

**MODUL MATA KULIAH**

# **BAHASA PEMROGRAMAN DASAR**

**PG168 – 3 SKS**



**FAKULTAS TEKNOLOGI INFORMASI  
UNIVERSITAS BUDI LUHUR**

**JAKARTA  
SEPTEMBER 2021**

**TIM Penyusun**

Agus Umar Hamdani, M.Kom  
Tri Ika Jaya Kusumawati, M.Kom<sup>1</sup>



## MODUL PERKULIAHAN #10 MODULE DAN PACKAGE

Capaian Pembelajaran	:	<b>Mahasiswa Mampu:</b> <ol style="list-style-type: none"><li>1. Memahami konsep dasar module, package dan library yang ada pada bahasa pemrograman Python.</li><li>2. Memahami penggunaan module, package dan library yang ada pada bahasa pemrograman Python</li></ol>
Sub Pokok Bahasan	:	<ol style="list-style-type: none"><li>1. <i>Module</i></li><li>2. <i>Package</i></li><li>3. <i>Library</i></li></ol>
Daftar Pustaka	:	<ol style="list-style-type: none"><li>1. Zarman, Wendi dan Wicaksono, Mochamad Fajar. <i>"Implementasi Algoritma dalam bahasa Python"</i>. Edisi Pertama. Bandung : Penerbit Informatika. 2020.</li><li>2. Kurniawati, Arik. <i>"Algoritma dan Pemrograman menggunakan Python"</i>. Edisi Pertama. Yogyakarta : Depublish. 2016.</li><li>3. Ismah. <i>"Pemrograman Komputer Dasar-dasar Python"</i>. Jakarta : Fakultas Ilmu Pendidikan Universitas Muhammadiyah Jakarta. 20110.</li><li>4. Irfani, M. Haviz dan Dafid. <i>"Modul Praktikum Dasar Pemrograman dengan bahasa Python"</i>. Palembang : Sekolah Tinggi Manajemen Informatika Global Informatika Multidata Palembang. 2016.</li><li>5. Fikri, Rijalul. <i>"Praktikum Algoritma dan Pemrograman Komputer"</i>. Surabaya : Program Studi Teknik Komputer dan Telematika Jurusan Teknik Elektro Fakultas Teknologi Industri Institut Teknologi Sepuluh Nopember. 2010.</li><li>6. Wiratmaja, I Gede Harjumawan, et.all. 2021. Program Menghitung Banyak Bata pada Ruangan Menggunakan Bahasa Python. <i>TIERS Information Technology Journal</i>. Vol. 2(1). Undiknas.</li><li>7. Nuraini, Rini. 20110. Desk Check Table Pada Flowchart Operasi Perkalian Matriks. <i>Jurnal Petir</i>. Vol. 10(1). Sekolah Tinggi Teknik – PLN (STT-PLN).</li><li>8. Romzi, Muhammad dan Kurniawan, Budi. 2020. Pembelajaran Pemrograman Python Dengan Pendekatan Logika Algoritma. <i>JTIM : Jurnal Teknik Informatika Mahakarya</i>. Vol. 03(2). Hal. 37-44.</li><li>9. Programiz.com. Python Operators (<a href="https://www.programiz.com/python-programming/operators">https://www.programiz.com/python-programming/operators</a>) diakses pada 29 September 2021 pukul 21.32 WIB)</li></ol>

## PRAKTIKUM 10

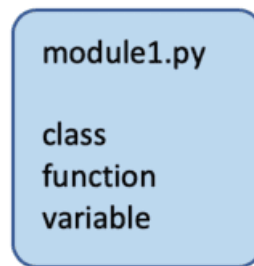
### MODULE DAN PACKAGE

#### 10.1 Teori Singkat

Kita telah mempelajari dan mengetahui bahwa fungsi (*Function*) berguna untuk membuat suatu kode program yang dapat digunakan secara berulang-ulang kali. Namun, apabila program yang dibuat sangat kompleks, maka membutuhkan banyak kode program (subprogram). Kode-kode program yang dapat dibuat dalam file-file terpisah yang disebut dengan modul (*module*). Sekumpulan file-file atau modul-modul yang didalamnya terdapat fungsi yang digunakan untuk membuat suatu program disebut dengan paket (*packages*). Gabungan dari sekumpulan *package* dan *module* dengan fungsionalitas yang sama dengan tujuan untuk memudahkan kalian dalam membuat suatu aplikasi, tanpa harus menulis banyak kode disebut dengan *Library*. Kita bisa membuat modul-modul program (*Modules*) dan paket-paket program (*Packages*) dan menggabungkannya menjadi sebuah *library* untuk menyederhanakan masalah yang kompleks sehingga menjadi lebih sederhana.

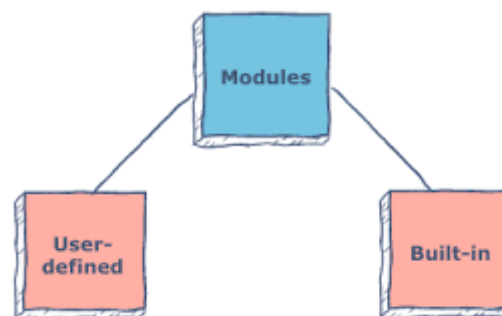
#### 10.2 MODUL (*MODULE*)

Module pada Python adalah sebuah file yang berisikan sekumpulan kode fungsi, *class* dan variabel yang disimpan dalam satu file berekstensi *.py* dan dapat dieksekusi oleh interpreter Python. Nama dari module *.py* merupakan nama nama dari file itu sendiri. Misalkan kita memiliki file bernama "dqlab.py", maka kita telah membuat sebuah module bernama "dqlab". Dan module sendiri bisa memiliki berbagai macam isi, baik itu fungsi, *class*, maupun variabel. Module digunakan untuk memecah sebuah program besar menjadi file yang lebih kecil agar lebih mudah dimanage dan diorganisir. Modul membuat kode bersifat *reusable*, artinya satu module bisa dipakai berulang kali dimana saja diperlukan. Struktur modul dapat dilihat pada gambar 10.1.



Gambar 10.1 Struktur Modul (*Module*)

Modul yang tersedia dan siap digunakan yang merupakan bawaan dari Python disebut dengan *Built-In Modules*. Python menyediakan beberapa modul di *Python's Standard Library* dan sudah terinstal bersamaan pada saat instalasi Python. Modul lainnya dapat diinstal melalui *Python's Package Manager PIP*, seperti *matplotlib*. Selain itu, kita juga membuat modul Python untuk kebutuhan sendiri (*User-Defined Modules*).



Gambar 10.2 Jenis-jenis Modul (*Modules*)

Fungsi **help()** berfungsi untuk menampilkan bantuan atau keterangan dari objek. Fungsi **help()** berfungsi di mode interaktif Python. Fungsi **help()** memiliki sintaks sebagai berikut:

`Help(objects)`

Fungsi **help()** memiliki satu parameter, yaitu:

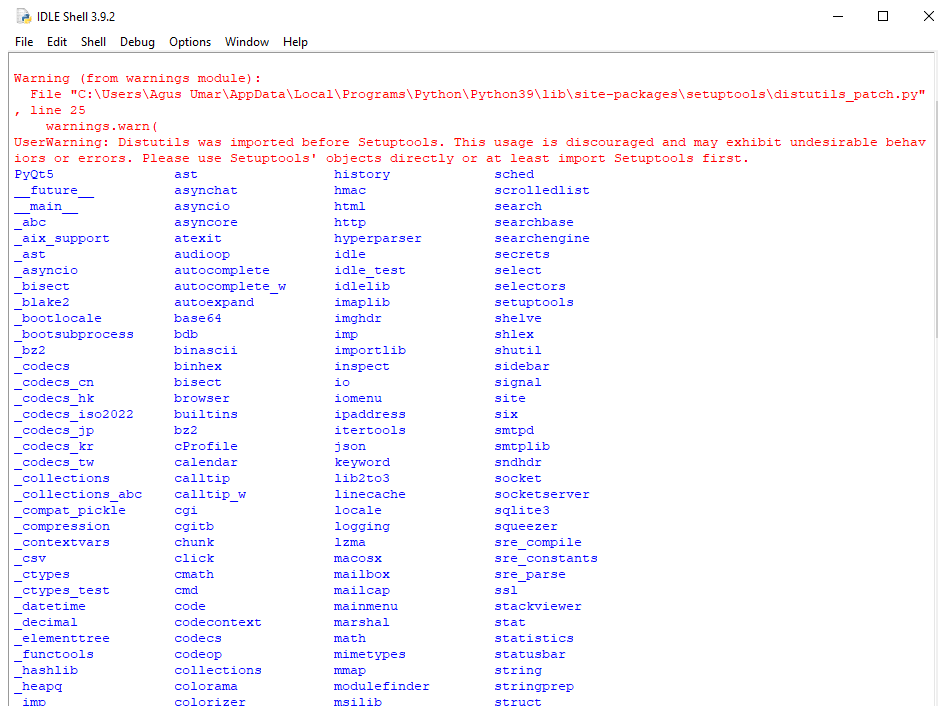
- Object – objek yang akan ditampilkan keterangan atau bantuannya.

Untuk menampilkan semua modul yang tersedia dalam Python, maka kita dapat menggunakan fungsi **help()** berikut pada prompt IDLE Shell Python.

```
>>> help('modules')
```

Gambar 10.3 Perintah Untuk Menampilkan Daftar Modul

Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 10.4.



```
IDLE Shell 3.9.2
File Edit Shell Debug Options Window Help

Warning (from warnings module):
  File "C:\Users\Agus Umar\AppData\Local\Programs\Python\Python39\lib\site-packages\setuptools\distutils_patch.py", line 25
    warnings.warn(
UserWarning: Distutils was imported before Setuptools. This usage is discouraged and may exhibit undesirable behaviors or errors. Please use Setuptools' objects directly or at least import Setuptools first.

PyQt5          ast          history        sched
__future__     asynchat       hmac           scrolledlist
__main__       asyncio        html           search
__abc__        asyncoore      http           searchbase
__aix_support atexit         hyperparser    searchengine
__ast__        audioop        idle           secrets
__asyncio     autocomplete   idle_test     select
__bisect__    autocomplete_w idllib         selectors
__blake2__    autoexpand     imaplib        setuptools
__bootlocale  base64         imghdr         shelve
__bootsubprocess bdb            imp            shlex
__bz2__       binascii       importlib      shutil
__codecs__    binhex         inspect        sidebar
__codecs_cn   bisect         io             signal
__codecs_hk   browser        iomenu         site
__codecs_iso2022 builtins       ipaddress     six
__codecs_jp   bz2            itertools     smtpd
__codecs_kr   cProfile       json           smtplib
__codecs_tw   calendar       keyword        sndhdr
__collections collections    lib2to3        socket
__collections_abc calltip        linecache     socketserver
__compat_pickle cgi             locale         sqlite3
__compression cgitb          logging        squizeer
__contextvars chunk           lzma           sre_compile
__csv__       click          macosx         sre_constants
__ctypes__    cmath          mailbox        sre_parse
__ctypes_test cmd            mailcap        ssl
__datetime__ code           mainmenu      stackviewer
__decimal__  codecontext    marshal        stat
__elementtree codecs         math           statistics
__functools  codeop         mimetypes      statusbar
__hashlib__  collections    mmap           string
__heapq__    colorama       modulefinder   stringprep
__imp__      colorizer      msilib         struct
```

Gambar 10.4 Daftar *Built-In Modules* di Python

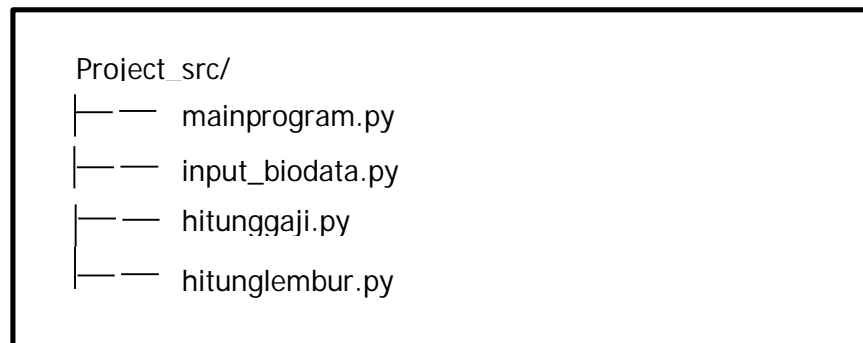
Dalam pemrograman dengan pendekatan modular, setiap program akan dipecah menjadi beberapa bagian atau kecil yang terpisah, terorganisir, dikelola dengan lebih rapi sesuai dengan tugasnya masing-masing.

Ada beberapa keuntungan jika kita menerapkan pendekatan modular, di antaranya:

- Simplicity*: kode program kita menjadi lebih sederhana.
- Maintainability*: kode program kita menjadi lebih mudah di-maintain atau "dipelihara".
- Reusability*: potongan-potongan kode program bisa digunakan dari berbagai tempat membuatnya menjadi lebih *reusable*.

### Studi Kasus 10.1

Gambar 10.5 merupakan contoh pembuatan modul program untuk merekam data pegawai dengan nama "mainprogram.py" yang berisi prosedur "input\_biodata\_md1.py" dan fungsi "hitung\_lembur.py" dan "hitung\_gaji.py" dengan struktur sebagai berikut :

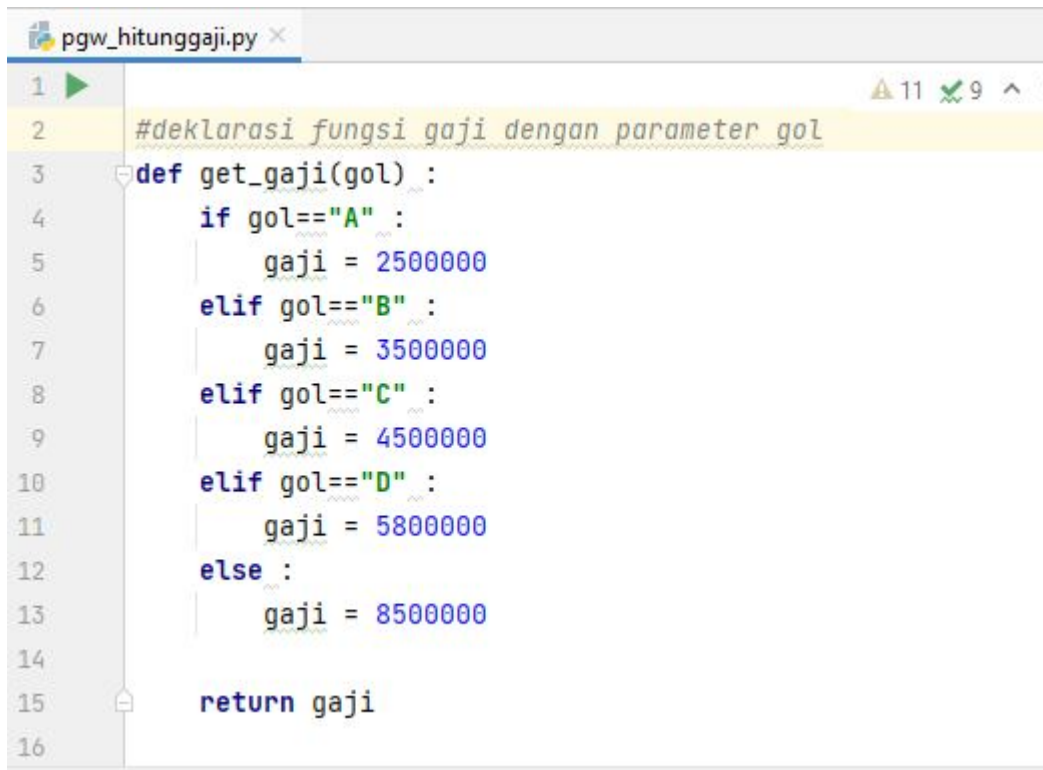


Gambar 10.5 Contoh Struktur Module

Keterangan :

- File "**mainprogram.py**" merupakan modul utama / file utama yang akan dirunning.
- File "**input\_biodata.py**" merupakan prosedur yang digunakan untuk menginput dan mencetak biodata pegawai.
- File "**hitunggaji.py**" merupakan fungsi yang akan digunakan untuk menghitung gaji pegawai.
- File "**hitunglembur.py**" merupakan fungsi yang akan digunakan untuk menghitung honor lembur pegawai.

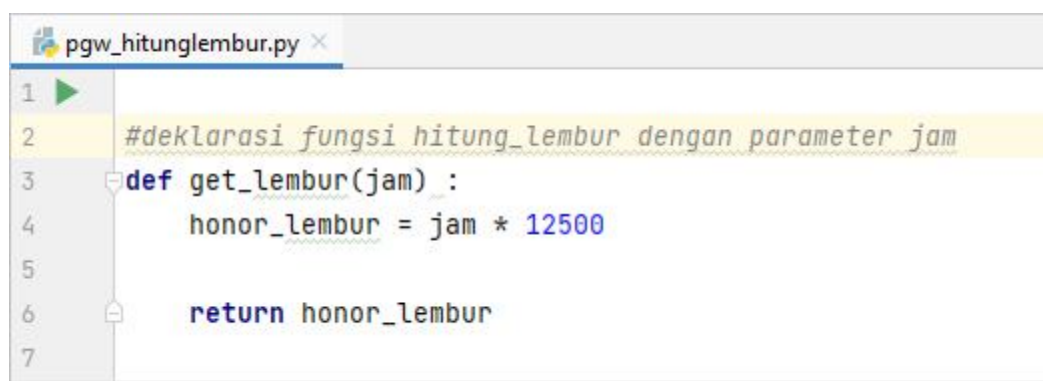
Perhatikan gambar 10.6, terdapat deklarasi fungsi dengan nama "**get\_gaji**" dengan parameter berupa variabel gol. Terdapat kondisi percabangan IF untuk menentukan gaji yang didapatkan oleh setiap pegawai. Fungsi tersebut mengembalikan nilai dari gaji yang didapatkan.



```
1  pgw_hitunggaji.py x
2 #deklarasi fungsi gaji dengan parameter gol
3 def get_gaji(gol) :
4     if gol=="A" :
5         gaji = 2500000
6     elif gol=="B" :
7         gaji = 3500000
8     elif gol=="C" :
9         gaji = 4500000
10    elif gol=="D" :
11        gaji = 5800000
12    else :
13        gaji = 8500000
14
15    return gaji
16
```

Gambar 10.6 Contoh Pembuatan Fungsi get\_gaji()

Perhatikan gambar 10.7, terdapat deklarasi fungsi / modul dengan nama **"get\_lembur"** dengan parameter berupa variabel jumlah lembur. Terdapat kondisi percabangan IF untuk menentukan honor lembur yang didapatkan oleh setiap pegawai. Fungsi tersebut mengembalikan nilai dari honor lembur yang didapatkan.



```
1  pgw_hitunglembur.py x
2 #deklarasi fungsi hitung_lembur dengan parameter jam
3 def get_lembur(jam) :
4     honor_lembur = jam * 12500
5
6     return honor_lembur
7
```

Gambar 10.7 Contoh Pembuatan Fungsi get\_lembur()

Perhatikan gambar 10.8, terdapat deklarasi file / modul dengan nama **"input\_biodata"** yang berisi prosedur **"isi\_biodata"** yang digunakan untuk



merekam biodata pegawai dan mencetak informasi mengenai gaji dan honor lembur yang diterima.

```
pgw_input_biodata.py
1 #memanggil function
2 import pgw_hitunggaji
3 import pgw_hitunglembur
4
5 #variabel global
6 global id, nama, golongan, jml_lembur
7
8 def isi_biodata():
9     print("\nModul Utama Pegawai")
10    print("-----")
11    print("ID Pegawai : ", id)
12    print("Nama Pegawai : ", nama)
13    print("Golongan kerja: ", golongan)
14    print("Jumlah Lembur : ", jml_lembur)
15    #memanggil fungsi get_gaji()
16    gj = pgw_hitunggaji.get_gaji(golongan)
17    print("Gaji Pegawai: ", gj)
18    #memanggil fungsi hitung_lembur()
19    lb = pgw_hitunglembur.get_lembur(jml_lembur)
20    print("Honor Lembur :", lb)
21
22    id = input("Masukkan ID Pegawai : ")
23    nama = input("Masukkan Nama Pegawai : ")
24    golongan = input("Masukkan Golongan Kerja : ")
25    jml_lembur = int(input("Masukkan Jumlah Lembur : "))
```

Gambar 10.8 Contoh Pembuatan Prosedur input\_biodata()

Perhatikan gambar 10.9, terdapat deklarasi file / modul dengan nama “mainprogram.py” yang mengimpor file / modul “pgw\_input\_biodata.py” dan memanggil prosedur “isi\_biodata()”.

```
pgw_mainprogram.py
1 #memanggil prosedur pgw_biodata_module
2 import pgw_input_biodata
3
4
5 #membuat modul utama
6 pgw_input_biodata.isi_biodata()
7
```

Gambar 10.9 Contoh Pembuatan Module mainprogram()



Untuk menjalankan program, maka anda harus berada pada modul “**mainprogram.py**”, karena memang modul ini yang digunakan sebagai modul utama. Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 10.10.

```
Masukkan ID Pegawai : 122
Masukkan Nama Pegawai : Bagus Sajiwo
Masukkan Golongan Kerja : B
Masukkan Jumlah Lembur : 5

Modul Utama Pegawai
-----
ID Pegawai : 122
Nama Pegawai : Bagus Sajiwo
Golongan kerja: B
Jumlah Lembur : 5
Gaji Pegawai: 3500000
Honor Lembur : 62500
```

Gambar 10.10 Hasil Keluaran Program

### 10.3 *Defined-User Module*

Berikut ini beberapa modul bawaan (*Built-In Module*) yang sering digunakan dalam pemrograman Python:

#### a. **Modul Matematika (*Math Module*)**

Beberapa fungsi matematika yang paling populer didefinisikan dalam modul matematika. Ini termasuk fungsi trigonometri, fungsi representasi, fungsi logaritma, fungsi konversi sudut, dll. Selain itu, dua konstanta matematika juga didefinisikan dalam modul ini. Selengkapnya modul matematika dapat dilihat pada tabel 10.1.

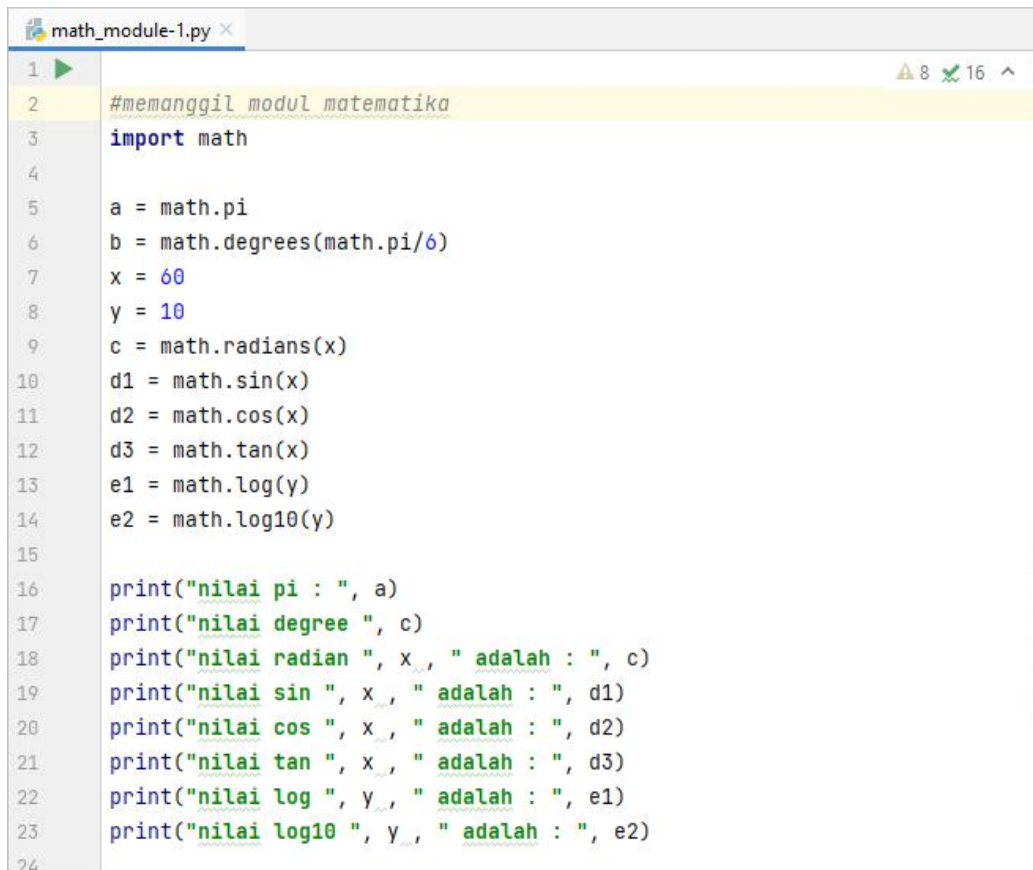
Tabel 10.1 Daftar Modul Matematika

Nama Modul Matematika ( <i>Math Module</i> )	Deskripsi	Sintak Penulisan
Pi	Pi adalah konstanta matematika, yang didefinisikan sebagai rasio keliling dengan diameter lingkaran dan nilainya adalah 3.141592653589793.	Math.pi
radian()	Untuk mengkonversi sebuah obyek ke dalam nilai radian.	Math.radians(object)
degrees()	Untuk mengkonversi sebuah obyek ke dalam nilai derajat (degree).	Math.degrees(object)
sin()	Untuk mengkonversi sebuah obyek ke dalam nilai sine.	Math.sin(object)
cos()	Untuk mengkonversi sebuah obyek ke dalam nilai cosine.	Math.cos(object)
tan()	Untuk mengkonversi sebuah obyek ke dalam nilai tangent.	Math.tan(object)
log()	Untuk mengembalikan logaritma dari angka tertentu. Logaritma dihitung ke basis e.	Math.log(object)
log10()	Untuk mengembalikan logaritma basis-10 dari angka tertentu. Ini disebut sebagai logaritma standar.	Math.log10(object)
exp()	Untuk mengembalikan angka float setelah dinaikkan e ke pangkat dari angka yang diberikan. Dengan kata lain, $\exp(x)$ memberikan $e^{**x}$ .	Math.exp(object)
pow()	Menerima dua argument bertipe Float, menaikkan nilai yang	Math.pow(object1, object2)

	pertama ke nilai yang kedua dan mengembalikan hasilnya. Contoh : $\text{pow}(4,4)$ sama dengan $4^{**}4$ .	
<code>sqrt()</code>	Untuk mengembalikan akar kuadrat dari angka yang diberikan.	<code>Math.sqrt(object)</code>
<code>ceil()</code>	Fungsi ini mendekati angka yang diberikan ke bilangan bulat terkecil, lebih besar atau sama dengan angka floating point yang diberikan.	<code>Math.ceil(object)</code>
<code>floor()</code>	Fungsi ini mengembalikan bilangan bulat terbesar yang kurang dari atau sama dengan angka yang diberikan.	<code>Math.floor(object)</code>

### Studi Kasus 10.1

Program pada gambar 10.10 merupakan contoh penggunaan modul matematika dalam program dengan memanggil modul-modul Python, yaitu : `pi`, `degrees()`, `radian()`, `sin()`, `cos()`, `tan()`, `log()` dan `log10()`.



```
1   math_module-1.py x
2 #memanggil modul matematika
3 import math
4
5 a = math.pi
6 b = math.degrees(math.pi/6)
7 x = 60
8 y = 10
9 c = math.radians(x)
10 d1 = math.sin(x)
11 d2 = math.cos(x)
12 d3 = math.tan(x)
13 e1 = math.log(y)
14 e2 = math.log10(y)
15
16 print("nilai pi : ", a)
17 print("nilai degree ", c)
18 print("nilai radian ", x, " adalah : ", c)
19 print("nilai sin ", x, " adalah : ", d1)
20 print("nilai cos ", x, " adalah : ", d2)
21 print("nilai tan ", x, " adalah : ", d3)
22 print("nilai log ", y, " adalah : ", e1)
23 print("nilai log10 ", y, " adalah : ", e2)
24
```

Gambar 10.10 Contoh Penggunaan Modul Matematika (1)

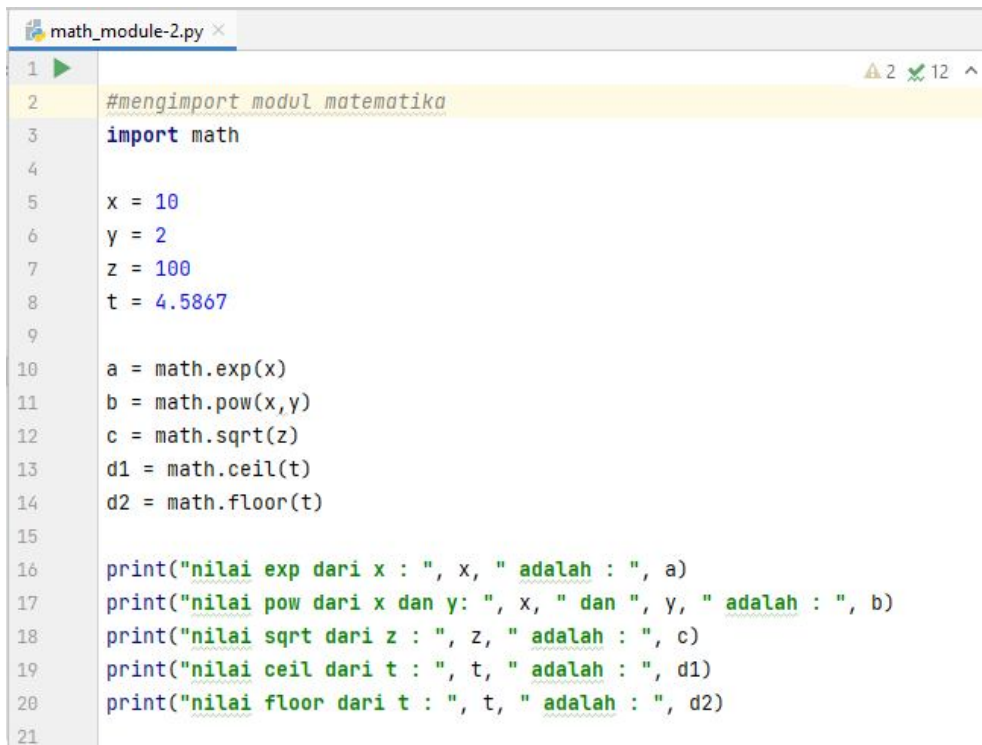
Perhatikan gambar 10.10, terdapat variabel x dan y untuk merekam nilai yang akan diolah ke dalam modul-modul matematika. Variabel x berisi angka 60 dan variabel y berisi angka 10. Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 10.11.

```
nilai pi : 3.141592653589793
nilai degree 1.0471975511965976
nilai radian 60 adalah : 1.0471975511965976
nilai sin 60 adalah : -0.3048106211022167
nilai cos 60 adalah : -0.9524129804151563
nilai tan 60 adalah : 0.320040389379563
nilai log 10 adalah : 2.302585092994046
nilai log10 10 adalah : 1.0
```

Gambar 10.11 Hasil Keluaran Program

## Studi Kasus 10.2

Program pada gambar 10.12 merupakan contoh penggunaan modul matematika (*Math Modules*) dalam program dengan memanggil modul-modul Python, yaitu : `exp()`, `pow()`, `sqrt()`, `ceil` dan `floor()`.



```
1   math_module-2.py
2 #mengimport modul matematika
3 import math
4
5 x = 10
6 y = 2
7 z = 100
8 t = 4.5867
9
10 a = math.exp(x)
11 b = math.pow(x,y)
12 c = math.sqrt(z)
13 d1 = math.ceil(t)
14 d2 = math.floor(t)
15
16 print("nilai exp dari x : ", x, " adalah : ", a)
17 print("nilai pow dari x dan y: ", x, " dan ", y, " adalah : ", b)
18 print("nilai sqrt dari z : ", z, " adalah : ", c)
19 print("nilai ceil dari t : ", t, " adalah : ", d1)
20 print("nilai floor dari t : ", t, " adalah : ", d2)
21
```

Gambar 10.12 Contoh Penggunaan Modul Matematika (2)

Perhatikan gambar 10.12, terdapat variabel `x`, `y`, `z` dan `t` untuk merekam nilai yang akan diolah ke dalam modul-modul matematika. Variabel `x` berisi angka 10, variabel `y` diberi angka 2, variabel `z` diberi nilai 100 dan variabel `t` berisi angka 4.5867. Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 10.13.

```
nilai exp dari x : 10  adalah : 22026.465794806718
nilai pow dari x dan y: 10  dan 2  adalah : 100.0
nilai sqrt dari z : 100  adalah : 10.0
nilai ceil dari t : 4.5867  adalah : 5
nilai floor dari t : 4.5867  adalah : 4
```

Gambar 10.13 Hasil Keluaran Program

### b. Modul Statistik (*Statistics Module*)

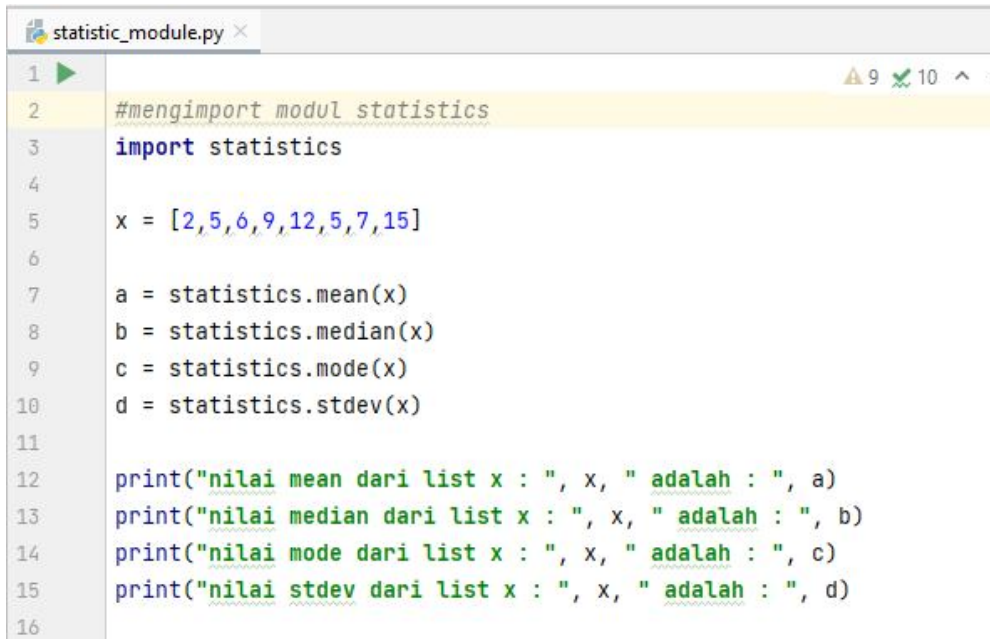
Modul statistik menyediakan fungsi untuk statistik matematis dari data numerik. Fungsi statistik populer berikut didefinisikan dalam modul ini. Selengkapnya modul statistik dapat dilihat pada tabel 10.2.

Tabel 10.2 Daftar Modul Statistik

Nama Modul Statistik ( <i>Statistic Module</i> )	Deskripsi	Sintak Penulisan
mean()	Untuk menghitung nilai mean dari angka-angka di dalam daftar (list).	Statistics.mean(object)
median(object)	Untuk menghitung nilai tengah (mean) dari angka-angka di dalam daftar (list).	Statistics.median(object)
mode(object)	Untuk mengembalikan nilai yang paling umum di dalam daftar (list).	Statistics.mode(object)
stdev(object)	Untuk menghitung nilai simpangan baku pada sampel yang diberikan di dalam daftar (list).	Statistics.stdev(object)

### Studi Kasus 10.3

Program pada gambar 10.14 merupakan contoh penggunaan modul statistika (*Statistics Modules*) dalam program dengan memanggil modul-modul Python, yaitu : mean(), median(), mode() dan stdev().



```
1 2
2  #mengimport modul statistics
3  import statistics
4
5  x = [2,5,6,9,12,5,7,15]
6
7  a = statistics.mean(x)
8  b = statistics.median(x)
9  c = statistics.mode(x)
10 d = statistics.stdev(x)
11
12 print("nilai mean dari list x : ", x, " adalah : ", a)
13 print("nilai median dari list x : ", x, " adalah : ", b)
14 print("nilai mode dari list x : ", x, " adalah : ", c)
15 print("nilai stdev dari list x : ", x, " adalah : ", d)
16
```

Gambar 10.14 Contoh Penggunaan Modul Statistik

Perhatikan gambar 10.14, terdapat variabel x bertipe List yang sudah berisi nilai yang akan diolah ke dalam modul-modul statistik. Variabel x berisi nilai [2,5,6,9,,12,5,7,15]. Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 10.15.

```
nilai mean dari list x : [2, 5, 6, 9, 12, 5, 7, 15] adalah : 7.625
nilai median dari list x : [2, 5, 6, 9, 12, 5, 7, 15] adalah : 6.5
nilai mode dari list x : [2, 5, 6, 9, 12, 5, 7, 15] adalah : 5
nilai stdev dari list x : [2, 5, 6, 9, 12, 5, 7, 15] adalah : 4.2067123233504535
```

Gambar 10.15 Hasil Keluaran Program

### c. Modul Koleksi (*Collection Module*)

<https://www.tutorialsteacher.com/python/collections-module>

Modul Koleksi (*Collections Module*) menyediakan alternatif untuk tipe data koleksi atau array, seperti List, Tuple, dan Dictionary. Selengkapnya modul koleksi dapat dilihat pada tabel 10.3.

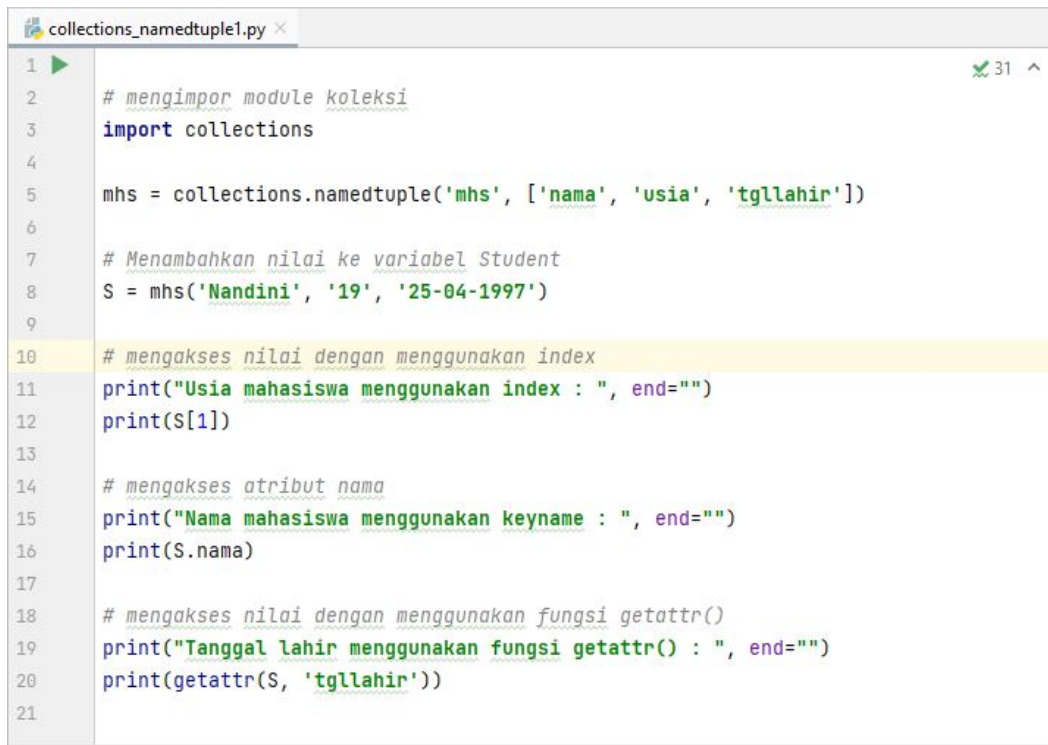
Tabel 10.3 Daftar Modul Koleksi



Nama Modul Koleksi ( <i>Collections Module</i> )	Deskripsi	Sintak Penulisan
namedtuple()	Untuk mengembalikan obyek, seperti : Tuple dengan nama field. Atribut field ini dapat diakses dengan pencarian atau indeks.	Collections.namedtuple (type_name, field-list)
OrderedDict()	Mirip seperti Dictionary di Python. Namun, ia merekam urutan kunci di mana pertama kali mereka dimasukkan.	Collections.OrderDict()
deque()	Obyek deque() mendukung penambahan dan pemunculan dari kedua ujung nilai dari daftar / list. Ini lebih menghemat memori daripada obyek List. Dalam obyek List, penghapusan item dapat menyebabkan semua item ke kanan digeser ke kiri oleh satu indeks. Oleh karena itu, prosesnya sangat lambat.	Collections.deque()

#### Studi Kasus 10.4

Program pada gambar 10.16 merupakan contoh penggunaan modul koleksi (*Collections Module*) dalam program dengan memanggil modul **NamedTuple** dan fungsi `getattr()`.



```

1 # mengimpor module koleksi
2 import collections
3
4 mhs = collections.namedtuple('mhs', ['nama', 'usia', 'tgllahir'])
5
6 # Menambahkan nilai ke variabel Student
7 S = mhs('Nandini', '19', '25-04-1997')
8
9
10 # mengakses nilai dengan menggunakan index
11 print("Usia mahasiswa menggunakan index : ", end="")
12 print(S[1])
13
14 # mengakses atribut nama
15 print("Nama mahasiswa menggunakan keyname : ", end="")
16 print(S.nama)
17
18 # mengakses nilai dengan menggunakan fungsi getattr()
19 print("Tanggal lahir menggunakan fungsi getattr() : ", end="")
20 print(getattr(S, 'tgllahir'))
21

```

Gambar 10.17 Contoh Penggunaan Modul NamedTuple

Perhatikan gambar 10.17, terdapat variabel `mhs` bertipe `Tuple` dengan menggunakan modul **`namedtuple`**. Kemudian variabel `S` memiliki tipe `mhs` dan sudah berisi nilai. Kemudian tiga cara untuk mengakses nilai dari modul `namedtuple` yaitu menggunakan ***index***, ***keyname*** dan ***fungsi getattr()***. Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 10.18.

```

Usia mahasiswa menggunakan index : 19
Nama mahasiswa menggunakan keyname : Nandini
Tanggal lahir menggunakan fungsi getattr() : 25-04-1997

```

Gambar 10.18 Hasil Keluaran Program

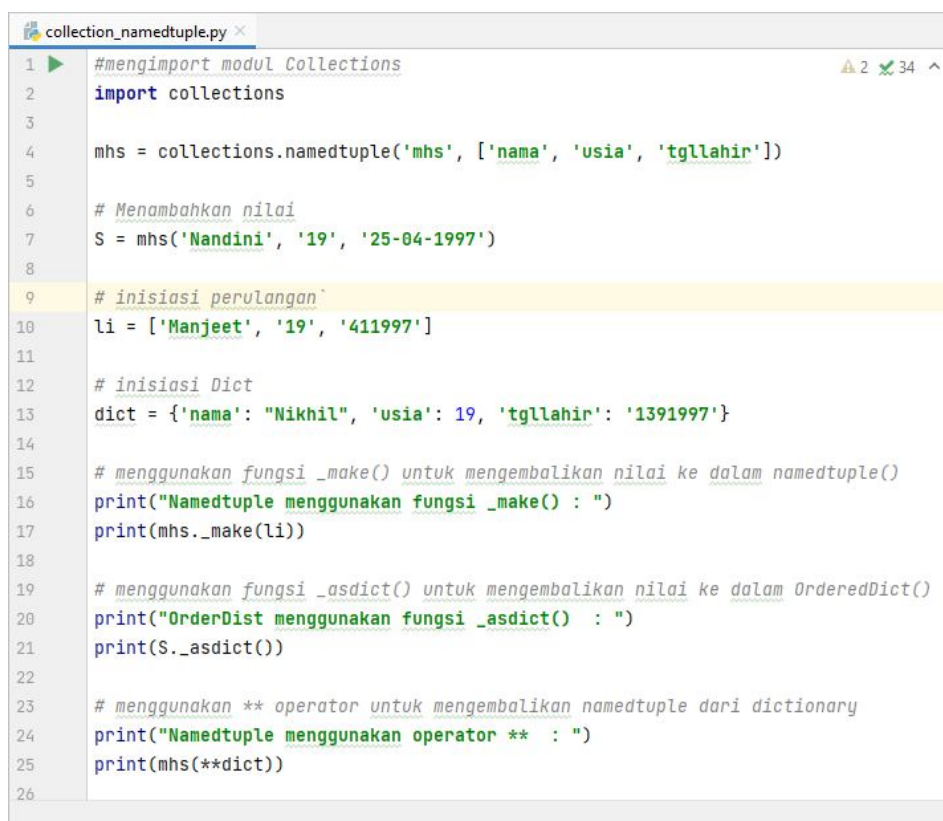
Beberapa fungsi populer mengubah koleksi lain seperti daftar, kamus, dan tuple menjadi tuple bernama dan mengembalikan informasi tuple bernama. Berikut ini adalah fungsi yang mengubah koleksi lain menjadi `Namedtuple`:

- a. **`_make()`** digunakan untuk mengkonversi obyek seperti List dan Tuple menjadi `namedtuple`.

- b. **\_asdict()** digunakan untuk membuat OrderDist dari namedtuple dan mengembalikan nilainya.
- c. **\*\* Operator** digunakan untuk mengkonversi Dictionary menjadi namedtuple.

### Studi Kasus 10.5

Program pada gambar 10.19 merupakan contoh penggunaan modul koleksi (*Collections Module*) dalam program dengan memanggil modul **Namedtuple** dengan fungsi `_make()`, `_asdict` dan `**` operator.



```
1 #mengimport modul Collections
2 import collections
3
4 mhs = collections.namedtuple('mhs', ['nama', 'usia', 'tgllahir'])
5
6 # Menambahkan nilai
7 S = mhs('Nandini', '19', '25-04-1997')
8
9 # inisiasi perulangan
10 li = ['Manjeet', '19', '411997']
11
12 # inisiasi Dict
13 dict = {'nama': "Nikhil", 'usia': 19, 'tgllahir': '1391997'}
14
15 # menggunakan fungsi _make() untuk mengembalikan nilai ke dalam namedtuple()
16 print("Namedtuple menggunakan fungsi _make() : ")
17 print(mhs._make(li))
18
19 # menggunakan fungsi _asdict() untuk mengembalikan nilai ke dalam OrderedDict()
20 print("OrderDist menggunakan fungsi _asdict() : ")
21 print(S._asdict())
22
23 # menggunakan ** operator untuk mengembalikan namedtuple dari dictionary
24 print("Namedtuple menggunakan operator ** : ")
25 print(mhs(**dict))
26
```

Gambar 10.19 Contoh Penggunaan Modul Nametuple() dengan fungsi `_make()`, `_asdict()` dan `**` Operator

Perhatikan gambar 10.19, terdapat variabel `mhs` bertipe Tuple dengan menggunakan modul **namedtuple**. Kemudian variabel `S` memiliki tipe `mhs` dan sudah berisi nilai. Kemudian terdapat variabel `li` bertipe List dan `dict` bertipe Dictionary. Kemudian tiga variabel tersebut dipanggil menggunakan fungsi **`_make()`**, **`_asdict`** dan **operator `**`**. Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 10.20.

```

Namedtuple menggunakan fungsi _make() :
mhs(nama='Manjeet', usia='19', tgllahir='411997')
OrderDist menggunakan fungsi _asdict() :
{'nama': 'Nandini', 'usia': '19', 'tgllahir': '25-04-1997'}
Namedtuple menggunakan operator ** :
mhs(nama='Nikhil', usia=19, tgllahir='1391997')

```

Gambar 10.21 Hasil Keluaran Program

### Studi Kasus 10.6

Program pada gambar 10.22 merupakan contoh penggunaan modul koleksi (*Collections Module*) dalam program dengan memanggil modul **OrderedDict**.



```

1  #collections_module.py
2  #mengimport modul Collections
3  import collections
4
5  d1 = collections.OrderedDict()
6  d1['A'] = 65
7  d1['C'] = 67
8  d1['B'] = 66
9  d1['D'] = 68
10
11  for k,v in d1.items():
12      print("key ke-", k, " : ",v)
13

```

Gambar 10.22 Contoh Penggunaan Modul Collections.OrderDict

Perhatikan gambar 10.23, terdapat variabel d1 digunakan untuk menyimpan nilai yang bertipe Dictionary dengan menggunakan modul OrderedDict. Kemudian variabel d1 tersebut diisi dengan nilai disertai dengan penomoran index atau nomor urut untuk setiap item datanya, kemudian pasangan (*pairs*) akan muncul sesuai urutan penyisipannya. Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 10.24.

```
key ke- A : 65
key ke- C : 67
key ke- B : 66
key ke- D : 68
```

Gambar 10.24 Hasil Keluaran Program

#### d. Modul Acak (*Random Module*)

Modul acak (*Random Module*) adalah modul bawaan untuk menghasilkan variabel *pseudo-random*. Modul ini dapat digunakan untuk melakukan tindakan secara acak, seperti untuk mendapatkan nomor acak, memilih elemen acak dari daftar / list, mengacak elemen secara acak, dll. Selengkapnya modul acak dapat dilihat pada tabel 10.4.

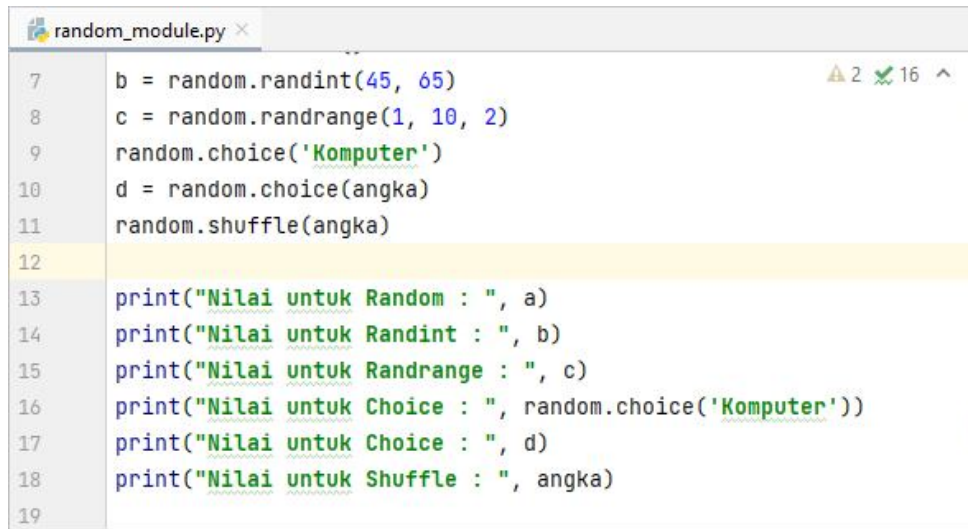
Tabel 10.4 Daftar Modul Acak

Nama Modul Acak ( <i>Random Module</i> )	Deskripsi	Sintak Penulisan
random()	Untuk mengembalikan angka float acak antara 0,0 hingga 1,0. Fungsi tidak memerlukan argumen apa pun.	random()
Randint()	mengembalikan bilangan bulat acak di antara bilangan bulat yang ditentukan.	randint(object)
Randrange()	Mengembalikan elemen yang dipilih secara acak dari rentang yang dibuat oleh argumen start, stop, dan step. Nilai awal adalah 0 secara default. Demikian pula, nilai langkah adalah 1 secara default.	randrange(object)
Choice()	Mengembalikan elemen yang dipilih secara acak dari urutan yang tidak kosong. Urutan kosong sebagai argumen memunculkan IndexError.	choice(object)

Shuffle()	Secara acak menyusun ulang elemen yang ada di dalam daftar / list.	<code>shuffle(object)</code>
-----------	--	------------------------------

### Studi Kasus 10.7

Program pada gambar 10.25 merupakan contoh penggunaan modul koleksi (*Collections Module*) dalam program dengan memanggil modul **Random**.



```

7   b = random.randint(45, 65)
8   c = random.randrange(1, 10, 2)
9   random.choice('Komputer')
10  d = random.choice(angka)
11  random.shuffle(angka)
12
13  print("Nilai untuk Random : ", a)
14  print("Nilai untuk Randint : ", b)
15  print("Nilai untuk Randrange : ", c)
16  print("Nilai untuk Choice : ", random.choice('Komputer'))
17  print("Nilai untuk Choice : ", d)
18  print("Nilai untuk Shuffle : ", angka)
19

```

Gambar 10.25 Contoh Penggunaan Modul Random

Perhatikan gambar 10.25, terdapat variabel `angka` digunakan untuk menyimpan nilai yang bertipe List. Terdapat deklarasi variabel `a` untuk memanggil modul **random()**, variabel `b` untuk memanggil modul **randint()**, variabel `c` untuk memanggil modul **randrange()**, variabel `d` untuk memanggil modul **choice()** berdasarkan nilai yang tersebut dalam perintah **random.choice('Komputer')** dan variabel `e` untuk memanggil modul **shuffle()**. Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 10.26.

```

Nilai untuk Random : 0.9934873958866896
Nilai untuk Randint : 51
Nilai untuk Randrange : 9
Nilai untuk Choice : o
Nilai untuk Choice : 56
Nilai untuk Shuffle : [42, 14, 35, 7, 63, 70, 28, 49, 56, 21]

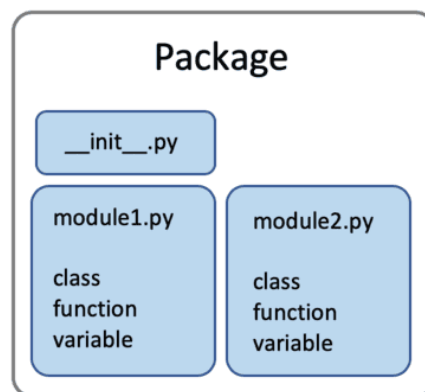
```

Gambar 10.27 Hasil Keluaran Program

## 10.4 PAKET (*PACKAGE*)

Paket pada Python adalah sekumpulan modul Python yang berada dalam sebuah folder, serta memiliki satu modul constructor (`__init__.py`). Paket ini merupakan sebuah cara untuk mengelola dan mengorganisir modul-modul Python dalam bentuk direktori, memungkinkan sebuah module untuk diakses menggunakan "namespace" dan dot lokasi. File constructor berfungsi untuk memberi tahu Interpreter Python bahwa folder tersebut adalah sebuah *package*. Jadi, setiap direktori atau folder yang berisi module constructor `__init__.py` akan diperlakukan sebagai *package*.

Secara sederhana gambaran dari paket (*package*) yang ada di Python dapat dilihat pada gambar 10.28.



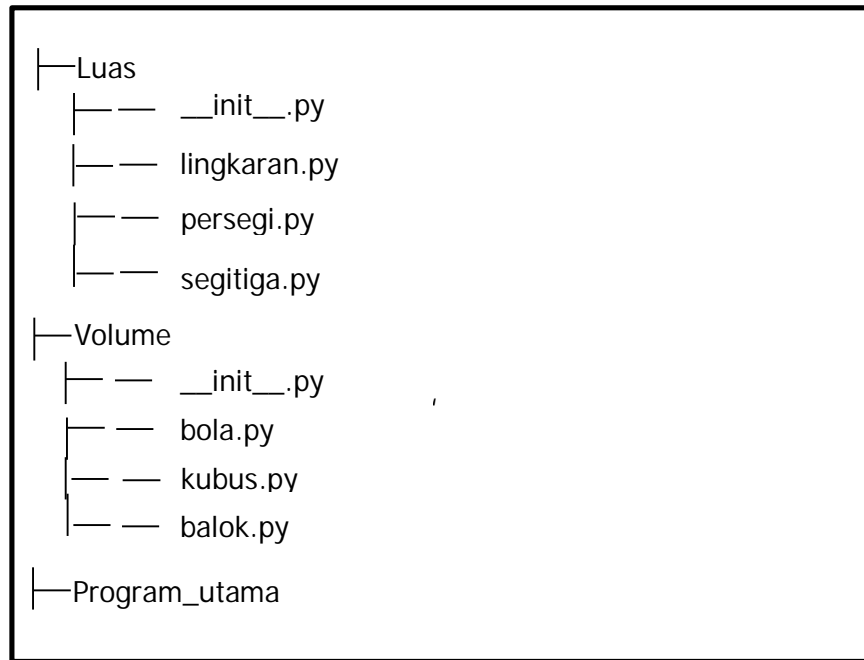
Gambar 10.28 Contoh Struktur Paket (*packages*)

### Studi Kasus 10.8

<https://jagongoding.com/python/menengah/modul-dan-paket/>

Gambar 10.29 merupakan contoh pembuatan 2 (dua) paket program yang terdiri atas 6 (enam) modul program. Kita akan membuat setiap bangun datar atau bangun ruang menjadi 1 (satu) modul tersendiri. Adapun dengan struktur paket tersebut sebagai berikut :





Gambar 10.29 Contoh Struktur Paket (*Package*)

Perhatikan gambar 10.30, terdapat deklarasi paket dengan nama **"luas.py"**. Paket **"luas.py"** terdiri atas 3 (tiga) fungsi, yaitu : **"luas\_lingkaran()"**, **"luas\_persegi()"**.

```
1  # deklarasi fungsi luas
2
3  def luas_lingkaran (radius):
4      return 22 / 7 * radius * radius
5
6  def luas_persegi (sisi):
7      return sisi * sisi
8
9  def luas_segitiga(alas, tinggi):
10     return 1/2 * alas * tinggi
```

The screenshot shows a code editor with a line number margin on the left (1 to 10) and a toolbar on the right. The code defines three functions for calculating area: 'luas\_lingkaran' (circle area), 'luas\_persegi' (square area), and 'luas\_segitiga' (triangle area). Each function is preceded by a comment indicating its purpose.

Gambar 10.30 Deklarasi Paket Luas.py

Perhatikan gambar 10.31, terdapat deklarasi paket dengan nama **"volume.py"**. Paket **"volume.py"** terdiri atas 3 (tiga) fungsi, yaitu : **volume\_bola()**, **volume\_kubus()** dan **volume\_balok()**.

```
volume.py x
1 ▶
2 def volume_bola (jarijari):
3     return 4 / 3 * 3.14 * jarijari**3
4
5 def volume_kubus(sisi):
6     return sisi * sisi * sisi
7
8 def volume_balok(panjang, lebar, tinggi):
9     return panjang * lebar * tinggi
10
```

Gambar 10.31 Deklarasi Paket Volume.py

Perhatikan gambar 10.32, terdapat deklarasi paket dengan nama **"volume.py"**.

```
program_utama.py x
1 ▶ #memanggil paket
2 import luas
3 import volume
4
5 a = int(input("Input radius : "))
6 b = int(input("Masukkan sisi : "))
7
8 #memanggil modul program yang ada di paket
9 ls = luas.luas_lingkaran(a)
10 vm = volume.volume_kubus(b)
11
12 print("Luas Lingkaran : ", ls)
13 print("Volume Kubus : ", vm)
14
```

Gambar 10.29 Modul Utama yang Memanggil Paket

## 10.5 LIBRARY

Python adalah bahasa pemrograman yang mempunyai banyak library. Library digunakan untuk membantu dalam mengolah atau mengerjakan suatu pekerjaan atau tugas tertentu. Library pada Python merupakan gabungan dari sekumpulan *package* dan *module* dengan fungsionalitas yang sama dengan tujuan untuk memudahkan kita dalam membuat suatu aplikasi, tanpa harus menulis

banyak kode. Library pada Python merupakan sebutan untuk kode program tambahan yang digunakan dalam kebutuhan tertentu. Python mempunyai lebih dari 140.000 library yang dikembangkan melalui *open source project*. Library juga bersifat *reusable* yang berarti bisa digunakan berkali - kali, dimana saja dan kapan saja. Berikut ini beberapa contoh library yang umum digunakan oleh praktisi data, yaitu:

#### a. Pandas

Pandas merupakan library yang disediakan oleh Python yang digunakan untuk memproses data yang meliputi pembersihan data, manipulasi data hingga analisis data. Pandas memudahkan pekerjaan memproses data menjadi lebih mudah dan terstruktur. Dengan Pandas kita juga dapat melakukan proses seperti pada SQL seperti agregasi, join, group by, dan lain-lain. Format file yang dapat dibaca oleh Pandas adalah CSV, TSV, dan TXT. Format penulisan ketika akan menggunakan Pandas pada Python yaitu **import pandas as pd**. Python akan memproses ini sebagai perintah untuk memanggil library Pandas. Sebutan pd ini umum dipakai saat menggunakan library Pandas.

Gambar 10.30 merupakan contoh penggunaan library Matplotlib Python pada data mahasiswa yang mempunyai atribut nama,alamat,jenis kelamin,uts,uas

```
nama,alamat,jenis kelamin,uts,uas
Faqih,Bandung,Laki-Laki,100,70
Ina,Jakarta,Perempuan,88,90
Fitri,Bandung,Perempuan,99,80
Dana,Surabaya,Perempuan,80,70
Abi,Surabaya,Laki-Laki,90,50
Dika,Jakarta,Laki-Laki,70,100
```

Gambar 10.30 Data Nilai Mahasiswa

Dari data diatas, kita ingin membuat Categorical Data menggunakan Library Pandas, maka kode programnya adalah sebagai berikut :

```

1 from sklearn.preprocessing import LabelEncoder
2
3 labelencoder = LabelEncoder()
4 df['alamat'] = labelencoder.fit_transform(df['alamat'])
5 df['jenis kelamin'] = labelencoder.fit_transform(df['jenis kelamin'])

```

Gambar 10.xx

Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 10.xx.

	nama	alamat	jenis kelamin	uts	uas
0	Faqih	0	0	100	70
1	Ina	1	1	88	90
2	Fitri	0	1	99	80
3	Dana	2	1	80	70
4	Abi	2	0	90	50
5	Dika	1	0	70	100

Gambar 10.31 Hasil Keluaran Program

## b. Numpy

Numerical Python atau biasa disebut dengan Numpy merupakan library Python yang digunakan untuk melakukan komputasi data yang bertipe numerik. Numpy bisa memproses operasi vektor, matriks, dan juga operasi matematika atau statistik. Beberapa tipe data yang ada dalam Numpy yaitu boolean, integer, unsigned integer, dan float. Format penulisan dalam Python untuk memanggil library Numpy adalah **import numpy as np**. Penggunaan sebutan np umum digunakan ketika menggunakan Numpy. Contoh operasi sederhana yang bisa dilakukan dengan Numpy yaitu penjumlahan, pengurangan, perkalian, dan pembagian. Simbol yang digunakan juga sama dengan standar yang digunakan untuk operasi tersebut yaitu (+) untuk penjumlahan, (-) untuk pengurangan, (\*) untuk perkalian, dan (/) untuk pembagian. Operasi lain seperti pangkat bisa dituliskan dengan dua bintang (\*\*). Numpy juga menyediakan fungsi universal function (*unfunc*) untuk menjalankan operasi seperti sin dan cos.

Gambar 10.32 merupakan contoh penggunaan Library Numpy.

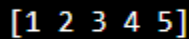
```
import numpy as np

arr = np.array((1, 2, 3, 4, 5))

print(arr)
```

Gambar 10.32 Contoh Penggunaan Library Numpy

Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 10.33.

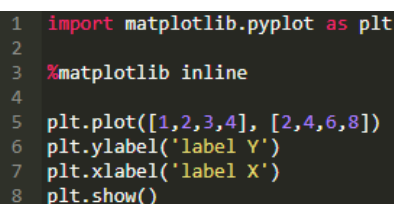
The image shows the output of the Python program, which is the array [1 2 3 4 5] displayed in a black box with yellow text.

Gambar 10.33 Hasil Keluaran Program

### c. Matplotlib

Matplotlib merupakan library pada Python yang digunakan untuk melakukan visualisasi data menjadi lebih rapi dan menarik. Sebagai seorang praktisi data mungkin terbiasa melihat data berbentuk tabel yang sangat banyak. Namun bagi orang lain tentunya akan membingungkan. Maka seorang praktisi data perlu memvisualisasikannya. Dengan Matplotlib data dapat divisualisasikan dalam bentuk 2D atau 3D sesuai dengan kebutuhan. Biasanya bentuk visualisasi dengan Matplotlib berupa grafik yang memiliki satu sumbu atau lebih. Ukuran dan warna grafik juga bisa diatur sesuai keinginan agar data tersaji dengan menarik dan memperoleh informasi yang berguna bagi perusahaan. Cara penulisannya di Python yaitu **import matplotlib.pyplot as plt**. Sebutan **plt** merupakan singkatan umum yang dipakai untuk menyebut matplotlib.

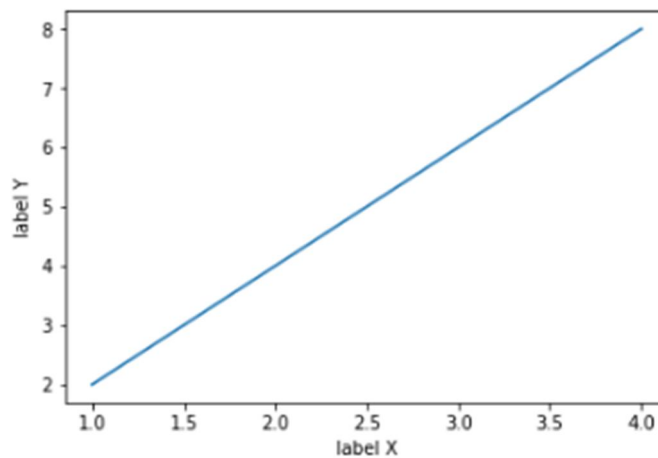
Gambar 10.34 merupakan contoh memanggil library matplotlib Python.

The image shows a snippet of Python code for using Matplotlib. It includes importing the library, setting inline mode, creating a plot with two data series, labeling the axes, and showing the plot.

```
1 import matplotlib.pyplot as plt
2
3 %matplotlib inline
4
5 plt.plot([1,2,3,4], [2,4,6,8])
6 plt.ylabel('label Y')
7 plt.xlabel('label X')
8 plt.show()
```

Gambar 10.34 Memanggil Library Matplotlib

Adapun hasil keluaran dari program diatas ditampilkan dapat dilihat pada gambar 10.35.



Gambar 10.35 Hasil Keluaran Program

## 10.6 Praktikum

### Langkah-langkah Praktikum

1. Buka Editor Python (IDLE / Pycharm / VSCode).
2. Buatlah file baru dengan membuka menu File > New > Source File atau dengan shortcut Ctrl + N.
3. Tulislah kode program berikut ini :

### Program 10.1 : Praktikum10-1. Py

1. Buatlah modul program untuk menampilkan informasi pelanggan, barang dan transaksi dengan struktur sebagai berikut :

```
|— project_src
|— — pelanggan.py
|— — barang.py
|— — transaksi.py
```

Gambar 10.36 Contoh Struktur Modul Program

- Input dan mencetak informasi pelanggan pada file **pelanggan.py** yang terdiri atas : kode pelanggan, nama pelanggan, alamat, nomor hp dan email.
- Input dan mencetak informasi barang pada file **barang.py** yang terdiri atas : kode barang, nama barang, satuan, harga satuan dan stok.
- Input dan mencetak informasi barang pada file **transaksi.py** yang terdiri atas : nomor transaksi, tanggal transaksi, kode pelanggan, nama pelanggan, kode barang, nama barang, harga satuan, jumlah pesanan dan total harga.
- Total harga dapat dicari dengan rumus harga satuan dikali dengan jumlah pesanan.

Gambar 10.37 ini adalah kode program untuk membuat modul "barang.py".



```

1  ► #definisi prosedur input_barang
2
3  global kdbrg, nmbrg, satuan, harga
4  def input_barang():
5      print("\nData Barang")
6      print("-----")
7      print("Kode barang : ", kode)
8      print("Nama barang : ", nama)
9      print("Satuan : ", satuan)
10     print("Harga satuan : ", harga)
11
12     kode = input("Masukkan Kode Barang : ")
13     nama = input("Masukkan Nama Barang : ")
14     satuan = input("Masukkan Satuan : ")
15     harga = input("Masukkan Harga Satuan : ")
16

```

Gambar 10.37 Membuat Modul barang.py

Gambar 10.38 ini adalah kode program untuk membuat modul "pelanggan.py".



```
praktikum1_pelanggan.py x
1 #membuat modul input_pelanggan
2 global kdplg, nmplg, alamat, nohp
3 def input_pelanggan():
4     print("\nData Pelanggan")
5     print("-----")
6     print("Kode pelanggan : ", kode)
7     print("Nama pelanggan : ", nama)
8     print("Alamat : ", alamat)
9     print("Nomor HP : ", nohp)
10
11
12     kode = input("Masukkan Kode Pelanggan : ")
13     nama = input("Masukkan Nama Pelanggan : ")
14     alamat = input("Masukkan Alamat : ")
15     nohp = input("Masukkan Nomor HP : ")
16
```

Gambar 10.38 Membuat Modul pelanggan.py

Gambar 10.39 ini adalah kode program untuk membuat modul "transaksi.py".

```
praktikum1_transaksi.py x
1 #mengimpor modul program
2 import praktikum1_pelanggan
3 import praktikum1_barang
4
5 global notrans, tgltrans, kdbrg, nmbrg, harga, jumlah
6 def input_transaksi():
7     print("\nData Transaksi Penjualan")
8     print("-----")
9     print("Nomor transaksi : ", notrans)
10    print("Tanggal transaksi : ", tgltrans)
11    #memanggil modul pelanggan
12    praktikum1_pelanggan.input_pelanggan()
13
14    #memanggil modul barang
15    praktikum1_barang.input_barang()
16    harga = int(praktikum1_barang.harga)
17    print("Jumlah pesan : ", jumlah)
18    totalhrp = harga * jumlah
19    print("Total harga : ", totalhrp)
20
21    notrans = input("Masukkan Nomor Transaksi : ")
22    tgltrans = input("Masukkan Tanggal Transaksi : ")
23    jumlah = int(input("Masukkan Jumlah Pesanan : "))
```

Gambar 10.39 Membuat Modul transaksi.py

- a. Simpan Program ini dengan nama `Praktikum10-1.py`
- b. Jalankan program `praktikum10-1` di atas, kemudian tuliskan apa yang tercetak di layar pada saat memanggil modul "**pelanggan.py**"

- c. Jalankan program `praktikum10-1` di atas, kemudian tuliskan apa yang tercetak di layar pada saat memanggil modul "**barang.py**"

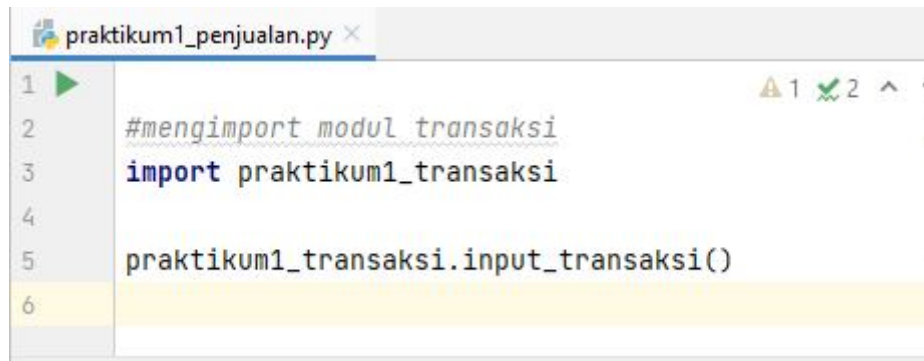
- d. Jalankan program `praktikum10-1` di atas, kemudian tuliskan apa yang tercetak di layar pada saat memanggil modul "**transaksi.py**"

2. Buatlah paket program dengan nama "**penjualan.py**" untuk memanggil modul-modul program yang dibuat pada praktikum 1 dengan struktur sebagai berikut :

```
|— penjualan.py
|— — pelanggan.py
|— — barang.py
|— — transaksi.py
```

Gambar 10.40 Contoh Struktur Modul Program

Gambar 10.41 ini adalah kode program untuk membuat modul "`transaksi.py`".



```
1  #mengimport modul transaksi
2
3  import praktikum1_transaksi
4
5  praktikum1_transaksi.input_transaksi()
6
```

Gambar 10.41 Membuat Modul Transaksi.py

Jalankan program praktikum10-2 di atas, kemudian tuliskan apa yang tercetak di layar pada saat memanggil paket program **"penjualan.py"**

## 10.7 Rangkuman

- Pendekatan pemrograman modular adalah satu konsep di mana kita memecah bagian besar program menjadi bagian-bagian kecil yang lebih bermakna.
- Penerapan modular programming membuat kode program lebih terstruktur dan mudah diorganisir.
- Modul pada python adalah sebuah file berekstensi .py yang berisi kode program python.
- Modul bisa kita panggil dari file yang lain.
- Paket adalah sebuah direktori yang memiliki satu file `__init__.py` dan di dalamnya berisi modul-modul python.
- Python memiliki modul dan paket-paket secara default.

## 10.8 Latihan

1. Buatlah program untuk mencetak informasi nilai mahasiswa menggunakan modul, prosedur dan fungsi sehingga menghasilkan keluaran program sebagai berikut :

```
*****
PROGRAM HITUNG NILAI MAHASISWA
*****

*****
Masukan Nilai UTS : 90
Masukan Nilai UAS : 90
Masukan Nilai QUIZ : 90
Masukan Banyak Tugas : 2
Masukan Nilai Tugas ke-1 : 90
Masukan Nilai Tugas ke-2 : 100
Rata-Rata Nilai Tugas      : 95.0
Nilai Akhir                : 91.0
Nilai indeks               : A
Nilai Predikat             : BAIK SEKALI
Nah COBA LAGI(y/t) ?
```

Gambar 10.42 Keluaran program

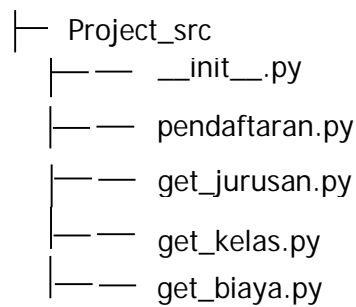
Modul-modul program yang harus dibuat adalah :

- a. Modul program untuk menginput data nilai dengan nama "input\_nilai.py".
- b. Modul program untuk menghitung nilai rata-rata dengan nama "hitung\_nilairatarata.py"
- c. Modul program untuk menghitung nilai akhir dengan nama "hitung\_nilaiakhir.py"
- d. Modul program untuk mencari nilai indeks dengan nama "get\_grade.py"
- e. Modul program untuk mencari nilai predikat dengan nama "get\_predikat.py".

## 10.9 Tugas Mandiri

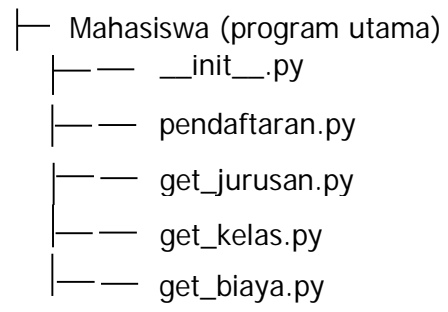
Kerjakan soal-soal berikut :

3. Buatlah modul program untuk menampilkan informasi mahasiswa sesuai dengan program studi dan fakultasnya dengan struktur sebagai berikut :



Gambar 10.43 Contoh Struktur Modul Program

- **Mahasiswa.py** terdiri atas :
  - Input dan mencetak informasi mahasiswa pada file **pendaftaran.py** yang terdiri atas : NIM, Nama Lengkap, No. HP, Email, Alamat, NEM.
  - Fungsi **get\_jurusan()** berdasarkan kode program studi yang ada di NIM.
    - Contoh : 19**12**502029, angka 12 (tebal) menjelaskan kode program studi.  
**11 = Teknik Informatika, 12 = Sistem Informasi, 13 = Sistem Komputer**
  - Fungsi **get\_kelas()** yang diambil berdasarkan kode kelas yang ada di dalam NIM
    - Contoh : 19125**0**021, angka 0 (tebal) menjelaskan kelas regular.
    - Contoh : 19125**1**021, angka 1 (tebal) menjelaskan kelas sore.
  - Fungsi **get\_biaya()** yang diambil berdasarkan kode program studi yang ada di dalam NIM :
    - Contoh : 19**11**50021, angka 11 (tebal) menjelaskan biaya kuliah = 9.800.000.
    - Contoh : 19**12**51021, angka 12 (tebal) menjelaskan biaya kuliah = 8.500.000.
    - Contoh : 19**13**51021, angka 13 (tebal) menjelaskan biaya kuliah = 9.200.000.
4. Buatlah paket program dengan nama "**mahasiswa.py**" untuk menampilkan informasi mahasiswa sesuai dengan program studi dan fakultasnya dengan struktur sebagai berikut :



Gambar 10.44 Contoh Struktur Modul Program



**FAKULTAS TEKNOLOGI INFORMASI**  
**UNIVERSITAS BUDI LUHUR**

Jl. Raya Ciledug, Petukangan Utara, Pesanggrahan

Jakarta Selatan, 12260

Telp: 021-5853753 Fax : 021-5853752

<http://fti.budiluhur.ac.id>