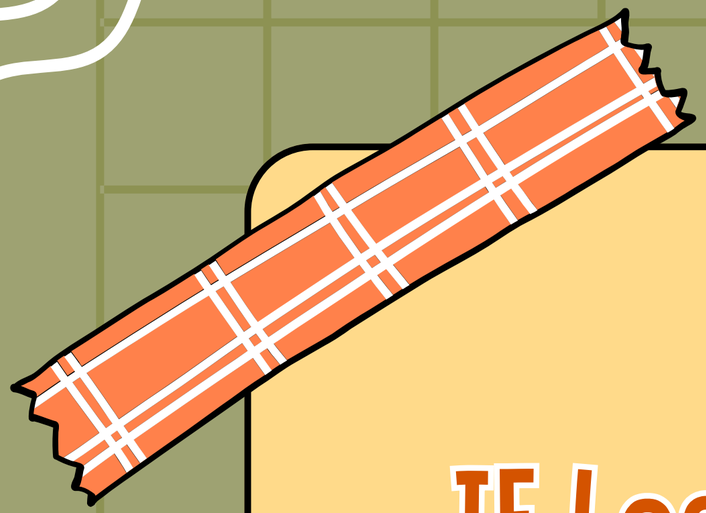
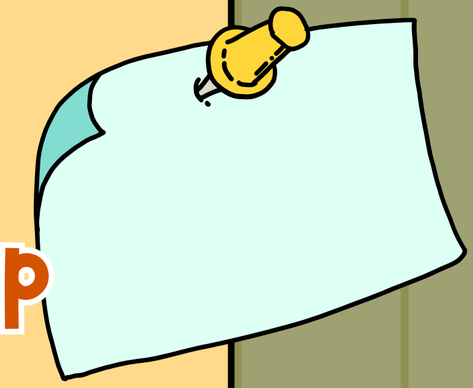
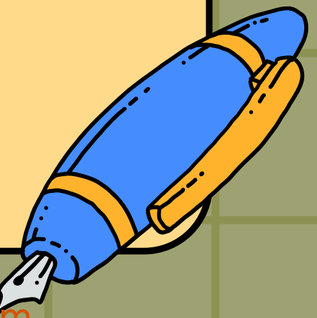


Python:
IF Logic, Function, and Loop

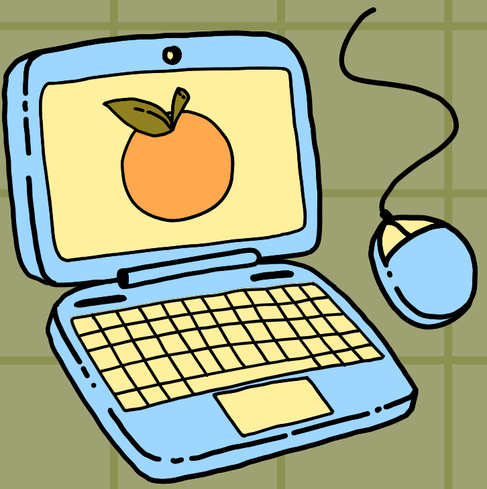


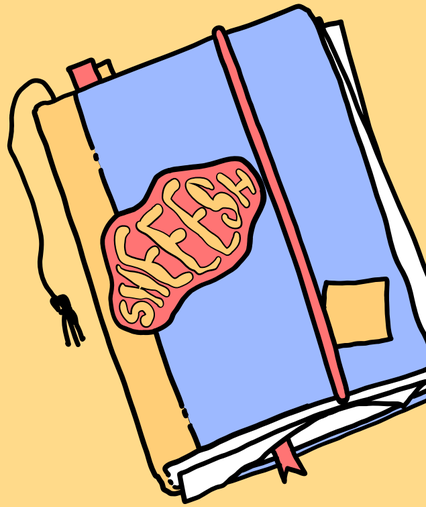
Anita Mila Oktafani



DQLab Live Class Data Analyst with SQL & Python in Google Platform

Table of Content

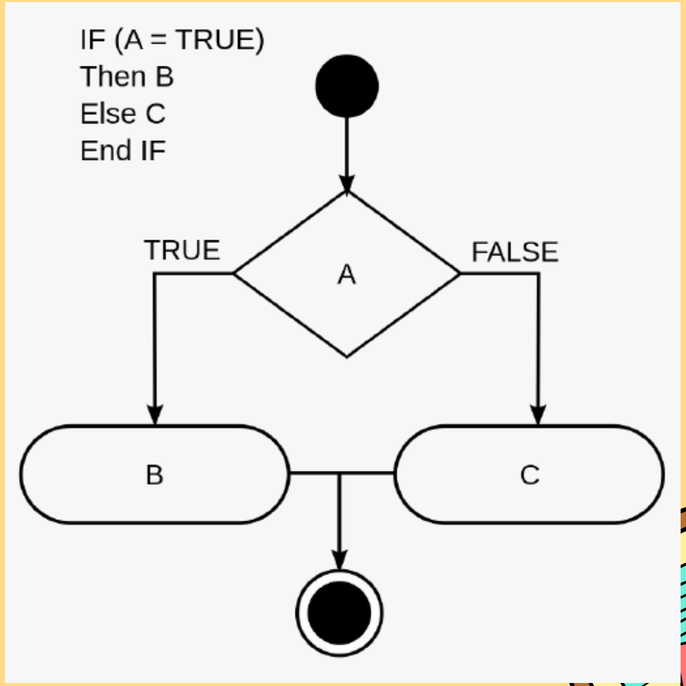




IF LOGIC

IF LOGIC

IF is the logic that used in the branching so the computer can select blocks of code which one to run based on certain conditions.



IF Condition

The computer will check the condition in front of the IF:

IF <condition> ;
<code block>

```
lapar = True

if lapar == True:
  print("Makan")
```

```
Makan
```

- If the condition evaluates to TRUE, the code block contained in IF will be executed

- If the condition evaluates to FALSE, nothing is done

```
lapar = False

if lapar == True:
  print("Makan")
```

IF ELSE Condition

```
IF <condition> :
        <code block>
ELSE :
        <code block>
```

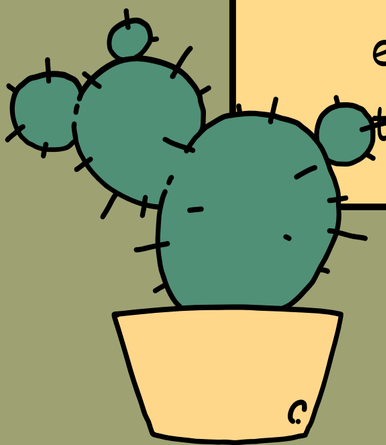
The computer will check the condition in front of the if

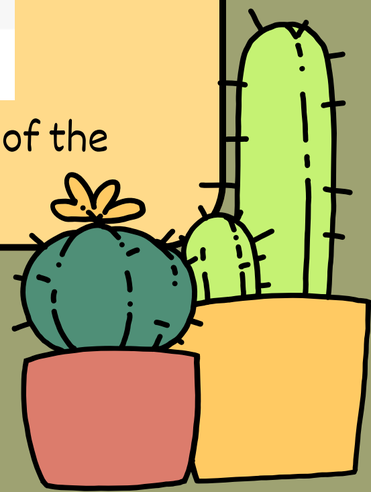
- If the condition evaluates to TRUE, the code block exists inside if will be executed
- The code block inside ELSE will be executed if absolutely no conditions are true

```
lapar = False

if lapar == True:
  print("Makan")
else:
  print("Terus Belajar")
```

```
Terus Belajar
```

"Terus Belajar" is displayed because NONE of the conditions above it are true





IF ELIF Condition

Use ELIF if we want to try several possible conditions

- The first condition is TRUE, the code block will be executed then the computer will exit the if block
- We can add some ELIFs

```
IF <condition> :
        <code block>
ELIF <condition> :
        . . .
ELSE :
        <code block>
```

```
kondisi = 'capek'

if kondisi == 'lapar':
  print("Makan")
elif kondisi == 'capek':
  print("Istirahat")
elif kondisi == 'ngantuk':
  print("Tidur")
else:
  print("Terus Belajar")
```

```
Istirahat
```

FUNCTION

Function

- FUNCTIONs are blocks of code that have specific tasks
- The function is simply defined once
- Functions can be used repeatedly

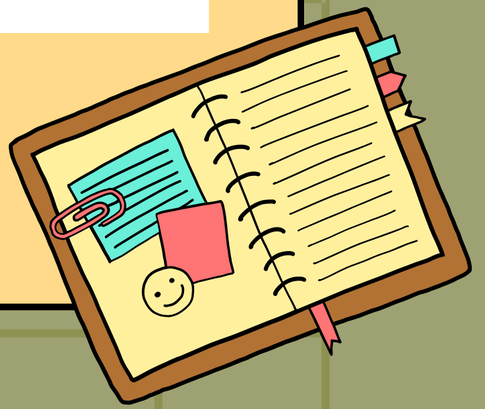
Functions can have no parameters or returns, for example:

def <function_name> () :
<code block>

```
def tulis_hello():
  print('Hello')
```

```
tulis_hello()

Hello
```

To run the function above, simply execute tulis_hello()



def <function_name> (parameter) :
<code block>

return <code/variable>

The function is used to calculate the area of a square by returning a value in the form of multiplying the length by the width

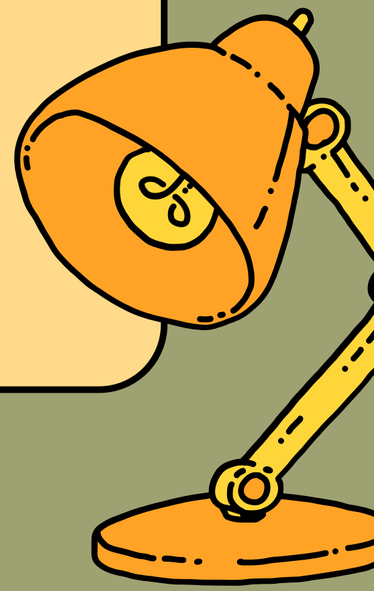
```
def luas_persegi(panjang, lebar):
    return panjang*lebar
```

```
L = luas_persegi(10,50)
print(L)

500
```

If a function does not have a return then after it is executed, the function has NO VALUE or is NONE

```
def luas_persegi(panjang, lebar):
    print(panjang*lebar)
```

```
L = luas_persegi(3,4)
print(L)

12
None
```



LOOP

For Loop

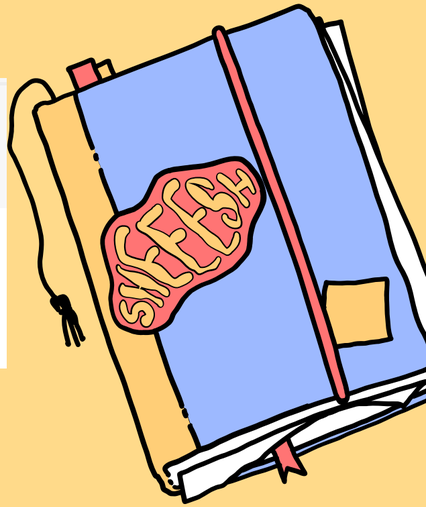
FOR LOOP is used to iterate over sequences (list, tuple, dictionary, set, string)

The code will be executed repeatedly until the last element in the sequence

for <variable> in <sequence> :
<code block>

```
for buah in  ['apel','mangga','anggur']:
  print(buah)
```

```
apel
mangga
anggur
```



Range()

if we want to make a certain number of iterations, then the sequence can be built using the range function

- range(n) will produce integers from 0 to n-1

```
for i in range(5):
  print(i)
```

```
0
1
2
3
4
```

- range(a,b) will produce integers from a to b-1

```
for i in range(1,5):
  print(i)
```

```
1
2
3
4
```

- range(a,b,s) will produce integers from a to b-1 by jumping s

```
for i in range(1,10,2):
  print(i)
```

```
1
3
5
7
9
```

While Loop

WHILE LOOP is an iteration method where a block of code will continue to execute as long as the condition evaluates to true. WHILE CODE will continue to run as long as the condition is met.

while <condition> :
<code block>

```
perintah = 'jalan'

while perintah != 'berhenti':
  print('jalan terus')
  perintah = input()
```

```
jalan terus
lanjut
```

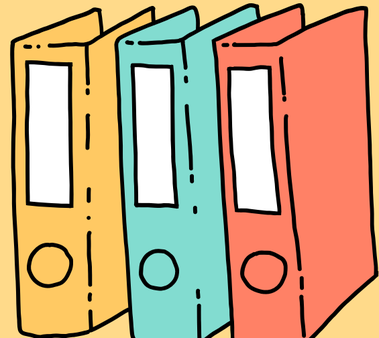
```
lanjut
jalan terus
jalan
jalan terus
berhenti
```

code that displays 'jalan terus' as long as there is no 'berhenti' command

The x value will be displayed and reduced by 1 as long as x > 0. Therefore, make sure the value of x always changes at each iteration to avoid an infinite loop

```
x = 100
while x > 0:
  print(x)
  x = x - 1
```

```
100
99
98
97
96
95
94
```



Break and Continue

BREAK is used to exit the loop even if the iteration is not finished

```
for i in range(10):
  print(i)
  if i == 6:
    break
```

```
0
1
2
3
4
5
6
```

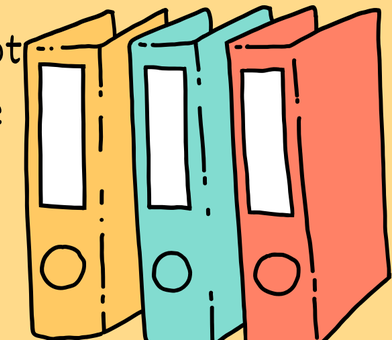
example: the loop will stop when i is 6

CONTINUE is used to stop the current iteration and continue to the next iteration

```
for i in range(6):
  if i == 4:
    continue
  print(i)
```

```
0
1
2
3
5
```

example: when i is 4, the value of i is not written but immediately jumps to the next iteration



Thank You

LinkedIn:

https://www.linkedin.com/in/anitamilaoktafani/