Python History

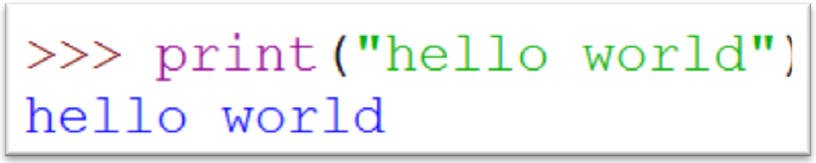
Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991

BY::Abdallah Elsokary

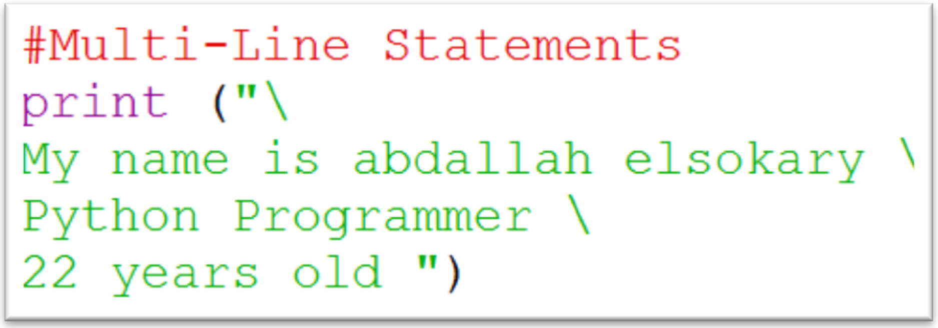
Python3 Basic Syntax

Frist hello world program with python3

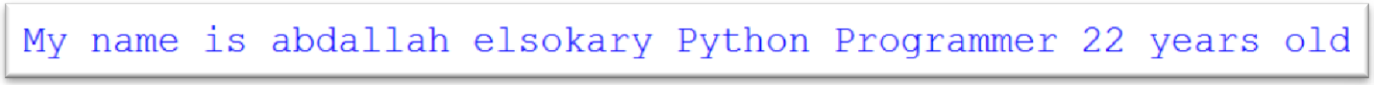
Print Function



```
>>> print("hello world")
hello world
```



```
#Multi-Line Statements
print ("\
My name is abdallah elsokary \
Python Programmer \
22 years old ")
```



```
My name is abdallah elsokary Python Programmer 22 years old
```

Python3 Basic Syntax

Quotation in Python3

Single (‘)

Double (“)

Triple (“””” or ‘’’)

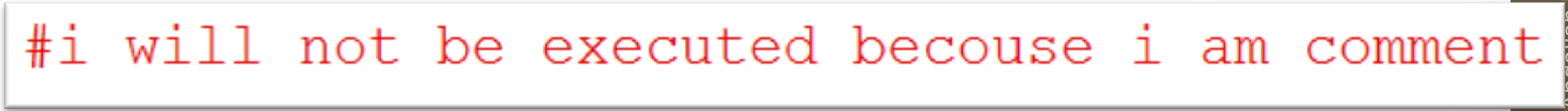
The triple quotes are used to span the string across multiple lines

Python3 Basic Syntax

comments in Python3

# comment syntax

Any syntax after # Python interpreter ignores them.

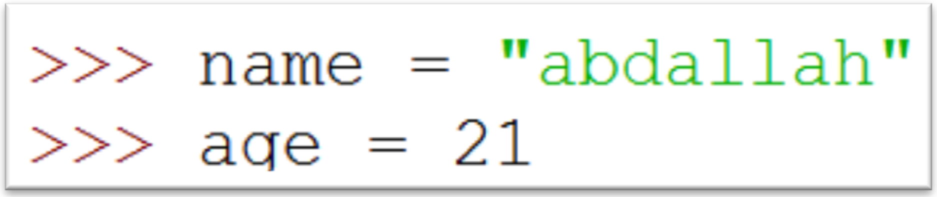


```
#i will not be executed becouse i am comment
```

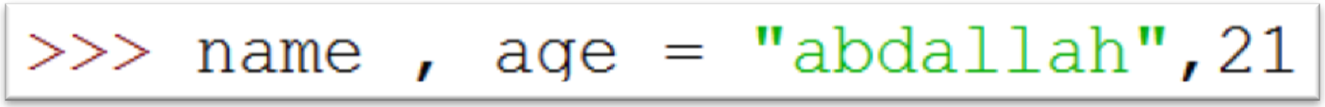
Python3 Variables Types

Var_name = Value

Var1 , Var2 = Value1 ,Value2



```
>>> name = "abdallah"
>>> age = 21
```



```
>>> name , age = "abdallah",21
```

Python3 Variables Types

Python has five standard data types

- Numbers
- String
- List
- Tuple
- Dictionary

Python3 Variables Types

Numbers

Python supports three different numerical types:

int (signed integers) [100]

float (floating point real values) [100.5]

complex (complex numbers) [100j]

Python3 Arithmetic Operators

+ Addition

- Subtraction

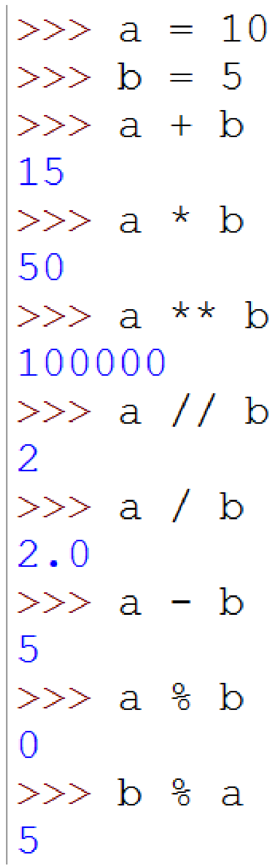
*** Multiplication**

/ Division

// Floor Division

% Modulus

**** Exponent**



```
>>> a = 10
>>> b = 5
>>> a + b
15
>>> a * b
50
>>> a ** b
100000
>>> a // b
2
>>> a / b
2.0
>>> a - b
5
>>> a % b
0
>>> b % a
5
```

BY::Abdallah Elsokary

Python3 Assignment Operators

=

+= Add AND

-= Subtract AND

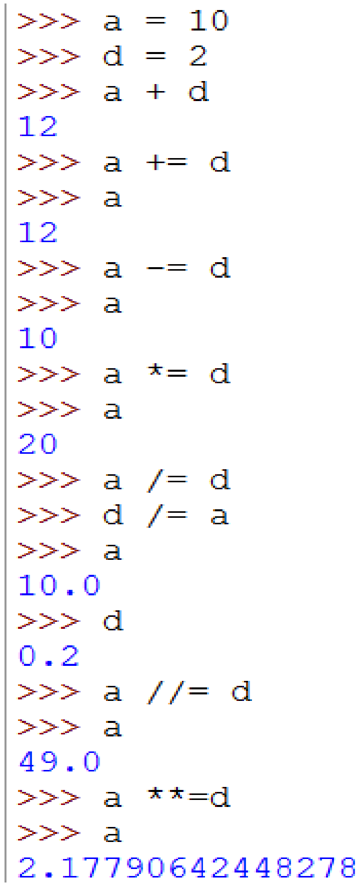
*= Multiply AND

/= Divide AND

%= Modulus AND

//= Floor Division

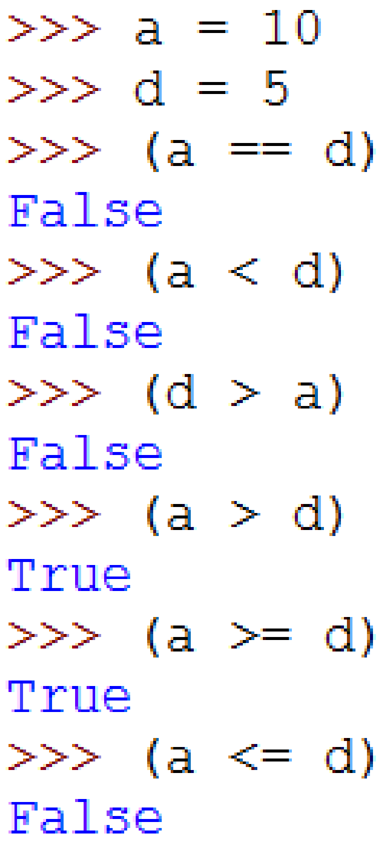
**= Exponent AND



```
>>> a = 10
>>> d = 2
>>> a + d
12
>>> a += d
>>> a
12
>>> a -= d
>>> a
10
>>> a *= d
>>> a
20
>>> a /= d
>>> d /= a
>>> a
10.0
>>> d
0.2
>>> a //= d
>>> a
49.0
>>> a **=d
>>> a
2.17790642448278
```

Python3 Comparison Operators

==	equal
!=	not equal
>	greater than
<	less than
>=	greater than or equal
<=	less than or equal

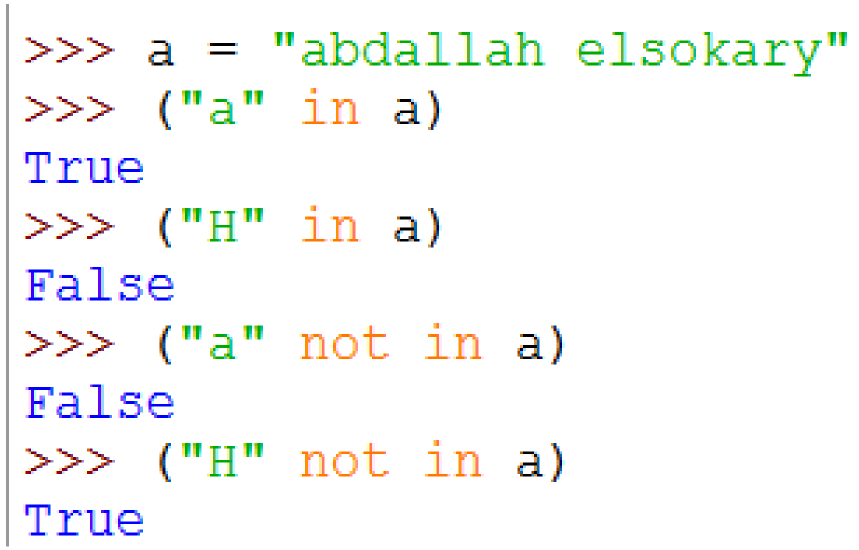


```
>>> a = 10
>>> d = 5
>>> (a == d)
False
>>> (a < d)
False
>>> (d > a)
False
>>> (a > d)
True
>>> (a >= d)
True
>>> (a <= d)
False
```

Python3 Membership Operators

In

not in

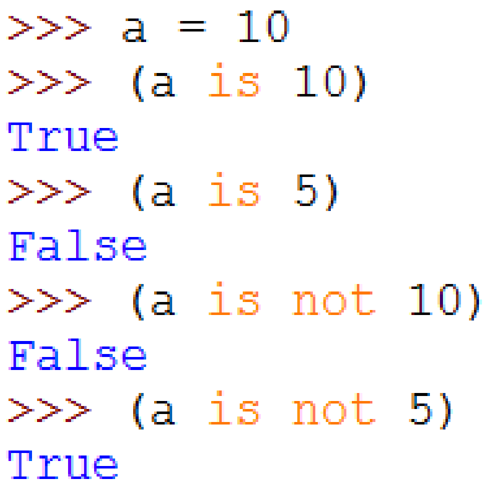


```
>>> a = "abdallah elsokary"
>>> ("a" in a)
True
>>> ("H" in a)
False
>>> ("a" not in a)
False
>>> ("H" not in a)
True
```

Python3 Identity Operators

Is

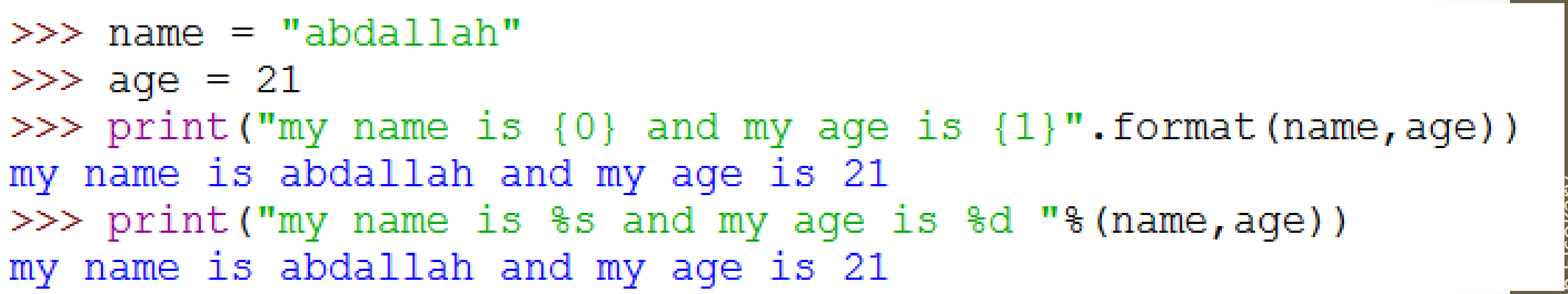
is not



```
>>> a = 10
>>> (a is 10)
True
>>> (a is 5)
False
>>> (a is not 10)
False
>>> (a is not 5)
True
```

format

Format(args)



```
>>> name = "abdallah"
>>> age = 21
>>> print("my name is {0} and my age is {1}".format(name,age))
my name is abdallah and my age is 21
>>> print("my name is %s and my age is %d "%(name,age))
my name is abdallah and my age is 21
```

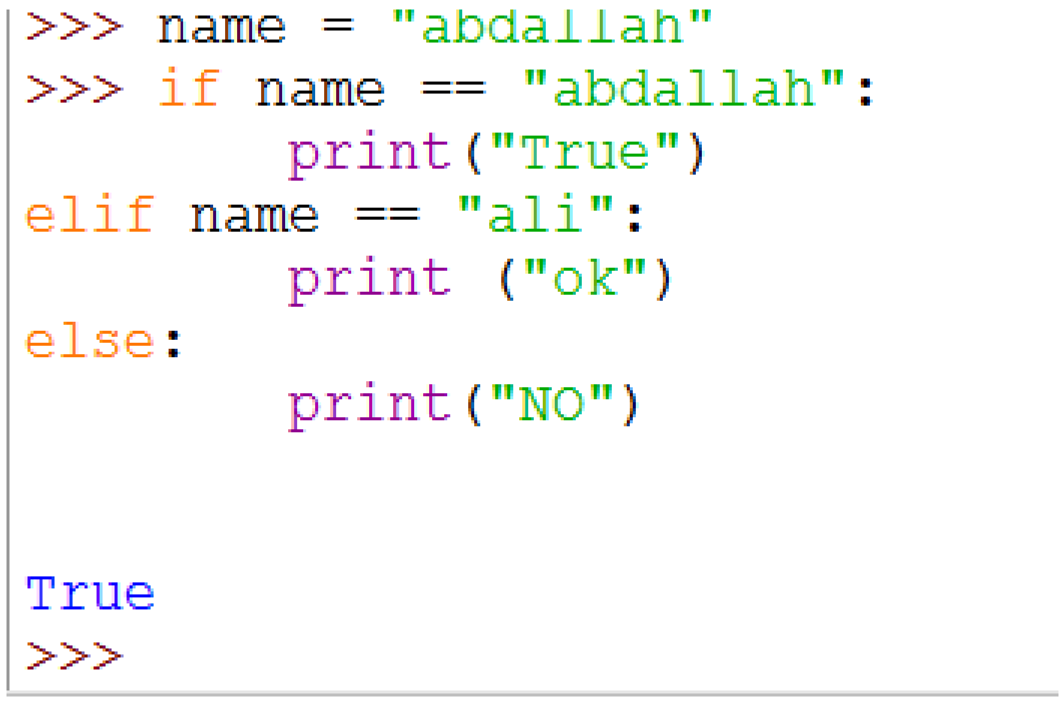
BY::Abdallah Elsokary

If statement

If

elif

else



```
>>> name = "abdallah"
>>> if name == "abdallah":
        print("True")
elif name == "ali":
        print ("ok")
else:
        print("NO")


True
>>>
```

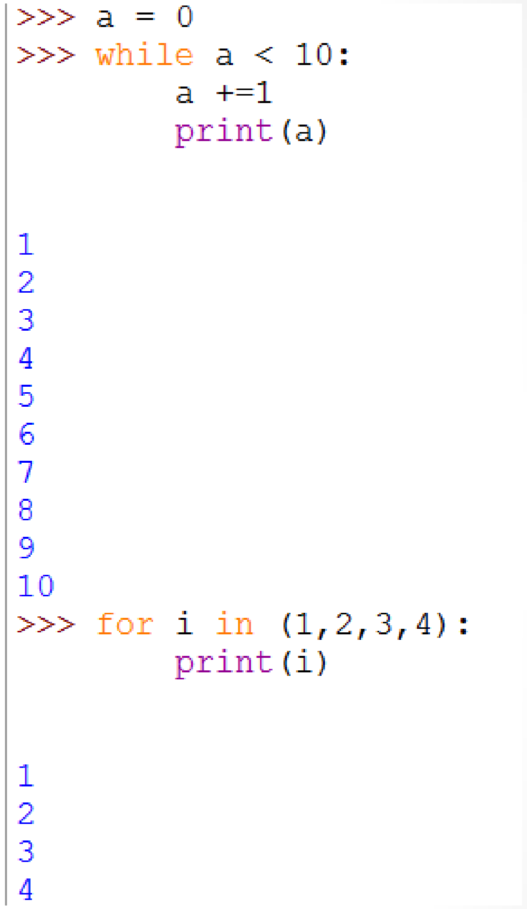
loop

While

For

Control:

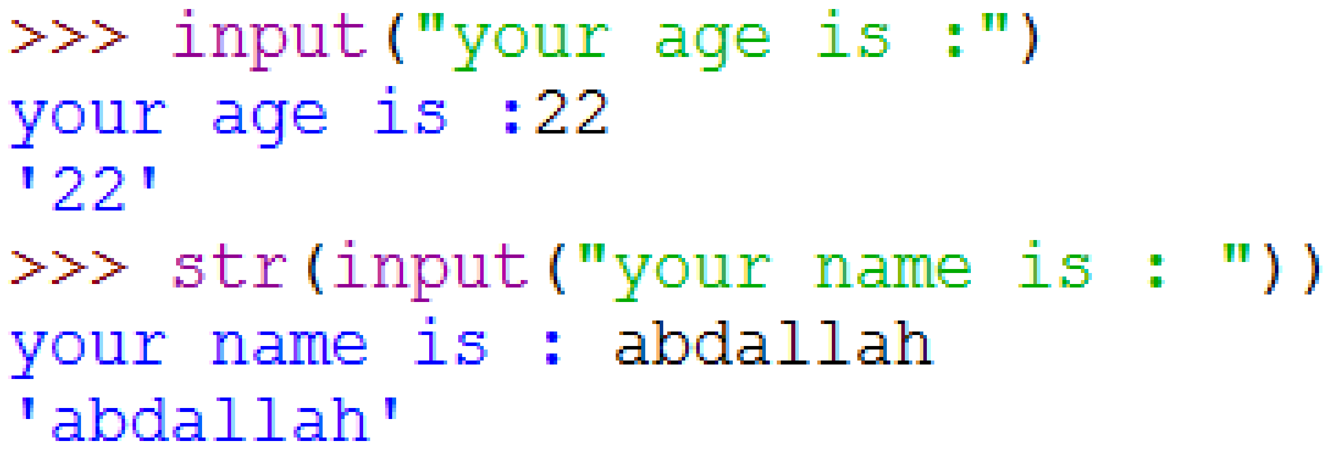
- **Continue**
- **Break**
- **pass**



```
>>> a = 0
>>> while a < 10:
	a +=1
	print(a)


1
2
3
4
5
6
7
8
9
10
>>> for i in (1,2,3,4):
	print(i)


1
2
3
4
```

input

Input(“ask for? ”)



```
>>> input("your age is :")
your age is :22
'22'
>>> str(input("your name is : "))
your name is : abdallah
'abdallah'
```

Built-in String Methods

capitalize() Capitalizes first letter of string

title() Returns "titlecased" version of string, that is, all words begin with upper case and the rest are lowercase.

center(width, fillchar) Returns a string padded with *fillchar* with the original string centered to a total of *width* columns.

count(str, beg= 0,end = len(string)) Counts how many times str occurs in string or in a substring of string if starting index beg and ending index end are given.

encode(encoding='UTF-8',errors='strict') Returns encoded string version of string; on error, default is to raise a ValueError unless errors is given with 'ignore' or 'replace'.

decode(encoding='UTF-8',errors='strict') Decodes the string using the codec registered for encoding. encoding defaults to the default string.

Built-in String Methods

endswith (suffix, beg=0, end=len(string)) Determines if string or a substring of string (if starting index beg and ending index end are given) ends with suffix; returns true if so and false otherwise.

find(str, beg=0 end=len(string)) Determine if str occurs in string or in a substring of string if starting index beg and ending index end are given returns index if found and -1 otherwise.

index(str, beg=0, end=len(string)) Same as find(), but raises an exception if str not found.

isalnum() Returns true if string has at least 1 character and all characters are alphanumeric and false otherwise.

Built-in String Methods

isalpha() Returns true if string has at least 1 character and all characters are alphabetic and false otherwise.

isdigit() Returns true if the string contains only digits and false otherwise.

islower() Returns true if string has at least 1 cased character and all cased characters are in lowercase and false otherwise.

isnumeric() Returns true if a unicode string contains only numeric characters and false otherwise.

istitle() Returns true if string is properly "titlecased" and false otherwise.

isspace() Returns true if string contains only whitespace characters and false otherwise.

Built-in String Methods

isupper() Returns true if string has at least one cased character and all cased characters are in uppercase and false otherwise.

len(string) Returns the length of the string

rstrip() Removes all trailing whitespace of string.

lstrip() Removes all leading whitespace in string.

strip([chars]) Performs both lstrip() and rstrip() on string

lower() Converts all uppercase letters in string to lowercase.

upper() Converts lowercase letters in string to uppercase.

Built-in String Methods

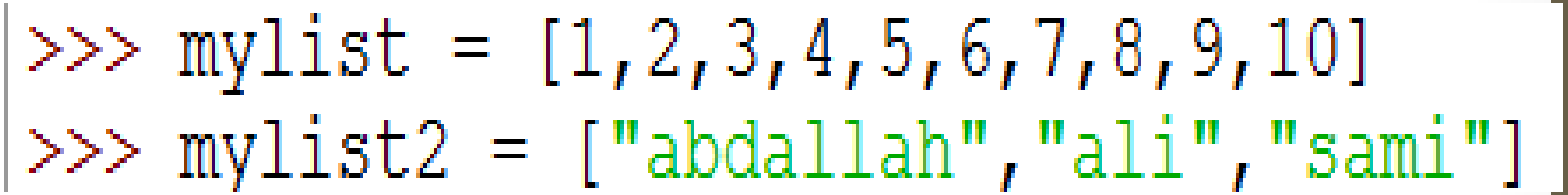
isdecimal() Returns true if a unicode string contains only decimal characters and false otherwise.

split(str="", num=string.count(str)) Splits string according to delimiter str (space if not provided) and returns list of substrings; split into at most num substrings if given.

splitlines(num=string.count('\n')) Splits string at all (or num) NEWLINEs and returns a list of each line with NEWLINEs removed.

Python List

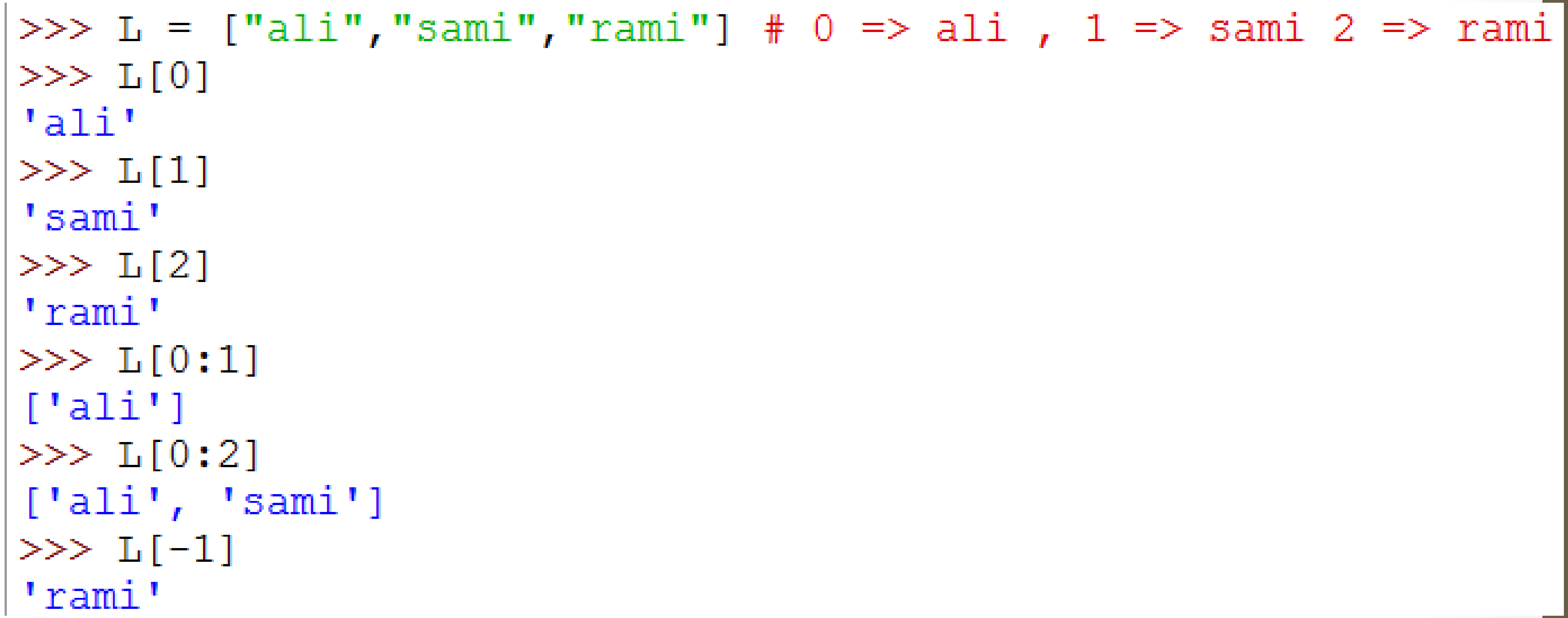
Versatile data type available in Python, which can be written as a list of comma-separated values (items) between square brackets.



```
>>> mylist = [1,2,3,4,5,6,7,8,9,10]
>>> mylist2 = ["abdallah","ali","sami"]
```

Python List

List index start from => 0



```
>>> L = ["ali","sami","rami"] # 0 => ali , 1 => sami 2 => rami
>>> L[0]
'ali'
>>> L[1]
'sami'
>>> L[2]
'rami'
>>> L[0:1]
['ali']
>>> L[0:2]
['ali', 'sami']
>>> L[-1]
'rami'
```

List Functions

len(list) Gives the total length of the list

max(list) Returns item from the list with max value.

min(list) Returns item from the list with min value.

list(seq) Converts a tuple into list.

List methods

list.append(obj) Appends object obj to list

list.count(obj) Returns count of how many times obj occurs in list

list.extend(seq) Appends the contents of seq to list

list.index(obj) Returns the lowest index in list that obj appears

list.insert(index, obj) Inserts object obj into list at offset index

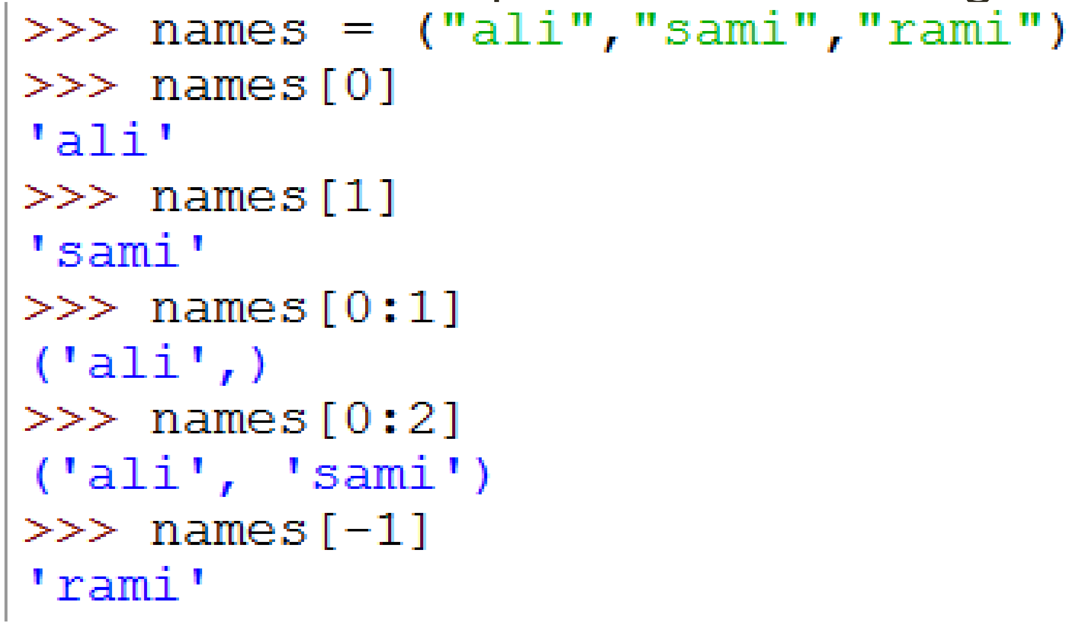
list.pop(obj=list[-1]) Removes and returns last object or obj from list

list.remove(obj) Removes object obj from list

list.sort([func]) Sorts objects of list, use compare func if given

Python Tuples

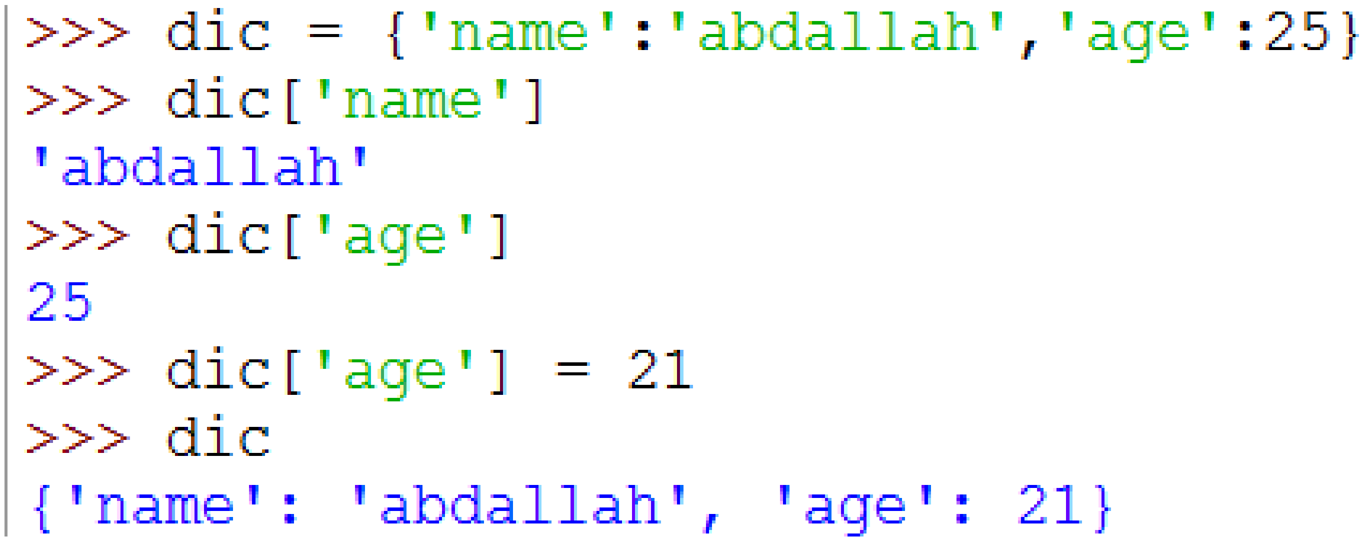
Tuples are sequences, just like lists The main difference between the tuples and the lists is that the tuples cannot be changed unlike lists



```
>>> names = ("ali","sami","rami")
>>> names[0]
'ali'
>>> names[1]
'sami'
>>> names[0:1]
('ali',)
>>> names[0:2]
('ali', 'sami')
>>> names[-1]
'rami'
```

Python Dictionary

Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces.



```
>>> dic = {'name':'abdallah','age':25}
>>> dic['name']
'abdallah'
>>> dic['age']
25
>>> dic['age'] = 21
>>> dic
{'name': 'abdallah', 'age': 21}
```

Dictionary Methods

dict.clear() Removes all elements of dictionary *dict.*

dict.copy() Returns a shallow copy of dictionary *dict.*

dict.fromkeys() Create a new dictionary with keys from seq and values *set* to *value*.

dict.get(key, default=None) For *key* , returns value or default if key not in dictionary

dict.has_key(key) Removed, use the **in** operation instead.

dict.items() Returns a list of *dict*'s (key, value) tuple pairs.

dict.keys() Returns list of dictionary dict's keys.

dict.setdefault(key, default=None) Similar to get(), but will set dict[key]=default if *key* is not already in dict.

Dictionary Methods

dict.update(dict2) Adds dictionary *dict2*'s key-values pairs to *dict.*

dict.values() Returns list of dictionary *dict*'s values.

Files I/O

Modes:

- w+
- a+
- r+
- w
- a
- r
- rb / rb+
- ab / ab +
- wb / wb+

methods:

name

mode

closed

flush

read

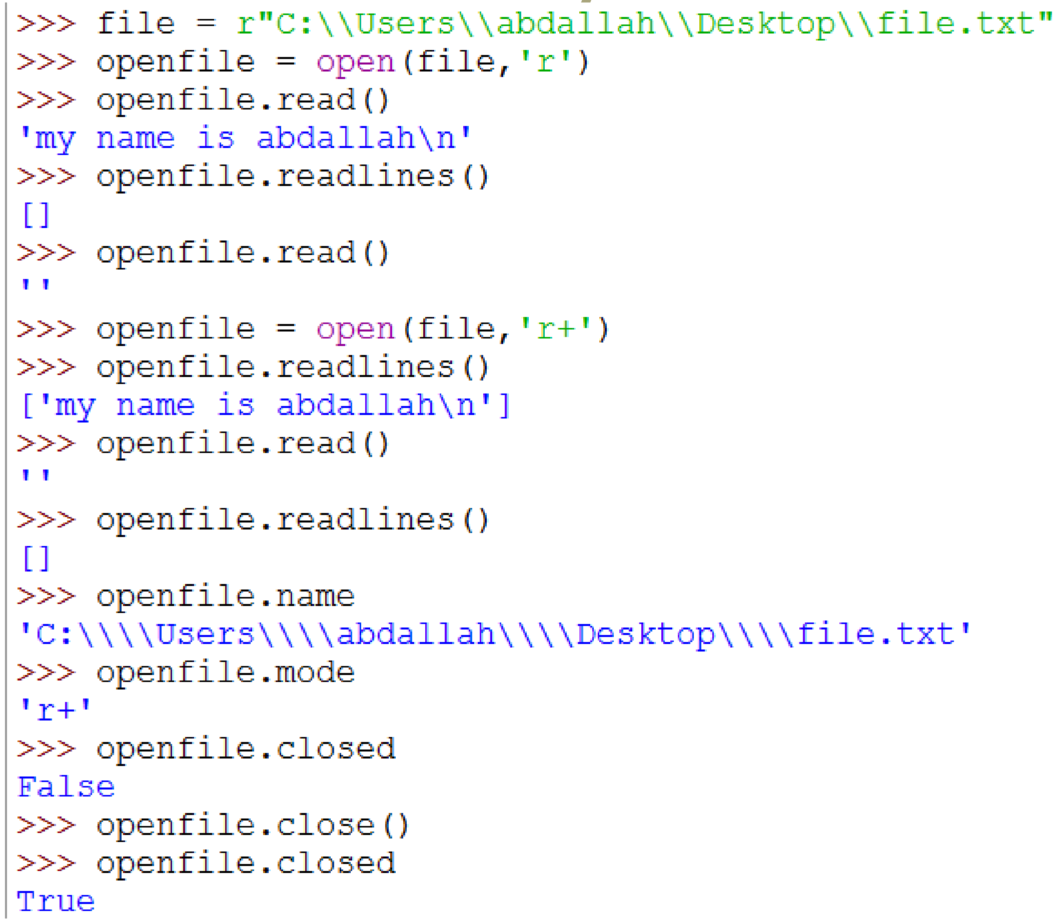
write

readlines

writelines

Files I/O

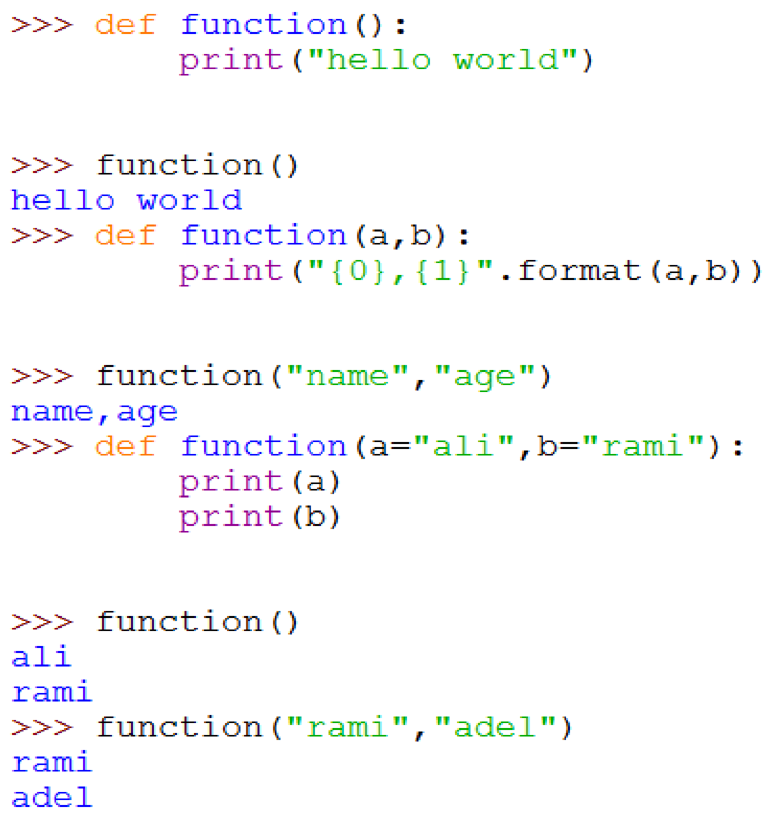
Ex:



```
>>> file = r"C:\\Users\\abdallah\\Desktop\\file.txt"
>>> openfile = open(file,'r')
>>> openfile.read()
'my name is abdallah\n'
>>> openfile.readlines()
[]
>>> openfile.read()
''
>>> openfile = open(file,'r+')
>>> openfile.readlines()
['my name is abdallah\n']
>>> openfile.read()
''
>>> openfile.readlines()
[]
>>> openfile.name
'C:\\\\Users\\\\abdallah\\\\Desktop\\\\file.txt'
>>> openfile.mode
'r+'
>>> openfile.closed
False
>>> openfile.close()
>>> openfile.closed
True
```

Python Method

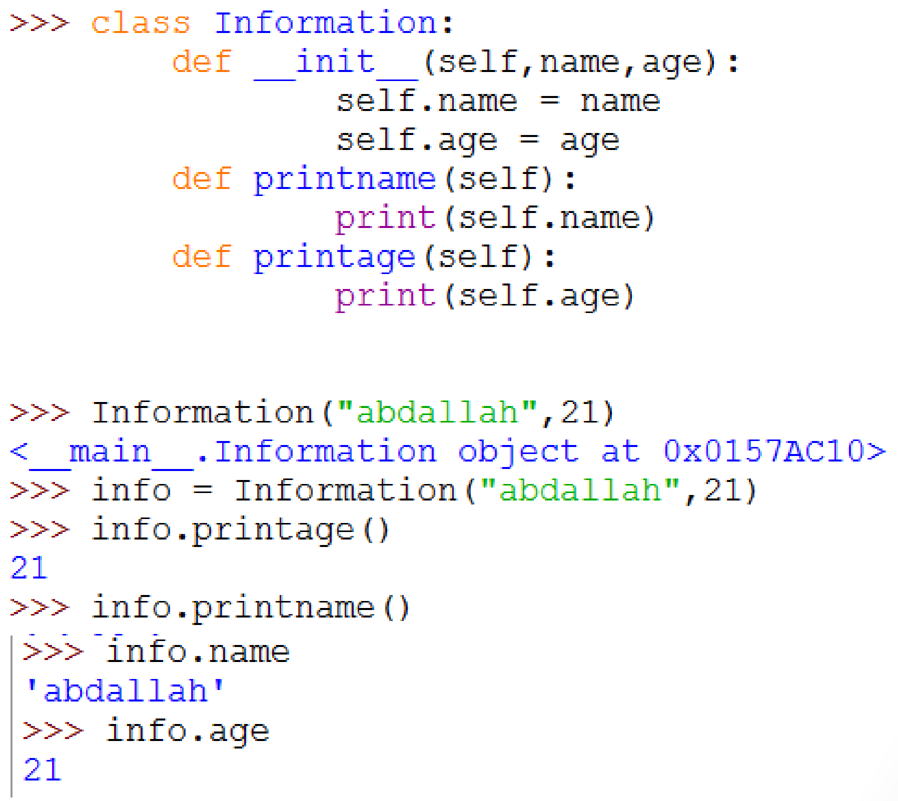
Python Method



```
>>> def function():
        print("hello world")


>>> function()
hello world
>>> def function(a,b):
        print("{0},{1}".format(a,b))


>>> function("name","age")
name,age
>>> def function(a="ali",b="rami"):
        print(a)
        print(b)


>>> function()
ali
rami
>>> function("rami","adel")
rami
adel
```

Python class

Python class

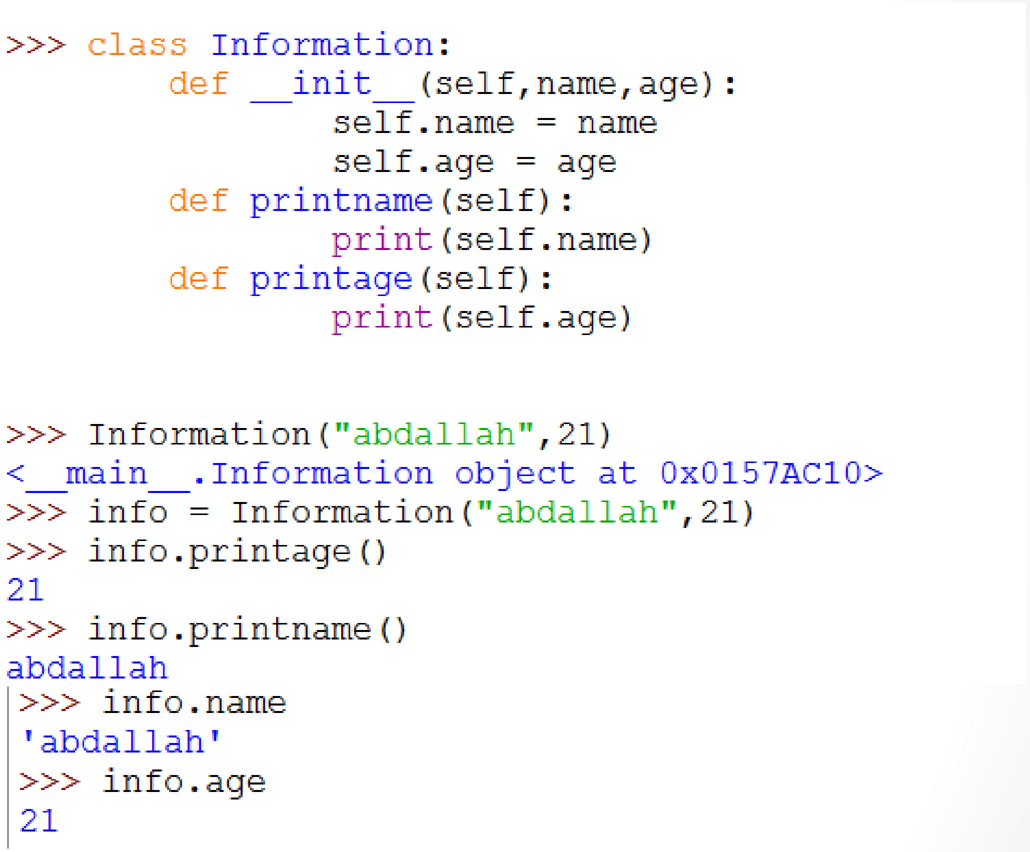


```
>>> class Information:
        def __init__(self,name,age):
                self.name = name
                self.age = age
        def printname(self):
                print(self.name)
        def printage(self):
                print(self.age)


>>> Information("abdallah",21)
<__main__.Information object at 0x0157AC10>
>>> info = Information("abdallah",21)
>>> info.printage()
21
>>> info.printname()
>>> info.name
'abdallah'
>>> info.age
21
```

BY::Abdallah Elsokary

Python class

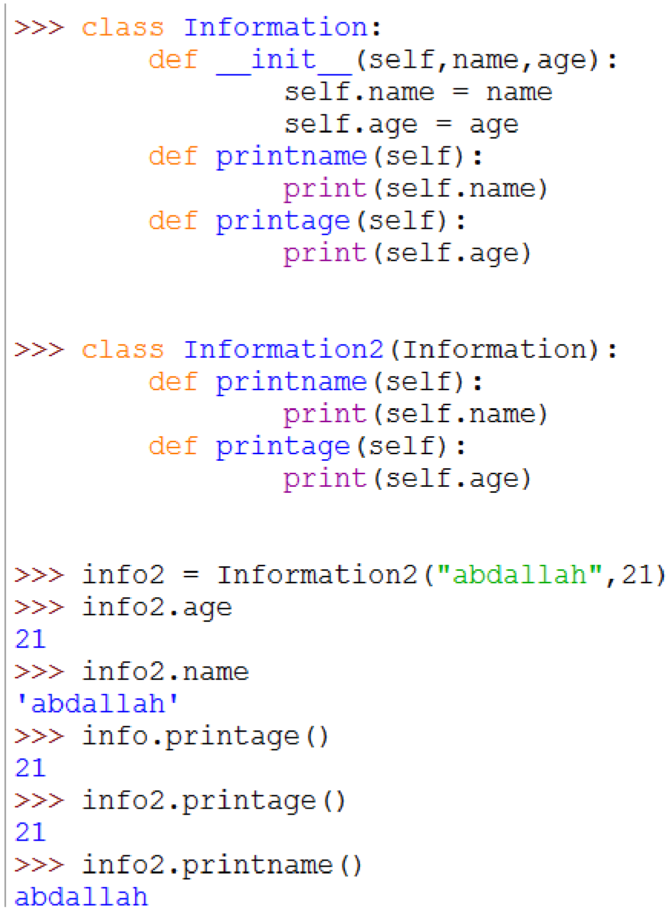
Python class



```
>>> class Information:
        def __init__(self,name,age):
                self.name = name
                self.age = age
        def printname(self):
                print(self.name)
        def printage(self):
                print(self.age)


>>> Information("abdallah",21)
<__main__.Information object at 0x0157AC10>
>>> info = Information("abdallah",21)
>>> info.printage()
21
>>> info.printname()
abdallah
>>> info.name
'abdallah'
>>> info.age
21
```

Python class

Python class inheritance



```
>>> class Information:
        def __init__(self,name,age):
                self.name = name
                self.age = age
        def printname(self):
                print(self.name)
        def printage(self):
                print(self.age)


>>> class Information2(Information):
        def printname(self):
                print(self.name)
        def printage(self):
                print(self.age)


>>> info2 = Information2("abdallah",21)
>>> info2.age
21
>>> info2.name
'abdallah'
>>> info.printage()
21
>>> info2.printage()
21
>>> info2.printname()
abdallah
```