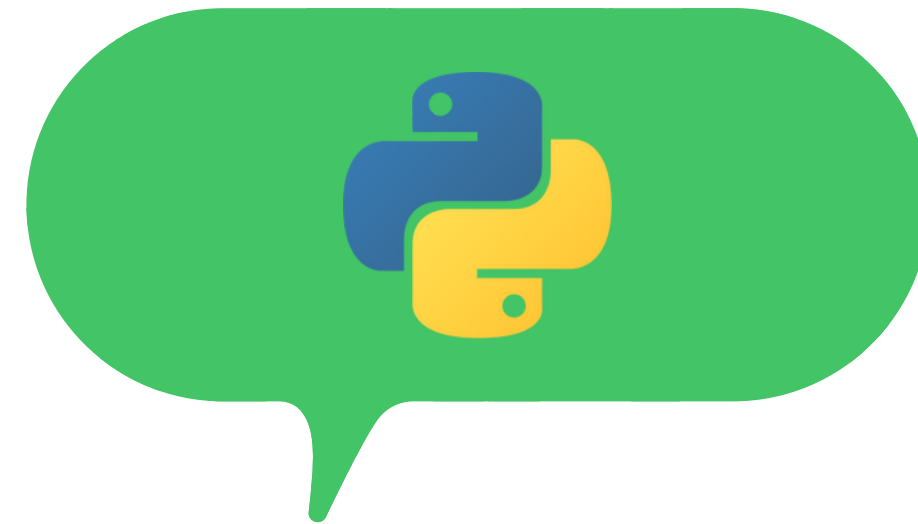


Learning Progress Review



Intro To Control Flow

Seperti lazimnya bahasa pemrograman, python juga mempunyai mekanisme dan sintaks untuk kontrol alir (flow control)



Control Flow pada Python:

1. Conditional Statement

terdiri dari If, If – else, If – elif – else dan Nested If Else

2. Repetitive/Iterative

Statement terdiri dari For loop dan While loop

3. Transfer Statement Terdiri dari Break, Continue, Pass

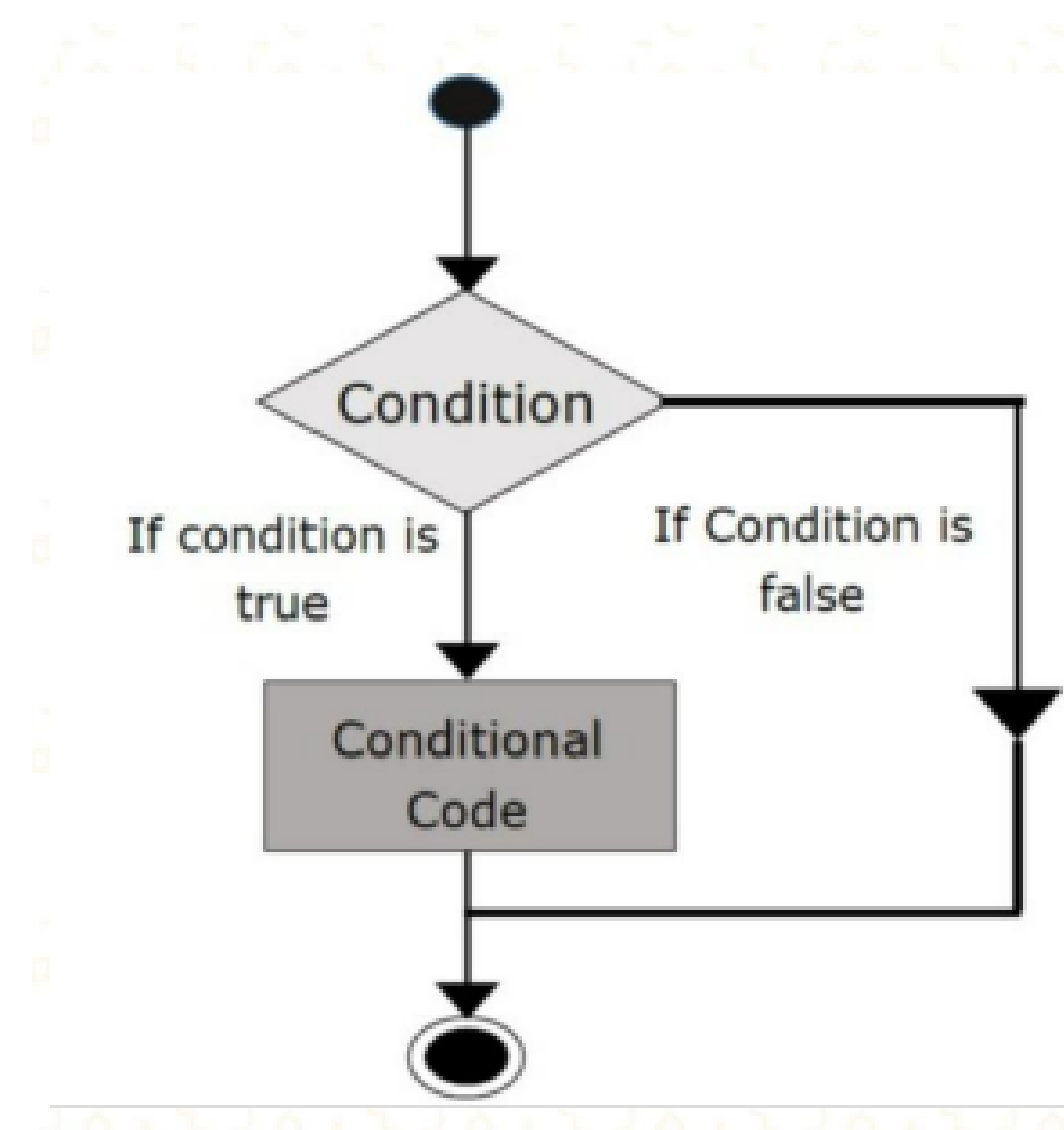
Conditional Statement

Pernyataan Statement If

```
8 x= 5
  y= 2

  if x > y:
    print (f"{x} lebih besar dari {y}")

5 lebih besar dari 2
```



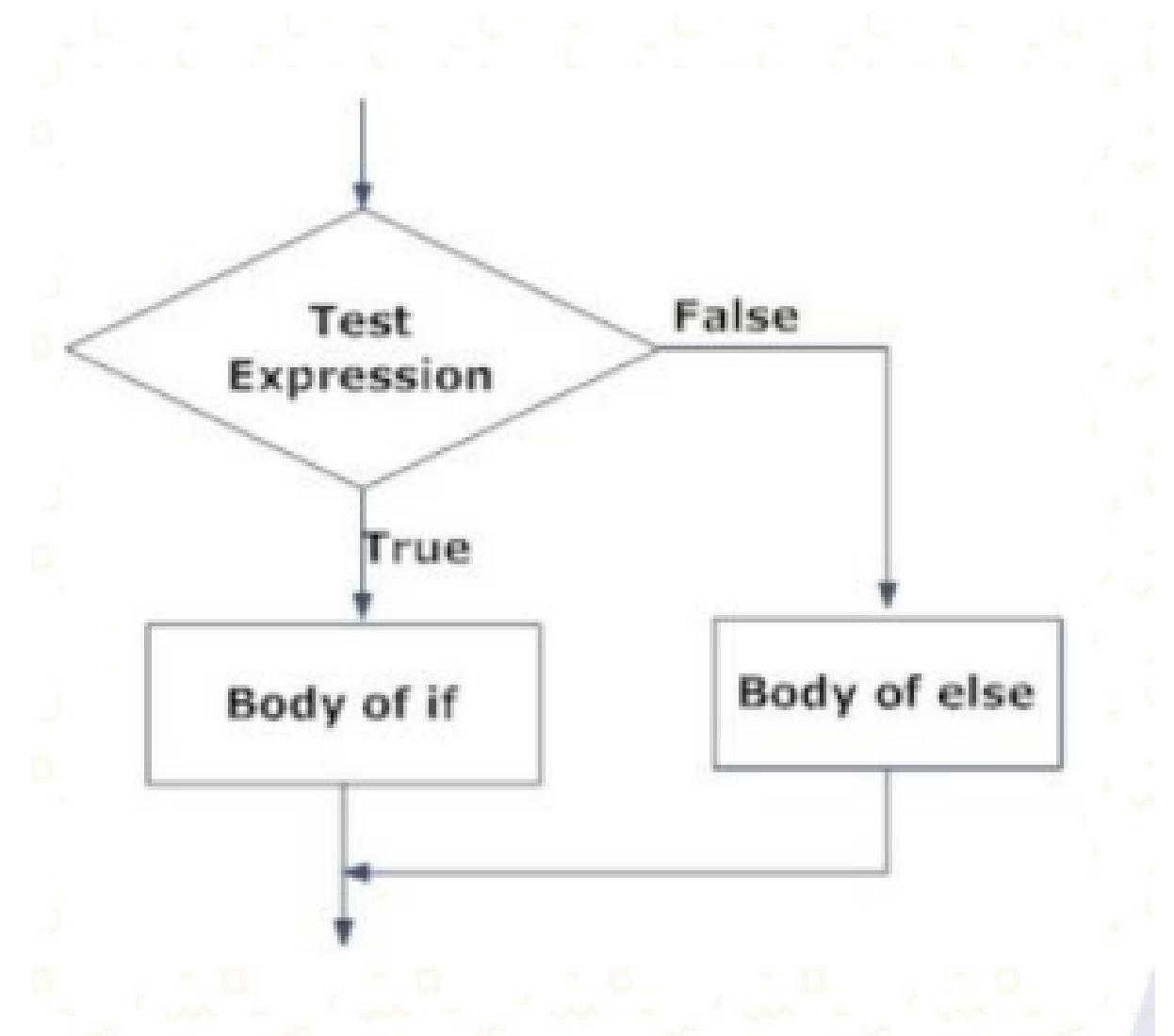
Conditional Statement

Pernyataan Statement If else



```
x= 1  
y= 2  
  
if x > y:  
    print (f"{x} lebih besar dari {y}")  
else:  
    print(f"{x} lebih kecil dari {y}")
```

1 lebih kecil dari 2

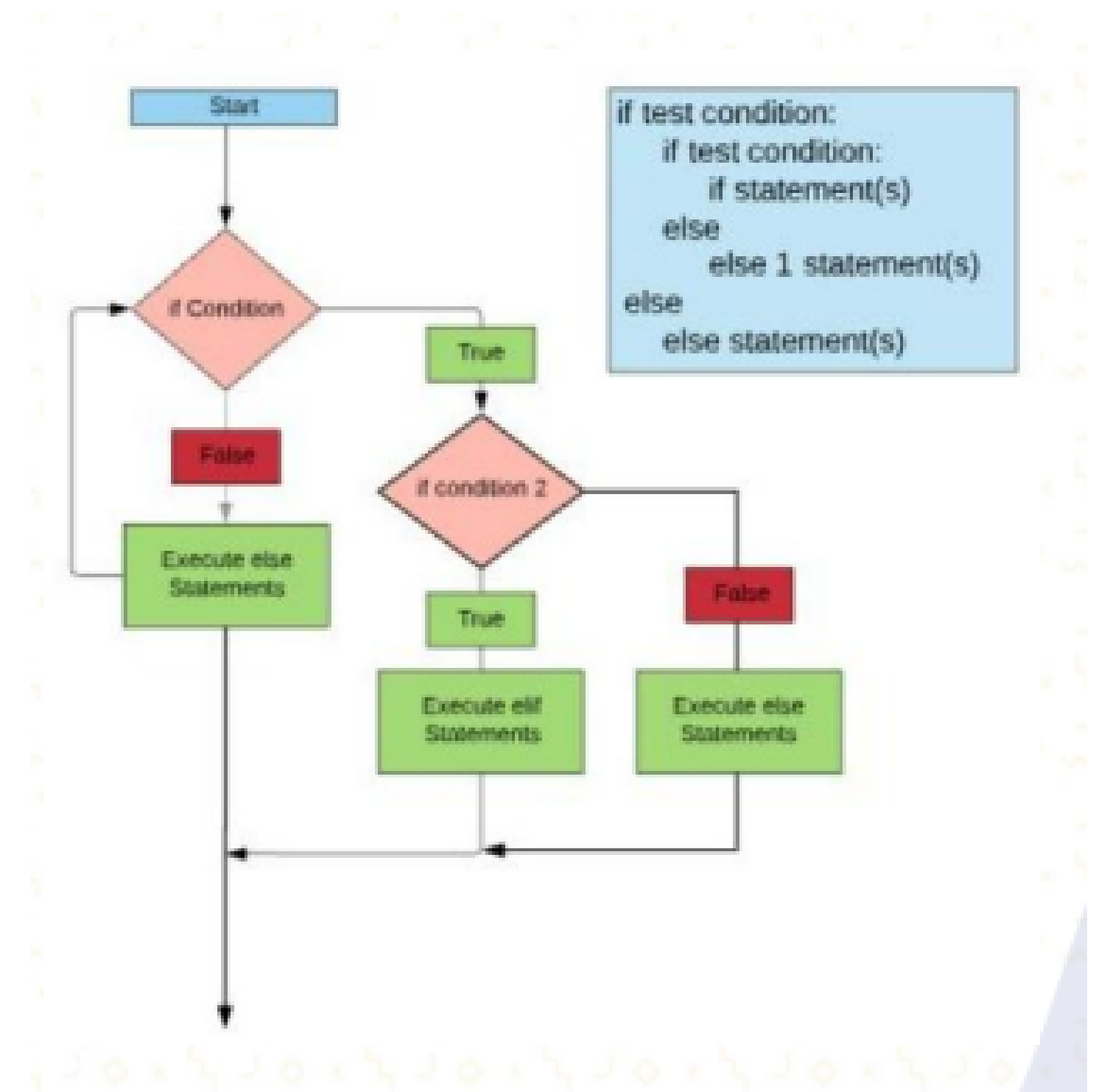


Conditional Statement

Pernyataan Nested If

```
x= 1
y= 2
z = 1

if x > y:
    if x > z:
        print(f"nilai x lebih besar dari z")
    else:
        print(f"nilai c paling besar")
elif y > z:
    print(f"nilai y paling besar")
else:
    print(f"nilai z paling besar")
```



Repetitive/Iterative Statement:

Perulangan atau juga sering dikenal dengan looping merupakan pernyataan atau instruksi yang diberikan kepada komputer agar ia mau melakukan sesuatu entah itu memproses data, menampilkan data, atau yang lainnya secara berulang. Dengan menggunakan perulangan, waktu yang dibutuhkan untuk membuat suatu program akan lebih singkat

For Loops

Eksekusi terhadap blok kode dilakukan berulang kali sesuai dengan variabel yang mengatur perulangan.

```
In [8]: ulang = 7

for i in range(ulang):
    print(f"Loop ke-{i}")
```

```
Loop ke-0
Loop ke-1
Loop ke-2
Loop ke-3
Loop ke-4
Loop ke-5
Loop ke-6
```

While Loops

Perulangan dilakukan selama keadaan masih TRUE, akan dilakukan pengecekan kondisi terlebih dahulu sebelum blok kode dieksekusi

```
In [2]: print("Program Bilangan Bulat 1 hingga x")
        i = 1;
        x = int(input("Masukkan bilangan bulat x = "));

        while i <= x:
            print(i);
            i=i+1;
```

```
Program Bilangan Bulat 1 hingga x
Masukkan bilangan bulat x = 5
1
2
3
4
5
```


Operator

Operator dalam bahasa pemrograman merupakan simbol-simbol yang memberitahu compiler untuk melakukan operasi matematika, relasional atau logis tertentu dan menghasilkan hasil akhir

Operator Aritmatika

No	Operator dan Simbol	Deskripsi
1	Penjumlahan (+)	Menjumlahkan 2 buah Operand.
2	Pengurangan (-)	Mengurangkan 2 buah Operand.
3	Perkalian (*)	Mengalikan 2 buah Operand.
4	Pembagian (/)	Membagi 2 buah Operand.
5	Modulus (%)	Menghasilkan sisa bagi dari pembagian 2 bilangan.
6	Pemangkatan (**)	Memangkatkan nilai Operand.
7	Pembagian Bulat (//)	Sama dengan Pembagian hanya saja, angka dibelakang koma akan dihilangkan/dibulatkan.

Operator

Operator

perbandingan/relasi adalah operator yang bertugas untuk membandingkan antar dua Operand. Jika hasil perbandingan benar, maka akan menghasilkan nilai True, dan sebaliknya jika salah maka akan menghasilkan nilai False

Operator Perbandingan/ Relasi

No	Operator	Simbol
1	Lebih Besar	>
2	Lebih Kecil	<
3	Sama Dengan	==
4	Tidak Sama dengan	!=
5	Lebih Besar Sama dengan	>=
6	Lebih Kecil Sama dengan	<=

Operator

Operator Assignment adalah operator untuk memasukkan suatu nilai ke dalam variabel. Dalam Bahasa Pemrograman Python, Operator Assignment menggunakan tanda sama dengan (=). Misal nilai = 29, artinya nilai telah diberi tugas untuk menyimpan angka 29

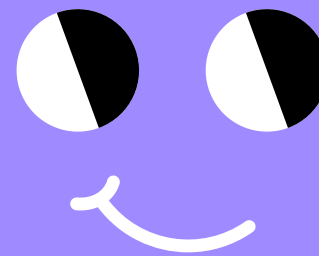
Operator Penugasan (Assignment)

No	Operator	Simbol
1	Pengisian	=
2	Penjumlahan	+=
3	Pengurangan	-=
4	Perkalian	*=
5	Pembagian	/=
6	Sisa Bagi	%=
7	Pemangkatan	**=

dan Operator lain seperti operator logika/boolean, operator keanggotaan, operator identitas dan operator bitwise

Intro To Functions

Function adalah Code yang dapat digunakan berulang-ulang yang digunakan untuk melakukan tindakan/action



def ()

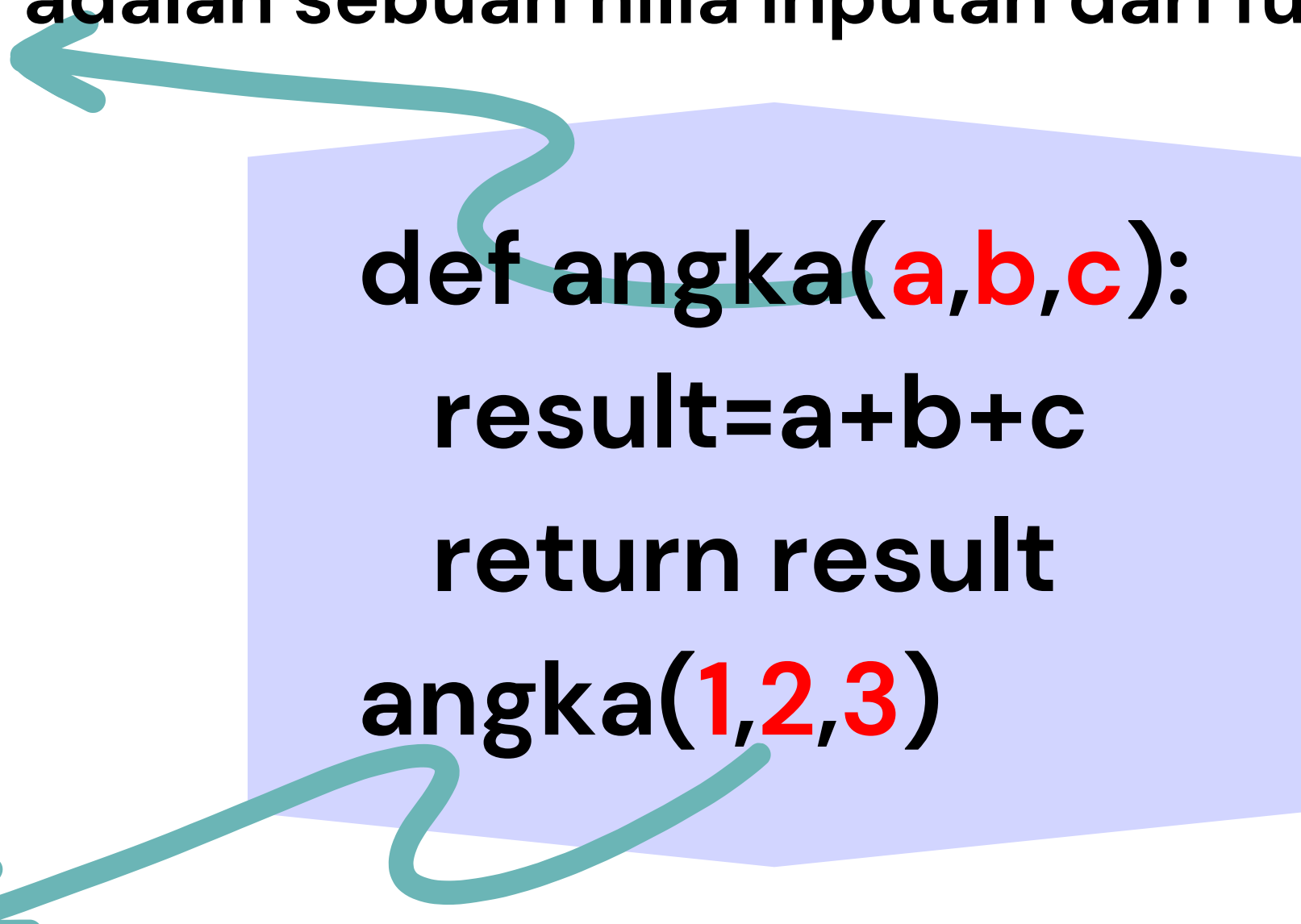


Aturan Sederhana Fungsi Python

1. dimulai dengan def diikuti oleh nama fungsi dan tanda kurung ()
2. Setiap parameter atau argumen harus ditempatkan di dalam tanda kurung ini
3. Blok kode dalam setiap fungsi dimulai dengan titik dua (:) dan indentasi

Arguments & Parameter

Parameter adalah sebuah nilai inputan dari fungsi



The diagram features a light blue rectangular box containing Python code. A teal arrow originates from the word 'Parameter' in the text above and points to the parameter 'a' in the function definition. Another teal arrow originates from the word 'Argument' in the text below and points to the value '1' in the function call.

```
def angka(a,b,c):  
    result=a+b+c  
    return result  
angka(1,2,3)
```

Argument adalah sebutan dari nilai inputan fungsi saat dipanggil

Return Type

Mengembalikan ke hasil

```
def angka(a,b,c):  
    result=a+b+c  
    return result
```

Mengembalikan ke fungsi

```
def angka(a,b,c):  
    tambah=a+b+c  
    kurang=a-b-c  
    return tambah,kurang
```

Mengembalikan ke fungsi menjadi nilai

```
def angka(a,b,c):  
    return {"a":a,"b":b,"c":c}
```

Mengembalikan ke karakter

```
def angka(a,b,c):  
    result=a+b+c  
    return str(result)
```

Mengembalikan ke parameter

```
def angka(a,b,c):  
    return a*2,b*2,c*2
```

Mengembalikan menjadi boolean

```
def angka(a,b,c):  
    return a>b<c
```

Story Function

```
# Storing functions
def one():
    print("storing function 1")
def two():
    print("storing function 2")
def three():
    print("storing function 3")

f1=one
f2=two
f3=three

list_func = [f1, f2, f3]
for func in list_func:
    func()
```

Fungsi yang bisa
menyimpan dan
memanggil dengan
menggunakan nama
lain,

ataupun menyatukan
fungsi menggunakan
list

Object – Oriented Programming (OOP)

pemrograman yang berorientasikan kepada objek, dimana semua data dan fungsi dibungkus dalam class-class atau object-object.

Kelebihan

Parallel development

Reusable

Scalability

Kekurangan

Tidak efisien

Membutuhkan manajemen data
yang ketat

Kemungkinan duplikasi

Konsep pada OOP (Object – Oriented Program ming

1 | Abstrak Class →

- Deskripsi abstrak informasi dan tingkah laku dari sekumpulan data.
- Suatu cetak biru(blueprint) atau prototipe yang digunakan untuk menciptakan objek.
- Suatu struktur yang terdiri atas data kelas (data field), prosedur atau fungsi (method), dan sifat kelas (property).

2 | Encapsulation →

- Kombinasi data dan fungsionalitas dalam sebuah unit tunggal sebagai bentuk untuk menyembunyikan detail informasi.
- Proses enkapsulasi memudahkan untuk menggunakan sebuah objek dari suatu kelas karena tidak perlu mengetahui segala hal secara rinci.

3 | Inheritance →

- Suatu class baru dengan mewarisi sifat dari class lain yang sudah ada.
- Penurunan sifat ini bisa dilakukan secara bertingkat tingkat, sehingga semakin ke bawah class menjadi semakin spesifik.
- Sub class memungkinkan untuk melakukan spesifikasi detail dan perilaku khusus dari class supernya.

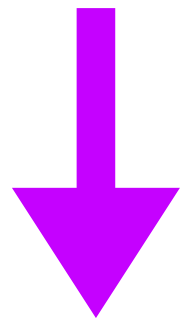
4 | Polymorphism ✓

- Kemampuan objek yang berbeda class namun terkait dalam pewarisan untuk merespon secara berbeda terhadap suatu pesan yang sama.
- Kemampuan sebuah objek memutuskan method yang akan diterapkan padanya, tergantung letak objek tersebut pada jenjang pewarisan.

Class

Class atau kelas dapat didefinisikan sebagai kumpulan objek. Class adalah sebuah "template" untuk membuat sebuah objek yang memiliki karakteristik (**attribute**) dan perilaku (**behavior** atau **method**).

Parent Class



Child Class

Suatu class yang mempunyai class turunan

Class turunan yang dapat mewarisi apa-apa yang dipunyai oleh parent class.

Attribute

Attribute atau *properties* merupakan karakteristik atau variabel dari sebuah class (class attribute dan instance attribute).

```
class Student:  
    #class attribute  
    name = "Jane"  
    course = "Data Engineer"
```

Class attribute adalah variabel yang didefinisikan secara langsung di dalam class yang dapat digunakan oleh semua objek pada class.

```
class Student:  
    school = "Digital Skola"  
    #constructor  
    def __init__(self, name, course):  
        self.name = name #Instance Attribute  
        self.course = course  
Student1 = Student("Jane", "Data Engineer")
```

Instance attribute adalah variabel yang hanya terdapat pada satu objek. Variabel tersebut hanya dapat diakses pada objek tersebut dan didefinisikan pada constructor.



Methods

Instance Method

- Menggunakan parameter self
- Tidak memerlukan decorator
- Dapat diakses oleh object dan class instance

Class Method

- Menggunakan parameter cls
- Memerlukan decorator @classmethod
- Dapat diakses langsung oleh class (tidak memerlukan class instance)

Static Method

- Memerlukan decorator @staticmethod
- Dapat diakses langsung (tidak memerlukan class maupun class instance)



Inheritance

Konsep inheritance, memungkinkan sebuah class untuk mengakses method dan attribute dari parent class

Class Student sebagai parent class

```
class Student():
    def __init__(self, name): #constructor
        self.name = name
    def setEmail(self, email): #instance method/behavioral
        self.email = email
    def setHome(self, home):
        self.home = home
    def setBirthday(self, birthday):
        self.birthday = birthday
    def registerForClass(self, class_obj):
        self.class_obj = class_obj

class Undergraduate(Student):
    def __init__(self, name):
        super().__init__(name) #untuk akses atribut dari parent class
        self.parties = []
    def addParty(self, party):
        self.parties.append(party)
    def getParties(self):
        return self.parties
```

Class Undergraduate sebagai child class

Super().__init__() untuk mengakses atribut dari parent class

Secara singkat, konsep inheritance dapat dilihat pada contoh code disamping

Overriding

Masih merupakan bagian dari inheritance, yang memungkinkan sebuah child class memiliki method yang sama namun behavior yang berbeda dengan parent class-nya

```
class Binatang:
    nama_latin = 'Animalia'

    def makan(self):
        print(f"{self.nama} sedang makan")

class Kucing(Binatang):
    nama_latin = 'Felis Catus'

    def makan(self):
        print(f"{self.nama} sedang makan ikan")
```

Behavior makan pada parent class
Binatang

Override behavior makan pada parent
class Binatang

Overloading

Teknik yang mengatur berbagai perilaku pada fungsi berdasarkan argumen yang diterima. **Satu fungsi atau objek bisa memiliki perilaku yang berbeda tergantung kondisi yang diterima**

fungsi minum menerapkan metode overloading.

Parameter minuman di-set None agar dapat menyesuaikan argumen yang diberikan

```
class Kucing(Binatang):  
    nama_latin = 'Felis Catus'  
  
    def makan(self):  
        print (f"{self.nama} sedang makan ikan")  
  
    def minum (self, minuman=None):  
        if minuman == None:  
            print('TIdak ada minuman')  
  
        else:  
            print (f"{self.nama} sedang minum {minuman}")
```

```
kucing.minum()  
kucing.minum("susu")
```

```
TIdak ada minuman  
tom sedang minum susu
```


Sampai jumpa di
Learning Progress Review
kami berikutnya!

Thank you!

1 Hilda Meiranita Prastika Dewi

2 Nur Indrasari

3 Rezha Sulvian

4 Thasha Dinya Ainsha