MODUL

1. Mengenal Modul

- **Modul** merupakan sebuah teknik dalam pemrograman Python yang digunakan untuk mengorganisasikan dan mengemas file-file program.
- Modul berisikan sekumpulan data dan fungsi dari suatu file program Python (*.py).
- Ketika membuat program yang komplek dan terdiri atas beberapa file yang berkaitan satu sama lain, teknik ini (Modul) sangat berperan dalam menghubungkan fungsi-fungsi dan/atau data-data yang terdapat pada satu file ke file yang lainnya.

2. Modul & File

- **Modul** dan ***file*** merupakan dua hal yang berbeda, akan tetapi merupakan satu kesatuan yang tidak terpisahkan.
- **Modul** merupakan ***file*** yang berisikan **kode-kode program Python**.
- File bisa berisi **kelas** (*Class*), **fungsi** (*Function*), dan **data**.

3. Jenis Modul dalam Python

Dalam pemrograman Python, dikenal 2 jenis **modul**, yaitu:

1. **Modul bawaan (*built-in*).**

   Modul (atau juga disebut dengan ***library/package***) yang secara ***default*** telah tersedia/terintegrasi di dalam *installer* ***interpreter*** Python dan dapat langsung digunakan oleh *user/programmer*.

   Contoh modul/*package* ***built-in*** : ***sys***, ***math***, ***cmath***, ***string***, dll

2. **Modul buatan user.**

   Modul yang memiliki fungsi khusus dan **dibuat** oleh *user* atau ***developer/programmer*** Python untuk tujuan tertentu.

4. Cara Memanggil Modul

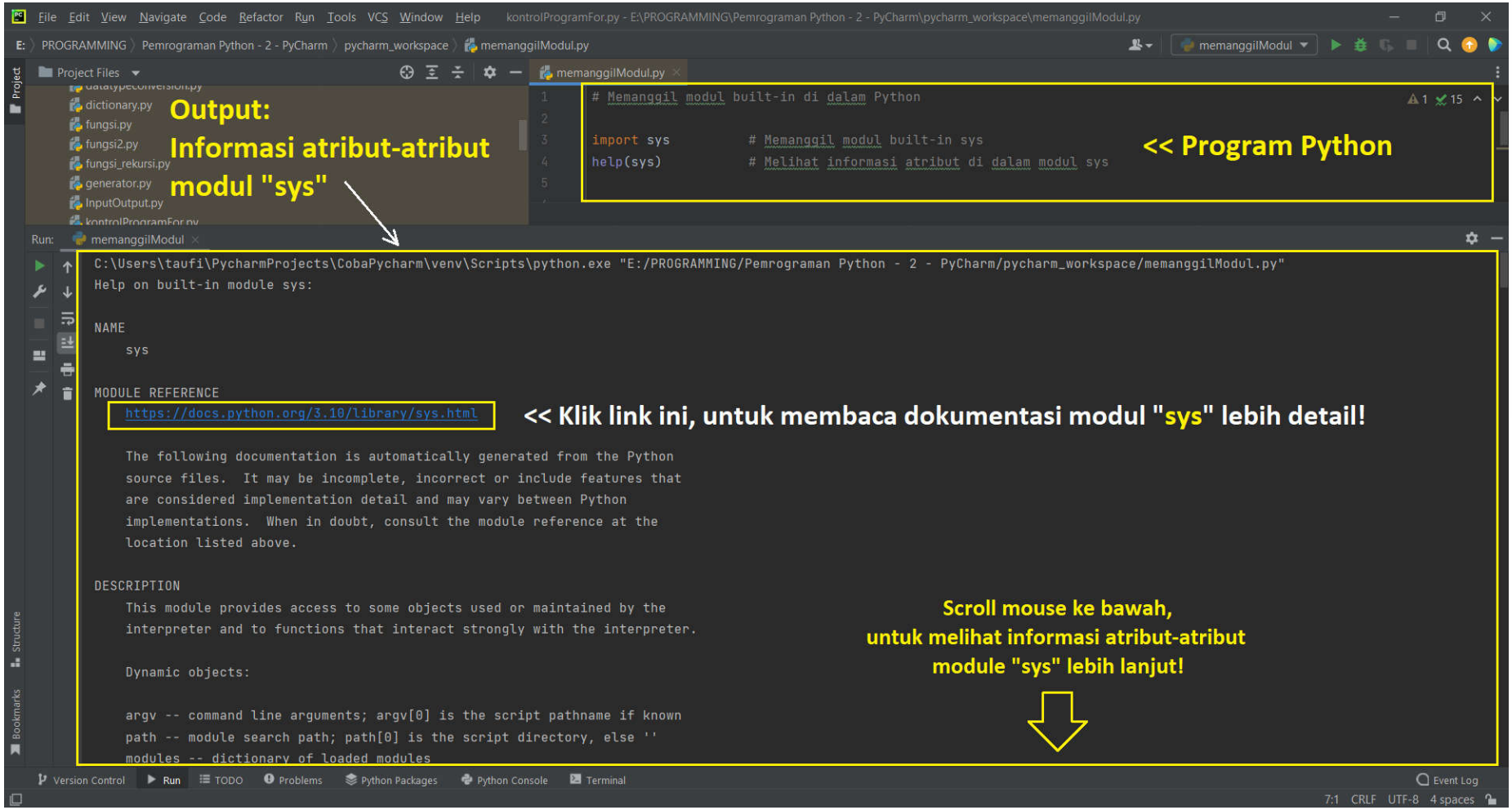
Modul dapat dipanggil dengan menggunakan perintah **`import`**, berikut format/sytanx-nya:

```
import nama_modul
```

Kemudian untuk mengetahui informasi mengenai atribut-atribut yang dimiliki oleh suatu modul dapat dilakukan dengan menggunakan fungsi ***help*** pada python, berikut format/sytanx-nya:

```
help(nama_modul)
```

Contoh 1: Melihat informasi atribut-atribut suatu modul



Klik link, https://docs.python.org/3.10/library/sys.html

Contoh 2:
Mengakses suatu atribut di dalam modul

```
import sys              # Memanggil modul built-in sys

a = sys.maxsize             # Mengakses atribut maxsize pada modul sys
b = 2**63 - 1
print("Nilai a = ",a)
print("Nilai b = ",b)
```

```
# Output
Nilai a =  9223372036854775807
Nilai b =  9223372036854775807
```

5. Membuat Modul

- **Membuat program Python** (*.py) adalah **sama dengan** **membuat modul**.
- Langkah membuat modul:
  1. Membuat direktori (**lingkaran**) penyimpan file modul dan program Python.
  2. Membuat file program modul (`lingkaran.py`).
  3. Membuat file program (`hitung_lingkaran.py`).
  4. Simpan kedua file Python di dalam direktori "**lingkaran**" yang telah dibuat pada langkah 1.
  5. Mengeksekusi (run) program.

Demonstrasi Membuat Modul

Langkah 1: Membuat direktori (**lingkaran**) penyimpan file modul dan program Python.

Langkah 2: Membuat **modul** lingkaran (**lingkaran.py**)

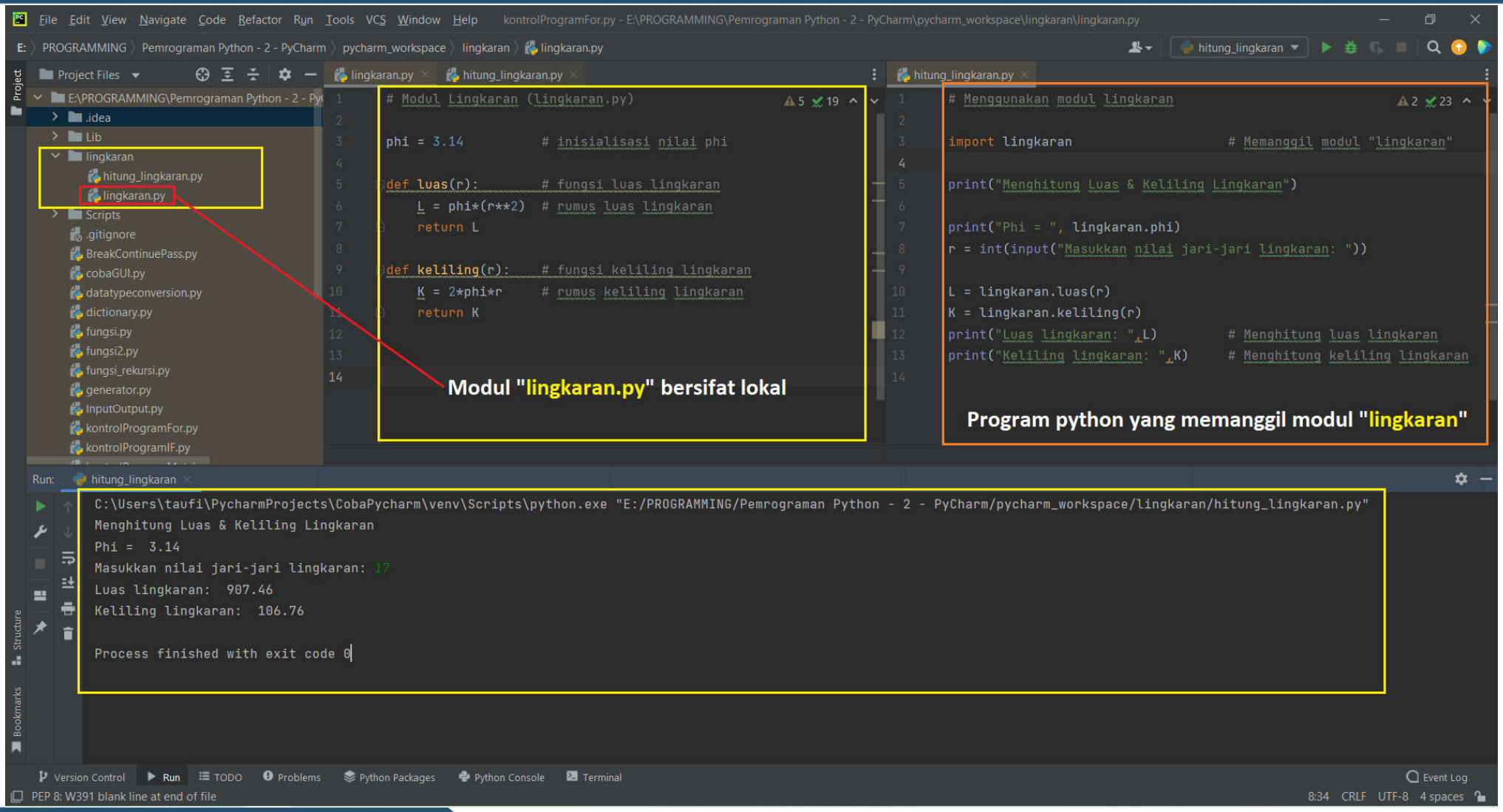
```
# Modul Lingkaran
phi = 3.14              # inisialisasi nilai phi

def luas(r):            # fungsi luas lingkaran
    L = phi*(r**2)      # rumus luas lingkaran
    return L

def keliling(r):        # fungsi keliling lingkaran
    K = 2*phi*r         # rumus keliling lingkaran
    return K
```

Langkah 3: Membuat program menggunakan **modul** lingkaran (**hitung_lingkaran.py**)

```
import lingkaran                        # Memanggil modul "lingkaran"

print("Menghitung Luas & Keliling Lingkaran")

print("Phi = ", lingkaran.phi)          # Memanggil variabel "phi"
r = int(input("Masukkan nilai jari-jari lingkaran: "))

L = lingkaran.luas(r)                   # Memanggil fungsi "luas"
K = lingkaran.keliling(r)               # Memanggil fungsi "keliling"
print("Luas lingkaran: ",L)             # Menghitung luas lingkaran
print("Keliling lingkaran: ",K)         # Menghitung keliling lingkaran
```

Langkah 4: Simpan kedua file Python di dalam direktori "**lingkaran**" yang telah dibuat pada langkah 1.

Langkah 5: Mengeksekusi program (run).



Modul Di Dalam Direktori (Path) Lokal

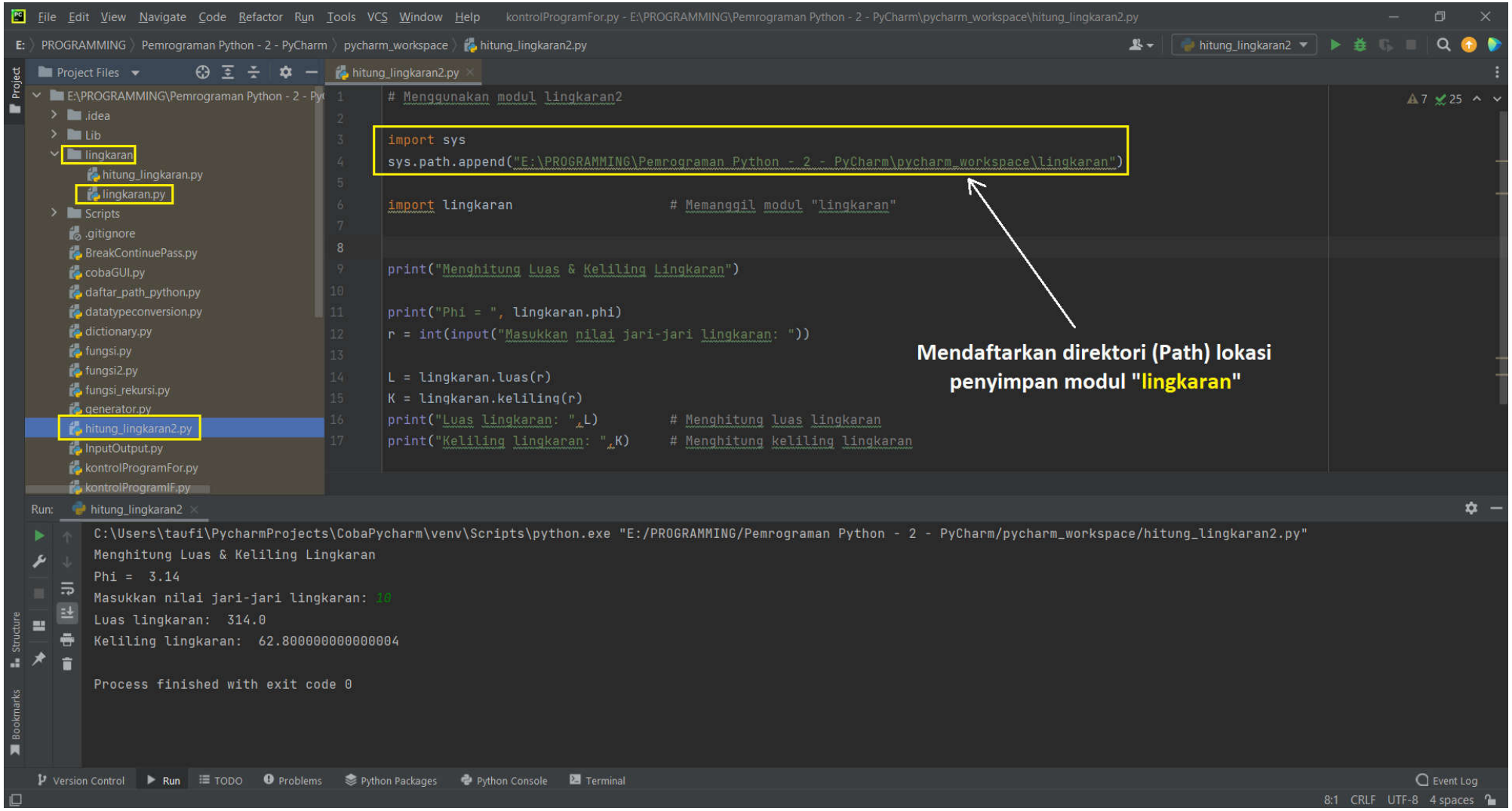
- Modul "**lingkaran**" berada di dalam direktori (*path*) yang sama dengan **file program Python**-nya, maka disebut sebagai modul yang bersifat **lokal**.
- Dengan kata lain, modul tersebut **hanya bisa dipanggil** oleh program yang berada **di dalam direktori yang sama** dengan modul tersebut (lingkaran) dan **tidak bisa dipanggil** atau digunakan oleh program lain yang berada **diluar direktori** tersebut.

Supaya modul "lingkaran" menjadi modul global,
sehingga dapat digunakan oleh program lain,
bagaimana solusinya?

1. Mendaftarkan Direktori (*Path*) Modul

Supaya suatu modul (*custom* atau buatan sendiri) yang berada **di dalam direktori yang kita tentukan sendiri** (bukan direktori **C:\Program Files\Python310\Lib**) dapat digunakan/dipanggil oleh program lain yang tidak berada di dalam direktori (*path*) yang sama dengan modul tersebut, maka kita dapat **mendaftarkan direktori penyimpan modul** tersebut supaya dikenali oleh *interpreter* Python dengan format perintah sebagai berikut:

```
import sys
sys.path.append("Nama_directory/path")
```



Program memanggil modul lingkaran yang telah didaftarkan

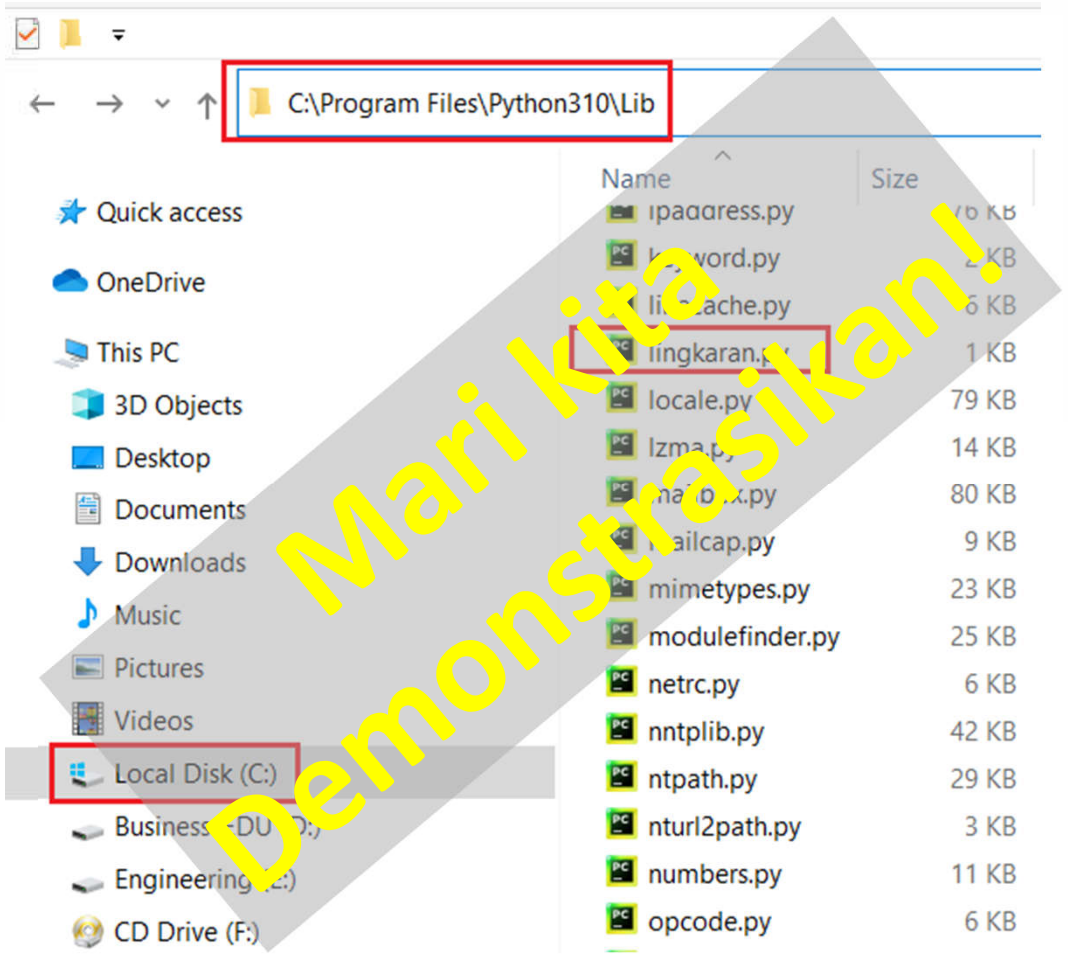
```
# Menggunakan modul lingkaran yang telah didaftarkan
import sys
sys.path.append("E:\PROGRAMMING\Pemrograman Python - 2 - PyCharm\pycharm_workspace\lingkaran")
import lingkaran                    # Memanggil modul "lingkaran"

print("Menghitung Luas & Keliling Lingkaran")
print("Phi = ", lingkaran.phi)
r = int(input("Masukkan nilai jari-jari lingkaran: "))

L = lingkaran.luas(r)
K = lingkaran.keliling(r)
print("Luas lingkaran: ",L)         # Menghitung luas lingkaran
print("Keliling lingkaran: ",K)     # Menghitung keliling lingkaran
```

```
# Output
Menghitung Luas & Keliling Lingkaran
Phi =  3.14
Masukkan nilai jari-jari lingkaran: 7
Luas lingkaran:  153.86
Keliling lingkaran:  43.96
```

2. Memindah Letak Direktori (*Path*) File Modul

Supaya suatu modul ***custom*** (buatan sendiri) dapat digunakan/dipanggil oleh program lain yang tidak berada di dalam direktori (*path*) yang sama dengan modul tersebut, maka modul tersebut harus dipindah (**Cut & paste**) letaknya ke dalam direktori/*path* (**Lib**) standar ***interpreter*** **Python**, yaitu direktori:

C:\Program Files\Python310\Lib



CARA MEMANGGIL MODUL (Lebih Lanjut)

Cara Memanggil Modul (Lebih Lanjut)

1. Pemanggilan modul secara penuh
2. Pemanggilan modul dengan alias
3. Pemanggilan atribut modul secara spesifik

1. Pemanggilan Modul Secara Penuh

Pemanggilan (**import**) modul secara penuh akan memanggil modul secara keseluruhan, sehingga seluruh atribut yang dimiliki sebuah modul dapat digunakan. Berikut adalah format penulisan/sintaksisnya:

```
import nama_modul                    # Panggil sebuah modul
import nama_modul1, nama_modul2      # Panggil beberapa modul
```

Contoh: Program memanggil modul penuh

```
# Memanggil Modul Secara Penuh
import math                          # Panggil modul 'math'
pilih='y'
print("Menghitung Faktorial Suatu Bilangan")
while pilih=='y' or pilih=='Y':
    t=int(input("Tulis bilangan: "))
    print("Faktorial",t, "adalah",math.factorial(t))
    pilih=input("Tulis bilangan lagi? (y/n): ")
```

```
# Output
Menghitung Faktorial Suatu Bilangan
Tulis bilangan: 5
Faktorial 5 adalah 120
Tulis bilangan lagi? (y/n):
```

Contoh: Program memanggil lebih dari satu modul

```
# Memanggil Modul Lebih dari Satu
import math, random                 # Panggil modul 'math' dan 'random'


print("Menghitung Faktorial 5 Bilangan Acak")
for i in range(5):
    x=random.randint(1,30)          # Membangkitkan bilangan acak antara 1-30 dengan fungsi "randint"
    print(i+1,". Faktorial",x,"=", math.factorial(x))
```

```
# Output

Menghitung Faktorial 5 Bilangan Acak
1 . Faktorial 2 = 2
2 . Faktorial 30 = 265252859812191058636308480000000
3 . Faktorial 7 = 5040
4 . Faktorial 5 = 120
5 . Faktorial 23 = 25852016738884976640000
```

2. Pemanggilan Modul dengan Alias

Pemanggilan (**import**) modul dengan **alias**, sebenarnya sama dengan cara pemanggilan modul secara penuh, akan tetapi modul yang dipanggil dengan **nama aliasnya** yang biasanya berupa variabel. Berikut adalah format penulisan/sintaksisnya:

```
import nama_modul as nama_alias
```

Contoh: Program memanggil alias modul

```
# Memanggil Modul dengan Alias
import math as berhitung                # Panggil alias modul 'math'


pilih='y'
print("Menghitung Faktorial Suatu Bilangan")
while pilih=='y' or pilih=='Y':
    t=int(input("Tulis bilangan: "))
    print("Faktorial",t, "adalah",berhitung.factorial(t))
    pilih=input("Tulis bilangan lagi? (y/n): ")
```

```
# Output
Menghitung Faktorial Suatu Bilangan
Tulis bilangan: 7
Faktorial 7 adalah 5040
Tulis bilangan lagi? (y/n):
```

3. Pemanggilan Atribut Modul Secara Spesifik

Pemanggilan (**import**) **atribut** dari suatu modul dapat dilakukan secara **spesifik** (langsung). Berikut adalah format penulisan/sintaksisnya:

```
from nama_modul import atribut1, atribut2, ... atributn
```

Dengan cara pemanggilan atribut secara spesifik, maka program dapat memanggil/mengakses atribut suatu modul secara langsung tanpa perlu menggunakan format **nama_modul.nama_atribut**.

Contoh:
Program memanggil atribut modul secara spesifik

```
# Memanggil Atribut Modul Secara Spesifik
from random import randint
from math import factorial, pow

print("Menghitung Faktorial & Kuadrat Bilangan")
for i in range(4):
    data=randint(1,15)      # Membangkitkan bilangan acak rentang 1-15
    print("Faktorial ",data,"\t:",factorial(data))
    print("Kuadrat   ",data,"\t:",pow(data,2))
```

```
# Output
Menghitung Faktorial & Kuadrat Bilangan
Faktorial  8       :  40320
Kuadrat    8       :  64.0
Faktorial  3       :  6
Kuadrat    3       :  9.0
Faktorial  8       :  40320
Kuadrat    8       :  64.0
Faktorial  6       :  720
Kuadrat    6       :  36.0
```

NAMESPACE

Namespace & Modul

- Ketika suatu program Python yang dibangun **semakin besar**, maka penggunaan nama-nama variabel atau fungsi dalam suatu modul semakin banyak dan bervariasi.
- Hal ini memungkinkan dapat terjadinya **konflik penamaan** variabel atau fungsi dalam program.
- Mengatasi permasalahan ini, Python menerapkan konsep **namespace.**
- **Konsep** dan **kegunaan umum** dari **namespace** dalam pemrograman Python adalah untuk **mencegah terjadinya konflik penamaan** di dalam program **dengan cara memetakan** nama-nama tersebut **ke dalam objek**.
- Hal ini diterapkan pada modul, ketika di dalam dua buah modul (atau lebih) yang berbeda memiliki **fungsi** dan/atau **atribut** dengan **nama yang sama**, **sehingga tidak terjadi konflik** ketika modul tersebut **dipanggil** dalam program Python.

Contoh Namesapce: Membuat Modul yang Dibutuhkan

siswa.py

```
# Modul siswa.py
def nama():
    a="Budi"
    print("Nama Siswa: ", a)

def alamat():
   b="Sleman, Yogyakarta"
   print("Alamat Siswa: ", b)
```

guru.py

```
# Modul guru.py
def nama():
    r="Pak Guru"
    print("Nama Guru: ", r)

def alamat():
    s="Bandung, Jawa Barat"
    print("Alamat Guru: ", s)
```

Contoh Namesapce

```
# Program contoh Namespace modul

import sys
sys.path.append("E:\PROGRAMMING\Pemrograman Python - 2 - PyCharm\pycharm_workspace\SiswaDanGuru")
import siswa,guru                # Mengimpor modul siswa dan modul guru

print("\nTampilkan nama & alamat Siswa:")
siswa.nama()
siswa.alamat()

print("\nTampilkan nama & alamat Guru:")
guru.nama()
guru.alamat()
```

```
# Output
Tampilkan nama & alamat Siswa:
Nama Siswa:  Budi
Alamat Siswa:  Sleman, Yogyakarta

Tampilkan nama & alamat Guru:
Nama Guru:  Pak Guru Umar
Alamat Guru:  Bandung, Jawa Barat
```

Terima Kasih