagaimana kamus *English-French*, namun tidak berlaku untuk *French-English*.

Dictionary kosong tanpa barang ditulis hanya dengan dua kurung kurawal, seperti ini: {}. Nilai kamus bisa berupa tipe apa pun, namun *key* harus berupa tipe data yang tidak berubah seperti string, angka, atau tupel. Bentuk umum untuk membuat list dalam python adalah :

```
nama_dict = {key1:nilai1, key2:nilai2, key3:nilai3,...}
```

Dictionary bersifat tidak berurutan sehingga pada *dictionary* tidak dapat digunakan index untuk mengaksesnya melainkan menggunakan nilai *key* untuk mengakses *value* nya. *Dictionary* juga bersifat *changeable (mutable)* sebagaimana list.

Lakukan percobaan berikut :

```
dict1 = {"cat" : "chat", "dog" : "chien", "horse" : "cheval"}  
phoneNumbers = {'boss' : 5551234567, 'Suzy' : 22657854310}  
emptyDictionary = {}  
  
print(dict1)  
print(phoneNumbers)  
print(emptyDictionary)
```

Akses Nilai pada *Dictionary*

Untuk mengakses elemen *Dictionary*, Anda dapat menggunakan tanda kurung siku yang sudah dikenal bersama dengan *key* untuk mendapatkan nilainya. Lakukan percobaan berikut :

```
dict1 = {"cat" : "chat", "dog" : "chien", "horse" : "cheval"}  
phoneNumbers = {'boss' : 5551234567, 'Suzy' : 22657854310}  
  
print(dict1['cat'])  
print(phoneNumbers['Suzy'])
```

Update Nilai pada *Dictionary*

Anda dapat memperbarui *Dictionary* dengan menambahkan entri baru atau pasangan nilai kunci, memodifikasi entri yang ada, atau menghapus entri yang ada. Lakukan percobaan berikut :

```
dict1 = {"cat" : "chat", "dog" : "chien", "horse" : "cheval"}  
  
dict1['cat'] = 'minou'  
print(dict1)  
dict1['swan'] = 'cygne'  
print(dict1)
```

Anda dapat menghapus elemen *Dictionary* individual atau menghapus keseluruhan isi *Dictionary* dengan menggunakan fungsi **del()**. Anda juga dapat menghapus seluruh *Dictionary* dalam satu operasi. Untuk menghapus seluruh *Dictionary* secara eksplisit, cukup gunakan **del()** statement. Lakukan percobaan berikut :

```
dataSiswa = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
del dataSiswa['Name']
print(dataSiswa)

del dataSiswa
print(dataSiswa)
```

Luaran print terakhir akan bernilai *error* karena variabel sudah tidak ada.

Fungsi **clear()** dapat digunakan untuk menghapus isi *dictionary* tanpa menghilangkan variabel *dictionary*.

```
dataSiswa = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
dataSiswa.clear()
print(dataSiswa)
```

Keys, Values dan Items

Terdapat fungsi pada python yang dapat mengakses *key*- nya saja, *value*- nya saja atau items (*key-value*). Untuk mengakses *key* saja digunakan perintah **keys()**. Lakukan percobaan berikut :

```
dict1 = {"cat" : "chat", "dog" : "chien", "horse" : "cheval"}

for key in dict1.keys():
    print(key, "->", dict1[key])
```

Keys juga dapat di sort dengan menggunakan perintah **sorted()**. Lakukan percobaan berikut :

```
for key in sorted(dict1.keys()):
    print(key, "->", dict1[key])
```

Value saja juga bisa diakses dengan perintah **values()**. Lakukan percobaan berikut :

```

dict1 = {"cat" : "chat", "dog" : "chien", "horse" : "cheval"}

for french in dict1.values():
    print(french)

```

Untuk mengakses keduanya dapat digunakan perintah **items()**. Lakukan percobaan berikut :

```

dict1 = {"cat" : "chat", "dog" : "chien", "horse" : "cheval"}

for english, french in dict1.items():
    print(english, "->", french)

```

D. SOAL LATIHAN/ TUGAS

Buatlah sebuah sistem untuk menghitung total belanjaan pada sebuah toko sayur dengan list harga sebagai berikut :

Kode	Sayur	Harga / buah
1	Bayam	Rp 1500
2	Terong	Rp. 2000
3	Kubis	Rp 3000
4	Labu	Rp 4000
5	Brokoli	Rp 4500

Input:

- Jumlah jenis sayur yang akan dibayar
- Masukan kode sayur
- Masukan jumlah (buah)

Notes: lakukan perulangan b dan c sejumlah input a

Output:

- List belanjaan sayur (key) dan jumlahnya (value) dalam bentuk dictionary
- Print total belanjaan yang harus dibayar dengan kondisi sebagai berikut:
 - Jika total belanjaan lebih dari Rp 20.000 maka didapatkan diskon 10%.
 - Jika total belanjaan lebih dari Rp 50.000 maka didapatkan diskon 15%.
 - Jika total belanjaan lebih dari Rp 100.000 maka didapatkan diskon 20%.

Ujicoba:

Barang Belanjaan	Total yang harus dibayar
Bayam 5, Kubis 3, Brokoli 7	Rp 43.200
Terong 8, Labu 5, Bayam 3	Rp 36.450
Brokoli 8, Labu 10, Kubis 15, Terong 4	Rp 103.200
Bayam 5, Terong 7, Kubis 6, Labu 4, Brokoli 5	Rp 66.300

Mekanisme Pengerjaan:

- a. Penejelasan potongan source (per bagian) kode program yang dibuat.
- b. Penjelasan (bisa diisikan alur perhitungan manual) dan *screenshot* setiap uji coba.

BAB VI

FUNCTION

CAPAIAN PEMBELAJARAN (*Learning Out Come*)

1. Mahasiswa memahami cara membuat fungsi
2. Mahasiswa memahami cara membuat module
3. Mahasiswa memahami cara membuat package

A. PENDAHULUAN

Kompleksnya pemrograman membuat beberapa kali dalam program akan digunakan berulang. Selain akan menyulitkan ketika trouble shooting, program akan terlihat berantakan dan tidak terstruktur. Kelemahan lainnya adalah ketika program harus dikerjakan bersama, akan menyulitkan jika dikerjakan dalam satu file yang prosedural. Python menyediakan solusi untuk kondisi ini yaitu dengan fungsi, yaitu komputasi yang dapat digunakan berulang didalam satu file, kemudian modul yang memuat beberapa fungsi, dan yang terakhir adalah package yang memuat beberapa modul. Hal ini akan memudahkan developer untuk mengembangkan dan juga melakukan *trouble shooting* terhadap programnya.

B. FUNGSI

Fungsi merupakan susunan *statement* yang teratur dengan tujuan agar dapat dijalankan/eksekusi lebih dari satu kali. Ada beberapa kelebihan dari penggunaan fungsi yaitu:

- Memaksimalkan penggunaan ulang sebuah operasi dan mengurangi pengulangan pengetikan operasi tersebut.
- Dekomposisi prosedur

Dalam bahasa pemrograman yang berbeda fungsi disebut juga dengan *subroutines* atau *procedure*. simbol juga yang memisahkan antara fungsi dan *procedure*. Di dalam Python fungsi ditulis menggunakan *statement* baru yaitu **def**. Selain penggunaan statemen

tersebut, ada beberapa ketentuan dalam fungsi yang juga harus dicermati yang akan dijelaskan di bawah ini.

1. Statement def

```
def name(arg1, arg2, ..., argN):  
    statements
```

Dengan statemen ini, objek fungsi dibuat dan diberi nama dengan sintaksnya:

2. Statement return

Dengan menggunakan statemen ini maka keluaran dari sebuah fungsi dapat

```
def name(arg1, arg2, ..., argN):  
    statements  
return value
```

dipanggil oleh program yang memanggil. Sintaksnya sebagai berikut:

```
def maks(x, y):  
    if x > y:  
        return x  
    else:  
        return y
```

Contoh syntax diatas merupakan fungsi yang menguji nilai paling besar antara 2 nilai masukan lalu mengembalikannya ke program pemanggil.

3. Memanggil Fungsi

Sebuah fungsi dapat dipanggil langsung di dalam program utama (program pemanggil). Jika sebuah fungsi memiliki sebanyak N argument input maka di dalam program utama harus memberikan nilai pada N argument input tersebut. Jika tidak,

```
def maks(x, y):  
    if x > y:  
        return x  
    else:  
        return y  
  
a = int(input('Nilai a: '))  
b = int(input('Nilai b: '))  
c = maks(a, b)  
print('Nilai maksimum antara', a, 'dan', b, ':', c)
```

maka akan memunculkan error pada program. Contoh program dibawah merupakan program utama yang memanggil fungsi pada sebelumnya yang juga telah ditanamkan pada program utama tersebut.

Perlu diingat bahwa sebuah fungsi yang akan dipanggil oleh program utama harus terlebih dahulu di definisikan sebelum dipanggil. Pada berikutnya posisi dari pendefinisian fungsi diletakkan setelah dipanggil.

```
a = int(input('Nilai a: '))
b = int(input('Nilai b: '))
c = maks(a, b)
print('Nilai maksimum antara', a, 'dan', b, '=', c)

def maks(x, y):
    if x > y:
        return x
    else:
        return y
```

Jalankanlah program pada diatas dan perhatikan keluarannya. Hal ini berbeda dengan kemampuan C karena pendefinisian fungsi dapat dituliskan sebelum dan sesudah *main program* hanya saja untuk kasus dimana fungsi didefinisikan setelah program utama, terlebih dahulu diciptakan sebelum program utama dituliskan. Namun, bukan berarti Python tidak memiliki fleksibilitas dalam menyatakan fungsi. Fungsi dapat ditulis di dalam program utama seperti pada syntax dibawah. Jalankanlah program untuk mendapatkan hasil program yang bisa anda observasi terkait fleksibilitas fungsi dalam Python.

```
a = int(input('Bilangan: '))
b = int(input('Pangkat: '))
def xpangkatn(x, n):
    return x ** n
c = xpangkatn(a, b)
print(a, '^', b, '=', c)
```

4. Scope Varibel

Semua variabel di dalam sebuah fungsi bersifat *symbo*. Yang mengandung makna bahwa meskipun dua buah variabel memiliki nama yang sama tapi salah satu berada di dalam fungsi dan yang lain di dalam program utama maka keduanya adalah dua

variabel yang berbeda. Perhatikan program dibawah yang menjelaskan tentang hal ini.

Ketika program tersebut dijalankan, akan menampilkan hasil sebagai berikut

```
x = 4  
z = 4  
x = 3  
y = 2
```

```
def f(x):  
    y = 1  
    x = x + y  
    print ('x =', x)  
    return x  
  
x = 3  
y = 2  
z = f(x)  
print ('z =', z)  
print ('x =', x)  
print ('y =', y)
```

5. Fungsi Bersarang

Sama halnya dengan bahasa pemrograman yang lain, Python juga memiliki fleksibilitas dalam penempatan fungsi. Sebuah fungsi bisa menjadi bagian dari fungsi yang lain dan tidak terbatas jumlahnya. Perhatikan program dibawah yang menjelaskan penggunaan fungsi (*children*) di dalam fungsi lain (*parent*).

```
x = 3  
z = f(x)  
print ('x =', x)  
print ('z =', z)  
z()
```

```
def f(x):  
    def g():  
        x = 'abc'  
        print ('x =', x)  
    def h():  
        z = x  
        print ('z =', z)  
    x = x + 1  
    print ('x =', x)  
    h()  
    g()  
    print ('x =', x)  
    return g
```

ketik dan jalankan program tersebut dan cermati satu demi satu keluarannya.

6. Lambda

Jika sebuah fungsi sederhana yang hanya berisi tentang evaluasi sebuah ekspresi, maka fungsi tersebut dapat dituliskan sebagai *lambda*. Perhatikan program dibawah yang menjelaskan tentang penggunaan *lambda*.

```
pangkat = lambda a, b: a ** b  
  
a = 2  
b = 3  
c = pangkat(a, b)  
print(a, '^', b, '=', c)
```

Ketik dan jalankan program untuk mendapatkan hasil yang jelas tentang penggunaan *lambda*.

7. Rekursi

Diartikan sebagai kemampuan sebuah fungsi memanggil dirinya sendiri. Dengan menggunakan rekursif, biasanya program menjadi lebih sederhana. Tidak semua persoalan dapat diselesaikan dengan rekursif, hanya untuk persoalan yang memiliki pola – pola khusus saja dalam ekspresi matematisnya. Perhatikan program dibawah yang menjelaskan tentang rekursif.

```
def factR(n):  
    if n == 1:  
        return n  
    return n*factR(n-1)
```

C. SOAL LATIHAN/ TUGAS

Buatlah sebuah program kalkulator dengan menggunakan fungsi. Pilihan dari kalkulator tersebut minimal ada enam komputasi, seperti tambah kurang dan lainnya. Alur adalah user memasukkan input, dan program akan memanggil fungsi yang dipilih.

BAB VII

MODULE, PACKAGE DAN PIP

CAPAIAN PEMBELAJARAN (*Learning Out Come*)

1. Mahasiswa memahami bagaimana cara membuat module
2. Mahasiswa memahami bagaimana cara membuat package
3. Mahasiswa memahami guna PyPi

A. PENDAHULUAN

Kode komputer memiliki kecenderungan untuk berkembang, semakin besarnya program akan semakin banyak masalah yang ditemukan. Kode yang banyak memerlukan maintenance yang besar juga, trouble shoot akan jauh lebih mudah ketika kode tidak terlalu banyak. Jika anda menginginkan pembangunan project anda lancar, maka anda harus dapat membagi pekerjaan kebeberapa orang dan kemudian menyatukannya lagi dalam program yang utuh. Contohnya adalah membagi program ke bagian UI (user interface) dan juga logic, dan seterusnya, kedua bagian ini akan juga dibagi ke beberapa bagian yang lebih kecil.

B. MODULE

Module merupakan kumpulan beberapa fungsi yang kemungkinan akan dipakai berulang pada program anda. Peran didalam modul ada dua jenis yaitu :

- User (pengguna) : menggunakan module yang sudah ada kedalam programnya
- Supplier (pembuat) : membuat modul dan membagikannya kepada user.

Module diidentifikasi dengan Namanya. Setiap modul memiliki entitas (sebagaimana chapter didalam buku) yang mengandung fungsi, variabel, konstanta, class dan juga objects.

Posisikan diri anda sebagai user yang akan menggunakan module. Kita akan memanggil module yang ada pada python kemudian menggunakananya pada program. cara memanggil

sebuah module adalah dengan menggunakan instruksi `import` sebagaimana contoh dibawah yaitu meng-import module math pada program yang akan dikerjakan.

```
import math
```

a. Namespace

Namespace pada kasus ini adalah dimaksudkan agar fungsi yang kita panggil terdapat pada dua module yang berbeda, maka cara memanggilnya adalah nama module kemudian diikuti dengan fungsinya. Sebagaimana kita memanggil dua atau yang sama dari dua keluarga yang berbeda, maka akan disertakan dengan nama belakangnya. Perhatikan contoh dibawah :

```
import numpy
import math
import scipy

pi=200
print(math.pi)
print(numpy.pi)
print(scipy.pi)
print(pi)
```

Ketiga module numpy, math maupun scipy memiliki pi. Namun ketika pi tersebut dipanggil, maka nama module dipanggil dibagian depannya. Seperti `math.pi`.

Karena nama module akan selalu dipanggil terutama ketika program menggunakan sejumlah module bersamaan, maka nama module bisa disingkat (aliasing) sebagaimana syntax dibawah :

```
import math as m
print(m.pi)
```

b. Directory (dir)

Directory merupakan sebuah instruksi untuk mengetahui fungsi apa saja yang ada pada module tersebut. Instruksi dapat dilihat pada Gambar dibawah :

```
import math  
dir(math)
```

c. Import Sebagian module

Sebuah module memuat beberapa fungsi, ada module dengan sangat banyak fungsi dan ada juga yang tidak. Instruksi import sesungguhnya mengambil semua fungsi dari satu module untuk digunakan dalam program. Bagaimana jika hanya sebagian yang ingin digunakan? Andaikan kita menggunakan beberapa fungsi dari module yang berbeda-beda. Jika seluruh modul harus di import maka akan membutuhkan komputasi yang cukup lama. Cara mengambil sebagian fungsi dari modul adalah sebagai berikut :

```
from math import sin, pi
```

Program ini menunjukkan bahwa fungsi yang diambil adalah sin dan pi saja dari seluruh module math. Untuk lebih memahami fungsi yang dipanggil, lakukan program berikut :

```
from math import sin, pi  
  
print(sin(pi/2))  
pi = 3.14  
  
def sin(x):  
    if 2 * x == pi:  
        return 0.99999999  
    else:  
        return None  
print(sin(pi/2))
```

TUGAS 1 : Jelaskan hasil yang didapatkan.

d. Module math

Module math berisi fungsi-fungsi matematika dasar, seperti akar pangkat (square root), pangkat (power) sampai dengan trigonometri.

TUGAS 2 : Gunakan fungsi berikut dan jelaskan hasil luarannya

```
from math import ceil, floor, trunc

x = 1.4
y = 2.6

print(floor(x), floor(y))
print(floor(-x), floor(-y))
print(ceil(x), ceil(y))
print(ceil(-x), ceil(-y))
print(trunc(x), trunc(y))
print(trunc(-x), trunc(-y))
```

TUGAS 3 : Gunakan fungsi berikut dan jelaskan hasil luarannya dengan perhitungan matematika

```
from math import e, exp, log

print(pow(e, 1) == exp(log(e)))
print(pow(2, 2) == exp(2 * log(2)))
print(log(e, e) == exp(0))
```

e. Module random

Module random digunakan untuk men-generate nilai-nilai random yang dapat digunakan untuk berbagai tujuan. Nilai random tidak dapat di prediksi, dan ketika kita meng eksekusinya ulang, akan didapatkan hasil yang berbeda pula. Cobalah instruksi berikut, dan eksekusi berulang- ulang, maka akan didapatkan nilai yang berbeda-beda pula :

```
from random import random

for i in range(5):
    print(random())
```

Jika kita menginginkan nilai integer yang random maka dapat menggunakan beberapa instruksi berikut :

- randrange(end)
- randrange(beg, end)
- randrange(beg, end, step)
- randint(left, right)

Cobalah perintah berikut dan lakukan eksekusi berulang -ulang :

```
from random import randrange, randint

print(randrange(10))
print(randrange(0, 10))
print(randrange(0, 35, 2))
print(randint(0, 10))
```

TUGAS 4 : Dari hasil yang didapatkan jelaskan maksud dari instruksi tersebut.

Fungsi lainnya yang ada didalam module Random adalah choice dan sample. Choice digunakan untuk memilih acak salah satu dari data yang ada, sedangkan sample adalah instruksi untuk mengambil beberapa sampel acak dari data yang ada. Perhatikan contoh berikut :

```
from random import choice, sample

lst = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

print(choice(lst))
print(sample(lst, 5))
print(sample(lst, 10))
```

Kali ini posisikan diri anda sebagai supplier, atau pembuat module. Mari kita mencoba membuat module sederhana dengan instruksi sebagai berikut :

1. Buatlah dua file python dengan nama **aritmatika.py** dan **main.py**
2. Isikan file **aritmatika.py** dengan syntax berikut :

```
def tambah(a,b):
    return a+b

def kurang(a,b):
    return a-b

def kali(a,b):
    return a*b

def bagi(a,b):
    return a/b
```

3. Isikan file **main.py** dengan syntax berikut :

```
import aritmatika

a=aritmatika.tambah(3,4)
b=aritmatika.kurang(3,4)
c=aritmatika.kali(3,4)
d=aritmatika.bagi(3,4)

print(a)
print(b)
print(c)
print(d)
```

Pada percobaan ini, kita telah membuat module kita sendiri dengan nama aritmatika yang kemudian module tersebut di import pada file main.py

TUGAS 5 : Buatlah module lainnya dengan metode yang sama, dan coba terapkan

C. PACKAGE

Seperti yang dibahas diatas bahwa module adalah wadah dari sekelompok fungsi. Kita dapat berfungsi sebagai user, yaitu dengan menggunakan module yang tersedia, dan juga bisa berperan sebagai supplier, yaitu dengan membuat module dan mendistribusikannya ke seluruh dunia. Sebaiknya fungsi yang ada didalam sebuah module adalah sejenis atau digunakan pada aplikasi yang sama. Jika kita ingin membuat beberapa module, akan sangat mungkin tidak tertata dengan baik. Maka ada package yaitu wadah berisi beberapa modul yang dapat juga digunakan dan dibuat oleh kita sendiri.

Kali ini kita akan membuat package namun akan disimpan pada komputer kita sendiri. Lakukan instruksi berikut :

1. Pada path python anda, buatlah folder dengan nama LATIHAN PYTHON
2. Dalam folder LATIHAN PYTHON, buatlah folder dengan nama latihan_package, sebuah file python bernama main.py dan sebuah file python bernama __init__.py
3. Dalam folder latihan_package, buatlah dua file python dengan nama alpha.py dan beta.py
4. Isilah file alpha.py dengan syntax berikut dan simpan :

```
#alpha.py
def alphaSatu():
    print("alpha Satu")

def alphaDua():
    print("alphaDua")
```

5. Isilah file beta.py dengan syntax berikut dan simpan:

```
#beta.py
def betaSatu():
    print("betaSatu")

def betaDua():
    print("betaDua")
```

6. Isilah file main.py dengan syntax berikut dan simpan :

```
import latihan_package.alpha as a
import latihan_package.beta as b

a.alphaSatu()
```

coba eksekusi dan perhatikan folder LATIHAN_PYTHON yang tadi dibuat.

D. PYTHON PACKAGE INSTALLER (PIP)

Python adalah software open-source, sehingga perkembangannya tidak hanya dilakukan oleh satu badan usaha namun juga memberikan kesempatan untuk developer membantu untuk mengembangkannya. Tidak hanya berkontribusi sebagai supplier, namun juga dapat sebagai user. Python mengajak penggunanya untuk menjadi ekosistem python agar tetap terbuka. Agar tujuan ini tercapai, maka diperlukan sebuah tools untuk menerbitkan, menjaga dan mengatur package yang di upload. Terdapat dua hal yang sudah ada yaitu :

- Gudang repository yaitu perkumpulan package yang tersentralisasi
- Tools untuk mengakses repository tersebut

Repository atau disingkat dengan repo Bernama PyPI (Python Package Index) yang diatur oleh Packaging Working Group (PWG) yang dapat diakses pada link berikut : <https://pypi.org/>. Untuk mengakses PyPI dibutuhkan sebuah tools yang dinamakan pip atau singkatan dari Python Package Installer.

Pip umumnya sudah terbawa bersamaan dengan python yang anda install dari Pyhton.org. Begitu juga jika anda menggunakan Anaconda. Jika tidak, dapat diinstall secara manual melalui link : <https://bootstrap.pypa.io/get-pip.py>

Beberapa instruksi untuk mengetahui kondisi pip pada komputer anda adalah sebagai berikut :

- `pip -version` → melihat versi dari pip
- `pip list` → melihat semua package yang sudah terinstall
- `pip help` → melihat bantuan

- `pip help install` → melihat bantuan (help) untuk command install
- `pip install package_name` → meng-install package symbol package_name
- `pip install pygame` → meng-install package symbol pygame

- `pip install -U package_name` → meng-install package symbol package_name dengan versi terbaru

- `pip install -U pygame` → meng-install package symbol pygame dengan versi terbaru
- `pip install package_name==package_version` → meng-install package symbol package_name dengan versi package_version
- `pip install pygame==1.9.2` → meng-install package symbol package_name dengan versi 1.9.2

- `pip show package_name` → melihat detail dari package package_name
- `pip show pygame` → melihat detail dari package pygame

- `pip uninstall package_name` → meng-uninstall package symbol package_name
- `pip uninstall pygame` → meng-uninstall package symbol pygame

TUGAS 6 : Cobalah instruksi diatas dan screenshot hasilnya

BAB VIII

EKSEPSI, LIST PROCESSING AND FILES

CAPAIAN PEMBELAJARAN (*Learning Out Come*)

1. Mahasiswa memahami penggunaan eksepsi di python
2. Mahasiswa memahami bagaimana memproses list dan build in function yang ada pada string
3. Memahami cara mengambil, menyimpan dan memproses file

A. EKSEPSI

Kesalahan merupakan hal yang sering terjadi pada saat proses membuat pemrograman. Sebab terjadinya kesalahan bisa karena kesalahan dalam penulisan syntax, sehingga error dan tidak dapat dijalankan sama sekali. Namun bisa juga karena kesalahan pada saat program sedang di eksekusi. Terdapat dua buah cara untuk memeriksa sebuah kesalahan yaitu :

- menggunakan blok try....except
- menggunakan statement assert

Contoh kesalahan sederhana yang terjadi adalah user menginputkan string atau bilangan negatif, yang mana input yang kita harapkan adalah berbentuk integer. Jika ini terjadi, maka program akan otomatis terhenti. Contoh program yang akan menghasilkan error adalah sebagai berikut :

```
import math

x = float(input("Enter x: "))
y = math.sqrt(x)

print("The square root of", x, "equals to", y)
```

Masukkan input sebuah karakter huruf, dan lihat hasilnya. Syntax diatas akan menghasilkan error dan memunculkan (raising) eksepsi. Python akan memunculkan eksepsi ketika tidak tahu lagi apa yang harus dilakukan dengan kode anda, yang menyebabkan beberapa kejadian sebagai berikut :

- eksepsi mengharapkan ada sebuah perintah yang dapat menangani atau mencegah terjadinya kesalahan
- jika tidak ada perintah untuk menangani atau mencegah terjadinya kesalahan tersebut, python akan menghentikan atau **terminated** program, sehingga akan muncul pesan **error**
- jika kesalahan dapat ditangani, python akan melanjutkan pada kode program selanjutnya

Perhatikan contoh dibawah :

```
try:  
    print("1")  
    x = 1 / 0  
    print("2")  
except:  
    print("Ada yang salah nih")  
  
print("3")
```

Tugas 1 : Coba modifikasi syntax dengan error diatas dengan menggunakan try...except

Error yang sering sekali terjadi adalah `NameError`, `ValueError`, `ZeroDivisionError`
Perhatikan contoh dibawah untuk jenis error yang spesifik.

```
try:  
    y = 1 / 0  
except ZeroDivisionError:  
    print("ada yang salah nih...")  
  
print("THE END.")
```

Block `try ... except` dapat digunakan untuk menangani lebih dari satu eksepsi, dengan menggunakan bentuk umum:

```
try:  
:  
except excl:
```

```
:  
except exc2:  
:  
except:  
:  
:
```

Satu atau beberapa statement yang terdapat dalam blok try dapat menimbulkan lebih dari satu tipe ekspesi. Perhatikan contoh dibawah :

```
try:  
    x = int(input("Masukkan angka: "))  
    y = 1 / x  
    print(y)  
except ZeroDivisionError:  
    print("tidak dapat dibagi nol.")  
except ValueError:  
    print("silahkan masukkan angka bulat.")  
except:  
    print("ada yang salah dari input anda")  
  
print("THE END.")
```

Tugas 2 : Coba cari jenis error yang lain yang kerap ditemukan, dan coba buat syntaxnya

B. STRINGS

Pada tabel dibawah, anda akan menemukan daftar fungsi dari strings yang dapat digunakan langsung dalam memanipulasi variabel strings.

Fungsi	Penjelasan
format()	It's used to create a formatted string from the template string and the
split()	Python string split() function is used to split a string into the list of strings
join()	This function returns a new string that is the concatenation of the strings
strip()	Used to trim whitespaces from the string object.
format map()	Python string format map() function returns a formatted version of the
upper()	We can convert a string to uppercase in Python using str.upper() function.
lower()	This function creates a new string in lowercase.
replace()	Python string replace() function is used to create a new string by replacing
find()	Python String find() method is used to find the index of a substring in a

translate()	Python String translate() function returns a new string with each character
-------------	---

Tabel 4.2 Fungsi – fungsi tambahan

Fungsi	Penjelasan
<u>isalnum()</u>	Python string isalnum() function returns True if it's made of alphanumeric
<u>isalpha()</u>	Python String isalpha() function returns True if all the characters in the
<u>isdecimal()</u>	Python String isdecimal() function returns True if all the characters in the
<u>isdigit()</u>	Python String isdigit() function returns True if all the characters in the
<u>isidentifier()</u>	Python String isidentifier() function returns True if the string is a valid
<u>islower()</u>	Python String islower() returns True if all cased characters in the string are
<u>isnumeric()</u>	Python String isnumeric() function returns True if all the characters in the
<u>isprintable()</u>	Python String isprintable() function returns True if all characters in the
<u>isspace()</u>	Python String isspace() function returns True if there are only whitespace
<u>istitle()</u>	Python String istitle() returns True if the string is title cased and not empty,
<u>isupper()</u>	Python String isupper() function returns True if all the cased characters are
<u>rjust(), ljust()</u>	Utility functions to create a new string of specified length from the source
<u>swapcase()</u>	Python String swapcase() function returns a new string with uppercase
<u>partition()</u>	Python String partition() function splits a string based on a separator into
<u>splittlines()</u>	Python String splittlines() function returns the list of lines in the string.
<u>title()</u>	Python String title() function returns a title cased version of the string.
<u>zfill()</u>	Python String zfill(width) function returns a new string of specified width.

Perhatikan syntax dibawah yang menggunakan salah satu fungsi yang terdapat pada modul *strings*. Jika anda amati dengan seksama, cara penulisan fungsi – fungsi manipulasi strings adalah **var.(fungsi)** nya.

```
nama = "Sekolah Vokasi ugM"  
NAMA = nama.upper()  
print(nama)  
print(NAMA)
```

Tugas 3 : Gunakan minimal tiga build in function diatas untuk dipraktekan dalam program

C. FILES

Python juga memiliki kemampuan untuk membaca dan menuliskan data dari dan ke file *text processing*.

1. Membaca dan Menulis Data Text

Untuk membaca dan menulis *text data* dapat mengikuti Langkah – Langkah berikut ini:

- Membuka file yang dituju dengan perintah *with* dan *open*
- Menggunakan perintah *read* jika ingin membaca isi dari *text*
- Menggunakan perintah *write* jika ingin menuliskan *data* ke dalam *text*
- Dengan menggunakan perintah *with*, maka setelah proses membaca dan menulis data selesai, maka *object* dari *text* secara otomatis ditutup.
- Jika tidak menggunakan perintah *with*, maka di akhir operasi menulis dan membaca, harus ditutup dengan perintah *close*.

Perhatikan dibawah yang menggambarkan cara menuliskan data ke sebuah *text processor*. Biasanya *text processor* yang dimaksud adalah Notepad (Windows).

```
namaFile = 'foo.txt'  
with open(namaFile, 'wt') as f:  
    f.write('Belajar Python dengan Pycharm\n')  
    f.write('Sekolah Vokasi')
```

Pada diatas penulisan data dilakukan menggunakan perintah *write*. Sementara untuk penulisan baris demi baris, menggunakan perintah *print*.

```
namaFile = 'foo1.txt'  
with open(namaFile, 'wt') as f:  
    print('Belajar Python dengan Pycharm', file = f)  
    print('Sekolah Vokasi', file = f)  
    print('Universitas Gadjah Mada', file = f)
```

Fungsi *write* tidak secara *default* dilengkapi dengan fitur *newline* seperti pada *print*. Oleh karena itu, pada fungsi *write* dapat ditambahkan ‘\n’ yang menambahkan baris baru untuk perintah berikutnya.

Pada contoh berikutnya, diperlihatkan bagaimana membaca isi dari *text* yang dihasilkan dari contoh sebelumnya yang dibaca secara sekaligus.

```
namaFile = 'foo.txt'  
with open(namaFile, 'rt') as f:  
    data = f.read()  
print(data)
```

```
namaFile = 'foo1.txt'  
with open(namaFile, 'rt') as f:  
    for baris in f:  
        print(baris, end='')
```

Sementara pembacaan *text* yang dilakukan baris demi baris diperlihatkan pada contoh berikutnya.

Pembacaan data pada sebuah *text* baris demi baris sangat bermanfaat ketika ada proses lanjutan yang hanya bisa dikenakan baris data demi baris data. Jika pembacaan atau penulisan data pada *text* tidak menggunakan *with*, maka object *file* harus ditutup menggunakan perintah *close* seperti pada syntax dibawah.

```
namaFile = 'foo1.txt'  
f = open(namaFile, 'rt')  
data = f.read()  
f.close()  
print(data)
```

2. Cek Adanya File *Text* Yang Dituju

Ketika penulisan data pada *text*, python akan secara otomatis menciptakan file tersebut jika tidak ada pada direktori program. Yang menjadi permasalahan adalah jika penulisan data ke *text* file yang sudah ada terlebih dahulu, maka file tersebut akan dihapus sebelum file yang sama diciptakan kembali. Untuk menghindari hal ini, maka python menyediakan fitur yang dapat mengecek terlebih dahulu adanya file tersebut atau tidak. Fitur tersebut terdapat dalam *package* OS yang dapat diimport ke dalam program. Perhatikan contoh dibawah yang menunjukkan cara kerjanya.

```
import os

namaFile = 'foo.txt'
if not os.path.exists(namaFile):
    print('File', namaFile, 'tidak ada')
else:
    print('File', namaFile, 'sudah ada')
```

Jika ingin menambahkan barisan data pada file *text* yang juga sudah memiliki data, maka syntax dibawah dapat digunakan sebagai salah satu caranya. Jalankan program tersebut lebih dari satu kali untuk melihat perubahan pada isi dari file *text*.

```
import os

namaFile = 'foo.txt'
if not os.path.exists(namaFile):
    print('File', namaFile, 'tidak ada')
    print('Membuatkan file text bernama', namaFile)
    with open(namaFile, 'wt') as f:
        print('Python dengan Pycharm', file=f)
        print('Sekolah Vokasi', file=f)
        print('Universitas Gadjah Mada', file=f)
    print('Selesai')
else:
    print('File', namaFile, 'sudah ada')
    print('Membaca isi dari file text bernama', namaFile)
    with open(namaFile, 'rt') as f:
        data = f.read()
    print('Menambahkan satu baris kalimat di bagian akhir')
    with open(namaFile, 'wt') as f:
        print(data, file=f, end='')
        print('Praktikum Pemrograman Lanjut', file=f)
    print('Selesai')
```

Library OS juga memiliki fitur untuk mengecek apakah sebuah *path* yang diberikan pada program merujuk ke sebuah file atau direktori. Fungsi yang dipakai adalah *isfile* dan *isdir*. Perhatikan contoh dibawah tentang cara penggunaannya.

```
import os

namaFile = 'test'
if os.path.isfile(namaFile):
    print(namaFile, 'adalah sebuah file')
else:
    print(namaFile, 'adalah sebuah', end='')

if os.path.isdir(namaFile):
    print(' folder')
else:
    print(' Invalid Directory or File')
```

3. Membuat dan Menghapus Folder dan File

Tidak jarang seorang programmer ingin menyimpan hasil proses program ke dalam folder yang terpisah dari program sebagai bagian dari dokumentasi yang baik. Hal ini bisa dilakukan secara otomatis pada Python. Fitur ini juga dimiliki oleh hampir semua bahasa pemrograman level tinggi lainnya. Perhatikan contoh -contoh dibawah yang menggambarkan cara membuat dan menghapus sebuah folder/file.

```
import os
folder = 'hasil'

if os.path.exists(folder) == False:
    print('Folder belum ada')
    print('Membuatkan folder bernama', folder)
    os.mkdir(folder)
else:
    print('Folder sudah ada')
```

Pada contoh diatas folder bernama *hasil* sebelumnya tidak ada pada folder dimana anda menyimpan program yang sedang anda jalankan. Dengan menjalankan program tersebut, anda akan mendapatkan sebuah folder baru yang kosong pada direktori yang sama dengan program Python yang anda jalankan. Perintah yang digunakan adalah *mkdir* bagian dari *package OS*.

```
import os
folder = 'hasil'
file = 'foo.txt'
# Menghapus folder
if os.path.exists(folder) == True:
    print('Folder <', folder, '> ditemukan.')
    print('Menghapus folder <', folder, '> ...')
    os.rmdir(folder)
    print('Selesai.')
else:
    print('Folder <', folder, '> tidak ditemukan')
```

Pada syntax dibawah folder *hasil* yang telah dibuat pada contoh sebelumnya akan dihapus dengan menggunakan perintah *rmdir* yang masih bagian dari *package OS*.

```
import os
folder = 'hasil'
file = 'foo.txt'
# Menghapus folder
if os.path.exists(folder) == True:
    print('Folder <', folder, '> ditemukan.')
    print('Menghapus folder <', folder, '> ...')
    os.rmdir(folder)
    print('Selesai.')
else:
    print('Folder <', folder, '> tidak ditemukan')
```

Untuk menghapus file yang ada pada direktori yang sama dengan program, dapat digunakan perintah *remove* yang juga bagian dari *package OS*.

```
import os
file = 'foo.txt'
# Menghapus file
if os.path.exists(file) == True:
    print('File <', file, '> ditemukan.')
    print('Menghapus file <', file, '> ...')
    os.remove(file)
    print('Selesai.')
else:
    print('File <', file, '> tidak ditemukan')
```

Jika folder yang akan dihapus bukan folder kosong, karena ada beberapa file di dalamnya, maka dapat digunakan perintah *rmtree* dari *package shutil*. Perhatikan syntax dibawah yang menjelaskan cara kerjanya.

Pada syntax berikutnya, pastikan bahwa anda sudah memiliki sebuah folder bernama *hasil* yang di dalamnya terdapat sebuah file *notepad* bernama *foo.txt*. Lalu jalankan program pada contoh tersebut dan perhatikan perbedaan antara sebelum dan sesudah program tersebut dijalankan.

```
import os
import shutil

folder = 'hasil'
# Menghapus folder
if os.path.exists(folder) == True:
    print('Folder <', folder, '> ditemukan.')
    print('Menghapus folder <', folder, '> ...')
    shutil.rmtree(folder, True)
    print('Selesai.')
else:
    print('Folder <', folder, '> tidak ditemukan')
```

4. Membuat Sub-Folder

Jika ingin membuat beberapa sub-folder di dalam folder utama, maka syntax dibawah dapat digunakan sebagai dasar. Terdapat penggunaan perulangan FOR pada program tersebut yang akan secara otomatis menghasilkan penamaan sub-folder dan *path* nya.

Sebelum menjalankan program tersebut, terlebih dahulu buatlah sebuah folder *parent* yang bernama Parent. Jika folder tersebut tidak dipersiapkan, maka akan timbul pesan *error*. Anda bisa saja menambahkan bagian program pada contoh dibawah agar terlebih dahulu menciptakan sebuah folder *parent* untuk diisi dengan sub-folder. Agar sub-folder dapat dibuat, maka terlebih dahulu *parent* dan *children* folder disatukan dalam tataran *symbolic path* dengan menggunakan perintah *join*.

```
import os

mainFolder = 'Parent'
cf = 'Child-'
for m in range(10):
    childFolder = cf + str(m)
    print('Membuat Sub-Folder', mainFolder, '\\', childFolder)
    newfolder = os.path.join(mainFolder, childFolder)
    os.mkdir(newfolder)
```

5. Mendapatkan Daftar File Dalam Folder

Seandainya dalam sebuah folder data terdapat banyak file – file terpisah yang akan diproses oleh program, maka program juga harus dilengkapi dengan pembacaan direktori file – file tersebut.

```
import os  
  
folder = 'Parent'  
lstfiles = os.listdir(folder)  
print(lstfiles)
```

Pada contoh diatas, disediakan program yang menjadi landasan untuk program yang dimaksud. Pada sistem operasi windows, file atau sub-folder selalu diawali dengan urutan/indeks ke – 3 karena urutan 1 dan 2 biasanya titik tunggal dan ganda (. dan ..). Namun, Python sudah secara otomatis hanya membaca nama sub folder atau file yang valid.

Dengan menggunakan perintah *listdir*, anda bisa mengakses seluruh file di dalam folder *parent*, bukan hanya sub folder tapi juga file lainnya. Hasil dari pembacaan tersebut disimpan dalam bentuk list, sehingga untuk membaca satu demi satu juga menggunakan prinsip list.

6. Bekerja Dengan File CSV

Selain dengan *text* biasa, Python juga memiliki kemampuan berkomunikasi dengan file CSV (*Comma Separated Value*). Biasanya CSV dapat dibaca dengan *software* Ms. Excel pada Windows.

i) Membaca isi csv

Misalkan ada sebuah file CSV bernama *stocks.csv* yang telah disertakan dengan modul praktikum ini. Untuk membaca isi dari file tersebut, Python sudah menyediakan *library* bernama *csv*. Perhatikan contoh dibawah yang menggambarkan cara kerjanya.

```
import csv

with open('stocks.csv') as f:
    f_csv = csv.reader(f)
    headers = next(f_csv)
    print('Judul Kolom')
    print(headers)
    print('\nIsi Tabel')
    for row in f_csv:
        print(row)
```

Jika diperhatikan maka tampak bahwa *header* dan *content* nya berbentuk *list* yang tentu saja dapat dioperasikan dengan aturan *list*. Agar dapat mengumpulkan isi tiap kolom dalam satu variabel, dapat menggunakan *dictionary* dengan terlebih dahulu mengubah csv menjadi *dictionary* dengan menggunakan perintah *DictReader* seperti diabawah.

```
import csv

with open('stocks.csv') as f:
    f_csv = csv.DictReader(f)
    headers = next(f_csv)
    data = []

or row in f_csv:
    data.append(row['value'])
print(data)
```

dengan cara itu, manipulasi matematis dapat dilakukan dengan mudah seperti menghitung total dari angka – angka yang berada pada kolom *value* seperti pada contoh berikut ini.

```
import csv

with open('stocks.csv') as f:
    f_csv = csv.DictReader(f)
    headers = next(f_csv)
    total = 0
    print('Total: ', end='')
    for row in f_csv:
        total = total + float(row['value'])
    print(total)
```

karena angka – angka dalam kolom tersebut belum bertipe *numeric* maka diubah dulu. Pada contoh tersebut saya mengubahnya menjadi bertipe data *float*. Anda bisa saja mengubah menjadi integer dengan perintah **int(data)**.

7. Membuat file csv

Untuk membuat file csv juga dapat dilakukan dengan mudah menggunakan *library* pada csv. Terlebih dahulu file csv yang lengkap dengan nama filenya diciptakan. Lalu *object writer* pada csv harus dibentuk menggunakan perintah *writer*. Perhatikan contoh 7.18 yang menjelaskan cara kerjanya.

Nama file csv adalah test.csv. *Headers* atau judul kolom dan konten/isi tabelnya dibuat terpisah dalam format *list*. Perintah *writerow* digunakan jika ingin menuliskan satu baris pada file csv sedangkan untuk baris yang lebih dari satu digunakan perintah *writerows*. Fungsi *newline= ''* pada program adalah untuk menghindari adanya baris kosong tiap baris.

```
import csv

headers = ['Symbol', 'Price', 'Date', 'Time', 'Change', 'Volume']
rows = [('AA', 39.48, '6/11/2007', '9:36am', -0.18, 181800),
        ('AIG', 71.38, '6/11/2007', '9:36am', -0.15, 195500),
        ('AXP', 62.58, '6/11/2007', '9:36am', -0.46, 935000),
        ]
with open('test.csv', 'wt', newline='') as f:
    f_csv = csv.writer(f)
    f_csv.writerow(headers)
    f_csv.writerows(rows)
```

D. TUGAS

Buatlah sebuah Aplikasi (dengan menggunakan Python) sederhana untuk mengelola kasir pada suatu Restoran. Aplikasi mempunyai mekanisme kerja sebagai berikut.

- a. Aplikasi meminta User untuk Login (*Username* dan *Password*) sebelum dapat mengoperasikan aplikasi. Aplikasi akan membaca isi file ‘**user.txt**’ yang teman-teman **buat sendiri** yang berisi username dan password. User ada pada baris pertama (Sekolah Vokasi) dan password (1234) pada pada baris kedua)
- b. Aplikasi akan menampilkan pesan *error* jika user dan/atau password tidak sesuai, dan akan meneruskan jika user dan password sesuai.
- c. Masukkan nominal harga makanan yang dipesan melalui instruksi input. (input akan terus tampil selama input tidak dimasukkan ‘0’ . berapapun jumlah pesanan, input akan terus muncul sampai diberikan ‘0’)
- d. Masukkan seluruh input kedalam sebuah list, yang akan menjadi parameter dari **fungsi summary**.

- e. Panggil **fungsi summary** dan return hasil summarynya.
- f. Masukkan input tanggal transaksi (hanya tanggalnya saja). Dan tampilkan diskon dari tanggal (jika ada) menggunakan **fungsi diskon akhir bulan / fungsi diskon pertengahan bulan.**
- g. Masukkan input apakah member atau tidak, Jika input ‘Y’, maka panggil **fungsi diskon member** dan tampilkan total akhirnya. Jika ‘N’, maka total akan tetap
- h. Masukkan input apakah transaksi pribadi (‘P’) / perusahaan (‘U’). Terapkan **fungsi pajak pribadi/ pajak perusahaan** sesuai dengan fungsi pada modul pajak.
- i. Kenakan pajak restoran JIKA total dari poin h melebihi Rp.500.000 dengan menggunakan **fungsi pajak restoran**
- j. Tuliskan total summary pada sebuah file .txt symbol ‘**transaksi.txt**’.

Masukan input apakah akan memasukkan transaksi berikutnya atau tidak. Jika input adalah ‘Y’ maka program akan Kembali ke poin c dan hasil transaksi akan Kembali disimpan pada baris berikutnya pada file ‘**transaksi.txt**’. Jika input adalah ‘N’ maka program akan Kembali ke poin a.

BAB VIII

OBJECT ORIENTED PROGRAMMING 1

CAPAIAN PEMBELAJARAN (*Learning Out Come*)

1. Mahasiswa memahami konsep OOP
2. Mahasiswa memahami komponen yang ada didalam OOP

A. PENDAHULUAN / DESKRIPSI SINGKAT

Object Oriented Programming (OOP) sering disebut sebagai paradigma pemrograman modern yang sebelumnya adalah gaya pemrograman prosedural. Pendekatan OOP akan sangat memudahkan jika diterapkan pada proyek yang besar dan kompleks, yang terdiri dari beberapa personel dan pengembang, dengan membagi proyek menjadi bagian-bagian kecil dan independent. Berikut merupakan beberapa perbedaan dari OOP dan prosedural. Konsep dari OOP dalam python ini berfokus dalam pembuatan kode yang reusable (dapat digunakan kembali). Konsep ini juga dikenal dengan DRY (Don't Repeat Yourself).

Prosedural	OOP
Didalam pemrograman prosedural, program dibagi ke beberapa bagian yang dinamakan functions .	Didalam OOP, program dibagi ke beberapa bagian yang dinamakan objects .
Pemrograman prosedural menggunakan pendekatan : top down approach .	OOP menggunakan pendekatan bottom up approach .
Tidak memiliki penentu hak akses	Memiliki penentu hak akses seperti : private, public, protected etc.
Tidak mudah menambahkan data dan fungsi baru.	Mudah menambahkan data dan fungsi baru.

Tidak memiliki cara untuk menyembunyikan data sehingga less secure .	Memiliki cara untuk menyembunyikan data sehingga more secure .
Tidak memungkinkan overloading	Memungkinkan overloading
Function lebih penting daripada data	Data lebih penting daripada function

Pondasi dasar dari struktur OOP adalah :

- *Class*
- *Object*
- *Properties (instance variable, class variables, attributes)*
- *Method*

Jika diibaratakan dalam kehidupan nyata, class adalah mobil, maka property nya adalah warna mobil, bentuk mobil, merk mobil dan lainnya. Method adalah aksi yang dapat dilakukan oleh mobil seperti maju, berhenti, belok dan lainnya. Dan object adalah mobil berwarna biru yang dihasilkan oleh class mobil.

B. CLASS

Class adalah sebuah blueprint untuk **object**.

Kalau kita berbicara mengenai parrot (burung beo), blueprint atau desainnya,

- Parrot tersebut akan memiliki nama, warna, ukuran, dll.
- Parrot juga nantinya bisa singing dan dancing.

Oleh karena itu kita bisa membuat parrot menjadi sebuah **class**. Untuk membuat **class** parrot yang sederhana kita bisa menuliskan kode berikut :

```
class Parrot
```

Kata kunci **class** diikuti dengan Parrot mendefinisikan blueprint dari **class** parrot.

Blueprint ini nantinya akan di realisasikan dalam sebuah **Object**.

Cobalah syntax dibawah untuk membuat class Bernama Parrot

```

class Parrot:
    species = "bird"

    def __init__(self, name, age):
        self.name = name
        self.age = age

# instantiate the Parrot class
papi = Parrot("Papi", 3)
greeny = Parrot("Greeny", 5)
# access the class attributes
print("Papi is a " + papi.species)
print("Greeny is also a " + greeny.species)
# access the instance attributes
print(papi.name + " is " + str(papi.age) + " years old")
print(greeny.name + " is " + str(greeny.age) + " years old")

```

Pada program diatas, kita telah membuat sebuah class bernama parrot. Sebagaimana yang dijelaskan sebelumnya bahwa class merupakan blueprint. Dengan kata lain kita telah membuat sebuah blueprint bernama parrot yang akan mencetak banyak parrot dengan nama yang berbeda. Perhatikan ketika papi dan greeny (yang merupakan objects) dibuat dari class Parrot. Hanya dengan memanggil classnya, dan kemudian mengisi identitas name dan age, maka dua parrot sudah terbentuk.

C. Objects

Sebelumnya kita membahas dulu apa itu objek. Jadi, objek itu memiliki dua karakteristik.

1. *attributes*
2. *behavior (method)*

Mari kita lihat contohnya:

- Parrot (burung beo) adalah sebuah object,
- name, age, color (nama, usia, warna) adalah atributnya singing, dancing (menyanyi, menari) adalah behavior nya

Konvensi OOP mengasumsikan bahwa setiap objek yang ada dapat dilengkapi dengan tiga kelompok atribut:

- sebuah objek memiliki **nama yang unik** mengidentifikasinya dalam namespace rumahnya (meskipun mungkin ada beberapa objek anonim juga)
- sebuah objek memiliki sekumpulan **properti individual** yang membuatnya orisinal, unik, atau luar biasa (meskipun mungkin beberapa objek mungkin tidak memiliki properti sama sekali)

- sebuah objek memiliki sekumpulan **kemampuan** untuk melakukan aktivitas tertentu, mampu mengubah objek itu sendiri, atau beberapa objek lainnya.

Setiap kali Anda mendeskripsikan suatu objek dan Anda menggunakan:

- sebuah kata benda - Anda mungkin mendefinisikan nama objek
- sebuah kata sifat - Anda mungkin mendefinisikan properti objek
- kata kerja - Anda mungkin mendefinisikan aktivitas objek.

Perhatikan contoh program sebelumnya. Papi dan greeny adalah objects yang dibuat oleh class parrot.

D. ATTRIBUTE

Atribut bukan merupakan fungsi biasa namun merupakan konstruktor. Fungsi konstruktor adalah mempersiapkan object baru, dengan menerima argument untuk men-set beberapa variabel yang akan terbentuk pada class. Konstruktor tidak memiliki return luaran. Object yang immutable atau tidak dapat dirubah umumnya diinisialisasi pada konstruktor. Konstruktor berisi parameter diri, yang akan nanti dibawa ke class turunannya.

Untuk membentuk sebuah konstruktor, digunakan instruksi `__init__`. Perhatikan contoh class Parrot. Name dan age adalah bagian dari atribut yang dituliskan dalam bentuk fungsi, yang akan ada pada setiap object yang dibentuk, yaitu pada contoh adalah papi dan greeny.

```
class Parrot:
    species = "bird"

    def __init__(self, name, age):
        self.name = name
        self.age = age

# instantiate the Parrot class
papi = Parrot("Papi", 3)
greeny = Parrot("Greeny", 5)
# access the class attributes
print("Papi is a " + papi.species)
print("Greeny is also a " + greeny.species)
# access the instance attributes
print(papi.name + " is " + str(papi.age) + " years old")
print(greeny.name + " is " + str(greeny.age) + " years old")
```

E. METHOD

Sama halnya seperti atribut, method dibangun sebagai sebuah fungsi. **Methods** adalah function yang didefinisikan dalam sebuah class. Function ini seharusnya mendefinisikan behavior dari objeknya. Perhatikan syntax dibawah :

```
class Parrot:

    # instance attributes
    def __init__(self, name, age):
        self.name = name
        self.age = age

    # instance method
    def sing(self, song):
        return "{} sings {}".format(self.name, song)

    def dance(self):
        return "{} is now dancing".format(self.name)

# instantiate the object
papi = Parrot("Papi", 10)

# call our instance methods
print(papi.sing("'Happy'"))
print(papi.dance())
```

Sing dan dance merupakan method dari kelas Parrot. Selayaknya behaviour, method umumnya berisi fungsi yang memiliki luaran atau return value, sehingga didalam fungsi umumnya ada komputasi yang dilakukan.

F. TUGAS

Buatlah sebuah class bernama Kardus yang memiliki atribut berupa panjang , lebar , tinggi .

Kemudian buat metode (method) yang dimiliki class tersebut sebagai berikut:

- volumeKardus(self) : panjang * lebar * tinggi
- luasPermKardus(self) : 2x(panjang*lebar+panjang*tinggi+tinggi*lebar)
- massaJenis(self,masa) : masa / volumeKardus()

Buatlah dua buah instance / object dari class Kardus berupa :

A. KotakBiru : dengan Panjang = 10, lebar = 8 dan tinggi = 4

B. KotakMerah : dengan Panjang = 15, lebar = 5 dan tinggi =1

Tampilkan volume , luas_permukaan dan massa_jenis kedua object tersebut.

BAB IX

OBJECT ORIENTED PROGRAMMING 2

CAPAIAN PEMBELAJARAN (*Learning Out Come*)

1. Mahasiswa memahami lebih dalam tentang properties method
2. Mahasiswa memahami build in function yang ada didalam python untuk class

A. `__init__`

Konstruktor atau pada bab sebelumnya juga disebut dengan atribut merupakan salah satu method. Fungsi konstruktor adalah mempersiapkan object baru, dengan menerima argument untuk men-set beberapa variabel yang akan terbentuk pada class. Konstruktor tidak memiliki return luaran. Object yang immutable atau tidak dapat dirubah umumnya diinisialisasi pada konstruktor. Konstruktor berisi parameter diri, yang akan nanti dibawa ke class turunannya. Untuk membentuk sebuah konstruktor, digunakan instruksi `__init__`. Cobalah instruksi dibawah ini dan pahami definisi konstruktor.

```
class cobaClass:  
    def __init__(self, argInput):  
        self.var = argInput  
  
objek_1 = cobaClass("objek")  
  
print(objek_1.var)
```

B. `__str__`

Method dengan menggunakan `__str__` akan mengeluarkan return dalam bentuk string. Jika luaran tidak dalam bentuk string maka object tidak akan terbentuk. Cobalah instruksi dibawah ini dan pahami.

```
class classSTR:  
    def __init__(self, value = None):  
        self.var = value  
  
    def __str__(self, value = None):  
        return self.var  
  
objek_1 = Classy("objek")  
objek_2 = Classy(5)  
  
print(objek_1)  
print(objek_2)
```

C. __dict__

Instruksi ini digunakan untuk memeriksa parameter yang ada pada class dan object. Output akan dalam bentuk dictionar yang berisi parameter yang dimiliki oleh object maupun class. Hal ini perlu dilakukan untuk mengetahui identitas dan komponen yang terbawa, namun tidak digunakan. Cobalah program dibawah dan pahami maksud dari instruksi ini :

```
class Classy:  
    varia = 1  
    def __init__(self):  
        self.var = 2  
  
    def method(self):  
        pass  
  
    def __hidden(self):  
        pass  
  
  
obj = Classy()  
  
print(obj.__dict__)  
print(Classy.__dict__)
```

D. __name__

Instruksi ini digunakan untuk mengetahui nama dari class dan juga tipe dari object. Perhatikan instruksi dibawah ini dan pahami :

```
class namaClass:  
    pass  
  
print(namaClass.__name__)  
objek = namaClass()  
print(type(objek).__name__)
```

E. __module__

Instruksi ini digunakan untuk memeriksa nama module pada class. Lakukan program dibawah dan pahami fungsi instruksinya :

```
class modClass:  
    pass  
  
print(modClass.__module__)  
objek = modClass()  
print(objek.__module__)
```

F. __bases__

Instruksi ini akan mengembalikan basis class dari class input. Perhatikan syntax dibawah dan pahami .

```
class SuperOne:  
    pass  
  
class SuperTwo:  
    pass  
  
class Sub(SuperOne, SuperTwo):  
    pass  
  
def printBases(cls):  
    print('(', end=' ')  
  
    for x in cls.__bases__:  
        print(x.__name__, end=' ')  
    print(')')  
  
printBases(SuperOne)  
printBases(SuperTwo)  
printBases(Sub)
```

G. `__call__`

Callable adalah sebuah keadaan dimana objectnya dapat dipanggil. Untuk membuat object menjadi callable maka dapat digunakan `__call__` ketika membuat method. Cobalah syntax dibawah dan pahami maksudnya.

```
class Human:
    def __init__(self, first_name, last_name):
        print("I am inside __init__ method")
        self.first_name = first_name
        self.last_name = last_name

    def __call__(cls):
        print("I am inside __call__ method")

human_obj = Human("Virat", "Kohli")
# Output: I am inside __init__ method

# Both human_obj() and human_obj.__call__() are equivalent
human_obj()
# Output: I am inside __call__ method

human_obj.__call__()
# Output: I am inside __call__ method

callable(human_obj)
# Output: True
```

BAB X

OOP – INHERITANCE 1

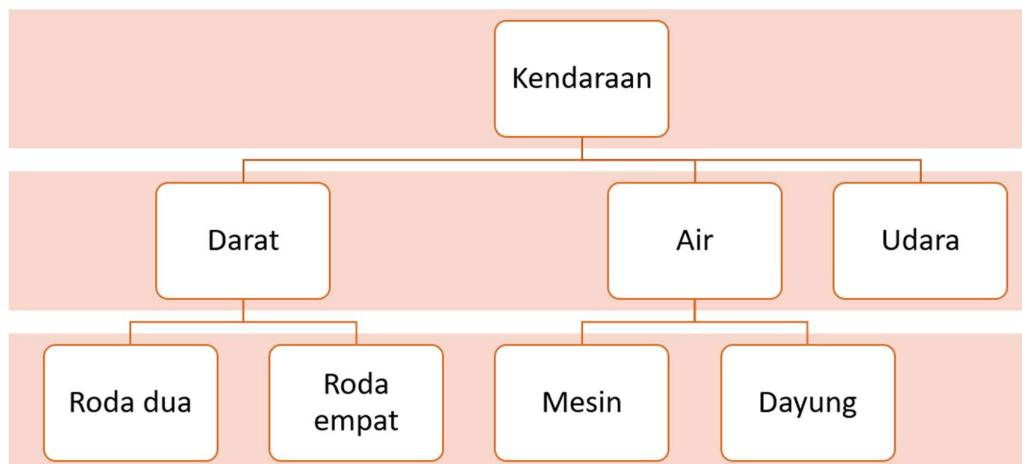
CAPAIAN PEMBELAJARAN (*Learning Out Come*)

1. Mahasiswa memahami cara mewariskan komponen class
2. Mahasiswa memahami aplikasi perwarisan (inheritance)

A. KONSEP DASAR PEWARISAN

Class sebagaimana yang telah diajarkan di bab sebelumnya merupakan sebuah cetakan atau blue print yang akan menghasilkan object melalui cetakan tersebut. Tidak hanya mencetak object, class juga mampu menurunkan sifat kepada class lainnya. Anggap saja anda akan membuat sebuah class baru yang memiliki atribut dan method yang sama, namun ada beberapa bagian yang ditambahkan.

Perhatikan Gambar dibawah untuk memahami ilustrasi terhadap inheritance :



Kita contohkan kendaraan. Kendaraan merupakan alat transportasi yang dapat berpindah dari satu tempat ke tempat lainnya. Dibagi berdasarkan media bergeraknya, kendaraan dapat berjalan di darat, air dan udara. Untuk kendaraan yang bergerak didarat, ada dua jenis kendaraan berdasarkan jumlah rodanya, yaitu roda dua dan roda empat.

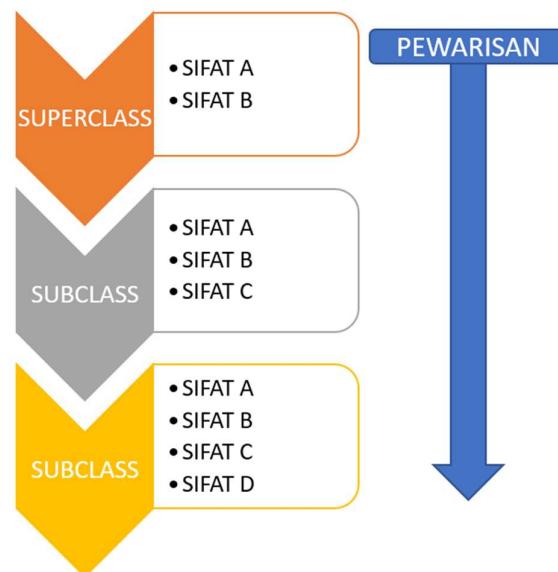
Anggaplah kendaraan merupakan sebuah kelas, dan darat, air, dan udara adalah subclass yang terbentuk dari class kendaraan (superclass). Semua subclass memiliki

sifat pewarisan (inheritance) dari superclass atau parentnya, yaitu alat yang mampu bergerak dari titik A ke titik B, namun bisa juga memiliki sifat lainnya yang baru. Pewarisan sifat turun dari atas ke bawah seperti akar pohon, bukan cabang. Superclass akan selalu diatas, dan subclass ada dibawahnya.

Perlu dipahami bahwa **Class** (di antara definisi lainnya) adalah sekumpulan **object** dan **Object** adalah makhluk yang termasuk dalam **class**. **Object** adalah penjelmaan dari persyaratan, sifat, dan kualitas yang ditetapkan ke class tertentu. **Class** membentuk hierarki. Ini mungkin berarti bahwa **object** yang termasuk dalam class tertentu menjadi milik semua **superclass** pada saat yang sama. Ini juga dapat berarti bahwa object apa pun yang termasuk dalam class super mungkin bukan milik **subclass**.

B. PERWARISAN SIFAT

Dua konsep utama dari perwarisan sifat adalah yang pertama, superclass (parent) bersifat lebih umum daripada subclass. Pewarisan merupakan salah satu konsep dasar dari pemrograman berbasis objek. Objek apapun yang terikat ke tingkat tertentu dari hierarki kelas akan mewarisi semua sifat dari superclass nya. Perhatikan Gambar dibawah :



Perhatikan contoh syntax dibawah :

```

class Hewan:
    def __init__(self,nama):
        self.nama = nama
    def gerak(self):
        print('gerak-gerak!!!')

class HewanDarat(Hewan):
    def __init__(self,nama,kaki=0):
        super().__init__(nama)
        self.kaki = kaki

    def gerak(self):
        print(self.nama,'gerak di darat dengan kaki',self.kaki)

class HewanAir(Hewan):
    def __init__(self,nama,sirip='kecil'):
        super().__init__(nama)
        self.sirip = sirip

    def gerak(self):
        print(self.nama,'gerak di air dengan sirip', self.sirip)

```

class Hewan merupakan superclass dan class HewanDarat dan HewanAir adalah subclass. Instruksi `super().__init__(nama)` merupakan instruksi yang akan menurunkan atribut superclass ke subclassnya. Setelah membuat class, akan dibuat object dari class dengan perintah berikut :

```

hewan = Hewan('pokoknya hewan')
kambing = HewanDarat('kambing',kaki=4)
hiu = HewanAir('hiu',sirip='lebar')

hewan.gerak()
kambing.gerak()
hiu.gerak()

```

Coba eksekusi perintah diatas dan perhatikan hasilnya. Kesimpulannya adalah superclass akan mewariskan seluruh sifat yang dimilikinya kepada subclassnya.

C. TUGAS

Buatlah sebuah simulasi turunan dengan contoh yang baru. Diawali dengan membuat class parent, kemudian dilanjutkan dengan dua subclass yang baru, yang merupakan turunan dari superclassnya. Kemudian buatlah object masing-masing dari ketiga class tersebut.

BAB XI

OOP – INHERITANCE 2

CAPAIAN PEMBELAJARAN (*Learning Out Come*)

1. Mahasiswa memahami membuat subclass dari beberapa superclass
2. Mahasiswa memahami bagaimana pewarisan properties ke subclass
3. Memahami memahami permasalahan diamond problem

A. MULTIPLE INHERITANCE

Python mendukung multiple inheritance yaitu sebuah subclass dibuat dari beberapa superclass. Namun jika kedua class memiliki variabel yang sama, maka variabel tersebut akan ter-replace dengan nilai yang baru. Analisa syntax berikut dan pahami :

```
class Satu:  
    var = "satu"  
    varSatu = "satusatu"  
    def coba(self):  
        return "SATU"  
  
class Dua:  
    var = "dua"  
    varDua = "duadua"  
    def coba(self):  
        return "DUA"  
  
class Sub(Dua, Satu):  
    pass  
  
obj = Sub()  
print(obj.var, obj.varSatu, obj.varDua, obj.coba())
```

B. MENCARI PROPERTIES DAN METHOD

Bagaimana Python mencari properties dan methods pada object? Ada beberapa cara yang dilakukan secara urutan, yaitu mencari dalam objek itu sendiri. Yang kedua adalah mencari pada classes yang terlibat dalam pembuatan object inheritance **dari bawah ke atas**. Jika terdapat satu atau lebih class dalam tingkatan jalur inheritance, maka Python akan mencari dari **kiri ke kanan**. Perhatikan syntax dibawah dan pahami bagaimana Python mencari propertiesnya. Cobalah syntax dibawah dan lakukan eksplorasi dengan mengganti superclass dan subclassnya :

```

class L1:
    var = 100
    def fun(self):
        return 101

class L2(L1):
    var = 200
    def fun(self):
        return 201

class L3(L2):
    pass

obj = L3()

print(obj.var, obj.fun())

```

C. issubclass() dan isinstance()

Perintah issubclass() digunakan untuk mengetahui apakah suatu class merupakan turunan dari class lainnya. Sedangkan isinstance() adalah fungsi untuk mengetahui apakah sebuah object merupakan instance dari satu class. Cobalah syntax dibawah dan lakukan eksplorasi terhadap syntax dan bandingkan hasilnya.

```

print(issubclass(Sub, Satu))
print(issubclass(Sub, Dua))
print(isinstance(obj, Satu))
print(isinstance(obj, Dua))

```

D. is Operator

is operator digunakan untuk memperlihatkan apakah dua variabel mengacu pada object yang sama. Coba syntax berikut dan eksplorasi hasilnya :

```

string_1 = "Mary had a little "
string_2 = "Mary had a little lamb"
string_1 += "lamb"

print(string_1 == string_2, string_1 is string_2)

```

```

class SampleClass:
    def __init__(self, val):
        self.val = val

object_1 = SampleClass(0)
object_2 = SampleClass(2)
object_3 = object_1
object_3.val += 1

print(object_1 is object_2)
print(object_2 is object_3)
print(object_3 is object_1)
print(object_1.val, object_2.val, object_3.val)

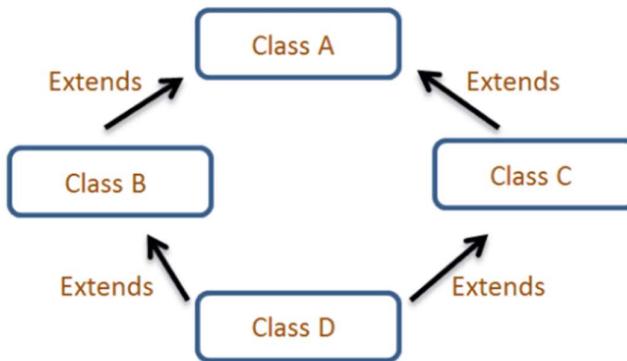
```

E. METHOD RESOLUTION ORDER (MRO)

Untuk memahami MRO, cobalah instruksi berikut ini. Lakukan eksplorasi dan jelaskan pengertian MRO dengan mengganti object pada tanda (#).

```
class Top:  
    def m_top(self):  
        print("top")  
  
class Middle(Top):  
    def m_middle(self):  
        print("middle")  
  
# MRO Problem  
class Bottom(Top, Middle):  
    def m_bottom(self):  
        print("bottom")  
  
object = Bottom()  
#object.m_bottom()  
#object.m_middle()  
#object.m_top()
```

F. DIAMOND PROBLEM



Permasalahan yang mungkin terjadi pada multiple inheritance adalah diamond problem. Gambar diatas merupakan ilustrasi dari diamond problem. Sebagaimana yang terlihat pada Gambar diatas, Class B dan Class C merupakan subclass dari Class D, sedangkan class A berasal dari Class B dan Class C. Untuk hal dengan ilustrasi diatas dapat dilakukan, namun jika arah sebaliknya, semisal Class A dibuat dari kombinasi Class D dan Class C akan menghasilkan error.

Tugas 1 : Cobalah syntax dibawah dan kemudian eksplorasi dengan menukar superclass dan subclassnya.

```
class Top:
    def m_top(self):
        print("top")

class Middle_Left(Top):
    def m_middle(self):
        print("middle_left")

class Middle_Right(Top):
    def m_middle(self):
        print("middle_right")

class Bottom(Middle_Left, Middle_Right):
    def m_bottom(self):
        print("bottom")

object = Bottom()
object.m_bottom()
object.m_middle()
object.m_top()
```

BAB XII

OOP – ENKAPSULASI

CAPAIAN PEMBELAJARAN (*Learning Out Come*)

1. Mahasiswa memahami penggunaan enkapsulasi

A. DEFINISI

Sebagaimana yang dijelaskan diawal bahwa OOP juga dapat membuat program lebih private. Enkapsulasi merupakan sebuah metode untuk menjaga beberapa variabel didalam class hanya digunakan didalam method nya saja namun tidak dapat diakses diluar, meskipun object telah dibuat, variabel yang bersifat private tidak dapat diakses. Cobalah syntax berikut dan eksekusi.

```
class Rekening:  
    def __init__(self):  
        self.__saldo = 0  
  
    def kredit(self, jumlah):  
        if jumlah < 0:  
            print("Gagal Kredit, Jumlah tidak bisa kurang dari 0")  
            return  
  
        self.__saldo += jumlah  
  
    def debit(self, jumlah):  
        if jumlah > self.__saldo:  
            print("Gagal Debit, Jumlah melebihi Saldo!")  
            return  
  
        if jumlah < 0:  
            print("Gagal Debit, Jumlah tidak bisa kurang dari 0")  
            return  
  
        self.__saldo -= jumlah  
  
    def cetakSaldo(self):  
        print("Saldo saat ini " +str(self.__saldo))  
  
bniBudi = Rekening()  
  
bniBudi.kredit(1000000)  
bniBudi.debit(100000)  
bniBudi.cetakSaldo()
```

Didalam class Rekening terdapat variable saldo yang ditulis dengan dua underscore didepannya (`__saldo`). Hal ini menandakan bahwa saldo merupakan variabel yang private. Saldo hanya diakses didalam class, meskipun object sudah dibuat. Namun

saldo tidak dapat dikeluarkan. Meskipun terlihat pada luaran ketika di eksekusi, namun nilai tersebut hanya di cetak, bukan return daripada class.

Coba lakukan dan perhatikan contoh dibawah :

```
class Siswa:

    __mapel = []
    __nilai = []

    def __init__(self, nama, alamat):
        self.nama=nama
        self.alamat=alamat

    def getNama(self):
        return self.nama

    def tambahNilaiMapel(self, mapel, nilai):
        self.__mapel.append(mapel)
        self.__nilai.append(nilai)

    def ambilNilai(self):
        print(self.__nilai)
        print(self.__mapel)

s = Siswa("Ratna", "Yogya")
print(s.nama)
print(s.alamat)
s.tambahNilaiMapel("Math",90)
s.tambahNilaiMapel("Bio",70)
print(s.getNama())
s.ambilNilai()
```

Tugas 1 : Jelaskan program diatas, apa saja variabel yang di private

Tugas 2 : Buatlah sebuah contoh class enkapsulasi