

Python Tutorial

1. Data types

Example	Data Type
<code>x = "Hello World"</code>	str
<code>x = 20</code>	int
<code>x = 20.5</code>	float
<code>x = 1j</code>	complex
<code>x = ["apple", "banana", "cherry"]</code>	list
<code>x = ("apple", "banana", "cherry")</code>	tuple
<code>x = range(6)</code>	range
<code>x = {"name" : "John", "age" : 36}</code>	dict
<code>x = {"apple", "banana", "cherry"}</code>	set
<code>x = frozenset({"apple", "banana", "cherry"})</code>	frozenset
<code>x = True</code>	bool
<code>x = b"Hello"</code>	bytes
<code>x = bytearray(5)</code>	bytearray
<code>x = memoryview(bytes(5))</code>	memoryview

2. Numbers and Casting

Numbers:

Int : 1, 2, 3, 4, 5, ...

Float : 1.5, 2.5, 3.5, 4.5, ...

Complex : 1j, 1+1j, 4+7j, ...

Example :

```
x = 1000
y = 13.5987
z = 4 + 7j

print (type(x))
print (type(y))
print (type(z))
```

Casting :

Int () - constructs an integer number from an integer literal

Float () - constructs a float number from an integer literal

Str () - constructs a string from a wide variety of data types, including strings, integer literals and float literals

Example:

```
a = int (1) → a will be 1
```

```
b = int (15.8) → b will be 15
```

```
c = float (6) → c will be 6.0
```

```
d = float (12.3) → d will be 12.3
```

```
e = str ("3.9") → e will be 3.9
```

```
f = str ("123ABC") → f will be 123ABC
```

```
print(a)
```

```
print(b)
```

```
print(c)
```

```
print(d)
```

```
print(e)
```

```
print(f)
```

3. String

```
a = " Xin chao, toi ten la Hua Ngoc Truc Tien "
```

```
a = ' Xin chao, toi ten la Hua Ngoc Truc Tien '
```

```
print (a)
```

```
print (b)
```

4. If ... elif ... else

Python supports the usual logical conditions from mathematics:

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`

An "if statement" is written by using the if keyword.

Example 1:

```
n = int(input("Enter n: "))

if (n > 0) :
    print(n, "+", 1, "=", n+1)
    print(n + 1)
```

Example 2:

```
a = 200
b = 33
if b > a:
    print("b is greater than a")
elif a == b:
    print("a and b are equal")
else:
    print("a is greater than b")
```

5. FOR

for <Variable Name> in <List, Tuple, Set etc>:
 <statement>

Example 1: Print each fruit in a fruit list

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
```

Example 2: Loop through the letters in the word "banana"

```
for x in "banana":
    print(x)
```

Example 3 : With the **break** statement we can stop the loop before it has looped through all the items

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    print(x)
    if x == "banana":
        break
```

Example 4 : Exit the loop when x is "banana", but this time the **break** comes before the **print**

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    if x == "banana":
        break
    print(x)
```

Example 5 : Do not print banana

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

6. FOR - range()

The range(start, stop,step_size) function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

Range (start, stop,step_size)

for <Variable Name> in range (start, stop,step_size)

Example 1 :

```
for x in range(6):  
    print(x)  
  
## range(6) is not the values of 0 to 6, but the values 0 to 5 ##
```

Example 2 :

```
for x in range(2, 30):  
    print(x)
```

Example 3 :

```
for x in range(1, 30, 3):  
    print(x)
```

7. Else - Break - Nested - Pass in For Loop

Example 1 : The `else` keyword in a `for` loop specifies a block of code to be executed when the loop is finished

```
for x in range(6):  
    print(x)  
else:  
    print("Finally finished!")
```

Example 2 : If the loop breaks, the `else` block is not executed.

```
for x in range(6):  
    if x == 3: break  
    print(x)  
else:  
    print("Finally finished!")
```

Example 3: A nested loop is a loop inside a loop. The "inner loop" will be executed one time for each iteration of the "outer loop"

```
colors = ["red", "yellow", "black"]  
fruits = ["apple", "banana", "cherry"]  
  
for x in colors:  
    for y in fruits:  
        print(x, y)
```

Example 4 : `for` loops cannot be empty, but if you for some reason have a `for` loop with no content, put in the `pass` statement to avoid getting an error.

```
for x in [0, 1, 2]:  
    pass  
# having an empty for loop like this, would raise an error without the  
pass statement
```

8. While loops

With the while loops we can execute a set of statements as long as a condition is true.

The while loops is also quite similar to the for loops

Note : With while loops you need to be careful. Because if you are careless in the programming process, it will lead to an infinite loop

```
while <condition> :  
    <command block>
```

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`

Example 1 : Print i as long as i is less than 6

```
i = 1  
while i < 6:  
    print(i)  
    i += 1
```

Example 2 : Sum of positive integers less than 8

```
n = 0  
sum = 0  
while n < 8:  
    sum = sum + n  
    n = n + 1  
print("Sum of numbers less than 8 is : ", sum)
```

9. Else - break - continue - pass in while loops

Example 1 : With the `else` statement we can run a block of code once when the condition no longer `is` true

```
i = 1
while i < 20:
    print(i)
    i += 1
else:
    print("i is no longer less than 20")
```

Example 2 : With the `break` statement we can stop the loop even `if` the `while` condition `is` true

```
i = 1
while i < 6:
    print(i)
    if i == 3:
        break
    i += 1
```

Example 3 : With the `continue` statement we can stop the current iteration, and `continue` with the `next`

```
i = 0
while i < 6:
    i += 1
    if i == 3:
        continue
    print(i)
```

Example 4 : `while` loops cannot be empty, but `if` you `for` some reason have a `while` loops with no content, put `in` the `pass` statement to avoid getting an error.

```
number = 0
for number in range(10):
    if number == 5:
        pass # pass here
    print('Number is ' + str(number))
print('Out of loop')
```