Contents

..Thuật toán
Python
Framework
Machine learning
Deep learning
Mấy cái sota
Linear algebra
Statistic & probability
Pytorch, sklearn, caffe, tensorflow, keras

Python Tutorial BASIC

Python Getting Started

```
#!/bin/python3
print("Hello, World!")
```

Python Syntax

```
 if 5 > 2:
 print("Five is greater than two!")
if 5 > 2:
        print("Five is greater than two!")
x = 5
y = "Hello, World!"
print(x)
print(y)
#Five is greater than two!
#Five is greater than two!
#5
#Hello, World!
```

Python Comments

```
#This is a comment.
print("Hello, World!")
"""
This is a comment
written in
more than just one line
"""
print("Hello, World!")
```

```
Hello, World!
Hello, World!
```

Python - Variable Exercises

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
print(x,y,z)
x = 5
y = "John"
print(type(x),type(y))
x, y, z = "Orange", "Banana", "Cherry"
print(x,y,z)
x = "awesome"
print("Python is " + x)
```

```
3 3 3.0
<class 'int'> <class 'str'>
Orange Banana Cherry
Python is awesome
```

Python Data Types

Text Type:	str
Numeric Types:	int, float, complex

Sequence Types:	list, tuple, range
Mapping Type:	dict
Set Types:	set, frozenset
Boolean Type:	bool
Binary Types:	bytes, bytearray, memoryview

```
x = ["apple", "banana", "cherry"]
#display x:
print(x)
#display the data type of x:
print(type(x))
```

```
['apple', 'banana', 'cherry']
<class 'list'>
```

Python Numbers

```
#convert from int to float:
x = float(1)
#convert from float to int:
y = int(2.8)
#convert from int to complex:
z = complex(x)
print(x,y,z)
print(type(x),type(y),type(z))
```

```
1.0 2 (1+0j)
<class 'float'> <class 'int'> <class 'complex'>
```

Python Casting

```
x = int(1)
y = int(2.8)
z = int("3")
print(x,y,z)
x = str("s1")
y = str(2)
z = str(3.0)
print(x,y,z)
```

```
1 2 3
s1 2 3.0
```

Python Strings

Basic:

```
a = "Hello"
print(a)
print(len(a))
print(a[1])
for x in "banana":
  print(x,end=' ')
print()
a = """Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua."""
print(a)
if "Lorem" in a:
  print("Yes, 'free' is present.")
```

```
Hello
5
e
b a n a n a
Lorem ipsum dolor sit amet,
consectetur adipiscing elit,
sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua.
Yes, 'free' is present.
```

Slicing Strings

```
b = "Hello, World!"
print(b[2:5])
print(b[:5])
print(b[2:])
print(b[-5:-2])
```

```
llo
Hello
llo, World!
orl
```

Python - **Modify Strings**

```
a = " Hello, World! "
print(a.upper())
print(a.lower())
print(a.strip()) #Remove Whitespace
print(a.split(","))
```

```
HELLO, WORLD!
 hello, world!
Hello, World!
[' Hello', ' World! ']
```

```
txt = "Hello my friends"
txt.upper()
print(txt)
txt=txt.upper()
print(txt)
```

```
Hello my friends
HELLO MY FRIENDS
```

String Format

```
age = 36
txt = "My name is John, and I am {}"
```

```
print(txt.format(age))
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {}
dollars."
print(myorder.format(quantity, itemno, price))
myorder = "I want to pay {2} dollars for {0} pieces o
item {1}."
print(myorder.format(quantity, itemno, price))
```

```
My name is John, and I am 36
I want 3 pieces of item 567 for 49.95 dollars.
I want to pay 49.95 dollars for 3 pieces of item 567.
```

String Methods

Python has a set of built-in methods that you can use on strings.

Note: All string methods returns new values. They do not change the original string.

Method	**Description**
capitalize()	Converts the first character to upper case
casefold()	Converts string into lower case
center()	Returns a centered string
count()	Returns the number of times a specified value occurs in a string
encode()	Returns an encoded version of the string
endswith()	Returns true if the string ends with the specified value
expandtabs()	Sets the tab size of the string
find()	Searches the string for a specified value and returns the position of where it was found
format()	Formats specified values in a string
format_map()	Formats specified values in a string
index()	Searches the string for a specified value and returns the position of where it was found
isalnum()	Returns True if all characters in the string are alphanumeric
isalpha()	Returns True if all characters in the string are in the alphabet
isdecimal()	Returns True if all characters in the string are decimals
isdigit()	Returns True if all characters in the string are digits
isidentifier()	Returns True if the string is an identifier
islower()	Returns True if all characters in the string are lower case
isnumeric()	Returns True if all characters in the string are numeric
isprintable()	Returns True if all characters in the string are printable
isspace()	Returns True if all characters in the string are whitespaces
istitle()	Returns True if the string follows the rules of a title
isupper()	Returns True if all characters in the string are upper case
join()	Joins the elements of an iterable to the end of the string
ljust()	Returns a left justified version of the string
lower()	Converts a string into lower case
lstrip()	Returns a left trim version of the string
maketrans()	Returns a translation table to be used in translations
partition()	Returns a tuple where the string is parted into three parts
replace()	Returns a string where a specified value is replaced with a specified value
rfind()	Searches the string for a specified

	value and returns the last position of where it was found
rindex()	Searches the string for a specified value and returns the last position of where it was found
rjust()	Returns a right justified version of the string
rpartition()	Returns a tuple where the string is parted into three parts
rsplit()	Splits the string at the specified separator, and returns a list
rstrip()	Returns a right trim version of the string
split()	Splits the string at the specified separator, and returns a list
splitlines()	Splits the string at line breaks and returns a list
startswith()	Returns true if the string starts with the specified value
strip()	Returns a trimmed version of the string
swapcase()	Swaps cases, lower case becomes upper case and vice versa
title()	Converts the first character of each word to upper case
translate()	Returns a translated string
upper()	Converts a string into upper case
zfill()	Fills the string with a specified number of 0 values at the beginning

Python Booleans

```
print(10 > 9)
print(10 == 9)
print(10 < 9)
print(bool("Hello"))
print(bool(15))
print(bool(False))
print(bool(None))
print(bool(0))
print(bool(""))
print(bool(()))
print(bool([]))
print(bool({}))
def myFunction() :
  return True
print("end ",myFunction())
```

```
True
False
False
True
True
False
False
False
False
False
False
False
end  True
```

Bộ sưu tập Python (Mảng)
Có bốn kiểu dữ liệu thu thập trong ngôn ngữ lập trình Python:

- Danh sách là một tập hợp được sắp xếp và có thể thay đổi. Cho phép các thành viên trùng lặp.
- **Tuple** là một bộ sưu tập có thứ tự và không thể thay đổi. Cho phép các thành viên trùng lặp.
- **Tập hợp** là một tập hợp không có thứ tự và không được lập chỉ mục. Không có thành viên trùng lặp.
- **Từ điển** là một bộ sưu tập không có thứ tự và có thể thay đổi. Không có thành viên trùng lặp.

Khi chọn một kiểu tập hợp, sẽ rất hữu ích khi hiểu các thuộc tính của kiểu đó. Chọn loại phù hợp cho một tập dữ liệu cụ thể có thể có nghĩa là duy trì ý nghĩa và, nó có thể có nghĩa là tăng hiệu quả hoặc bảo mật.

Python Lists

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
print(len(thislist))
print(type(thislist))

list1 = ["abc", 34, True, 40, "male"]
print(list1)
print(type(list1))
```

```
['apple', 'banana', 'cherry']
3
```

```
<class 'list'>
['abc', 34, True, 40, 'male']
<class 'list'>
```

Python - Access List Items

```
thislist = ["apple", "banana", "cherry", "orange",
"kiwi", "melon", "mango"]
print(thislist[1])
print(thislist[-1])
print(thislist[2:5])
print(thislist[:4])
print(thislist[2:])
print(thislist[-4:-1])
if "apple" in thislist:
  print("Yes, 'apple' is in the fruits list")
```

```
banana
mango
['cherry', 'orange', 'kiwi']
['apple', 'banana', 'cherry', 'orange']
['cherry', 'orange', 'kiwi', 'melon', 'mango']
['orange', 'kiwi', 'melon']
Yes, 'apple' is in the fruits list
```

Python - **add List Items**

```
thislist = ["apple", "banana", "cherry"]
thislist[1] = "watermelon"
print(thislist)
thislist.insert(2, "kiwi")
print(thislist)
thislist.append("orange")
print(thislist)
```

```
['apple', 'watermelon', 'cherry']
['apple', 'watermelon', 'kiwi', 'cherry']
['apple', 'watermelon', 'kiwi', 'cherry', 'orange']
```

Python - Remove List Items

```
thislist = ["apple", "banana",
"cherry","union","peach"]
thislist.remove("banana")
print(thislist)
thislist.pop(1)
print(thislist)
del thislist[1]
print(thislist)
thislist.pop()
print(thislist)
thislist.clear()
print(thislist)
del thislist
#print(len(thislist)) : #this will cause an error because
you have succsesfully deleted "thislist".
```

```
['apple', 'cherry', 'union', 'peach']
['apple', 'union', 'peach']
['apple', 'peach']
['apple']
[]
```

Python - Loop Lists

```
thislist = ["apple", "banana", "cherry"]
for x in thislist:
  print(x,end=' ')
print()
for i in range(len(thislist)):
  print(thislist[i],end=' ')
print()
i = 0
while i < len(thislist):
  print(thislist[i],end=' ')
  i = i + 1
print()
[print(x,end=' ') for x in thislist]
```

```
apple banana cherry
apple banana cherry
apple banana cherry
apple banana cherry
```

Python - **Sort Lists**

```
thislist = ["orange", "mango", "kiwi", "pineapple",
"banana"]
thislist.sort()
print(thislist)
thislist.sort(reverse = True)
print(thislist)

thislist = [100, 50, 65, 82, 23]
thislist.sort()
print(thislist)
def myfunc(n):
  return abs(n - 50)
thislist.sort(key = myfunc)
print(thislist)

thislist = ["banana", "Orange", "Kiwi", "cherry"]
thislist.sort()
print(thislist)
thislist.sort(key = str.lower)
print(thislist)
thislist.reverse()
print(thislist)
```

```
['banana', 'kiwi', 'mango', 'orange', 'pineapple']
['pineapple', 'orange', 'mango', 'kiwi', 'banana']
[23, 50, 65, 82, 100]
[50, 65, 23, 82, 100]
['Kiwi', 'Orange', 'banana', 'cherry']
['banana', 'cherry', 'Kiwi', 'Orange']
['Orange', 'Kiwi', 'cherry', 'banana']
```

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist.copy()
print(mylist)
mylist = list(thislist)
print(mylist)
['apple', 'banana', 'cherry']
['apple', 'banana', 'cherry']
```

Python - List Methods

```
list1 = ["a", "b", "c"]
list2 = [1, 2, 3]
list3 = list1 + list2
print(list3)
list3=list1
for x in list2:
  list3.append(x)
print(list3)
['a', 'b', 'c', 1, 2, 3]
['a', 'b', 'c', 1, 2, 3]
```

List Methods

Python has a set of built-in methods that you can use on lists.

Method	**Description**
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Python Tuples

Tuples are used to store multiple items in a single variable.
A tuple is a collection which is ordered and unchangeable.
Tuples are written with round brackets.

Tuples được sử dụng để lưu trữ nhiều mục trong một biến duy nhất.
Bộ tuple là một bộ sưu tập được sắp xếp theo thứ tự và không thể thay đổi .
Tuples được viết bằng dấu ngoặc tròn.
basic

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
print(len(thistuple))
('apple', 'banana', 'cherry')
3
```

```
thistuple = ("apple",)#you have to add a comma
print(type(thistuple))
#NOT a tuple
thistuple = ("apple")
print(type(thistuple))
```

```
tuple1 = ("abc", 34, True, 40, "male")
print(tuple1)
<class 'tuple'>
<class 'str'>
('abc', 34, True, 40, 'male')
```

Python - Access Tuple Items

```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[1])
print(thistuple[-1])
print(thistuple[2:5])
print(thistuple[:4])
print(thistuple[2:])
print(thistuple[-4:-1])
if "apple" in thistuple:
  print("Yes, 'apple' is in the fruits tuple")

banana
mango
('cherry', 'orange', 'kiwi')
('apple', 'banana', 'cherry', 'orange')
('cherry', 'orange', 'kiwi', 'melon', 'mango')
('orange', 'kiwi', 'melon')
Yes, 'apple' is in the fruits tuple
```

Python - Update Tuples

Once a tuple is created, you cannot change its values. Tuples are unchangeable, or immutable as it also is called.

But there is a workaround. You can convert the tuple into a list, change the list, and convert the list back into a tuple.

```
thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.append("orange")
thistuple = tuple(y)
print(thistuple)
#Remove Items : same
('apple', 'banana', 'cherry', 'orange')

fruits = ("apple", "banana", "cherry")
(green, yellow, red) = fruits
print(green,yellow,red)
fruits = ("apple", "banana", "cherry", "strawberry",
"raspberry")
(green, yellow, *red) = fruits
print(green,yellow,end=' ')
print(red)#assigned to the variable as a list
apple banana cherry
apple banana ['cherry', 'strawberry', 'raspberry']
```

Python - Loop Tuples

```
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
  print(x,end=' ')
print()
for i in range(len(thistuple)):
  print(thistuple[i],end=' ')
print()
i = 0
while i < len(thistuple):
  print(thistuple[i],end=' ')
  i = i + 1
apple banana cherry
apple banana cherry
apple banana cherry
```

Python - Join Tuples

```
tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)
tuple3 = tuple1 + tuple2
print(tuple3)
mytuple = tuple1 * 2
print(mytuple)
('a', 'b', 'c', 1, 2, 3)
('a', 'b', 'c', 'a', 'b', 'c')
```

Tuple Methods

Python has two built-in methods that you can use on tuples.

Method	**Description**
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

Python Sets

Sets are used to store multiple items in a single variable.

A set is a collection which is both *unordered* and *unindexed*.

Sets are written with curly brackets.

Tập hợp được sử dụng để lưu trữ nhiều mục trong một biến duy nhất.

Một tập hợp là một bộ sưu tập mà là cả hai *không có thứ tự* và *unindexed* .

Tập hợp được viết bằng dấu ngoặc nhọn.

basic:

```
# Note: the set list is unordered, meaning: the items will appear in a random order.
# Refresh this page to see the change in the result.
#Sets cannot have two items with the same value.
thisset = {"apple", "banana", "cherry"}
print(thisset)
print(len(thisset))
print(type(thisset))
{'apple', 'banana', 'cherry'}
3
<class 'set'>
```

Access Set Items

```
thisset = {"apple", "banana", "cherry"}
for x in thisset:
  print(x,end=' ')
print()
print("banana" in thisset)
```

```
cherry apple banana
True
```

Python - Add Set Items

```
thisset = {"apple", "banana"}
thisset.add("orange")
print(thisset)
tropical = {"pineapple",}
thisset.update(tropical)
print(thisset)
#it can be any iterable object (tuples, lists, dictionaries etc.).
mylist = ["kiwi", "orange"]
thisset.update(mylist)
print(thisset)
```

```
{'banana', 'orange', 'apple'}
{'banana', 'orange', 'pineapple', 'apple'}
{'apple', 'kiwi', 'banana', 'orange', 'pineapple'}
```

Python - Remove Set Items

```
#If the item to remove does not exist, remove() will raise an error.
thisset = {"apple", "banana", "cherry","kiwi","oil"}
thisset.remove("banana")
print(thisset)
#If the item to remove does not exist, discard() will NOT raise an error.
thisset.discard("apple")
print(thisset)
x = thisset.pop()
print(x)
print(thisset)
thisset.clear()
print(thisset)
del thisset
# print(thisset) sẽ bị lỗi
```

```
{'kiwi', 'oil', 'cherry', 'apple'}
{'kiwi', 'oil', 'cherry'}
kiwi
{'oil', 'cherry'}
set()
```

Python - Loop Sets

```
thisset = {"apple", "banana", "cherry"}
for x in thisset:
  print(x,end=' ')
print()
```

```
banana apple cherry
```

Python - Join Sets

```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}
set3 = set1.union(set2)
print(set3)
set1.update(set2)
print(set1)
```

```
{'a', 1, 2, 3, 'c', 'b'}
{'a', 1, 2, 3, 'c', 'b'}
```

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
x.intersection_update(y)
print(x)
z = x.intersection(y)
print(z)
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}
x.symmetric_difference_update(y)
print(x)
z = x.symmetric_difference(y)
print(z)
```

```
{'apple'}
{'apple'}
{'banana', 'google', 'cherry', 'microsoft'}
{'banana', 'apple', 'cherry'}
```

Set Methods

Python has a set of built-in methods that you can use on sets.

Method	**Description**
add()	Adds an element to the set
clear()	Removes all the elements from the set
copy()	Returns a copy of the set
difference()	Returns a set containing the difference between two or more sets
difference_update()	Removes the items in this set that are also included in

	another, specified set
discard()	Remove the specified item
intersection()	Returns a set, that is the intersection of two other sets
intersection_update()	Removes the items in this set that are not present in other, specified set(s)
isdisjoint()	Returns whether two sets have a intersection or not
issubset()	Returns whether another set contains this set or not
issuperset()	Returns whether this set contains another set or not
pop()	Removes an element from the set
remove()	Removes the specified element
symmetric_difference()	Returns a set with the symmetric differences of two sets
symmetric_difference_update()	inserts the symmetric differences from this set and another
union()	Return a set containing the union of sets
update()	Update the set with the union of this set and others

Python Dictionaries

Dictionaries are used to store data values in key:value pairs.
A dictionary is a collection which is ordered*, changeable and does not allow duplicates.
Dictionaries are written with curly brackets, and have keys and values:
Từ điển được sử dụng để lưu trữ các giá trị dữ liệu trong các cặp key: value.
Từ điển là một tập hợp được sắp xếp theo thứ tự *, có thể thay đổi và không cho phép trùng lặp.
Từ điển được viết bằng dấu ngoặc nhọn và có các khóa và giá trị:

Basic:

```
thisdict =	{
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964,
  "year": 2020

}
print(thisdict)
print(thisdict["brand"])
print(len(thisdict))
print(type(thisdict))
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
Ford
3
<class 'dict'>
```

Python - **Access Dictionary Items**

```
thisdict =	{
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict["model"])
x = thisdict["model"]
print(x)
x = thisdict.get("model")
print(x)
x = thisdict.keys()
print(x)
thisdict["name"]="huy"
print(thisdict.keys())
x = thisdict.values()
print(x)
thisdict["name"]="HUY"
print(thisdict.values())
if "model" in thisdict:
  print("Yes, 'model' is in the thisdict dictionary")
#The items() method will return each item in a dictionary, as tuples in a list.Make a change in the original dictionary, and see that the items list gets updated as well:
x = thisdict.items()
print(x)
```

```
Mustang
Mustang
Mustang
dict_keys(['brand', 'model', 'year'])
dict_keys(['brand', 'model', 'year', 'name'])
dict_values(['Ford', 'Mustang', 1964, 'huy'])
dict_values(['Ford', 'Mustang', 1964, 'HUY'])
Yes, 'model' is in the thisdict dictionary
dict_items([('brand', 'Ford'), ('model', 'Mustang'),
('year', 1964), ('name', 'HUY')])
```

Python - **Change Dictionary Items**

```
thisdict =	{
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["year"] = 2018
print(thisdict)
thisdict.update({"year": 2020})
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2018}
{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

Python - **Remove Dictionary Items**

```
thisdict =	{
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964,
  "name":"Huy"
}
thisdict.pop("model")
print(thisdict)
thisdict.popitem()#removes the last inserted item
print(thisdict)
thisdict.clear()
print(thisdict)
```

```
{'brand': 'Ford', 'year': 1964, 'name': 'Huy'}
{'brand': 'Ford', 'year': 1964}
{}
```

Python - **Loop Dictionaries**

```
thisdict =	{
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
for x in thisdict:
  print(x,end=' ')
print()
for x in thisdict:
  print(thisdict[x],end=' ')
print()
for x in thisdict.values():
  print(x,end=' ')
print()
for x in thisdict.keys():
  print(x,end=' ')
print()
for x, y in thisdict.items():
  print(x, y,end=' ')
print()
```

```
brand model year
Ford Mustang 1964
Ford Mustang 1964
brand model year
brand Ford model Mustang year 1964
```

Python - **Copy Dictionaries**

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
mydict = thisdict.copy()
print(mydict)
mydict = dict(thisdict)
print(mydict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964}
```

Python - Nested Dictionaries : từ điển lồng

```
child1 = {
  "name" : "Emil",
  "year" : 2004
}
child2 = {
  "name" : "Tobias",
  "year" : 2007
}
child3 = {
  "name" : "Linus",
  "year" : 2011
}

myfamily = {
  "child1" : child1,
  "child2" : child2,
  "child3" : child3
}
print(myfamily)
```

```
{'child1': {'name': 'Emil', 'year': 2004}, 'child2':
{'name': 'Tobias', 'year': 2007}, 'child3': {'name':
'Linus', 'year': 2011}}
```

Dictionary Methods

Python has a set of built-in methods that you can use on dictionaries.

Method	Description
clear()	Removes all the elements from the dictionary
copy()	Returns a copy of the dictionary
fromkeys()	Returns a dictionary with the specified keys and value
get()	Returns the value of the specified key
items()	Returns a list containing a tuple for each key value pair
keys()	Returns a list containing the dictionary's keys
pop()	Removes the element with the specified key
popitem()	Removes the last inserted key-value pair
setdefault()	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
update()	Updates the dictionary with the specified key-value pairs
values()	Returns a list of all the values in the dictionary

Python If ... Else

```
#The elif keyword is pythons way of saying "if the previous conditions were not true, then try this condition".
#The else keyword catches anything which isn't caught by the preceding conditions.
# and+or
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")
if a > b: print("a is greater than b")
print("A") if a > b else print("B")
c = 500
if a > b and c > a:
  print("Both conditions are True")
if a > b or a > c:
  print("At least one of the conditions is True")
```

```
a is greater than b
a is greater than b
A
Both conditions are True
At least one of the conditions is True
```

Python While Loops

```
i = 1
while i < 6:
  print(i,end=' ')
  i += 1
else:
  print("\ni is no longer less than 6")
```

```
1 2 3 4 5
i is no longer less than 6
```

Python For Loops

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
  if x == "banana":break
for x in fruits:
  if x == "banana":continue
  print(x)
for x in range(6):
  print(x,end= ' ')
else:
  print("Finally finished!")
```

```
apple
banana
apple
cherry
0 1 2 3 4 5 Finally finished!
```

Python Functions

```
def my_function(fname):
  print(fname + " Refsnes")
my_function("Emil")

def my_function1(country = "Norway"):
  print("I am from " + country)
my_function1("India")
my_function1()

def my_function3(x):
  return 5 * x
print(my_function3(3))
print(my_function3(5))
```

```
Emil Refsnes
I am from India
I am from Norway
15
25
```

```
def tri_recursion(k):
  if(k > 0):
    result = k + tri_recursion(k - 1)
    print(result,end=' ')
  else:
    result = 0
  return result
tri_recursion(6)
```

```
1 3 6 10 15 21
```

Python Lambda (ẩn danh)

```
x = lambda a: a + 10
print(x(5))
x = lambda a, b: a * b
print(x(5, 6))
x = lambda a, b, c: a + b + c
print(x(5, 6, 2))
def myfunc(n):
  return lambda a : a * n
mydoubler = myfunc(2)
print(mydoubler(11))
```

```
15
30
13
```

Python Arrays

Note: This page shows you how to use LISTS as ARRAYS, however, to work with arrays in Python you will have to import a library, like the NumPy library.

```
cars = ["Ford", "Volvo", "BMW"]
cars[0] = "Toyota"
print(cars)
x = len(cars)
print(x)
for x in cars:
  print(x,end=' ')
cars.append("Honda")
print('\n',cars)
cars.pop(3)
print(cars)
```

```
['Toyota', 'Volvo', 'BMW']
3
Toyota Volvo BMW
 ['Toyota', 'Volvo', 'BMW', 'Honda']
['Toyota', 'Volvo', 'BMW']
```