

Phát triển phần mềm nâng cao cho tính toán khoa học Làm quen với Python

Vũ Tiến Dũng

Khoa Toán - Cơ - Tin học

Trường ĐH Khoa học Tự Nhiên Hà Nội

Nội dung

- 1 Giới thiệu về Python
- 2 Một số khái niệm cơ bản

Giới thiệu về ngôn ngữ lập trình Python

- Ngôn ngữ lập trình Python được tác giả Guido van Rossum tạo ra vào những năm cuối thập niên 1980.
- Python hướng tới một ngôn ngữ có cú pháp đơn giản nhưng hiệu quả hơn so với các ngôn ngữ thông dụng như Java, C#, C, C++.
- Python đã khẳng định được những ưu điểm của mình như dễ học, dễ đọc, dễ bảo trì, cùng với hệ thống thư viện phong phú, Python dần trở thành một trong những ngôn ngữ phổ biến nhất hiện nay.
- Một người mới học lập trình có thể dễ dàng tiếp cận và sử dụng Python để giải quyết các bài toán một cách nhanh chóng so với các ngôn ngữ phức tạp khác.

Giới thiệu về ngôn ngữ lập trìn Python

- Python được sử dụng trong nhiều lĩnh vực như giáo dục, nghiên cứu hay trong công nghiệp.
- Các công ty, tổ chức chuyên về lĩnh vực công nghệ sử dụng Python để phát triển các phần mềm có thể kể đến như: Google, Yahoo, Facebook,...
- Tổ chức Nghiên cứu Hạt nhân châu Âu (European Organisation for Nuclear Research - CERN), Industrial Light and Magic, và Cơ quan Hàng không và Vũ trụ Hoa Kỳ (NASA).
- Python hỗ trợ trên đa nền tảng như Microsoft Windows, Mac OS X, Linux. Các thông tin về phiên bản mới nhất của Python có thể xem tại địa chỉ <http://www.python.org>

Giới thiệu về ngôn ngữ lập trìn Python

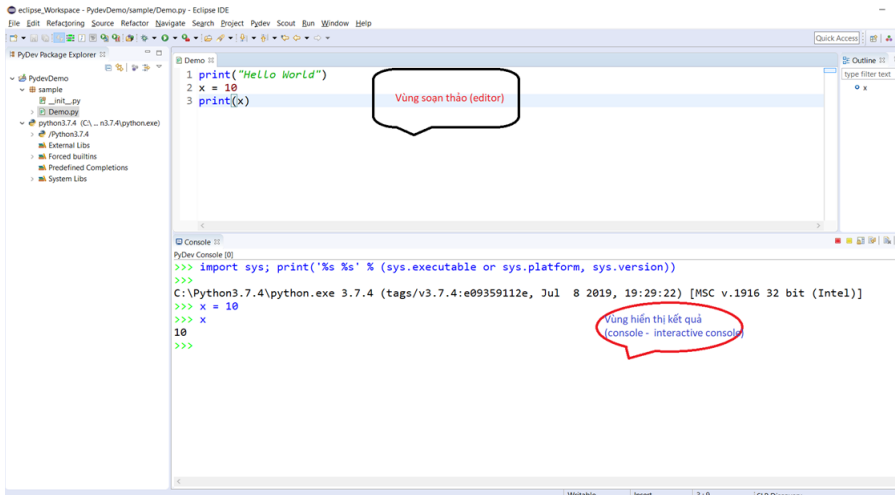
- Hiện tại Python có 2 phiên bản là Python 2 và Python 3.
- Python hướng tới một ngôn ngữ có cú pháp đơn giản nhưng hiệu quả hơn so với các ngôn ngữ thông dụng như Java, C#, C, C++.
- Python đang dần chuyển dịch từ Python 2 sang Python 3, ngày càng nhiều các tài nguyên và thư viện cho Python 3 được tạo ra và được sử dụng rộng rãi hơn.
- Trong nội dung môn học chúng ta sẽ làm việc với Python 3 (3.7.4)

Một số công cụ và tiện ích cho lập trình Python

- Cùng với sự phát triển của ngôn ngữ lập trình Python, ngày càng có nhiều công cụ hỗ trợ việc lập trình và phát triển phần mềm với Python như các IDE (Integrated Development Environment) Pycharm, Wing, Eric, PyDev (Eclipse), Spyder,...
- IDE được lựa chọn sử dụng trong môn học là Eclipse với plugin Pydev. Bên cạnh đó có thể dùng Google Colab <https://colab.research.google.com/> để thực lập trình Python.
- Nhìn chung, Eclipse là một IDE khá thông dụng và được sử dụng nhiều, Eclipse có các plugin hỗ trợ nhiều ngôn ngữ như JAVA, C, C++, Python, PHP, JSP,... cùng với đó Eclipse hỗ trợ đa nền tảng, giao diện thân thiện và khá tương đồng trên các hệ điều hành khác nhau.

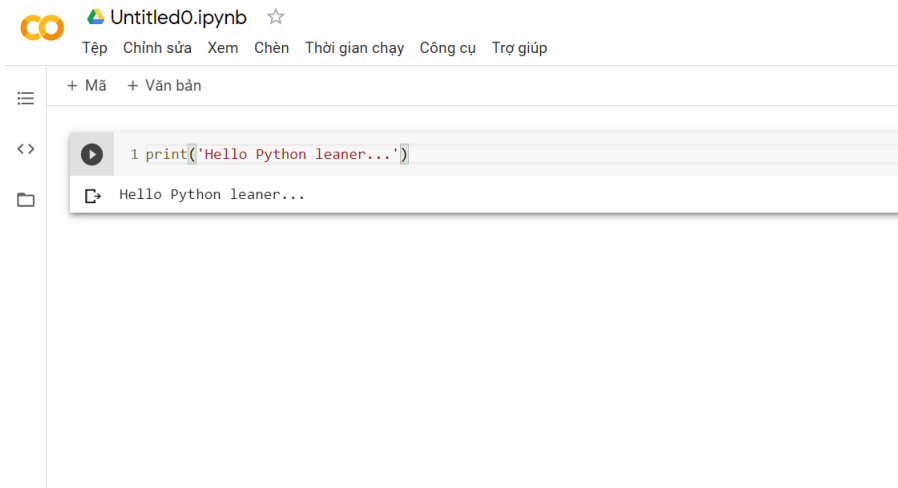
Lập trình Python trên eclipse với plugin Pydev

Hình 1: Giao diện Plugin Pydev trên eclipse



Lập trình Python trên Google colab

Hình 2: Giao diện Google colab

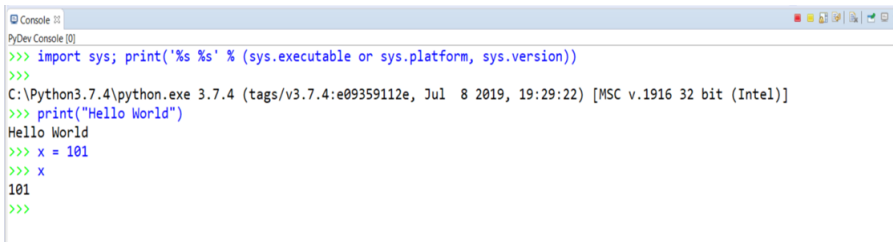


Dịch và chạy chương trình Python

- Để dịch và chạy một chương trình Python, trước hết cần cài đặt Python từ trang chủ
<https://www.python.org/downloads/release>.
- Trong môn học này chúng ta sẽ làm việc với phiên bản Python 3.7.4.
- Nếu cài đặt trên hệ điều hành Windows, cần chú ý việc thêm đường dẫn đến thư mục cài đặt Python vào biến môi trường Path.

Dịch và chạy chương trình Python

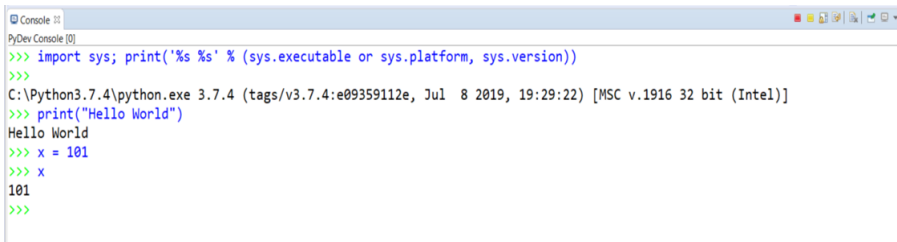
- Có một số cách để chạy một chương trình Python. Cách thứ nhất là sử dụng cửa sổ tương tác trực tiếp (interactive console) của trình thông dịch Python, tại đây các câu lệnh được thông dịch và thực thi trực tiếp.



```
Console
PyDev Console [0]
>>> import sys; print('%s %s' % (sys.executable or sys.platform, sys.version))
>>>
C:\Python3.7.4\python.exe 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)]
>>> print("Hello World")
Hello World
>>> x = 101
>>> x
101
>>>
```

Dịch và chạy chương trình Python

- Trình thông dịch sẽ xác định dòng lệnh người dùng nhập vào là biểu thức hay câu lệnh và sẽ thực thi chúng. Nếu người dùng nhập vào $x = 101$, trình thông dịch hiểu nó là một phép gán và nó không trả lại giá trị gì. Nếu người dùng nhập vào x , trình thông dịch sẽ trả lại giá trị của x là 101.

A screenshot of a PyDev Console window. The title bar says 'Console'. Below it, the text 'PyDev Console [0]' is visible. The console contains the following text:

```
>>> import sys; print('%s %s' % (sys.executable or sys.platform, sys.version))
>>>
C:\Python3.7.4\python.exe 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)]
>>> print("Hello World")
Hello World
>>> x = 101
>>> x
101
>>>
```

Dịch và chạy chương trình Python

- Cách thứ hai là dịch và chạy cả file mã nguồn Python, việc này có thể thực hiện trực tiếp trên Eclipse hoặc trên cửa sổ dòng lệnh thông qua lệnh **python tên_file.py**

```
D:\eclipse_Workspace\PydevDemo\sample>python Demo.py
Hello World
10

D:\eclipse_Workspace\PydevDemo\sample>
```

Một số đặc điểm của ngôn ngữ Python

- **Học và sử dụng một cách dễ dàng:** Python dễ học và dễ sử dụng bởi vì nó là ngôn ngữ lập trình bậc cao, dễ được tiếp cận bởi người dùng.
- **Ngôn ngữ biểu đạt (Expressive Language):** Ngôn ngữ Python có tính biểu đạt cao có nghĩa là nó dễ đọc và dễ hiểu.
- **Ngôn ngữ thông dịch (Interpreted Language):** Python là một ngôn ngữ thông dịch, trình thông dịch thực thi từng lệnh tại mỗi thời điểm, điều này giúp việc kiểm soát và gỡ lỗi dễ dàng hơn, do đó nó phù hợp với những người mới học lập trình.

Một số đặc điểm của ngôn ngữ Python

- **Ngôn ngữ đa nền tảng** (Cross-platform Language): Python hoạt động đồng nhất trên các nền tảng khác nhau như as Windows, Linux, Unix và Macintosh,... Do vậy chúng ta cũng có thể gọi Python là một ngôn ngữ portable (portable language).
- **Miễn phí và mã nguồn mở**: Giấy phép Python Software Foundation, Python được cung cấp miễn phí trên trang chủ <https://www.python.org>.
- **Ngôn ngữ hướng đối tượng** (Object-Oriented Language): Python hỗ trợ ngôn ngữ lập trình hướng đối tượng và đưa ra các khái niệm về các lớp (class) và đối tượng (object).

Một số đặc điểm của ngôn ngữ Python

- **Có khả năng mở rộng (Extensible):** Python có thể viết mã chương trình bằng các ngôn ngữ khác như C, C++,... và cũng có thể biên dịch mã chương trình đó trong các ngôn ngữ tương ứng.
- **Thư viện khổng lồ (Large Standard Library):** Python có một hệ thống thư viện khổng lồ và trải rộng trên các lĩnh vực, nó cung cấp một nguồn tài nguyên phong phú giúp việc phát triển các ứng dụng trở nên nhanh chóng.
- **Hỗ trợ lập trình giao diện (GUI Programming Support):** Giao diện người dùng có thể được phát triển bằng Python.

Một số đặc điểm của ngôn ngữ Python

- **Khả năng tích hợp** (Integrated): Python có thể dễ dàng tích hợp với các ngôn ngữ khác như C, C++, JAVA,...
- **Ngôn ngữ kiểu dữ liệu động** (Dynamically Typed Language): Kiểu dữ liệu của một biến được quyết định khi thực thi chứ không phải khi khai báo, khởi tạo. Do vậy không cần khai báo kiểu dữ liệu cho các biến trong Python.

Biến

- Một biến trong Python có thể nhận giá trị số và giá trị không phải là số.
- Ví dụ dưới đây dùng biến x để lưu giá trị số nguyên, và in ra giá trị của biến x. Chú ý nếu dùng câu lệnh `print('x')` sẽ in ra ký tự x thay vì giá trị của biến x là 10, vì 'x' là biểu diễn của chuỗi ký tự.

```
>>> x = 10
>>> print(x)
10
```

Biến

- Các biến có thể được khai báo và gán giá trị theo bộ (tupe) bằng dấu phẩy ','. Ví dụ
- Chú ý số lượng biến phải bằng số lượng giá trị, nếu không sẽ là một câu lệnh lỗi.

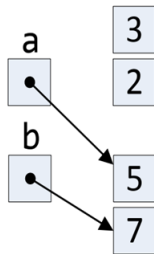
```
>>> x, y, z = 10, 20, 30
>>> x
10
>>> y
20
>>> z
30
>>> |
```

```
>>> x, y, z = 10, 20
Traceback (most recent call last):
  File "<input>", line 1, in <module>
ValueError: not enough values to unpack (expected 3, got 2)
>>> x, y, z = 10, 20, 30, 40
Traceback (most recent call last):
  File "<input>", line 1, in <module>
ValueError: too many values to unpack (expected 3)
>>>
```

Biến

- Hình dưới đây minh họa về việc gán và thay đổi giá trị cho các biến

```
a = 2  
b = 5  
a = 3  
a = b  
b = 7
```



Biến

- Để lấy ra kiểu giá trị của biến, ta dùng từ khóa type
- Python không yêu cầu xác định kiểu dữ liệu cho biến, kiểu dữ liệu của biến chỉ được xác định khi thực thi, do vậy một biến có thể mang các kiểu giá trị khác nhau tại các thời điểm khác nhau.

```
>>> a = 507
>>> print('giá trị của a = ', a, ' kiểu dữ liệu là', type(a))
giá trị của a = 507 kiểu dữ liệu là <class 'int'>
>>> a = 'MIM'
>>> print('giá trị của a = ', a, ' kiểu dữ liệu là', type(a))
giá trị của a = MIM kiểu dữ liệu là <class 'str'>
>>> |
```

Biến

- Một biến cần được gán giá trị để thực thi và gọi tới trong các câu lệnh tiếp theo, nếu một biến chưa được gán giá trị và được gọi tới sẽ có thông báo lỗi biến đó chưa được khai báo.

```
>>> x = 503
>>> x
503
>>> y
Traceback (most recent call last):
  File "<input>", line 1, in <module>
NameError: name 'y' is not defined
>>> |
```

Biến

- Để xóa một biến khỏi phiên làm việc hiện tại sử dụng từ khóa `del`.

```
>>> a,b,c = 1,'abc',4
>>> a,b,c
(1, 'abc', 4)
>>> del a
>>> a
Traceback (most recent call last):
  File "<input>", line 1, in <module>
NameError: name 'a' is not defined
>>> b
'abc'
>>> c
4
>>> del b, c
>>> b
Traceback (most recent call last):
  File "<input>", line 1, in <module>
NameError: name 'b' is not defined
>>> c
Traceback (most recent call last):
  File "<input>", line 1, in <module>
NameError: name 'c' is not defined
>>> |
```

Biến

Chú ý đến quy ước của việc đặt tên biến, việc đặt tên biến, hay các định danh (Identifier) trong Python được quy ước như sau:

- Một định danh cần chứa ít nhất một ký tự
- Ký tự đầu tiên của một định danh cần là chữ cái viết hoa hoặc thường hoặc dấu gạch dưới ' _ ' (underscore).
Python3.7.4 có thể đặt tên biến bằng ký tự unicode
- Các ký tự sau ký tự đầu tiên có thể là chữ cái, dấu gạch dưới, hoặc số
- Các ký tự đặc biệt không được sử dụng để đặt tên cho các định danh như (@, #, !, *, ...)

Biến

Hình 3: Ví dụ về đặt tên biên trong Python

```
>>> x = 10
>>> y = 20
>>> tổng = x+y
>>> print('tổng = ', tổng)
tổng = 30
>>>
```


- Tên của các định danh không được đặt trùng với các từ khóa trong Python, bảng dưới đây là danh sách các từ khóa trong Python

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

Biến

- Trong Python, các định danh được phân biệt chữ viết hoa và chữ viết thường, ví dụ 'for' là một từ khóa và không được sử dụng để đặt tên cho một định danh thì 'FOR', hoặc 'For' không phải là một từ khóa và có thể sử dụng để đặt tên cho một định danh.
- Tương tự như vậy, các định danh có cách viết hoa, thường khác nhau là các định danh khác nhau.

```
>>> FOR = 100
>>> FOR
100
>>> for = 10
      File "<input>", line 1
        for = 10
          ^
      SyntaxError: invalid syntax
```

Biến

- Việc đặt tên cho một định danh, hoặc biến cần thể hiện được mục đích và ý nghĩa của định danh đó, nên tránh việc đặt tên định danh gần tương tự nhau.
- Việc đặt tên các định danh theo quy tắc giúp cho mã nguồn chương trình dễ đọc, dễ hiểu hơn, đồng thời cũng tránh được các lỗi do nhầm lẫn và dễ dàng tìm và sửa lỗi hơn.

Nhập dữ liệu (input)

- Dữ liệu nhập từ luồng nhập xuất chuẩn được đọc thông qua lệnh 'input()'

```
>>> print('Nhập vào tên của bạn')
Nhập vào tên của bạn
>>> name = input()
Bá Khá
>>> print('Xin chào bạn ',name)
Xin chào bạn  Bá Khá
>>> print(type(name))
<class 'str'>
>>> print('Nhập vào một giá trị số')
Nhập vào một giá trị số
>>> num = input()
123.6
>>> print('Giá trị số = ',num,' Kiểu dữ liệu',type(num))
Giá trị số =  123.6  Kiểu dữ liệu <class 'str'>
>>> num = float(num)
>>> print('Giá trị số = ',num,' Kiểu dữ liệu',type(num))
Giá trị số =  123.6  Kiểu dữ liệu <class 'float'>
>>> |
```

Nhập dữ liệu (input)

- Kiểu dữ liệu mặc định của dữ liệu được trả về từ input() là kiểu chuỗi ký tự
- Có thể chuyển kiểu dữ liệu chuỗi ký tự sang các kiểu dữ liệu số như int hoặc float.
- Lệnh input còn có thể truyền tham số là chuỗi ký tự, và dữ liệu có thể chuyển đổi trực tiếp sang kiểu mong muốn
- Chú ý việc chuyển đổi dữ liệu (ép kiểu) yêu cầu dữ liệu được nhập đúng định dạng, nếu không sẽ có thông báo lỗi.

Nhập dữ liệu (input)

Hình 4: Ví dụ về nhập dữ liệu bằng input

```
>>> num = int(input('Nhập vào một số nguyên:'))
Nhập vào một số nguyên:3
>>> print('Giá trị của số nguyên vừa nhập là num = ',num)
Giá trị của số nguyên vừa nhập là num = 3
>>> num = int(input('Nhập vào một số nguyên: '))
Nhập vào một số nguyên: 3.14
Traceback (most recent call last):
  File "<input>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '3.14'
>>> |
```

Xuất dữ liệu (print)

- Việc xuất dữ liệu ra luồng xuất chuẩn, ví dụ cửa sổ dòng lệnh (console) được thực hiện thông qua lệnh print
- Một số định dạng in với lện print được thể hiện trong ví dụ dưới đây.

```
>>> print('abc')
abc
>>> print('abc',end='')
abc>>> print('abc',end='\n')
abc
>>> print('abc',end='abc')
abccabc>>> print('',end='abc\n')
abc
```

Xuất dữ liệu (print)

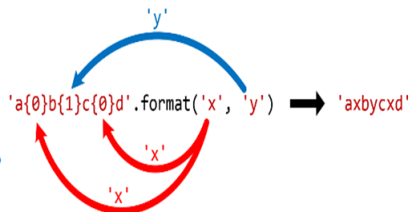
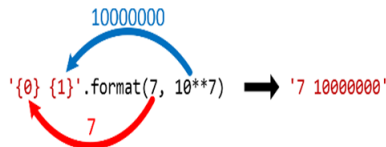
- Ví dụ về định dạng chuỗi phân cách giữa các giá trị được in ra trong lệnh print qua tham số sep, giá trị mặc định là sep = "

```
>>> w, x, y, z = 10, 15, 20, 25
>>> print(w, x, y, z)
10 15 20 25
>>> print(w, x, y, z, sep=',')
10,15,20,25
>>> print(w, x, y, z, sep='')
10152025
>>> print(w, x, y, z, sep=':')
10:15:20:25
>>> print(w, x, y, z, sep='-----')
10-----15-----20-----25
>>>
```


Xuất dữ liệu (print)

- Ví dụ về định dạng chuỗi in thông qua hàm format

```
>>> print(0, 10**0)
0 1
>>> print(0, 10**1)
0 10
>>> print(0, 10**3)
0 1000
>>> print(0, 10**6)
0 1000000
>>> print(0, 10**9)
0 1000000000
>>> print(0, 10**12)
0 1000000000000
>>> print('{0} {1}'.format(0, 10**5))
0 100000
>>> print('{0} {1}'.format(10, 10**10))
10 100000000000
>>> |
```



Xuất dữ liệu (print)

- Ví dụ về định dạng chuỗi in thông qua hàm format, căn lề bên phải

```
>>> print('{0:>3} {1:>16}'.format(4, 10**4))
4          10000
>>> print('{0:>3} {1:>16}'.format(4, 10**5))
4          100000
>>> print('{0:>3} {1:>16}'.format(4, 10**6))
4          1000000
>>> print('{0:>3} {1:>16}'.format(4, 10**7))
4          10000000
>>> print('{0:>3} {1:>16}'.format(4, 10**8))
4          100000000
>>> print('{0:>3} {1:>16}'.format(4, 10**9))
4          1000000000
>>> print('{0:>3} {1:>16}'.format(4, 10**10))
4          10000000000
```

Xuất dữ liệu (print)

- Cặp 3 dấu “‘ ’”, hoặc “ “ “ ” ” ” dùng để đánh dấu chuỗi trên nhiều dòng, xem ví dụ dưới đây

```
>>> multiline = '''Đây là chuỗi trên nhiều dòng
... mỗi dòng sẽ được đánh dấu bởi ký tự xuống dòng '\n' '''
>>> multiline
"Đây là chuỗi trên nhiều dòng\nmỗi dòng sẽ được đánh dấu bởi ký tự xuống dòng '\n' "
>>> |
```

Biểu thức và phép toán

- Một số tự nhiên 34 hay một biến x được xem là một biểu thức. Chúng ta có thể sử dụng các toán tử để kết hợp các giá trị và biến thành các biểu thức phức tạp hơn.

Biểu thức	Ý nghĩa	Ví dụ
$x + y$	Phép cộng giá trị x với giá trị y nếu x, y là số. Nối y vào x thành chuỗi xy nếu x và y là chuỗi ký tự	$4 + 5 \Rightarrow 9$ <code>'abc' + 'def' => 'abcdef'</code>
$x - y$	Phép trừ x với y nếu x và y là số	$7 - 8 \Rightarrow -1$
$x * y$	Phép nhân x với y nếu x và y là số Nhân x lên y lần nếu x là chuỗi ký tự và y là số nguyên Nhân y lên x lần nếu x là số nguyên và y là chuỗi ký tự	$3 * 5 \Rightarrow 15$ <code>"ab" * 2 => "abab"</code> $2 * "ab" \Rightarrow "abab"$
x / y	Phép chia x cho y nếu x và y là số	$6 / 3 \Rightarrow 2$
$x // y$	Phép lấy phần nguyên của phép chia x cho y nếu x và y là số	$7 // 3 \Rightarrow 2$
$x \% y$	Phép lấy phần dư của phép chia x cho y nếu x và y là số	$7 \% 3 \Rightarrow 1$
$x ** y$	Phép tính x mũ y nếu x và y là số	$2 ** 3 \Rightarrow 8$

Biểu thức và phép toán

- Các toán tử có thể sử dụng kết hợp với phép gán “=” để có phép gán gọn hơn như:
 - $x += y$ tương đương với $x = x + y$
- Các biểu thức đều có một giá trị nào đó. Giá trị của biểu thức được tính thông qua các giá trị và giá trị của các biến, kết hợp với các phép toán trên các giá trị để tính được giá trị của các biểu thức phức tạp.
- Các phép toán trong bảng trên là các phép toán hai ngôi (toán tử hai ngôi), bên cạnh đó còn có các toán tử một ngôi như dấu $-$ cho giá trị âm, và dấu $+$ cho các giá trị dương.

Biểu thức và phép toán

- Các giá trị bị giới hạn bởi kiểu dữ liệu. Ví dụ dưới đây cho thấy biểu thức `2.0 ** 10000` vượt quá giá trị của kiểu số thực

```
>>> 2.0 ** 1000
1.0715086071862673e+301
>>> 2.0 ** 10000
Traceback (most recent call last):
  File "<input>", line 1, in <module>
OverflowError: (34, 'Result too large')
>>> |
```

Biểu thức và phép toán

- Cần chú ý đến sai số trong các giá trị của biểu thức, ví dụ dưới đây cho thấy có sai số (rất nhỏ) trong phép toán $1^{1/3} 1^{1/3} 1^{1/3} \neq 0$
- Các phép tính trên số thập phân vô hạn không tuần hoàn thường sẽ có sai số

```
>>> one = 1.0
>>> ot = 1.0/3.0
>>> z = one - ot - ot - ot
>>> z
1.1102230246251565e-16
>>> 1.0 - 1/3 - 1/3 - 1/3
1.1102230246251565e-16
>>> |
```

Biểu thức và phép toán

- Các phép toán có thứ tự ưu tiên khi thực hiện, bảng dưới đây thể hiện độ ưu tiên của các phép toán được sắp xếp giảm dần
- Các phép toán cùng độ ưu tiên thì sẽ thực hiện theo thứ tự từ trái sang phải hoặc từ phải sang trái.
- Dấu # để đánh dấu một dòng ghi chú trong Python.

Số ngôi	Toán tử	Thứ tự	Ví dụ
Hai	**	Phải sang trái	$2 ** 3 ** 2 = 2 ** (3 ** 2) = 2 ** 9 = 512$
Một	+, -		-3, +4
Hai	*, /, //, %	Trái sang phải	$2 * 3 / 2 = (2 * 3) / 2 = 3$
Hai	+, -	Trái sang phải	$2 + 3 - 4 = (2 + 3) - 4$
Hai	=	Phải sang trái	$x = y = 4$