DigitalSkola

Learning Progress Review

OMICRON

Data Science Bootcamp Batch 11

OMICRON

Data Science Bootcamp Batch 11

Muhammad Fikri Fadila

https://www.linkedin.com/in/muhammad-fikri-fadila-a551161a6/

Fajar Achmad

https://www.linkedin.com/in/fajar-achmad-755945111/

Anugrah Yazid

https://www.linkedin.com/in/anugrah-yazid-7253bb221/

Edo M Hadad Gibran

https://www.linkedin.com/in/edo-gibran-38505a142/

DigitalSkola

Week 4

Basic Programming

DigitalSkola

01 ---------- Conditions & Control Flow

02 ---------- IF Statements

03 ---------- Match Case

04 ---------- Iteration

05 ---------- Array and Other Data Types

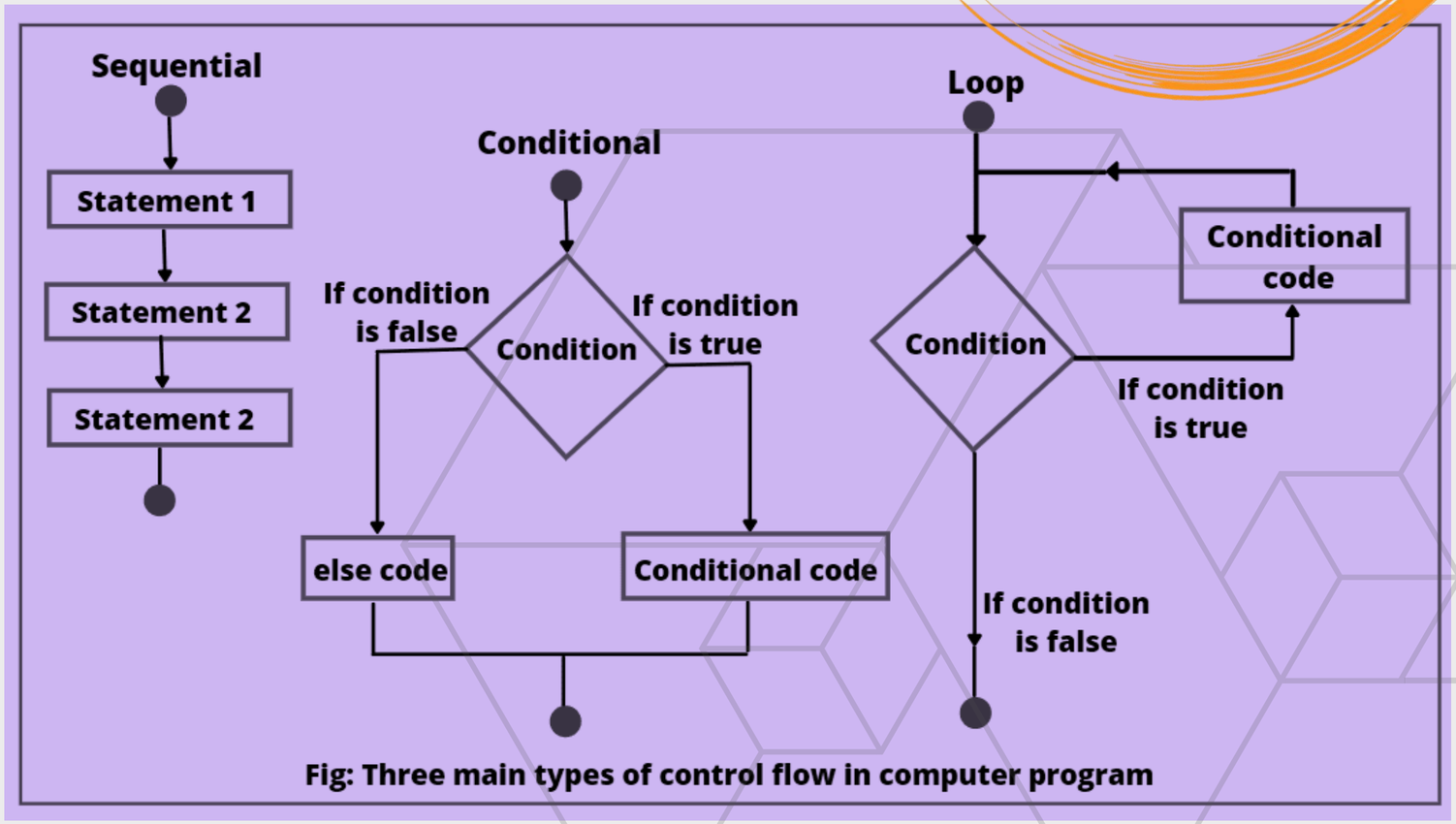
DigitalSkola

01 Conditions & Control Flow

Conditions & Control Flow

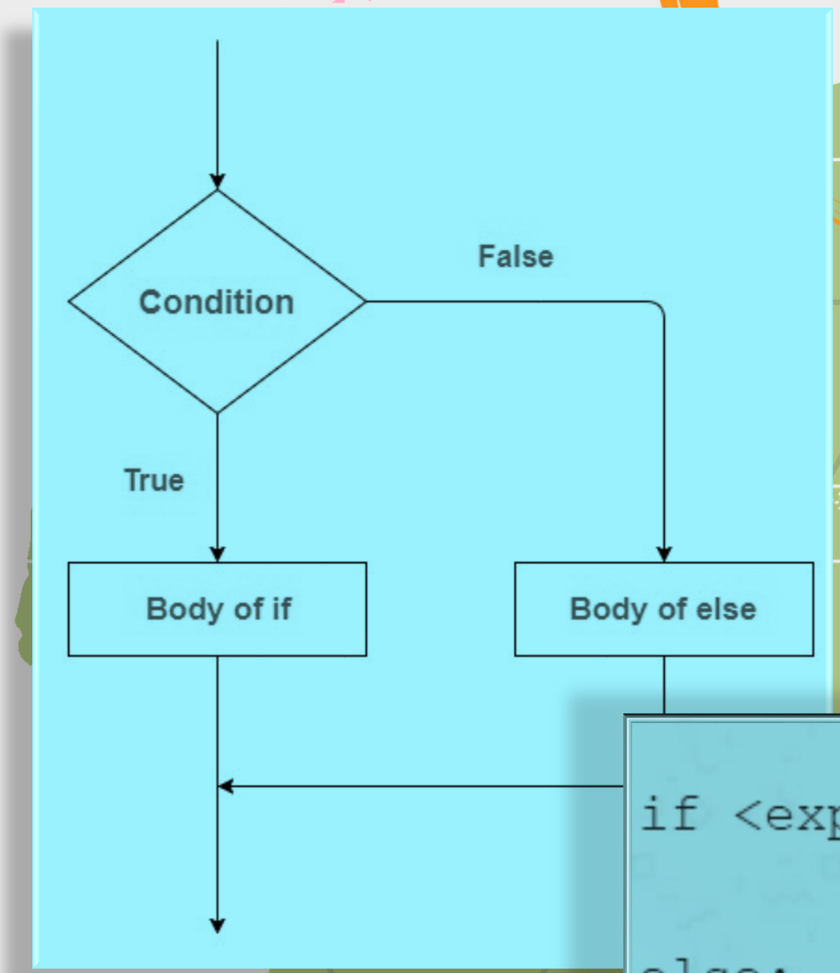
DigitalSkola

- Control Flow adalah perintah yang dapat mengeksekusi kode kode pemrograman.
- Control Flow pada program Python diatur dengan pernyataan conditional, loop dan function.
- Hanya menjalankan kode tertentu ketika syarat terpenuhi.



Conditions & Control Flow

- Dalam Python, pernyataan if adalah bagaimana anda melakukan pengambilan keputusan semacam ini.
- Ini berisi kode yang berjalan hanya ketika kondisi yang diberikan dalam pernyataan if adalah benar/memenuhi syarat.
- Jika kondisinya salah, maka pernyataan opsional else berjalan, yang berisi beberapa kode untuk kondisi lain.

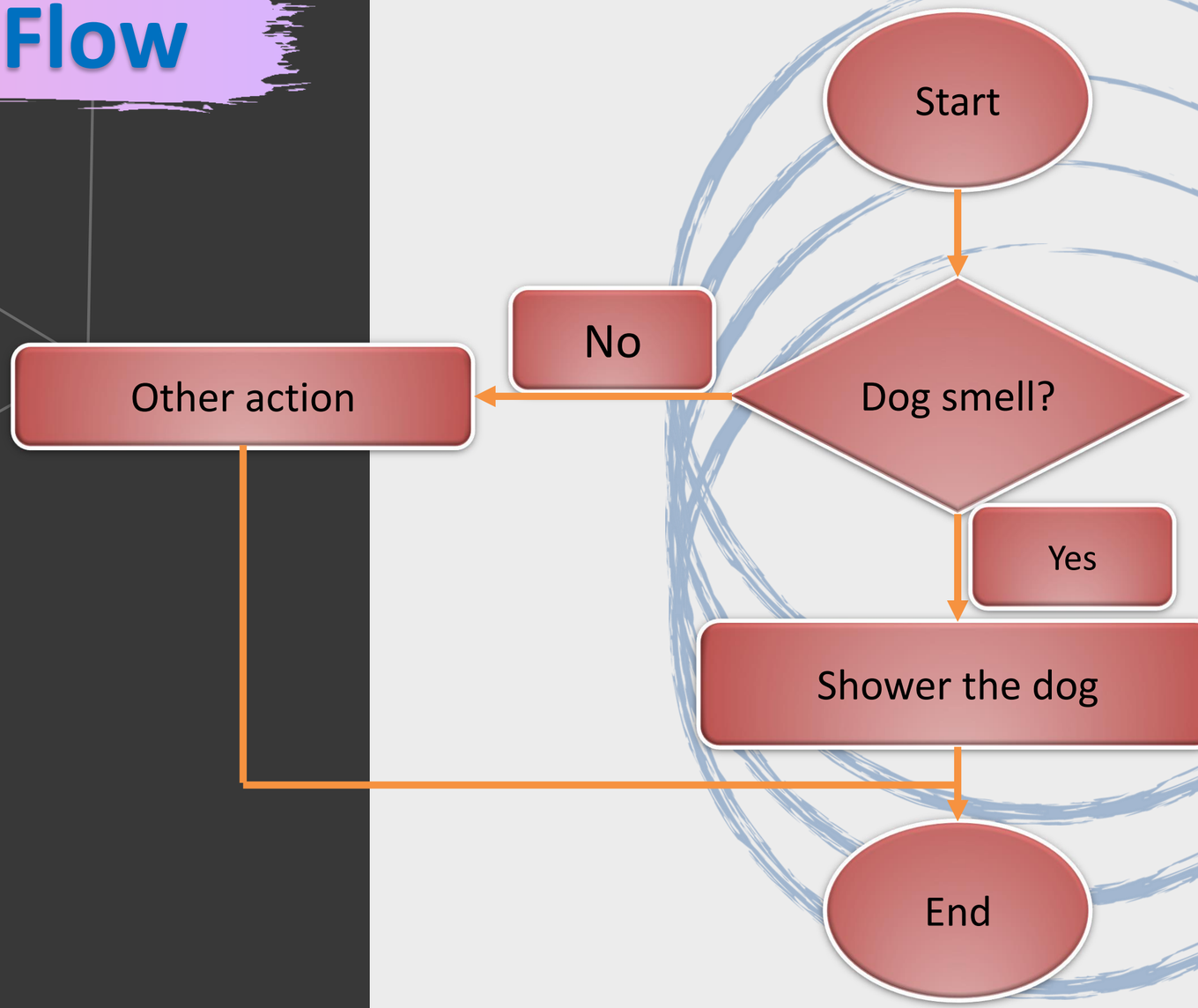



```
if <expression>:
        <statement>
else:
        <statement>
```

Conditions & Control Flow

if dog is smell then
    shower the dog
else
    do other things

```
if dog == 'smell':
    shower_the_dog()
else:
    other_action()
```



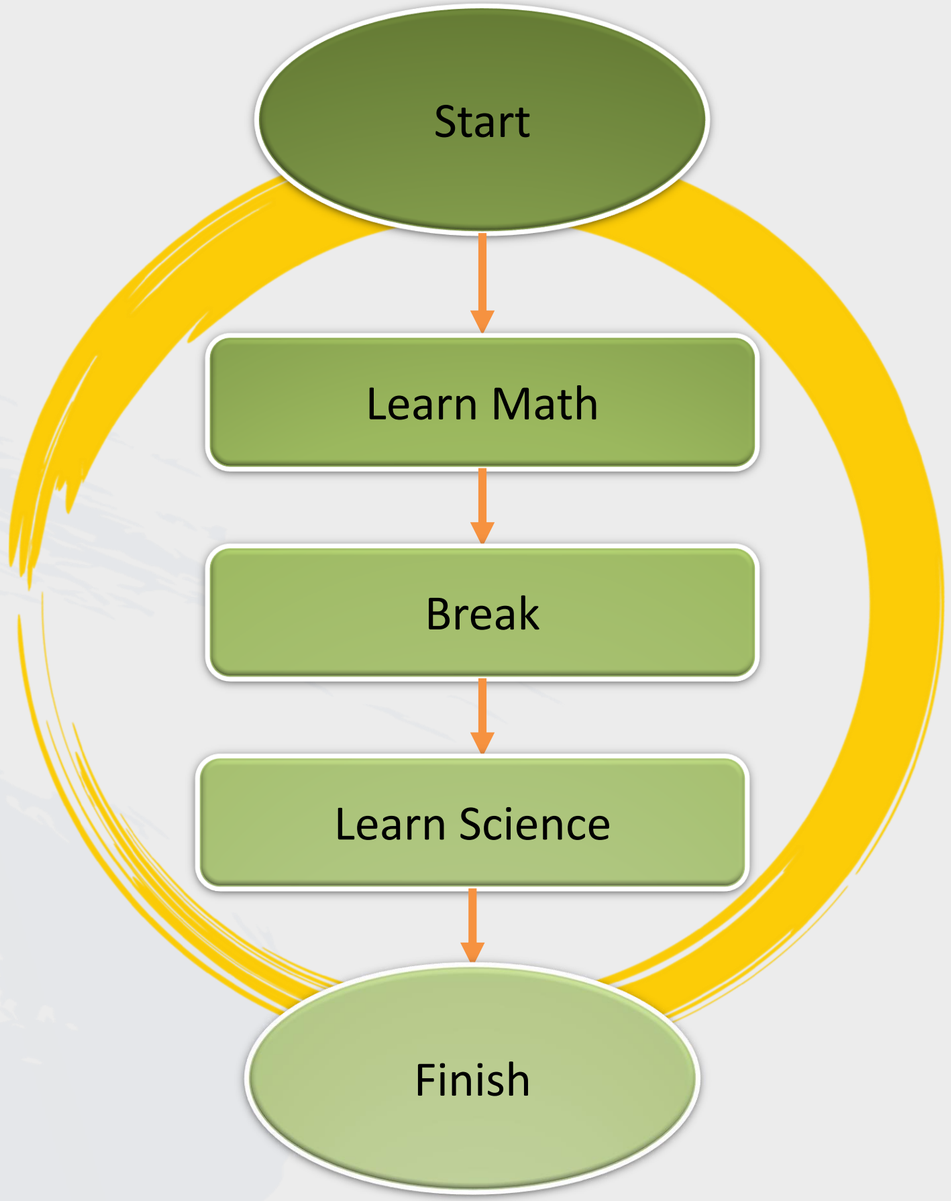
DigitalSkola

Conditions & Control Flow

Sequential:

- **Learn Math**
- **Break Time**
- **Learn Science**

```
Other Example:
a=30
b=15
c=a-b
print("a-b =",c)
```



DigitalSkola

Logical Conditions

Equals	:	a == b
Not Equals	:	a != b
Less Than	:	a < b
Less Than Or Equal To	:	a <= b
Greater Than	:	a > b
Greater Than Or Equal To	:	a >= b

Logical Operators

and	True if both the operands are true
or	True if either of the operands is true
not	True if operand is false (complements the operand)

DigitalSkold

DigitalSkola

02 IF Statements

“IF” Statement

DigitalSkola

Conditions

- **Single Condition** (`if`)

Full Cast

```
a = 7
if a == 7:
  print("a is seven")
```

```
a is seven
```

Shorthand Version

```
a = 7
if a == 7: print("a is seven")
```

```
a is seven
```

- **Multiple Conditions** (`if -> elif-> else`)

```
a = 13
b = 7
if a > b:
  print("a is bigger than b")
elif a < b:
  print("a is smaller than b")
else:
  print('a and b is the same')
```

```
a is bigger than b
```

- **Two Conditions** (`if -> else`)

Full Cast

```
a = 13
b = 13
if a != b:
  print("a and b is not the same")
else:
  print("a and b is the same")
```

```
a and b is the same
```

Shorthand Version

```
[25] a = 13
     b = 13
     print("a and b is not the same") if a == b else print("a and b is the same")
```

```
a and b is not the same
```

“IF” Statement

Expressions

- Single Expression

```
a = 13
b = 9
if a > b:
  print("a is bigger than b")
```

```
a is bigger than b
```

- Multiple Expressions

“and” Expression :

```
[19] a = 12
     b = 8
     c = 21
     if a > b and b < c:
       print("a is bigger than b")
```

```
a is bigger than b
```

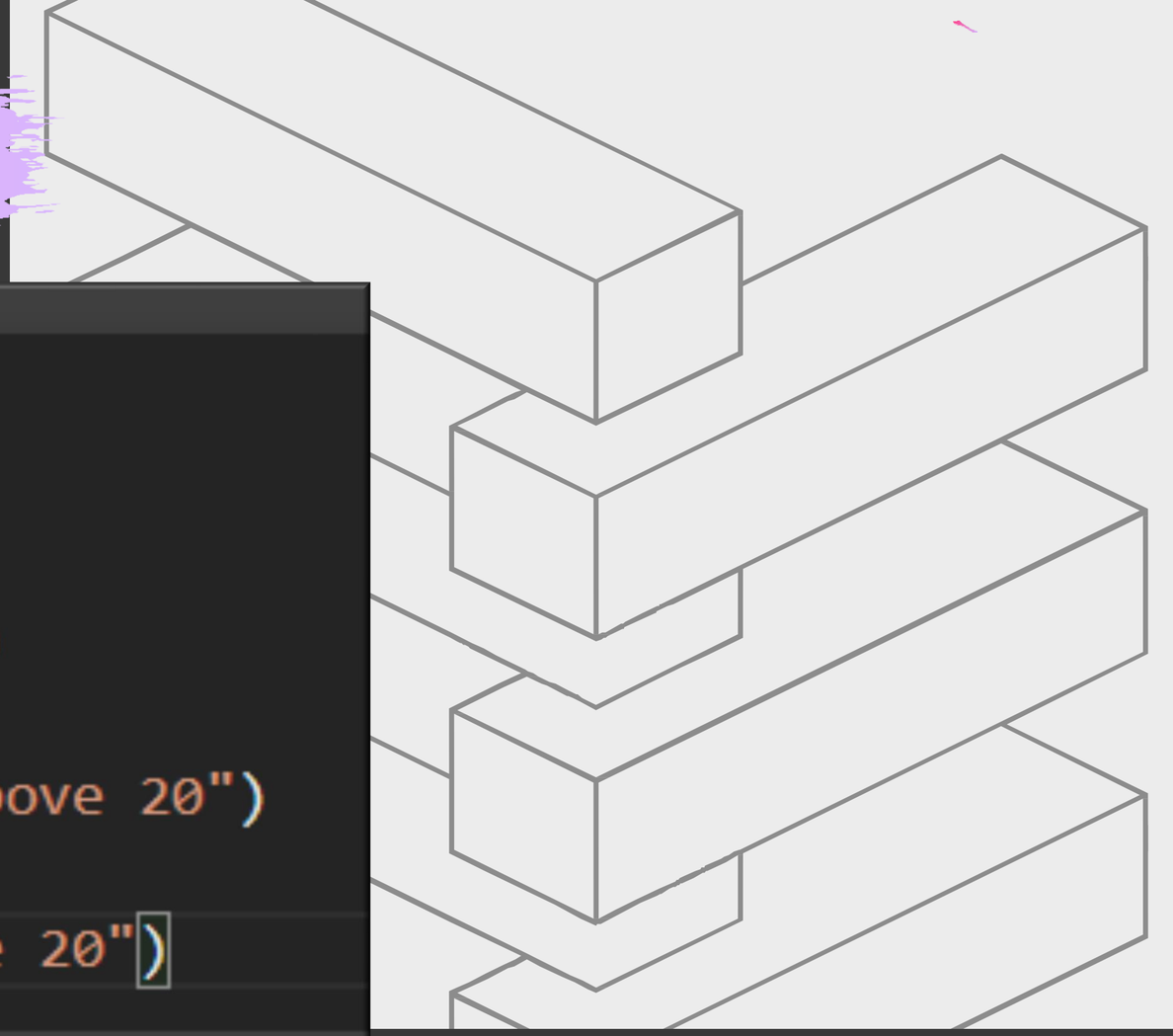
“or” Expression :

```
[21] a = 5
     b = 11
     c = 25
     if c > b or c < a:
       print("at least one of the expresion is True")
```

```
at least one of the expresion is True
```

DigitalSkola

“IF” Statement [nested]



```
x = 30

if x > 10:
  print("Above ten, ")
  if x > 20:
    print("and also above 20")
else:
  print("but not above 20")
```

```
Above ten,
and also above 20
```

DigitalSkola

Implementasi IF Statement

DigitalSkola

Cek Angka (Positif, Negatif, atau nol)

```
# Check Number (Positif, Negative, or Zero)

number = float(input("Enter number : "))

if number > 0:
  print(number, "is a positif number.")
elif number == 0:
  print(number, "is a zero number.")
else:
  print(number, "is a negative number.")
```

OUTPUT:

```
Enter number : 8
8.0 is a positif number.
```

```
Enter number : 0
0.0 is a zero number.
```

```
Enter number : -4
-4.0 is a negative number.
```

Cek Bilangan Ganjil atau Genap

```
# Check Odd or Even Number

number = int(input("Enter number : "))

if number % 2 != 0:
  print(number, "is an odd number.")
else:
  print(number, "is an even number.")
```

OUTPUT:

```
Enter number : 3
3 is an odd number.
```

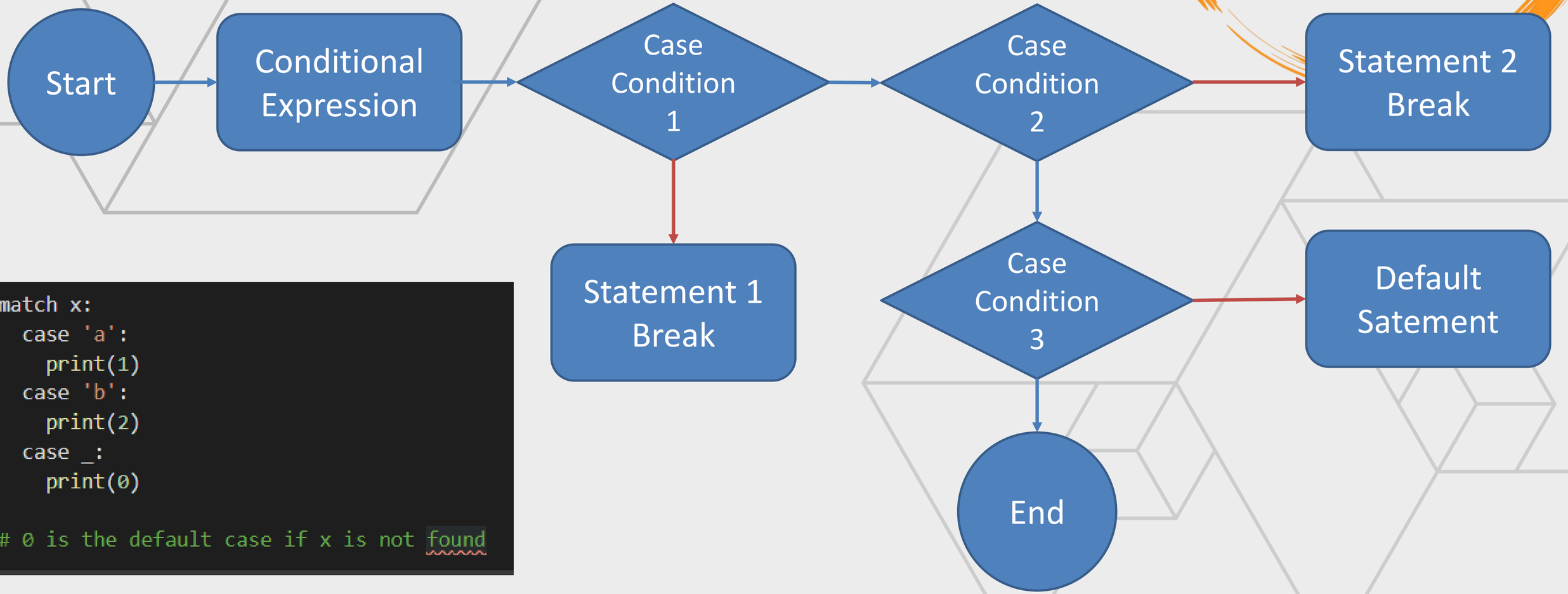
```
Enter number : 4
4 is an even number.
```

DigitalSkola

03 Match Case

Match Case

DigitalSkola




```
match x:
  case 'a':
    print(1)
  case 'b':
    print(2)
  case _:
    print(0)

# 0 is the default case if x is not found
```

DigitalSkola

04 Iteration

Iteration

Iteration atau **Loop** pada *programming* adalah instruksi yang dilakukan secara berulang. **Iteration / Loop** akan terus dijalankan sampai kondisi tertentu, yang telah diatur sebelumnya, tercapai.

- Iteration Types
  - Iterasi sampai kondisi tertentu tercapai
  - Iterasi sampai sejumlah perulangan yang ditentukan
  - Iterasi dalam elemen pada list ataupun *array*

Jenis Loop

While Loop

Digunakan untuk menjalankan suatu *block of statements* secara berulang hingga memenuhi kondisi tertentu, dan ketika kondisinya telah terpenuhi, maka iterasi baris berikutnya akan dihentikan.

Termasuk dalam kategori *indefinite iteration*: Jumlah perulangan loop tidak didefinisikan secara eksplisit di awal.

Syntax while loop:

```
while <expression>:
    <statement>
```

Implementasi while Loop

N – Urutan Awal Bilangan **Fibonacci** Menggunakan Loop **WHILE**

```
# Program to display the Fibonacci sequence up to n-th term in list

nterms = int(input("How many terms? "))
fibonacci_sequence = []

# first two terms
n1, n2 = 0, 1
count = 0

# check if the number of terms is valid
if nterms <= 0:
  print("Please enter a positive integer")
# if there is only one term, return n1
elif nterms == 1:
  print("Fibonacci sequence upto",nterms,":")
  print(n1)
# generate fibonacci sequence using WHILE
else:
  print("Fibonacci sequence:")
  while count < nterms:
    fibonacci_sequence.append(n1)
    nth = n1 + n2
    # update values
    n1 = n2
    n2 = nth
    count += 1

print(fibonacci_sequence)
```

OUTPUT:

```
How many terms? 10
Fibonacci sequence:
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

10 urutan awal bilangan Fibonacci dengan pengulangan WHILE

Implementasi while Loop

N – Urutan Awal Bilangan **Fibonacci Ganjil** dan **Genap** Menggunakan Loop **WHILE**

```
# Program to display the Fibonacci sequence up to n-th term
# Divided by ODD & EVEN number list

nterms = int(input("How many terms? "))
odd_list = []
even_list = []

# first two terms
n1, n2 = 0, 1
count = 0

# check if the number of terms is valid
if nterms <= 0:
  print("Please enter a positive integer")
# if there is only one term, return n1
elif nterms == 1:
  print("Fibonacci sequence upto",nterms,":")
  print(n1)
# generate fibonacci sequence using WHILE
else:
  while count < nterms:
    if n1 % 2 != 0:
      odd_list.append(n1)
    else:
      even_list.append(n1)

    nth = n1 + n2
    # update values
    n1 = n2
    n2 = nth
    count += 1

print("Odd Fibonacci Number :" ,odd_list)
print("Even Fibonacci Number :" ,even_list)
```

OUTPUT:

```
How many terms? 10
Odd Fibonacci Number : [1, 1, 3, 5, 13, 21]
Even Fibonacci Number : [0, 2, 8, 34]
```

Pada 10 urutan pertama bilangan Fibonacci terdapat:

- Ganjil: 6 angka
- Genap: 4 angka

Implementasi while Loop

N – Urutan Awal Bilangan **Fibonacci** yang **< Angka Limit** Menggunakan Loop **WHILE**

```
# Program to display the Fibonacci sequence less than limit number in list

nlimit = int(input("Enter limit number :"))
fibonacci_sequence = []

# first two terms
n1, n2 = 0, 1
count = 0

# check if the limit number is valid
if nlimit <= 0:
  print("Please enter a positive integer")
# generate fibonacci sequence using WHILE
else:
  print("Fibonacci sequence <", nlimit, ":")
  while n1 < nlimit:
    fibonacci_sequence.append(n1)
    nth = n1 + n2
    # update values
    n1 = n2
    n2 = nth
    count += 1

print(fibonacci_sequence)
```

OUTPUT:

```
Enter limit number :20
Fibonacci sequence < 20 :
[0, 1, 1, 2, 3, 5, 8, 13]
```

Terdapat 8 angka Fibonacci yang lebih kecil dari 20

Jenis Loop

For Loop

Secara umum digunakan untuk menjalankan perintah yang berurutan (*sequential*).

Termasuk dalam kategori *definite iteration*: Jumlah perulangan loop sudah didefinisikan secara eksplisit di awal.

Syntax for loop:

```
for i in interable:
    <statement>
```

Implementasi for Loop

N – Urutan Awal Bilangan **Fibonacci** Menggunakan Loop **FOR**

```
# Program to display the Fibonacci sequence up to n-th term in list

nterms = int(input("How many terms? "))
fibonacci_sequence = []

# first two terms
n1, n2 = 0, 1
count = 0

# check if the number of terms is valid
if nterms <= 0:
  print("Please enter a positive integer")
# if there is only one term, return n1
elif nterms == 1:
  print("Fibonacci sequence upto",nterms,":")
  print(n1)
# generate fibonacci sequence using FOR
else:
  print("Fibonacci sequence:")
  fibonacci_sequence.extend((n1, n2))
  for i in range(2,nterms):
    nth = n1 + n2
    n1 = n2
    n2 = nth
    fibonacci_sequence.append(nth)

print(fibonacci_sequence)
```

OUTPUT:

```
How many terms? 10
Fibonacci sequence:
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]
```

10 urutan awal bilangan Fibonacci dengan pengulangan FOR

Implementasi for Loop

N – Urutan Awal Bilangan **Fibonacci Ganjil** dan **Genap** Menggunakan Loop **FOR**

```
# Program to display the Fibonacci sequence up to n-th term
# Divided by ODD & EVEN number list

nterms = int(input("How many terms? "))
odd_list = []
even_list = []

# first two terms
n1, n2 = 0, 1
count = 0

# check if the number of terms is valid
if nterms <= 0:
  print("Please enter a positive integer")
# if there is only one term, return n1
elif nterms == 1:
  print("Fibonacci sequence upto",nterms,":")
  print(n1)
# generate fibonacci sequence using FOR
else:
  even_list.append(n1)
  odd_list.append(n2)
  for i in range(2,nterms):
    nth = n1 + n2
    n1 = n2
    n2 = nth
    if nth % 2 != 0:
      odd_list.append(nth)
    else:
      even_list.append(nth)

print("Odd Fibonacci Number :" ,odd_list)
print("Even Fibonacci Number :" ,even_list)
```

OUTPUT:

```
How many terms? 10
Odd Fibonacci Number : [1, 1, 3, 5, 13, 21]
Even Fibonacci Number : [0, 2, 8, 34]
```

Pada 10 urutan pertama bilangan Fibonacci terdapat:

- Ganjil: 6 angka
- Genap: 4 angka

Control Loop

Break	Menghentikan loop yang sedang berjalan dan melanjutkannya ke baris selanjutnya setelah loop.
Continue	Menolak sisa *statement* pada loop yang sedang dijalankan dan melanjutkannya ke iterasi selanjutnya.
Pass	Digunakan ketika suatu *statement* memerlukan *syntactically* namun tidak menginginkan kode apapun untuk dijalankan.

Implementasi control Loop

DigitalSkola

Cek Bilangan Prima

```
# Check Prime Number

number = int(input("Enter any number : "))
if number>1:
  for i in range(2,number):
    if (number%i)==0:
      print(number, "is not prime number.")
      break
  else:
    print(number, "is prime number.")
else:
  print(number, "is not prime number.")
```

OUTPUT:

```
Enter any number : 7
7 is prime number.
```

```
Enter any number : 8
8 is not prime number.
```

Nested Loop

Nested Loop adalah loop didalam Loop.

Syntax nested while loop:

```
while <expression>:
    <statement>
    while <expression>:
    <statement>
```

Syntax nested for loop:

```
for i in iterable:
    <statement>
    for j in iterable_2:
        <statement>
```

Implementasi nested Loop

N – Urutan Awal Bilangan **Prima** yang **< Angka Limit**
Menggunakan Loop **FOR**

```
# Program to display the Prime numbers less than limit number in list

nlimit = int(input("Enter limit number: "))
prime_number = []
# Using FOR loop
for num in range(1, nlimit + 1):
  if(num > 1):
    for i in range(2,num):
      if(num % i)==0:
        break
    else:
      prime_number.append(num)

print("Prime number <", nlimit, ":")
print(prime_number)
```

OUTPUT:

```
Enter limit number: 20
Prime number < 20 :
[2, 3, 5, 7, 11, 13, 17, 19]
```

Terdapat 8 angka prima yang kurang dari 20

DigitalSkold

Implementasi nested Loop

N – Urutan Awal Bilangan **Prima** yang **< Angka Limit**
Menggunakan Loop **WHILE**

```
# Program to display the Prime numbers less than limit number in list

nlimit = int(input("Enter limit number: "))
prime_number = []
num = 1
# Using WHILE loop
while(num <= nlimit):
  count = 0
  i = 2
  while(i <= num//2):
    if(num % i == 0):
      count += 1
      break
    i += 1
  if (count == 0 and num!= 1):
    prime_number.append(num)
  num += 1

print("Prime number <", nlimit, ":")
print(prime_number)
```

OUTPUT:

```
Enter limit number: 20
Prime number < 20 :
[2, 3, 5, 7, 11, 13, 17, 19]
```

Terdapat 8 angka prima yang kurang dari 20

DigitalSkola

Implementasi nested Loop

N – Urutan Awal Bilangan **Prima** Menggunakan Loop **WHILE**

```
# Program to display the Prime numbers up to n-th term

nterms = int(input("How many terms? "))
prime_number = []
num = 1
count_prime = 1
# Using WHILE loop
while(count_prime <= nterms):
    count = 0
    i = 2
    while(i <= num//2):
        if(num % i == 0):
            count += 1
            break
        i += 1
    if (count == 0 and num!= 1):
        prime_number.append(num)
        count_prime += 1
    num += 1

print("Prime number :")
print(prime_number)
```

OUTPUT:

```
How many terms? 10
Prime number :
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

10 urutan awal bilangan Prima
dengan pengulangan While

DigitalSkold

Implementasi nested Loop

N – Urutan Awal Bilangan **Prima** Menggunakan Loop **WHILE**

```
# Program to display the Prime & Fibonacci numbers up to n-th term

nterms = int(input("How many terms? "))
fibonacci_prime = []
n1, n2 = 0, 1
# Using WHILE loop
while(n1 <= nterms):
    count = 0
    i = 2
    while(i <= n1//2):
        if(n1 % i == 0):
            count += 1
            break
        i += 1
    if (count == 0 and n1!= 1 and n1!=0):
        fibonacci_prime.append(n1)
    fibonacci = n1 + n2
    n1 = n2
    n2 = fibonacci

print("Fibonacci Prime Number :", len(fibonacci_prime))
print(fibonacci_prime)
```

OUTPUT:

```
How many terms? 500
Fibonacci Prime Number : 6
[2, 3, 5, 13, 89, 233]
```

Dari 500 urutan awal bilangan Fibonacci, terdapat 6 angka Prima

DigitalSkola

DigitalSkola

05 Array and Other Data-types

Array

Termasuk dalam non-primitive data- type, merupakan variable yang di-*assigned* by user.

Built-in array dalam Python:

- [list]
- (tuple)
- {Dictionary}

Characteristic:

- Dapat berisi multiple data-types.
- Kumpulan data dinomorkan / *indexed* mulai dari 0.
- Index tertinggi adalah total elements dikurangi 1.
  (Total Element = (n(max)-1))

Data Type

DigitalSkola

Primitives	Non-Primitives
Integer	Array
Float	List
String	Tuples
Boolean	Dictionary

Non-Primitives Data-types

DigitalSkola

List

Represented by []

- ✓ Ordered : menjaga urutan sesuai dengan saat input data
- ✓ Mutable : dapat diubah dan dimodifikasi itemnya
- ✓ Heterogenous : data dapat mengandung segala jenis data-types
- ✓ Contain Duplicate : mengizinkan data serupa/duplikat

```
[ ]  List = [11, 27.02, 'Omicron', ['Anugrah', 'Edo', 'Fajar', 'Fikri'] ]
```

List[0] List[1] List[2] List[3]

Non-Primitives Data-types

DigitalSkola

Dictionary

Represented by {}

- ✓ Unordered : items yg disimpan dalam dictionary tidak memiliki nilai index
- ✓ Mutable : dapat diubah dan dimodifikasi key/itemnya
- ✓ Unique : Key harus unique tidak boleh serupa/duplikat

```
Dict = {1: 'Anugrah',
        2: 'Edo',
        3: 'Fajar',
        4: 'Fikri'}
```

Dict{1}

Dict{2}

Dict{3}

Dict{4}

Implementasi Dictionary

Kebiasaan/History Transaksi Pelanggan menggunakan **Dictionary**

```
behaviour_user_transaction = [{
    'name': 'Anugrah',
    'platfrom': 'Web',
    'transaction': [{
        'product_name': 'Laptop',
        'product_price': 10000000,
        'product_qty': 2,
        'product_category': 'Electronics'
      },
      {
        'product_name': 'Handphone',
        'product_price': 2000000,
        'product_qty': 4,
        'product_category': 'Electronics'
      }]
},
{
    'name': 'Omicron',
    'platfrom': 'Mobile',
    'transaction': [{
        'product_name': 'PC',
        'product_price': 30000000,
        'product_qty': 1,
        'product_category': 'Electronics'
      },
      {
        'product_name': 'Tablet',
        'product_price': 5000000,
        'product_qty': 3,
        'product_category': 'Electronics'
      }]
}]
```

```
behaviour_user_transaction[0]['transaction']
```

OUTPUT:

```
[{'product_category': 'Electronics',
  'product_name': 'Laptop',
  'product_price': 10000000,
  'product_qty': 2},
 {'product_category': 'Electronics',
  'product_name': 'Handphone',
  'product_price': 2000000,
  'product_qty': 4}]
```

Output transaksi dari indeks list ke-0 (dengan nama user Anugrah)

Non-Primitives Data-types

DigitalSkola

Tuple

Represented by ()

- ✓ Ordered : menjaga urutan sesuai dengan saat input data
- ✓ Unchangeable : tidak dapat diubah dan dimodifikasi itemnya
- ✓ Heterogenous : data dapat mengandung segala jenis data-types
- ✓ Contain Duplicate : mengizinkan data serupa/duplikat

```
Tup = (11, 27.02, 'Omicron')
```

Tup(0) Tup(1) Tup(2)

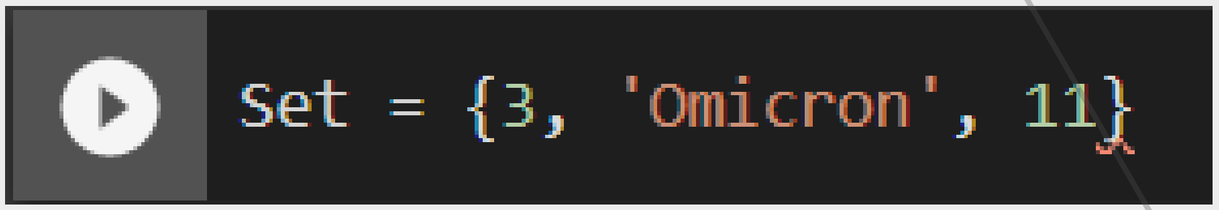
Non-Primitives Data-types

DigitalSkola

Set

Represented by ()

- ✓ Unordered : tidak menjaga urutan sesuai dengan saat input data
- ✓ Unchangeable : tidak dapat diubah dan dimodifikasi itemnya
- ✓ Heterogenous : data dapat mengandung segala jenis data-types
- ✓ Unique : tidak mengizinkan data serupa/duplikat



Penggunaan Pandas

Penggunaan **Pandas** dalam pembuatan Tabel Data Transaksi

```
import pandas as pd

data = [
        ['Laptop', 2, 10000000],
        ['Handphone', 4, 2000000],
        ['PC', 1, 30000000],
        ['Tablet', 3, 5000000]
       ]
df = pd.DataFrame(data)
df.columns = ['Product Name', 'Quantity', 'Price']
print(df)
```

OUTPUT:

```
  Product Name  Quantity     Price
0       Laptop         2  10000000
1    Handphone         4   2000000
2           PC         1  30000000
3       Tablet         3   5000000
```

TERIMA KASIH !

DigitalSkola