ACARA PRAKTIKUM 1

PENGANTAR PYTHON

TUJUAN

- Mengetahui dasar-dasar pemrograman bahasa pemrograman python.
- Terampil menggunakan python untuk menyelesaikan permasalahan.

DASAR TEORI

1. Pengenalan Python
   a. Apa itu Python?

      Python adalah bahasa pemrograman tingkat tinggi yang menggunakan ***interpreter***, **beorientasi objek** dengan **semantik yang dinamis**. Python diciptakan oleh Guido van Rossum dan dirilis pertama kali pada tahun 1991. Sejak itu, Python telah berkembang menjadi salah satu bahasa pemrograman paling populer di dunia.

   b. Mengapa Mempelajari Python?

      Python digunakan pada pembelajaran praktikum kecerdasan buatan karena beberapa alasan berikut:

      1) Syntax yang sederhana sehingga mudah untuk dipelajari.
      2) Ekosistem library yang luas untuk pengembangan kecerdasan buatan.
      3) Dapat digunakan diberbagai bidang seperti *web development*, *data science*, IOT, dll yang dapat diintegrasikan dengan kecerdasan buatan.
      4) Komunitas yang besar dan aktif.

2. Dasar Pemrograman Python
   a. Syntax Dasar Python
      1) Komentar

         Komentar adalah teks dalam kode program yang tidak dieksekusi oleh program. Komentar digunakan untuk memberikan penjelasan atau catatan dalam kode.

         a) Komentar satu baris (*single-line comments*)

            Komentar satu baris dimulai dengan tanda pagar (#). Semua teks setelah tanda pagar (#) pada baris tersebut akan dianggap sebagai komentar.

```
# Ini adalah komentar satu baris
print("Hello, World!") # Ini juga komentar
```

b) Komentar multi-baris (*multi-line comments*)

Komentar multi-baris merupakan komentar antar baris yang diapit oleh *triple single quotes* ('''…''') atau *triple double quotes* (""""..."""").

```
"""
Ini adalah komentar multi baris
bagian ini tidak akan dieksekusi oleh program
"""

'''
Ini juga komentar multi baris
bagian ini juga tidak akan dieksekusi oleh program
'''
```

2) Variabel

Variabel adalah tempat untuk menyimpan nilai atau data dalam memori computer untuk mereferensikan atau menggunakan nilai tersebut di berbagai bagian program.

Python merupakan bahasa pemrograman yang *loosely typed*, artinya pengguna tidak harus menentukan tipe data variabel secara eksplisit saat mendeklarasikannya karena python dapat secara otomatis menentukan tipe data berdasarkan nilai yang diberikan.

Untuk memberikan atau menetapkan nilai pada variabel dilakukan dengan melakukan *assignment*.

variabel = nilai

Contoh:

```
x = 11
nama = “Fardan Maula Azizi”
```

3) Input

Untuk mendapatkan suatu inputan dari pengguna dapat menggunakan fungsi `input()`

variabel = input('Pesan untuk pengguna: ')

Contoh :

```
name = input("Masukkan nama kamu: ")
```

Ketika code dijalankan

```
Output:
```

```
Masukkan nama kamu: Alvin Aryanta
```

Dalam contoh di atas, nilai yang dimasukkan adalah "Alvin Aryanta". Variabel `name` akan menyimpan nilai "Alvin Aryanta" yang dimasukkan oleh pengguna, dan nilai ini dapat digunakan di bagian lain dari program.

4) Output

Untuk menampilkan sesuatu kepada pengguna dapat menggunakan fungsi `print()`.

print(nilai atau variabel)

Contoh:

a) Menampilkan teks sederhana

```
print(“Hello World!”)
```

Ketika code dijalankan

```
Output:
Hello World!
```

b) Menampilkan nilai dari variabel

```
text = "Informatika Unsoed"
print(text)
```

Ketika code dijalankan

```
Output:
Informatika Unsoed
```

c) Menampilkan gabungan teks dan variabel

```
universitas = "Unsoed"
print("Saya berkuliah di", universitas)
print(f"Saya berkuliah di {universitas}")
print("Saya berkuliah di {}.".format(universitas))
```

Ketika code dijalankan

```
Output:
Saya berkuliah di Unsoed
Saya berkuliah di Unsoed
Saya berkuliah di Unsoed.
```

b. Operasi Dasar Python

1) Operasi Aritmatika

Operasi	Simbol	Contoh
Penjumlahan (a + b)	+	11 + 4
Pengurangan (a - b)	-	4 - 2

Perkalian (a × b)	*	11 * 2
Pembagian (a : b)	/	12 / 4
Pangkat (a^b)	**	2 ** 4

2) Operasi Perbandingan

Operasi	Simbol	Contoh
Sama dengan	==	a == b
Tidak sama dengan	!=	a != b
Lebih besar	>	a > b
Lebih kecil	<	a < b
Lebih besar atau sama dengan	>=	a >= b
Lebih kecil atau sama dengan	<=	a <= b

3) Operasi Logika

Operasi		Simbol	Contoh
Dan	Mengembalikan *True* jika kedua pernyataan benar	and	x > 4 and x < 11
Atau	Mengembalikan *True* jika salah satu pernyataan benar	or	x > 4 or x < 11
Tidak	Membalikkan hasil, mengembalikan *False* jika hasilnya benar	not	not(x > 4 and x < 11)

c. Struktur Kontrol

Struktur kontrol adalah cara mengarahkan alur eksekusi program berdasarkan kondisi tertentu.

1) Percabangan

Percabangan adalah alur pemrograman yang membuat keputusan berdasarkan kondisi tertentu. Hal tersebut memungkinkan program untuk memilih jalur eksekusi yang berbeda tergantung pada apakah suatu kondisi benar (*True*) atau salah (*False*).

```
if kondisi1 :
    #blok kode yang dieksekusi apabila kondisi1 benar
elif kondisi2 :
```

#blok kode yang dieksekusi apabila kondisi2 benar

else :

#blok kode yang dieksekusi apabila semua kondisi sebelumnya salah

Contoh:

```
if nilai>=80:
   print("Anda mendapat nilai A")
elif nilai>=70:
   print("Anda mendapat nilai B")
elif nilai>=60:
   print("Anda mendapat nilai C")
else:
   print("Anda mendapat nilai D")
```

2) Perulangan

Perulangan adalah alur pemrograman untuk mengulangi eksekusi kode beberapa kali selama memenuhi kondisi. Hal tersebut memungkinkan program melakukan tindakan yang sama berulang kali.

a) For

For adalah perulangan yang dilakukan untuk setiap item dalam koleksi atau rentang angka. Digunakan Ketika tahu jumlah perulangan atau ingin mengulangi setiap elemen dalam koleksi.

for variabel in iterable:

#blok kode yang akan diulang

Contoh:

```
var_list = [1,2,3,4]
for i in var_list:
    print(i)
```

Ketika code dijalankan

```
Output :
1
2
3
4
```

Perulangan for juga dapat dilakukan berdasarkan panjang suatu nilai yang didefinisikan dengan menggunakan fungsi `range()`

range(start, stop, step)

- start merupakan nilai awal dari urutan bilangan, bersifat **opsional** dengan nilai default 0 dan bersifat **inklusif**.
- stop merupakan nilai akhir dari urutan bilangan, **wajib** didefinisikan dan bersifat **ekslusif**.
- step merupakan nilai penambahan antara dua bilangan dari urutan bilangan, bersifat **opsional** dengan default 1.

Contoh:

```
for i in range(1,11,2):
    print(i)
```

Ketika code dijalankan

```
Output:
1
3
5
7
9
```

b) While

While adalah perulangan yang akan mengeksekusi serangkaian code selama suatu kondisi bernilai *True*. Digunakan ketika tidak tahu pasti berapa kali perulangan harus terjadi.

while kondisi:

#blok kode yang akan diulang

Contoh:

```
counter = 1
while counter <= 5:

    print(counter)
    counter += 1
```

Ketika code dijalankan

```
Output:
1
2
3
4
5
```

c) Kontrol Perulangan

Kontrol perulangan digunakan untuk mengendalikan alur eksekusi di dalam perulangan. Dua perintah yang sering digunakan adalah *break* dan *continue*.

- *Break*

  Perintah *break* digunakan untuk menghentikan eksekusi dari perulangan secara paksa, meskipun kondisi perulangan belum selesai.

  Contoh:

```
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

for number in numbers:
    if number == 4:
        print("Angka 4 ditemukan!")
        print("Perulangan berhenti")
        break
    print(number)
```

  Ketika code dijalankan

```
Output:
1
2
3
Angka 4 ditemukan!
Perulangan berhenti
```

- *Continue*

  Perintah *continue* digunakan untuk melewati sisa kode di dalam loop dan langsung melanjutkan ke iterasi berikutnya.

  Contoh:

```
for number in range(1, 5):
    if number == 2:
        continue
    print(number)
```

  Ketika code dijalankan

```
Output:
1
3
4
```

d. Function

Function adalah blok kode yang dirancang untuk melakukan tugas tertentu, yang dapat digunakan kembali kapan saja di dalam program dengan memecah program menjadi bagian-bagian kecil yang lebih mudah dikelola dan digunakan kembali, sehingga membuat kode lebih bersih dan lebih mudah dibaca.

Python memiliki 2 jenis fungsi yaitu ***built-in function*** dan ***user-defined function***

1) *Built-in Function*

Built-in function adalah function yang sudah disediakan oleh Python dan bisa langsung digunakan tanpa perlu didefinisikan. Contohnya:

- **print()**: Untuk menampilkan output.
- **len()**: Untuk menghitung panjang sebuah objek seperti list atau string.

2) *User-defined Function*

*User-Defined Function*adalah yang didefinisikan oleh pengguna sesuai kebutuhan.

def namaFunction(parameter):

#blok kode yang akan dijalankan ketika fungsi dipanggil

Contoh:

```
def salam():
    print("Halo, selamat datang di Praktikum KB!")

salam()
```

Ketika code dijalankan

```
Output:
Halo, selamat datang di Praktikum KB!
```

3. Struktur Data pada Python

Struktur data adalah cara untuk menyimpan dan mengatur data di dalam program. Struktur data yang paling sering digunakan pada python adalah list, set, dan tuple. Masing-masing struktur data memiliki sifat dan kegunaan yang berbeda, tergantung pada kebutuhan program.

a. List

List adalah struktur data yang dapat menampung banyak nilai (elemen), di mana setiap elemen bisa berupa tipe data apa saja (angka, string, atau bahkan list lainnya). List bersifat **terurut** dan **dapat diubah (mutable)**.

Contoh:

```
buah = ["apel", "pisang", "jeruk"]
angka = [1, 2, 3, 4, 5]
```

Operasi dasar pada list:

- Mengakses data pada list (`[index]`)

  Contoh:

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

  Ketika code dijalankan

  ```
  Output:
  apel
  ```

  Pada python terdapat fitur untuk mengakses index sebuah list dari urutan terakhir yaitu dengan menggunakan index negatif.

  Contoh:

  ```
  buah = ["apel", "pisang", "jeruk", "mangga"]
  print(buah[-1])
  ```

  Ketika code dijalankan

  ```
  Output:
  mangga
  ```

- Mengubah elemen pada list

  variabel[index] = nilaiBaru

  Contoh:

  ```
  buah = ["apel", "pisang", "jeruk"]
  buah[1] = "mangga"
  print(buah)
  ```

  Ketika code dijalankan

  ```
  Output:
  ['apel', 'mangga', 'jeruk']
  ```

- Menambahkan elemen pada list (`append` atau `insert`)

  variabel.append("nilaiBaru")

  Contoh:

  ```
  buah = ["apel", "pisang", "jeruk"]
  buah.append("anggur")
  ```

```
print(buah)
```

Ketika code dijalankan

```
Output:
['apel', 'pisang', 'jeruk', 'anggur']
```

variabel.insert(index, "nilaiBaru")

Contoh:

```
buah = ["apel", "pisang", "jeruk"]
buah.insert(1,"anggur")
print(buah)
```

Ketika code dijalankan

```
Output:
['apel', 'anggur', 'pisang', 'jeruk']
```

- Menghapus elemen pada list (`pop` atau `remove`)

variabel.pop()

Contoh:

```
buah = ["apel", "pisang", "jeruk"]
buah.pop()
print(buah)
```

Ketika code dijalankan

```
Output:
['apel', 'pisang']
```

variabel.remove("namaItem")

Contoh:

```
buah = ["apel", "pisang", "jeruk"]
buah.remove("apel")
print(buah)
```

Ketika code dijalankan

```
Output:
['pisang', 'jeruk']
```

b. Set

Set adalah kumpulan elemen **unik** dan **tidak terurut**. Set tidak mengizinkan elemen duplikat. Elemen-elemen di dalam set tidak memiliki urutan tertentu.

Contoh:

```
angka = {1, 2, 3, 4, 5}
buah = {"apel", "pisang", "jeruk"}
```

Operasi dasar pada set:

- Menambahkan elemen pada set (`add()`)

  variable.add("item")

  Contoh:

  ```
  buah = {"apel", "pisang"}
  buah.add("jeruk")
  print(buah)
  ```

  Ketika code dijalankan

  ```
  Output:
  {'apel', 'pisang', 'jeruk'}
  ```

- Menghapus elemen pada set (`remove()`)

  variabel.remove("item")

  Contoh:

  ```
  buah = {"apel", "pisang"}
  buah.remove("pisang")
  print(buah)
  ```

  Ketika kode dijalankan

  ```
  Output:
  {'apel'}
  ```

- Operasi himpunan

  1) Union

     Operasi **union** menggabungkan dua set dan menghasilkan set baru yang berisi semua elemen dari kedua set, tanpa elemen duplikat.

     Contoh:

     ```
     set1 = {1, 2, 3}
     set2 = {3, 4, 5}
     ```

```
# Menggunakan metode union
union_set = set1.union(set2)

# Menggunakan operator |
union_set = set1 | set2

print(union_set)
```

Ketika code dijalankan

```
Output:
{1, 2, 3, 4, 5}
```

2) Intersect

Operasi **intersection** menghasilkan set baru yang hanya berisi elemen yang ada di **kedua** set.

Contoh:

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}

# Menggunakan metode intersection
intersect_set = set1.intersection(set2)

# Menggunakan operator &
intersect_set = set1 & set2

print(intersect_set)
```

Ketika code dijalankan

```
Output:
{3}
```

3) Difference

Operasi **difference** menghasilkan set baru yang berisi elemen dari set pertama yang **tidak ada** di set kedua.

```
set1 = {1, 2, 3}
set2 = {3, 4, 5}

# Menggunakan metode difference
```

```
difference_set = set1.difference(set2)

# Menggunakan operator -
difference_set = set1 - set2

print(difference_set)
```

Ketika code dijalankan

```
Output:
{1, 2}
```

c. Tuple

Tuple mirip dengan list, tetapi bersifat **tetap (immutable),** artinya Anda tidak bisa mengubah elemen-elemen di dalamnya setelah tuple dibuat.

Contoh:

```
angka = (1, 2, 3)
buah = ("apel", "pisang", "jeruk")
```

Operasi dasar pada tuple:

- Mengakses elemen pada tuple (`[index]`)

variabel[index]

Contoh:

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

Ketika code dijalankan

```
Output:
Apel
```

4. Penggunaan Library pada Python

Library adalah kumpulan modul yang berisi fungsi atau kode yang dapat digunakan kembali untuk menyelesaikan tugas tertentu. Untuk menggunakan library di Python, harus mengimpor modul atau fungsi dari library tersebut ke dalam program menggunakan perintah `import`.

Python memiliki banyak **python standard library** dan **python external library** yang dapat diinstal.

a. Python standard library

Library ini sudah disertakan dalam instalasi Python dan tidak memerlukan instalasi tambahan.

Contoh:

```
import random

buah = ["apel", "pisang", "jeruk"]
buah_acak = random.choice(buah)
print(buah_acak)
```

b. Python external library

Library ini dikembangkan oleh komunitas dan harus diinstal terlebih dahulu sebelum melakukan import.

- Menginstal external library

  pip install namaLibrary

  Contoh:

  ```
  pip install numpy
  ```

- Menggunakan external library

  Contoh:

  ```
  import numpy as np

  matriks = np.array([[1, 2, 3], [4, 5, 6]])
  print(matriks)
  ```

  Perintah **import numpy as np** digunakan untuk mengimpor library **NumPy** dan memberikan alias **np** agar fungsinya dapat dipanggil dengan lebih singkat, sehingga **numpy.array()** dapat ditulis sebagai **np.array()**.

  Ketika code dijalankan

  ```
  Output:
  [[1 2 3]
   [4 5 6]]
  ```

TUGAS

Buatlah sebuah program sederhana dengan menggunakan bahasa pemrograman python yang mengimplemetasikan konsep:

1. Struktur kontrol
2. Struktur data
3. Library (minimal 2 library)

Kemudian, upload source code program tersebut ke GitHub dengan nama repositori sebagai berikut,

NIM-PraktikumKB-Pertemuan1

Selamat mengerjakan!