Group 5

TIM SERU

Learning Progress Review

18 - 24 Oktober 2025

Outlines

01. **Python Programming II**

List, dictionary, tuple, condition, loop

02. **Python Programming III**

Function, lambda, module, package

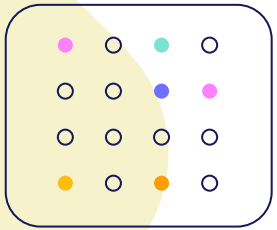
03. **Introduction to Numpy**

Array, mendefinisikan array, manipulasi Array

01.

Python Programming II

List, dictionary, tuple, condition, loop



Python LIST

Definisi

List adalah **tipe data kolektif** yang disimpan secara urut dan bisa diubah nilainya (disebut juga tipe data *sequence*).

Karakteristik

- **Dapat diubah** (**Mutable**): Elemen-elemen di dalam list dapat diubah.
- Terurut (Ordered): Item-item di dalam list memiliki **urutan tertentu**.
- Heterogen (Heterogeneous): List dapat berisi **berbagai tipe data** misalnya string, integer, boolean, atau tipe data lainnya
- Mengizinkan duplikasi (Duplicates):List dapat berisi **elemen-elemen yang sama**, artinya dua atau lebih item boleh memiliki **nilai yang identik.**
- Data dalam list biasa disebut dengan **element**. Setiap elemen disimpan dalam list secara urut dengan penanda urutan yang disebut **index**. Nilai index dimulai dari angka 0

Manfaat :

1. Pengolahan data
2. Memudahkan mengakses data dengan menggunakan indeks
3. Memudahkan proses iterasi
4. Memudahkan proses struktur data bersarang (*nested*)

Bagaimana Membuat List ?

List dapat dibuat dengan dua cara :

1. Dengan **menuliskan elemen-elemen di dalam tanda kurung siku** []
2. Dengan **menggunakan kelas list**().

```
#Contoh sederhana pembuatan list pada bahasa pemrograman python
list1 = ['kimia', 'fisika', 1993, 2017]
list2 = [1, 2, 3, 4, 5 ]
list3 = ["a", "b", "c", "d"]
```

```
#Cara mengakses nilai di dalam list Python

list1 = ['fisika', 'kimia', 1993, 2017]
list2 = [1, 2, 3, 4, 5, 6, 7 ]

print ("list1[0]: ", list1[0])
print ("list2[1:5]: ", list2[2:6])
```

```
list1[0]:  fisika
list2[1:5]:  [3, 4, 5, 6]
```

Python Dictionary

Definisi

Dictionary dalam Python adalah koleksi pasangan key-value yang bersifat tidak berurutan.

Karakteristik

1. Tidak Berurutan (**Unordered**) – untuk Python **versi 3.6** dan sebelumnya: item-item di dalam dictionary disimpan tanpa nilai indeks seperti pada list (biasanya berupa urutan angka).
2. Berurutan (**Ordered**) – untuk Python **versi 3.7** dan setelahnya: dictionary dapat dianggap seperti tabel hash (Hash Table) yang berisi pasangan key-value yang tersusun dalam urutan semu (pseudo-random) berdasarkan perhitungan dari fungsi hash (Hash Function).
3. **Unik**: Seperti disebutkan sebelumnya, setiap nilai memiliki kunci, dan setiap kunci dalam dictionary harus unik.
4. Dapat Diubah (**Mutable**): Dictionary merupakan koleksi yang dapat diubah, artinya kita bisa

Cara Membuat Dictionary

Terdapat beberapa cara membuat Dictionary:

- **Penggunaan Kurung Kurawal** {}

```
# Contoh 1: Membuat Dictionary dengan beberapa pasangan key-value
person = {'nama': 'John', 'umur': 30, 'pekerjaan': 'analyst'}
print("Contoh 1:", person)
```

output:

Contoh 1: {'nama': 'John', 'umur': 30, 'pekerjaan': 'analyst'}

- **Mengguanakan fungsi dict()**

```
# Contoh 2: Membuat Dictionary dengan fungsi dict()
colors = dict(biru='blue', merah='red', kuning='yellow')
print("Contoh 2:", colors)
```

Output:

Contoh 2: {'biru': 'blue', 'merah': 'red', 'kuning': 'yellow'}

- **Menggunakan List of Tuples**

```
# Contoh 3: Membuat Dictionary dengan list of tuples
fruits = dict([('apel', 'apple'), ('jeruk', 'orange'), ('anggur', 'grape')])
print("Contoh 3:", fruits)
```

Output :

Contoh 3: {'apel': 'apple', 'jeruk': 'orange', 'anggur': 'grape'}

- **Menggunakan Metode fromkeys()**

```
# Contoh 4: Membuat Dictionary dengan metode fromkeys()
keys = ['nama', 'umur', 'alamat']
default_value = 'belum diisi'
info = dict.fromkeys(keys, default_value)
print("Contoh 4:", info)
```

Output :

Python Tuple

Definisi

Struktur data yang digunakan untuk menyimpan sekumpulan data.

Karakteristik

1. Ordered : Elemen disimpan dalam urutan tetap, bisa diakses dengan indeks (tuple[0], tuple[1], dst)

   Contoh:

```
buah = ("apel", "pisang", "jeruk")
print(buah[0])  # apel
print(buah[1])  # pisang
print(buah[2])  # jeruk
```

2. Immutable : Setelah dibuat, isinya tidak bisa ditambah, dihapus, atau diganti.

   Contoh:

```
buah = ("apel", "pisang", "jeruk")
buah[0] = "mangga"  # Error: tuple tidak bisa diubah
```

   Jika dijalankan, Python akan menampilkan pesan:

```
TypeError: 'tuple' object does not support item assignment
```

3. Indexed : Setiap elemen punya posisi (indeks) mulai dari 0.

   Contoh:

```
angka = (10, 20, 30, 40)
print(angka[0])  # 10
print(angka[3])  # 40
```

4. Allow Duplicates : Boleh ada nilai yang sama.

   Contoh:

```
buah = ("apel", "apel", "pisang", "jeruk")
print(buah)
```

5. Can contain different data types : Bisa berisi campuran.

   Contoh:

```
data = ("apel", 10, True, 3.14)
print(data)
```

6. Defined by Parentheses ():

   Contoh:

```
warna = ("merah", "biru", "kuning")
print(type(warna))
```

Python Condition

Definisi

syarat atau pernyataan logika yang digunakan untuk menentukan apakah suatu blok kode akan dijalankan atau tidak.

Konsep

Kondisi ini hanya memiliki dua kemungkinan hasil:

- True → kode di dalam blok dijalankan (*if*)
- False → kode di dalam blok dilewati atau ganti ke kondisi lain (*else*)

Macam-macam Condition

1. If: Menjalankan kode hanya jika kondisi bernilai True.

Contoh:

```
umur = 18
if umur >= 17:
    print("Boleh membuat KTP")
```

Output:

```
Boleh membuat KTP
```

2. If-Else: Jika kondisi True, jalankan blok if.
Jika False, jalankan blok else.

Contoh:

```
nilai = 60

if nilai >= 70:
    print("Lulus")
else:
    print("Tidak lulus")
```

Output:

```
Tidak lulus
```

3. If-Elif-Else: Digunakan jika ada lebih dari dua kemungkinan kondisi.

Contoh:

```
nilai = 80

if nilai >= 90:
    print("Nilai A")
elif nilai >= 80:
    print("Nilai B")
elif nilai >= 70:
    print("Nilai C")
else:
    print("Nilai D")
```

Output:

```
Nilai B
```

4. Nested If:

- Kondisi `if` berada di dalam `if` lain.
- Biasanya digunakan jika keputusan tergantung pada dua syarat yang bertingkat.

Contoh:

```
umur = 18
izin = True

if umur >= 17:
    if izin:
        print("Boleh mengemudi")
    else:
        print("Perlu izin dulu")
```

Output:

```
Boleh mengemudi
```

5. Logical Conditions: Digunakan untuk menggabungkan beberapa kondisi dalam satu pernyataan.

a. and: Semua kondisi harus True

Contoh:

```
umur = 18
izin = True

if umur >= 17 and izin:
    print("Boleh mengemudi")
```

b. or: Salah satu kondisi True sudah cukup

Contoh:

```
nilai = 75
kehadiran = 60

if nilai >= 70 or kehadiran >= 70:
    print("Naik kelas")
```

c. not: Membalik nilai kondisi

Contoh:

```
izin = False

if not izin:
    print("Belum mendapat izin")
```

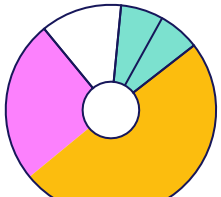
Python Loop (1/7)

Definisi

Loop merupakan teknik untuk mengulang-ulang eksekusi suatu blok kode, atau mengiterasi elemen milik tipe data kolektif (contohnya: list).

Di dalam bahasa pemrograman **Python** pengulangan dibagi menjadi **3 bagian,** yaitu :

- For Loop
- While Loop
- Nested Loop



For Loop

Loop for digunakan untuk **mengiterasi** melalui **setiap** elemen **dalam** suatu urutan, seperti **list, tuple, string, atau objek lainnya**

Syntax dasar For Loop

```
for i in range/sequencee:
    statement 1
    statement 2
    statement n
```

Python Loop (2/7)

Python for loop

A for loop is **used for iterating over a sequence and iterables** (like range, list, a tuple, a dictionary, a set, or a string).

```
for i in range(5):
    statement 1
    statement 2
    ...
    statement n
else:
    statement(s)
```

Definite iterations.
(Total 5 iterations)

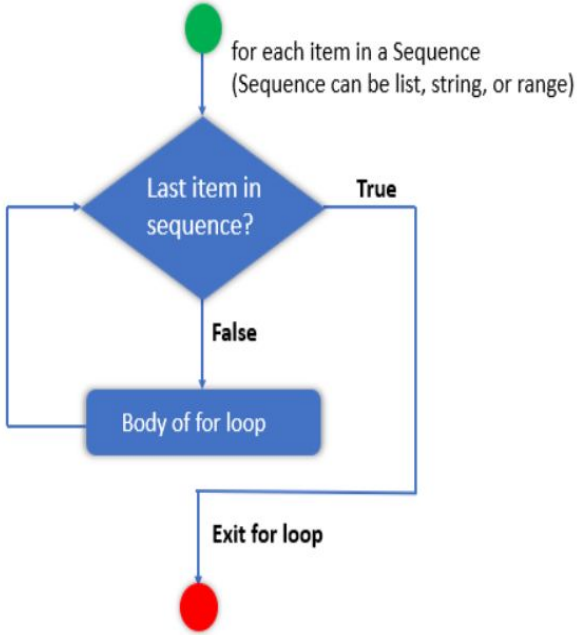
Indentation
Loop body is must be properly indented

Body of for loop
Execute till the last item of a sequence

Else Block (optional)
Execute only when for loop executes normally

PYnative.com

Flow Chart For Loop



Source : https://pynative.com/python-for-loop/#h-how-for-loop-works

Python Loop (3/7)

Contoh Code For Loop

Contoh menggunakan range()

```
for i in range(1, 6):
    print(i)
```

```
1
2
3
4
5
```

Contoh menggunakan String

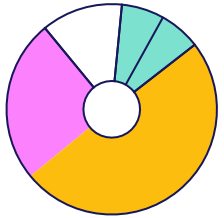
```
for x in "banana":
  print(x)
```

```
b
a
n
a
n
a
```

Contoh menggunakan List

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

```
apple
banana
cherry
```



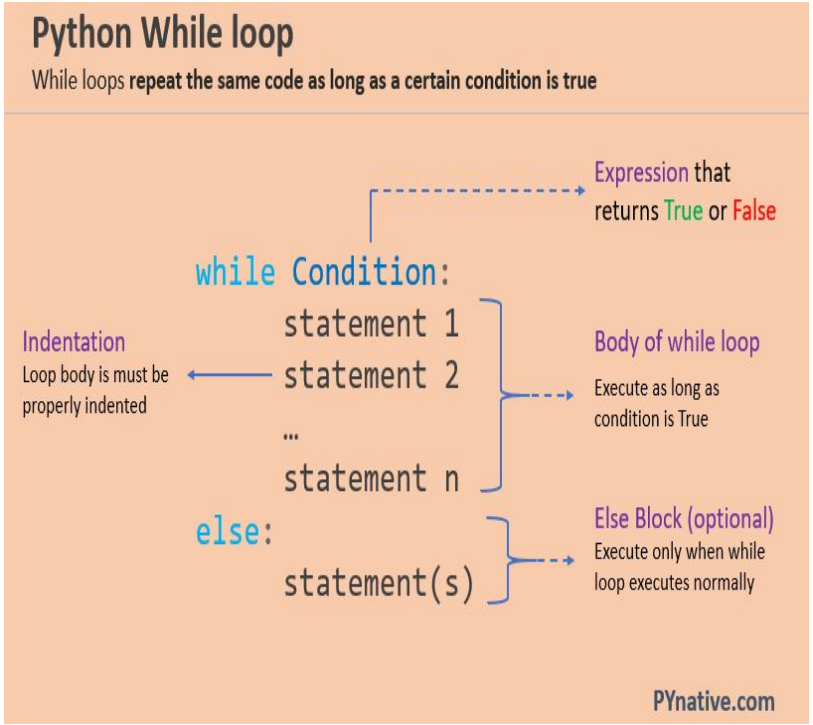
Python Loop (4/7)

While Loop

Perulangan **while** adalah salah satu bentuk perulangan dalam Python yang digunakan untuk menjalankan suatu blok kode **selama kondisi yang diberikan terus terpenuhi**

Syntax dasar While Loop

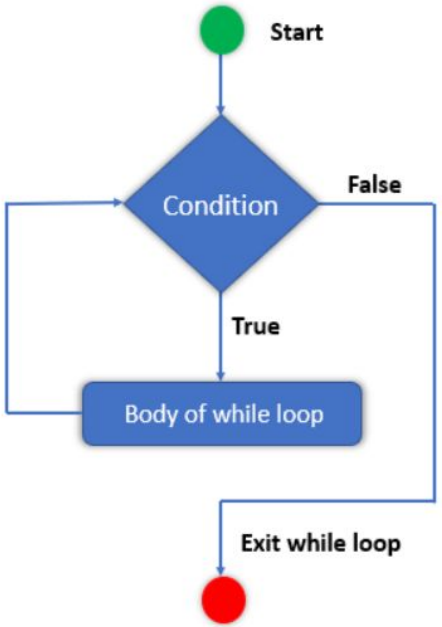
```
while loop_expression:
    command_expression
```



Source : https://pynative.com/python-for-loop/#h-how-for-loop-works

Python Loop (5/7)

Flow Chart While Loop



Contoh dalam Program

Enter any number between 100 and 500

```
number = int(input('Enter any number between 100 and 500 '))
# number greater than 100 and less than 500
while number < 100 or number > 500:
    print('Incorrect number, Please enter correct number:')
    number = int(input('Enter a Number between 100 and 500 '))
else:
    print("Given Number is correct", number)
```

```
Enter any number between 100 and 500 10
Incorrect number, Please enter correct number:
Enter a Number between 100 and 500 200
Given Number is correct 200
```

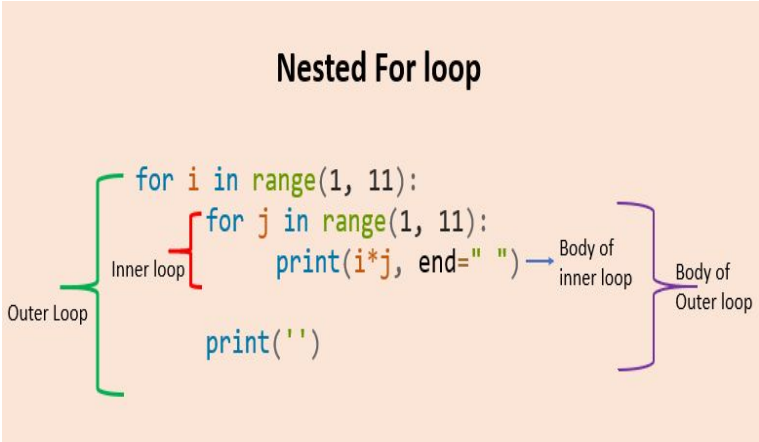
Python Loop (6/7)

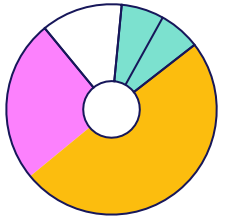
Nested Loop

Nested loop atau loop bersarang, merupakan jenis loop pada Python yang mengizinkan p**enggunaan loop didalam loop**.

Syntax Nested Loop

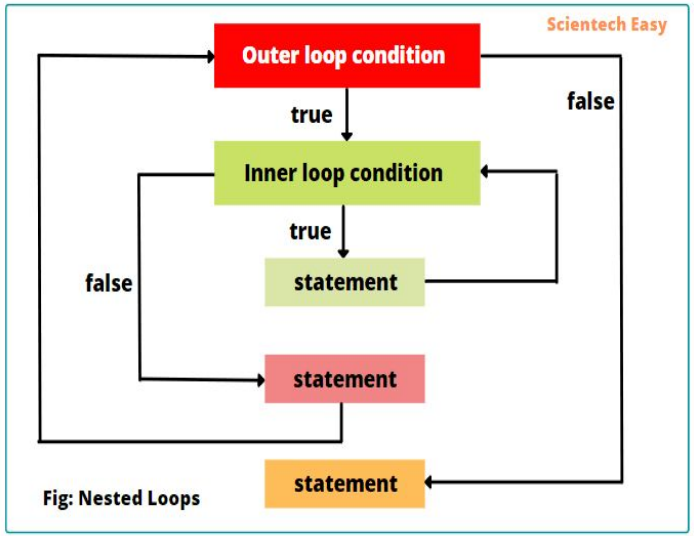
```
# outer for loop
for element in sequence
   # inner for loop
   for element in sequence:
       body of inner for loop
   body of outer for loop
```





Python Loop (7/7)

Flow Chart Nested Loop



Source: https://www.scientecheasy.com/2022/11/nested-loops-in-python.html/

Contoh Program Menggunakan Nested Loop

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]

for x in adj:
  for y in fruits:
    print(x, y)
```

```
red apple
red banana
red cherry
big apple
big banana
big cherry
tasty apple
tasty banana
tasty cherry
```

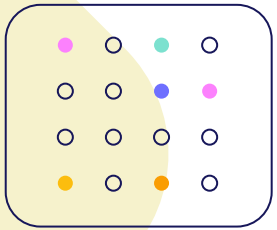
```
i = 1
while i <= 5:
    j = 1
    while j <= 10:
        print(j, end='')
        j = j + 1
    i = i + 1
    print()
```

```
12345678910
12345678910
12345678910
12345678910
12345678910
```

02.

Python Programming III

Function, lambda, module, package



Python Function

Tentang

Kumpulan perintah/kode untuk melakukan tugas tertentu, dapat digunakan berulang kali (reusable).

Fitur

- Dapat menerima parameter, mengembalikan nilai, dan dipanggil secara independen.
- Untuk membuat fungsi buatan sendiri (user-defined function), gunakan kata kunci *def*

Manfaat

- Membantu memecah program besar menjadi bagian-bagian kode kecil dan spesifik.
- Simplifikasi coding memudahkan pemahaman program .
- Menjadikan kode pemrograman lebih terstruktur dan sistematis.

Function dengan parameter

```
def selamat_datang(nama):
    print(f"Selamat datang dan sukses {nama}")
```

Function dengan kembalian nilai

```
def luas_persegi_panjang(panjang, lebar):
    luas = panjang * lebar
    return luas
```

Function scope

- Local scope: variabel di dalam fungsi, berlaku lokal
- Global scope: variabel diluar fungsi, berlaku global
- Global keyword: membuat variabel menjadi global

Lambda Expression

Tentang

- Lambda adalah ekspresi untuk membuat fungsi sederhana yang mengembalikan nilai.
- Disebut sebagai anonymous function.

Fitur

- Dapat memiliki lebih dari satu parameter argumen.
- Hanya dapat memiliki satu baris perintah.
- Untuk kemudahan pemanggilan, simpan ke dalam sebuah variabel.

Manfaat

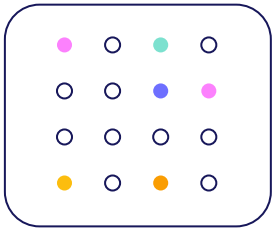
- Dipakai dalam pemrograman level menengah atas, misal untuk membuat sebuah high-order function.

Contoh

status = **lambda** umur: ‘remaja’ if umur > 10 and umur < 20 else ‘bukan remaja’

print(status(15))

Output: remaja



Python Module

Tentang

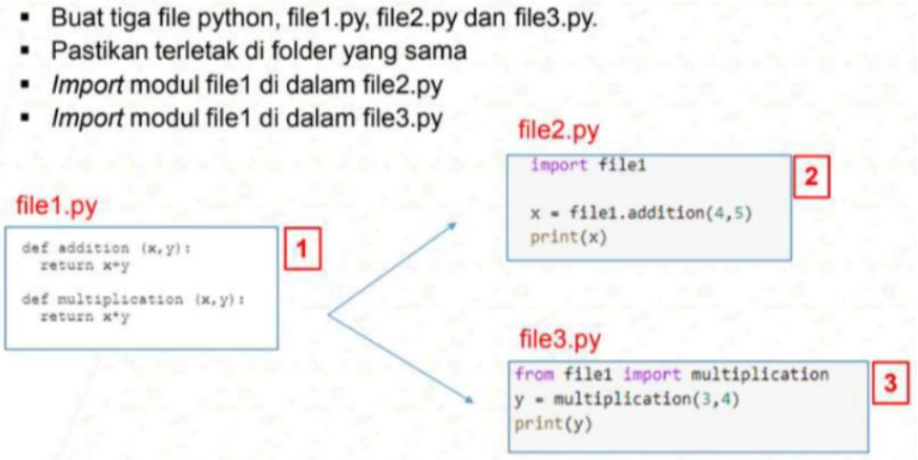
Sebuah file berisi sekumpulan kode, seperti fungsi, class, dan variabel yang disimpan dengan ekstensi .py, dan dapat dieksekusi oleh interpreter Python.

Manfaat

- Simplicity: mengelompokkan kode-kode yang berkaitan dalam satu file.
- Maintainability: membantu mengorganisir kode-kode yang akan digunakan.
- Reusability: untuk menggunakan, cukup dengan mengimpor modul tanpa mengetik berulang kali.

Penerapan Modul

- Buat tiga file python, file1.py, file2.py dan file3.py.
- Pastikan terletak di folder yang sama
- *Import* modul file1 di dalam file2.py
- *Import* modul file1 di dalam file3.py



Python Package

Tentang

Suatu fitur untuk mengelola dan mengorganisir module-module Python ke dalam suatu bentuk direktori.

Cara kerja

- Sama seperti ketika mengorganisir file-file pada komputer ke dalam suatu direktori.
- Perbedaan hanya harus ada sebuah file bernama __init__.py (module constructor).
- File __init__.py akan memberitahu Python bahwa suatu direktori merupakan sebuah package yang berisi modul-modul.
- Modul-modul dalam package kemudian dapat diimpor menggunakan notasi dot.

Gambaran package

```
├── luas
│   ├── __init__.py
│   ├── lingkaran.py
│   ├── persegi.py
│   └── segitiga.py
├── volume
│   ├── __init__.py
│   ├── bola.py
│   ├── kubik.py
│   └── trapesium.py
└── skrip-utama.py
```

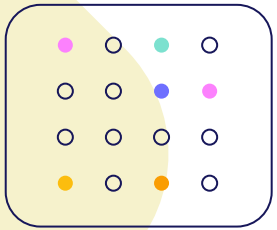
Cara mengimpor package

```
from luas.segitiga import luas_segitiga
from luas import lingkaran, persegi
from volume.kubik import volume_kubik
import volume.bola
from volume.trapesium import *
```

03.

Introduction to Numpy

Array, mendefinisikan array, manipulasi Array



Pengenalan Numpy

Pengertian

Merupakan library open-source Python untuk melakukan komputasi numerik dan ilmiah. NumPy mendukung berbagai operasi matematika seperti aljabar linear, geometri, dan statistik.

Kelebihan Numpy

- Menggunakan struktur data **array multidimensi (NumPy array)**
- Operasi numerik menjadi **lebih cepat dan efisien** dibanding list biasa
- Penting dalam **Data Science**, khususnya untuk **Data Preparation**

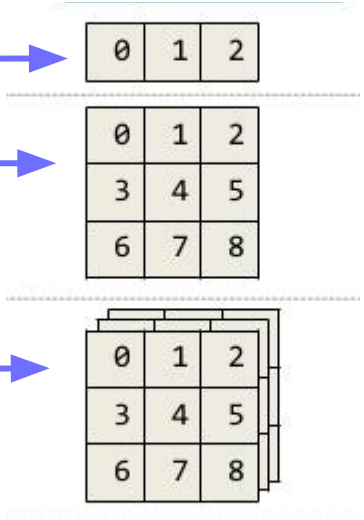
Cara Mengimpor Numpy

- import numpy
- # atau
- import numpy as np

Array Di Numpy

Array adalah sekumpulan data yang hanya dapat berisi satu tipe data saja. Array mampu menyimpan data berisi integer, float, boolean dan complex.
Ada beberapa jenis array yaitu:

1. Vector
   Contoh: ([0,1,2])
2. Matrix
   Contoh: ([[0,1,2]
   [3,4,5]
   [6,7,8]])
3. 3D Matrix (Tensor)
   Contoh: ([[[0,1]
   [3,4]]
   [[5,6]
   [7,8]]
   [[9,10]
   [11,12]]])



Mendefinisikan Array

1. **numpy.array()**

Fungsi untuk membuat list menjadi array.
Input: numpy.array([[1,2,3,4]])
Output: ([[1,2],
[3,4]])

2. **numpy.arange()**

Fungsi untuk menampilkan array dalam interval tertentu dan value berjarak sama.
Parameter : start, stop, step, dtype.
Contoh
Input: numpy.arange(3)
Output: [0,1,2]
Input: numpy.arange(3,7)
Output: [3,4,5,6]
Input: numpy.arange(2,8,2)
Output: [2,4,6]

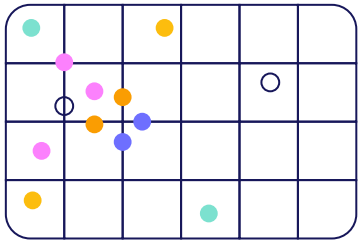
3. **numpy.zeros()**

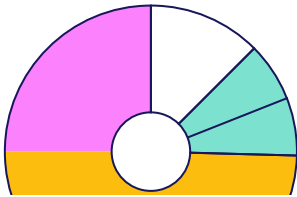
Fungsi untuk menampilkan array baru dengan nilai berisi 0 (default float).
Parameter: shape dan dtype.
Input: numpy.zeros(5)
Output: [0.,0.,0.,0.,0.]

4. **numpy.ones()**

Fungsi untuk menampilkan array baru dengan nilai berisi 1 (default float).
Parameter: shape dan dtype.
Input : numpy.ones(5)
Output: [1.,1.,1.,1.,1.]

Manipulasi Array: Reshape





numpy.reshape()

Berfungsi untuk memberikan bentuk baru kepada array tanpa mengubah data (Dari 1-D ke 2-D atau 1-D ke 3-D)

Parameter Fungsi reshape()

- **Arr** → array yg mau direshape
- **Newshape** → bentuk baru harus kompatibel dengan bentuk aslinya
- **Order** → F style array atau C style array

Contoh:

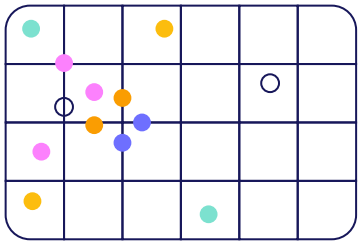
```
[25]
✓ 0s
input = np.array([[1,2,3,4,5,6,7,8]])
print("sebelum diubah masih matrix 1-D")
print(input,"\n")
print("sesudah diubah dengan matrix 2x4")
input2 = input.reshape(2,4)
print(input2)
```

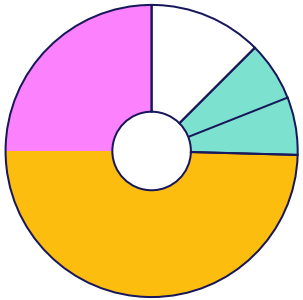
```
sebelum diubah masih matrix 1-D
[[1 2 3 4 5 6 7 8]]

sesudah diubah dengan matrix 2x4
[[1 2 3 4]
 [5 6 7 8]]
```

Manipulasi Array: Flatten

ndarray.flatten()

Berfungsi untuk menampilkan array yang sudah diubah bentuknya menjadi Single-Dimensional Array



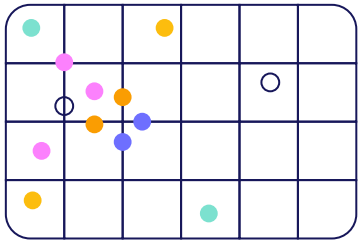


Contoh:

```
print("sebelum diubah masih matrix 2-D")
print(input2,"\n")
input3 = np.ndarray.flatten(input2)
print("sesudah diubah dengan matrix 1-D")
print(input3)
```

```
sebelum diubah masih matrix 2-D
[[1 2 3 4]
 [5 6 7 8]]

sesudah diubah dengan matrix 1-D
[1 2 3 4 5 6 7 8]
```

Manipulasi Array: Transpose



numpy.transpose()

Berfungsi untuk memberikan bentuk baru ke array tanpa mengubah data

Parameter Fungsi transpose()

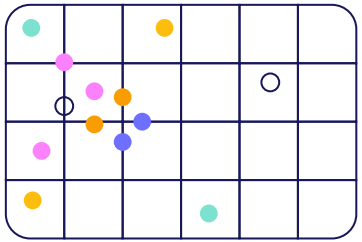
- **Arr** → array yg mau di-transpose
- **Axis** → Default reversed (optional)

Contoh:

```
print("sebelum ditranspose masih matrix 2x4")
print(input2,"\n")
input3 = np.transpose(input2)
print("sesudah ditranspose masih matrix 4x2")
print(input3,"\n")
```

```
sebelum ditranspose masih matrix 2x4
[[1 2 3 4]
 [5 6 7 8]]

sesudah ditranspose masih matrix 4x2
[[1 5]
 [2 6]
 [3 7]
 [4 8]]
```

Manipulasi Array: Concatenate



numpy.concatenate()

Berfungsi untuk menggabungkan dua atau lebih array dengan bentuk yang sama di sepanjang axis yang sudah ada

Parameter Fungsi Concatenate()

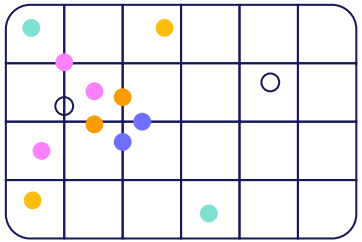
- **Arr1, Arr2** → array yg mau digabungkan
- **Axis** → Nilainya adalah 0 atau 1, axis dimana array harus digabungkan. Default adalah 0 (Baris)

Contoh:

```
arr1 = np.array([[1,2,3],[4,5,6],[7,8,9]])
print("Isi dari array 1")
print(arr1,"\n")

arr2 = np.array([[20,21,22],[23,24,25],[26,27,28]])
print("Isi dari array 2")
print(arr2,"\n")

arr3 = np.concatenate((arr1, arr2), axis=0)
print("Isi dari hasil gabungan array 1 dan 2")
print(arr3,"\n")
```

```
Isi dari array 1
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Isi dari array 2
[[20 21 22]
 [23 24 25]
 [26 27 28]]

Isi dari hasil gabungan array 1 dan 2
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [20 21 22]
 [23 24 25]
 [26 27 28]]
```

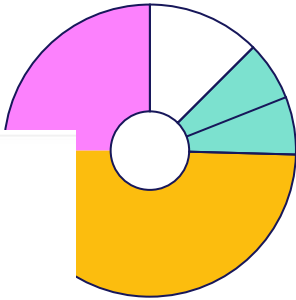
Manipulasi Array: Concatenate



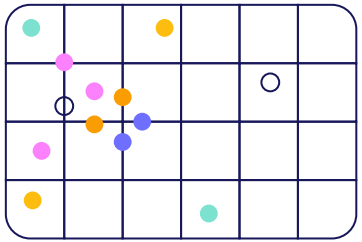
Contoh:

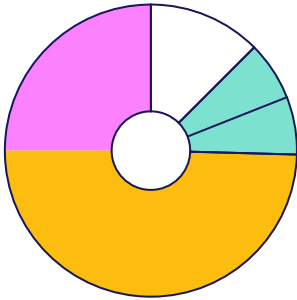
```
arr1 = np.array([[1,2,3],[4,5,6],[7,8,9]])
print("Isi dari array 1")
print(arr1,"\n")

arr2 = np.array([[20,21,22],[23,24,25],[26,27,28]])
print("Isi dari array 2")
print(arr2,"\n")

arr3 = np.concatenate((arr1, arr2), axis=0)
print("Isi dari hasil gabungan array 1 dan 2 dengan axis 0")
print(arr3,"\n")

arr4 = np.concatenate((arr1, arr2), axis=1)
print("Isi dari hasil gabungan array 1 dan 2 dengan axis 1")
print(arr4,"\n")
```

```
Isi dari array 1
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Isi dari array 2
[[20 21 22]
 [23 24 25]
 [26 27 28]]

Isi dari hasil gabungan array 1 dan 2 dengan axis 0
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [20 21 22]
 [23 24 25]
 [26 27 28]]

Isi dari hasil gabungan array 1 dan 2 dengan axis 1
[[ 1  2  3 20 21 22]
 [ 4  5  6 23 24 25]
 [ 7  8  9 26 27 28]]
```



Manipulasi Array: Stack





numpy.stack()

Berfungsi untuk menggabungkan sequence array di sepanjang axis baru

Parameter Fungsi Stack()

- **Arr1, Arr2** → array yg mau digabungkan
- **Axis** → Nilainya adalah -1, 0 atau 1, axis dimana array harus digabungkan. Default adalah 0

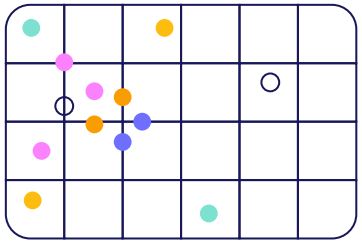
Manipulasi Array: Stack

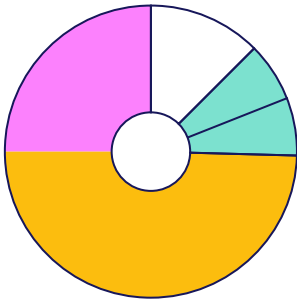
Contoh:

```
arr1 = np.array([[1,2,3],[4,5,6],[7,8,9]])
print("Isi dari array 1")
print(arr1,"\n")

arr2 = np.array([[20,21,22],[23,24,25],[26,27,28]])
print("Isi dari array 2")
print(arr2,"\n")

arr3 = np.stack((arr1, arr2), axis=1)
print("Isi dari hasil gabungan array 1 dan 2 dengan axis 1")
print(arr3,"\n")

arr4 = np.stack((arr1, arr2), axis=-1)
print("Isi dari hasil gabungan array 1 dan 2 dengan axis -1")
print(arr4,"\n")
```

```
Isi dari array 1
[[1 2 3]
 [4 5 6]
 [7 8 9]]

Isi dari array 2
[[20 21 22]
 [23 24 25]
 [26 27 28]]

Isi dari hasil gabungan array 1 dan 2 dengan axis 1
[[[ 1  2  3]
  [20 21 22]]

 [[ 4  5  6]
  [23 24 25]]

 [[ 7  8  9]
  [26 27 28]]]

Isi dari hasil gabungan array 1 dan 2 dengan axis -1
[[[ 1 20]
  [ 2 21]
  [ 3 22]]

 [[ 4 23]
  [ 5 24]
  [ 6 25]]

 [[ 7 26]
  [ 8 27]
  [ 9 28]]]
```

Manipulasi Array: Split





numpy.split()

Berfungsi untuk membagi 1-D array menjadi subarray

Parameter Fungsi Split()

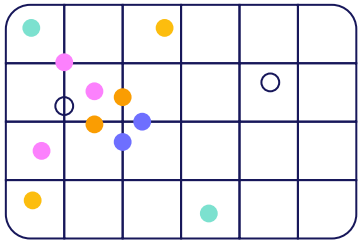
- **Arr** → array yg mau dibagi
- **Indices_or_sections** → menunjukkan jumlah subarray berukuran sama yang akan dibuat dari array input
- **Axis** → Axis dimana array harus dibagi. Default adalah 0

Contoh:

```
input = np.arange(9)
print("Isi dari array sebelum displit")
print(input)
print("\nIsi dari array setelah displit")
output = np.split(input,3)
print(output)
```

```
Isi dari array sebelum displit
[0 1 2 3 4 5 6 7 8]

Isi dari array setelah displit
[array([0, 1, 2]), array([3, 4, 5]), array([6, 7, 8])]
```

Manipulasi Array: Resize



numpy.resize()

Berfungsi untuk mengubah array dengan ukuran yang ditentukan. Jika ukuran baru lebih besar dari aslinya, maka terdapat salinan value-value sebelumnya

Parameter Fungsi Resize()

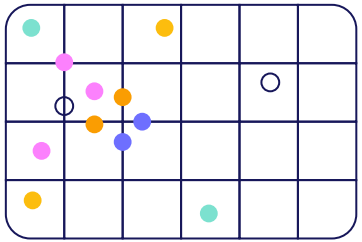
- **Arr** → array yg mau diresize
- **Shape** → Bentuk baru dari arraynya

Contoh:

```
input = np.array([[1,2,3],[4,5,6]])
print("Isi dari array sebelum diresize")
print(input)
print("\nIsi dari array setelah diresize")
output = np.resize((input),[3,3])
print(output)
```

```
Isi dari array sebelum diresize
[[1 2 3]
 [4 5 6]]

Isi dari array setelah diresize
[[1 2 3]
 [4 5 6]
 [1 2 3]]
```

Manipulasi Array: Append



numpy.append()

Berfungsi untuk menambahkan value di akhir array

Parameter Fungsi Append()

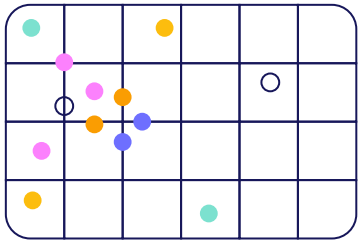
- **Arr** → array awal
- **Values** → Array yang mau digabung. Bentuk harus sama dengan array awal
- **Axis** → Axis dimana penggabungan dilakukan, Default adalah flatten array (1-D)

Contoh:

```
input = np.array([[1,2,3],[4,5,6]])
print("Isi dari array sebelum diappend")
print(input)
print("\nIsi dari array setelah diappend")
output = np.append(input, [7,8,9])
print(output)
```

```
Isi dari array sebelum diappend
[[1 2 3]
 [4 5 6]]

Isi dari array setelah diappend
[1 2 3 4 5 6 7 8 9]
```

Manipulasi Array: Insert



numpy.insert()

Berfungsi untuk menyisipkan value dalam array input di sepanjang axis yang diberikan dan sebelum indeks yang diberikan

Parameter Fungsi Insert()

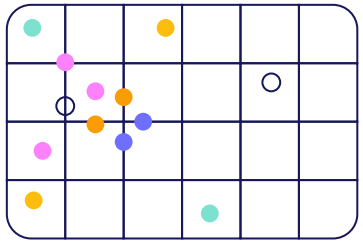
- **Arr** → array awal
- **Obj** → Indeks array dimana value akan disisipkan
- **Values** → Array yang mau disisipkan
- **Axis** → Axis dimana penyisipan dilakukan, Default adalah flatten array (1-D)

Contoh:

```
input = np.array([[1,2],[3,4],[5,6]])
print("Isi dari array sebelum diinsert")
print(input)
print("\nIsi dari array setelah diinsert")
output = np.insert(input, 2, [11,12])
print(output)
```

```
Isi dari array sebelum diinsert
[[1 2]
 [3 4]
 [5 6]]

Isi dari array setelah diinsert
[ 1  2 11 12  3  4  5  6]
```

Manipulasi Array: Delete



numpy.delete()

Berfungsi untuk menghapus value dalam array input di sepanjang axis yang diberikan dan sebelum indeks yang diberikan

Parameter Fungsi Delete()

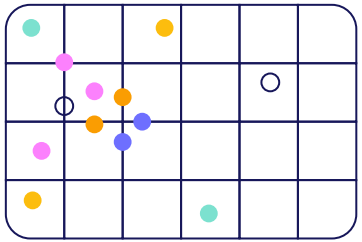
- **Arr** → array awal
- **Obj** → Indeks array dimana value akan dihapus
- **Values** → Array yang mau dihapus
- **Axis** → Axis dimana penyisipan dilakukan, Default adalah flatten array (1-D)

Contoh:

```
input = np.arange(1,9).reshape(2,4)
print("Isi dari array sebelum didelete")
print(input)
print("\nIsi dari array setelah didelete")
output = np.delete(input, 2)
print(output)
```

```
Isi dari array sebelum didelete
[[1 2 3 4]
 [5 6 7 8]]

Isi dari array setelah didelete
[1 2 4 5 6 7 8]
```

Manipulasi Array: Unique



numpy.unique()

Berfungsi untuk mencari elemen unik dalam array

Contoh:

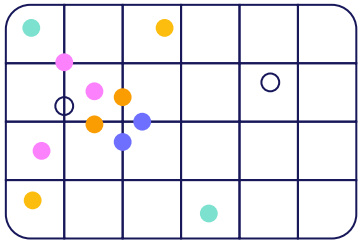
```
input = np.array([5,2,6,2,7,5,6,8,2,4,9,2,2])
print("Isi dari array sebelum cek unik")
print(input)
print("\nIsi dari array setelah cek unik")
output = np.unique(input)
print(output)
```

```
Isi dari array sebelum cek unik
[5 2 6 2 7 5 6 8 2 4 9 2 2]

Isi dari array setelah cek unik
[2 4 5 6 7 8 9]
```

Parameter Fungsi Unique()

- **Arr** → array awal
- **return_index** → menampilkan indeks dari elemen array
- **return_inverse** → menampilkan indeks dari elemen yang unik dalam array
- **return_counts** → menampilkan ada beberapa kali elemen yang unik ditampilkan di array asli

Manipulasi Array: Slicing



numpy.slicing()

Berfungsi untuk mengambil elemen dari satu indeks yang diberikan ke indeks lain

Parameter Fungsi Slicing()

- **Start** → awal interval
- **Stop** → akhir interval
- **Step** → jarak antar value

Contoh:

```
input = np.arange(10)
print("Isi dari array sebelum dislicing")
print(input)
print("\nIsi dari array setelah cek unik")
output = input[2:7:2]
print(output)
```

```
Isi dari array sebelum dislicing
[0 1 2 3 4 5 6 7 8 9]

Isi dari array setelah cek unik
[2 4 6]
```

Thanks!