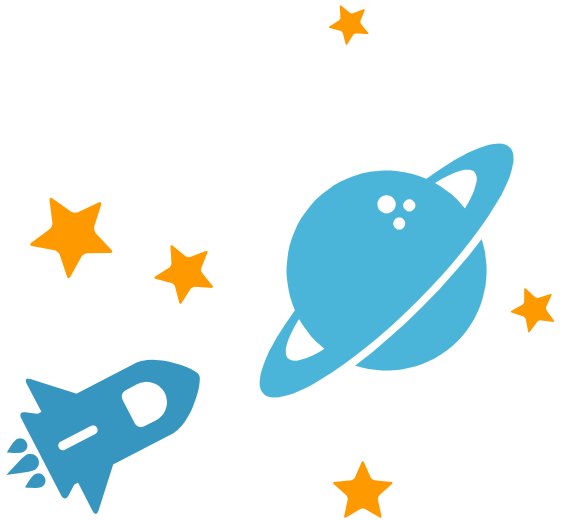
DigitalSkola

LEARNING PROGRESS REVIEW WEEK 5

OPTIMISTIC TEAM



TEAM MEMBER:

ALDIVA WIBOWO

ASPRIZAL RIZKY

GILANG RAHMAT R

LUTFIA HUMAIROSI

MILLENIA WINADYA P

1. BASIC PROGRAMMING ; FUNCTION

KONSEP FUNCTIONAL PROGRAMMING

- Pemrograman fungsional adalah paradigma pemrograman di mana kita mencoba untuk mengikat segala sesuatu dalam gaya fungsi matematika murni. Ini adalah jenis gaya pemrograman deklaratif. Fokus utamanya adalah pada "apa yang harus dipecahkan" berbeda dengan gaya imperatif di mana fokus utamanya adalah "bagaimana menyelesaikannya". Ini menggunakan ekspresi alih-alih pernyataan. Ekspresi dievaluasi untuk menghasilkan nilai sedangkan pernyataan dieksekusi untuk menetapkan variabel.

MANFAAT FUNCTION

1. Ketika Anda memiliki proses berulang, Anda dapat menggunakan fungsi untuk menyederhanakan kode.
2. Pisahkan setiap alur kerja sesuai fungsinya. Misalnya, Bersihkan data, Panggil data dari csv, dll.
3. Memiliki fungsi dalam kode akan memudahkan kita untuk fokus pada pemecahan masalah, bukan debugging proses yang sama lagi dan lagi.
4. Setiap masalah atau kesalahan mudah dilacak melalui fungsi.
5. Relatif berguna untuk mengurangi penggunaan memori.

DigitalSkola

KOMPONEN FUNCTION

INPUT

- Dalam python setiap input disebut parameter. Jadi, ketika 'parameter' disebutkan, ingatlah bahwa itu adalah input dari suatu fungsi.
- Fungsi dalam Python cukup fleksibel dibandingkan dengan bahasa pemrograman lain seperti C++ atau Java karena Python tidak secara ketat mendefinisikan tipe data parameternya.
- Kita dapat memasukkan tipe data apa saja di dalam parameter, misalnya string, integer, float, array, dictionary, bahkan fungsi itu sendiri.

OUTPUT

- Keluaran dari fungsi dapat berupa pesan atau nilai. Tergantung bagaimana kita mendefinisikannya.
- Untuk menghasilkan pesan, kita dapat menggunakan print. Sedangkan untuk menghasilkan nilai kita bisa menggunakan return.
- Fungsi Python dapat menghasilkan beberapa nilai sekaligus.
- Jika kita ingin memilih nilai tertentu dari beberapa return, kita dapat menggunakan garis bawah (_).

PROCESS

- Proses di dalam fungsi seperti urutan langkah-langkah untuk mencapai sesuatu (seperti output).
- Variabel lokal di dalam fungsi hanya bisa tinggal di dalamnya dan akan dihapus setelahnya.
- Kita dapat melakukan hampir semua hal di dalam fungsi, seperti perulangan, pernyataan kondisional, panggilan fungsi lainnya.
- Setiap proses dari perpustakaan manapun juga dapat bekerja di dalam fungsi.

VARIABLE LOCAL DAN GLOBAL

- Variabel Global adalah variabel yang bisa diakses dari semua fungsi, sedangkan variabel lokal hanya bisa diakses di dalam fungsi tempat ia berada saja.
- Pada Python, urutan pengaksesan variabel *(scope)* dikenal dengan sebutan LGB (Local, Global, dan Build-in).
- Jadi program python mulai mencari vairabel lokal terlebih dahulu, kalau ada maka itu yang digunakan.
- Tapi kalau tidak ada, pencarian terus ke Global, dan Build-in.
- Variabel Build-in adalah variabel yang sudah ada di dalam Python.

CREATE FUCTION

- ARITHMETIC

```
[ ]  def pembagian(a, b, c):
       d = a / b / c
       print("Proses Selesai")
       return d

     angka_pertama = 10
     angka_kedua = 5
     angka_ketiga = 2

     e = pembagian(angka_pertama, angka_kedua, angka_ketiga)

     print("Hasil pembagian adalah:", e)
```

```
Proses Selesai
Hasil pembagian adalah: 1.0
```

CREATE FUCTION

- ARITHMETIC

```
[ ]  def pembagian(a, b, c):
       d = a / b / c
       print("Proses Selesai")
       return d

     angka_pertama = 10
     angka_kedua = 5
     angka_ketiga = 2

     e = pembagian(angka_pertama, angka_kedua, angka_ketiga)

     print("Hasil pembagian adalah:", e)
```

```
Proses Selesai
Hasil pembagian adalah: 1.0
```

CREATE FUCTION

- FUNCTION WITH CONDITIONAL STATEMNT

```
[ ]  def nilaiUjian(nilai, th_atas = 80, th_bwh = 60):
       if nilai > th_atas:
         keputusan = 'sangat baik'
       elif nilai < th_bwh:
         keputusan = 'remidi'
       else:
         keputusan = 'oke'

       return keputusan

     print(nilaiUjian(90))
```

```
sangat baik
```

CREATE FUCTION

» Function with for loop and function inside function

```
[ ]  list_nilai = [90,80,30,50,20,10,52,49,80,75]

     def koreksiNilai(kumpulan):
         for i in kumpulan:
             keputusan = nilaiUjian(i)
             print('Nilai', i, keputusan)

     koreksiNilai(list_nilai)
```

```
Nilai 90 sangat baik
Nilai 80 oke
Nilai 30 remidi
Nilai 50 remidi
Nilai 20 remidi
Nilai 10 remidi
Nilai 52 remidi
Nilai 49 remidi
Nilai 80 oke
Nilai 75 oke
```

INTRODUCTION TO LIBRARY

- *Python library* merupakan sebuah kumpulan *function* dan *method* yang memudahkan kita dalam melakukan analisis data. *Library* biasanya mengandung *built-in module* yang menyediakan fungsi-fungsi yang berbeda yang dapat langsung kita gunakan.

Contoh: Pandas, Numpy, Matplotlib, Scipy, Seaborn, Scikit-Learn

STRING MANUPULATION

- Memanipulasi string dalam Ilmu Data adalah keterampilan penting yang harus dimiliki. Karena terkadang data disediakan dalam teks kotor. Ada cabang Ilmu Data tingkat lanjut yang hanya berspesialisasi dalam bahasa dan disebut Natural Language Processing. Ini mendapatkan wawasan dari teks dan bahkan suara.

Concatenate

```
[ ]  #concatenate
     message = 'Aldiva' + ' ' + 'Wibowo'
     print(message)
```

```
Aldiva Wibowo
```

```
[ ]  def nama(depan, belakang):
       lengkap = depan + ' ' + belakang
       return lengkap

     print(nama("Aldiva", "Wibowo"))
```

```
Aldiva Wibowo
```

Multiply

```
[1]  #multiply
     star = '*' * 10

     print(star)
```

```
**********
```

Length

```
[ ]  message2 = 'awdakwdadjal wajkwldjawldjalwk alwkdjalkwdjalwkdjalkwj klajdklaj

     len(message2)
```

```
102
```

```
[ ]  # String length
     message2 = 'lalalalala~12412312312f12f12f11212f121we12f11we212f21we1f12f21e'
     if len(message2) > 30:
         print('kepanjangan!')
     else:
         print('cukup')
```

```
kepanjangan!
```

Lowercase, Uppercase, Capitalized, Title

```
[ ]  text = 'AldIvA wIbOwO'

     # lower case
     lower = text.lower()
     print(lower)

     # upper case
     upper = text.upper()
     print(upper)

     #Capitalized
     capitalize = text.capitalize()
     print(capitalize)

     #Title
     title = text.title()
     print(title)
```

```
aldiva wibowo
ALDIVA WIBOWO
Aldiva wibowo
Aldiva Wibowo
```

Slice

```
[ ]  #Slice
     print(message3[3:7])
```

```
lo W
```

Replace

```
[ ]  #Replace
     message3 = 'Hello World'
     message4 = message3.replace('l', 'r')
     print(message4)
```

```
Herro Worrd
```

2. DATABASE PROGRAMMING

DATABASE PROGRAMMING

- **File**
- File adalah kumpulan bytes yang digunakan untuk menyimpan data. Urutan untuk melakukan operasi di file di python ialah, Open a file, Read or write (perform operation), dan Close the file.

- **File Paths**
- Untuk mengakses file di operating system maka dibuthkan file paths nya, file path mendeskripsikan dimana lokasi dari suatu file berdasarkan struktur foldernya.

DATABASE PROGRAMMING

- **Opening a File Stream**
- Beberapa operasi yang dapat dilakukan
  1. Reading
  2. Writing
  3. Append
  4. Reading and Writing
  5. Text
  6. Binary

DATABASE PROGRAMMING

- **Writing Files**
- Code: write(str)

Contoh:

Output = open(“example.txt”)

Output.write(“Welcome to Digital Skola”)

DATABASE PROGRAMMING

- **Writing Files**
- Code: write(str)

Contoh:

Output = open(“example.txt”)

Output.write(“Welcome to Digital Skola”)

DATABASE PROGRAMMING

- **Reading Files**
- Code:
- read( ) – reads all characters from a file
- readline( ) – reads all characters up to an inclusing \n
- raedlines( ) – reads all lines

Contoh:

Output = open (“example.txt”)

Print(output.read( ))

DATABASE PROGRAMMING

- **Database Progrmming in Python**
- Bahasa pemrograman python memiliki fitur canggih untuk database programming.
- Python support berbagai macam databases seperti MySQL, Oracle, Postgrasql, dan lainnya.
- Python juga supports Data Defiition Language (DDL), Data Manipulation Language (DML), dan Data Query Statements.

DATABASE PROGRAMMING

» Database Connection

```
import psycopg2

#connection database
conn = psycopg2.connect(
        host = "digitalskoladb.c04me33o8tni.ap-southeast-1.rds.amazonaws.com",
        port = 5432,
        database= "sandbox",
        user = "group_a",
        password = "hZrJQJ8RMttts9VZnUYZX93pr")

#creating object using the cursor method
cursor = conn.cursor()
```

DATABASE PROGRAMMING

» Create Table

```
conn = psycopg2.connect(
        host = "digitalskoladb.c04me33o8tni.ap-southeast-1.rds.amazonaws.com",
        port = 5432,
        database= "sandbox",
        user = "group_a",
        password = "hZrJQJ8RMttts9VZnUYZX93pr")

#creating object using the cursor method
cursor = conn.cursor()

cursor.execute("DROP TABLE IF EXISTS sandbox.batch_2.bus")

sql = '''
      create table sandbox.batch_2.bus (
      id int,
      origin varchar,
      destination varchar,
      time time
)'''

cursor.execute(sql)
print("Table created successfully..")
```

DATABASE PROGRAMMING

» Insert Data

```
[ ]  conn = psycopg2.connect(
            host = "digitalskoladb.c04me33o8tni.ap-southeast-1.rds.amazonaws.com",
            port = 5432,
            database= "sandbox",
            user = "group_a",
            password = "hZrJQJ8RMttts9VZnUYZX93pr")

    #creating object using the cursor method
    cursor = conn.cursor()

    sql ='''insert into sandbox.batch_2.bus values
            (100, 'Jakarta', 'Bandung', '10:00'),
            (200, 'Jakarta', 'Palembang', '11:00'),
            (300, 'Jakarta', 'Jogja', '12:00')'''

    cursor.execute(sql)

    conn.commit()
    print("Data inserted successfully..")

    conn.close()
```

DATABASE PROGRAMMING

» Select Data

```
conn = psycopg2.connect(
        host = "digitalskoladb.c04me33o8tni.ap-southeast-1.rds.amazonaws.com",
        port = 5432,
        database= "sandbox",
        user = "group_a",
        password = "hZrJQJ8RMttts9VZnUYZX93pr")

#creating object using the cursor method
cursor = conn.cursor()

sql ='''select * from sandbox.batch_2.bus'''

cursor.execute(sql)

result = cursor.fetchall();
print(result)

conn.commit()

conn.close()
```

```
[(100, 'Jakarta', 'Bandung', datetime.time(10, 0)), (200, 'Jakarta', 'Palembang', datetime.time(11, 0)), (300, 'Jakarta', 'Jogja', datetime.time(12, 0))]
```

DATABASE PROGRAMMING

» Update Table

```
conn = psycopg2.connect(
        host = "digitalskoladb.c04me33o8tni.ap-southeast-1.rds.amazonaws.com",
        port = 5432,
        database= "sandbox",
        user = "group_a",
        password = "hZrJQJ8RMttts9VZnUYZX93pr")

#creating object using the cursor method
cursor = conn.cursor()

print('Data sebelum diupdate')
sql = '''
      select * from sandbox.batch_2.bus
'''
cursor.execute(sql)
print(cursor.fetchall())
```

```
print('Data sesudah diupdate')
sql = '''
      update sandbox.batch_2.bus
      set id = 100, origin = 'Medan', destination = 'Bandung', time = '10:00'
      where id = 100
    '''
cursor.execute(sql)

sql = '''
      select * from sandbox.batch_2.bus
'''
cursor.execute(sql)
print(cursor.fetchall())

conn.commit()

conn.close()
```

DATABASE PROGRAMMING

» Drop Table

```
conn = psycopg2.connect(
        host = "digitalskoladb.c04me33o8tni.ap-southeast-1.rds.amazonaws.com",
        port = 5432,
        database= "sandbox",
        user = "group_a",
        password = "hZrJQJ8RMttts9VZnUYZX93pr")

#creating object using the cursor method
cursor = conn.cursor()

sql = '''
      drop table sandbox.batch_2.bus
'''

cursor.execute(sql)
print("Table sudah dihapus")

conn.commit()

conn.close()
```

```
Table sudah dihapus
```

3. INTRODUCTION TO NUMPY

APA ITU NUMPY?

- Numpy merupakan salah satu library python yang berfungsi untuk scientific computing.
- Numpy digunakan untuk fungsi array.

KEGUNAAN NUMPY

- Komputasi dan manipulasi matrix
- Menyortir dan memilih data
- Operasi aljabar linear
- Operasi mathematics dan statistics dasar
- Sebagai alternative penggunaan fungsi lists
- Meminimalisir penggunaan memori
- Numpy dapat dikolaborasikan dengan Pandas, Matplotlib, SciPy, dan tkinter.

MENGAPA HARUS MENGGUNAKAN NUMPY?

- Dalam python, numpy berguna untuk menyediakan objek array lebih cepat daripada fungsi list.
- Numpy dibuat menggunakan bahasa pemrograman C/C++ sehingga prosesnya lebih cepat.
- Penggunaan numpy mudah dimengerti dan mudah dijalankan untuk keperluan komputasi.

DigitalSkola

Python For Data Science *Cheat Sheet*

NumPy Basics

Learn Python for Data Science Interactively at www.DataCamp.com

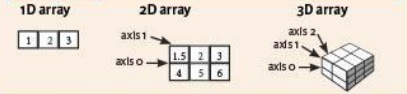
NumPy

The NumPy library is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays.

Use the following import convention:

```
>>> import numpy as np
```

NumPy Arrays



Creating Arrays

```
>>> a = np.array([1,2,3])
>>> b = np.array([[1.5,2,3), (4,5,6)], dtype = float)
>>> c = np.array([[(1.5,2,3), (4,5,6)], [(3,2,1), (4,5,6)]],
                 dtype = float)
```

Initial Placeholders

```
>>> np.zeros((3,4))                  Create an array of zeros
>>> np.ones((2,3,4),dtype=np.int16)  Create an array of ones
>>> d = np.arange(10,25,5)           Create an array of evenly
                                     spaced values (step value)
>>> np.linspace(0,2,9)               Create an array of evenly
                                     spaced values (number of samples)
>>> e = np.full((2,2),7)             Create a constant array
>>> f = np.eye(2)                    Create a 2X2 identity matrix
>>> np.random.random((2,2))          Create an array with random values
>>> np.empty((3,2))                  Create an empty array
```

I/O

Saving & Loading On Disk

```
>>> np.save('my_array', a)
>>> np.savez('array.npz', a, b)
>>> np.load('my_array.npy')
```

Saving & Loading Text Files

```
>>> np.loadtxt("myfile.txt")
>>> np.genfromtxt("my_file.csv", delimiter=',')
>>> np.savetxt("myarray.txt", a, delimiter=" ")
```

Data Types

```
>>> np.int64       Signed 64-bit integer types
>>> np.float32     Standard double-precision floating point
>>> np.complex     Complex numbers represented by 128 floats
>>> np.bool        Boolean type storing TRUE and FALSE values
>>> np.object      Python object type
>>> np.string_     Fixed-length string type
>>> np.unicode_    Fixed-length unicode type
```

Inspecting Your Array

```
>>> a.shape          Array dimensions
>>> len(a)           Length of array
>>> b.ndim           Number of array dimensions
>>> e.size           Number of array elements
>>> b.dtype          Data type of array elements
>>> b.dtype.name     Name of data type
>>> b.astype(int)    Convert an array to a different type
```

Asking For Help

```
>>> np.info(np.ndarray.dtype)
```

Array Mathematics

Arithmetic Operations

```
>>> g = a - b                    Subtraction
  array([[-0.5,  0. ,  0. ],
         [-3. , -3. , -3. ]])
>>> np.subtract(a,b)             Subtraction
>>> b + a                        Addition
  array([[ 2.5,  4. ,  6. ],
         [ 5. ,  7. ,  9. ]])
>>> np.add(b,a)                  Addition
>>> a / b                        Division
  array([[ 0.66666667,  1.        ,  1.        ],
         [ 0.25      ,  0.4       ,  0.5       ]])
>>> np.divide(a,b)               Division
>>> a * b                        Multiplication
  array([[  1.5,   4. ,   9. ],
         [  4. ,  10. ,  18. ]])
>>> np.multiply(a,b)             Multiplication
>>> np.exp(b)                    Exponentiation
>>> np.sqrt(b)                   Square root
>>> np.sin(a)                    Print sines of an array
>>> np.cos(b)                    Element-wise cosine
>>> np.log(a)                    Element-wise natural logarithm
>>> e.dot(f)                     Dot product
  array([[ 7.,  7.],
         [ 7.,  7.]])
```

Comparison

```
>>> a == b                       Element-wise comparison
  array([[False,  True,  True],
         [False, False, False]], dtype=bool)
>>> a < 2                        Element-wise comparison
  array([True, False, False], dtype=bool)
>>> np.array_equal(a, b)         Array-wise comparison
```

Aggregate Functions

```
>>> a.sum()              Array-wise sum
>>> a.min()              Array-wise minimum value
>>> b.max(axis=0)        Maximum value of an array row
>>> b.cumsum(axis=1)     Cumulative sum of the elements
>>> a.mean()             Mean
>>> b.median()           Median
>>> a.corrcoef()         Correlation coefficient
>>> np.std(b)            Standard deviation
```

Copying Arrays

```
>>> h = a.view()         Create a view of the array with the same data
>>> np.copy(a)           Create a copy of the array
>>> h = a.copy()         Create a deep copy of the array
```

Sorting Arrays

```
>>> a.sort()             Sort an array
>>> c.sort(axis=0)       Sort the elements of an array's axis
```

Subsetting, Slicing, Indexing

Also see Lists

Subsetting

```
>>> a[2]
  3
```
Select the element at the 2nd index

```
>>> b[1,2]
  6.0
```
Select the element at row 1 column 2 (equivalent to b[1][2])

Slicing

```
>>> a[0:2]
  array([1, 2])
```
Select items at index 0 and 1

```
>>> b[0:2,1]
  array([ 2.,  5.])
```
Select items at rows 0 and 1 in column 1

```
>>> b[:1]
  array([[1.5, 2., 3.]])
```
Select all items at row 0 (equivalent to b[0:1, :])

```
>>> c[1,...]
  array([[[ 3.,  2.,  1.],
          [ 4.,  5.,  6.]]])
```
Same as [1, :, :]

```
>>> a[ : :-1]
  array([3, 2, 1])
```
Reversed array a

Boolean Indexing

```
>>> a[a<2]
  array([1])
```
Select elements from a less than 2

Fancy Indexing

```
>>> b[[1, 0, 1, 0],[0, 1, 2, 0]]
  array([ 4. ,  2. ,  6. ,  1.5])
```
Select elements (1,0),(0,1),(1,2) and (0,0)

```
>>> b[[1, 0, 1, 0]][:,[0,1,2,0]]
  array([[ 4. ,  5. ,  6. ,  4. ],
         [ 1.5,  2. ,  3. ,  1.5],
         [ 4. ,  5. ,  6. ,  4. ],
         [ 1.5,  2. ,  3. ,  1.5]])
```
Select a subset of the matrix's rows and columns

Array Manipulation

Transposing Array

```
>>> i = np.transpose(b)
>>> i.T
```
Permute array dimensions
Permute array dimensions

Changing Array Shape

```
>>> b.ravel()
>>> g.reshape(3,-2)
```
Flatten the array
Reshape, but don't change data

Adding/Removing Elements

```
>>> h.resize((2,6))
>>> np.append(h,g)
>>> np.insert(a, 1, 5)
>>> np.delete(a,[1])
```
Return a new array with shape (2,6)
Append items to an array
Insert items in an array
Delete items from an array

Combining Arrays

```
>>> np.concatenate((a,d),axis=0)
  array([ 1,  2,  3, 10, 15, 20])
```
Concatenate arrays

```
>>> np.vstack((a,b))
  array([[ 1. ,  2. ,  3. ],
         [ 1.5,  2. ,  3. ],
         [ 4. ,  5. ,  6. ]])
```
Stack arrays vertically (row-wise)

```
>>> np.r_[e,f]
>>> np.hstack((e,f))
  array([[ 7.,  7.,  1.,  0.],
         [ 7.,  7.,  0.,  1.]])
```
Stack arrays vertically (row-wise)
Stack arrays horizontally (column-wise)

```
>>> np.column_stack((a,d))
  array([[ 1, 10],
         [ 2, 15],
         [ 3, 20]])
```
Create stacked column-wise arrays

```
>>> np.c_[a,d]
```
Create stacked column-wise arrays

Splitting Arrays

```
>>> np.hsplit(a,3)
  [array([1]),array([2]),array([3])]
```
Split the array horizontally at the 3rd index

```
>>> np.vsplit(c,2)
  [array([[[ 1.5,  2. ,  1. ],
           [ 4. ,  5. ,  6. ]]]),
   array([[[ 3. ,  2. ,  3. ],
           [ 4. ,  5. ,  6. ]]])]
```
Split the array vertically at the 2nd index

DataCamp
Learn Python for Data Science Interactively

THANKS!