(Ghi chú)

Ngôn ngữ lập trình Python

**Thực hiện: Nguyễn Văn Hào**

**Email:** [hao2205tb@gmail.com](mailto:hao2205tb@gmail.com)

(Tài liệu được biên xoạn lại từ nhiều nguồn và không dùng trong mục đích kinh doanh kiếm tiền)

# Mục lục

[Mục lục 1](#_Toc127443335)

[Tổng quan 3](#_Toc127443336)

[Đặc điểm của python 4](#_Toc127443337)

[Học Python để làm gì và khi nào thì nên học Python 4](#_Toc127443338)

[Phát triển phần mềm chạy trên máy tính 4](#_Toc127443339)

[Phát triển trí tuệ nhân tạo 4](#_Toc127443340)

[Phân tích dữ liệu 4](#_Toc127443341)

[Người không nên học Python 5](#_Toc127443342)

[Tạo ứng dụng điện thoại thông minh (iPhone, Android) 5](#_Toc127443343)

[Phát triển ứng dụng và dịch vụ web 5](#_Toc127443344)

[Phát triển hệ thống/ Ứng dụng nhúng. 5](#_Toc127443345)

[Hello world 6](#_Toc127443346)

[Chương trình đầu tiên 6](#_Toc127443347)

[Comment 6](#_Toc127443348)

[Đặt mã ký tự: 6](#_Toc127443349)

[print() 7](#_Toc127443350)

[Cú pháp: 7](#_Toc127443351)

[Căn trái, phải, giữa: 7](#_Toc127443352)

[Chèn biến vào chuỗi sử dụng toán tử % 8](#_Toc127443353)

[Định dạng số khi in ra màn hình trong Python. 8](#_Toc127443354)

[In list, tuple và dictionary 8](#_Toc127443355)

[Pprint(): prety-print 9](#_Toc127443356)

[So sánh với print: 9](#_Toc127443357)

[Cú pháp 10](#_Toc127443358)

[Chỉ định số ký tự tối đa trên mỗi dòng in | đối số width 10](#_Toc127443359)

[Chỉ định số phần tử con tối đa được in trong dữ liệu lồng nhau | đối số depth 11](#_Toc127443360)

[Chỉ định độ thụt lề trên mỗi dòng in | đối số indent 11](#_Toc127443361)

[Giảm thiểu tối đa số lần ngắt dòng khi in các list có nhiều phần tử | Đối số compact 12](#_Toc127443362)

[input() 13](#_Toc127443363)

[Nhập dữ liệu 13](#_Toc127443364)

[Nhập cùng lúc nhiều dữ liệu trên một dòng 13](#_Toc127443365)

[Nhập nhiều dữ liệu trên nhiều dòng 13](#_Toc127443366)

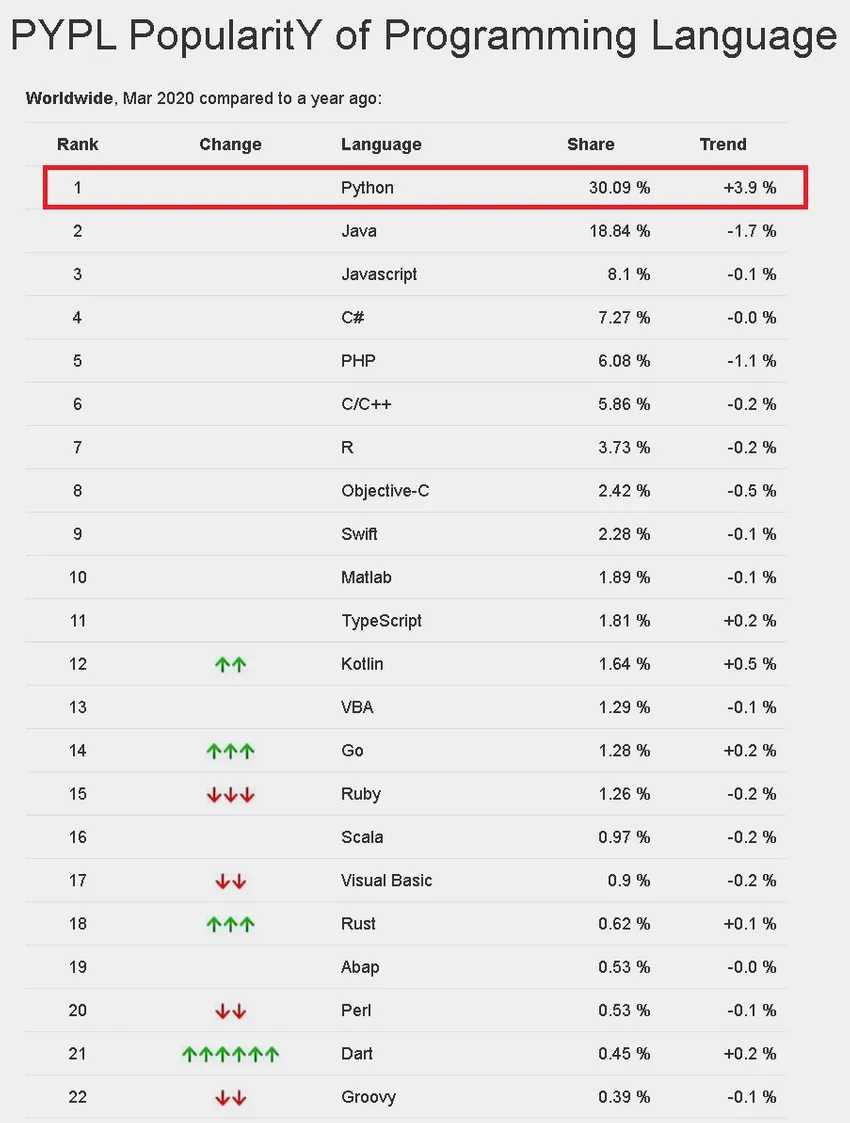
[Nhập số 13](#_Toc127443367)

[Tham khảo 15](#_Toc127443368)

# Tổng quan

Python ban đầu được phát triển bởi Guido van Rossum vào năm 1991, nhằm quản lý một hệ điều hành có tên Amoeba.

Python đang là ngôn ngữ lập trình được ưu chuộng nhất thế giới.



## Đặc điểm của python

Tham khảo: <http://laptrinhcanban.com/python/nhap-mon-lap-trinh-python/gioi-thieu-python/python-la-gi/>

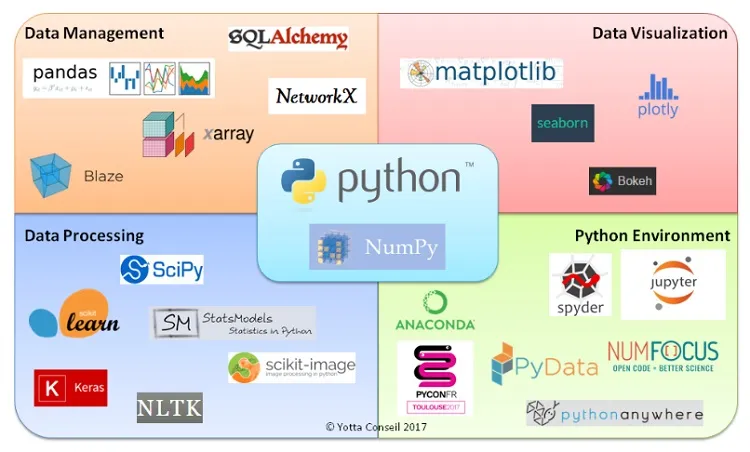
## Học Python để làm gì và khi nào thì nên học Python

### Phát triển phần mềm chạy trên máy tính

Tạo phần mềm tìm kiếm một file trong ổ cứng, chỉnh sửa ảnh, tạo video mp4 từ ảnh chụp sẵn. Xử lý nhanh chóng từ các file dữ liệu như excel, json đến các file âm thanh, hình ảnh thông dụng …

### Phát triển trí tuệ nhân tạo

Python được tích hợp rất nhiều thư viện được sử dụng cho nghiên cứu, phát triển deep learning.



### Phân tích dữ liệu

Làm việc với Big DATA

Python, với library điển hình là Pandas giúp bạn thực hiện các lệnh như đọc thông tin file CSV, sau dó tiến hành thêm, sửa, xóa và phân tích chúng một cách dễ dàng và hiệu quả.

## Người không nên học Python

### Tạo ứng dụng điện thoại thông minh (iPhone, Android)

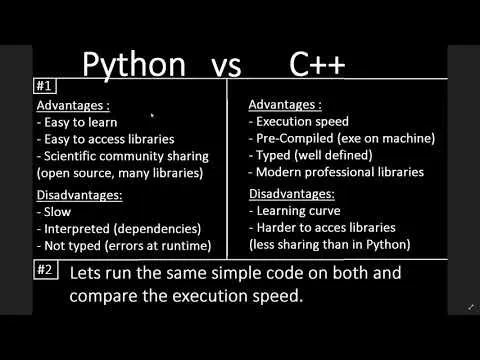
Nếu số sánh với Swift của iPhone hay Java của Android, và gần đây là Kotlin chẳng hạn, thì Python không có lợi thế trong lĩnh vực này.

### Phát triển ứng dụng và dịch vụ web

Tại Nhật Bản, thì hầu như các dự án tạo web được viết bởi Ruby và PHP. Do đó python tuy có tiềm năng nhưng thực tế chưa được ứng dụng nhiều như một ngôn ngữ chính để tạo web, mà được sử dụng chủ yếu để quản lý dữ liệu của trang web.

### Phát triển hệ thống/ Ứng dụng nhúng.

Do là ngôn ngữ tự dịch ( interpreter) - việc dịch python ra ngôn ngữ mà máy tính hiểu và việc thực hiện tác vụ được thực hiện song song, nên tốc độ của python trong hệ thống sẽ không thể so sánh được với các ngôn ngữ thông dụng khác đang được sử dụng như C/C++.



# Hello world

### Chương trình đầu tiên

Chương trình python được viết trong file .py

Nội dung chương trình in ra dòng hello world:

File: main.py

print("Hello world!")

Chạy chương trình trên cmd: python main.py

### Comment

Comment 1 dòng:

# comment

Comment nhiều dòng: dùng nháy kép hoặc nháy đơn đều được.

“““

Comment line 1

Comment line 2

”””

Hoặc

‘‘‘

Comment line 1

Comment line 2

’’’

## Đặt mã ký tự:

Tham khảo

<http://laptrinhcanban.com/python/nhap-mon-lap-trinh-python/kien-thuc-can-ban-ve-chuong-trinh-python/dat-ma-ky-trong-file-python/>

## print()

### Cú pháp:

print ( \*objects , sep=' ', end='\n', file=sys.stdout, flush=False )

Trong đó:

* \*objects : đối tượng (dữ liệu) cần in ra màn hình. Dấu \* có ý nghĩa là số nhiều và chúng ta cũng có thể chỉ định nhiều đối tượng khác nhau và in chúng cùng lúc ra màn hình.
* sep: đối tượng chỉ định sẽ được phân tách thành các phần nhỏ bằng ký tự phân tách sep trước khi được in, và mặc định giá trị này là một khoảng trắng ' '.
* end: giá trị cuối cùng được in ra màn hình, và mặc định giá trị này là ký tự xuống dòng \n. Đối số này sẽ quyết định việc in xuống dòng hay in không xuống dòng trong Python.
* file=sys.stdout: chỉ định lưu kết quả đầu ra vào bộ nhớ đệm sys.stdout.
* flush=False: chỉ định cưỡng chế lưu giữ kết quả vào bộ nhớ đệm, và giá trị mặc định là false, có nghĩa là KHÔNG lưu giữ kết quả vào bộ nhớ.

### Căn trái, phải, giữa:

Sử dụng ljust(), rjust(), center().

VD:

str1="học lập trình"

print("[" + str1.ljust(20) + "]")

#>> [học lập trình       ]

VD:

str1="học lập trình"

print("[" + str1.center(20) + "]")

#>> [   học lập trình    ]

### Chèn biến vào chuỗi sử dụng toán tử %

Cú pháp:

string % variable

VD:

s = 'Shin chan'

i = 2

print('Shin chan is %d years old' % i)

#>> Shin chan is 2 years old

print('%s is %d years old' % (s, i))

#>> Shin chan is 2 years old

### Định dạng số khi in ra màn hình trong Python.

VD:

print("{:,d}".format(1234567))

#>> 1,234,567

print("{:\_d}".format(1234567))

#>> 1\_234\_567

Chỉ định số chữ số đằng sau dấu phẩy và làm tròn số.

print("{:f}".format(1.2345))

#>>1.234500

print("{:.1f}".format(1.2345))

#>> 1.2

### In list, tuple và dictionary

VD:

l = [0, 1, 2]

print(l)

#>> [0, 1, 2]

t = (0, 1, 2)

print(t)

#>> (0, 1, 2)

d = { 1: 'red', 2: 'green' , 3: 'black'}

print(d)

#>> {1: 'red', 2: 'green', 3: 'black'}

Phân tách phần tử sử dụng ký hiệu \* vào trước list, tuple hay dictionary.

VD:

l = [0, 1, 2]

print(\*l)

#>> 0 1 2

t = (0, 1, 2)

print(\*t)

#>> 0 1 2

d = { 1: 'red', 2: 'green' , 3: 'black'}

print(\*d)

#>> 0 1 2

## Pprint(): prety-print

<http://laptrinhcanban.com/python/nhap-mon-lap-trinh-python/nhap-xuat-trong-python/ham-pprint-trong-python-va-cach-in-list-va-tu-dien-thong-minh/>

### So sánh với print:

Ví dụ, chúng ta có một list có cấu trúc phức tạp như sau:

l = [{'Name': 'Yamada', 'Age': 20, 'Points': [80, 20]},

{'Name': 'Kiyoshi', 'Age': 30, 'Points': [90, 10]},

{'Name': 'Honda', 'Age': 40, 'Points': [70, 30]}]

Kết quả khi sử dụng print

print(l)

#>> [{'Name': 'Yamada', 'Age': 20, 'Points': [80, 20]}, {'Name': 'Kiyoshi', 'Age': 30, 'Points': [90, 10]}, {'Name': 'Honda', 'Age': 40, 'Points': [70, 30]}]

Kết quả khi sử dụng pprint

import pprint

pprint.pprint(l)

#>> [{'Age': 20, 'Name': 'Yamada', 'Points': [80, 20]},

#>>  {'Age': 30, 'Name': 'Kiyoshi', 'Points': [90, 10]},

#>>  {'Age': 40, 'Name': 'Honda', 'Points': [70, 30]}]

### Cú pháp

pprint.pprint ( object, stream=None, indent=1, width=80, depth=None,compact=False, sort\_dicts=True )

Trong đó các đối số có tác dụng như sau:

* width : chỉ định chiều rộng đầu ra (số ký tự), mặc định bằng 80
* indent: chỉ định độ rộng thụt lề, mặc định bằng 1
* depth: chỉ định độ sâu của phần tử đầu ra
* compact: giảm thiểu ngắt dòng

### Chỉ định số ký tự tối đa trên mỗi dòng in | đối số width

VD:

pprint.pprint(l)

#>> [{'Age': 20, 'Name': 'Yamada', 'Points': [80, 20]},

#>>  {'Age': 30, 'Name': 'Kiyoshi', 'Points': [90, 10]},

#>>  {'Age': 40, 'Name': 'Honda', 'Points': [70, 30]}]

pprint.pprint(l, width=40)

#>> [{'Age': 20,

#>>  'Name': 'Yamada',

#>>  'Points': [80, 20]},

#>> {'Age': 30,

#>>  'Name': 'Kiyoshi',

#>>  'Points': [90, 10]},

#>> {'Age': 40,

#>>  'Name': 'Honda',

#>>  'Points': [70, 30]}]

pprint.pprint(l, width=10)

#>> [{'Age': 20,

#>>   'Name': 'Yamada',

#>>   'Points': [80,

#>>              20]},

#>>  {'Age': 30,

#>>   'Name': 'Kiyoshi',

#>>   'Points': [90,

#>>              10]},

#>>  {'Age': 40,

#>>   'Name': 'Honda',

#>>   'Points': [70,

#>>              30]}]

### Chỉ định số phần tử con tối đa được in trong dữ liệu lồng nhau | đối số depth

Độ sâu là số phần tử con tối đa được in của mỗi phần tử.

Nếu dữ liệu lồng nhau chứa số phần tử con vượt quá depth thì các phần tử con này sẽ được biểu thị bởi dấu ba chấm ... khi in ra màn hình như sau:

VD:

pprint.pprint(l, depth=1)

#>> [{...}, {...}, {...}]

pprint.pprint(l, depth=2)

#>> [{'Age': 20, 'Name': 'Yamada', 'Points': [...]},

#>>  {'Age': 30, 'Name': 'Kiyoshi', 'Points': [...]},

#>>  {'Age': 40, 'Name': 'Honda', 'Points': [...]}]

### Chỉ định độ thụt lề trên mỗi dòng in | đối số indent

VD:

pprint.pprint(l)

##> [{'Age': 20, 'Name': 'Yamada', 'Points': [80, 20]},

##>  {'Age': 30, 'Name': 'Kiyoshi', 'Points': [90, 10]},

##>  {'Age': 40, 'Name': 'Honda', 'Points': [70, 30]}]

pprint.pprint(l, indent=4)

##> [   {'Age': 20, 'Name': 'Yamada', 'Points': [80, 20]},

##>     {'Age': 30, 'Name': 'Kiyoshi', 'Points': [90, 10]},

##>     {'Age': 40, 'Name': 'Honda', 'Points': [70, 30]}]

### Giảm thiểu tối đa số lần ngắt dòng khi in các list có nhiều phần tử | Đối số compact

Với các list có quá nhiều phần tử, nếu tổng ký tự của các phần tử không vừa với độ dài của một dòng in thì python sẽ tự động ngắt dòng và in từng phần tử ra màn hình:

import pprint

l\_long = [list(range(10)), list(range(1000, 1010))]

print(l\_long)

#>> [[0, 1, 2, 3, 4, 5, 6, 7, 8, 9], [1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009]]

pprint.pprint(l\_long, width=40)

#>>[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9],

#>> [1000,

#>>  1001,

#>>  1002,

#>>  1003,

#>>  1004,

#>>  1005,

#>>  1006,

#>>  1007,

#>>  1008,

#>>  1009]]

Để có thể giảm thiểu tối đa số ngắt chuỗi khi in như trên, chúng ta có thể chỉ định đối số compact = True khi sử dụng với các list có quá nhiều phần tử trong nó.

pprint.pprint(l\_long, width=40, compact=True)

#>> [[0, 1, 2, 3, 4, 5, 6, 7, 8, 9],

#>>  [1000, 1001, 1002, 1003, 1004, 1005,

#>>   1006, 1007, 1008, 1009]]

## input()

input() gửi về kiểu dữ liệu string.

### Nhập dữ liệu

dulieu = input("Hãy nhập dữ liệu:")

#>> Hãy nhập dữ liệu:123abc

print(dulieu)

#>> 123abc

### Nhập cùng lúc nhiều dữ liệu trên một dòng

s = input(">>").split()

print(s)

>> 1 23 ab

['1', '23','ab']

### Nhập nhiều dữ liệu trên nhiều dòng

s = [input(">>") for i in range(3)]

print(s)

>> 1

>> 23

>> ab

['1', '23','ab']

### Nhập số

Input gửi về kiểu dữ liệu chuỗi. Để nhận số cần ép kiểu.

VD:

val = int(input('Enter number: '))

print(val)

print(type(val))

# Enter number: 50

# 50

# <class 'int'>

# Số trong python

Kiểu dữ liệu số trong python cũng thuộc dạng Literal - một giá trị mà nó thể hiện chính nó.

## Số nguyên int

Kiểu dữ liệu số nguyên trong python bao gồm số nguyên dương (1,2,3…), số nguyên âm (-1,-2,-3…) và số 0.

Khi viết số nguyên trong một chương trình, chúng ta viết giá trị của chúng như ví dụ dưới đây:

# Số nguyên dương

1

23

# Số nguyên âm

-1

-123

# Số 0

0

Python cho phép chúng ta thêm dấu gạch chân \_ vào vị trí bất kỳ để phân chia chữ số cho dễ nhìn khi viết kiểu dữ liệu số nguyên int như sau:

10000

10\_000

1000\_0

1\_0000

### Biểu diễn số nguyên dưới dạng nhị phân (binary) , 8 (octal) và 16 (Hex – hexadecimal):

0b0110 # nhị phân

0o1234 # bát phân

0xFFAA # Thập lục phân

### Biến đổi, ép kiểu dữ liệu về số nguyên

# Gán chuỗi "15" vào biến a

a="15"

# Chuyển a từ kiểu dữ liệu chuỗi thành kiểu dữ liệu số nguyên

# Sau đó gán giá trị này vào b

b = int(a)

print(b)

#>>15

# Chuyển b từ số thập phân sang số nhị phân

print(bin(b))

#>> 0b1111

# Chuyển b từ số thập phân sang số bát phân

print(oct(b))

#>> 0o17

# Chuyển b từ số thập phân sang số thập lục phân

print(hex(b))

#>> 0xf

## Kiểu dữ liệu số thực float trong python | float python là gì

Kiểu dữ liệu số thực trong python hay còn gọi là float python là kiểu số được biểu diễn dưới dạng số thực dấu phẩy động( tiếng anh : Floating point number; tiếng Nhật:浮動小数点 ).

VD

12.49

-19.5

Chúng ta cũng có thể dùng dấu gạch chân \_ vào vị trí bất kỳ để phân chia chữ số cho dễ nhìn khi viết kiểu dữ liệu số thực float như sau:

1\_223.496\_6

-1\_143\_249.5

### Sử dụng số e

VD:

8.5e+7

2.34e-12

Trong đó, 8.5e+7 = và 2.34e-12 =.

### Ép kiểu dữ liệu về kiểu float

# Gán chuỗi "15" vào biến a

a="15"

# Chuyển a từ kiểu dữ liệu chuỗi thành kiểu dữ liệu số thực

# Sau đó gán giá trị này vào b

b = float(a)

print(b)

#>>15.0

## Kiểu dữ liệu số phức complex trong python

Kiểu dữ liệu số phức trong python là các số được biểu thị bằng phần thực và phần ảo như dưới đây:

a + bj

Nếu bạn muốn viết kiểu dữ liệu số phức trong Python, hãy tham khảo như ví dụ dưới đây:

4.2 + 5j

3 + 4j

4j

(3.2 + 4j)

VD

print("Số nguyên:",12345)

#>> Số nguyên: 12345

print("Số thực  :",2.34e-12)

#>> Số thực  : 2.34e-12

print("Số phức  :", 4.2 + 5j)

#>> Số phức  : (4.2+5j)

## Toán tử số học

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Toán tử | Ý nghĩa |  |
| 1 | (expressions...), | Hiển thị tuple, list, danh sách, từ điển |  |
|  | [expressions...], |  |  |
|  | {key: value...}, |  |  |
|  | {expressions...} |  |  |
| 2 | x[index], x[index:index], | Đăng ký, cắt, gọi, tham chiếu thuộc tính |  |
|  | x(arguments...), x.attribute |  |  |
| 3 | await x | Biểu thức await |  |
| 4 | \*\* | Lũy thừa |  |
| 5 | +x, -x, ~x | Giữ dấu, Đổi dấu,bit NOT |  |

Sử dụng toán tử số học trong python khi tính toán giữa các loại số khác nhau.

Kết quả trả về sẽ là kiểu dữ liệu có ít hạn mức nhất trong các kiểu dữ liệu ban đầu.

Thứ tự kiểu dữ liệu số có hạn mức từ thấp tới cao trong python là như sau:

Số phức < Số thực < Số nguyên

Ví dụ như khi sử dụng toán tử số học để tính toán giữa số nguyên và số thực thì kết quả trả về sẽ là số thực. Hay giữa số thực và số phức thì kết quả trả về sẽ là số phức.

VD

print("Tinh toan giua cac loai so khac nhau")

print("2 + 1.485 = " + str(2 + 1.485))

print("9 \* 5.23 = " + str(9 \* 5.23))

print("16 / 4.23 = " + str(16 / 4.23))

print("16 \*\* 4.23 = " + str(16 \*\* 4.23))

Tinh toan giua cac loai so khac nhau

2 + 1.485 = 3.4850000000000003

9 \* 5.23 = 47.07000000000001

16 / 4.23 = 3.7825059101654843

16 \*\* 4.23 = 124001.66787161745

## Toán tử bit

x & y AND

x | y OR

~x NOT

x ^ y XOR

x << n Phép toán dịch bit trái

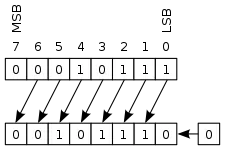
x >> n Phép toán dịch bit phải

### Phép dịch bit

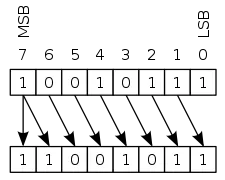
Phép toán dịch bit trong python

Khi dịch bit, các số được dịch chuyển qua đầu hoặc đuôi sẽ bị loại bỏ.

Khi dịch chuyển bit qua trái, số 0 sẽ được thêm vào bên phải để lấp chỗ trống để lại.



Khi dịch chuyển bit qua phải, bit thể hiện dấu sẽ được thêm vào bên trái để lấp chỗ trống để lại, và dấu của số sẽ được giữ nguyên.



Tham khảo: <http://laptrinhcanban.com/python/nhap-mon-lap-trinh-python/so-trong-python/toan-tu-thao-tac-bit-trong-python/>

## Thứ tự ưu tiên toán tử

### Bảng ưu tiên

|  |  |  |
| --- | --- | --- |
| STT | Toán tử | Ý nghĩa |
| 1 | (expressions...), | Hiển thị tuple, list, danh sách, từ điển |
|  | [expressions...], |  |
|  | {key: value...}, |  |
|  | {expressions...} |  |
| 2 | x[index], x[index:index], | Đăng ký, cắt, gọi, tham chiếu thuộc tính |
|  | x(arguments...), x.attribute |  |
| 3 | await x | Biểu thức await |
| 4 | \*\* | Lũy thừa |
| 5 | +x, -x, ~x | Giữ dấu, Đổi dấu,bit NOT |
| 6 | \*, @, /, //, % | Nhân, matrix, chia, chia làm tròn và chia lấy dư |
| 7 | +, - | Cộng trừ |
| 8 | <<, >> | Phép toán dịch bit |
| 9 | & | bit AND |
| 10 | ^ | bit XOR |
| 11 |  | bit OR |
| 12 | in, not in, is, is not, | So sánh |
|  | <, <=, >, >=, !=, == |  |
| 13 | not x | Boolean NOT |
| 14 | and | Boolean AND |
| 15 | or | Boolean OR |
| 16 | if -- else | Biểu thức điều kiện |
| 17 | lambda | Biểu thức Lambda |

## Làm tròn số

### Round()

round(number , ndigits)

Trong đó

* number là số cần làm tròn
* ndigits là vị trí thập phân muốn làm tròn. Nếu không chỉ định ndigits thì mặc định ndigits bằng 0.

Nếu không chỉ định ndigits, kết quả trả về kiểu int, ngược lại, kết quả trả về kiểu float.

VD:

a = 123.456

b= round(a, 2)

print(type(b), b)

b= round(a, 0)

print(type(b), b)

b= round(a)

print(type(b), b)

b= round(a, -2)

print(type(b), b)

<class 'float'> 123.46

<class 'float'> 123.0

<class 'int'> 123

<class 'float'> 100.0

Tham khảo: <http://laptrinhcanban.com/python/nhap-mon-lap-trinh-python/so-trong-python/lam-tron-so-trong-python-bang-ham-round/>

## Số pi

VD:

import math

math.pi

Lấy số pi trong python

import math

math.piimport math

print(math.pi)

##> 3.141592653589793

Tính sin bằng số pi trong python

import math

print("sin(π/4) = ",math.sin(math.pi/4))

##> sin(π/4) =  0.7071067811865475

Tính cos bằng số pi trong python

import math

print("cos(π/4) = ",math.cos(math.pi/4))

##> cos(π/4) =  0.7071067811865476

Tính tan bằng số pi trong python

import math

print("tan(π/4) = ",math.tan(math.pi/4))

#>> tan(π/4) =  0.9999999999999999

Tính chu vi hình tròn bằng số pi trong python

import math

r = 8

print("Chu vi = ", 2 \* r \* math.pi)

##> Chu vi =  50.26548245743669

Tính diện tích hình tròn bằng số pi trong python

import math

r = 8

print("Dien tich = ", r \* r \* math.pi)

##> Dien tich =  201.06192982974676

## Kiểm tra số chẵn lẻ

C1: chia phần trăm cho 2

def check\_odd\_even(n):

    #flag = 1 => số lẻ

    #flag = 0 => số chẵn

    flag = 1

    if( n % 2 == 0):

        flag= 0

    return flag

C2: AND với 1

def check\_odd\_even\_bit(n):

    #flag = 1 => số lẻ

    #flag = 0 => số chẵn

    flag = 0

    if( n & 1 == 1 ):

        flag= 1

    return flag

## Kiểm tra số nguyên, số thực

### Kiểm tra số nguyên số thực trong python | Hàm isinstance()

Cú pháp:

isinstance(num, type)

Trong đó num chính là số cần kiểm tra , và type là kiểu cần kiểm tra. type có thể là int hoặc float.

Hàm isinstance() sẽ trả về True nếu số đã cho thuộc kiểu type chỉ định , và ngược lại trả về False trong các trường hợp còn lại.

VD:

i = 6

f = 6.78

print(isinstance(i, int))

# True

print(isinstance(i, float))

# False

print(isinstance(f, int))

# False

print(isinstance(f, float))

# True

Lưu ý là trong trường hợp số nguyên được viết có phần thập phân, ví dụ như số 1 viết thành 1.0 hoặc 1.000 chẳng hạn thì hàm isinstance() sẽ coi đây là số thực trong Python.

print(isinstance(1, int))

# True

print(isinstance(1.00, int))

# False

print(isinstance(1.00, float))

# True

### Kiểm tra số thực có phải là số nguyên trong python | Phương thức is\_integer()

is\_integer() là một phương thức cài sẵn trong kiểu dữ liệu float, do đó chúng ta có thể sử dụng phương thức này với một số thực bất kỳ để kiểm tra coi nó có phải là số nguyên hay không.

Phương thức float.is\_integer() sẽ trả về True nếu số đã cho thuộc là số nguyên, và ngược lại trả về False trong các trường hợp còn lại.

VD:

f1 = 1.234

print(f1.is\_integer())

# False

f2 = 100.0

print(f2.is\_integer())

# True

## Kiểm tra số nguyên tố

Số nguyên tố là số lớn hơn 1 và chỉ có đúng hai ước số là 1 và chính nó.

Để làm được điều này, chúng ta chỉ cần kiểm tra trong phạm vi lớn hơn 1 và nhỏ hơn số đã cho có tồn tại ước số nào không, và nếu tồn tại dù chỉ một ước số thì có thể khẳng định số đã cho không phải là số nguyên tố.

VD

def check\_prime\_number(n):

    #flag = 0 => không phải số nguyên tố

    #flag = 1 => số nguyên tố

    flag = 1

    if (n <2):

        flag = 0

        return flag  #Số nhỏ hơn 2 không phải số nguyên tố => trả về 0

    #Sử dụng vòng lặp while để kiểm tra có tồn tại ước số nào khác không

    for p in range(2, n):

        if n % p == 0:

            flag = 0

            break #Chỉ cần tìm thấy 1 ước số là đủ và thoát vòng lặp

    return flag

## Kiểm tra số chính phương

Số chính phương hay còn gọi là số hình vuông là số tự nhiên có căn bậc hai là một số tự nhiên, hay nói cách khác, số chính phương bằng bình phương của một số tự nhiên

Cách 1: Kiểm tra các số trong khoảng [0, n).

def find\_square\_number(n):

    #flag = 1 => số chính phương

    #flag = 0 => không phải số chính phương

    flag = 0

    #Tìm số bất kỳ nhỏ hơn hoặc bằng n mà bình phương bằng n

    if any(i\*\*2 == n for i in range(n+1)):

        flag = 1

    return flag

Cách 2: Kiểm tra xem căn bậc 2 của n có phải là số nguyên hay không.

def find\_square\_number\_2(n):

    #flag = 1 => số chính phương

    #flag = 0 => không phải số chính phương

    flag = 0

    #Kiểm tra căn bậc 2 của số đó có phải số tự nhiên hay không

    if (n \*\* .5).is\_integer(): ##  if (math.sqrt(n)).is\_integer():

        flag = 1

    return flag

## Số hoàn hảo

Số hoàn hảo (hay còn gọi là số hoàn chỉnh, số hoàn thiện hoặc số hoàn thành) là một số nguyên dương mà tổng các ước nguyên dương chính thức của nó (số nguyên dương bị nó chia hết ngoại trừ nó) bằng chính nó.

def find\_perfect\_num(n):

    ## flag = 1 => số số hoàn hảo

    ## flag = 0 => không phải số hoàn hảo

    flag = 0

    #Tìm ước số

    divisor= [i for i in range(1,n) if n % i == 0]

    #Tìm tổng ước số

    total = 0

    for i in range(len(divisor)):

        total += divisor[i]

    #So sánh

    if total == n:

        flag = 1

    return flag

## Số đối xứng

số nguyên n gọi là số đối xứng nếu đọc từ trái qua phải, hay từ phải qua trái đều được số giống nhau. Ví dụ: 11,121,101 là các số đối xứng.

#Hàm kiểm tra số đối xứng trong python

def symmetrical\_num(n):

    # flag = 1 => số đối xứng

    # flag = 0 => không phải số đối xứng

    n = str(n)

    flag =0

    if ( n[::-1] == n):

      flag = 1

    return flag

# Biến

## Khái niệm

Có hai trường phái định nghĩa biến trong python như sau:

Trường phái đầu tiên coi biến trong python giống như một cái hộp để lưu trữ dữ liệu. Các dữ liệu này có thể là số hoặc chuỗi mà bạn có thể ưu trữ vào biến và sử dụng nhiều lần. Kết quả của các phép xử lý như tính toán giá trị số, chỉnh sửa chuỗi ký tự sẽ tạm thời được giữ vào biến và dùng để sử dụng cho chương trình sau này.

Trường phái thứ hai coi biến trong python giống như thẻ ghi địa chỉ của dữ liệu. Các dữ liệu được lưu giữ tại các vị trí riêng biệt trong bộ nhớ với địa chỉ khác nhau, và biến trong python là thẻ dùng để ghi địa chỉ của dữ liệu đó trong bộ nhớ. Khi sử dụng dữ liệu, chúng ta sẽ truy cập vào địa chỉ được ghi trên biến của dữ liệu đó.

## Khởi tạo

Cú pháp:

name = value

Trong đó

* name là tên biến được đặt theo quy tắc đặt tên biến trong Python mà bạn đã học trong bài Biến trong python là gì.
* Dấu = dùng để gán biến trong python. Giá trị value bên phải dấu = sẽ được gán vào biến ở bên trái.
* value là giá trị dùng để khai báo biến. Giá trị này có thể là chuỗi, số, hoặc các kiểu đối tượng khác trong python.

Khi bạn khởi tạo biến trong python, Python sẽ tự động xác định kiểu dữ liệu của biến được khai báo.

Kiểm tra kiểu dữ liệu sau khi khai báo biến bằng hàm typle()

myvar = 1234

print(type(myvar))

# <class 'int'>

mystr = "hello"

print(type(mystr))

# <class 'str'>

print(symmetrical\_num(n))

## Xoá biến

Cú pháp:

del tên biến

VD:

num = 10

print(num)

del num

# Lệnh điều kiện

## If

Cú pháp:

if biểu-thức-điều-kiện:

câu lệnh 1

câu lệnh 2

câu lệnh 3

if biểu-thức-điều-kiện:

câu lệnh xử lý nếu biểu-thức-điều-kiện là True (đúng)

else:

câu lệnh xử lý nếu biểu-thức-điều-kiện là False (sai)

if biểu-thức-điều-kiện-1:

câu lệnh xử lý nếu biểu-thức-điều-kiện-1 là True (đúng)

elif biểu-thức-điều-kiện-2:

câu lệnh xử lý nếu biểu-thức-điều-kiện-2 là True (đúng)

elif biểu-thức-điều-kiện-3:

câu lệnh xử lý nếu biểu-thức-điều-kiện-3 là True (đúng)

…

else:

câu lệnh xử lý nếu tất cả các biểu thức điều kiện đều False (sai)

## Toán tử so sánh

|  |  |  |
| --- | --- | --- |
| Toán tử | Biểu thức điều kiện | Ý nghĩa |
| == | x == y | x và y bằng nhau |
| != | x != y | x và y không bằng nhau |
| > | x > y | x lớn hơn y |
| < | x < y | x nhỏ hơn y |
| >= | x >= y | x bằng hoặc lớn hơn y |
| <= | x <= y | x bằng hoặc nhỏ hơn y |
| in | x in y | phần tử x tồn tại trong y |
| not in | x not in y | phần tử x không tồn tại trong y |

Tham khảo: <http://laptrinhcanban.com/python/nhap-mon-lap-trinh-python/lenh-dieu-kien-trong-python/toan-tu-so-sanh-trong-python/>

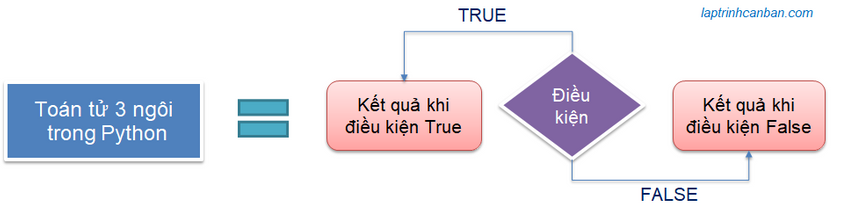
## Toán tử logic

|  |  |  |
| --- | --- | --- |
| Toán tử | Biểu thức | Ý nghĩa |
| AND | X and Y | True nếu cả X và Y đều đúng |
| OR | X or Y | True nếu ít nhất một trong hai vế X hoặc Y đúng |
| NOT | not X | True nếu X sai và False nếu X đúng |

Thứ tự ưu tiên

not > and > or

## Toán tử 3 ngôi



Cú pháp:

(true\_value) if (condition) else (false\_value);

VD:

x = "OK" if n == 10 else "NG"

# Tham khảo

Lập trình căn bản:

<https://laptrinhcanban.com/cpp/lap-trinh-cpp-co-ban/>