



Loops



Table of Contents



- ▶ Definitions
- ▶ while Loop
- ▶ for Loop
- ▶ Working with the Iterators
- ▶ Operations with for Loop
- ▶ Nested for Loop



Definitions

```
for i in iterator :  
    print(i)
```

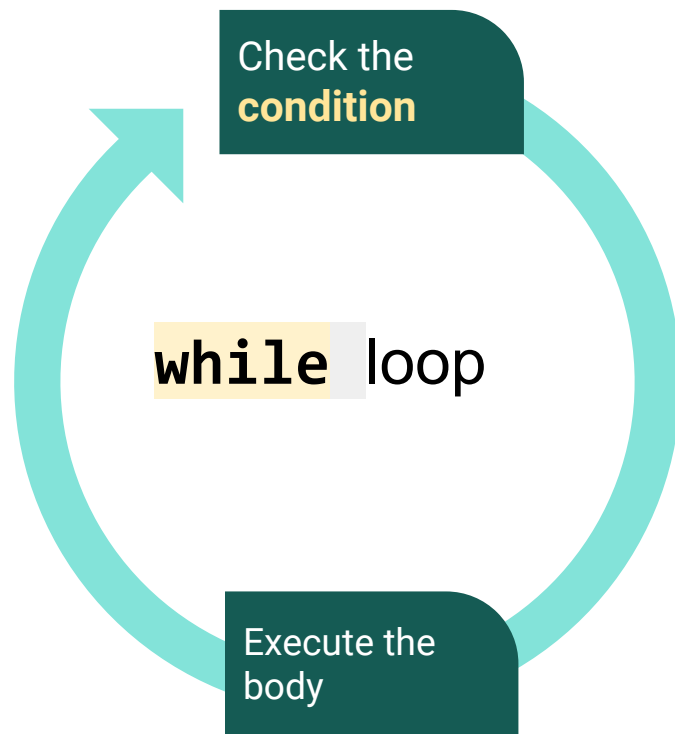
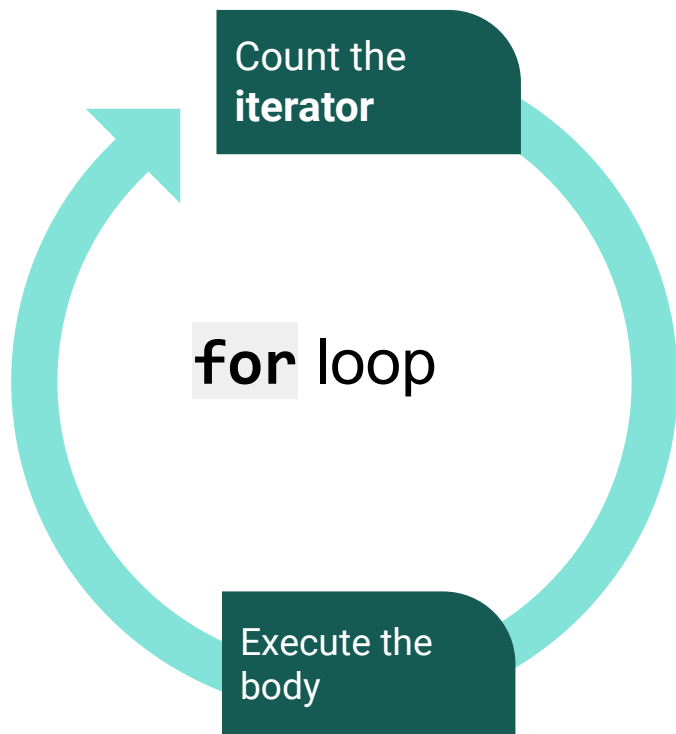
Can you explain the
importance and logic of
the loops?



Students, write your response!



Definitions - Loops





while Loop



while Loop (review the pre-class)

- ▶ The simple syntax 🖐️ of a `while` loop is :

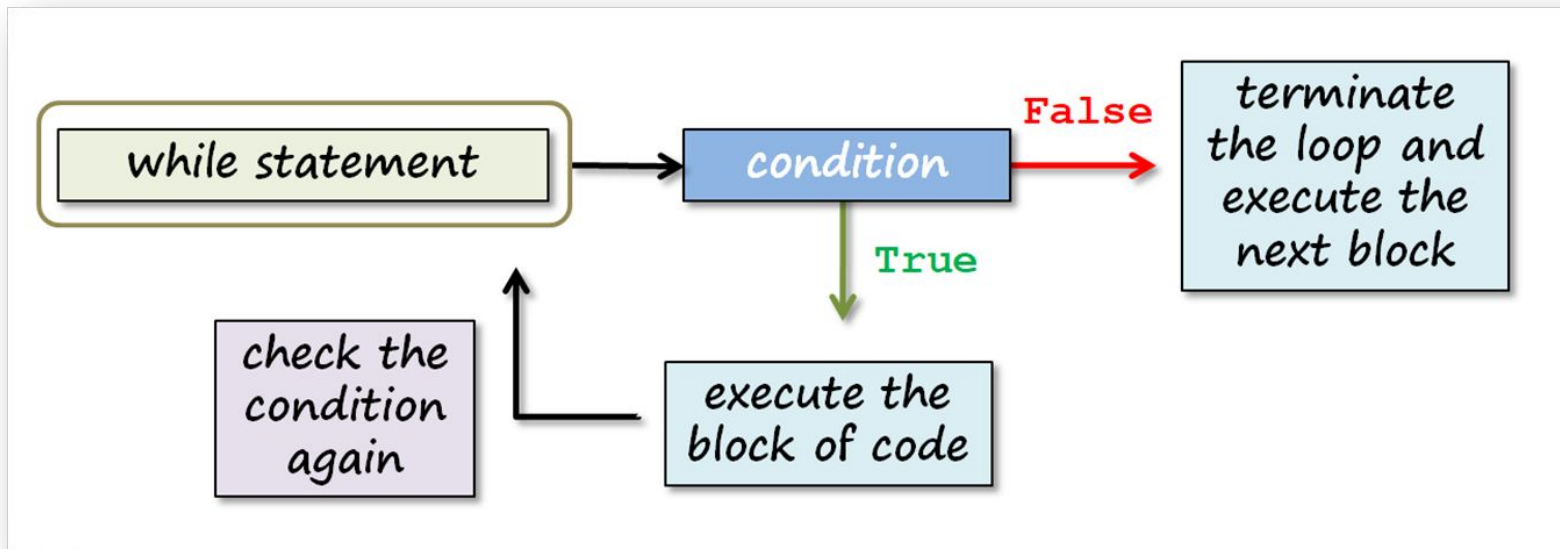
A diagram illustrating the syntax of a `while` loop. The code is shown in a light gray box: `while condition:` on the first line and `body` on the second line, indented. A red arrow points from the text "A 4-space indentation" to the indentation of the `body` line. Another red arrow points from the text "A colon" to the colon at the end of the `while condition:` line.

```
while condition:
    body
```



while Loop (review the pre-class)

- ▶ The basic diagram 📍 of a `while` loop works as follows :





while Loop (review the pre-class)

- ▶ Let's take a look at the first `while` loop in the pre-class content :

```
1 number = 0
2
3 while number < 6:
4     print(number)
5     number += 1
6 print('now, number is bigger or equal to 6')
7
```



while Loop (review the pre-class)

- The output :

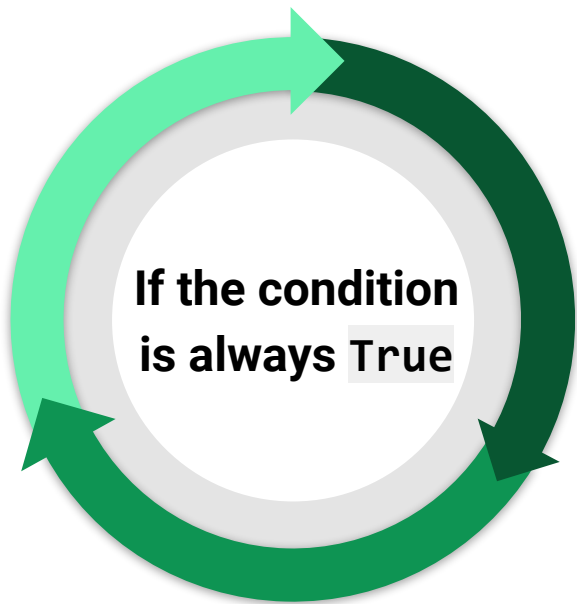
```
1 number = 0
2
3 while number < 6:
4     print(number)
5     number += 1
6 print('now, number is bigger or equal to 6')
7
```

```
1 0
2 1
3 2
4 3
5 4
6 5
7 now, number is bigger or equal to 6
8
```



while Loop

- ▶ Pay attention not to start an infinite loop.





while Loop (review the pre-class)

- ▶ We can use an iterator object in a `while` loop. Let's call a `list` in this example :

```
1 my_list=["a", "b", "c", "d", "e"]
2
3 a = 0
4
5 while a < len(my_list):
6     print('square of {} is : {}'.format(a, a**2))
7     a+=1
8
```



while Loop (review the pre-class)

- ▶ The output :

```
1 my_list=["a", "b", "c", "d", "e"]
2
3 a = 0
4
5 while a < len(my_list):
6     print('square of {} is : {}'.format(a, a**2))
7     a+=1
8
```

```
1 square of 0 is : 0
2 square of 1 is : 1
3 square of 2 is : 4
4 square of 3 is : 9
5 square of 4 is : 16
6
```



while Loop (review the pre-class)

- ▶ **Task:** Take the age of the user using `input()` and `while` loop.
 - ▷ Write a program that ;
 - ▷ Takes the age from user,
 - ▷ Check the age if it is correct numeric format.



while Loop (review the pre-class)

- ▶ The code can be like :

```
1 age = input("Enter your age please : ")
2
3 while not age.isdigit():
4     print ("You entered incorrectly!")
5     age = input("Enter your age please : ")
6
7 print("Great! You enter valid input : ", age)
8
```



▶ while Loop (review the pre-class)

▶ Task:

Let's play famous 'guessing a number game' using `while` loop.

- ▶ Write a program that ;
 - ▶ Takes the numbers from user,
 - ▶ Compares the number the user entered with the number you assigned and then gives a message “**Little lower**” or “**Little higher**” till the user knows it.



while Loop (review the pre-class)

- ▶ The code can be like : In this case, we are trying to find number 28.

```
1 answer = 28
2
3 question = 'What a two-digit number am I thinking of? '
4 print ("Let's play the guessing game!")
5
6 while True:
7     guess = int(input(question))
8
9     if guess < answer:
10         print('Little higher')
11     elif guess > answer:
12         print('Little lower')
13     else: # guess == answer
14         print('Are you a MINDREADER!!!')
15         break
16
```



while Loop

- ▶ Lastly, let's play famous 'mind reader' game' using **while** loop.

We have written a program that does not exit the **while** loop until you find the correct number.

```
1 answer = 28
2
3 question = 'What a two-digit number am I thinking of? '
4 print ("Let's play the guessing game!")
5
6 while True:
7     guess = int(input(question))
8
9     if guess < answer:
10         print('Little higher')
11     elif guess > answer:
12         print('Little lower')
13     else: # guess == answer
14         print('Are you a MINDREADER!!!')
15         break
16
```

When the user knows the answer (28) and enters **input**, it takes the value of 28 and assigns to variable **guess**, in the end, **else** works and breaks the loop.

We used **break** keyword in order to quit and exit the **while** loop.



while Loop

► Task:

Find and print the length of the **longest word**.

- ▷ Write a program that ;
 - ▷ Takes a **string sentence** consisting of a couple of words from the user,
 - ▷ Compares and find out the **longest** word and **prints** the whole sentence and the **length** of the longest word as **int** type.
 - ▷ Use **while** loop.



while Loop

- ▶ The code can be like :

```
1 sentence = input("Give me a sentence :")
2 words = sentence.split()
3 i = 0
4 longest = 0
5 while i < len(words) :
6     if len(words[i]) > longest:
7         longest = len(words[i])
8     i += 1
9 print("the length of the longest word :", longest)
10
11
```



for Loop

the world and
even life itself are
nothing but loops.



for Loop (review the pre-class)

- ▶ The simple syntax 🖱️ of a `for` loop is :

A 4-space indentation

```
for variable in iterable:  
    body
```

A colon



for Loop (review the pre-class)

- ▶ To create a **for** loop, you need a variable and an iterable object.
- ▶ Let's examine the subject through an example :

```
1 for i in [1, 2, 3, 4, 5] :  
2     print(i)  
3
```



for Loop (review the pre-class)

- ▶ To create a **for** loop, you need a variable and an iterable object.
- ▶ Let's examine the subject through an example :

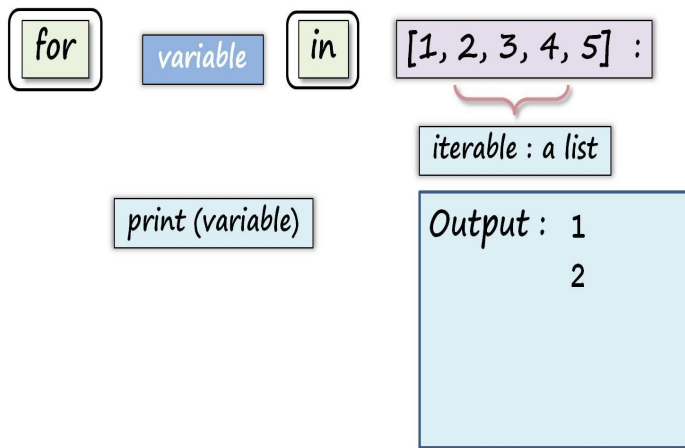
```
1 for i in [1, 2, 3, 4, 5] :  
2     print(i)  
3
```

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




for Loop (review the pre-class)

- ▶ You can follow the animated diagram of this `for` loop for a better understanding.





for Loop (review the pre-class)

- ▶ Another example :

```
1 seasons = ['spring', 'summer', 'autumn', 'winter']  
2  
3 for season in seasons :  
4     print(season)  
5
```



for Loop (review the pre-class)

- ▶ In the structure of the `for` loop, you can use also a variable as an iterable.
- ▶ Let's see it in an example :

```
1 seasons = ['spring', 'summer', 'autumn', 'winter']  
2  
3 for season in seasons :  
4     print(season)  
5
```

```
1 spring  
2 summer  
3 autumn  
4 winter  
5
```



for Loop

- ▶ In the structure of the `for` loop, you can use also a variable as an iterable.
- ▶ Let's see it in an example :

```
1 seasons = ['spring', 'summer', 'autumn', 'winter']  
2  
3 for season in seasons :  
4     print(season)  
5
```

```
1 spring  
2 summer  
3 autumn  
4 winter  
5
```



for Loop

- ▶ In the structure of the `for` loop, you can use also a variable as an iterable.
- ▶ Let's see it in an example :

```
1 seasons = ['spring', 'summer', 'autumn', 'winter']  
2  
3 for season in seasons :  
4     print(season)  
5
```

```
1 spring  
2 summer  
3 autumn  
4 winter  
5
```



for Loop

- ▶ In the structure of the `for` loop, you can use also a variable as an iterable.
- ▶ Let's see it in an example :

```
1 seasons = ['spring', 'summer', 'autumn', 'winter']  
2  
3 for season in seasons :  
4     print(season)  
5
```

```
1 spring  
2 summer  
3 autumn  
4 winter  
5
```



for Loop

- ▶ In the structure of the `for` loop, you can use also a variable as an iterable.
- ▶ Let's see it in an example :

```
1 seasons = ['spring', 'summer', 'autumn', 'winter']  
2  
3 for season in seasons :  
4     print(season)  
5
```

```
1 spring  
2 summer  
3 autumn  
4 winter  
5
```



for Loop



► Task : Python Program to say “hello name”

- ▷ Write a program to say “hello names” from the following list.
- ▷ Print the result such as : “hello Samuel”

“hello Victor”

```
names = ["Ahmed", "Aisha", "Adam", "Joseph", "Gabriel"]
```




for Loop

- The code might be like :

```
1 names = ["Ahmed", "Aisha", "Adam", "Joseph", "Gabriel"]
2
3 for i in names:
4     print("hello", i)
5
```

Output

```
hello Ahmed
hello Aisha
hello Adam
hello Joseph
hello Gabriel
```



for Loop



- ▶ **Task : Python Program to create numbers using range()**
 - ▷ Write a program to create a **list** consisting of numbers from **1** to **5**.
 - ▷ Print the result such as : `[1, 2, 3, 4, 5]`

for Loop



- The code might be like :

```
1 numbers = []  
2  
3 for i in range(1, 6):  
4     numbers.append(i)  
5 print(numbers)  
6
```

Output

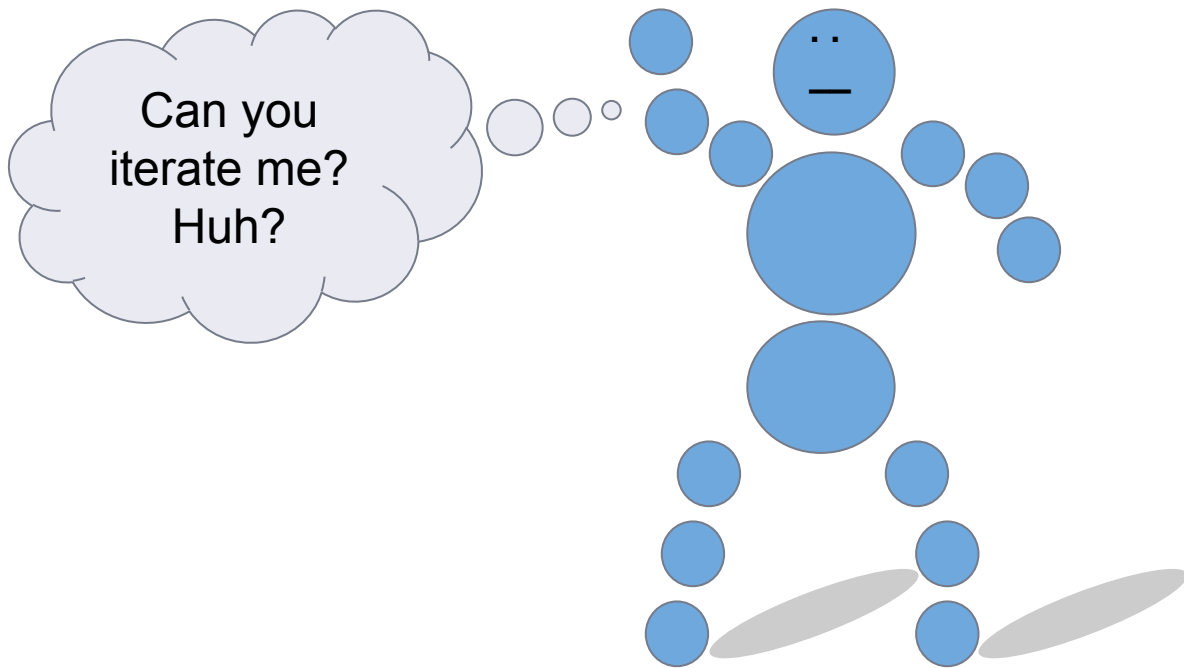
```
[1, 2, 3, 4, 5]
```



Working with the Iterators



Working with the Iterators

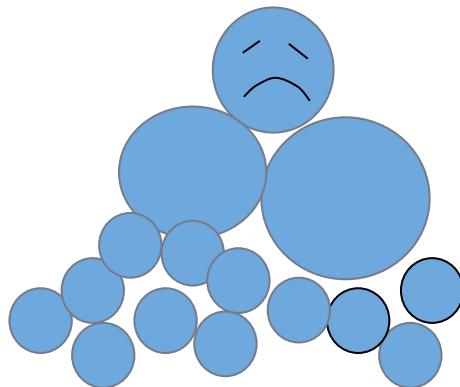




Working with the Iterators (review)

- ▶ The most common iterable types :

- `str`
- `list`
- `tuple`
- `dict`
- `set`





Working with the Iterators (review)

- ▶ Consider this example of a `str` type.

```
1 course = 'clarusway'
2
3 for i in course :
4     print(i)
5
```



Working with the Iterators (review)

- Consider this example of a `str` type.

```
1 course = 'clarusway'  
2  
3 for i in course :  
4     print(i)  
5
```

```
1 c  
2 l  
3 a  
4 r  
5 u  
6 s  
7 w  
8 a  
9 y  
10
```




Working with the Iterators

- ▶ **Task : Python Program to separate the string into its characters.**
 - ▷ Write a program to separate the string taken from the user into its characters using **for** loop.
 - ▷ Print the result such as :

```
input : "Clarusway"  
desired output : c-l-a-r-u-s-w-a-y
```



Working with the Iterators

- **The code might be like :**

```
1 word = input("Give me a word :")
2 count = 0
3 for i in word:
4     count += 1
5     if count < len(word) :
6         i = i + "-"
7     print(i, end="")
8
9
```

input: "Clarusway"

Output

```
c-l-a-r-u-s-w-a-y
```



Working with the Iterators

- ▶ **Let's solve this without using a loop :**

input: "Clarusway"

```
string = "clarusway"  
print("-".join(string))
```

Output

```
c-l-a-r-u-s-w-a-y
```



Working with the Iterators

- Take a look at the other iterable type : dict.

```
1 user = {  
2     "name": "Daniel",  
3     "surname": "Smith",  
4     "age": 35  
5 }  
6  
7 for attribute in user:  
8     print(attribute)  
9
```

What is the output? Try to figure out in your mind...





Working with the Iterators

► The output :

```
1 user = {  
2     "name": "Daniel",  
3     "surname": "Smith",  
4     "age": 35  
5 }  
6  
7 for attribute in user:  
8     print(attribute)  
9
```



Output

```
name  
surname  
age
```



Working with the Iterators

- Take a look at the other iterable type : dict.

```
1 user = {  
2     "name": "Daniel",  
3     "surname": "Smith",  
4     "age": 35  
5 }  
6  
7 for i in user.values():  
8  
9     print (i, end=" ")  
10
```

What is the output? Try to figure out in your mind...





Working with the Iterators

► The output :

```
1 user = {  
2     "name": "Daniel",  
3     "surname": "Smith",  
4     "age": 35  
5 }  
6  
7 for i in user.values():  
8  
9     print (i, end=" ")  
10
```

Output

```
Daniel Smith 35
```



Working with the Iterators

- Take a look at the other iterable type : dict.

```
1 user = {  
2     "name": "Daniel",  
3     "surname": "Smith",  
4     "age": 35  
5 }  
6  
7 for key, value in user.items():  
8     print (key, ":", value)  
9
```

What is the output? Try to figure out in your mind...





Working with the Iterators (review)

► The output :

```
1 user = {  
2     "name": "Daniel",  
3     "surname": "Smith",  
4     "age": 35  
5 }  
6  
7 for key, value in user.items():  
8     print (key, ":", value)  
9
```

Output

```
name : Daniel  
surname : Smith  
age : 35
```