

# Ngôn Ngữ Lập Trình Python

## Biến, Từ Khóa, Định Danh, Kiểu Dữ Liệu

# Nội Dung

- Từ Khóa
- Định Danh
- Chú Thích
- Biến
- Nhập dữ liệu từ bàn phím/ Xuất dữ liệu
- Câu lệnh / Khối lệnh
- Toán Tử Trong Python
- Kiểu số

# Từ Khóa Trong Python

- Những từ chỉ dành riêng cho Python
- Trong Python, ngoại trừ **True**, **False** và **None** được viết hoa ra thì các keyword khác đều được viết dưới dạng chữ thường, đây là điều bắt buộc.

• False	• class	• finally	• is	• return
• None	• continue	• for	• lambda	• try
• True	• def	• from	• nonlocal	• while
• and	• del	• global	• not	• with
• as	• elif	• if	• or	• yield
• assert	• else	• import	• pass	
• break	• except	• in	• raise	

# Định Danh Trong Python

- Định danh là tên được đặt cho các thực thể như class, function, biến,... trong Python. Nó giúp phân biệt thực thể này với thực thể khác.
- Quy tắc viết định danh và vài tips cho định danh trong Python:
  - Định danh có thể là sự kết hợp giữa các chữ cái viết thường (**từ a đến z**) hoặc viết hoa (**A đến Z**) hoặc các số (**từ 0 đến 9**) hoặc dấu gạch dưới (\_). Định danh hợp lệ sẽ giống như thế này: `bien_1`, `tinh_tong_0_9`, `firstClass`.
  - Định danh **không thể bắt đầu bằng một chữ số**, ví dụ `1bien` là không hợp lệ, nhưng `bien1` thì đúng.
  - Định danh phải **khác các keyword**. Bạn thử nhập `and = 1` rồi chạy sẽ xuất hiện thông báo lỗi `"SyntaxError: invalid syntax"`.
  - Python **không hỗ trợ các ký tự đặc biệt** như `!`, `@`, `#`, `$`, `%`,... trong định danh. Nếu cố tình gán các ký tự này trong định danh bạn cũng sẽ nhận được thông báo lỗi `"SyntaxError: invalid syntax"` hoặc `"NameError:..."`.
  - Python là ngôn ngữ lập trình **phân biệt chữ hoa, chữ thường**, nghĩa là `bien` và `Bien` là không giống nhau.

# Chú Thích (comment)

- Python sử dụng **kí tự #** để chú thích các đoạn code
- Tất cả các nội dung sau **kí tự #** sẽ không được dịch

tor Run Tools VCS Window Help

Comment.py x

```
1 # đây là lệnh dùng để kiểm tra kiểu dữ liệu của biến
2 x=113
3 print(type(x))
```

# Chú Thích (comment)

- Ghi chú nhiều dòng: Dùng `""" """` (3 cặp nháy đôi) hoặc `''' '''` (3 cặp nháy đơn)

```
1  """
2  Giải phương trình bậc 1: ax+b=0
3  Có 3 trường hợp để biện luận
4  Nếu hệ số a =0 và hệ số b=0 ==> vô số nghiệm
5  Nếu hệ số a =0 và hệ số b !=0 ==> vô nghiệm
6  Nếu hệ số a !=0 ==> có nghiệm -b/a
7  """
8  a = 0
9  b = 113
10 if a == 0 and b == 0:
11     print("Vô số nghiệm")
12 elif a == 0 and b != 0:
13     print("Vô nghiệm")
14 else:
15     print("Có No X=", -b/a)
```

# Chú Thích (comment)

- Ví dụ dùng 3 cặp nháy đơn:

```
1  '''
2  Đây là lệnh kiểm tra năm nhuận year
3  Năm nhuận là năm chia hết cho 4 nhưng không chia hết cho 100 hoặc
4  '''
5  year=2016
6  if (year % 4==0 and year %100 !=0) or year % 400 ==0:
7      print(year, " Là năm nhuận")
8  else:
9      print(year, " KO là năm nhuận")
```

# Biến Trong Python

- Một biến Python là một khu vực bộ nhớ được dành riêng để lưu trữ các giá trị.
- Mỗi giá trị trong Python có một kiểu dữ liệu. Các kiểu dữ liệu khác nhau trong Python là
  - kiểu số (number)
  - kiểu danh sách (list)
  - kiểu bộ (tuple)
  - kiểu chuỗi (string)
  - kiểu từ điển (dictionary)
  - ...



# Biến trong Python

- Biến trong python:
  - Có tên, phân biệt chữ hoa/thường
  - Không cần khai báo trước
  - Không cần chỉ ra kiểu dữ liệu
  - Có thể thay đổi sang kiểu dữ liệu khác
  - Nên gán giá trị ngay khi bắt đầu xuất hiện
  - Không được trùng với từ khóa
- **Cách khai báo và sử dụng biến**
  - Ví dụ : **n = 12** # biến n là kiểu nguyên  
**n = n + 0.1** # biến n chuyển sang kiểu thực  
print(n)

# Biến trong Python

```
1 # Declare a variable and initialize it
2 f = 0
3 print(f)
4 # re-declaring the variable works
5 f = 'guru99'
6 print(f)
7
8
9
10
11
12
```

Bạn có thể khai báo lại một biến dù đã khai báo trước đó. Điều này hoàn toàn được chấp nhận.

Run Python5.1

"C:\Users\DK...  
5/PythonCode5/1

0  
guru99

thon Test\Py

# Biến trong Python

- Tất cả mọi biến trong python đều là các đối tượng, vì thế nó có kiểu (type) và vị trí trong bộ nhớ (id)

```
C:\Users\sony>python
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] :: Ana
conda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a= 10
>>> b = 1.535
>>> c= "hello"
>>>
>>> id(a)
8791359468656
>>> id(a),id(b),id(c)
(8791359468656, 4366984, 43086488)
>>> type(a),type(b),type(c)
(<class 'int'>, <class 'float'>, <class 'str'>)
>>>
```

# Biến trong Python

## Gán nhiều giá trị:

- Trong Python bạn có thể thực hiện gán nhiều giá trị trong một lệnh như sau:

**s, x, f = "Vang", 3, 5.5**

- Nếu muốn gán giá trị giống nhau cho nhiều biến thì có thể viết lệnh như sau:

**s, x, f = 3**

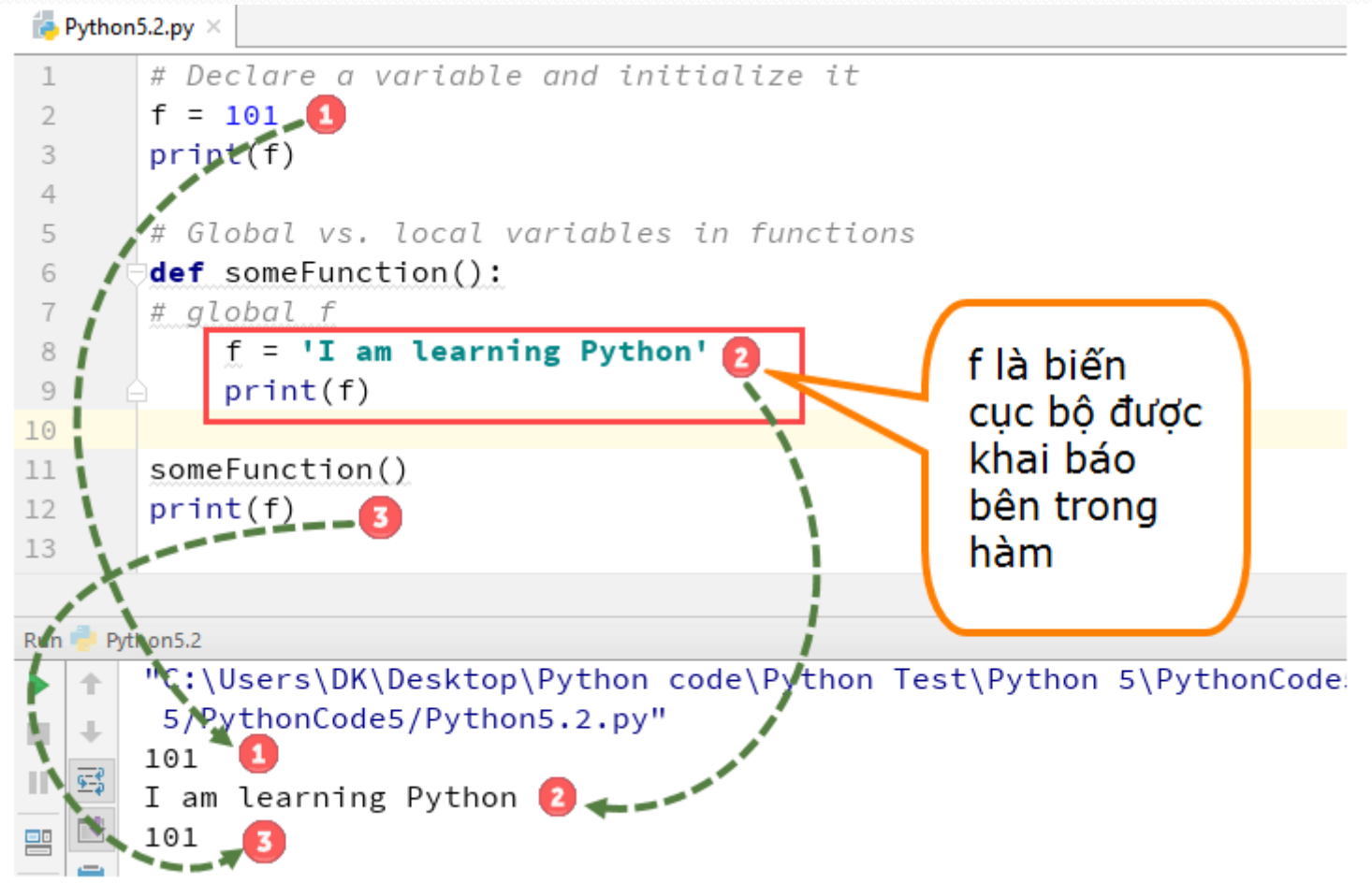
# Biến trong Python

## Biến toàn cục và biến cục bộ

- Trong Python khi bạn muốn sử dụng **cùng một biến cho toàn bộ chương trình hoặc mô-đun**, bạn cần khai báo nó dưới dạng biến toàn cục.
- Trường hợp bạn muốn sử dụng **biến trong một hàm hoặc phương thức cụ thể**, bạn hãy sử dụng biến cục bộ.

# Biến trong Python

```
# Declare a variable and initialize it
f = 101
print(f)
# Global vs. local variables in functions
def someFunction():
    f = 'I am learning Python'
    print(f)
someFunction()
print(f)
```



The screenshot shows a Python IDE window titled "Python5.2.py". The code is as follows:

```
1 # Declare a variable and initialize it
2 f = 101
3 print(f)
4
5 # Global vs. local variables in functions
6 def someFunction():
7     # global f
8     f = 'I am learning Python'
9     print(f)
10
11 someFunction()
12 print(f)
13
```

Annotations on the code:

- A red circle with the number 1 is next to line 2 (`f = 101`).
- A red circle with the number 2 is next to line 8 (`f = 'I am learning Python'`), which is highlighted with a red box.
- A red circle with the number 3 is next to line 12 (`print(f)`).

A dashed green line connects the red circle 1 to the first "101" in the output. Another dashed green line connects the red circle 2 to the "I am learning Python" in the output. A third dashed green line connects the red circle 3 to the second "101" in the output.

An orange callout box points to the red box around line 8, containing the text: "f là biến cục bộ được khai báo bên trong hàm".

The output window shows the following results:

```
"C:\Users\DK\Desktop\Python code\Python Test\Python 5\PythonCode.
5\PythonCode5\Python5.2.py"
101
I am learning Python
101
```

Red circles with numbers 1, 2, and 3 are placed next to the output lines "101", "I am learning Python", and "101" respectively, corresponding to the annotations in the code.

# Biến trong Python

global f

```
1 f = 101;
2 print(f) ①
3
4 # Global vs. local variables in functions
5 def someFunction():
6     global f ②
7     print(f)
8     f = "changing global variable"
9
10 someFunction()
11 print(f) ③
12
13
14
```

someFunction()

Run Python5.3

"C:\Users\DK\Desktop\Python code\Python Test\Python 5\PythonCode5\Python5.3.py"

101

101

changing global variable

Giờ chúng ta đang truy xuất và thay đổi giá trị của biến toàn cục f

# Biến trong Python

- Dữ liệu kiểu chuỗi rất quan trọng trong lập trình python
- Khai báo dữ liệu kiểu chuỗi có thể nằm bên trong cặp nháy đơn ('), hoặc nháy kép (") hoặc 3 dấu nháy kép liên tiếp (""")
- Ví dụ:

**S\_name= 'AN'**

# chuỗi trong nó có chứa dấu nháy đơn

**S\_sentence = " Hello Hello"**

# chuỗi có nội dung nằm trên 2 dòng

**S\_twoline= """This string has  
multiple lines in it"""**



# Xuất Dữ Liệu

- Sử dụng hàm **print** để in dữ liệu ra màn hình

```
>>> print(42)
42
>>> print("a = ", a)
a = 3.564
>>> print("a = \n", a)
a =
 3.564
>>> print("a","b")
a b
>>> print("a","b",sep="")
ab
>>> print(192,168,178,42,sep=".")
192.168.178.42
>>> print("a","b",sep=":-)")
a:-)b
```

# Nhập Dữ Liệu

- Sử dụng hàm **input** để nhập dữ liệu từ bàn phím

```
name = input("What's your name? ")
print("Nice to meet you " + name + "!")
age = input("Your age? ")
print("You are already " + age + " years old, " + name + "!")
```

```
In [5]: runfile('C:/Users/sony/Desktop/
python_co_ban/test/Test1.py', wdir='C:/Users/
sony/Desktop/python_co_ban/test')
```

```
What's your name? Dat
Nice to meet you Dat!
```

```
Your age? 30
You are already 30 years old, Dat!
```

# Ví Dụ Minh Họa

```
a = float(input("A = "))
b = float(input("B = "))
c = float(input("C = "))
delta = b*b-4*a*c
if delta==0:
    print("Nghiem kep: x = ", str(-b/2/a))
if delta<0:
    print("Phuong trinh vo nghiem")
if delta>0:
    print("X1 = " + str((-b+delta**0.5)/2/a))
    print("X2 = " + str((-b-delta**0.5)/2/a))
```

Nhập a,b,c kiểu  
số thực và tính  
delta

Biện luận các  
trường hợp của  
delta

Các khối lệnh con  
được viết thụt vào  
so với khối cha

Tính căn bậc 2  
bằng phép lũy  
thừa 0.5

# Câu lệnh trong Python

- Trong Python, một câu lệnh được kết thúc bằng ký tự dòng mới, nghĩa là một câu lệnh sẽ kết thúc khi bạn xuống dòng.
- Chúng ta có thể mở rộng câu lệnh trên nhiều dòng với ký tự tiếp tục dòng lệnh (\).

```
sum = 1 + 3 + 5 + \  
      7 + 9 + 11 + \  
      13 + 15 + 17
```

# Câu lệnh trong Python

- Trong Python, viết tiếp câu lệnh thường sử dụng dấu ngoặc đơn (), ngoặc vuông [] hoặc ngoặc nhọn {}

```
sum = (1 + 3 + 5 +  
       7 + 9 + 11 +  
       13 + 15 + 17)
```

Dấu () ở đây ngầm ý tiếp tục dòng lệnh

```
mau_sac = {"vang",  
           "xanh",  
           "cam"}
```

- Bạn cũng có thể đặt nhiều lệnh trên cùng một dòng, phân cách nhau bởi dấu **chấm phẩy ;**

```
a = 1; b = 2; c = 3
```

# Canh lề trong Python

- C, C++ hay Java bạn sẽ biết rằng những ngôn ngữ lập trình này sử dụng { } để xác định các khối code.
- Trong Python thì khác, những khối lệnh sẽ được nhận biết thông qua canh lề. Đó là lý do vì sao canh lề trong Python vô cùng quan trọng, nếu bạn thừa hoặc thiếu khoảng trắng là sẽ bị báo lỗi ngay.
  - Một khối code (thường là khối lệnh của hàm, vòng lặp, if ...) bắt đầu với canh lề và kết thúc với dòng đầu tiên không canh lề.

```
for i in range(1,11):  
    print(i)  
    if i == 5:  
        break  
print("hello")
```

# Canh lề trong Python

- Nếu lệnh trên được viết thành:

```
for i in range(1,11):  
    print(i)  
    if i == 5:  
        break
```

- Bạn sẽ nhận được thông báo lỗi ngay lập tức, và lỗi sẽ hiện trước lệnh `print(i)`.

```
30  
31 for i in range(1,11):  
32 print(i)  
33     if i == 5:  
34         break
```

```
File "C:/Users/sony/Desktop/python_co_ban/  
test/Test1.py", line 32  
    print(i)  
    ^  
IndentationError: expected an indented block
```

# Toán Tử Trong Python

- Python hỗ trợ nhiều phép toán số, logic, gán, so sánh và phép toán bit
- **Toán tử số học:** **+, -, \*, /, %, \*\*** ( %: phép chia lấy dư ; \*\*: lũy thừa)

Ví dụ :  $2**3 = 8$  ;  $3**3 = 27$

- Python có 2 phép chia:
  - chia đúng (/):  $10/3$  # 3.3333333333333335
  - chia nguyên (//):  $10/3$  # 3 (nhanh hơn phép /)

- **Các phép so sánh:** **<, <=, >, >=, !=, ==**

```
In [2]: runfile('C:/Users/sony/Desktop/
python_co_ban/test/Test1.py', wdir='C:/Users/
sony/Desktop/python_co_ban/test')
x > y is False
```

```
x = 4
y = 5
print('x > y is', x>y)
```



# Toán Tử Trong Python

- **Toán tử gán:** Các toán tử gán khác nhau được sử dụng trong Python là (=, +=, -=, \*=, /=, ...)

```
num1 = 4
num2 = 5

print(("Line 1 - Value of num1 : ", num1))
print(("Line 2 - Value of num2 : ", num2))
```

```
num1 = 4
num2 = 5

res = num1 + num2
res += num1

print(("Line 1 - Result of + is ", res))
```

```
In [3]: runfile('C:/Users/sony/Desktop/
python_co_ban/test/Test1.py', wdir='C:/Users/
sony/Desktop/python_co_ban/test')
('Line 1 - Result of + is ', 13)
```

# Toán Tử Trong Python

- Toán tử logic: **and, or, not**

```
a = True
b = False

print(('a and b is', a and b))
print(('a or b is', a or b))
print(('not a is', not a))
```

```
In [4]: runfile('C:/Users/son/python_co_ban/test/Test1.py'
sony/Desktop/python_co_ban/t
('a and b is', False)
('a or b is', True)
('not a is', False)
```

# Toán Tử Trong Python

- **Toán tử thành viên** : Các toán tử này kiểm tra tư cách thành viên trong một tập như danh sách, chuỗi hoặc tuple. Có hai toán tử thành viên được sử dụng trong Python là (**in**, **not in**). Kết quả trả về phụ thuộc vào việc biến có tồn tại trong chuỗi hoặc tập cho trước hay không.

```
x = 4
y = 8
list = [1, 2, 3, 4, 5 ];
if ( x in list ):
    print("Line 1 - x is available in the given list")
else:
    print("Line 1 - x is not available in the given list")
if ( y not in list ):
    print("Line 2 - y is not available in the given list")
else:
    print("Line 2 - y is available in the given list")
```

```
In [5]: runfile('C:/Users/sony/Desktop/
python_co_ban/test/Test1.py', wdir='C:/Users/
sony/Desktop/python_co_ban/test')
Line 1 - x is available in the given list
Line 2 - y is not available in the given list
```

# Kiểu Dữ Liệu: Number

- Python hỗ trợ số nguyên, số thập phân và số phức, chúng lần lượt được định nghĩa là các lớp int, float, complex trong Python
- Số nguyên trong Python không bị giới hạn độ dài, số thập phân bị giới hạn đến 16 số sau dấu thập phân.

Tiền tố hệ số cho các số Python:

Hệ thống số	Tiền tố
Hệ nhị phân	'0b' hoặc '0B'
Hệ bát phân	'0o' hoặc '0O'
Hệ thập lục phân	'0x' hoặc '0X'

(Bạn đặt tiền tố nhưng không có dấu ' ')

# Kiểu Dữ Liệu: Number

- Python viết số nguyên theo nhiều hệ cơ số

▪ A = 1234	# hệ cơ số 10
▪ B = 0xAF1	# hệ cơ số 16
▪ C = 0o772	# hệ cơ số 8
▪ D = 0b1001	# hệ cơ số 2

```
A = 1234
print("A:", type(A), A)
B = 0x1A
print("B:", type(B), B)
C = 0o720
print("C:", type(C), C)
D = 0b1010
print("D:", type(D), D)
```

```
In [26]: runfile('C:/Users/son
python_co_ban/test/Test1.py',
sony/Desktop/python_co_ban/tes
A: <class 'int'> 1234
B: <class 'int'> 26
C: <class 'int'> 464
D: <class 'int'> 10
```

# Kiểu Dữ Liệu: Number

- Chuyển đổi từ số nguyên thành string ở các hệ cơ số khác nhau

▪ <code>K = str(1234)</code>	# chuyển thành str ở hệ cơ số 10
▪ <code>L = hex(1234)</code>	# chuyển thành str ở hệ cơ số 16
▪ <code>M = oct(1234)</code>	# chuyển thành str ở hệ cơ số 8
▪ <code>N = bin(1234)</code>	# chuyển thành str ở hệ cơ số 2

```
K = str(1234)
print("K:", type(K), K)
L = hex(1234)
print("L:", type(L), L)
M = oct(1234)
print("M:", type(M), M)
N = bin(1234)
print("N:", type(N), N)
```

```
In [13]: runfile('C:/Users/sony/[
python_co_ban/test/Test1.py', wd:
sony/Desktop/python_co_ban/test')
K: <class 'str'> 1234
L: <class 'str'> 0x4d2
M: <class 'str'> 0o2322
N: <class 'str'> 0b10011010010
```

# Kiểu Dữ Liệu: Number

- Python hỗ trợ kiểu số phức, với chữ j đại diện cho phần ảo

- $A = 3+4j$

- $B = 2 - 2j$

- `print(A+B)`      # sã în ra  $(5+2j)$

# Kiểu Dữ Liệu: Number

## Chuyển đổi giữa các kiểu số trong Python

- Ví dụ: Nếu bạn thực hiện phép cộng giữa số nguyên là 2 và số thập phân là 3.0, thì 2 sẽ bị cưỡng chế chuyển thành số thập phân 2.0 và kết quả trả về sẽ là số thập phân 5.0.

```
>>> 2 + 3.0  
5.0
```

```
print(2+ 3.0)
```



# Kiểu Dữ Liệu: Number

- Có thể sử dụng các hàm Python tích hợp sẵn như **int()**, **float()** và **complex()** để chuyển đổi giữa các kiểu số một cách rõ ràng. Những hàm này thậm chí có thể chuyển đổi từ các chuỗi.

```
>>> int(3.6)
```

```
3
```

```
>>> int(-1.2)
```

```
-1
```

```
>>> float(7)
```

```
7.0
```

```
>>> complex('2+8j')
```

```
(2+8j)
```