

Program Fresh Graduate Academy Digital Talent Scholarship 2019 | Machine Learning

Pemrograman Python : Fungsi

M. Ramli & M. Soleh





Bagian 1

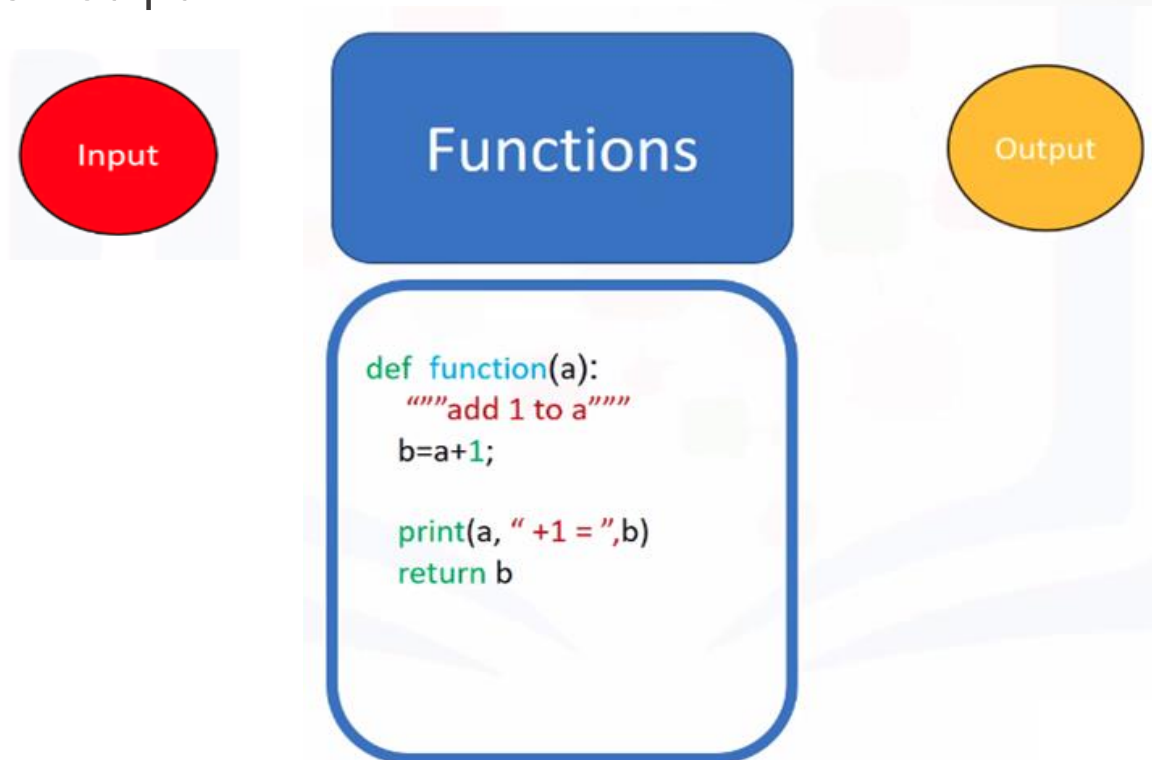
Fungsi



Fungsi

FUNGSI

Merupakan bagian program (block baris program) yang dapat dibuat berdasarkan kebutuhan dan dapat digunakan kembali (dipanggil kembali) pada bagian program yang lainnya. Input akan masuk ke dalam **"Fungsi"** dan menghasilkan output.

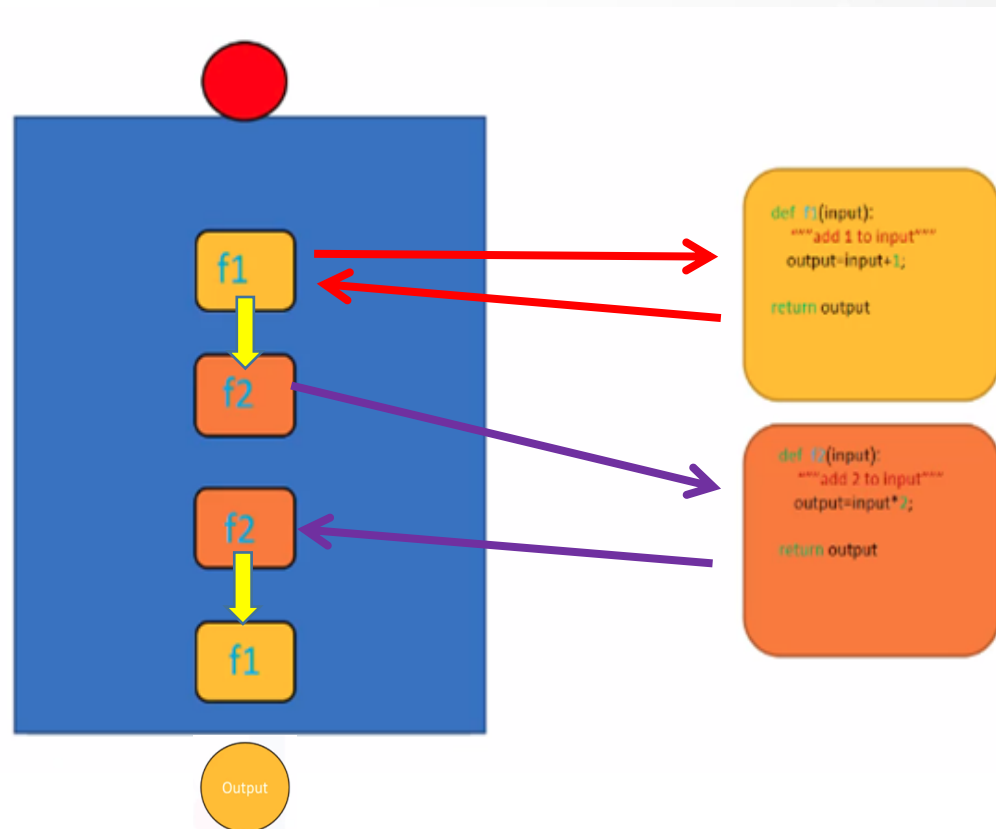


Fungsi

“Fungsi” merupakan block program yang dapat dipanggil berulang kali.

“Fungsi” menerima input dan akan menghasilkan output.

Output yang dihasilkan dari “Fungsi” dapat digunakan pada block program yang lainnya.





Bagian 2

Fungsi yang Sudah Tersedia di Python (Built in Function)

Fungsi Built in

“Fungsi Built in” merupakan fungsi yang sudah ada di Python dan dapat langsung digunakan.

Fungsi yang bisa langsung digunakan diantaranya adalah:

- Len, mendapatkan banyak anggota/elemen
- Sum, digunakan untuk menjumlahkan
- Sorted, digunakan untuk mengurutkan

Fungsi Built in

Fungsi “LEN”

Digunakan untuk mendapatkan banyak anggota/elemen.

Contoh :

```
album_ratings = [10.0,8.5,9.5,7.0,7.0,9.5,9.0,9.5]
```

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

```
L=len(album_ratings)
```

L:8

len

8



Fungsi Built in

Fungsi “**SUM**”

digunakan untuk menjumlahkan semua nilai yang berada dalam list.

Contoh :

```
album_ratings = [10.0,8.5,9.5,7.0,7.0,9.5,9.0,9.5]
```

```
10.0+8.5+9.5+7.0+7.0+9.5+9.0+9.5
```

```
S=sum(album_ratings)
```

```
S:70
```

sum

70



Fungsi Built in

Fungsi “**SORTED**”

Digunakan untuk mengurutkan data yang ada pada list. Hasil sort disimpan pada variabel baru, sehingga tidak mengubah data pada list awal.

Contoh :

```
album_ratings = [10.0,8.5,9.5,7.0,7.0,9.5,9.0,9.5]
```

Variabel baru

```
sorted_album_rating = sorted(album_ratings)
```

```
sorted_album_rating:
```

```
[7.0, 7.0, 8.5, 9.0, 9.5, 9.5, 9.5, 10.0]
```

```
album_ratings :
```

```
[10.0,8.5,9.5,7.0,7.0,9.5,9.0,9.5]
```

sorted

[7.0, 7.0, 8.5, 9.0, 9.5, 9.5, 9.5, 10.0]

Fungsi Built in

Beda Fungsi “**SORTED**” vs SORT

Fungsi “Sorted” hasilnya disimpan pada variabel baru, sedangkan

Metode sort dilakukan langsung pada data yang ada di list, dan tidak menggunakan variabel baru, sehingga data di variabel akan berubah.

```
album_ratings = [10.0,8.5,9.5,7.0,7.0,9.5,9.0,9.5]
```

```
album_ratings.sort()
```

```
album_rating:
```

```
[7.0, 7.0, 8.5, 9.0, 9.5, 9.5, 9.5, 10.0]
```

album_ratings



Bagian 3

Membuat Fungsi



Membuat Fungsi

“Fungsi” juga dapat dibuat sendiri sesuai dengan kebutuhan.

Aturan untuk membuat “Fungsi” adalah:

- Membuat “Fungsi” pada Python didahului dengan perintah “**def**” dan ditutup dengan “**return**” untuk mengirimkan nilai
- **Nama “Fungsi”** harus dideskripsikan dengan jelas, karena pemanggilan fungsi dilakukan dengan menggunakan Nama Fungsi ini.
- Tentukan **parameter** yang dituliskan setelah Nama fungsi
- Parameter dituliskan antara tanda kurung (...)
- Setelah penulisan parameter, diberikan tanda **titik dua (:)**

Bentuk Umum pembuatan “Fungsi”:

```
Def nama_fungsi (parameter):  
    Code program sesuai kebutuhan  
    .....  
    .....  
Return variabel_untuk_nilai_hasil_operasi
```

Membuat Fungsi

Contoh penulisan “Fungsi” yang dibuat sendiri:

Nama Fungsi

Formal parameter

```
def add1(a):  
    b=a+1  
    return b
```

code block

Membuat Fungsi

Contoh pemanggilan “Fungsi” :

Penulisan “Fungsi”

`def add1(a):`

`b=a+1` → `b = 5 + 1`

`return b` → `b = 6`

Pemanggilan “Fungsi”

`c=add1(5)`

Akan
diberikan
sebagai nilai
untuk
parameter **a**
yang ada di
Fungsi

Hasil pemanggilan “Fungsi”

`c:6`

Membuat Fungsi

“Fungsi” dapat didokumentasikan, yang berguna untuk memberikan penjelasan dari fungsi tersebut.

Dokumentasi pada fungsi berada diawali dengan tanda kutip sebanyak tiga (“’’’’”), dan juga ditutup dengan tanda yang sama.

Contoh penulisan dokumentasi:

```
def add1(a):
```

```
    """  
    add 1 to a  
    """
```

Documentation
String

```
    b=a+1;  
    return b
```

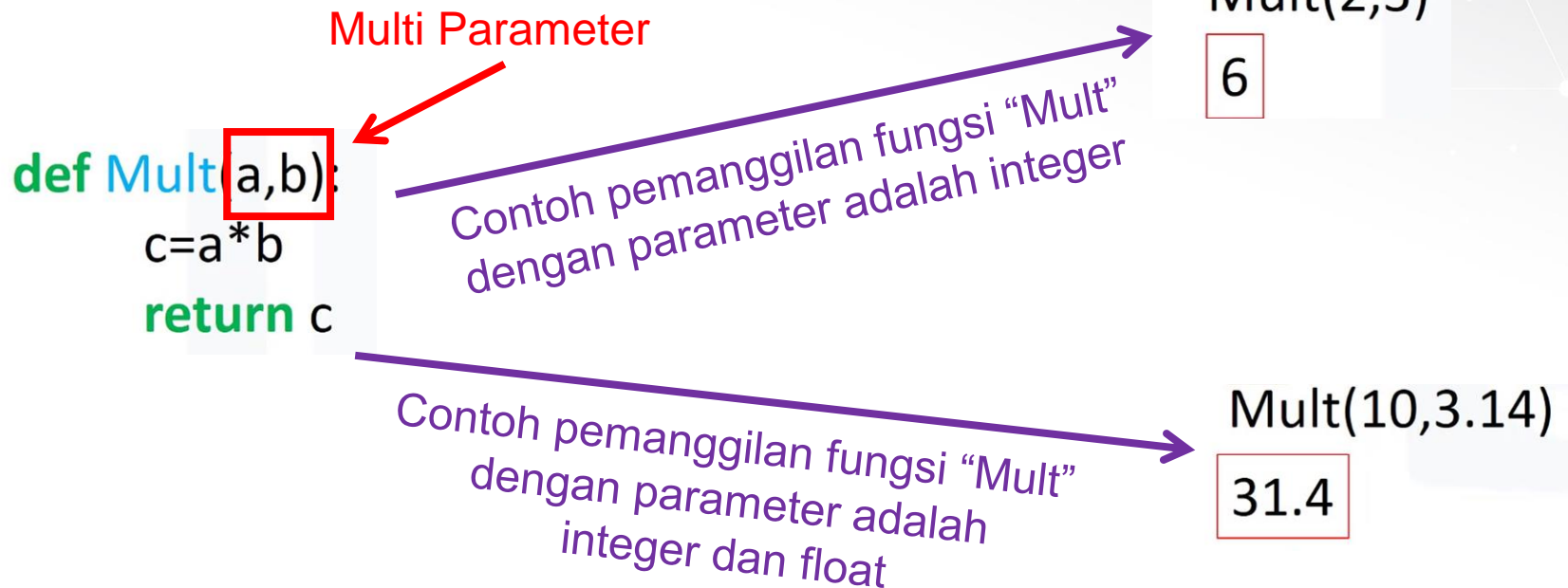
```
help(add1)
```

Dokumentasi dapat dipanggil
dengan “**help**”

```
Help on function add1 in module __main__: add1(a) add 1 to a
```


Membuat Fungsi

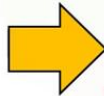
“Fungsi” juga dapat memiliki **Multi Parameter** (lebih dari satu parameter).



Membuat Fungsi

“Fungsi” juga dapat memiliki **Multi Parameter** (lebih dari satu parameter).
Dan parameter yang dikirimkan dapat berupa strings.

```
def Mult(a,b):  
    c=a*b  
    return c
```



```
2*"Michael Jackson "
```

```
"Michael Jackson Michael Jackson "
```

```
Mult(2, "Michael Jackson ")
```

```
"Michael Jackson Michael Jackson "
```

Membuat Fungsi

“Fungsi” juga dapat dibuat TANPA menggunakan “return”.

“Fungsi”

Pemanggilan “Fungsi” & Output

```
def MJ():  
    print('Michael Jackson')
```



MJ()

'Michael Jackson'

← Output

```
def NoWork():  
    pass
```



print(NoWork())

None

← Output

Python TIDAK mengizinkan block fungsi kosong, untuk mengatasinya dapat digunakan kata “**pass**”

Membuat Fungsi

Block “Fungsi” dapat berisi lebih dari satu baris perintah/kode program.

```
def add1(a):
```

```
    b=a+1;
```

```
    print(a, “plus 1 equals ”,b)
```

```
    return b
```

Lebih dari satu baris
Perintah/
Kode program



Membuat Fungsi

Block “Fungsi” dapat berisi lebih dari satu baris perintah/kode program.

“Fungsi”

```
def add1(a):
    b=a+1;
    print(a, "plus 1 equals ",b)
    return b
```

Pemanggilan “Fungsi”

add1(2)

Output:

2 plus 1 equals 3
3

Proses pada Fungsi adalah:

a	2
b	3
output of print(...	2 plus 1 equals 3
value returned	3

Membuat Fungsi

Block “Fungsi” dapat berisi lebih dari satu baris perintah/kode program, dan dapat berisi adanya perulangan (contoh menggunakan FOR).

```
def printStuff(Stuff):
```

```
    for i,s in enumerate(Stuff):
```

```
        print("Album", i, "Rating is ", s)
```

Block Fungsi
dengan nama “**printstuff**”

```
album_ratings = [10.0,8.5,9.5]
```

```
printstuff(album_ratings)
```

Program Utama

→ Perintah pemanggilan Fungsi



Membuat Fungsi

Contoh block “Fungsi” yang terdapat perintah perulangan:

Block “Fungsi”

```
def printStuff(Stuff):
```

```
    for i,s in enumerate(Stuff):
```

```
        print("Album", i, "Rating is ", s)
```

Program Utama

```
album_ratings = [10.0,8.5,9.5]
```

```
printstuff(album_ratings)
```

Perintah pemanggilan Fungsi

Pada perintah pemanggilan “Fungsi”, dikirimkan data list dari “album_ratings” ke parameter “Stuff” dari fungsi “printstuff”

Membuat Fungsi

Contoh: (lanjutan 1)

```
def printStuff(Stuff):
```

```
    for i,s in enumerate(Stuff):
```

```
        print("Album", i, "Rating is ", s)
```

```
album_ratings = [10.0,8.5,9.5]  
printstuff(album_ratings)
```

Stuff: [10.0, 8.5, 9.5]

Index:

0	1	2
---	---	---

Output:

Album 0 Rating is 10

Membuat Fungsi

Contoh: (lanjutan 2)

```
def printStuff(Stuff):
```

```
    for i,s in enumerate(Stuff):
```

```
        print("Album", i, "Rating is ", s)
```

```
album_ratings = [10.0,8.5,9.5]  
printstuff(album_ratings)
```

Stuff: [10.0, 8.5, 9.5]

Index:

0	1	2
---	---	---

Output:

Album 0 Rating is 10

Album 1 Rating is 8.5

Membuat Fungsi

Contoh: (lanjutan 3)

```
def printStuff(Stuff):
```

```
    for i,s in enumerate(Stuff):
```

```
        print("Album", i, "Rating is ", s)
```

```
album_ratings = [10.0,8.5,9.5]  
printstuff(album_ratings)
```

Stuff: [10.0, 8.5, 9.5]

Index:

0	1	2
---	---	---

Output:

Album 0 Rating is 10

Album 1 Rating is 8.5

Album 2 Rating is 9.5

Membuat Fungsi

Collecting Argument

Variadic parameter adalah satu parameter di “Fungsi” yang dapat menerima lebih dari satu nilai elemen yang dikirimkan ke parameter tersebut ketika Fungsi nya dipanggil. Dalam penulisannya menggunakan fungsi asterik (*) sebelum nama parameter.

```
def ArtistNames(*names):
```

Variadic Parameter dengan tanda asterik (*) yang merupakan collecting argument

```
    for name in names:
```

```
        print(name)
```

names=("Michael Jackson","AC/DC","Pink Floyd")

```
ArtistNames("Michael Jackson","AC/DC","Pink Floyd")
```

Perintah pemanggilan Fungsi

Akan dikirimkan ke parameter di “Fungsi”

Membuat Fungsi

Collecting Argument

Contoh ke-2:

```
def ArtistNames (*names):
```

```
    for name in names:
```

```
        print(name)
```



```
names=("Michael Jackson","AC/DC")
```

```
ArtistNames("Michael Jackson","AC/DC")
```

Membuat Fungsi

Scope

Scope/ruang lingkup suatu variabel adalah bagian dari program di mana variabel itu dapat diakses.

Lingkup variabel ada 2 (dua), yaitu: Global dan Lokal.

Block Fungsi

```
def AddDC(y):  
    x=x+"DC"  
    print(x)  
    return(x)
```

Variabel yang didefinisikan di luar fungsi apa pun dikatakan berada dalam lingkup global, artinya dapat diakses di mana saja setelah didefinisikan

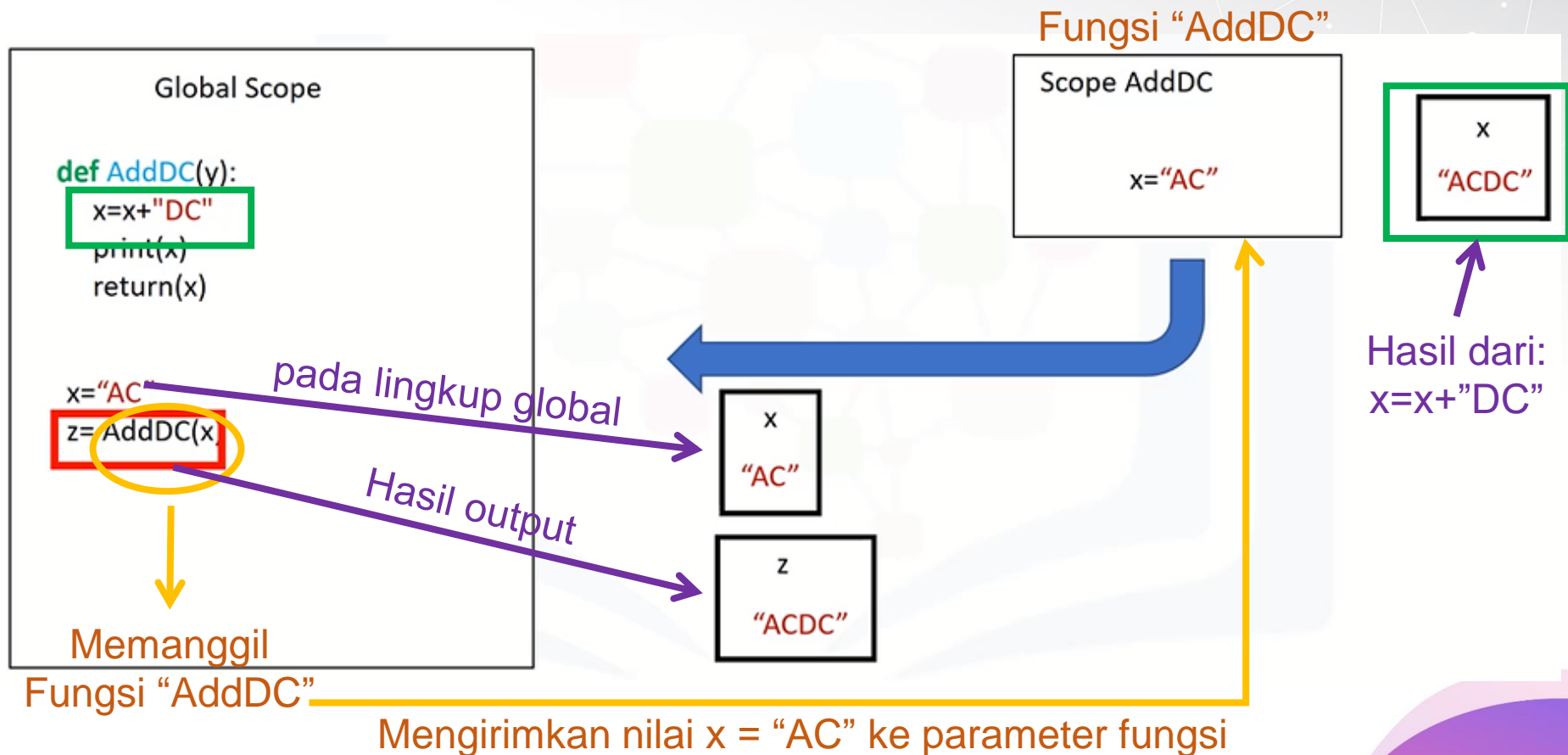
Program Utama

```
x="AC"  
z= AddDC(x)
```

Variabel ini ada pada lingkup global

Membuat Fungsi

Variabel yang ada dalam lingkup global disebut Variabel Global





Membuat Fungsi

Variabel lokal adalah variabel yang berada dalam lingkup lokal (Fungsi)

Variabel ini
ada pada lingkup lokal (Fungsi)



```
def Thriller():  
    Date=1982  
    return Date
```

Block Fungsi

```
Thriller()
```

Program Utama

Membuat Fungsi

Variabel lokal dan variabel global DAPAT memiliki **nama variabel** yang SAMA, jika variabel global tidak digunakan sebagai nilai ke parameter yang ada di fungsi.

Variabel lokal hanya memberikan nilai pada lokal/fungsinya saja, dan tidak mengganggu nilai pada variabel global.

Variabel Lokal →

```
def Thriller():  
    Date=1982  
    return (Date)
```

 } Block Fungsi

Variabel Global →

```
Date=2017
```

```
print(Thriller())  
print(Date)
```

 } Program Utama

Output:

1982
2017

Membuat Fungsi

Jika variabel tidak didefinisikan dalam suatu fungsi, Python akan memeriksa lingkup global.

Variabel "Rating"
TIDAK DIDEFINISIKAN
Pada block fungsi

```
def ACDC(y):  
    print(Rating)  
    return (Rating+y)
```

Block Fungsi

Rating=9

Z=ACDC(1)

print(Rating)

Program Utama

Output:

9
Z=10
9



Membuat Fungsi

Dapat menggunakan fungsi “global” untuk mendefinisikan suatu variabel berlaku global.

```
def PinkFloyd():  
    global ClaimedSales  
    ClaimedSales = '45 million'  
    return ClaimedSales
```

PinkFloyd()

print(ClaimedSales)

Output:

45 milion

Program Fresh Graduate Academy Digital Talent Scholarship 2019 | Machine Learning

Pemrograman Python: Object dan Class

M. Ramli & M. Soleh





Bagian 1

Tipe Data pada Python

Tipe Data pada Python

Tipe Data pada Python

Python memiliki banyak tipe data, seperti: integer, float, boolean.

Masing-masing tipe disebut sebagai object.

Berikut ini adalah tipe data pada Python:

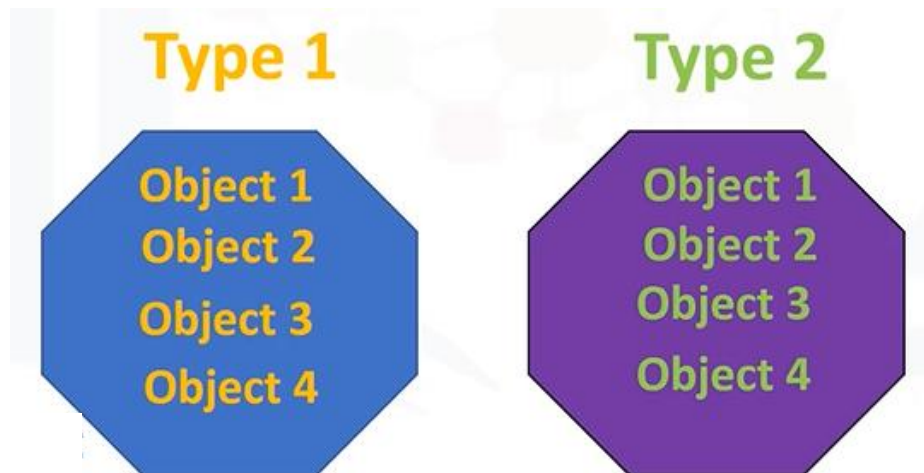
- Types:
 - int: `1, 2, 567..`
 - float: `1.2, 0.62..`
 - String: `'a', 'abc', 'The cat is yellow'`
 - List: `[1, 2, 'abc']`
 - Dictionary: `{"dog": 1, "Cat": 2}`
 - Bool: `False, True`

Tipe Data pada Python

Object merupakan instance dari tipe yang khusus.

Setiap object memiliki:

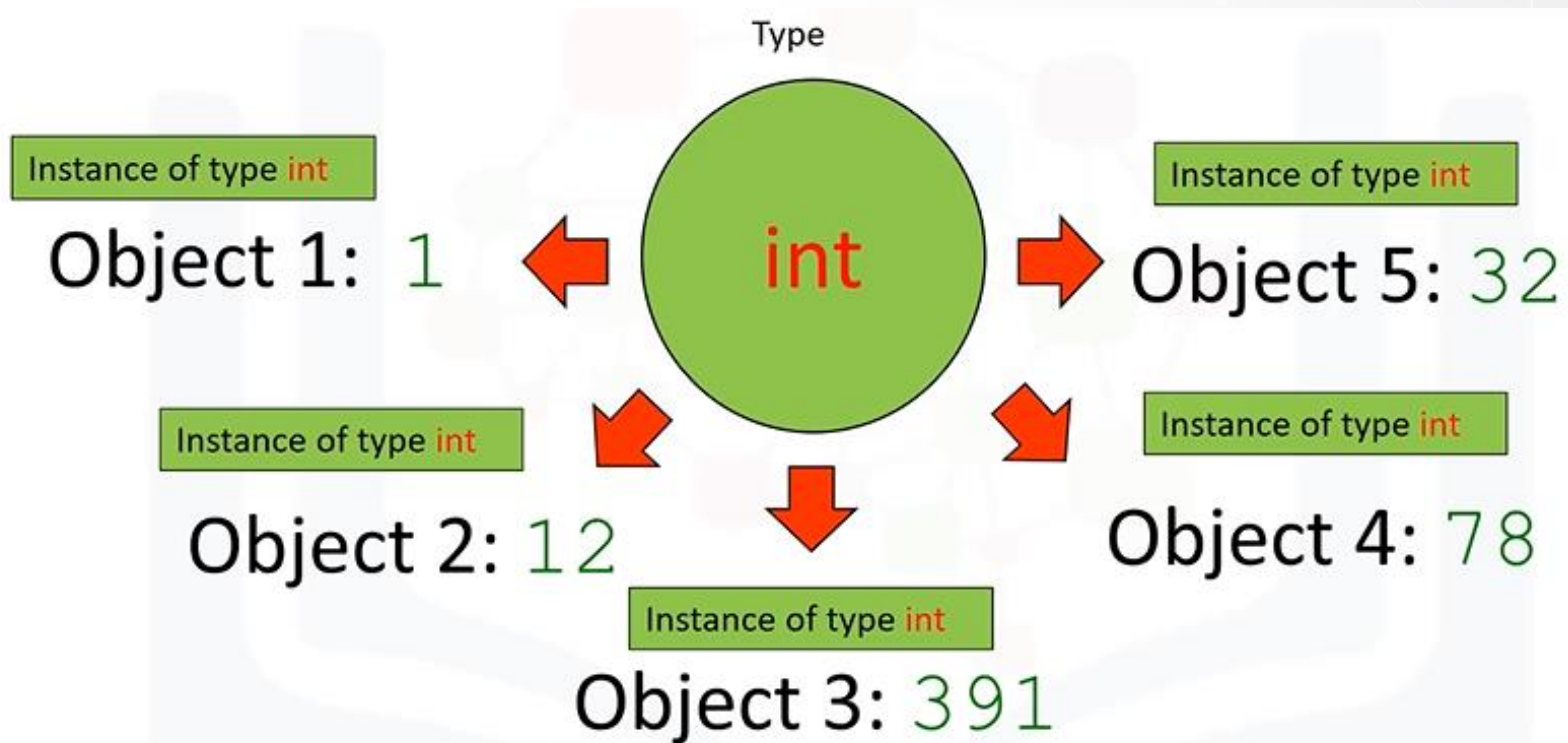
1. Tipe data
2. Representasi data internal (blueprint)
3. Kumpulan prosedur untuk interaksi antar object, disebut sebagai “Metode”



Object: Type

Contoh:

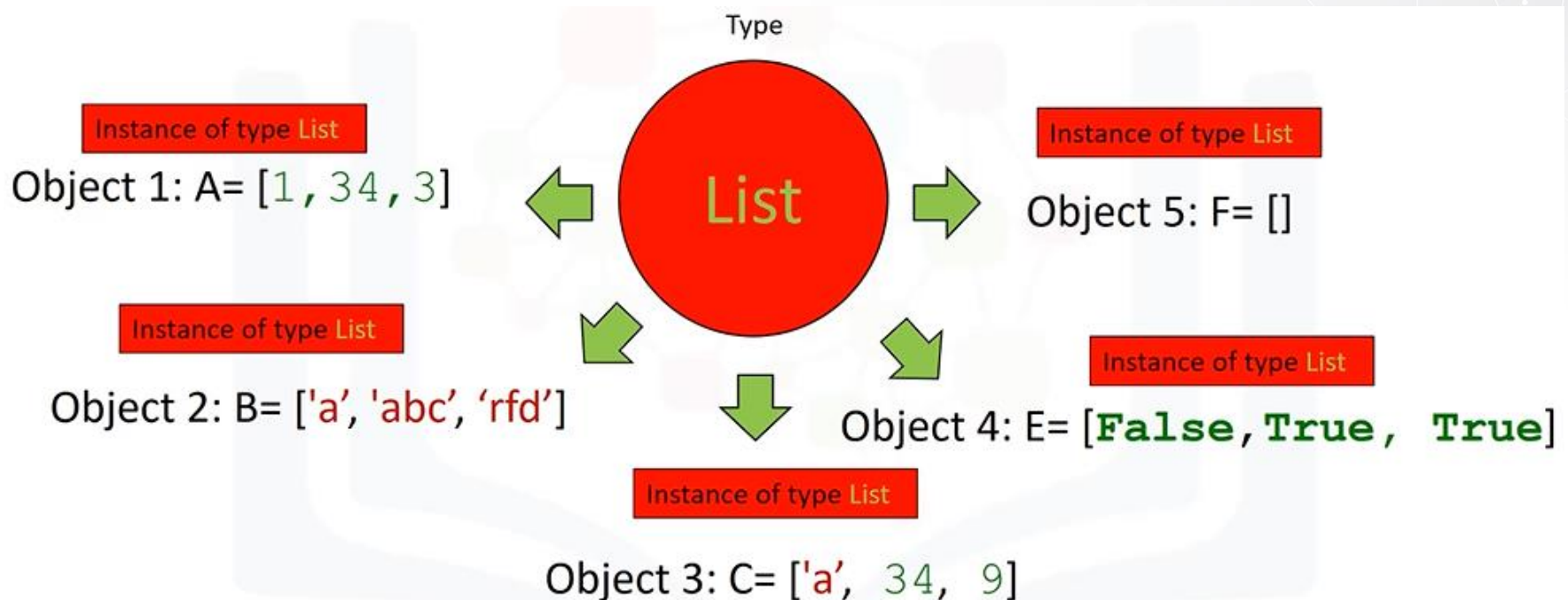
Tipe “integer” memiliki 5 (lima) object, seperti berikut:



Object: Type

Contoh:

Tipe “List” memiliki 5 (lima) object seperti berikut ini:



Object: Type

Tipe dari object dapat diketahui dengan menggunakan perintah “**type()**”

```
>>type([1, 34, 3])  
<class 'list'>
```

Instance of type List

List

```
>>type('The cat is yellow' )  
<class 'str'>
```

Instance of type str

str

```
>>type(1)  
<class 'int'>
```

Instance of type int

int

```
>>type( {"dog": 1, "Cat": 2})  
<class 'dict'>
```

Instance of type dict

dict

Metode

Metode

Class atau metode dari tipe berfungsi pada setiap instance/isi dari class atau tipe tersebut.

Metode memberikan cara bagaimana berinteraksi dengan data dari object.

Contoh dari “**Metode**” adalah pengurutan (sorting).

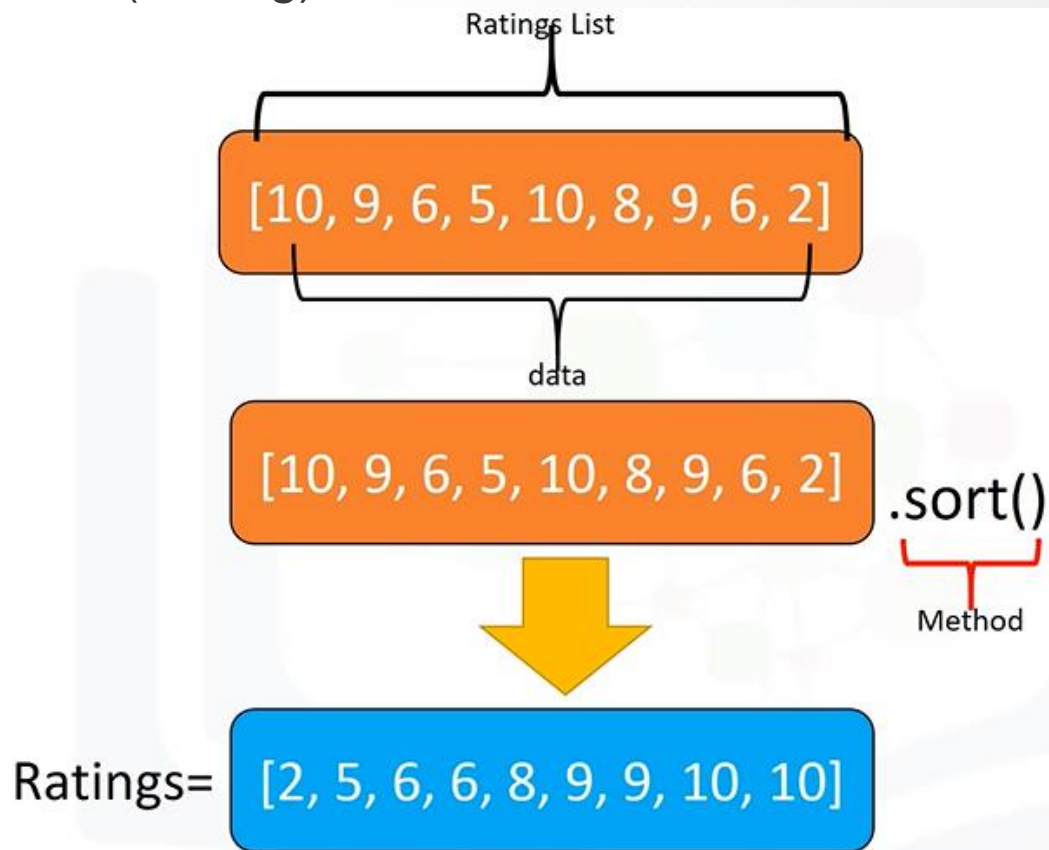
```
Ratings=[10, 9, 6, 5, 10, 8, 9, 6, 2]
```

```
Ratings.sort()
```

Metode

Contoh:

Metode pengurutan (sorting)



Metode

Contoh ke-2:

Metode “**Reverse**” yang dapat digunakan untuk membalik urutan dari suatu list yang sudah diurutkan dengan metode sort

Ratings=

[2, 5, 6, 6, 8, 9, 9, 10, 10]

[2, 5, 6, 6, 8, 9, 9, 10, 10]

.reverse()

Method

Ratings=

[10, 10, 9, 9, 8, 6, 6, 5, 2]



Bagian 2

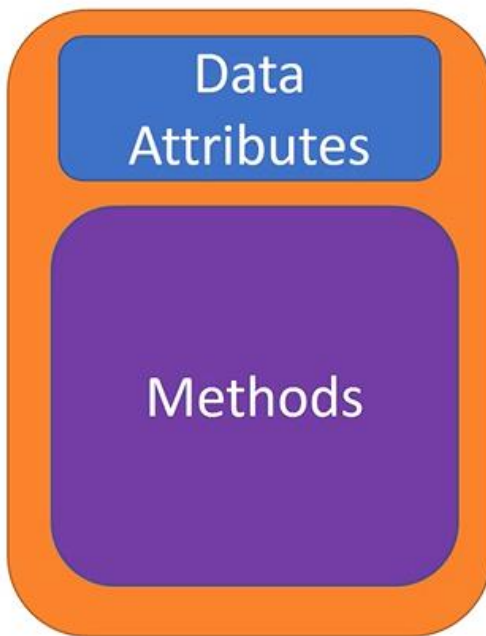
Membuat Type Sendiri dengan Mendefinisikan Class

Membuat Type

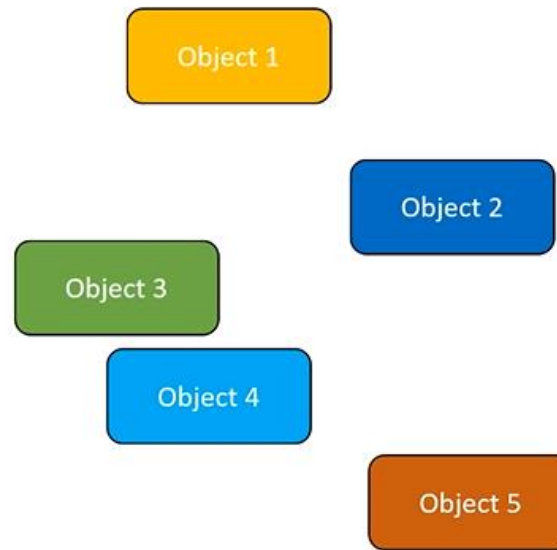
Class terdiri dari: atribut data dan metode.

Object/instance dari class dapat berisi lebih dari satu.

Class



Objects or Instances of that Class





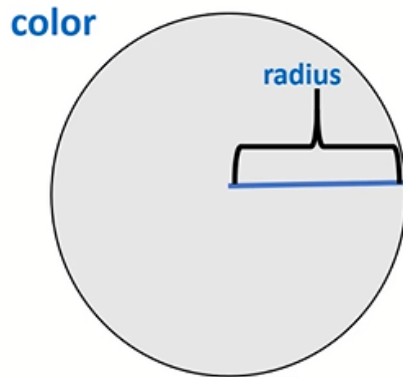
Membuat Type

Berikut ini contoh pada **class** "Circle" dan **class** "Rectangle"

Pada class "Circle", atribut datanya adalah radius dan color

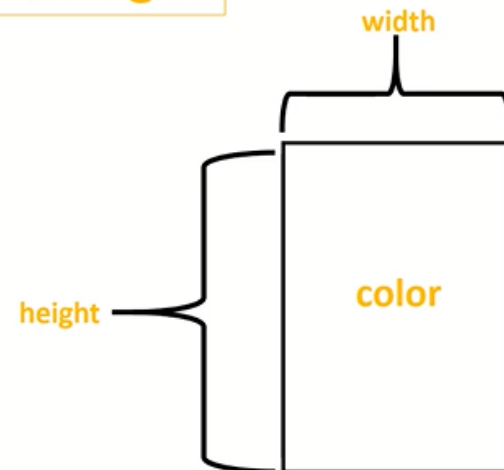
Pada class "Rectangle", atribut datanya adalah color, height, dan width

Class Circle



Data Attributes: **radius**, **color**

Class Rectangle



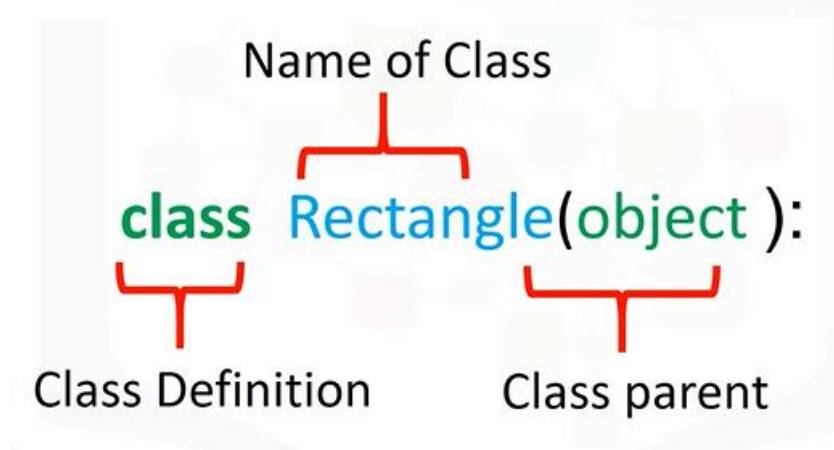
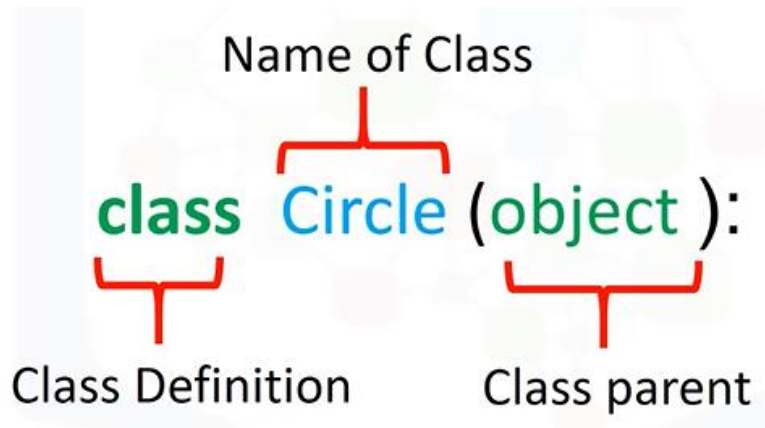
Data Attributes: **color**, **height** and **width**

Membuat Class

Berikut ini adalah 2 (dua) contoh membuat Class:

Contoh ke-1 nama class adalah “Circle”

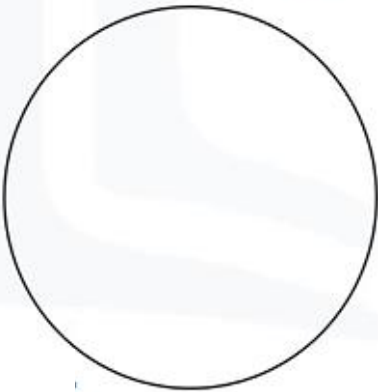
Contoh ke-2 nama class adalah “Rectangle”



Membuat Type

Definisi dari Class

```
class Circle(object):
```



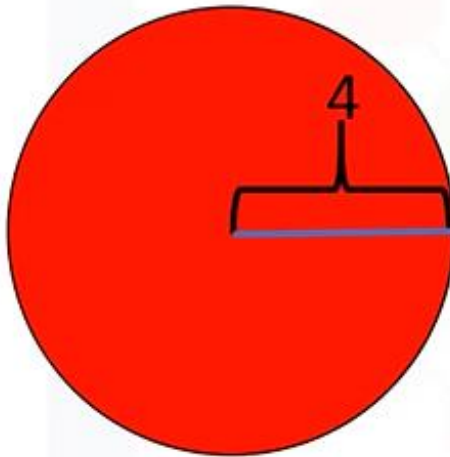
```
class Rectangle(object):
```



Membuat Type

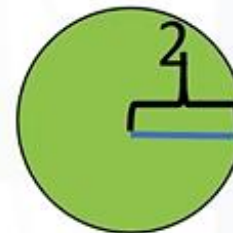
Attributes and Objects

Berikut adalah 2 contoh object yang termasuk pada class “Circle”



Object 1: Instance of type Circle

Data Attributes:
radius=4
color=red



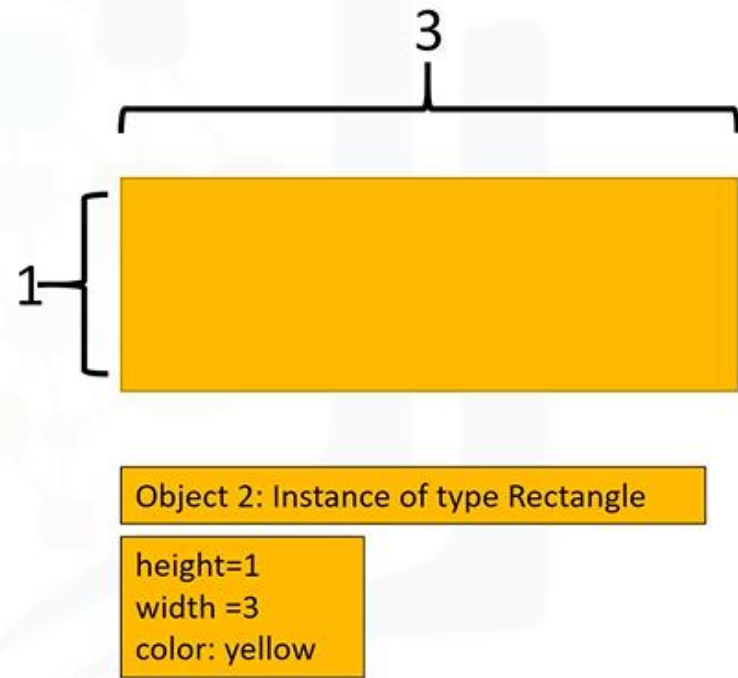
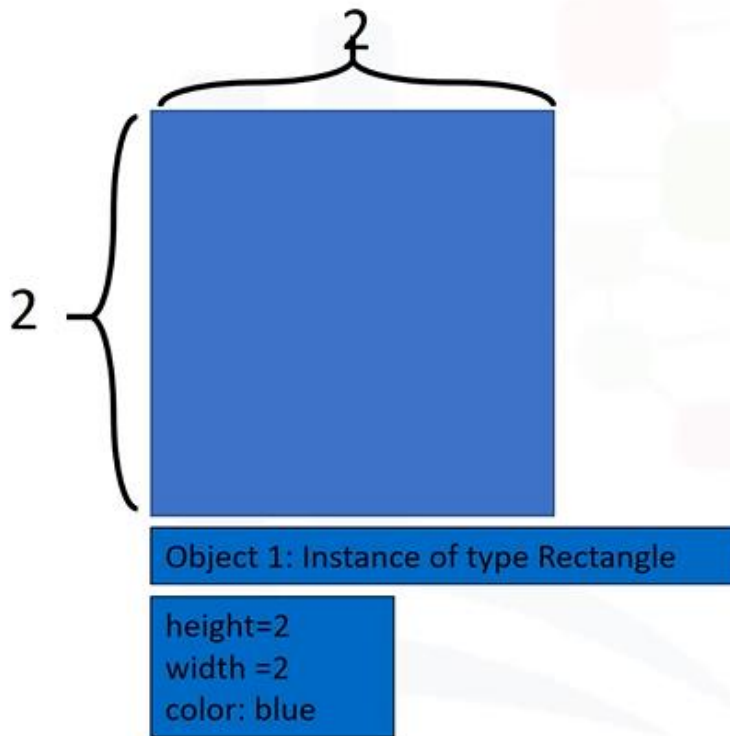
Object 2: Instance of type Circle

Data Attributes:
radius=2
color=green

Membuat Type

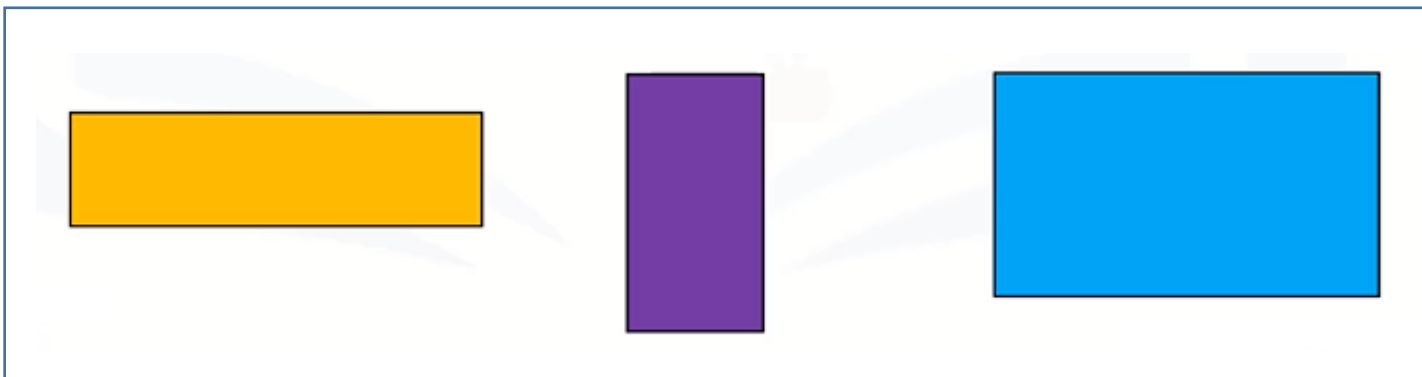
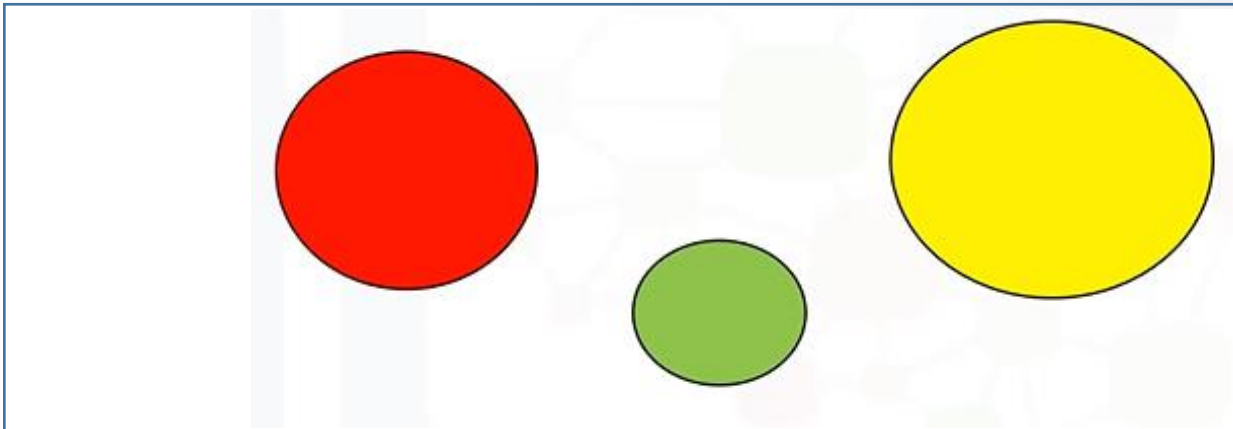
Attributes and Objects

Berikut adalah 2 contoh object yang termasuk pada class “Rectangle”



Membuat Type

Berikut adalah contoh object-object pada class “Circle” (type circle) dan contoh object-object pada class “Rectangle” (type Rectangle)



Membuat Type

Berikut ini adalah perintah pada Python untuk membuat class.
Contoh pada pendefinisian class dengan nama “Circle”

```
class Circle (object):
```

} Define your class

```
def __init__(self, radius , color):  
    self.radius = radius;  
    self.color = color;
```

} Data attributes used to
Initialize each instance of
the class

Membuat Type

Bentuk Umum pendefinisian atribut data pada class

special method or constructor used to initialize data attributes
parameters

```
def __init__(self, radius, color):
```

The self parameter

```
    self.radius = radius;  
    self.color = color;
```


Membuat Type

Berikut ini contoh membuat class dengan nama “Rectangle” dan melakukan pendefinisian atribut class dengan menggunakan `def __init__`

```
class Rectangle (object):
```

} Define your class

```
def __init__(self, color, height , width):
```

```
    self.height = height;
```

```
    self.width = width
```

```
    self.color = color;
```

} Initialize the object's
Data attributes

Membuat Type

Berikut adalah contoh membuat object dari class “Circle”.

Pembuatan object dilakukan dengan memberikan instance/isi dari atribut object tersebut.

```
RedCircle = Circle (10, "red")
```

Name of Class

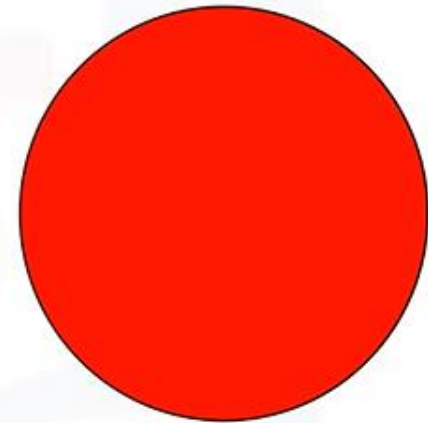
Attributes

Membuat Type

Contoh object C1 dari class "Circle

```
C1=Circle (10,'red' )
```

```
class Circle (object):  
    def __init__(self, 10, 'red'):  
        self.radius = 10;  
        self.color = 'red';
```



```
self .radius = 10;  
self. color = 'red';
```

Membuat Type

Contoh pemberian instance/isi pada atribut dari object suatu class.

```
class Circle (object):
```

```
    def __init__(self, radius , color):  
        self.radius = radius;  
        self.color = color;
```

self

self.radius = 10
self.color = 'red'

Membuat Type

Mengetahui nilai atribut dari suatu object dapat dilakukan dengan perintah:
`nama_object.nama_atribut`

```
C1=Circle (10, "red")
```

```
C1.radius
```

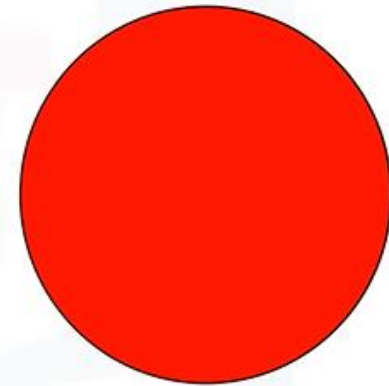
```
10
```

```
C1.color
```

```
"red"
```

```
C1
```

```
C1.radius = 10  
C1.color = 'red'
```



```
self.radius = 10;  
self.color = 'red';
```

Membuat Type

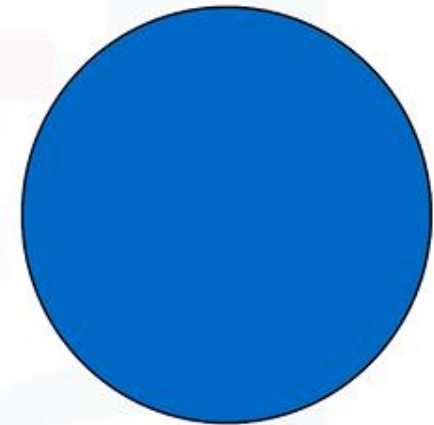
Mengetahui nilai atribut dari suatu object dapat dilakukan dengan perintah:
`nama_object.nama_atribut`

```
C1=Circle (10, "red")
```

```
C1.color="blue"
```

```
C1.color
```

```
"blue"
```



```
self.radius = 10;  
self.color = 'red';
```



Bagian 3

Method (Metode)

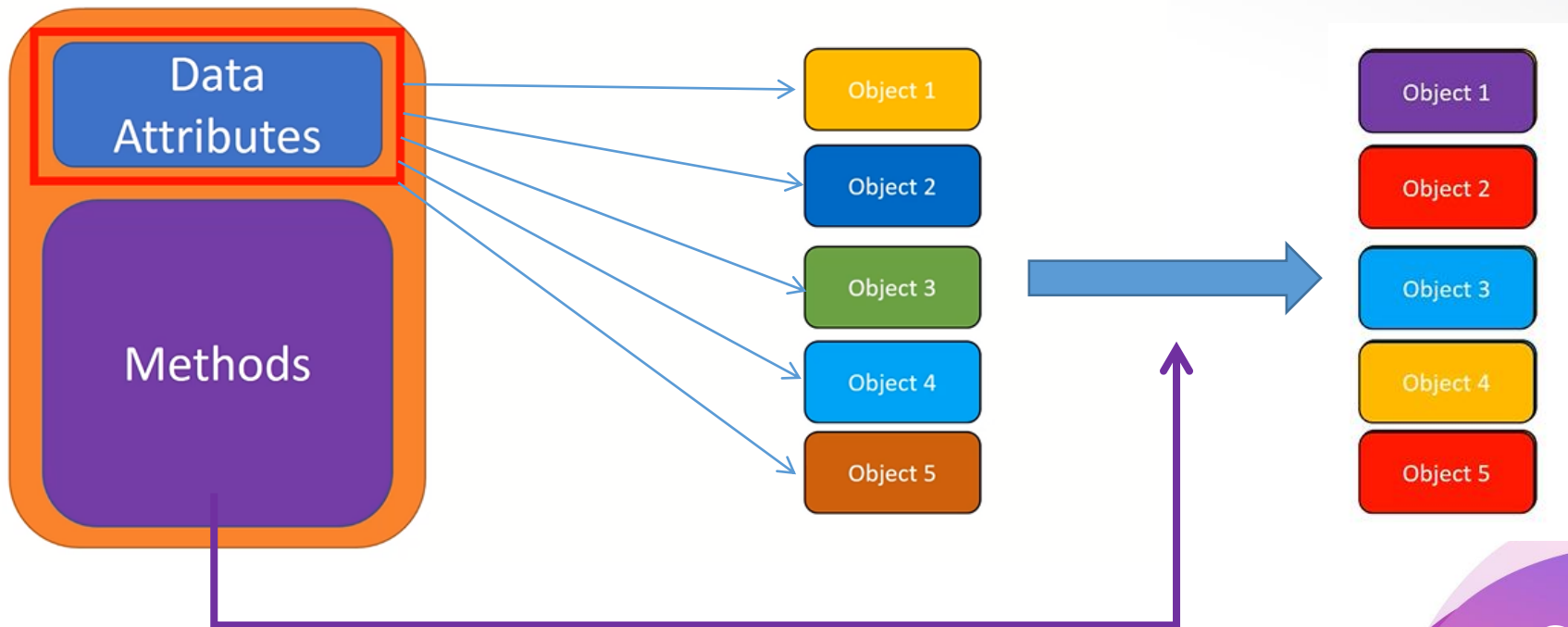
Metode

Class terdiri dari atribut data dan metode.

Metode akan mengubah object awal menjadi object baru sesuai dengan perintah pada metode tersebut.

Class

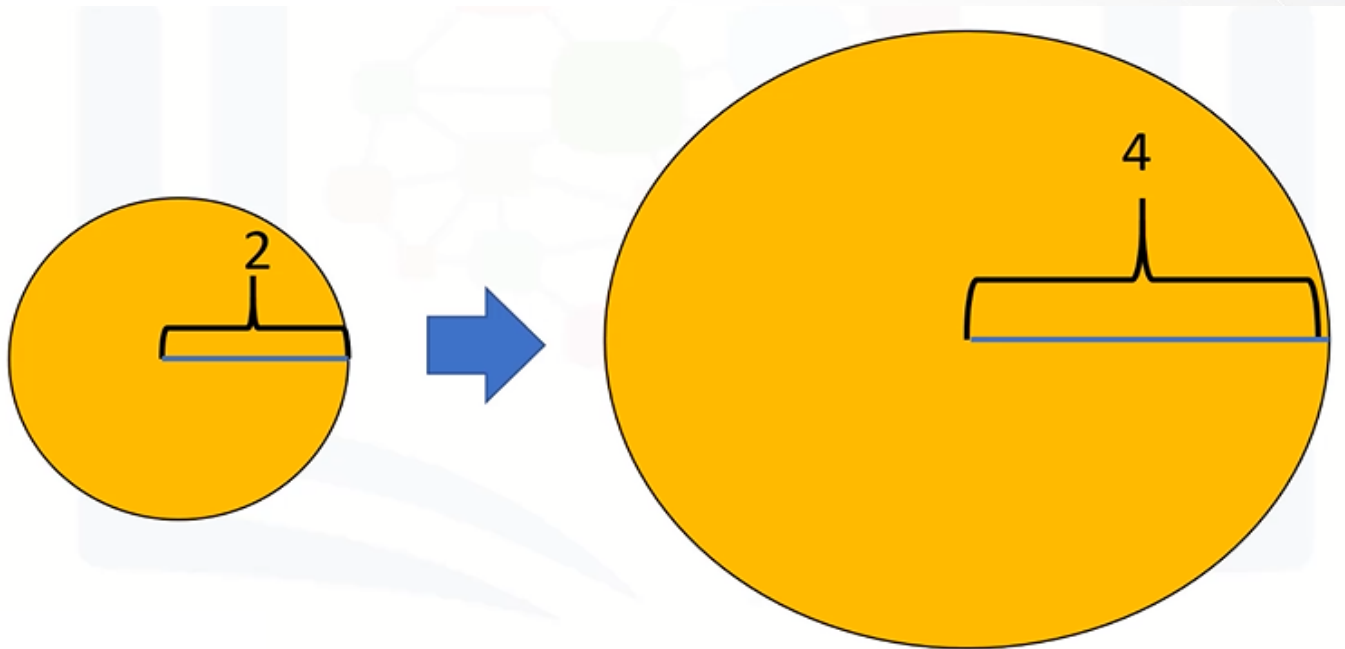
Objects or Instances of that Class



Metode

Contoh:

Object pada class “Circle” akan diubah dari object awal yang memiliki radius = 2 menjadi object baru yang memiliki radius = 4



Metode

Metode pada Class

```
class Circle(object):  
    def __init__(self, radius, color):  
        self.radius = radius;  
        self.color = color;  
  
    def add_radius(self, r):  
        self.radius = self.radius + r
```

} Method used to add r to radius

Metode

Contoh:

Object C1 dengan nilai atribut adalah 2 dan red

```
C1=Circle (2,' red ' )
```

```
def __init__(self, radius , color):  
    self .radius = radius;  
    self. color = radius;
```

```
C1=Circle (2,' red ' )
```

```
def __init__(self, 2, 'red'):  
    self .radius = 2;  
    self. color = 'red ';
```

Metode

Contoh:

Object C1 dengan nilai atribut adalah 2 dan red akan diberikan metode mengubah nilai radius dari 2 menjadi 2+8

```
C1=Circle (2, 'red' )
```

```
C1.add_radius(8)
```

Metode



```
def add_radius(self,8):  
    self.radius= 2 +8  
    return (10)
```

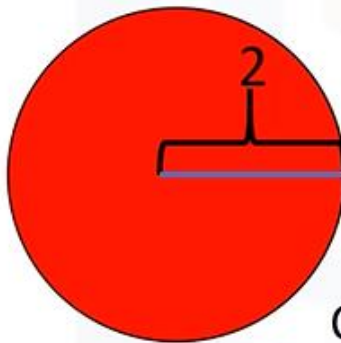
```
self .radius = 2  
self. color ='red'
```

```
self .radius = 10  
self. color ='red'
```

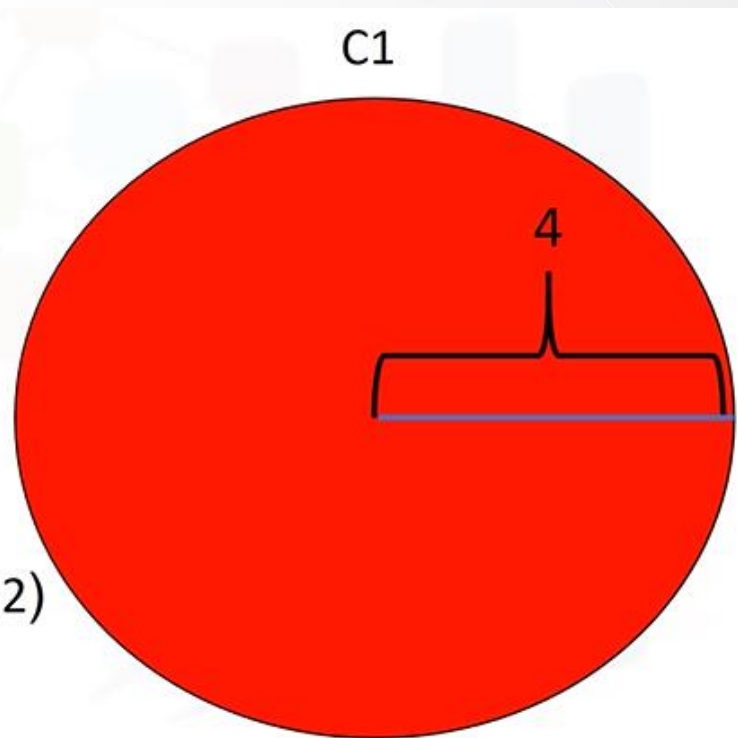
Metode

Contoh object C1 dari class “Circle” dengan nilai awal radius = 2, kemudian diberikan metode membuat radius menjadi 2+2

```
C1=Circle (2,' red' )
```



```
C1.add_radius(2)
```



Metode

Nilai dari parameter dapat langsung diberikan ketika mendefinisikan atribut dari object (tampak pada contoh dibawah ini)

```
class Circle(object):  
    def __init__(self, radius=3, color='red'):  
        self.radius = radius;  
        self.color = color;  
  
    def add_radius(self,r):  
        self.radius= self.radius +r  
  
    def drawCircle(self):
```

} We can add default values for parameters

} New method

Metode

Mendefinisikan object

```
RedCircle=Circle(3,'red')
```

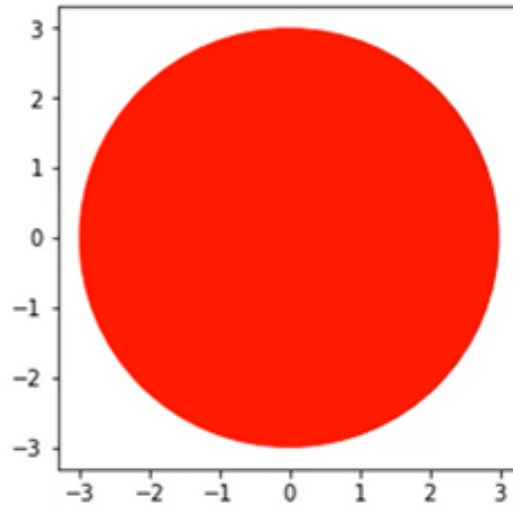
```
RedCircle.radius
```

```
3
```

```
RedCircle.color
```

```
'r'
```

```
RedCircle.drawCircle()
```



Metode

Contoh ke-2

```
BlueCircle=Circle(10,'blue')
```

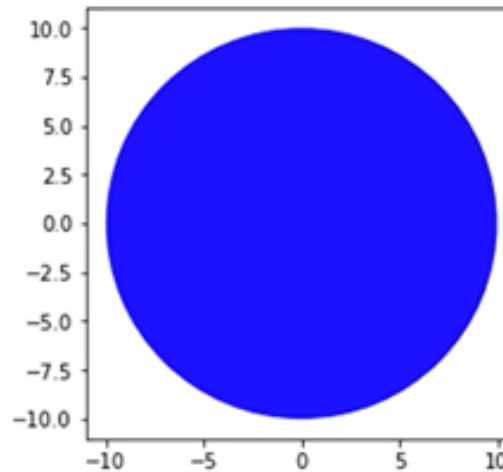
```
BlueCircle.radius
```

```
10
```

```
BlueCircle.color
```

```
'blue'
```

```
BlueCircle.drawCircle()
```

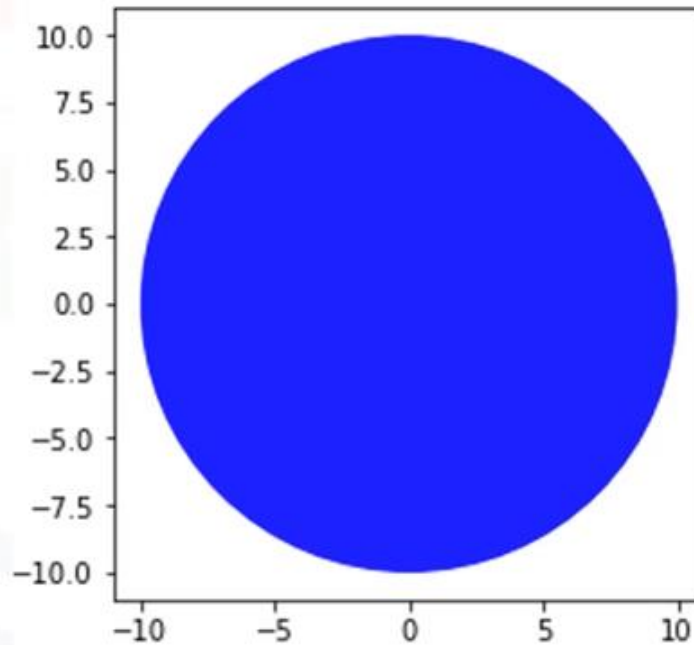
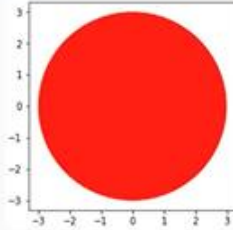


Metode

2 (dua) instances pada class

RedCircle=Circle (3, 'red'):

Attributes
Radius:3
Color: red



BlueCircle=Circle (10, 'blue'):

Attributes
Radius:10
Color: blue

Metode

Contoh ke-1

```
FatYellowRectangle = Rectangle(20,5,'yellow')
```

```
FatYellowRectangle.height
```

5

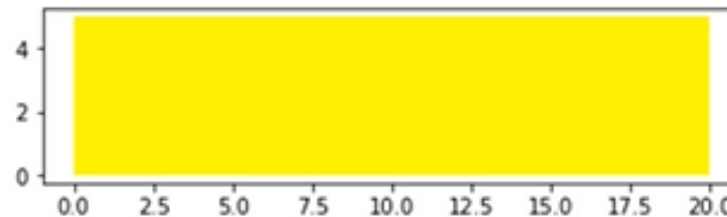
```
FatYellowRectangle.width
```

20

```
FatYellowRectangle.color
```

'yellow'

```
FatYellowRectangle.drawRectangle()
```



Metode

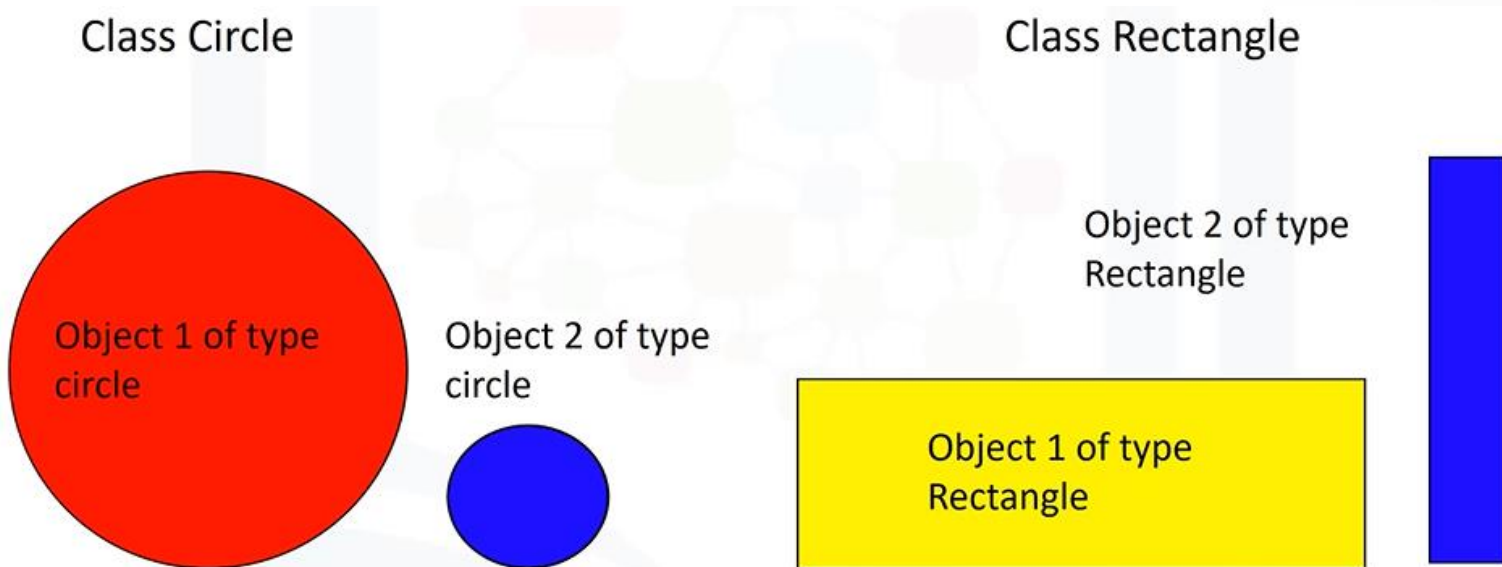
Class dapat dibuat sendiri dengan melakukan pendefinisian dengan perintah:

Def nama_class (object):

Class dapat berisi lebih dari 1 (satu) object.


1 (satu) object bisa memiliki lebih dari 1 (satu) atribut data.

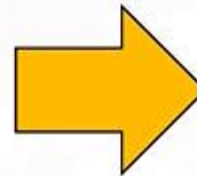
Masing-masing object dapat memiliki nilai atribut yang berbeda-beda.



Metode

Perintah “dir” digunakan untuk mengetahui atribut dan metode

Name of object

dir (Nameofobject):



```
[ '_class_',  
  '_delattr_',  
  '_dict_',  
  '_doc_',  
  '_format_',  
  '_getattrattribute_',  
  '_hash_',  
  '_init_',  
  '_module_',  
  '_new_',  
  '_reduce_',  
  '_reduce_ex_',  
  '_repr_',  
  '_setattr_',  
  '_sizeof_',  
  '_str_',  
  '_subclasshook_',  
  '_weakref_',  
  'add_radius',  
  'color',  
  'drawCircle',  
  'radius']
```