Python Tutorial

1. Data types

Example	Data Type
`x = "Hello World"`	str
`x = 20`	int
`x = 20.5`	float
`x = 1j`	complex
`x = ["apple", "banana", "cherry"]`	list
`x = ("apple", "banana", "cherry")`	tuple
`x = range(6)`	range
`x = {"name" : "John", "age" : 36}`	dict
`x = {"apple", "banana", "cherry"}`	set
`x = frozenset({"apple", "banana", "cherry"})`	frozenset
`x = True`	bool
`x = b"Hello"`	bytes
`x = bytearray(5)`	bytearray
`x = memoryview(bytes(5))`	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) → b will be 6.0

d = float (12.3) → c will be 12.3

e = str (“3.9”) → d will be 3.9

f = str (“123ABC”) → e 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**

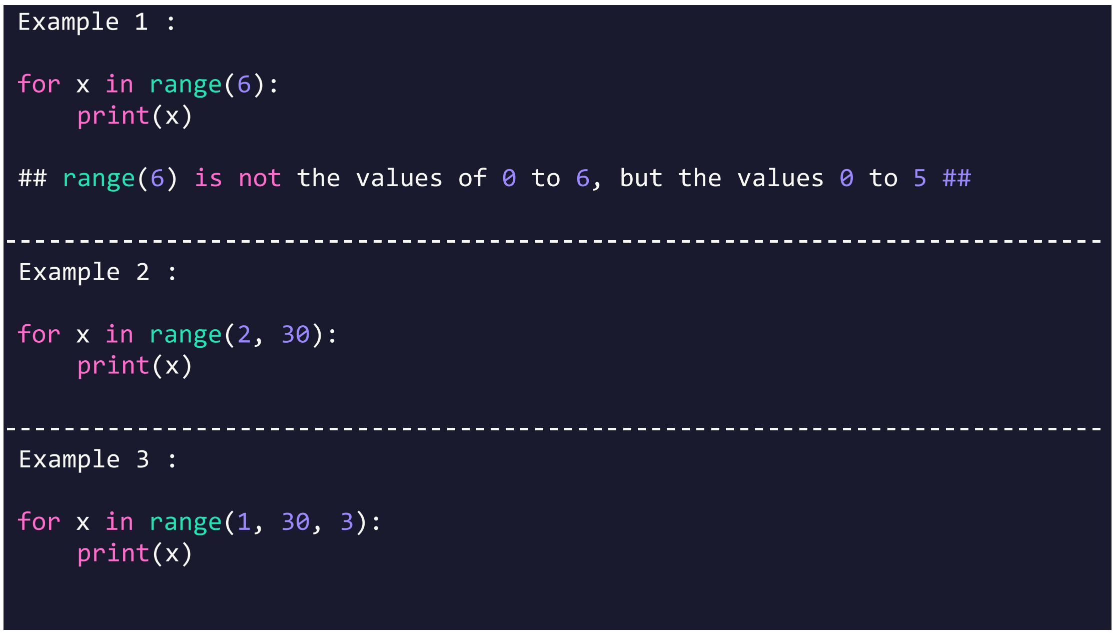
```
for <Variable Name> in <List, Tuple, Set etc>:
      <statement>
```

```
 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
```