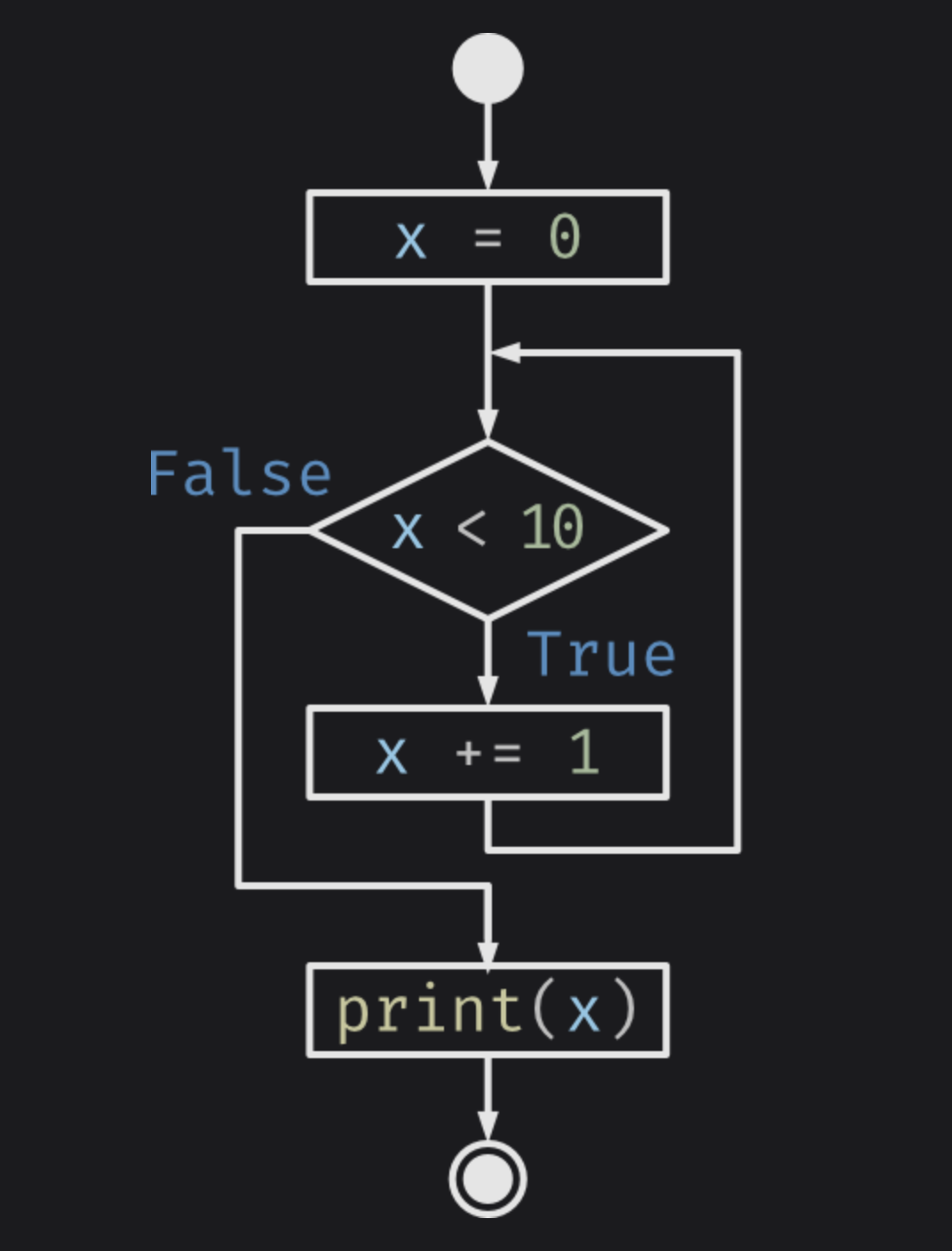
# Python Iterations / Loops

## While Loops

While loop dalam Python digunakan untuk mengulangi blok kode selama kondisinya True.

Biasanya loop ini digunakan ketika kita tidak tahu berapa kali harus melakukan looping.



Dalam while loop, ekspresi pengujian diperiksa terlebih dahulu. Hanya jika kondisi test menghasilkan nilai True, maka kode akan dieksekusi. Setelah satu iterasi, kondisi pengujian diperiksa lagi. Proses ini berlanjut hingga kondisi bernilai False.

## 

### 

| *# add natural numbers up to sum = 1+2+3+...+n* *# To take input from the user,* *# n = int(input("Enter n: "))* n = 10  *# initialize sum and counter* sum = 0 i = 1  while i <= 10:  sum = sum + i  i = i+1 *# update counter or i += 1*  print(i, sum)  *# print the sum* print("The sum is", sum) |
| --- |

### While loop dengan else

While loop juga dapat memiliki kondisi else secara opsional.

Bagian else dieksekusi jika kondisi dalam while loop bernilai False.

While loop dapat diakhiri dengan pernyataan break. Dalam kasus seperti itu, bagian lain diabaikan.

| '''Example to illustrate the use of else statement with the while loop'''  counter = 0  while counter < 3:  print("While loop")  counter = counter + 1 else:  print("else") |
| --- |

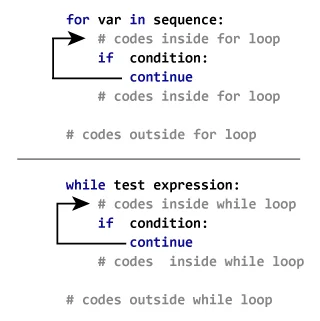
| count = 0 while count < 5:  print count, " is less than 5"  count = count + 1 else:  print count, " is not less than 5" |
| --- |

### 

### Break & Continue in While Loops

**While loop with Continue**

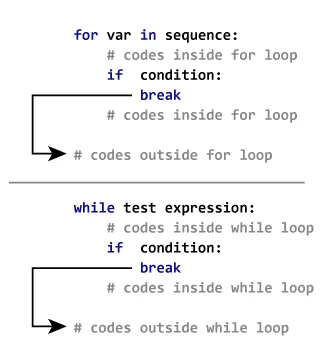
Continue dapat digunakan untuk melewati eksekusi pernyataan lebih lanjut dalam while loop dan melanjutkan iterasi while loop berikutnya.



| x = 3   while x < 10:  x += 1  if x > 7:  continue  print(x) |
| --- |

**While loop with Break**

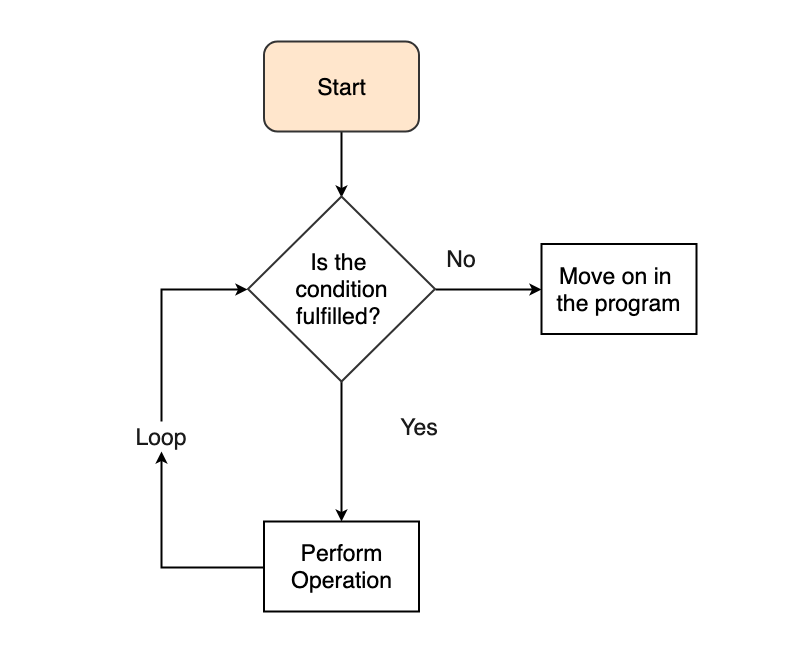
Ketika kondisi **break** adalah True, pernyataan **break** di-eksekusi dan keluar dari loop.



| i = 1 while i <= 100 :  print(i)  if i == 7 :  break  i += 1 |
| --- |

## For Loops

For loop di Python digunakan untuk mengulangi urutan (list, tuple, string) atau objek iterable lainnya. Iterasi terhadap sequence disebut traversal.



| for val in sequence:  loop body |
| --- |

Di sini, val adalah variabel yang mengambil nilai item di dalam urutan pada setiap iterasi.

Loop berlanjut sampai kita mencapai item terakhir dalam urutan.

| *# 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) |
| --- |

For loop dapat mengiterasi urutan data seperti string, list, set, atau tuple.

| if \_\_name\_\_ == "\_\_main\_\_":  s = "Hello!"   for c in s:  print(c) |
| --- |

### Python Loop List & Set

| H e l l o ! |
| --- |

| if \_\_name\_\_ == "\_\_main\_\_":  l = ['a', 'b', 'c']  s = {'a', 'b', 'c'}   print("list:")  for c in l:  print(c)   print("set:")  for c in s:  print(c) |
| --- |

| list: a b c set: c a b |
| --- |

### Python Loop Dictionary

Kita juga dapat mengiterasi sebuah dict Python. Menggunakan fungsi .keys() kita dapat mengiterasi nilai keys dari dist. Dan dengan fungsi .values() kita dapat mengiterasi semua nilai value dict.

| if \_\_name\_\_ == "\_\_main\_\_":  d = {'a': 0, 'b': 1, 'c': 2, 'd': 3}   print("keys:")  for k in d.keys():  print(k)   print("values:")  for v in d.values():  print(v)   print("key-value pairs:")  for k, v in d.items():  print(k, "->", v) |
| --- |

| keys a b c d values 0 1 2 3 key-value pairs a -> 0 b -> 1 c -> 2 d -> 3 |
| --- |

### 

### Range()

Kita dapat menghasilkan urutan angka menggunakan fungsi range(). range(10) akan menghasilkan angka dari 0 hingga 9 (10 angka).

Kita juga dapat mendefinisikan ukuran start, stop dan step sebagai range(start, stop,step\_size). Nilai step\_size adalah 1 jika tidak disediakan.

Objek range() bersifat "lazy" karena dia tidak akan menghasilkan output angka yang ada didalamnya ketika kita membuatnya.

Fungsi ini tidak menyimpan semua nilai dalam memori. Jadi dia hanya mengingat start, stop, step size dan menghasilkan angka berikutnya secara on the go.

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

| print(range(10)) print(list(range(10))) print(list(range(2, 8))) print(list(range(2, 20, 3))) |
| --- |

| range(0, 10) [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] [2, 3, 4, 5, 6, 7] [2, 5, 8, 11, 14, 17] |
| --- |

Kita dapat menggunakan fungsi range() di for loop untuk meng-iterasi melalui urutan angka. Dan dapat dikombinasikan dengan fungsi len() untuk meng-iterasi melalui urutan menggunakan pengindeksan.

| *# Iterate through a list using indexing*  genre = ['pop', 'rock', 'jazz']  *# iterate over the list using index* for i in range(len(genre)):  print("Saya menyukai", genre[i]) |
| --- |

| Saya menyukai pop Saya menyukai rock ​Saya menyukai jazz |
| --- |

### for loop dengan else

for loop juga dapat memiliki else secara opsional. Bagian else akan dijalankan jika item dalam urutan yang digunakan untuk loop telah habis.

Kata kunci break dapat digunakan untuk menghentikan for loop.

Bagian else dari for loop berjalan jika tidak ada break yang terjadi.

| digits = [0, 1, 5]  for i in digits:  print(i) else:  print("No items left.") |
| --- |

| 0 1 5 No items left. |
| --- |

Di sini, for loop mencetak item dari list hingga loop habis. Ketika for loop habis, ia mengeksekusi blok kode di else dan mencetak "No items left.".

Pernyataan for...else ini dapat digunakan dengan break untuk menjalankan blok else hanya ketika break tidak dieksekusi.

| *# to display student's marks from record* student\_name = 'Didu'  marks = {'Jono': 90, 'Jaspar': 55, 'Edward': 77}  for student in marks:  if student == student\_name:  print(marks[student])  break else:  print('No entry with that name found.') |
| --- |

| No entry with that name found. |
| --- |

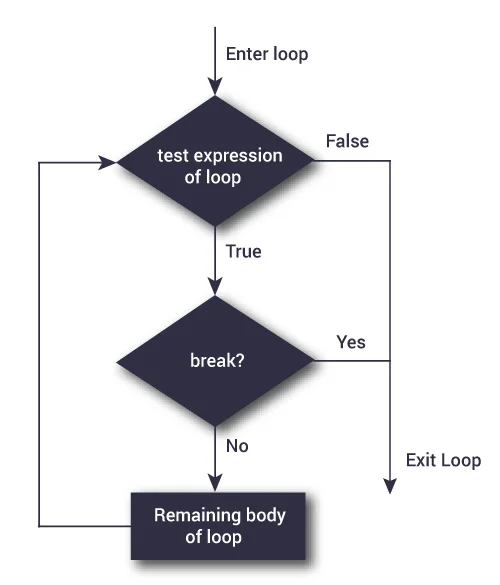
### Break & Continue

Pernyataan break dan continue dapat mengubah aliran loop.

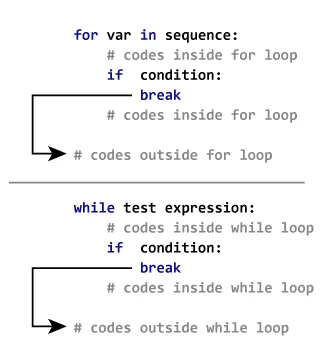
Loop mengulangi blok code hingga ekspresi pengujian menjadi False, tetapi terkadang kita ingin menghentikan iterasi saat ini juga atau bahkan seluruh loop tanpa memeriksa ekspresi pengujian yang lain.

Pernyataan break dan continue digunakan dalam kasus ini.

#### Python Break Statement

Pernyataan break digunakan untuk mengakhiri loop. 

Jika pernyataan break berada di dalam nested loop (loop di dalam loop lain), pernyataan break akan mengakhiri loop yang berada di paling dalam.

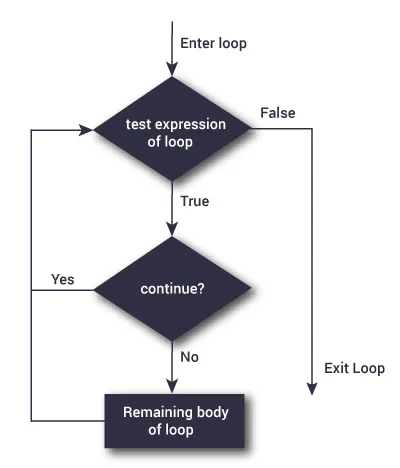


| *# Use of break statement inside the loop* for val in "string":  if val == "i":  break  print(val)  print("The end") |
| --- |

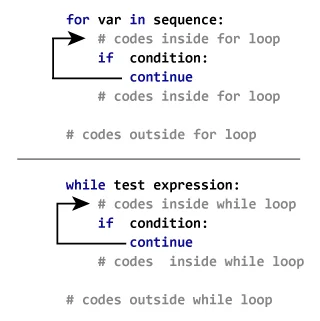
| s t r The end |
| --- |

#### Python Continue Statement

Pernyataan continue digunakan untuk melewati sisa kode di dalam satu loop untuk iterasi yang sedang dijalankan. Loop tidak berhenti tetapi berlanjut ke iterasi berikutnya.



Cara kerja dari continue pada perulangan for ditunjukkan di bawah ini.



| *# to show the use of continue statement inside loops*  for val in "string":  if val == "i":  continue  print(val)  print("The end") |
| --- |

| s t r n g The end |
| --- |

Program ini sama dengan contoh di atas kecuali pernyataan break telah diganti dengan continue. Jika string adalah i, maka code tidak akan dieksekusi. Oleh karena itu, kita melihat dalam output kita bahwa semua huruf kecuali i akan dicetak.

### Python Nested For Loops

Jika sebuah for loop berisi loop lain di dalamnya, maka ini dinamakan nested loop. Loop bagian luar dirancang untuk melakukan m iterasi dan loop bagian dalam dirancang untuk melakukan n iterasi, total loop akan dieksekusi sebanyak m \* n kali.

| for x in range(1,4):  for y in range(1,3):  print('x = ', x, ', y = ', y) |
| --- |

| x = 1, y = 1 x = 1, y = 2 x = 2, y = 1 x = 2, y = 2 x = 3, y = 1 x = 3, y = 2 |
| --- |

## Iterators

Iterator dalam Python adalah sebuah objek yang dapat diulang. Objek yang akan mengembalikan data, satu elemen pada satu waktu, satu per satu.

Secara teknis, objek iterator Python harus mengimplementasikan dua metode khusus, \_\_iter\_\_() dan \_\_next\_\_(), yang secara kolektif disebut protokol iterator. Suatu objek disebut iterable jika kita bisa diimplementasikan iterator. list, tuple, string etc. adalah iterable.

| *# define a list* my\_list = [4, 7, 0, 3]  *# get an iterator using iter()* my\_iter = iter(my\_list)  *# iterate through it using next()*  *# Output:* print(next(my\_iter)) print(next(my\_iter))  *# next(obj) is same as obj.\_\_next\_\_()*  *# Output:* print(my\_iter.\_\_next\_\_()) print(my\_iter.\_\_next\_\_())  *# This will raise error, no items left* next(my\_iter) |
| --- |

| 4 7 0 3 Traceback (most recent call last):  File "<string>", line 24, **in** <module>  next(my\_iter) StopIteration |
| --- |

### Iterator For Loop

For loop dapat mengiterasi suatu iterable.

| *# define a list* my\_list = [4, 7, 0, 3]  *# get an iterator using iter()* my\_iter = iter(my\_list)  while True:  try:  *# get the next item*  element = next(my\_iter)  *# do something with element*  except StopIteration:  *# if StopIteration is raised, break from loop*  break |
| --- |

## List Comprehension

Misalkan, kita ingin memisahkan huruf-huruf dari suatu kata dan membuatnya sebagai list. Kita dapat melakukannya dengan loop.

| h\_letters = []  for letter in 'apple':  h\_letters.append(letter)  print(h\_letters) |
| --- |

| ['a','p','p','l','e'] |
| --- |

Namun, Python memiliki cara yang lebih mudah untuk menyelesaikan masalah ini menggunakan List Comprehension.

| h\_letters = [letter for letter in 'apple'] print(h\_letters) |
| --- |

### Conditionals in List Comprehension

#### If with List Comprehension

For dengan if didalam loopnya seperti dibawah ini dapat disederhanakan menggunakan list comprehension:

| number\_list = [] for x in range(20):  if x % 2 == 0:  number\_list.append(x)  print(number\_list) |
| --- |

List comprehension ini akan menghasilkan output yang sama seperti kode diatas.

| number\_list = [ x for x in range(20) if x % 2 == 0] print(number\_list) |
| --- |

| [0, 2, 4, 6, 8, 10, 12, 14, 16, 18] |
| --- |

Nested If with List Comprehension

Nested if:

| num\_list = [] for y in range(100):  if y%2 == 0:  if y%5 == 0:  num\_list.append(y)  print(num\_list) |
| --- |

List Comprehension:

| num\_list = [y for y in range(100) if y % 2 == 0 if y % 5 == 0] print(num\_list) |
| --- |

| [0, 10, 20, 30, 40, 50, 60, 70, 80, 90] |
| --- |

#### if...else with List Comprehension

If…else

| obj = [] for i in range(10):  if i%2==0:  txt = "Even"  obj.append(txt)  else:  txt = "Odd"  obj.append(txt)  print(obj) |
| --- |

List Comprehension:

| obj = ["Even" if i%2==0 else "Odd" for i in range(10)] print(obj) |
| --- |

Output

| 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd'] |
| --- |

* List Comprehension adalah cara yang elegan untuk mendefinisikan dan membuat list berdasarkan list yang sudah ada.
* List Comprehension umumnya lebih ringkas dan lebih cepat daripada loop normal untuk membuat list.
* Namun, kita harus menghindari penulisan List Comprehension yang panjang dalam satu baris untuk memastikan bahwa kode tersebut mudah dibaca.
* Setiap List Comprehension dapat ditulis ulang dalam for loop, tetapi setiap for loop tidak dapat ditulis ulang dalam bentuk List Comprehension.