Python Week #7

Kampus Merdeka x MyEduSolve

Team 3 - Data Science A

DATE TIME MODULE

The datetime module supplies classes for manipulating dates and times. While date and time arithmetic is supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation.

Example :

import date time module

```
# import modul datetime
from datetime import datetime, timedelta
```

```
#contoh untuk fungsi tanggal dan jam sekarang
hari_ini = datetime.now()
print("Tgl hari ini adalah :", hari_ini)
print("Tgl hari ini adalah :", hari_ini.ctime())
```

```
Tgl hari ini adalah : 2022-10-03 09:24:22.392258
Tgl hari ini adalah : Mon Oct  3 09:24:22 2022
```

get current date & time example

```
#mengambil hari, bulan, tahun
print("hari ini adalah hari ke", hari_ini.day)
print("bulan ini adalah hari ke", hari_ini.month)
print("tahun ini adalah hari ke", hari_ini.year)
```

```
hari ini adalah hari ke 3
bulan ini adalah hari ke 10
tahun ini adalah hari ke 2022
```

Get current day, month, and year example

```
#mengambil timestamp waktu
waktu = hari_ini.time()
print(waktu)
```

```
09:24:22.392258
```

Get current timestamp example

```
#mengambl jam, menit, detik pada timestamp
print("Jam :", waktu.hour)
print("Menit :", waktu.minute)
print("Detik:", waktu.second)
```

```
Jam : 9
Menit : 24
Detik: 22
```

Get hours, minutes, and seconds from timestamp

```
hari_ini
```

```
datetime.datetime(2022, 10, 3, 9, 24, 22, 392258)
```

SYS MODULE

This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It is always available.

Example :

import sys module

```
#import module
import sys
```

```
#melihat letak argumen terminal
print(sys.argv)
```

```
['E:\\anaconda3\\lib\\site-packages\\ipykernel_launcher.py', '-f',
```

Print terminal location

```
#melihat versi python dan system
print('versi python dan system:', sys.version)
```

```
versi python dan system: 3.9.12 (main, Apr  4 2022, 05:22:27) [MSC
```

Print python and system version

```
#melihat platform system
print('platform system:', sys.platform)
```

```
platform system: win32
```

Print system platform

```
#melihat letak python interpreter (python.exe)
print('letak python executable:', sys.executable)
```

```
letak python executable: E:\anaconda3\python.exe
```

Print python.exe location

```
#melihat path import module
print('path import:', sys.path)
```

```
path import: ['D:\\Python Data\\Basic Python', 'E:\\anaconda3\\pyt
e-packages\\win32', 'E:\\anaconda3\\lib\\site-packages\\win32\\lib
```

Print import module path

```
# melihat module built-in
print('modul built-in',sys.builtin_module_names)
```

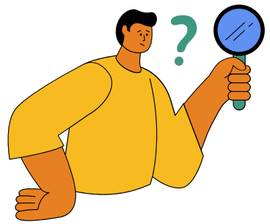
```
...
```

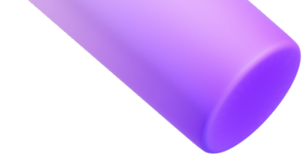
Print built-in module

```
# melihat module import
print('modul import',sys.modules)
```

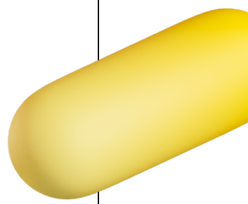
```
...
```

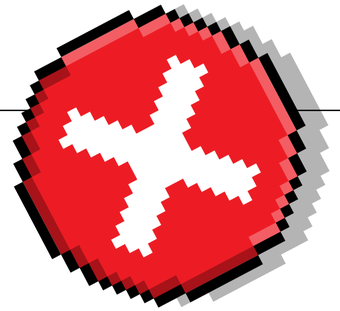
Print import module

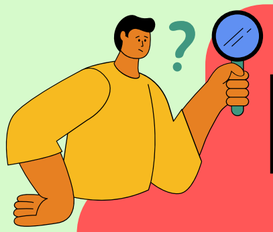




ERROR HANDLING	EXCEPTION HANDLING
• Syntax error • Runtime error • Logic error	• Try • Except • Finally

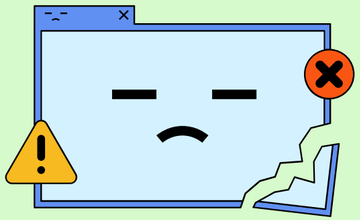






Error Handling

Error handling refers to the routines in a program that respond to abnormal input or conditions. The quality of such routines is based on the clarity of the error messages and the options given to users for resolving the problem.



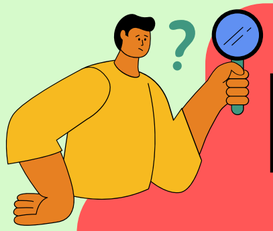
Syntax Error

- Syntax errors are mistakes in the use of the Python language, and are analogous to spelling or grammar mistakes in a language like English: for example, the sentence Would you some tea? does not make sense – it is missing a verb. Common Python syntax errors include: leaving out a keyword.

Example :

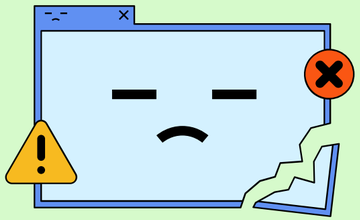
```
#contoh syntax error1
print('Hello)
```

```
  Input In [47]
    print('Hello)
                  ^
SyntaxError: EOL while scanning string literal
```



Error Handling

Error handling refers to the routines in a program that respond to abnormal input or conditions. The quality of such routines is based on the clarity of the error messages and the options given to users for resolving the problem.



Runtime Error

- A run-time error happens when Python understands what you are saying, but runs into trouble when following your instructions. In English, a syntax error would be like the sentence.

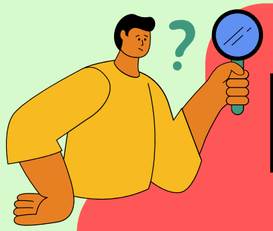
Example :

```
# contoh runtime error 1
suhu = input("masukkan suhu:")

if suhu > 30:
    print("Suhu panas")
elif suhu >=22 and suhu <=30:
    print("Suhu sejuk")
else :
    print("Suhu dingin")
```

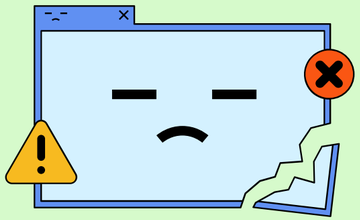
```
masukkan suhu: 29
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [50], in <cell line: 4>()
      1 # contoh runtime rror 1
      2 suhu = input("masukkan suhu:")
----> 4 if suhu > 30:
      5     print("Suhu panas")
      6 elif suhu >=22 and suhu <=30:

TypeError: '>' not supported between instances of 'str' and 'int'
```



Error Handling

Error handling refers to the routines in a program that respond to abnormal input or conditions. The quality of such routines is based on the clarity of the error messages and the options given to users for resolving the problem.



Logic Error

- They occur when the program runs without crashing, but produces an incorrect result. The error is caused by a mistake in the program's logic . You won't get an error message, because no syntax or runtime error has occurred.

Example :

```
# contoh logic error 1
suhu = int(input("masukkan suhu:"))

if suhu > 30:
    print("Suhu dingin")
elif suhu >=22 and suhu <=30:
    print("Suhu sejuk")
else :
    print("Suhu panas")
```

```
masukkan suhu: 31
Suhu dingin
```

Exception Handling

An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. In general, when a Python script encounters a situation that it cannot cope with, it raises an exception. An exception is a Python object that represents an error.

TRY	The try block lets you test a block of code for errors.
EXCEPT	The except block lets you handle the error.
FINALLY	The finally block lets you execute code, regardless of the result of the try- and except blocks.

Exception Handling Example

Code :

Python3

```
# Python code to illustrate
# working of try()
def divide(x, y):
    try:
        # Floor Division : Gives only Fractional
        # Part as Answer
        result = x // y
    except ZeroDivisionError:
        print("Sorry ! You are dividing by zero ")
    else:
        print("Yeah ! Your answer is :", result)
    finally:
        # this block is always executed
        # regardless of exception generation.
        print('This is always executed')

# Look at parameters and note the working of Program
divide(3, 2)
divide(3, 0)
```

Output :

```
Yeah ! Your answer is : 1
This is always executed
Sorry ! You are dividing by zero
This is always executed
```

BASIC OPERATION

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Import Numpy as np

```
import numpy as np
```

List to array

```
# rubah list ke array
np.array(list1)
```

```
array([1, 2, 3, 4])
```

Check array type

```
# buat array
list2 = [[5,6,7,8], [9,10,11,12]]
arr2 = np.array(range(1,7)) # dari tuple
arr3 = np.array([[5,6,7,8],
                 [9,10,11,12]]) #array 2D (matrix)
```

```
#cek tipe array
print(type(list1))
print(type(arr3))
```

```
<class 'list'>
<class 'numpy.ndarray'>
```

Check array dimension

```
#cek dimensi array
print("dimensi array 2 =", arr2.ndim, "dimensi") #vector
print("dimensi array 3 =", arr3.ndim, "dimensi") # matrix
```

```
dimensi array 2 = 1 dimensi
dimensi array 3 = 2 dimensi
```

Check array shape

```
# cek shape array
print(arr2.shape)
print(arr3.shape)
```

```
(6,)
(2, 4)
```

Check total elements

```
# cek total elements
print(arr2.size)
print(arr3.size)
```

```
6
8
```

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

numpy.random.rand

(Random values in a given shape.)

random.rand(d0, d1, ..., dn)

Example :

```
>>> np.random.rand(3,2)
array([[ 0.14022471,  0.96360618],  #random
       [ 0.37601032,  0.25528411],  #random
       [ 0.49313049,  0.94909878]]) #random
```

numpy.random.randint

(Return random integers from low (inclusive) to high (exclusive).)

random.randint(low, high=None, size=None, dtype=int)

Example :

Generate a 2 x 4 array of ints between 0 and 4, inclusive:

```
>>> np.random.randint(5, size=(2, 4))
array([[4, 0, 2, 1], # random
       [3, 2, 2, 0]])
```

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

numpy.random.randn

(Return a sample (or samples) from the “standard normal” distribution.)

random.randn(d0, d1, ..., dn)

Example :

```
>>> np.random.randn()
2.1923875335537315  # random
```

Two-by-four array of samples from N(3, 6.25):

```
>>> 3 + 2.5 * np.random.randn(2, 4)
array([[-4.49401501,  4.00950034, -1.81814867,  7.29718677],   # random
       [ 0.39924804,  4.68456316,  4.99394529,  4.84057254]])  # random
```

numpy.transpose

(Reverse or permute the axes of an array; returns the modified array.)

numpy.transpose(a, axes=None)[source]

Example :

```
>>> x = np.arange(4).reshape((2,2))
>>> x
array([[0, 1],
       [2, 3]])
```

```
>>> np.transpose(x)
array([[0, 2],
       [1, 3]])
```

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

numpy.zeros

(Return a new array of given shape and type, filled with zeros.)

numpy.zeros(shape, dtype=float, order='C', *, like=None)

Example :

```
>>> s = (2,2)
>>> np.zeros(s)
array([[ 0.,  0.],
       [ 0.,  0.]])
```

numpy.ones

(Return a new array of given shape and type, filled with ones.)

numpy.ones(shape, dtype=None, order='C', *, like=None)

Example :

```
>>> s = (2,2)
>>> np.ones(s)
array([[1.,  1.],
       [1.,  1.]])
```

MANIPULATE ARRAY

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

input to array

```
# contoh asarray, mengubah input menjadi array
np.asarray(list1)
```

hstack vertical and horizontal

```
#contoh hstack dan vstack vector
v1 = np.array([1,2,3])
v2 = np.array([4,5,6])
```

```
#print hasil stack
print('vector hstack:', np.hstack((v2,v1)))
print('vector vstack:\n', np.vstack((v1,v2)))
```

```
vector hstack: [4 5 6 1 2 3]
vector vstack:
 [[1 2 3]
 [4 5 6]]
```

Reshape array

```
# contoh reshape, mengubah shape array
arr = np.array([(1,2,3),(4,5,6),(7,8,9),(10,11,12)])
print(arr)

print("\n shape awal:", arr.shape)
```

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]

 shape awal: (4, 3)
```

```
# lakukan reshape, perkalian jumlah shape array harus sama
print(arr.reshape(2,6))
print("\nsetelah direshape:", arr.reshape(2,6).shape)
```

```
[[ 1  2  3  4  5  6]
 [ 7  8  9 10 11 12]]

setelah direshape: (2, 6)
```

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

ARRAY OPERATION

```
# buat array baru (random)
arr1 = np.random.randint(1,10,10)
print("array1 =", arr1)
```

```
array1 = [9 6 4 7 7 3 9 3 2 5]
```

Create new array using random

```
# buat array baru kedua
arr2 = np.random.randint(1,10,10)
print("array2 =", arr2)
```

```
array2 = [8 1 6 8 5 3 5 4 4 5]
```

```
# contoh operasi pada element array(dengan operator)
print("tambah =", arr1 +2)
print("kurang =", arr1 -2)
print("kali =", arr1 * 2)
print("bagi =", arr1 / 2)
print("modulo =", arr1 % 2)
print("pangkat =", arr1 ** 2)
```

```
tambah = [11  8  6  9  9  5 11  5  4  7]
kurang = [7 4 2 5 5 1 7 1 0 3]
kali = [18 12  8 14 14  6 18  6  4 10]
bagi = [4.5 3.  2.  3.5 3.5 1.5 4.5 1.5 1.  2.5]
modulo = [1 0 0 1 1 1 1 1 0 1]
pangkat = [81 36 16 49 49  9 81  9  4 25]
```

Array operation example

```
# contoh operasi pada element array(dengan operator)
print("arr1+arr2 =", arr1 + arr2)
print("arr1-arr2 =", arr1 - arr2)
print("arr1*arr2 =", arr1 * arr2)
print("arr1/arr2 =", arr1 / arr2)
print("arr1%arr2 =", arr1 % arr2)
print("arr1**arr2 =", arr1 ** arr2)
```

```
arr1+arr2 = [17  7 10 15 12  6 14  7  6 10]
arr1-arr2 = [ 1  5 -2 -1  2  0  4 -1 -2  0]
arr1*arr2 = [72  6 24 56 35  9 45 12  8 25]
arr1/arr2 = [1.125      6.         0.66666667 0.875      1.4        1.
 1.8        0.75       0.5        1.         ]
arr1%arr2 = [1 0 4 7 2 0 4 3 2 0]
arr1**arr2 = [43046721        6     4096  5764801    16807       27    59049       81
       16     3125]
```

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

ARRAY OPERATION

```
# fungsi sqrt
np.sqrt(arr1)
```

```
array([1.        , 2.23606798, 2.44948974, 2.44948974, 2.64575131,
       3.        , 2.23606798, 1.41421356, 2.23606798, 3.        ])
```

```
# fungsi exp
np.exp(arr1)
```

```
array([2.71828183e+00, 1.48413159e+02, 4.03428793e+02, 4.03428793e+02,
       1.09663316e+03, 8.10308393e+03, 1.48413159e+02, 7.38905610e+00,
       1.48413159e+02, 8.10308393e+03])
```

```
#fungsi max
np.max(arr1)
```

```
9
```

```
#fungsi min
np.min(arr1)
```

```
1
```

```
# fungsi add
np.add(arr1, arr2)
```

```
array([ 3,  7, 14, 10, 14, 17,  8,  3,  9, 15])
```

```
# fungsi subtract
np.subtract(3,arr1) # 3 - arr1
```

```
array([ 2, -2, -3, -3, -4, -6, -2,  1, -2, -6])
```

```
# fungsi divide
np.divide(arr1,2)
```

```
array([0.5, 2.5, 3. , 3. , 3.5, 4.5, 2.5, 1. , 2.5, 4.5])
```

```
# fungsi multiply
np.multiply(arr1,arr2)
```

```
array([ 2, 10, 48, 24, 49, 72, 15,  2, 20, 54])
```

```
# fungsi sum
np.sum(arr1)
```

```
55
```

sqrt()

exp()

max()

min()

add()

subtract()

divide()

multiply()

sum()

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

LINEAR ALGEBRA

numpy.linalg.solve

linalg.solve(a, b)

(Solve a linear matrix equation, or system of linear scalar equations.)

Example :

a) $2x + 3y = 10$

```
# step1 masukkan persamaan kedalamm bentuk list
hitungan = [[2,3], [1,2]]
hasil = [10,5]

# step2 rubah ke array
arr_hit = np.asarray(hitungan)
arr_has = np.asarray(hasil)

# print hasil array
print("array_hitung:", arr_hit)
print("array hasil:", arr_has)
```

```
array_hitung: [[2 3]
 [1 2]]
array hasil: [10  5]
```

```
# solve dengan fungsi linalg.solve
hasil_pers = np.linalg.solve(arr_hit, arr_has)
print("hasil perhitungan linear aljabar =", hasil_pers)
print("hasil x1 adalah =", hasil_pers[0])
print("hasil x2 adalah =", hasil_pers[1])
```

```
hasil perhitungan linear aljabar = [5. 0.]
hasil x1 adalah = 5.0
hasil x2 adalah = 0.0
```

insert the equality to list

list to array

print array

solve using linalg.solve

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

BASIC STATISTIC

min(), max(), mean(), median(), std(), quantile(), percentile()

Example:

```
#contoh buat array baru
norm_arr = np.random.normal(5, 0.5, 10)
print(norm_arr)
```

```
[4.40321087 5.57052122 5.75472254 5.53388756 4.65670526 5.00743666
 4.81216705 4.98088818 5.18398724 4.97763815]
```

#create new array using random

```
# contoh beberapa statistical function
print("nilai min :", np.min(norm_arr))
print("nilai max :", np.max(norm_arr))
print("nilai mean :", np.mean(norm_arr))
print("nilai median :", np.median(norm_arr))
print("nilai stdev :", np.std(norm_arr))
print("nilai quantile :", np.quantile(norm_arr, .25))
print("nilai percentile :", np.percentile(norm_arr, .25))
```

```
nilai min : 4.403210874027754
nilai max : 5.754722541132711
nilai mean : 5.088116474239295
nilai median : 4.994162419653291
nilai stdev : 0.40577740910590865
nilai quantile : 4.8535348264072695
nilai percentile : 4.408914497743704
```

#statistical function example