Understanding Logic (Equal, Greater / Less Than, Compare Arrays)

Arithmetic Operators

Operator merupakan suatu simbol dalam Python yang digunakan untuk menjalankan perhitungan aritmatika atau logika.

Nilai yang dioperasikan oleh operator disebut operan.

In [1]:
```
# Contoh
5 + 6

# simbol + merupakan operator (penjumlahan)
# angka 5 dan 6 merupakan operan
# angka 11 merupakan hasil dari operasi
```
Out[1]: 11

Macam-macam operator aritmatika

- "+" : Penjumlahan
- "-" : Pengurangan
- "*" : Perkalian
- / : Pembagian
- % : Modulo
- // : Floor Division
- ** : Eksponen

Catatan:

Modulo

Ketika 7 dibagi 4 itu sama dengan 1¾. Dengan modulo maka 7 % 4 = 3 (merupakan sisa dari pembagian 7:4) Contoh lainnya:

11 % 4 = 3 (11 sama dengan 4 dikali 2 ditambah 3)

8 % 2 = 0 (8 sama dengan 2 dikali 4 ditambah 0)

133 % 12 = 1 (132 sama dengan 12 dikali 11 ditambah 1)

> Floor Divison

hasil dari perhitungan pembagian yang dibulatkan ke nilai terbawah. misal 12 : 5 maka hasilnya 2 koma. Dengan floor division dibulatkan menjadi 2.

> Eksponen

Eksponen merupakan perpangkatan dalam python. misalnya 2 pangkat 3 maka cara menghitungnya 2**3

Operator	Meaning	Example
+	Add two operands or unary plus	x + y+ 2
-	Subtract right operand from the left or unary minus	x - y- 2
*	Multiply two operands	x * y
/	Divide left operand by the right one (always results into float)	x / y
%	Modulus - remainder of the division of left operand by the right	x % y (remainder of x/y)
//	Floor division - division that results into whole number adjusted to the left in the number line	x // y
**	Exponent - left operand raised to the power of right	x**y (x to the power y)

In [2]:
```
# Contoh
x = 18
y = 11

# Output Penjumlahan
print('Penjumlahan: ' + 'x + y =', x + y)

# Output Pengurangan
print('Pengurangan: ' + 'x - y =', x - y)

# Output Perkalian
print('Perkalian: ' + 'x * y =', x * y)

# Output Pembagian
print('Pembagian: ' + 'x / y =', x / y)

# Output Modulo
print('Modulo: ' + 'x % y =', x % y)

# Output Floor Divison
print('Floor Divison: ' + 'x // y =', x // y)

# Output eksponen
print('Eksponensial: ' + 'x ** y =', x ** y)
```
```
Penjumlahan: x + y = 29
Pengurangan: x - y = 7
Perkalian: x * y = 198
Pembagian: x / y = 1.6363636363636365
Modulo: x % y = 7
Floor Divison: x // y = 1
Eksponensial: x ** y = 64268410079232
```

Comparison operators

Operator perbandingan digunakan untuk membandingkan suatu nilai dengan nilai lainnya.

Operator ini akan menghasilkan output berupa pernyataan True atau False. Dalam bahasa sehari-hari, True berarti iya, False berarti tidak.

Jika diubah menjadi interger, maka True bernilai 1 sedangkan False bernilai 0.

Operator	Meaning	Example
>	Greater than - True if left operand is greater than the right	x > y
<	Less than - True if left operand is less than the right	x < y
==	Equal to - True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to - True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to - True if left operand is less than or equal to the right	x <= y

In [3]:
```
x = 15
y = 17

# output: x > y adalah False
print('Apakah ' + 'x > y, jawabannya adalah', x > y)

# output: x < y adalah True
print('Apakah ' + 'x < y, jawabannya adalah', x < y)

# output: x == y adalah False
print('Apakah ' + 'x == y, jawabannya adalah', x == y)

# output: x != y adalah True
print('Apakah ' + 'x != y, jawabannya adalah', x != y)

# output: x >= y adalah False
print('Apakah ' + 'x >= y, jawabannya adalah', x >= y)

# output: x <= y adalah True
print('Apakah ' + 'x > y, jawabannya adalah', x <= y)
```
```
Apakah x > y, jawabannya adalah False
Apakah x < y, jawabannya adalah True
Apakah x == y, jawabannya adalah False
Apakah x != y, jawabannya adalah True
Apakah x >= y, jawabannya adalah False
Apakah x > y, jawabannya adalah True
```

In [4]:
```
x = 19
y = 16

# output: x > y adalah True
print('Apakah ' + 'x > y, jawabannya adalah', x > y)

# output: x < y adalah False
print('Apakah ' + 'x < y, jawabannya adalah', x < y)

# output: x == y adalah False
print('Apakah ' + 'x == y, jawabannya adalah', x == y)

# output: x != y adalah True
print('Apakah ' + 'x != y, jawabannya adalah', x != y)

# output: x >= y adalah True
print('Apakah ' + 'x >= y, jawabannya adalah', x >= y)

# output: x <= y adalah False
print('Apakah ' + 'x > y, jawabannya adalah', x <= y)
```
```
Apakah x > y, jawabannya adalah True
Apakah x < y, jawabannya adalah False
Apakah x == y, jawabannya adalah False
Apakah x != y, jawabannya adalah True
Apakah x >= y, jawabannya adalah True
Apakah x > y, jawabannya adalah False
```

In [5]:
```
x = 2
y = 2

# output: x > y adalah False
print('Apakah ' + 'x > y, jawabannya adalah', x > y)

# output: x < y adalah False
print('Apakah ' + 'x < y, jawabannya adalah', x < y)

# output: x == y adalah True
print('Apakah ' + 'x == y, jawabannya adalah', x == y)

# output: x != y adalah False
print('Apakah ' + 'x != y, jawabannya adalah', x != y)

# output: x >= y adalah True
print('Apakah ' + 'x >= y, jawabannya adalah', x >= y)

# output: x <= y adalah True
print('Apakah ' + 'x > y, jawabannya adalah', x <= y)
```
```
Apakah x > y, jawabannya adalah False
Apakah x < y, jawabannya adalah False
Apakah x == y, jawabannya adalah True
Apakah x != y, jawabannya adalah False
Apakah x >= y, jawabannya adalah True
Apakah x > y, jawabannya adalah True
```

In [6]:
```
x = "hadi"
y = "Hadi"

# Huruf kapital juga menengaruhi output yang dihasilkan

# output: x > y adalah True
print('Apakah ' + 'x > y, jawabannya adalah', x > y)

# output: x < y adalah False
print('Apakah ' + 'x < y, jawabannya adalah', x < y)

# output: x == y adalah False
print('Apakah ' + 'x == y, jawabannya adalah', x == y)

# output: x != y adalah True
print('Apakah ' + 'x != y, jawabannya adalah', x != y)

# output: x >= y adalah True
print('Apakah ' + 'x >= y, jawabannya adalah', x >= y)

# output: x <= y adalah False
print('Apakah ' + 'x > y, jawabannya adalah', x <= y)
```
```
Apakah x > y, jawabannya adalah True
Apakah x < y, jawabannya adalah False
Apakah x == y, jawabannya adalah False
Apakah x != y, jawabannya adalah True
Apakah x >= y, jawabannya adalah True
Apakah x > y, jawabannya adalah False
```

Logical Operators

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

Contoh "AND", terdapat dua perhitungan, A dan B.

A menampilkan hasil dari comparasion operator yang mana hasilnya True, dan B menampilkan hasil dari comparasion operator yang mana hasilnya True.

Ketika A dan B digabungkan dengan logical operators maka hasilnya True. Namun jika salah satu antara A atau B menghasilkan output False, maka hasilnya False.

Keduanya harus True agar menghasilkan nilai True. Sedangkan untuk "OR", Keduanya tidak harus True untuk menghasilkan output True. Apabila hasil A itu True dan B itu False, maka logical OR akan menghasilkan output True.

In [7]:
```
x = True
y = False

print('x and y is', x and y)
print('x or y is', x or y)
print('not x is', not x)
```
```
x and y is False
x or y is True
not x is False
```

In [8]:
```
x = True
y = True

print('x and y is', x and y)
print('x or y is', x or y)
print('not x is', not x)
```
```
x and y is True
x or y is True
not x is False
```

In [9]:
```
x = False
y = False

print('x and y is', x and y)
print('x or y is', x or y)
print('not x is', not x)
```
```
x and y is False
x or y is False
not x is True
```

In [10]:
```
x = False
y = True

print('x and y is', x and y)
print('x or y is', x or y)
print('not x is', not x)

# not hanya untuk satu nilai. tidak bisa digabungkan seperti and atau or
```
```
x and y is False
x or y is True
not x is True
```

Bitwise Operators

Operator bitwise bertindak atas operan seolah-olah itu adalah string digit biner.

Mereka beroperasi sedikit demi sedikit. Misalnya, 2 adalah 10 dalam biner dan 7 adalah 111.

Operator	Meaning	Example
&	Bitwise AND	x & y = 0 (0000 0000)
|	Bitwise OR	x | y = 14 (0000 1110)
~	Bitwise NOT	~x = -11 (1111 0101)
^	Bitwise XOR	x ^ y = 14 (0000 1110)
>>	Bitwise right shift	x >> 2 = 2 (0000 0010)
<<	Bitwise left shift	x << 2 = 40 (0010 1000)

Assignment Operators

Operator penugasan digunakan dengan Python untuk menetapkan nilai ke variabel.

Terdapat berbagai operator gabungan dalam Python seperti a += 5 yang menambahkan variabel dan kemudian memberikan nilai yang sama. Yang mana hal tersebut sama dengan a = a + 5.

Operator	Example	Equivalent to
=	x = 5	x = 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5
//=	x //= 5	x = x // 5
**=	x **= 5	x = x ** 5
&=	x &= 5	x = x & 5
|=	x |= 5	x = x | 5
^=	x ^= 5	x = x ^ 5
>>=	x >>= 5	x = x >> 5
<<=	x <<= 5	x = x << 5

In [11]:
```
a = 19
b = 25
```

In [12]:
```
# operator assignment
a += 5   #>> sama seperti a = a + 5
a
```
Out[12]: 24

In [13]:
```
b += 7 #>> sama seperti b = b + 7
b
```
Out[13]: 32

In [14]:
```
a -= 5   #>> sama seperti a = a - 5
a
```
Out[14]: 19

In [15]:
```
c = 10
c += 10
c
```
Out[15]: 20

In [16]:
```
c = 15
c *= 2
c
```
Out[16]: 30

Special Operators

Operator yang berfungsi untuk membandingkan nilai satu dengan yang lainnya bersifat identik atau tidak.

Operator	Meaning	Example
is	True if the operands are identical (refer to the same object)	x is True
is not	True if the operands are not identical (do not refer to the same object)	x is not True

In [17]:
```
x1 = 5
y1 = 5
x2 = 'Hello'
y2 = 'Hello'

print('x1 is y1:', x1 is y1)
# output True karena identik atau sama

print('x1 is not y1:', x1 is not y1)
# output False karena tidak identik

print('x2 is y2:', x2 is y2)
# output True karena identik atau sama

print('x2 is not y2:', x2 is not y2)
# output False karena tidak identik
```
```
x1 is y1: True
x1 is not y1: False
x2 is y2: True
x2 is not y2: False
```

In [18]:
```
nama = 'Muhamad Hadiyansyah'
'M' in nama
```
Out[18]: True

In [19]:
```
'D' in nama # Output False dikarenakan sensitif terhadap huruf kapital atau tidak
```
Out[19]: False

Membership Operators

Disebut juga sebagai operator keanggotaan. Digunakan untuk menguji apakah nilai atau variabel berada atau tidak di dalam suatu urutan atau tempat yang sama (string, list, tuple, set dan kamus).

Operator	Meaning	Example
in	True if value/variable is found in the sequence	5 in x
not in	True if value/variable is not found in the sequence	5 not in x

In [20]:
```
x = 'Hello World'

print('H' in x) # maksudnya adalah apakah huruf H berada di dalam variabel x
print('h' in x) # besar kecilnya huruf menjadi sensitif
print('o' in x)
print('d' in x)
print('g' in x)
print('z' not in x)
```
```
True
False
True
True
False
True
```

In [21]:
```
# contoh menggunakan dictonary
y = {1:'a', 2:'b'}

print(1 in y)
print('c' not in y)
print('b' not in y)
print(2 not in y)
```
```
True
True
True
False
```

IF ELSE Statement

Digunakan untuk mengambilan keputusan, yang mana diperlukan ketika kita hanya ingin mengeksekusi kode jika kondisi tertentu terpenuhi. Rumus:

if test expression:

```
statement(s)
```

Dalam pelaksanaannya program akan mengevaluasi ekspresi tes dan akan mengeksekusi pernyataan hanya jika ekspresi tes tersebut adalah True.

Jika hasil dari uji ekspresi adalah False, maka pernyataan tersebut tidak dijalankan.

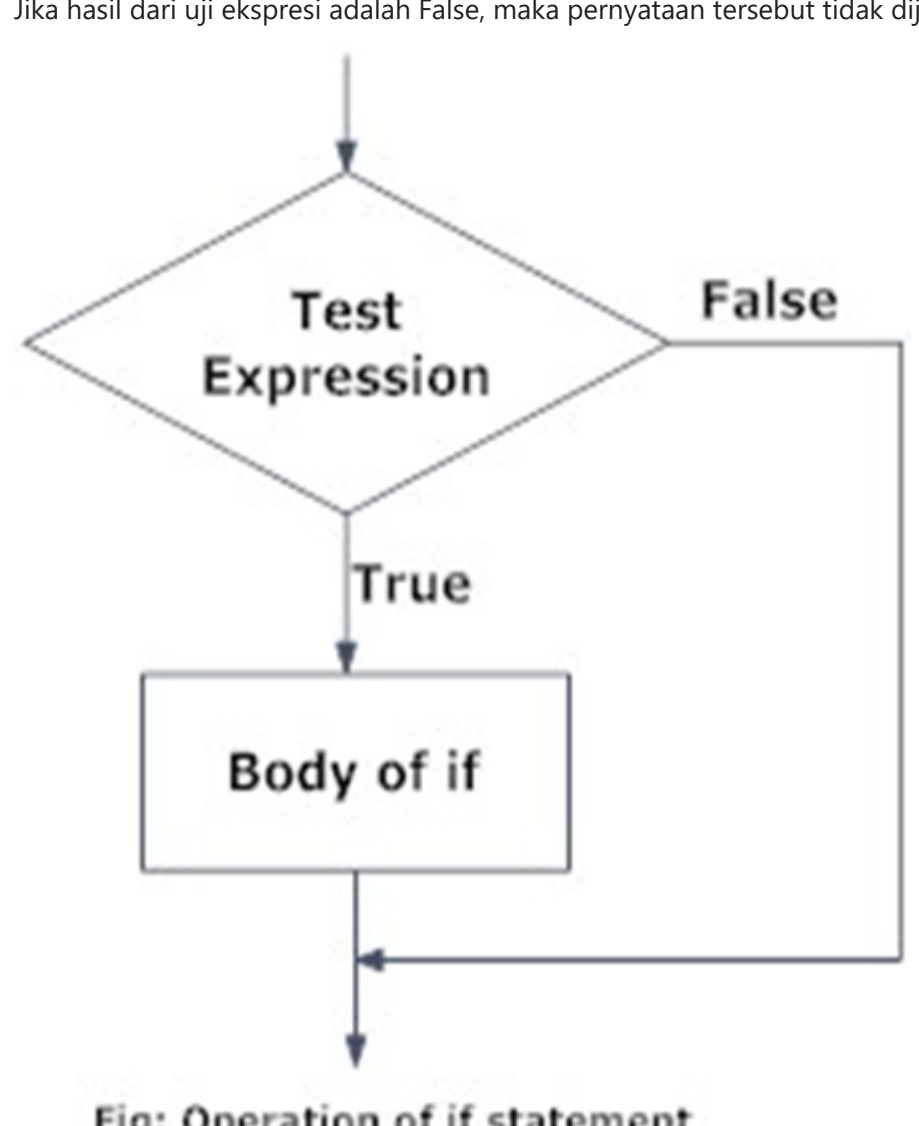


Fig: Operation of if statement

In [22]:
```
# misalnya nilai mahasiswa jika di atas 75 akan menghasilkan output LULUS

nilai = 90

if nilai > 75:
    print('Selamat kamu lulus')
```
```
Selamat kamu lulus
```

In [23]:
```
# menggunakan f string untuk menghasilkan output string dan angka, dengan cara ditambahkan f di depan dan {} un

nilai = 90

if nilai > 75:
    print(f'Nilai kamu {nilai}, maka kamu lulus')
```
```
Nilai kamu 90, maka kamu lulus
```

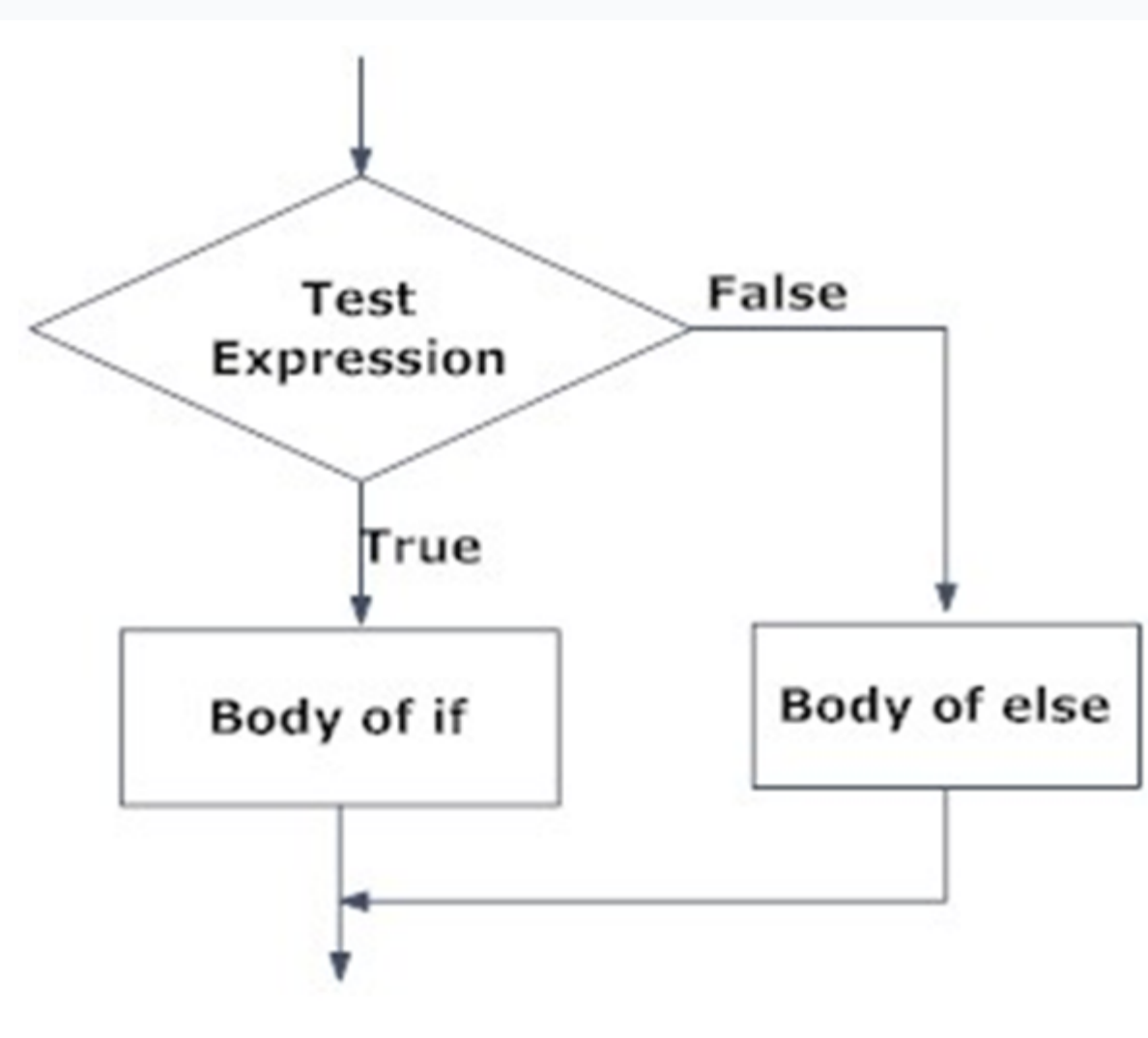
In [24]:
```
# If the number is positive, we print an appropriate message

num = 5
if num > 0:
  print(num, 'is a positive number.')
print('This is always printed.')

num = -2
if num > 0:
  print(num, 'is a positive number.')
print('This is always printed.')
```
```
5 is a positive number.
This is always printed.
This is always printed.
```

Python IF ELSE Statement

Pernyataan if..else mengevaluasi ekspresi pengujian dan akan mengeksekusi isi if hanya jika kondisi pengujian adalah True.

Jika kondisinya False, maka body of else dieksekusi. Indentasi digunakan untuk memisahkan blok.

```
if test expression:
    Body of if
else:
    Body of else
```



Alur kerja IF ELSE Statement

Fig: Operation of if...else statement

In [25]:
```
# Misalnya kita ingin membuat program untuk memeriksa apakah angka yang kita berikan itu positif atau negatif

angka = 3

if angka >= 0:
  print("Positive or Zero")
else:
  print("Negative")
```
```
Positive or Zero
```

In [26]:
```
# Membuat program untuk mengetahui nilai saya itu dinyatakan lulus atau tidak, dengan asumsi bahwa batas kelulu

nilai_saya = 80

if nilai_saya >= 75:
    print('Nilai saya', nilai_saya, 'maka saya lulus')
else:
    print('Saya tidak lulus')
```
```
Nilai saya 80 maka saya lulus
```

In [27]:
```
num = 74

if num >= 75:
  print("Selamat anda lulus ujian")
else:
  print("Maaf anda kurang beruntung")
```
```
Maaf anda kurang beruntung
```

In [28]:
```
num = 75

if num >= 75:
  print("Selamat anda lulus ujian")
else:
  print("Maaf anda kurang beruntung")
```
```
Selamat anda lulus ujian
```

IF ELIF ELSE Statement

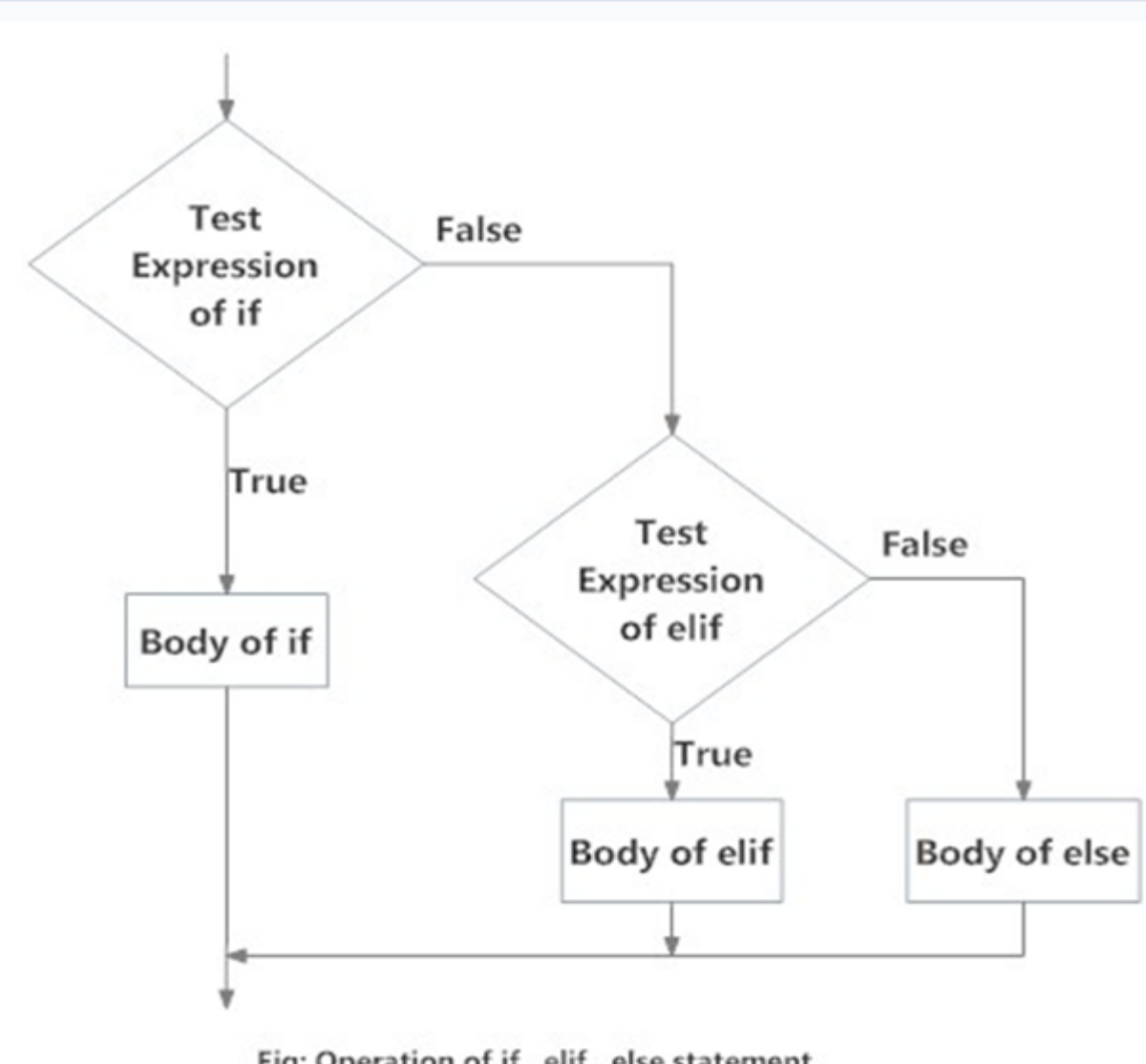
Elif merupakan singkatan dari else if.

Dengan menggunakan elif, kita dapat memeriksa beberapa ekspresi tidak terbatas hanya dua ekspresi yaitu if dan else.

Elif dapat digunakan sebanyak yang kita mau, sedangkan if atau else hanya satu ekspresi.

Dalam pelaksanaannya program akan memeriksa ekspresi if, jika false maka program akan memeriksa ekspresi elif, jika elif juga false maka program akan menampilkan hasil dari ekspresi else.

```
if test expression:
    Body of if
elif test expression:
    Body of elif
else:
    Body of else
```



Alur kerja if elif else:

Fig: Operation of if...elif...else statement

In [29]:
```
num = 3.4

if num > 0:
  print("Positive Number")
elif num == 0:
  print("Zero Number")
else:
  print("Negative Number")
```
```
Positive Number
```

In [30]:
```
nilai = 30

if nilai >= 90:
  print("Sangat Baik")
elif nilai >= 80:
  print("Baik")
elif nilai >= 70:
  print("Cukup Baik")
else:
  print("Kura")
```
```
Kura
```

In [31]:
```
nilai = int(input("Nilai anda: "))

if nilai >= 90:
  print("Sangat Baik")
elif nilai >= 80:
  print("Baik")
elif nilai >= 70:
  print("Cukup Baik")
else:
  print("Kurang")
```
```
Nilai anda: 80
Baik
```

In [32]:
```
a = 200
b = 33
if b > a:
  print('b is greater than a')
elif b == a:
  print('b is equal to a')
else:
  print('b less than a')
```
```
b less than a
```

In [33]:
```
tinggi_andi = 165
tinggi_berta = 174

if tinggi_andi > tinggi_berta:
  print('andi lebih tinggi dari berta')
elif tinggi_andi == tinggi_berta:
  print('andi memiliki tinggi yang sama dengan berta')
else:
  print('berta lebih tinggi dari andi')
```
```
berta lebih tinggi dari andi
```

Python Nested IF ELSE

Kita dapat memiliki pernyataan if ... elif ... else di dalam pernyataan if ... elif ... else.

Hal itu dapat disebut dengan istilah nested dalam pemrograman komputer.

Sejumlah pernyataan dapat bertumpuk di dalam satu sama lain. Indentasi adalah satu-satunya cara untuk mengetahui tingkat bersarang.

Penggunaan ini dapat membingungkan, oleh karena itu jika tidak perlu harus dihindari.

In [34]:
```
num = float(input("Enter a number: "))
if num >= 0:
    if num == 0:
        print("zero")
    else:
        print("positive")
else:
    print("negative")
```
```
Enter a number: 2
positive
```

In [35]:
```
nilai = int(input("Nilai Mahasiswa: "))
if nilai >= 75:
    if nilai == 75:
        print("KKM")
    else:
        print("LULUS")
else:
    print("TIDAK LULUS")
```
```
Nilai Mahasiswa: 76
LULUS
```

In [36]:
```
x = 41

if x > 10:
  print('above ten')
  if x > 20:
    print('and also above 20')
  else:
    print('but not above 20')
else:
  print('just number')
```
```
above ten
and also above 20
```

In [37]:
```
iq = 120

if iq > 100:
  print('Kamu pintar')
  if iq > 115:
    print('dan hebat')
  else:
    print('kamu di atas rata-rata')
else:
  print('kamu luar biasa')
```
```
Kamu pintar
dan hebat
```

Introduction for Loops

Perulangan for di Python digunakan untuk mengulangi urutan (list, tuple, string) atau objek iterable lainnya.

Iterasi pada suatu urutan disebut traversal.

```
for val in sequence:
        Body of for
```

Val adalah variabel yang mengambil nilai item di dalam urutan pada setiap iterasi.

Pengulangan berlanjut hingga kita mencapai item terakhir dalam urutan. Body for loop dipisahkan dari kode lainnya menggunakan indentasi.

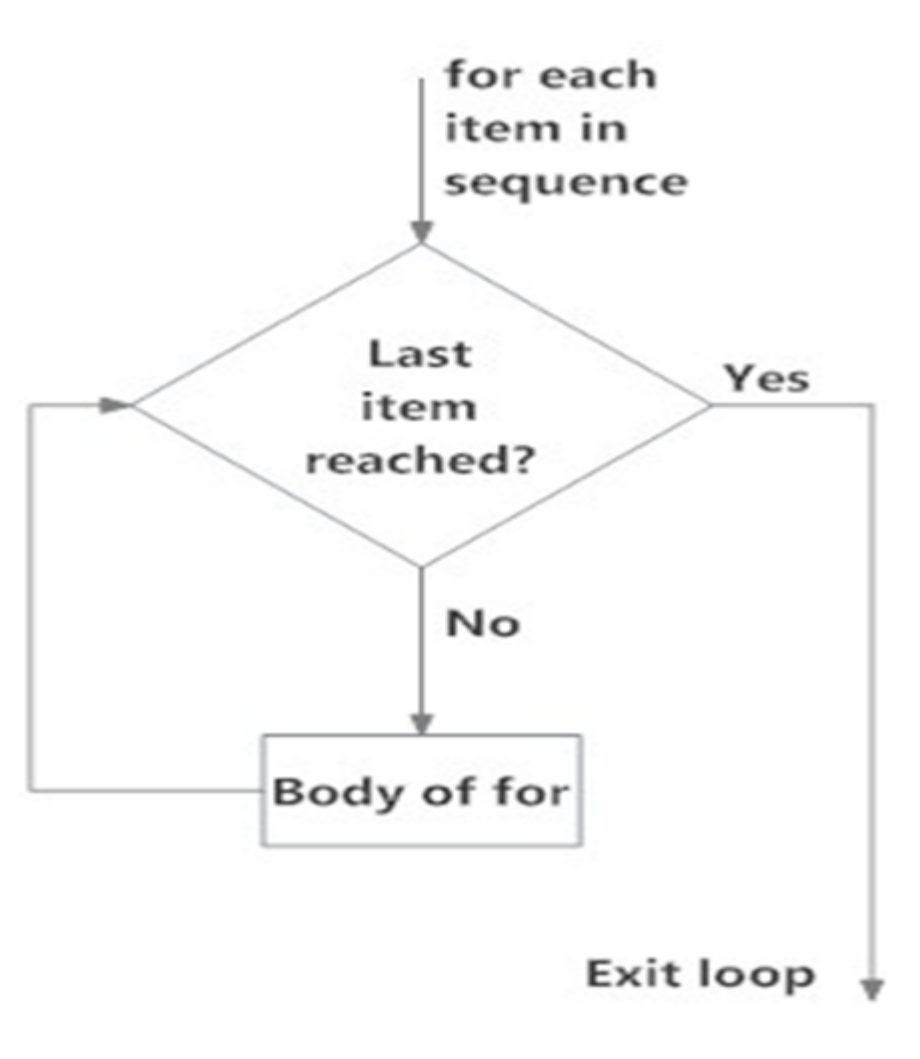


Fig: operation of for loop

Alur Kerja For Loops:

In [38]:
```
# Program to find the sum of all numbers stored in a list

# List of numbers
numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]

# variable to store the sum
sum = 0

# iterate over the list
for val in numbers:
  sum = sum + val

print("The sum is", sum)
```
```
The sum is 48
```

In [39]:
```
Total_skor = [75, 39, 48, 69, 30, 60, 90]
jumlah_total_skor = 0
for val in Total_skor:
  jumlah_total_skor += val

print('Total skor kamu adalah', jumlah_total_skor)
```
```
Total skor kamu adalah 411
```

The Range Function

Kita bisa menghasilkan urutan angka menggunakan fungsi range ().

range (10) akan menghasilkan angka dari 0 sampai 9 (10 angka).

Dapat menggunakan start, stop dan step sebagai range (start, stop, step_size).

step_size default ke 1 jika tidak disediakan.

Objek range adalah "malas" dalam arti karena tidak menghasilkan setiap angka yang "dikandungnya" saat kita membuatnya.

Dikarenakan ini bukan merupakan iterator, maka fungsi ini dapat mendukung operasi in, len dll

Fungsi ini tidak menyimpan semua nilai dalam memori; itu akan menjadi tidak efisien. Jadi, ia mengingat ukuran mulai, berhenti, langkah, dan menghasilkan angka berikutnya saat dalam pengoperasian.

Untuk memaksa fungsi ini mengeluarkan semua item, kita dapat menggunakan function list ().

In [40]:
```
# Contoh The Range Function
print(range(10))
print(list(range(10)))
```
```
range(0, 10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

In [41]:
```
# contoh menggunakan start:stop
# misal 2:8
print(list(range(2, 8)))

# misal 3 : 13
print(list(range(3, 13)))
```
```
[2, 3, 4, 5, 6, 7]
[3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
```

In [42]:
```
# contoh menggunakan start : stop : Step
# misal 2 : 20 : 3
print(list(range(2, 20, 3)))

# misal 1 : 44 : 2
print(list(range(1, 44, 2)))
```
```
[2, 5, 8, 11, 14, 17]
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43]
```

The Range Function

Kita dapat menggunakan fungsi range () di for loop untuk melakukan iterasi melalui urutan angka.

Ini dapat dikombinasikan dengan fungsi len () untuk melakukan iterasi melalui urutan menggunakan pengindeksan.

In [43]:
```
# program to iterate through a list using indexing
genre = ['pop', 'rock', 'jazz']

# iterate over the list using index
for i in range(len(genre)):
        print('I love', genre[i])
```
```
I love pop
I love rock
I love jazz
```

In [44]:
```
hadi = ['baik', 'sabar', 'ceria', 'ganteng', 'humoris', 'pinter', 'penyayang']

for i in range(len(hadi)):
  print('Hadi merupakan seseorang yang', hadi[i])
```
```
Hadi merupakan seseorang yang baik
Hadi merupakan seseorang yang sabar
Hadi merupakan seseorang yang ceria
Hadi merupakan seseorang yang ganteng
Hadi merupakan seseorang yang humoris
Hadi merupakan seseorang yang pinter
Hadi merupakan seseorang yang penyayang
```

In [45]:
```
for x in 'apple':
  print(x)
```
```
a
p
p
l
e
```

In [46]:
```
print('keranjang saya berisi')

keranjang_buah = ['apel', 'mangga', 'jambu', 'pepaya', 'duren']
for x in keranjang_buah:
  print(x)
```
```
keranjang saya berisi
apel
mangga
jambu
pepaya
duren
```

In [47]:
```
iter_ = 5

for i in range(iter_):
  print('Perulangan ke-', str(i))
```
```
Perulangan ke- 0
Perulangan ke- 1
Perulangan ke- 2
Perulangan ke- 3
Perulangan ke- 4
```

In [48]:
```
kode = 10

for i in range(kode):
  print('kode ke', str(i))
```
```
kode ke 0
kode ke 1
kode ke 2
kode ke 3
kode ke 4
kode ke 5
kode ke 6
kode ke 7
kode ke 8
kode ke 9
```

In [49]:
```
for x in range(8):
  print(x)
```
```
0
1
2
3
4
5
6
7
```

In [50]:
```
for x in range(0, 11):
  print('kode baju ke-', x)
```
```
kode baju ke- 0
kode baju ke- 1
kode baju ke- 2
kode baju ke- 3
kode baju ke- 4
kode baju ke- 5
kode baju ke- 6
kode baju ke- 7
kode baju ke- 8
kode baju ke- 9
kode baju ke- 10
```

Introduction while loops

Perulangan while dalam Python digunakan untuk mengulang satu blok kode selama ekspresi uji (kondisi) benar.

Biasanya menggunakan perulangan ini ketika kami tidak mengetahui berapa kali untuk mengulang sebelumnya.

```
while test_expression:
    Body of while
```

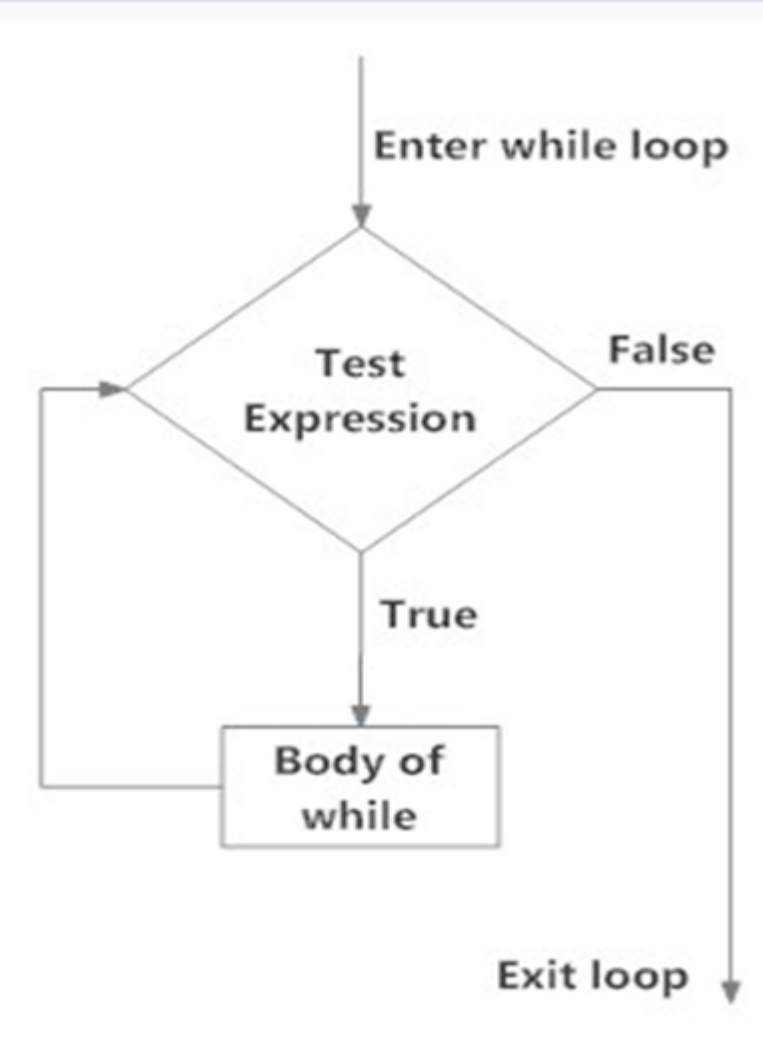


Fig: operation of while loop

Alur kerja While loop:

In [51]:
```
# program to add natural
# numbers up to
# sum = 1 + 2 + 3 + ... + n


n = 10

# initialize sum and counter
sum = 0
i = 1

while i <= n:
  sum = sum + i
  i = i + 1

# print the sum
print('The sum is', sum)
```
```
The sum is 55
```

In [52]:
```
n = 100

sum = 0
i = 2
while i <= 100:
  sum = sum + i
  i = i + 2

print("The sum is", sum)
```
```
The sum is 2550
```

In [53]:
```
i = 1
while i < 6:
  print(i)
  i += 1
```
```
1
2
3
4
5
```

In [54]:
```
i = 0
while i < 5:
  print(i)
  i += 1
```
```
0
1
2
3
4
```

List Iteration Using While Complex Conditional Expressions

While Loop with else

Sama seperti for loop, while loop juga dapat memiliki blok else opsional.

Bagian lain dijalankan jika kondisi di loop sementara bernilai False.

Perulangan while dapat diakhiri dengan pernyataan break. Dalam kasus seperti itu, bagian lain akan diabaikan.

Oleh karena itu, bagian lain loop sementara berjalan jika tidak ada pemutusan yang terjadi dan kondisinya salah.

In [55]:
```
counter = 0

while counter < 3:
  print("Inside loop")
  counter = counter + 1
else:
  print("Inside else")
```
```
Inside loop
Inside loop
Inside loop
Inside else
```

In [56]:
```
setuju = 0

while setuju <= 10:
  print("Kamu disetujui")
  setuju = setuju + 1
else:
  print("kamu ditolak")
```
```
Kamu disetujui
Kamu disetujui
Kamu disetujui
Kamu disetujui
Kamu disetujui
Kamu disetujui
Kamu disetujui
Kamu disetujui
Kamu disetujui
Kamu disetujui
Kamu disetujui
kamu ditolak
```

In [57]:
```
i = 1
while i < 7:
  print(i)
  i += 1
else:
  print('stop')
```
```
1
2
3
4
5
6
stop
```

In [58]:
```
i = 1
while i < 4:
    print(i)
    i += 1
else:
    print('Mulai!!!')
```
```
1
2
3
Mulai!!!
```

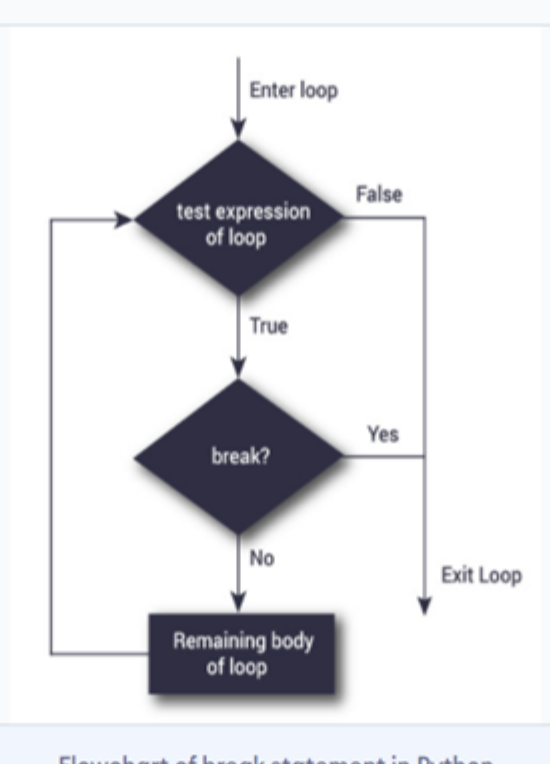
Break Statements (While Loops)

Pernyataan break menghentikan loop yang berisi itu. Kontrol program mengalir ke pernyataan segera setelah badan loop.

Jika pernyataan break berada di dalam loop nested (loop di dalam loop lain), pernyataan break akan menghentikan loop terdalam.

```
break
```

Flowchart of break



Flowchart of break statement in Python

```
for var in sequence:
    # codes inside for loop
    if  condition:
        break
    # codes inside for loop

# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if  condition:
        break
    # codes inside while loop

# codes outside while loop
```

In [59]:
```
# use of break statement inside the loop

for val in "string":
  if val == "i":
    break
  print(val)

print("The end")
```
```
s
t
r
The end
```

In [60]:
```
for val in list(range(10)):
  if val == 7:
    break
  print(val)

print("Stop")
```
```
0
1
2
3
4
5
6
Stop
```

Break (For Loops)

In [61]:
```
fruits = ['apple', 'banana', 'grape']
for x in fruits:
  print(x)
  if x == 'banana':
    break
```
```
apple
banana
```

In [62]:
```
nilai = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for x in nilai:
  print(x)
  if x == 7:
    break
```
```
1
2
3
4
5
6
7
```

Break (While Loops)

In [63]:
```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```
```
1
2
3
```

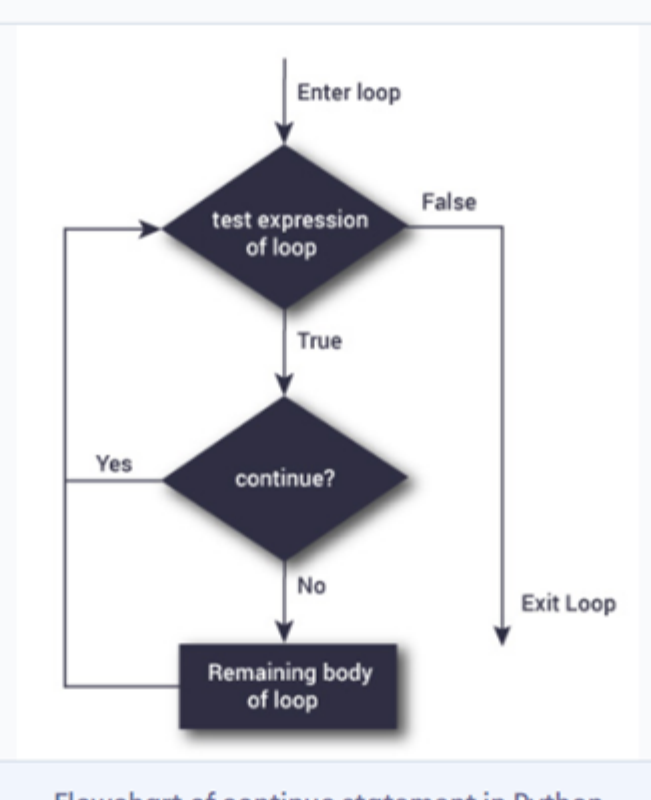
Continue Statements (While Loops)

Pernyataan continue digunakan untuk melewati sisa kode di dalam loop hanya untuk iterasi saat ini.

Loop tidak berhenti tetapi berlanjut dengan iterasi berikutnya.

```
continue
```

Flowchart of continue



Flowchart of continue statement in Python

```
for var in sequence:
    # codes inside for loop
    if  condition:
        continue
    # codes inside for loop

# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if  condition:
        continue
    # codes  inside while loop

# codes outside while loop
```

In [64]:
```
# Program to show the use of continue statement inside loops

for val in 'string':
  if val == 'n':
    continue
  print(val)

print("The end")
```
```
s
t
r
i
g
The end
```

In [65]:
```
nilai_mahasiswa = [77, 89, 35, 76, 87, 90, 50, 77]
print("Daftar Peserta yang Lolos Seleksi CPNS")

for val in nilai_mahasiswa:
  if val <= 75:
    continue
  print(val)

print("Terima kasih")
```
```
Daftar Peserta yang Lolos Seleksi CPNS
77
89
76
87
90
77
Terima kasih
```

Continue (For Loops)

In [66]:
```
fruits = ['apple', 'banana', 'cherry']
for x in fruits:
  if x == 'banana':
    continue
  print(x)
```
```
apple
cherry
```

In [67]:
```
angka = [1, 2, 3, 4, 5, 6]
for x in angka:
  if x == 4:
    continue
  print(x)
```
```
1
2
3
5
6
```

Loop Dictonary

Anda dapat melakukan loop melalui dictionary dengan menggunakan for loop.

Saat mengulang melalui dictionary, nilai yang dikembalikan adalah kunci dari dictionary, tetapi ada metode untuk mengembalikan nilai juga.

```
for x in thisdict:
  print(x)

for x in thisdict:
  print(thisdict[x])

for x in thisdict.values():
  print(x)

for x, y in thisdict.items():
  print(x, y)
```

In [68]:
```
thisdict = {
    'brand' : 'Ford',
    'model' : 'mustang',
    'year' : 1998
}
print(thisdict)
```
```
{'brand': 'Ford', 'model': 'mustang', 'year': 1998}
```

In [69]:
```
for x in thisdict:
  print(thisdict[x])
```
```
Ford
mustang
1998
```

In [70]:
```
for x in thisdict.values():
  print(x)
```
```
Ford
mustang
1998
```

In [71]:
```
for x in thisdict.keys():
  print(x)
```
```
brand
model
year
```

In [72]:
```
for x, y in thisdict.items():
  print(x, y)
```
```
brand Ford
model mustang
year 1998
```