# Shortcut trong pycharm

* Ctr + /: comment or uncomment multiple lines
* Alt + shift + E: execute selection part

# Biến

* Khai báo

x = str(3)    # x will be '3'  
y = int(3)    # y will be 3  
z = float(3)  # z will be 3.0

* Đặt tên cho biến có nhiều từ

myVariableName = "John"

MyVariableName = "John"

my\_variable\_name = "John"

* Gán nhiều biến

x, y, z = "Orange", "Banana", "Cherry"

x = y = z = "Orange"

fruits = ["apple", "banana", "cherry"]  
x, y, z = fruits

* Output biến

Để kết hợp cả văn bản và một biến, Python sử dụng ký tự +:

x = "awesome"  
print("Python is " + x)

# 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

# Chỉ định loại dữ liệu

int(): tạo một số nguyên từ một ký tự số nguyên, một ký tự float (bằng cách loại bỏ tất cả các số thập phân) hoặc một chuỗi ký tự (cung cấp chuỗi đại diện cho một số nguyên)

float(): khởi tạo một số thực từ một ký tự số nguyên, một ký tự float hoặc một ký tự chuỗi (cung cấp chuỗi đại diện cho một số thực hoặc một số nguyên)

str(): xây dựng một chuỗi từ nhiều kiểu dữ liệu, bao gồm chuỗi, ký tự số nguyên và ký tự float

# Strings

* Kiểm tra “free” xuất hiện trong txt:

txt = "The best things in life are free!"

if “free” in txt:

print(“true”)

* Kiểm tra không xuất hiện:

if “free” not in txt:

print(“false”)

* Cắt chuỗi
  + Lấy ký tự từ 2 đến 5 (không bao gồm 5)

b = "Hello, World!"  
print(b[2:5])

* Sửa chuỗi
  + Strip() loại bỏ mọi ký tự trằng từ bắt đầu hoặc kết thúc

a = " Hello, World! "  
print(a.strip()) # returns "Hello, World!"

* + Replace() thay thế một chuỗi với chuỗi khác

a = "Hello, World!"  
print(a.replace("H", "J"))

* + Split chia chuỗi thành chuỗi con nếu nó tìm thấy các dấu phân tách.

a = "Hello, World!"  
print(a.split(",")) # returns ['Hello', ' World!']

* Nối chuỗi

a = "Hello"  
b = "World"  
c = a + b  
print(c)

* Định dạng chuỗi

Dùng format() để chèn số vào chuỗi

age = 36  
txt = "My name is John, and I am {}"  
print(txt.format(age))

* Ký tự đặc biệt

Để in ra được ký tự “ thì phải dùng /”

txt = "We are the so-called \"Vikings\" from the north."

Một số ký tự khác

\' \\ \n \r \t \b \f \ooo \xhh

# Booleans

Mọi chuỗi đều là true, trừ chuỗi rỗng

Mọi số là true, trừ số 0

MỌi list, tuple, set, and dictionary là true, trừ cái rỗng

print(10 > 9) # In ra “True”  
print(10 == 9) # In ra “False”

print(bool("Hello")) # True

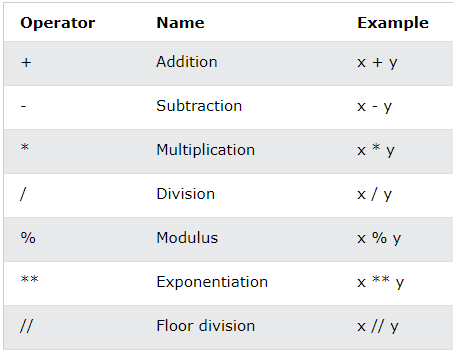
print(bool(15)) # True

print(bool(0)) # False

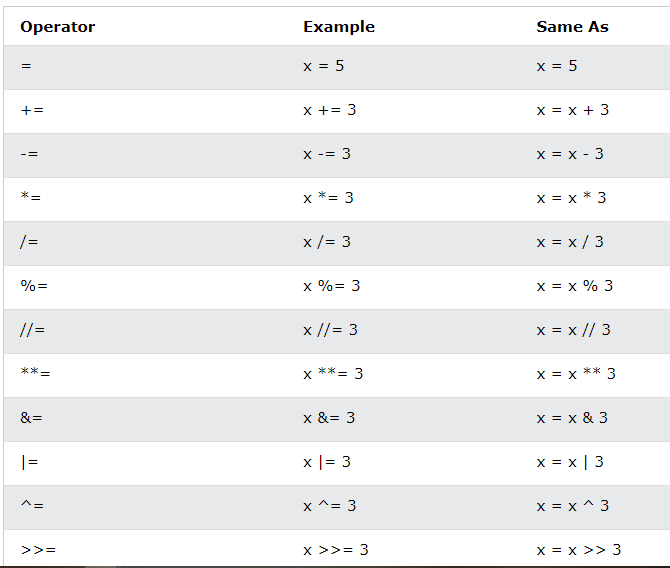
bool(False)  
bool(None)  
bool(0)  
bool("")

Operators

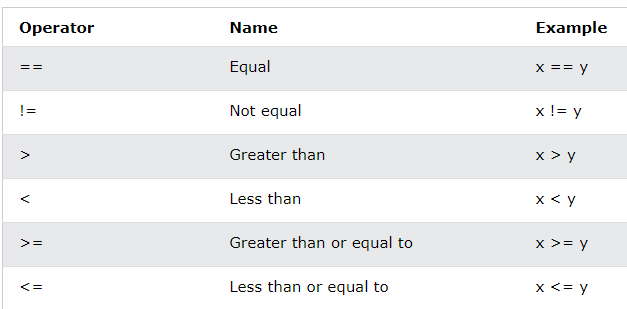
Toán tử số học



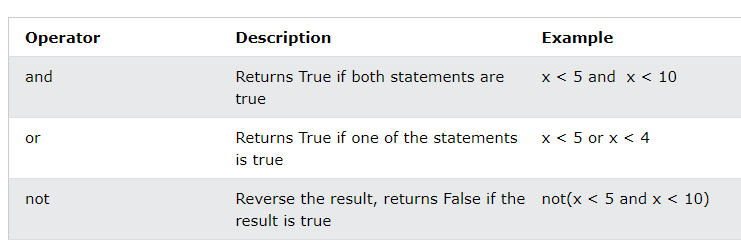
Toán tử gán



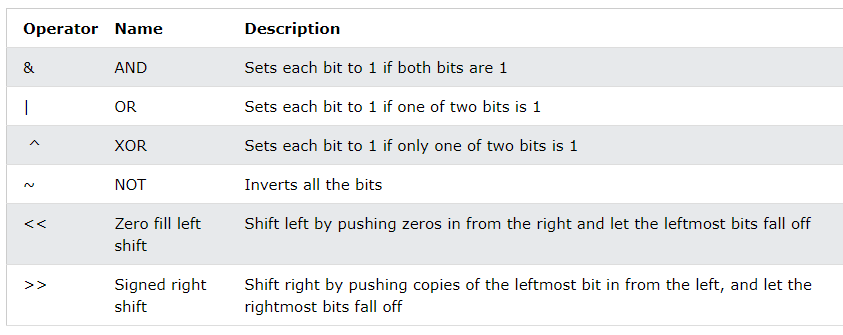
Toán tử so sánh



Toán tử Logic



Toán tử Bitwise



Toán tử Identity và toán tử Membershi

So sánh nếu đều cùng là một đối tượng

x is y

So sánh xem một chuỗi có nằm trong chuỗi khác hay không

x in y

# Methods

* str(i): int to string
* append(arr): add to end of array
* join(): Join all items in a tuple, list into a string, using a hash character as separator
  + syntax: string.join(iterable)

# Ex1

myTuple = ("John", "Peter", "Vicky")

x = "#".join(myTuple)

print(x)

# Ex2

myDict = {"name": "John", "country": "Norway"}

mySeparator = "TEST"

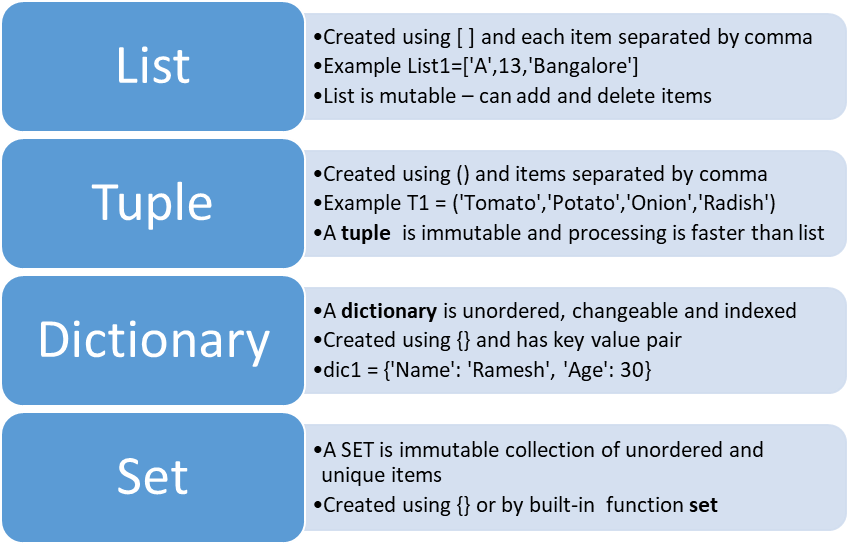
x = mySeparator.join(myDict)

print(x)

* type(students): loại dữ liệu
* set(words): xóa các phần tử lặp lạitrong list word
* xắp xếp các phần tử
  + words.sort()
  + words = sorted(words)
* c.isdigit(): kiểm tra biến c có phải là chữ cái
* c.isalpha(): kiểm tra biễn co phải là chữ số

# List, Set, Dictionary, Tuple

* 4 kiểu dữ liệu tích hợp sẵn trong Python được sử dụng để lưu trữ các bộ sưu tập dữ liệu là List, Tuple, Set và Dictionary, tất cả đều có mục đích và cách sử dụng khác nhau.
* Khi nói kiểu dữ liệu được sắp xếp theo thứ tự, không có nghĩa là được sắp xếp, nó có nghĩa là các items sẽ hiển thị theo thứ tự bạn đặt chúng. Điều này trái ngược với, chẳng hạn như SortedSet trong đó các item sẽ không được sắp xếp theo thứ tự bạn đặt chúng, chúng sẽ được sắp xếp theo bất kỳ tiêu chí sắp xếp nào bạn đặt hoặc HashMap, nơi chúng có thể hiển thị theo bất kỳ thứ tự nào tại tất cả, tùy thuộc vào hàm băm.



## List

* Các item trong list được sắp xếp theo thứ tự, có thể thay đổi và cho phép giá trị trùng lặp
* Khởi tạo list:

thislist = list(("apple", "banana", "cherry")) # note the double round-brackets

* Truy cập L
  + -1 đề cập đến phần tử cuối, -2 là phần tử gần cuối, …
  + Truy cập từ phần tử 2 đến 5 (không bao gồm 5)

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "melon", "mango"]  
print(thislist[2:5])

* + Xác định một phần tử cụ thể xuất hiện trong list: sử dụng ‘in’

thislist = ["apple", "banana", "cherry"]  
if "apple" in thislist:

* Thay đổi phạm vi giá trị items

# Thay đổi giá trị từ 1 đến 3 (không bao gồm 3)

thislist = ["apple", "banana", "cherry", "orange", "kiwi", "mango"]  
thislist[1:3] = ["blackcurrant", "watermelon"]  
print(thislist)

* Thêm list items
  + Insert vào vị trí cụ thể

thislist = ["apple", "banana", "cherry"]  
thislist.insert(1, "orange")

* + Mở rộng list

thislist = ["apple", "banana", "cherry"]  
tropical = ["mango", "pineapple", "papaya"]  
thislist.extend(tropical)

* Xóa items trong list
  + Dùng phương thức remove() để xóa item cụ thể

thislist = ["apple", "banana", "cherry"]  
thislist.remove("banana")

* + Dùng pop() để xóa chỉ mục cụ thể

thislist = ["apple", "banana", "cherry"]  
thislist.pop(1) # thislist.pop() sẽ xóa phần tử cuối cùng

* List comprehension
  + Ví dụ khi muốn tạo ra một list mới dựa vào list cũ

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]  
newlist = []  
for x in fruits:  
  if "a" in x:  
    newlist.append(x)

* + Có thể thực hiện chỉ bằng một dòng lệnh

newlist = [x for x in fruits if "a" in x]

* Sắp xếp list
  + Sắp xếp tăng dần

thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]  
thislist.sort()

* + Sắp xếp giảm dần

thislist.sort(reverse = True)

* Copy list
  + Nếu dùng list2 = list1, thì list2 chỉ tham chiếu đến list1, và khi list1 thay đổi thì list2 cũng sẽ thay đổi
  + Sử dụng phương thức copy() hoặc list để sao chép

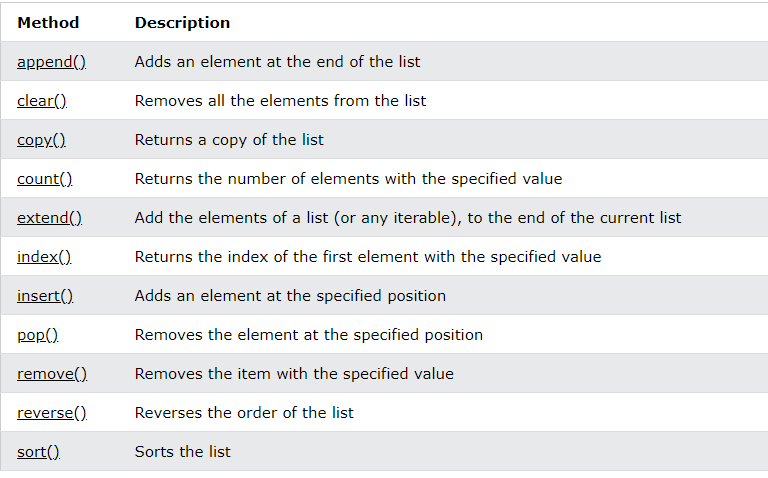
list2 = list1.copy()

list2 = list(list2)

* Nối list (Join list)

Có nhiều cách để nối 2 list, chẳng hạn

* + list3 = list2 + list1
  + for x in list2:
    - list1.append(x)
  + list.extend(list2)
* Các phương thức trong list



## Tuple

Các item trong tuple được sắp xếp theo thứ tự, không thể thay đổi và cho phép các giá trị trùng lặp

* Tổng quan
* Tuples được sử dụng để lưu trữ nhiều mục trong một biến duy nhất.
  + Một tuples được sắp xếp theo thứ tự và không thể thay đổi

thistuple = (“banana”, “apple”)

* + Tuples được lập chỉ mục

thistuple[0], thistuple[1]

* + Tuples length

len(thistuple)

* + Tuples items có thể là bất cứ loại dữ liệu nào, có thể chứa nhiều loại dữ liệu
  + Khai báo tuple

emptyTuple = ()

emptyTuple = tuple()

z = (4, 5, 1, 7)

z = 4, 5, 1, 6

tup1 = (“Micheal”,)

tup2 = “Micheal”,

* Truy cập tuple tương tự như List
* Cập nhật tuple tương tự như list

## Set

Các item trong set không theo thứ tự và không được lập chỉ mục. Không có item trùng lặp

## Dictionary

Các item trong dict được sắp xếp theo thứ tự, có thể thay đổi và không có giá trị trùng lặp

# For Loops

* Type one

fruits = ["apple", "banana", "cherry"]

for x in fruits:

print(x)

* Type two: print each character

for x in "banana":

print(x)

* Type three: break, continue

fruits = ["apple", "banana", "cherry"]

for x in fruits:

if x == "banana":

break

else:

print(x)

continue

* Type for: range, increment the sequence with 3 (default is 1):

for x in range(2, 30, 3):

print(x, end=" ")

* Type five: pass

for x in [0, 1, 2]:

pass

# Function

* Nếu bạn không biết sẽ truyền vào bao nhiêu đối số, đặt dấu ‘\*’ trước đối số

def my\_function(\*kids):  
  print("The youngest child is " + kids[2])  
my\_function("Emil", "Tobias", "Linus")

* Nếu bạn không biết sẽ truyền vào bao nhiêu đối số keyword, đặt dấu ‘\*\*’ trước đối số

def my\_function(\*\*kid):  
  print("His last name is " + kid["lname"])  
my\_function(fname = "Tobias", lname = "Refsnes")

* Định nghĩa giá trị mặc định

def my\_function(**country = "Norway"**):  
  print("I am from " + country)

# Iterator

Iterator  là một đối tượng có chứa một số giá trị có thể đếm được.

List, set, tuple, dict tất cả đề là iterable objects

mytuple = ("apple", "banana", "cherry")  
myit = iter(mytuple)  
  
print(next(myit))  
print(next(myit))

# Scope

* Một biến mà chỉ có sẵn từ bên trong vùng mà nó được tạo gọi là scope
* Local scope:

Một biến được tạo bên trong một hàm thuộc phạm vi cục bộ của hàm đó và chỉ có thể được sử dụng bên trong hàm đó

* Global scope:

Một biến được tạo trong phần thân chính của python là biến toàn cục

* Global Keyword
* Nếu bạn cần tạo một biến toàn cục nhưng đang kẹt trong local scope, thì bạn có thể sử dụng global keyword

def myfunc():  
 global x  
 x = 300

* Ngoài ra, có thể thay đổi biên toàn cục khi ở trong hàm

x = 300  
def myfunc():  
  global x  
  x = 200

# Modules

* Xem module giống như một code library. Tập tin chứa một tập hợp cá chức năng bạn muốn để đưa vào ứng dụng của mình
  + - * + Tạo một module

Để tạo một module chỉ cần lưu code bạn muốn trong một file với tên .py

* Ví dụ lưu đoan code sau với tên mymodule.py

def greeting(name):  
  print("Hello, " + name)

* Dùng lệnh import để sử dụng module vừa tạo

import mymodule  
mymodule.greeting("Jonathan")

* Tạo bí danh khi nhập module

import mymodule as mx  
a = mx.person1["age"]

* Hàm dir() được tích hợp sẵn để có thể liệt kê tất cả tên hàm hoặc biến trong module đó.

import platform  
x = dir(platform)  
print(x)

* Có thể chọn import một phần từ module bằng keyword from

Ví dụ mymodule có biến person1 và hàm greeting

from mymodule import person1, greeting

# Math

* Các hàm tích hợp sẵn

min(), max(), abs() (trị tuyệt đối), pow()

x = min(5, 10, 25)  
y = max(5, 10, 25)

x = abs(-7.25)

* Math module

import math để sử dụng

Một số hàm như

x = math.sqrt(64)

x = math.ceil(1.4) # return 2  
y = math.floor(1.4) # return 1

# RegEx

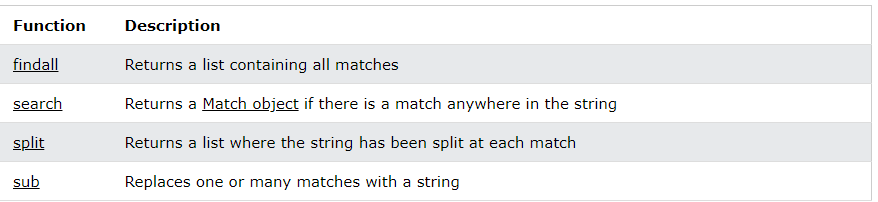
* Regular Expression, là một chuỗi ký tự hình thành một mẫu tìm kiếm. Nó có thể được sử dụng để kiểm tra nếu một chuỗi chứa mẫu tìm kiếm cụ thể.
  + Ví dụ: Tìm chuỗi để xem nếu nó bắt đầu với “The” và kết thúc với “Spain”:

import re

txt = “The rain in Spain”

x = re.search(“^The.\*Spain$”, txt)

* Hàm RegEx
  + re module cung cấp một tập hợp các hàm cho phép chúng ta tìm kiếm một chuỗi cho một kết quả phù hợp



# Pip

* Install a packet
  + pip install camelcase
* Liệt kê tất cả các gói đã được cài đặt

# Try .. Except

* Khối ‘try’ cho phép kiểm tra một khối code để tìm lỗi
* Khối ‘except” cho phép bạn xử lý lỗi
* Khối ‘finally’ cho phép bạn thực thi code, liên quan đến kết quả của khối ‘try’ và ‘except’
* Xử lý ngoại lệ
  + Khi lỗi xảy ra, hoặc exception như chúng ta gọi, python sẽ thường dùng lại và tạo ra một thông báo lỗi. Các exception này có thể được xử lý bằng câu lệnh try
  + Ví dụ: try sẽ tạo ra một ngoại lệ vì x chưa được định nghĩa

try:  
  print(x)  
except:  
  print("An exception occurred")

* Định nghĩa nhiều block để tìm exception đặc biệt

try:  
  print(x)  
except NameError:  
  print("Variable x is not defined")  
except:  
  print("Something else went wrong")

* Else

Khối này sẽ được thực thi nếu không có lỗi xảy ra

try:  
  print("Hello")  
except:  
  print("Something went wrong")  
else:  
  print("Nothing went wrong")

* Finally

Khối này sẽ được thực thi dù có lỗi xảy ra hay không

try:  
  print(x)  
except:  
  print("Something went wrong")  
finally:  
  print("The 'try except' is finished")

* Throw một exception

Throw một lỗi và dừng chương trình

x = -1  
  
if x < 0:  
  raise Exception("Sorry, no numbers below zero")

# String format()

* Cho phép bạn định dạng phần được chọn của chuỗi. Đôi khi, có các phần của text mà bạn không kiểm soát, có thể chúng đến từ database hoặc user input. Để kiểm soát các giá trị như vậy, thêm dấu ngoặc nhọn vào đoạn text, và chạy thông qua hàm format()

price = 49  
txt = "The price is {} dollars"  
print(txt.format(price))

* Trường hợp nhiều giá trị

quantity = 3  
itemno = 567  
price = 49  
myorder = "I want {} pieces of item number {} for {:.2f} dollars."  
print(myorder.format(quantity, itemno, price))

* Chỉ mục là số

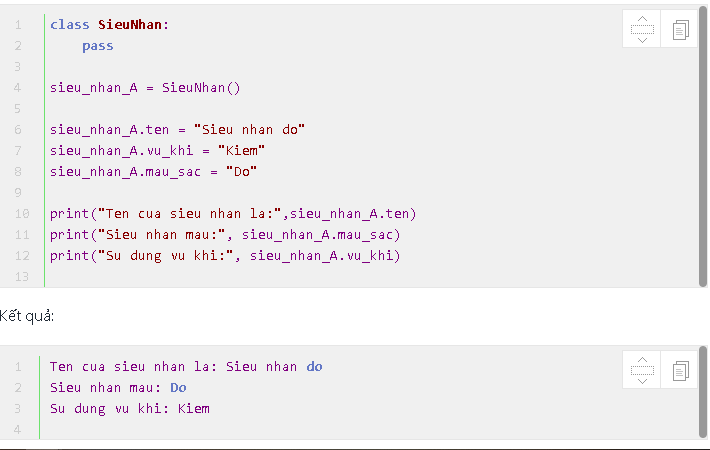
myorder = "I want {0} pieces of item number {1} for {2:.2f} dollars."  
print(myorder.format(quantity, itemno, price))

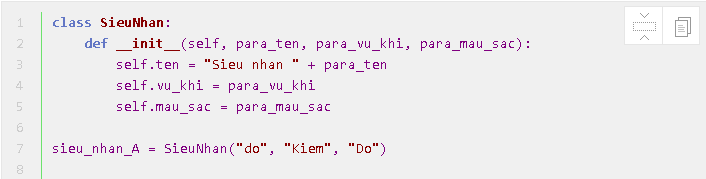
* Chỉ mục được đặt tên

myorder = "I have a {carname}, it is a {model}."  
print(myorder.format(carname = "Ford", model = "Mustang"))

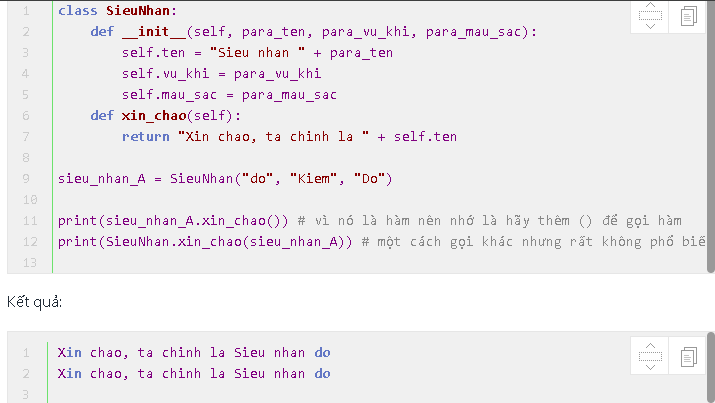
# Class

## Định nghĩa, lớp và đối tượng

* Class là một bản mẫu, một khuôn mẫu, hoặc một kiểu dữ liệu tự được định nghĩa
* Thuộc tính (attribute, proptertis) là thành phần dữ liệu hay còn gọi là các biến
* Phương thức (method) của class thường là hàm nằm trong class
* Khi nói hàm thì thường hiểu là chương trình bình thường được thực thi
* Class trong python không cần khai báo thuộc tính nhưng vẫn có thể hoạt động được nhờ việc tự định nghĩa và thêm vào
* Hàm constructor (initialize method)
  + Constructor sẽ được gọi khi ta khai báo một đối tượng
  + Ý nghĩa của self trong hàm này là chính đối tượng đó

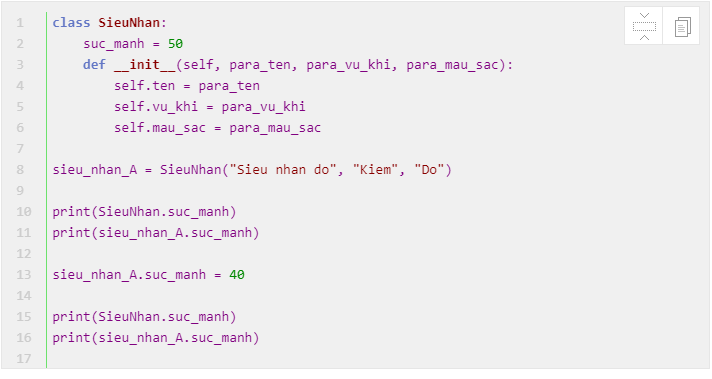


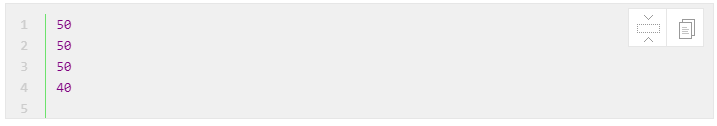
* Hai cách gọi hàm, phổ biến nhất là dùng cách đầu tiên



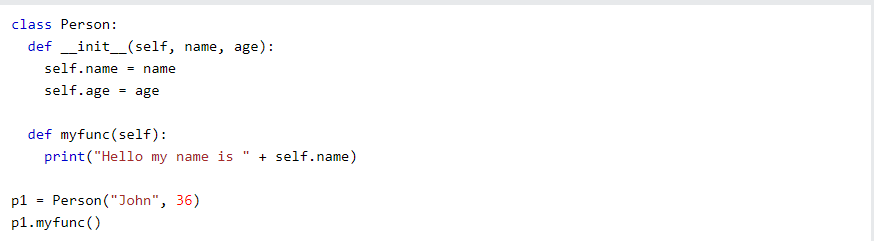
## Thuộc tính, method của lớp

* Khi lấy thuộc tính hoặc method ở class thì sử dụng dấu “.” để gọi ra
* Vd về sử dụng thuộc tính



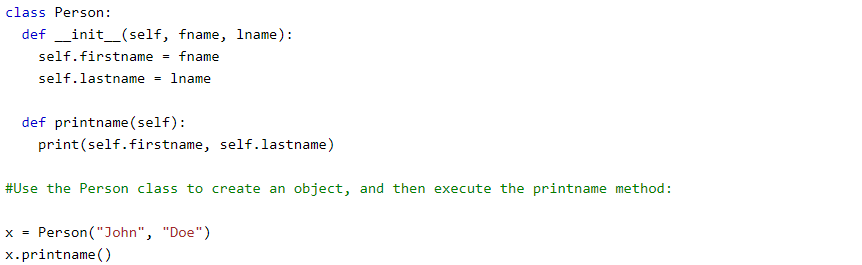


* Ví dụ về method

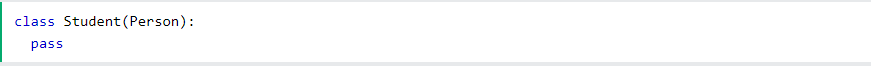


## Kế thừa

Tạo lớp cha và lớp con kế thừa



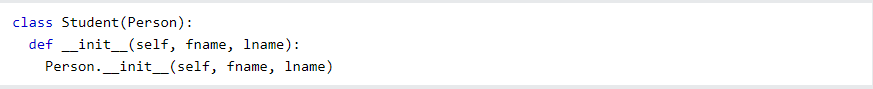
* Lưu ý: sử dụng pass keyword nếu không muốn thêm vào bất kỳ thuộc tính (properties, attribute) hoặc phương thức (method)



* Hàm \_\_init\_\_() của lớp con sẽ ghi đè lên sự kế thừa hàm \_\_init\_\_() của lớp cha.



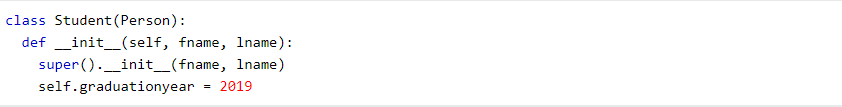
* Để giữ lại sự kế thừa hàm \_\_init\_\_() của lớp cha thì, thêm một dòng gọi hàm \_\_init\_\_() của lớp cha



* Python cũng có một hàm super() sẽ tạo ra lớp con kế thừa tất cả thuộc tính và phương thức từ lớp cha



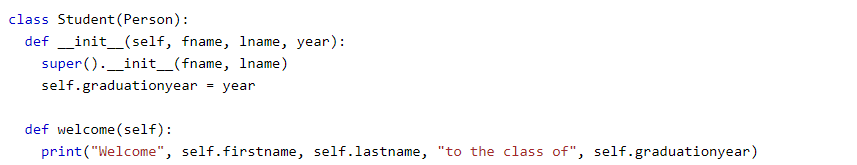
* Thêm thuộc tính bằng cách như sau khi kế thừa toàn bộ những thuộc tính khác của lớp cha
  + Cách 1:



* + Cách 2



* Thêm phương thức vào lớp con. Nếu thêm một phương thức ở lớp con với cùng tên phương thức ở lớp cha thì phương thức ở lớp cha sẽ bị ghi đè



# Lưu ý

* Nhập vào số nguyên

x = int(input("Nhập số nguyên: "))

* Khai báo dictionary

myDict = {}

d=dict()

* Chuyển từ list sang tuple

t = tuple(list)

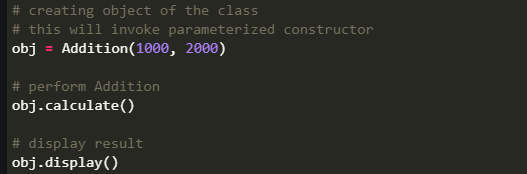
* Phương thức (method) của class thường là hàm nằm trong class. Ví dụ a.cappitalize()
* Khi nói hàm thì thường hiểu là chương trình bình thường được thực thi. Ví dụ print(
* Câu lệnh if else không thể để trống. Nên nếu vì lý do gì đó mà lệnh if else không có nội dung, đặt pass vào đó để tránh lỗi

if b > a:  
  pass

* Sự khác nhau giữa list và array
  + List trong python là một tập hợp các mục có thể chứa các phần tử của nhiều kiểu dữ liệu, có thể là số, giá trị lôgic ký tự, v.v.
  + Mảng là một vectơ chứa các phần tử thuần nhất, tức là thuộc cùng một kiểu dữ liệu. Các phần tử được cấp phát với các vị trí bộ nhớ liền kề cho phép dễ dàng sửa đổi, nghĩa là thêm, xóa, truy cập các phần tử.
* REST API định nghĩa một tập quy tắc, ràng buộc để ứng dụng web gửi và nhận dữ liệu

# Một số quy tắc viết code

* Sau khi kết thúc code của comment thì cách dòng



* Trước khi bắt đầu comment cũng cách dòng

