DigitalSkola

LEARNING PROGRESS REVIEW

Week 4

Entropy Team

OUR TEAM

Entropy Team

Adhang Muntaha Muhammad
https://www.linkedin.com/in/adhangmuntaha/

Aziz Fauzi
https://www.linkedin.com/in/aziz-fauzi-a6904711b/

Iwan Wahyu
https://www.linkedin.com/in/iwan-wahyu-setyawan-506809183

Marcellina Alvita F
https://www.linkedin.com/in/marcellina-alvita-faustina-63a284226

Ramadhan Luthfan
https://www.linkedin.com/in/luthfan-mahathir-91369b18b

DigitalSkola

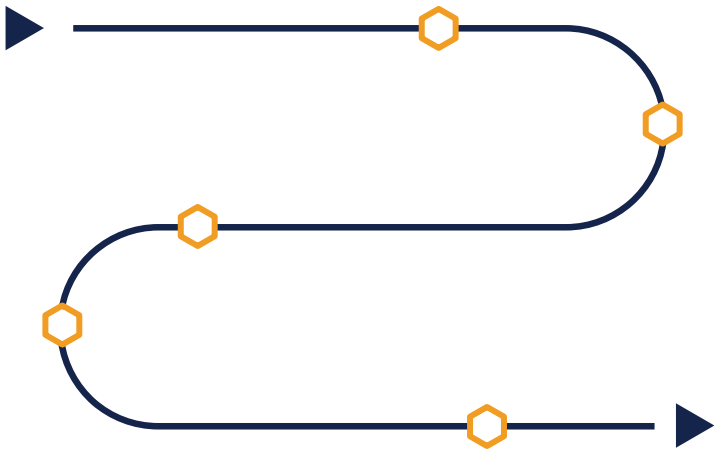
DAFTAR ISI

01

Basic Programming I: Conditions

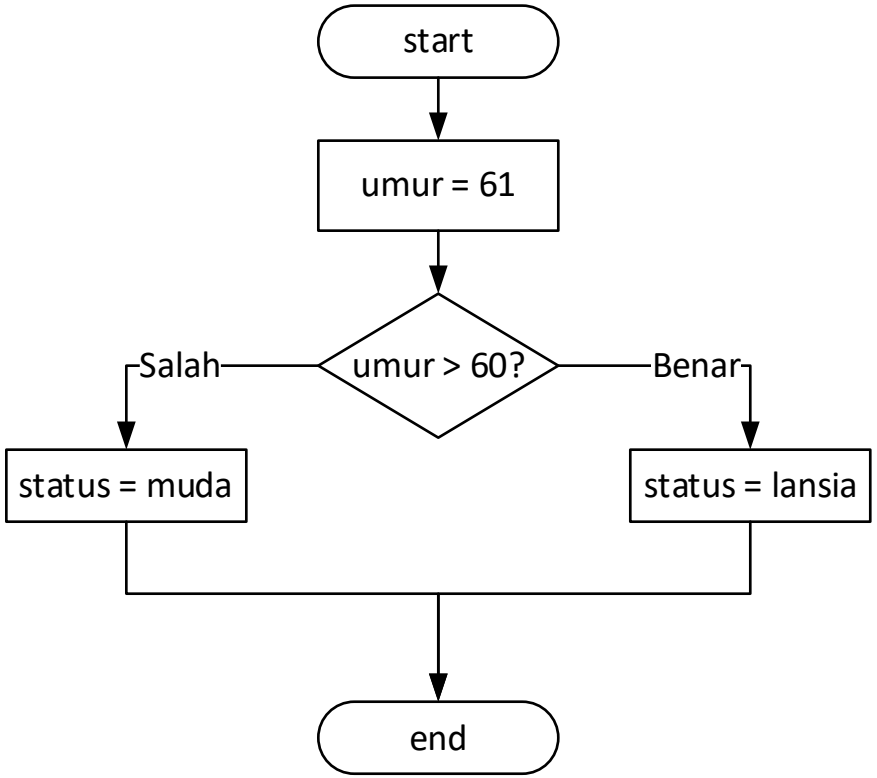
Pemrograman Dasar I:
Percabangan

Control Flow



- *Control flow* adalah **urutan** suatu program dijalankan
- *Control flow* dalam Python diatur oleh adanya **percabangan**, **perulangan**, dan pemanggilan **fungsi**

Percabangan



- Percabangan dapat digunakan untuk **menjalankan** beberapa kode ketika suatu **kondisi terpenuhi**
- Percabangan identik dengan pernyataan **“jika – maka”** atau “if – then”
- **Contoh**
  Jika seseorang memiliki umur lebih dari 60 tahun, maka statusnya adalah lansia. Jika tidak lebih dari 60 tahun, maka statusnya masih disebut muda

Operator Relasi

Operator Relasi			
Operator	**Deskripsi**	**Contoh**	**Hasil**
A == B	A sama dengan B	25 == 10	False
A != B	A tidak sama dengan B	25 != 10	True
A < B	A kurang dari B	25 < 10	False
A <= B	A kurang dari atau sama dengan B	25 <= 10	False
A > B	A lebih dari B	25 > 10	True
A >= B	A lebih dari atau sama dengan B	25 >= 10	True

Operator Logika

Operator Logika				
A	**B**	**A and B**	**A or B**	**not A**
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

0 = False
1 = True

- Operator **AND** akan menghasilkan TRUE jika **semua *operand*** bernilai **TRUE**
- Operator **OR** akan menghasilkan TRUE jika **sebagian atau semua *operand*** bernilai **TRUE**

IF Statement

SYNTAX

```
if expression:
   statements
```

CONTOH

```
if umur > 60:
  print('lansia')
```

- Pada Python, percabangan dilakukan menggunakan *if statement*
- ***Expression*** adalah representasi dari suatu nilai, dalam hal ini adalah kondisi
- Jika *expression* bernilai **True**, maka *statement* di bawahnya akan dijalankan
- Jika *expression* bernilai **False**, maka tidak ada *statement* yang dijalankan

IF – ELSE Statement

SYNTAX

```
if expression:
   statements
else:
   statements
```

CONTOH

```
if umur > 60:
  print('lansia')
else:
  print('muda')
```

- Jika *expression* bernilai **True**, maka *statement* di bawahnya akan dijalankan
- Jika *expression* bernilai **False**, maka *statement* dari ‘else’ yang dijalankan

IF – ELIF – ELSE Statement

SYNTAX

```
if expression:
   statements
elif expression:
   statements
else:
   statements
```

- ELIF merupakan singkatan dari ‘else – if’
- ELIF digunakan untuk melakukan **evaluasi** suatu **kondisi** jika kondisi sebelumnya tidak terpenuhi

Nested IF

- *Nested IF* yaitu percabangan dalam percabangan

SYNTAX

```
if expression:
   if expression:
      statements
   else:
      statements
else:
   if expression:
      statements
   else:
      statements
```

CONTOH

```
if gender == 'pria':
  if umur > 60:
    print('pria lansia')
  else:
    print('pria muda')
else:
  if umur > 60:
    print('wanita lansia')
  else:
    print('wanita muda')
```

Penghubung Kondisi

CONTOH

```
if (gender == 'pria') and (umur > 60):
   print('pria lansia')
elif (gender == 'wanita') and (umur > 60):
   print('wanita lansia')
else:
   print('pria atau wanita muda')
```

- Operator logika dapat digunakan untuk **menggabungkan** beberapa **kondisi** pada *if statement*

Match – Case

SYNTAX

```
match subject:
    case pattern_1:
        statements
    case pattern_2:
        statements
    . . .
    case _:
        statements
```

- Match – case merupakan **fitur baru** pada Python versi 3.10
- Match – case merupakan ***pattern matching***, yaitu membandingkan 'subject' dengan setiap 'pattern' pada *case statement*
- Jika suatu *pattern* terpenuhi, maka *statement* di bawahnya akan dijalankan
- Jika semua *pattern* tidak terpenuhi, maka *statement* pada '_' akan dijalankan

DigitalSkola

Penggunaan Percabangan

SYNTAX

```
if expression:
   statements
elif expression:
   statements
else:
   statements
```

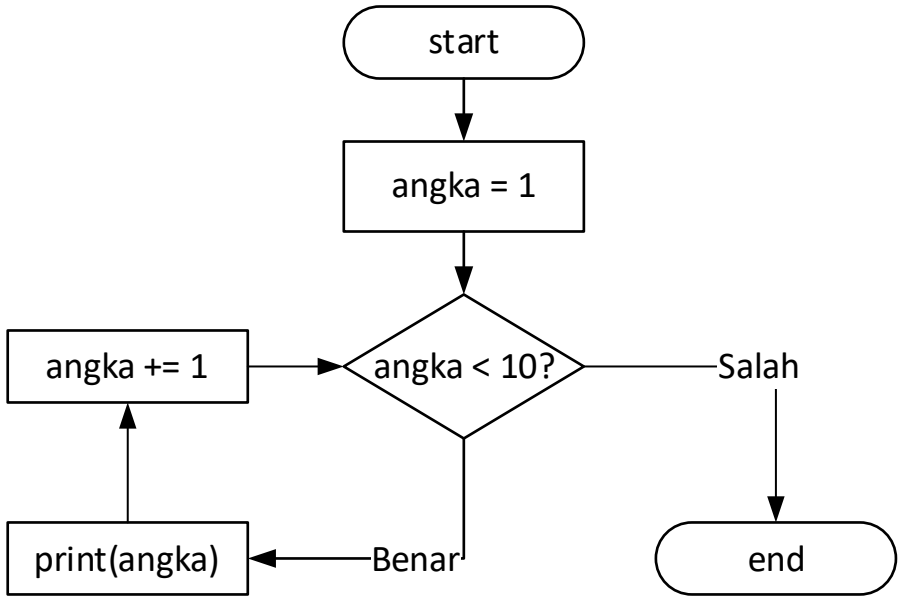
- Percabangan dengan *if statement*:
  - IF **wajib** digunakan
  - ELIF **tidak wajib** digunakan
  - ELSE **tidak wajib** digunakan

02

Basic Programming II: Iterations

Pemrograman Dasar II: Perulangan

Perulangan



- Perulangan digunakan untuk **menjalankan** beberapa *statement* **secara berulang**
- Perulangan akan terus terjadi **sampai** suatu **kondisi** akhir (*stopping condition*) **terpenuhi**
- **Contoh**
  Mencetak angka 1 – 9 dengan interval 1

DigitalSkola

FOR Loop

SYNTAX

```
for var in iterable:
   statements
```

CONTOH

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

for x in range(1,100):
  print(x)
```

- FOR digunakan untuk melakukan perulangan di mana **jumlah perulangan pasti**
- Jumlah perulangan ditentukan oleh jumlah **elemen** pada ***iterable***
- *Iterable* adalah *object* yang dapat digunakan seperti *sequence*, yaitu dapat dilihat elemennya satu per satu
- Contoh *iterable*: string, list, tuple,set, dictionary, range

DigitalSkola

WHILE Loop

SYNTAX

```
while (expression):
   statements
```

CONTOH

```
angka = 1
while (angka < 100):
  print(angka)
  angka += 2*angka
```

- WHILE digunakan untuk melakukan perulangan di mana **jumlah perulangan tidak pasti**
- Jumlah perulangan **tidak** disebutkan secara **eksplisit**
- Perulangan terus dijalankan selama *expression* bernilai True

Nested Loop

- *Nested loop* yaitu perulangan dalam perulangan

SYNTAX

```
# for dalam for
for var_1 in iterable:
   statements
   for var_2 in iterable:
      statements

# for dalam while
while (expression):
   statements
   for var in iterable:
      statements
```

CONTOH

```
# for dalam for
for i in range(0, 5):
  for x in range(0, i):
    print('*', end='')
  print()
```

DigitalSkola

Penggunaan Perulangan

CONTOH

```
# bentuk for
for i in range(0, 5):
  print(i)

# bentuk while
n = 0
while (n < 5):
  print(n)
  n += 1
```

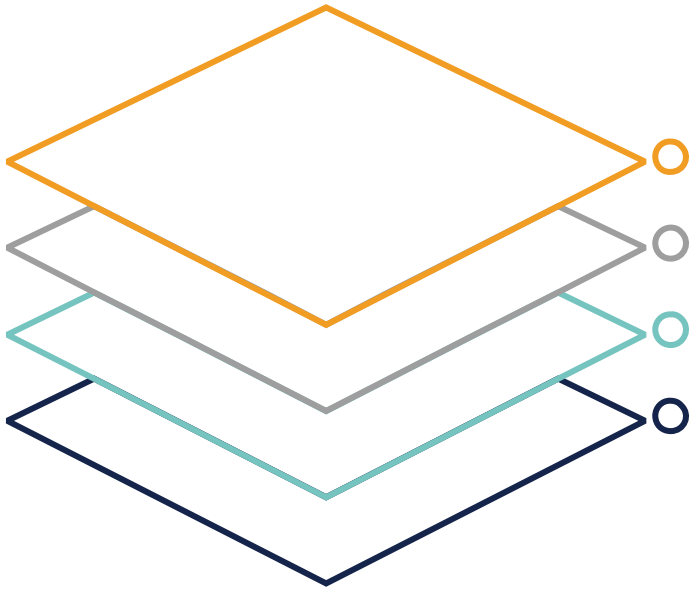
- Pada beberapa kasus, perulangan menggunakan FOR dapat diubah menjadi bentuk WHILE, begitu pun sebaliknya
- Pastikan untuk membuat *stopping condition*, jika tidak maka perulangan akan berlanjut tanpa akhir

03

Basic Programming III: Array & Other Data Type

Pemrograman Dasar III:
Tipe Data Array dan Lainnya

Array



- *Array* adalah salah satu tipe data yang dapat **berisikan beberapa elemen**
- **Nomor indeks** digunakan untuk **mengakses elemen** dalam array
- **List** merupakan ***built-in*** *array* pada Python

DigitalSkola

List

CONTOH

```
# list kosong
my_list = []
my_list = list()

# list ber-value
my_list = [1, 3, 5]
my_list = list((1, 3, 5))

# akses list
my_list[1] # 3
```

- Digunakan untuk menyimpan beberapa data, di mana data tersebut **dapat diubah** (*mutable*)
- **Pembuatan list** dapat menggunakan simbol **kurung siku** atau fungsi **list()**
- **Akses elemen** menggunakan **nomor indeks**
- Nomor indeks dimulai **dari 0** (nol)

Method pada List

Method	Deskripsi
append()	Menambahkan elemen di akhir list
clear()	Menghapus semua elemen dari list
copy()	Mendapatkan salinan dari list
count()	Menghitung jumlah elemen dengan nilai tertentu
extend()	Menambahkan elemen list (atau *iterable* lain) ke list yang lain
index()	Mencari indeks (pertama) dari suatu nilai
insert()	Menambah elemen pada posisi tertentu
pop()	Menghapus elemen pada posisi tertentu
remove()	Menghapus elemen dengan nilai tertentu
reverse()	Membalik urutan dari list
sort()	Mengurutkan list

Sumber: https://www.w3schools.com/python/python_lists_methods.asp

Tuple

CONTOH

```
# tuple kosong
my_tuple = ()
my_tuple = tuple()

# tuple ber-value
my_tuple = (1, 3, 5)
my_tuple = tuple([1, 3, 5])

# akses tuple
my_tuple[1] # 3
```

- Digunakan untuk menyimpan beberapa data, di mana data tersebut **tidak dapat diubah** (*immutable*)
- **Pembuatan tuple** dapat menggunakan simbol **kurung biasa** atau fungsi **tuple()**
- **Akses elemen** menggunakan **nomor indeks**
- Nomor indeks dimulai **dari 0** (nol)

Method pada Tuple

Method	Deskripsi
count()	Menghitung jumlah elemen dengan nilai tertentu
index()	Mencari indeks (pertama) dari suatu nilai

Sumber: https://www.w3schools.com/python/python_tuples_methods.asp

Set

CONTOH

```
# set kosong
my_set = set()

# set ber-value
my_set = {1, 3, 5}
my_set = set([1, 3, 5])

# akses set
my_set = {1, 3, 3, 5}
for i in my_set:
  print(i) # 1, 3, 5
```

- Digunakan untuk menyimpan beberapa data, di mana data tersebut **tidak bernilai sama** (*distinct value*)
- **Pembuatan set** dapat menggunakan simbol **kurung kurawal** atau fungsi **set()**
- **Tidak dapat diakses** menggunakan **nomor indeks**, tapi dapat menggunakan perulangan **FOR** sebagai ***iterable***

Method pada Set

Method	Deskripsi
add()	Menambahkan elemen pada set
clear()	Menghapus semua elemen dari set
copy()	Mendapatkan salinan dari set
difference()	Mendapatkan sebuah set yang berisi perbedaan antar-set
discard()	Menghapus elemen dengan nilai tertentu
intersection()	Mendapatkan sebuah set yang berisi irisan antar-set
isdisjoint()	Menentukan apakah terdapat irisan antar-set
issubset()	Menentukan apakah set lain mengandung set ini atau tidak
issuperset()	Menentukan apakah set ini mengandung set lain atau tidak
pop()	Menghapus sebuah elemen dari set
union()	Menggabungkan set

Sumber: https://www.w3schools.com/python/python_sets_methods.asp

Dictionary

CONTOH

```
# dictionary kosong
my_dict = {}
my_dict = dict()

# dict ber-value
my_dict = {'nama':'entropy',
           'kelas':'DS-11'}
my_dict = dict(nama='entropy',
               kelas='DS-11')

# akses dictionary
my_dict['nama'] # entropy
```

- Digunakan untuk memetakan nilai (*value*) dengan kuncinya (*key*) menjadi **key:value** di mana **key** harus ***unique***
- **Pembuatan dictionary** dapat menggunakan simbol **kurung kurawal** atau fungsi **dict()**
- **Akses elemen** menggunakan **key** yang telah dibuat

DigitalSkola

Method pada Dictionary

Method	Deskripsi
clear()	Menghapus semua elemen dari dictionary
copy()	Mendapatkan salinan dari dictionary
fromkeys()	Mendapatkan sebuah dictionary dengan *key* dan *value* tertentu
get()	Mendapatkan *value* dari *key* tertentu
items()	Mendapatkan sebuah list berisi tuple dari setiap pasang *key-value*
keys()	Mendapatkan sebuah list berisi seluruh *key*
pop()	Menghapus elemen dengan *key* tertentu
popitem()	Menghapus pasangan *key-value* yang terakhir dimasukkan
setdefault()	Mendapatkan *value* dari *key* tertentu. Jika *key* tidak ada, *key* dan *value* (pada parameter) dimasukkan ke dictionary
update()	Memperbarui dictionary dengan pasangan key-value tertentu
values()	Mendapatkan sebuah list berisi seluruh *value* dari dictionary

Sumber: https://www.w3schools.com/python/python_dictionaries_methods.asp

THANKS

Entropy Team