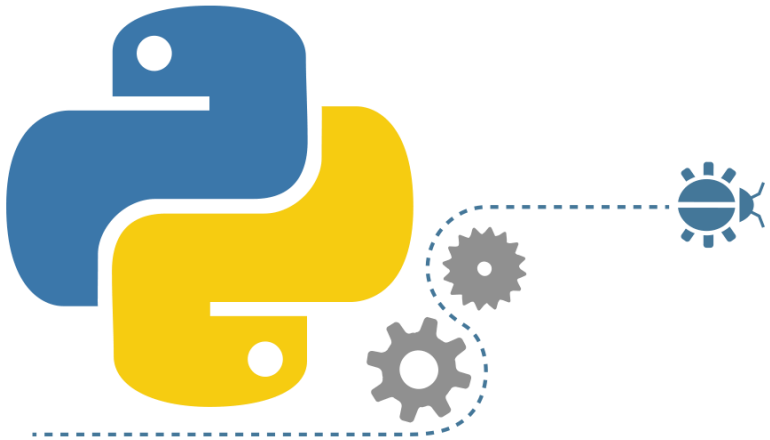
Kiểu dữ liệu cơ bản và khai báo biến trong Python



Nội dung bài học

1. Các kiểu dữ liệu cơ bản trong Python
2. Khai báo biến trong Python
3. Cách xóa biến
4. Cách kiểm tra vùng lưu trữ giá trị của các biến int, float

1. Các kiểu dữ liệu cơ bản trong Python

- Kiểu **int**: Kiểu số nguyên (không có chứa dấu chấm thập phân), có thể lưu các số nguyên âm và dương.
  - Ví dụ: 113, -114
- Kiểu **float**: Kiểu số thực (có chứa dấu chấm thập phân),
  - ví dụ: 5.2, -7.3

1. Các kiểu dữ liệu cơ bản trong Python

- Kiểu **complex**: Kiểu số phức,
  - ví dụ 1: z = 2+3j thì 2 là phần thực, 3 là phần ảo (j là từ khóa để đánh dấu phần ảo)
  - ví dụ 2: z=complex(2,3) thì 2 là phần thực, 3 là phần ảo
  - khi xuất kết quả ta có thể xuất:
    - print("Phần thực= ",z.real) ==>Phần thực= 2
    - print("Phần ảo= ",z.imag) ==> Phần ảo= 3

1. Các kiểu dữ liệu cơ bản trong Python

- Kiểu **str**: Kiểu chuỗi, để trong nháy đôi hoặc nháy đơn
  - Ví dụ: “Obama”, ‘Putin’
- Kiểu **bool**: Kiểu luận lý, để lưu True hoặc False
  - Ví dụ 1: t1=True
  - Ví dụ 2: t2=False

2. Khai báo biến trong Python

Trong Python một biến không cần khai báo kiểu dữ liệu, khi ta gán giá trị thì tự động Python sẽ nội suy ra kiểu dữ liệu của biến. Như vậy một biến có thể có nhiều kiểu dữ liệu tùy thuộc vào giá trị mà ta gán. Ta có thể dùng hàm type() để kiểm tra kiểu dữ liệu của biến:

2. Khai báo biến trong Python

```
x=5
print(type(x))
x='teo'
print(type(x))
x=True
print(type(x))
x=5.5
print(type(x))
x=complex(113,114)
print(type(x))
```

Với x = 5 ta có kiểu dữ liệu: <class 'int'>
Với x = 'teo' ta có kiểu dữ liệu:<class 'str'>
Với x = True ta có kiểu dữ liệu:<class 'bool'>
Với x = 5.5 ta có kiểu dữ liệu:<class 'float'>
Với x = complex(113,114) ta có kiểu dữ liệu:<class 'complex'>

```
print(x.real,x.imag)
```
➔ thực:113, ảo:114

3. Cách xóa biến

Trong Python có một điểm thú vị là: Nếu biến đó đang tồn tại mà ta xóa nó đi thì không còn sử dụng được nữa (tương tự trong C++ khi chúng ta thu hồi bộ nhớ của con trỏ vậy), Python dùng từ khóa del để xóa:

```
x="Obama"
print(x)
del   x
print(x)
```

```
Obama
Traceback (most recent call last):
  File "/XoaBien.py", line 4, in <module>
    print(x)
NameError: name 'x' is not defined
```

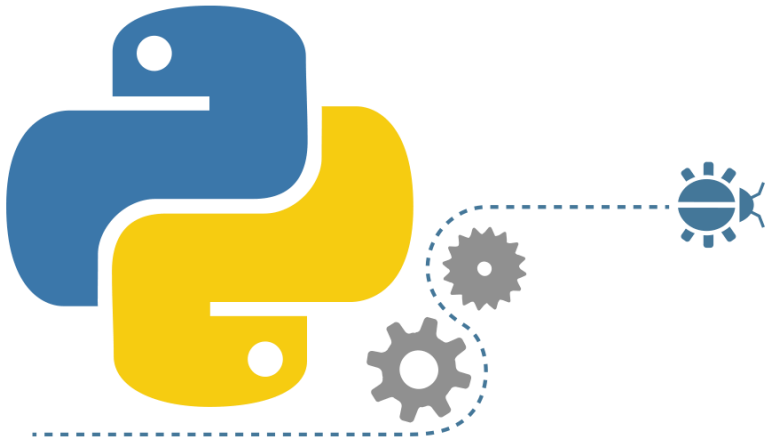
4. Cách kiểm tra vùng lưu trữ

Ta có thể kiểm tra vùng lưu trữ giá trị của các biến int, float bằng cách import thư viện sys để có thể xem được chi tiết:

```
import sys
print("Thông tin chi tiết của int:")
print(sys.int_info)
print("Thông tin chi tiết của float:")
print(sys.float_info)
```

Thông tin chi tiết của int:
sys.int_info(bits_per_digit=15, sizeof_digit=2)
Thông tin chi tiết của float:
sys.float_info(max=1.7976931348623157e+308, max_exp=1024, max_10_exp=308, min=2.2250738585072014e-308, min_exp=-1021, min_10_exp=-307, dig=15, mant_dig=53, epsilon=2.220446049250313e-16, radix=2, rounds=1)

Cách ghi chú lệnh trong Python



Nội dung bài học

1. Vì sao nên ghi chú khi lập trình
2. Ghi chú 1 dòng
3. Ghi chú nhiều dòng:

1. Vì sao nên ghi chú khi lập trình

Việc ghi chú lệnh một cách cẩn thận khi lập trình thể hiện tính chuyên nghiệp của Lập trình viên. Không phải nói ngoa nếu như các bạn được phỏng vấn xin việc, nếu Công ty kiểm tra coding từ các Project sample của bạn mà thấy bạn không có ghi chú một cách cẩn thận (cho dù bạn có lập trình giỏi tới mấy) thì khả năng bị loại cực cao, nếu giỏi mà cẩu thả thì càng nguy hiểm, vì độ "sát thương" cho các dự án rất cao.

Triển khai nhiều dự án, viết nhiều lệnh nếu không ghi chú: Khó khăn cho chính bản thân Programmer khi đọc lại và rất khó training khi có nhân viên mới vào làm việc.

2.Ghi chú 1 dòng

Python dùng từ khóa # để cho phép ta ghi chú 1 dòng:

GhiChu.py

```
1  #đây là ghi chú 1 dòng
2  a=5
3  b=2
4  c=a-b
5  if c<0:
6      print(c)
7  else:
8      print(c*2)
```

3. Ghi chú nhiều dòng

Để ghi chú nhiều dòng lệnh, Ta dùng """ """ (3 cặp nháy đôi) hoặc ''' '''(3 cập nháy đơn)

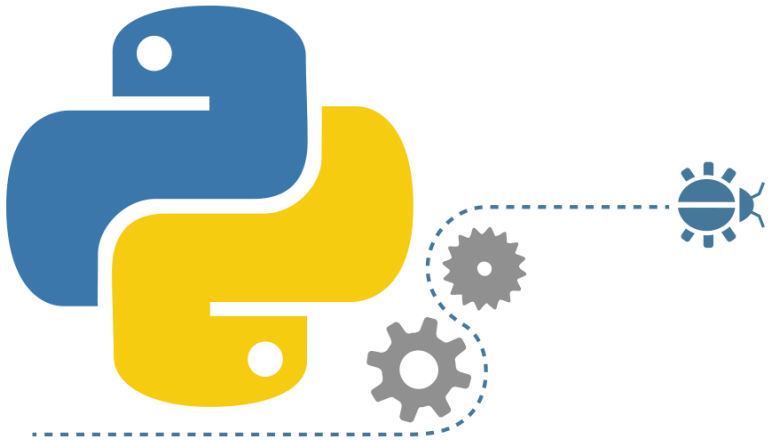
```
"""
Giải phương trình bậc 1: ax+b=0
Có 3 trường hợp để biện luận
Nếu hệ số a =0 và hệ số b=0 ==>vô số nghiệm
Nếu hệ số a =0 và hệ số b !=0 ==>vô nghiệm
Nếu hệ số a !=0 ==> có nghiệm -b/a
"""
a = 0
b = 113
if a == 0 and b == 0:
    print("Vô số nghiệm")
elif a == 0 and b != 0:
    print("Vô nghiệm")
else:
    print("Có No X=",-b/a)
```

3. Ghi chú nhiều dòng

Để ghi chú nhiều dòng lệnh, Ta dùng """ """ (3 cặp nháy đôi) hoặc ''' '''(3 cập nháy đơn)

```
'''
Đây là lệnh kiểm tra năm nhuần year
Năm nhuần là năm chia hết cho 4 nhưng không chia hết
cho 100 hoặc chia hết cho 400
'''
year=2016
if (year % 4==0 and year %100 !=0) or year % 400 ==0:
    print(year," Là năm nhuần")
else:
    print(year, " KO là năm nhuần")
```

Các toán tử thường dùng trong Python



Nội dung bài học

Mỗi một ngôn ngữ lập trình đều có tập các toán tử thường dùng và đa phần chúng khá giống nhau. Những bạn nào đã học C++, java, C# thì qua Python cũng tương tự.

Trong Python còn bổ sung thêm nhiều toán tử khá hữu ích khác nữa, dưới đây liệt kê 4 loại toán tử cơ bản thường dùng nhất trong Python (các loại khác bạn có thể xem thêm tại:

https://docs.python.org/3/library/stdtypes.html):

Nội dung bài học

1.Toán tử số học cơ bản

2.Toán tử gán

3.Toán tử So sánh

4.Toán tử Logic

5.Độ ưu tiên toán tử

1.Toán tử số học cơ bản

Toán tử	Mô tả	Ví dụ
+	Cộng	12 + 4.9 => kết quả 16.9
–	Trừ	3.98 – 4 => kết quả -0.02
*	Nhân	2 * 3.4 => kết quả 6.8
/	Chia	9 / 2 => kết quả 4.5
//	Chia lấy phần nguyên	9 // 2 => kết quả 4
%	Chia lấy phần dư	9%2 =>kết quả 1
**	Lũy thừa	3**4=>kết quả 81

2.Toán tử gán

Toán tử	Mô tả	Ví dụ	Tương đương với
=	Phép gán giá trị bên phải cho biến bên trái dấu bằng	x=5	
+=	Cộng và gán	x=2 x+=5 ==>x=7	x=x+5
-=	Trừ và gán	x=2 x-=5 ==>x=-3	x=x-5
*=	Nhân và gán	x=2 x*=5 ==>x=10	x=x*5

2.Toán tử gán

Toán tử	Mô tả	Ví dụ	Tương đương với
/=	Chia và gán	x=7 x/=5 ==>x=1.4	x=x/5
//=	Chia và gán (lấy nguyên)	x=7 x//=5 ==>x=1	x=x//5
%=	Chia lấy dư	x=7 x%=5 ==>x=2	x=x%5
**=	Lấy lũy thừa và gán	x=2 x**=3 ==>x là 2 mũ 3 =8	x=x**3

3.Toán tử So sánh

Toán tử	Mô tả	Ví dụ
==	So sánh bằng	5 == 5 => kết quả True
!=	So sánh không bằng	5 != 5 => kết quả False
<	So sánh nhỏ hơn	5 < 5 => kết quả False
<=	So sánh nhỏ hơn hoặc bằng	5 <= 5 => kết quả True
>	So sánh lớn hơn	5 > 5.5 => kết quả False
>=	So sánh lớn hơn hoặc bằng	113>= 5 => kết quả True
is	Trả về true nếu các biến ở hai bên toán tử cùng trỏ tới một đối tượng(hoặc cùng giá trị), nếu không là false	x=5 y=5 print(x is y) =>kết quả là True
is not	Trả về false nếu các biến ở hai bên toán tử cùng trỏ tới một đối tượng(hoặc cùng giá trị), nếu không là true	x=5 y=5 print(x is not y) =>kết quả là False

4.Toán tử Logic

Toán tử	Mô tả	Ví dụ
and	Toán tử Và: Nếu cả hai điều kiện là True thì kết quả sẽ là True	x=2016 print(x%4==0 and x%100!=0) =>True
or	Toán tử Hoặc: Chỉ cần một điều kiện True thì nó True, tất cả điều kiện False thì nó False	x=2016 print((x%4==0 and x%100!=0) or x%400==0) =>True
not	Toán tử Phủ định. Thông thường nó được dùng để đảo ngược trạng thái logic của toán hạng	x=4 if (not x>=5): print("Ngắm gà khỏa thân và nải chuối") else: print("Đậu")

5.Độ ưu tiên toán tử

Python có ràng buộc thứ tự ưu tiên của các toán tử. Tuy nhiên tốt nhất là các bạn hay điều khiển nó bằng cách dùng cặp ngoặc tròn ( ) để nó rõ nghĩa hơn. Bảng dưới đây để tham khảo độ ưu tiên từ cao xuống thấp (tuy nhiên có thể quên nó đi mà hãy dùng ngoặc tròn () để chỉ định rõ).

5.Độ ưu tiên toán tử

Thứ tự ưu tiên	Toán tử	Miêu tả
1	**	Toán tử mũ
2	* / % //	Phép nhân, chia, lấy phần dư và phép chia lấy phần nguyên
3	+ –	Toán tử Cộng, Trừ
4	<= < > >=	Các toán tử so sánh
5	<> == !=	Các toán tử so sánh
6	= %= /= //= -= += *= **=	Các toán tử gán
7	is , is not	Các toán tử so sánh
8	not, or, and	Các toán tử Logic