

# Cheat Sheet **UKK** Klaster 3

*Mengimplementasikan*

*Pemrograman Berorientasi Objek*

---

*Tasikmalaya, 28 Februari 2021*

# Didalam sini ada...

|  |   |
|--|---|
| 0. Persiapan   | 0.1 <b>Tools</b> Pemrograman  |
|  | 0.2 <b>Instalasi</b> Python   |
|  | 0.3 <b>Aturan dasar</b> Python  |
| 1. Membuat program berorientasi objek dengan memanfaatkan <b>class</b>                               | 1.1 Program dengan menggunakan <b>class</b> dibuat.                                   |
|  | 1.2 Properti <b>class</b> yang akan direalisasikan dalam bentuk fungsi dibuat.        |
|  | 1.3 Data didalam <b>class</b> dibuat mandiri.   |
|  | 1.4 Hak akses dari tipe data ( <b>public</b> , <b>protected</b> dan <b>private</b> ). |
| 2. Menggunakan tipe data dan <i>control program</i> pada metode atau operasi dari suatu <b>class</b> | 2.1 Tipe data diidentifikasi.   |
|  | 2.2 Sintaks program dikuasai sesuai dengan bahasa pemrogramannya.                     |
|  | 2.3 <i>Control program</i> dikuasai.  |
| 3. Membuat program dengan konsep berbasis objek  | 3.1 <b>Inheritance</b> pada <b>class</b> ditetapkan.                                  |
|  | 3.2 <b>Polymorphism</b> pada <b>class</b> ditetapkan.                                 |
|  | 3.3 <b>Overloading</b> pada <b>class</b> ditetapkan.                                  |
| 4. Membuat program <i>object oriented</i> dengan <b>interface</b> dan <b>package</b>                 | 4.1 <b>Interface class</b> program dibuat.  |
|  | 4.2 <b>Package</b> dengan program dibuat.   |
| 5. Mengkompilasi program   | 5.1 Kesalahan dapat dikoreksi.  |
|  | 5.2 Program bebas salah sintaks dihasilkan.   |

**O**

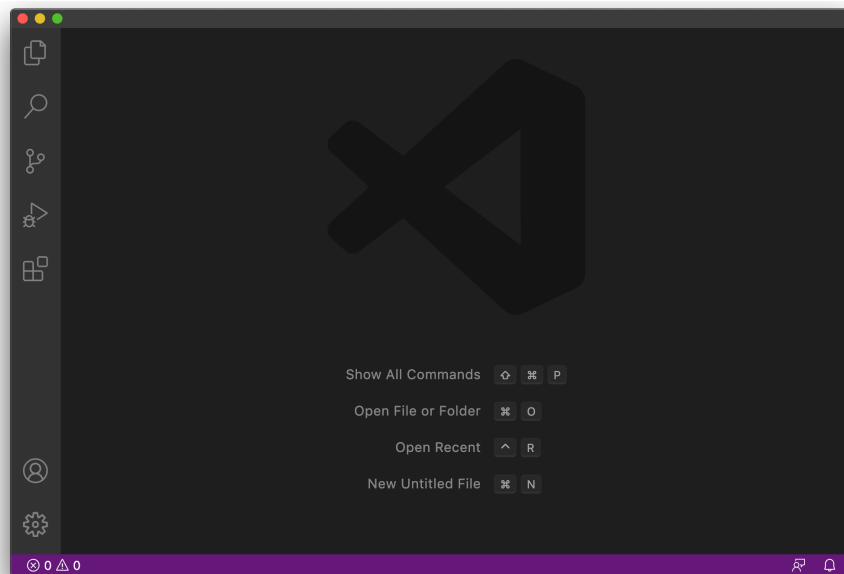
**PERSIAPAN**

# Tools Pemrograman

| NO. | Tools                   | Nama Tools         | Keterangan                | Link Unduh  |
|-----|-------------------------|--------------------|---------------------------|---|
| 1.  | Python                  | Python 3.x         | Bahasa pemrograman & IDE  | <a href="https://python.org/downloads">https://python.org/downloads</a>                     |
| 2.  | Text Editor             | Visual Studio Code | Untuk menulis program     | <a href="https://code.visualstudio.com/download">https://code.visualstudio.com/download</a> |
| 3.  | Cmd ( <i>opsional</i> ) | Cmd/Terminal       | Untuk menjalankan program | <i>include</i> di Sistem Operasi/VS Code  |

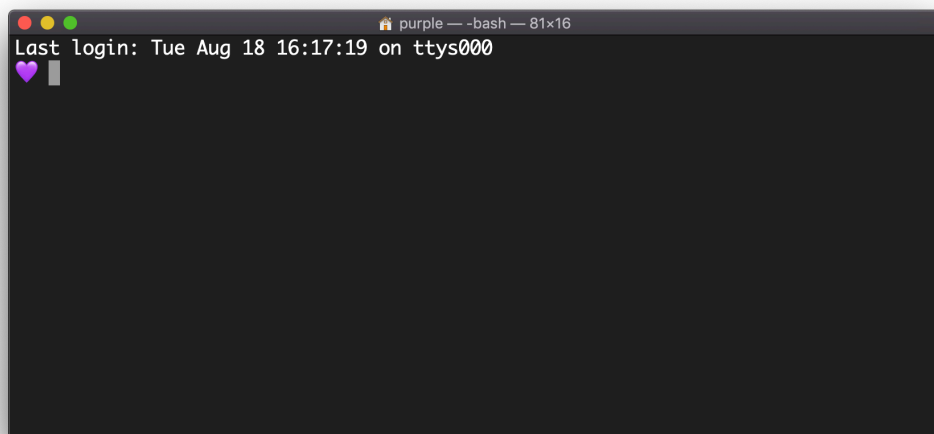
## Visual Studio Code

Kita samakan *tools* Text Editor menggunakan Visual Studio Code.



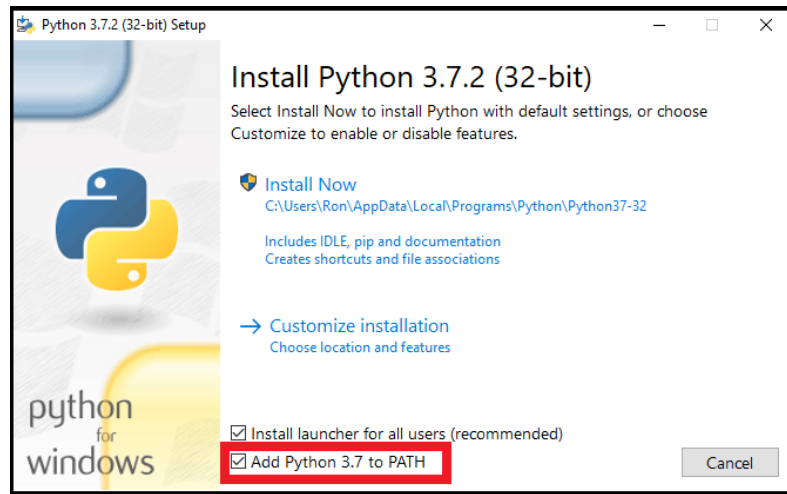
## Cmd/Terminal

Mungkin akan sedikit berbeda dengan Cmd di Windows. Cmd digunakan untuk menjalankan program. Untuk menjalankan program dapat menggunakan Cmd di Visual Studio Code.



# Instalasi Python

Pastikan **Add Python...to PATH** dicentang. **Install Now** dan tunggu hingga selesai. Perhatikan yang diberi tanda merah.



Buka kembali Cmd, ketik **python** lalu Enter. Pastikan hasilnya seperti berikut.

```
purple — Python — 81x16
Last login: Tue Aug 18 16:17:19 on ttys000
❖ python
Python 3.7.8 (default, Jul 8 2020, 14:16:55)
[Clang 11.0.0 (clang-1100.0.33.17)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Bila tampil interpreter Python, maka Python telah terpasang dengan benar.

# Aturan dasar Python

Banyak Bahasa Pemrograman, artinya banyak aturan penulisan sintaks. Penulisan sintaks C++ akan berbeda dengan Python, begitupun dengan bahasa pemrograman lainnya. Tetapi mereka memiliki tujuan yang sama, yaitu membuat *Apps* (aplikasi). Coba perhatikan kedua bahasa pemrograman berikut.

| PYTHON   | C++  |
|--|--|
| <pre>class Mobil:     nama = ""     merek = ""     warna = ""  mobil = Mobil() mobil.nama = "Brio" mobil.merek = "Honda" mobil.warna = "Merah"</pre> | <pre>#include &lt;iostream&gt; #include &lt;string&gt; using namespace std;  class Mobil { public:     string nama, merek, warna; };  int main() {     Mobil mobil;     mobil.nama = "Brio";     mobil.merek = "Honda";     mobil.warna = "Merah";     return 0; }</pre> |

Kedua contoh diatas memiliki tujuan sama, membuat objek **mobil** (*m kecil*) dari class **Mobil** (*M besar*). Program yang ditulis Python cenderung lebih sedikit daripada C++, tanpa ada deklarasi *header file*. Ini dikarenakan kedua bahasa ini menggunakan Penerjemah yang berbeda. C++ menggunakan **Compiler**, sedangkan Python menggunakan **Interpreter**.

## Tahukah Kamu

**Compiler:** Menerjemahkan program sekaligus dan memiliki proses yang panjang sampai program menjadi *executable file*.

**Interpreter:** Menerjemahkan program baris-per-baris.

Aturan dasar pada Python:

## File Python

Setiap membuat program dengan Python, selalu simpan dengan ekstensi **.py** (titik py) diakhir nama *file*. Misalkan **hitung\_segitiga.py**, **game.py**, **makanan.py**.

## Komentar

Penulisan komentar pada Python terdiri dari dua. Lihat perbedaanya.

```
# ini untuk menulis komentar satu baris

"""
ini untuk menulis
komentar lebih
dari satu baris
"""
```

## Variable

Setiap Bahasa pemrograman pasti memiliki *variable*, berguna untuk mengisi/mengolah data yang dioperasikan saat program berjalan. Pada Python, pendeklarasian *variable* tidak diawali atau diikuti dengan Tipe Data, karena apapun isi *variable*-nya, maka itulah Tipe Data-nya.

```
merek = "Honda"
tahun = 2020
```

*Variable* **merek** berisikan teks yang ditandai dengan tanda kutip, maka *variable* **merek** bertipe String. Begitupun dengan *variable* **tahun** yang isinya angka tanpa kutip, artinya *variable* **tahun** bertipe Number.



# **1**

## **MEMBUAT PROGRAM BERORIENTASI OBJEK DENGAN MEMANFAATKAN CLASS**

# Apa itu Class

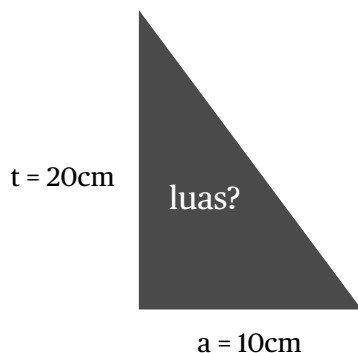
Dalam definisi Pemrograman Berorientasi Objek dikenal adanya *Class* dan *Object*. *Class* mendefinisikan karakteristik abstrak dari sesuatu termasuk atribut atau sifat-sifat dari sesuatu dan apa yang dapat dikerjakan oleh sesuatu (fungsi). Sebagai contoh, mobil adalah sebuah kelas yang memiliki atribut merek, warna dan tipe. Mobil juga memiliki fungsi/*method*, antara lain, maju, mundur dan berhenti.



# Apa itu Object

*Object* atau objek merupakan contoh dari kelas yang sudah didefinisikan. Atribut dan fungsi/*method* dari kelas secara otomatis akan menurun pada objek.

# Membuat Class Hitung Segitiga



Diketahui Segitiga siku-siku memiliki tinggi  $t = 20cm$  dan alas  $a = 10cm$ . Berapa luasnya?  
Kita buat konsep dasarnya seperti berikut.

| ATTRIBUTE/DATA | BEHAVIOUR/METHOD |
|----------------|------------------|
| Tinggi = 20    | Luas?            |
| Alas = 10      |                  |

Buat file baru **bangundatar.py** dan buat class Segitiga dengan **properti** yang telah ditentukan.

#### bangundatar.py

```
# class Segitiga dengan parameter a dan t diisi default yaitu 0
class Segitiga:
    def __init__(self, a=0, t=0):
        self.a = a
        self.t = t

    def luas(self):
        return (self.a * self.t) / 2

# Buat objek yang bernama sg dari class Segitiga()
# masukkan parameter alas=10 dan tinggi=20
sg = Segitiga(10, 20)
print(sg.luas())
```

## Hak Akses Properti

Agar setiap properti seperti alas, tinggi dan fungsi hitung luas tidak dapat diakses langsung dari luar class, maka diperlukan teknik Enkapsulasi untuk melindungi properti. Dalam kasus ini alas dan tinggi akan dibuat **private** dengan menambahkan `__` (*underscore*) dua kali didepannya menjadi **self.\_\_a**.

#### bangundatar.py

```
# class Segitiga dengan parameter a dan t diisi default yaitu 0
class Segitiga:
    def __init__(self, a=0, t=0):
        self.__a = a
        self.__t = t

    def luas(self):
        return (self.__a * self.__t) / 2

sg = Segitiga(10, 20)
print(sg.luas())
```

Lakukan hal yang sama pada fungsi **luas()** menjadi **\_\_luas()**, lalu buat fungsi baru yang bernama **cetakLuas()** yang bersifat *public* dan mengembalikan fungsi **\_\_luas()**.

### bangundatar.py

# class Segitiga dengan parameter a dan t diisi default yaitu 0

```
class Segitiga:
```

```
    def __init__(self, a=0, t=0):
```

```
        self.__a = a
```

```
        self.__t = t
```

```
    def __luas(self):
```

```
        return (self.__a * self.__t) / 2
```

→ **Private** function

```
    def cetakLuas(self):
```

```
        return self.__luas()
```

→ **Public** function

```
sg = Segitiga(10, 20)
```

```
print(sg.cetakLuas())
```

→ call public function cetakLuas()

Dengan begitu, alas, tinggi dan fungsi hitung luas hanya bisa diakses dari dalam kelas itu sendiri. Sedang fungsi **cetakLuas()** yang bersifat **public** memanggil fungsi **\_\_luas()** yang **private** untuk mendapatkan hasil dari perhitungan luas segitiga.

Berikut kode lengkapnya.

```
# bangundatar.py
```

```
class Segitiga:
```

```
    def __init__(self, a=0, t=0):
```

```
        self.__a = a
```

```
        self.__t = t
```

```
    def __luas(self):
```

```
        return (self.__a * self.__t) / 2
```

```
    def cetakLuas(self):
```

```
        return self.__luas()
```

```
sg = Segitiga(10, 20)
```

```
print(sg.cetakLuas())
```

# 2

## **MENGGUNAKAN TIPE DATA DAN CONTROL PROGRAM**

# Tipe Data

Tipe data atau disebut klasifikasi data dalam pemrograman berfungsi untuk mengklasifikasikan setiap data yang dimasukkan kedalam aplikasi sesuai data yang dibutuhkan.

| NO. | Tipe Data | Keterangan  |
|-----|-----------|---|
| 1.  | Karakter  | Tipe yang menyimpan sebuah karakter. contoh: 'L', 'P'                                   |
| 2.  | String    | Kumpulan dari karakter. contoh: "pemrograman berorientasi objek"                        |
| 3.  | Integer   | Tipe bilangan bulat. contoh: -2 -1 0 1 2 3  |
| 4.  | Float     | Tipe bilangan pecahan. contoh: 3.14   |
| 5.  | Boolean   | Hanya memiliki dua nilai, True dan False  |
| 6.  | Array     | Tipe yang menyimpan kumpulan item dalam elemen yang memiliki nomor index dimulai dari 0 |

Pada BAB sebelumnya telah dibuat sebuah *class* untuk menghitung luas Segitiga dengan alas dan tinggi yang tidak diketahui tipe data-nya. Dalam Python, untuk menentukan tipe data, tidak perlu menyematkan tipe data dalam sebuah variabel. Karena, apapun isi variabelnya, itulah tipe datanya. Dalam kasus *class* Segitiga, alas diisi 10 dan tinggi diisi 20. Artinya tipe data untuk *variable* alas dan tinggi adalah *number* yaitu **integer**.

## bangundatar.py

```
# class Segitiga dengan parameter a dan t diisi default yaitu 0
class Segitiga:
    ...

sg = Segitiga(10, 20)
print(sg.cetakLuas())
```

→ Tipe data integer

# Control Program

*Control program* lebih mengarah kepada alur program seperti pemilihan (**if-else**), perulangan (**for** dan **while**). Dalam kasus ini kita buat program lebih fleksibel dengan cara *input* manual alas dan tinggi lalu program akan bertanya setelah **luas** ditampilkan 'apakah akan menghitung lagi?'. Tambahkan *variable* alas dan tinggi yang diisi fungsi **input()** untuk menerima masukan dari *keyboard* dan konversikan kedalam **integer** dengan fungsi **int()**.

### bangundatar.py

```
# class Segitiga dengan parameter a dan t diisi default yaitu 0
class Segitiga:
```

```
...
```

```
tanya = True
```

```
while tanya:
```

```
    # input alas dan tinggi
```

```
    alas = int(input('Alas: '))
```

```
    tinggi = int(input('Tinggi: '))
```

```
    sg = Segitiga(alas, tinggi)
```

```
    print(persegi.cetakLuas())
```

```
    jawab = input("apakah akan menghitung lagi? [y/t] ")
```

```
    if jawab == "y":
```

```
        tanya = True
```

```
    elif jawab == "t":
```

```
        tanya = False
```

Buat looping dengan kondisi awal True yang ditetapkan dalam variable tanya

alas dan tinggi menerima input dari keyboard

Masukkan kedalam parameter

Jika jawab "y" maka akan mengulang.  
Jika "t" maka program akan berhenti.

Berikut kode lengkapnya.

```
# bangundatar.py
```

```
class Segitiga:
```

```
    def __init__(self, a=0, t=0):
```

```
        self.__a = a
```

```
        self.__t = t
```

```
    def __luas(self):
```

```
        return (self.__a * self.__t) / 2
```

```
    def cetakLuas(self):
```

```
        return self.__luas()
```

```
tanya = True
```

```
while tanya:
```

```
    alas = int(input("Alas: "))
```

```
    tinggi = int(input("Tinggi: "))
```

```
    sg = Segitiga(alas, tinggi)
```

```
    print(sg.luas())
```

```
    jawab = input("apakah akan menghitung lagi? [y/t] ")
```

```
    if jawab == "y":
```

```
        tanya = True
```

```
    elif jawab == "t":
```

```
        tanya = False
```

# 3

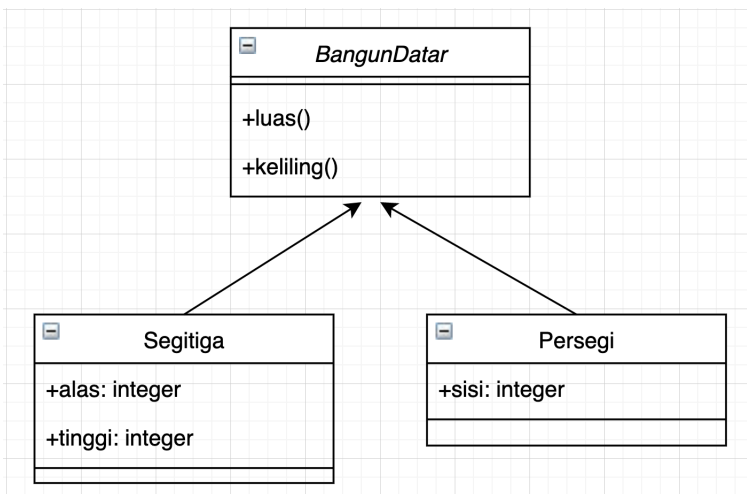
## **MEMBUAT PROGRAM DENGAN KONSEP BERBASIS OBJEK**



Pada BAB ini terdiri dari 3 Sub-BAB yang mungkin secara teknis saat mengerjakan *inheritance* menyinggung polimerfisme dan saat mengerjakan polimorfisme menyinggung *overloading*. Ketiga istilah ini sangat erat sekali dengan pemrograman objek, begitupun dengan *interface class* di BAB selanjutnya yang mungkin secara tidak sengaja akan disinggung di BAB ini.

## Inheritance / Pewarisan

*Class* dalam pemrograman objek bisa saling terhubung satu sama lain dengan menggunakan metode pewarisan. Dalam pewarisan terdiri dari **class induk** dan **class anak**. Untuk lebih jelas perhatikan ilustrasi dibawah ini.



Pada gambar disamping terdiri dari 3 *class*: *class* **BangunDatar** sebagai *class* induk dan *class* **Segitiga** dan **Persegi** sebagai *class* anak (*sub-class*). Setiap *data* dan fungsi yang dimiliki *class* induk, akan diwariskan ke *class* anak, sehingga *class* anak memiliki *data* dan fungsi yang sama dengan *class* induk.

Buat *class* **BangunDatar** dalam file **bangundatar.py** dengan menambahkan *class* **BangunDatar** dengan fungsi **luas()** dan **keliling()** diatas *class* **Segitiga**.

```
bangundatar.py

# class induk yang akan diwariskan kepada class Segitiga
class BangunDatar:
    def luas(self):
        print("Hitung Luas")
        return

    def keliling(self):
        print("Hitung Keliling")
        return

class Segitiga(BangunDatar):
    def __init__(self, a=0, t=0, s=0):
        self.__a = a
        self.__t = t
        self.__s = s

    def luas(self):
        return (self.__a * self.__t) / 2

    def keliling(self):
        return self.__a + self.__t + self.__s
```

Tambahkan class BangunDatar diatas class Segitiga

masukkan BangunDatar kedalam parameter class Segitiga

tambahkan parameter s=0 dan masukkan kedalam fungsi \_\_init\_\_

Buat fungsi keliling() didalam class Segitiga

Berikut kode lengkapnya.

```
# bangundatar.py

class BangunDatar: # class induk
    def luas(self):
        print("Hitung Luas")
        return
    def keliling(self):
        print("Hitung Keliling")
        return

class Segitiga(BangunDatar): # class anak yang mewarisi class BangunDatar
    def __init__(self, a=0, t=0, s=0):
        self.__a = a # data dibuat private
        self.__t = t
        self.__s = s

    def luas(self):
        return (self.__a * self.__t) / 2

    def keliling(self):
        return self.__a + self.__t + self.__s

tanya = True
while tanya:
    alas = int(input("Alas: "))
    tinggi = int(input("Tinggi: "))
    sisi = int(input("Sisi Miring: "))

    bangundatar = BangunDatar() # buat objek bangundatar
    sg = Segitiga(alas, tinggi, sisi) # buat objek segitiga

    bangundatar.luas() # panggil luas() dari BangunDatar
    print(sg.luas())

    bangundatar.keliling() # panggil keliling() dari BangunDatar
    print(sg.keliling())

    jawab = input("apakah akan menghitung lagi? [y/t] ")

    if jawab == "y":
        tanya = True
    elif jawab == "t":
        tanya = False
```

diwariskan ke class **Segitiga**

# Polimorfisme

Suatu *class* dapat memiliki banyak **bentuk** fungsi yang berbeda-beda meskipun nama fungsinya sama. Bentuk disini dapat diartikan sebagai isi, parameter dan tipe datanya yang berbeda-beda.

Dalam polimorfisme ada metode yang dinamakan **Overloading**. Metode ini terjadi pada *class* anak yang memiliki *nama fungsi yang sama*, tapi *parameter dan tipe datanya berbeda*.

Polimorfisme inilah yang terjadi pada contoh kasus sebelumnya.

```
# class induk

class BangunDatar:
    def luas(self):
        print("Hitung Luas")
        return
    def keliling(self):
        print("Hitung Keliling")
        return
```

```
# class anak

class Segitiga(BangunDatar):
    def __init__(self, a=0, t=0, s=0):
        self.__a = a
        self.__t = t
        self.__s = s

    def luas(self):
        return (self.__a * self.__t) / 2

    def keliling(self):
        return self.__a + self.__t + self.__s
```

## Overloading

Seperti yang dijelaskan sebelumnya, dibagian ini kita akan membuat *class* Persegi sama halnya dengan *class* Segitiga memiliki method **luas()** dan **keliling()**. Buat *class* Persegi dibawah *class* Segitiga didalam file **bangundatar.py**.

### bangundatar.py

```
# class induk yang akan diwariskan kepada class Segitiga dan Persegi
class BangunDatar:
    ...

class Segitiga(BangunDatar):
    ...

class Persegi(BangunDatar):
    def __init__(self, s=0):
        self.__s = s

    def luas(self):
        return self.__s * self.__s

    def keliling(self):
        return 4 * self.__s
```

Berikut kode lengkapnya.

```
# bangundatar.py

class BangunDatar: # class induk
    def luas(self):
        print("Hitung Luas")
        return
    def keliling(self):
        print("Hitung Keliling")
        return

class Segitiga(BangunDatar): # class anak yang mewarisi class BangunDatar
    def __init__(self, a=0, t=0, s=0):
        self.__a = a # data dibuat private
        self.__t = t
        self.__s = s

    def luas(self):
        return (self.__a * self.__t) / 2

    def keliling(self):
        return self.__a + self.__t + self.__s

class Persegi(BangunDatar): # class anak yang mewarisi class BangunDatar
    def __init__(self, s=0):
        self.__s = s # data dibuat private

    def luas(self):
        return self.__s * self.__s

    def keliling(self):
        return 4 * self.__s

bangundatar = BangunDatar()
sg = Segitiga(10, 20, 17)
persegi = Persegi(22)

print("Luas segitiga:",sg.luas())
print("Keliling segitiga:",sg.keliling())
print("Luas Persegi:",persegi.luas())
print("Keliling Persegi:",persegi.keliling())
```

# 4

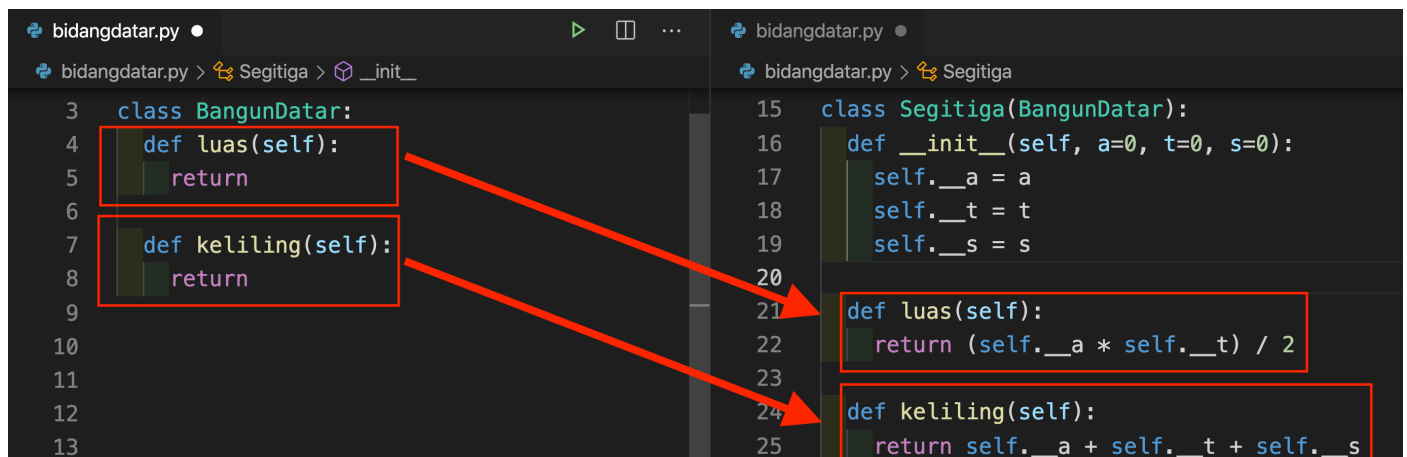
## **MEMBUAT PROGRAM OBJECT ORIENTED DENGAN INTERFACE DAN PACKAGE**

# Interface

Interface disebut juga dengan *blueprint*, ialah sebuah *class* kosong yang berisi *method-method* yang akan diimplementasikan di *class* lain.

Contohnya dalam *class* induk BangunDatar memiliki *method* **luas()** dan **keliling()** yang kosong, kedua *method* ini akan diisi di *class* Segitiga dan Persegi yang merupakan turunan dari *class* BangunDatar.

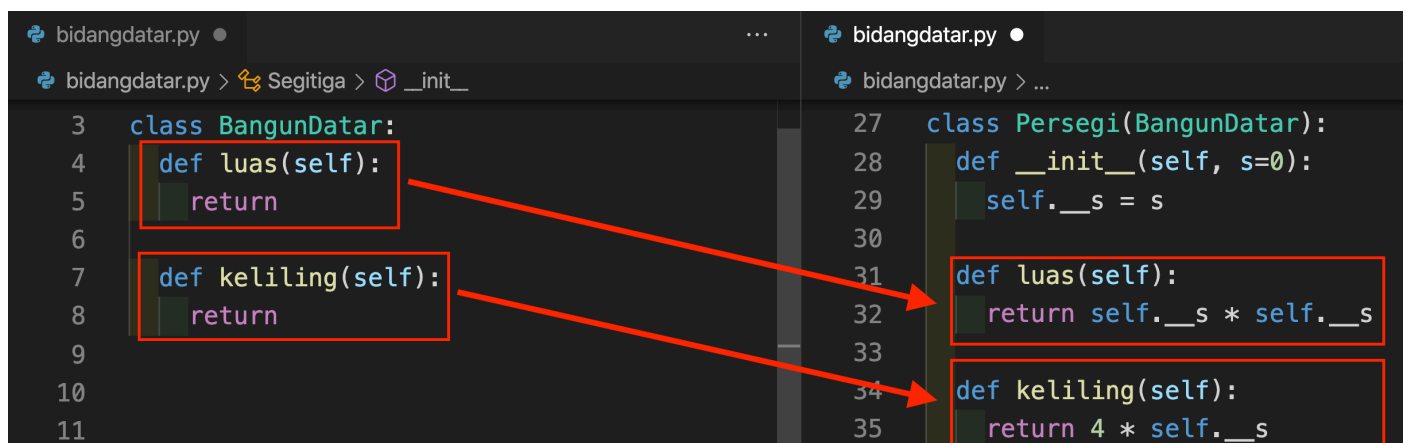
Implementasi **luas()** dan **keliling()** di *class* Segitiga.



```
bidangdatar.py •
bidangdatar.py > Segitiga > __init__
3 class BangunDatar:
4     def luas(self):
5         return
6
7     def keliling(self):
8         return
9
10
11
12
13

bidangdatar.py •
bidangdatar.py > Segitiga
15 class Segitiga(BangunDatar):
16     def __init__(self, a=0, t=0, s=0):
17         self.__a = a
18         self.__t = t
19         self.__s = s
20
21     def luas(self):
22         return (self.__a * self.__t) / 2
23
24     def keliling(self):
25         return self.__a + self.__t + self.__s
```

Implementasi **luas()** dan **keliling()** di *class* Persegi.



```
bidangdatar.py •
bidangdatar.py > Segitiga > __init__
3 class BangunDatar:
4     def luas(self):
5         return
6
7     def keliling(self):
8         return
9
10
11

bidangdatar.py •
bidangdatar.py > ...
27 class Persegi(BangunDatar):
28     def __init__(self, s=0):
29         self.__s = s
30
31     def luas(self):
32         return self.__s * self.__s
33
34     def keliling(self):
35         return 4 * self.__s
```

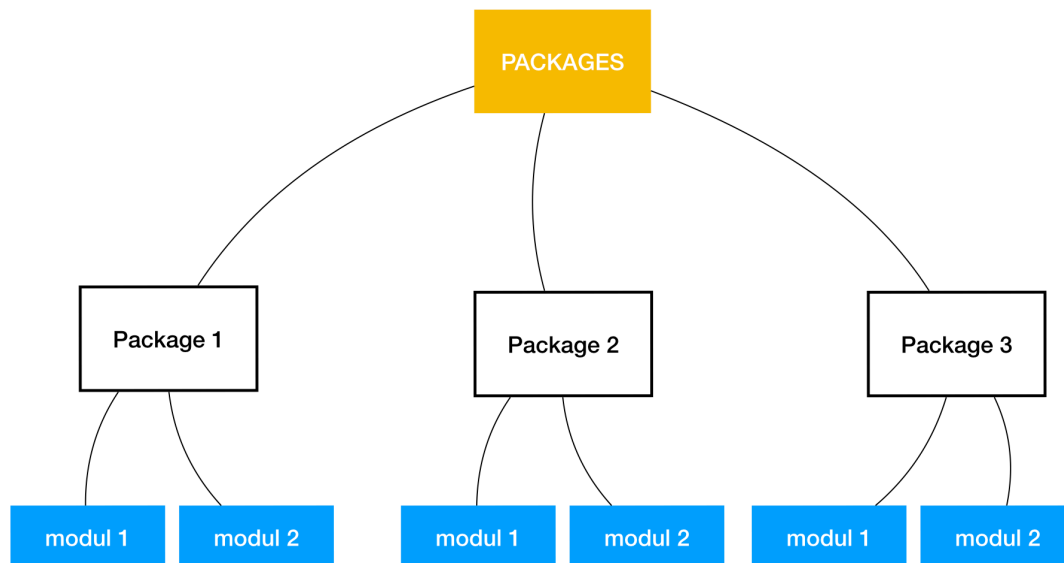
Untuk mempraktikkan **interface**, hapus isi *method* **luas()** dan **keliling()** di *class* BangunDatar, seperti pada gambar diatas.

```
bangundatar.py
class BangunDatar:
    def luas(self):
        return

    def keliling(self):
        return
```

# Package

Package ialah kumpulan modul. Modul ialah *file* program yang berisi kode program, kode tersebut bisa berupa *class*, *method*, struktur data atau *variable* yang bersifat *re-usable*.



Pada contoh kasus sebelumnya, ketiga Class: **BangunDatar**, **Segitiga** dan **Persegi** berada didalam satu *file* yaitu *file* **bangundatar.py**. Pada bagian Package ini kita akan memisahkan ketiga *class* tersebut menjadi 3 *file* yang berbeda. Buat struktur folder dan *file* seperti berikut.



Penjelasan:

- **bangundatar/**: folder ini adalah package
- **\_\_init\_\_.py**: *file* kosong yang memberitahu Python, bahwa folder bangundatar/ adalah package
- **bangundatar.py**: modul, yang akan diisi *class* BangunDatar (*class* induk)
- **persegi.py**: modul, yang akan diisi *class* Persegi (*sub-class*)
- **segitiga.py**: modul, yang akan diisi *class* Segitiga (*sub-class*)
- **main.py**: *file* utama yang akan menjalankan *package*. Dibuat diluar folder **bangundatar/**

Buka folder proyek dengan Visual Studio Code dan isi ketiga *file* dengan *class* yang sesuai dengan nama *file*-nya.

```
# bangundatar.py

class BangunDatar:
    def luas(self):
        return

    def keliling(self):
        return
```

**bangundatar.py**  
Memiliki method **luas()**  
dan **keliling()**

```
# segitiga.py

from .bangundatar import BangunDatar

class Segitiga(BangunDatar):
    def __init__(self, alas=0, tinggi=0, sisi=0):
        self.__alas = alas
        self.__tinggi = tinggi
        self.__sisi = sisi

    def luas(self):
        return (self.__alas * self.__tinggi) / 2

    def keliling(self):
        return self.__alas + self.__tinggi + self.__sisi
```

**segitiga.py**  
Import *class* BangunDatar  
dan masukkan kedalam  
parameter *class* Segitiga

```
# persegi.py

from .bangundatar import BangunDatar

class Persegi(BangunDatar):
    def __init__(self, sisi=0):
        self.__sisi = sisi

    def luas(self):
        return self.__sisi * self.__sisi

    def keliling(self):
        return 4 * self.__sisi
```

**persegi.py**  
Import *class* BangunDatar  
dan masukkan kedalam  
parameter *class* Persegi





```
APLIKASI BANGUN DATAR
=====
1. PERSEGI
2. SEGITIGA
3. SELESAI
```

```
PILIH MENU [1/2/3]:
```

Kita akan membuat program dengan tampilan bergaya menu interaktif seperti disamping saat program dijalankan.



```
# main.py

from bangundatar.bangundatar import BangunDatar
from bangundatar.persegi import Persegi
from bangundatar.segitiga import Segitiga

bangundatar = BangunDatar()

def garis():
    print("=====")

run = True
while run:
    print("APLIKASI BANGUN DATAR")
    garis()
    print("1. PERSEGI")
    print("2. SEGITIGA")
    print("3. SELESAI")

    pilih = input("PILIH MENU [1/2/3]: ")

    if pilih == "1":
        print("PERSEGI")
        sisi = int(input('SISI: '))
        persegi = Persegi(sisi)
        print("Luas:",persegi.luas())
        print("Keliling:",persegi.keliling())
        garis()
    elif pilih == "2":
        print("SEGITIGA")
        alas = int(input('ALAS: '))
        tinggi = int(input('TINGGI: '))
        sisi = int(input('SISI MIRING: '))
        segitiga = Segitiga(alas, tinggi, sisi)
        print("Luas:",segitiga.luas())
        print("Keliling:",segitiga.keliling())
        garis()
    elif pilih == "3":
        run = False
    else:
        print("Tidak ada menu:",pilih)
```

Buka *file* **main.py** lalu isi dengan kode seperti disamping.

# 5

## **MENGKOMPILASI PROGRAM**

# Menjalankan Program

BAB 5 ini lebih tepat diberi judul “Menjalankan Program”, karena Python adalah bahasa pemrograman Interpreter, bukan Kompiler. Maka lebih tepat disebut Menjalankan Program 😊.

File yang dijalankan hanya ada satu yaitu *file* **main.py**. Pada Visual Studio Code pilih menu **Terminal** -> **New Terminal** untuk membuka Cmd.

Jalankan program dengan perintah **python main.py**

```
c:\UKK-HILMI> python main.py
```

```
APLIKASI BANGUN DATAR
```

```
=====
```

```
1. PERSEGI  
2. SEGITIGA  
3. SELESAI
```

```
PILIH MENU [1/2/3]:
```

## Mengoreksi Kesalahan

Teknik cepat untuk mengetahui kesalahan pada kode Python dilihat dari:

1. Nama *file*-nya apa?
2. Baris beberapa?
3. Kode *error*-nya apa?

```
c:\UKK-HILMI> python main.py
```

```
python main.py
```

```
File "main.py", line 39
```

```
    print("Tidak ada menu:",pilih)
```

```
    ^
```

```
IndentationError: expected an indented block
```

Error berada di:

- file **main.py**

- baris ke **39**

- kode *error*-nya: **indentation**

# Referensi

1. Skema SKKNI Level II RPL
2. SKKNI 2016-282
3. Buku Pengantar Python 3
4. Buku Pemrograman Berorientasi Objek kelas XI
5. <https://realpython.com>