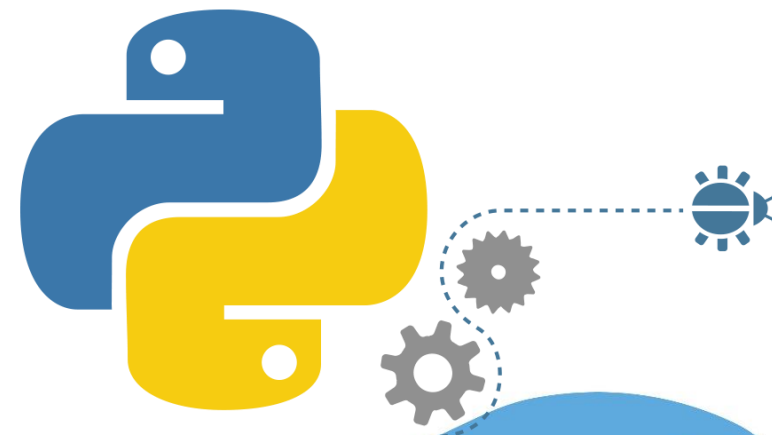


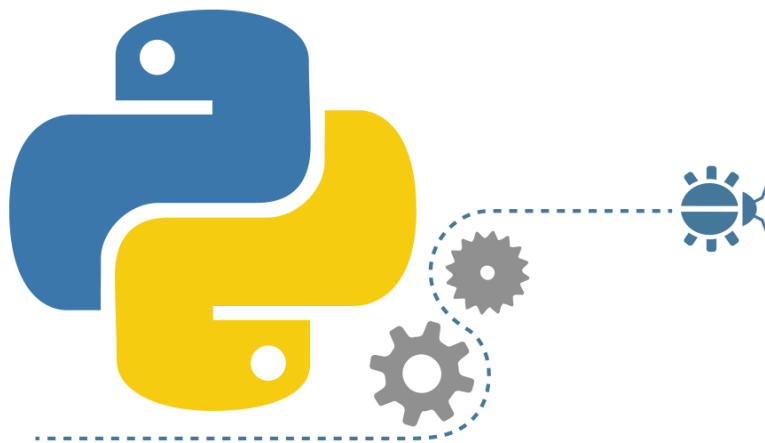


Môn Học **KỸ THUẬT LẬP TRÌNH** **VỚI PYTHON**

GV: Ths. Trần Duy Thanh
thanhtd@uel.edu.vn



CÁC KHÁI NIỆM CƠ BẢN





Mục tiêu bài học

- Hiểu và thực hiện được các kiểu dữ liệu cơ bản
- Khai báo được biến, ghi chú
- Thực hành được các toán tử
- Nhập liệu được từ bàn phím
- Xử lý được các lỗi



Nội dung bài học

2.1. Kiểu dữ liệu cơ bản và khai báo biến trong Python

2.2. Cách ghi chú lệnh trong Python

2.3. Các toán tử thường dùng trong Python

2.4. Cách nhập liệu từ bàn phím trong Python

2.5. Các kiểu xuất dữ liệu

2.6. Các loại lỗi trong Python



2.1. Kiểu dữ liệu cơ bản và khai báo biến trong Python

2.1.1. Các kiểu dữ liệu cơ bản trong Python

2.1. 2. Khai báo biến trong Python

2.1. 3. Cách xóa biến

2.1. 4. Cách kiểm tra vùng lưu trữ giá trị của các biến int, float

2.1.1. 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

2.1.1. 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: $z = \text{complex}(2,3)$ 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



2.1.1. 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

2.1.2. 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:

2.1.2. 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
```

2.1.3. 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`

2.1.4. 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)`



2.2. Cách ghi chú lệnh trong Python

2.2.1. Vì sao nên ghi chú khi lập trình

2.2.2. Ghi chú 1 dòng

2.2.3. Ghi chú nhiều dòng

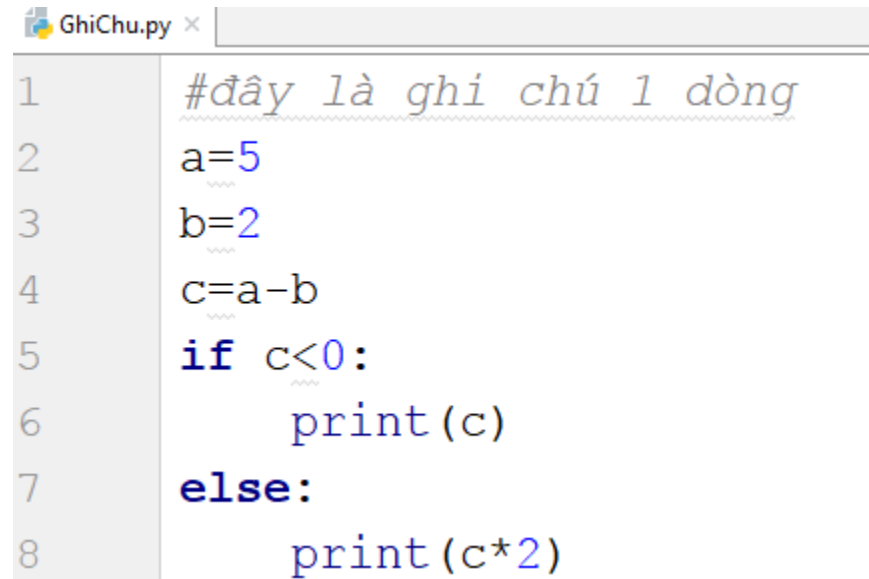
2.2.1. Vì sao nên ghi chú khi lập trình

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 cầu 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.

2.2.2. Ghi chú 1 dòng

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



```
GhiChu.py x
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)
```

2.2.3. Ghi chú nhiều dòng

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


2.2.3. Ghi chú nhiều dòng

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

```
'''  
    Đây là lệnh kiểm tra năm nhuận year  
    Năm nhuận 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 nhuận")  
else:  
    print(year, " KO là năm nhuận")
```



2.3. Các toán tử thường dùng trong Python

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 này 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>):

2.3.1. Toán tử số học cơ bản

2.3.2. Toán tử gán

2.3.3. Toán tử So sánh

2.3.4. Toán tử Logic

2.3.5. Độ ưu tiên toán tử

2.3.1.Toán tử số học cơ bản

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

2.3.2.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	$x=2$ $x+=5$ $\Rightarrow x=7$	$x=x+5$
-=	Trừ và gán	$x=2$ $x-=5$ $\Rightarrow x=-3$	$x=x-5$
=	Nhân và gán	$x=2$ $x=5$ $\Rightarrow x=10$	$x=x*5$

2.3.2. Toán tử gán

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

2.3.3.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

2.3.4.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	<pre>x=2016 print(x%4==0 and x%100!=0) =>True</pre>
or	Toán tử Hoặc; Chỉ cần một điều kiện True thì nó True, tất cả điều kiện False thì nó False	<pre>x=2016 print((x%4==0 and x%100!=0) or x%400==0) =>True</pre>
not	Toán tử Phủ định. Thông thường nó được dùng để đảo ngược trạng thái logic của toán hạng	<pre>x=4 if (not x>=5): print("Ngắm gà khỏa thân và nài chuỗi") else: print("Đậu")</pre>

2.3.5.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).

2.3.5.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



2.4. Cách nhập liệu từ bàn phím trong Python

Trong Python để nhập liệu từ bàn phím ta dùng hàm `input()`. Giá trị nhập vào của hàm `input()` thường là kiểu chuỗi, do đó ta cần chuyển kiểu nếu như muốn lưu trữ giá trị nhập vào không phải kiểu chuỗi.

```
print("Mời bạn nhập cái gì đó:")  
s=input()  
print("Bạn nhập:",s)  
print("Kiểu dữ liệu:",type(s))
```



2.4. Cách nhập liệu từ bàn phím trong Python

Muốn đưa về số int

```
print("Mời thím nhập int:")  
x=int(input())  
print("Bạn nhập:",x)  
print("Kiểu dữ liệu:",type(x))
```



2.4. Cách nhập liệu từ bàn phím trong Python

Muốn đưa về số float

```
print("Mời thím nhập int:")  
x=float(input())  
print("Bạn nhập:",x)  
print("Kiểu dữ liệu:",type(x))
```



2.4. Cách nhập liệu từ bàn phím trong Python

Muốn đưa về số boolean

```
def StrToBool(s):  
    return s.lower() in ("yes", "true", "t", "1")
```

```
print("Mời thím nhập bool:")  
x = StrToBool(input())  
print("Bạn nhập:", x)  
print("Kiểu dữ liệu:", type(x))
```



2.4. Cách nhập liệu từ bàn phím trong Python

Ngoài ra hàm `input()` còn có cho phép ta nhập nhãn tiêu đề vào như sau:

```
x=input("Mời bạn nhập giá trị gì đó:")  
print("Bạn nhập: ", x)
```

2.5. Các kiểu xuất dữ liệu

Để xuất lặp dữ liệu ra màn hình ta làm như sau:

```
print ('*' * 15)
```

➔ Xuất dấu * 15 lần *****

2.5. Các kiểu xuất dữ liệu

Dùng hàm format để xuất dữ liệu:

```
print('{0} {1}'.format(7, 10**7))
```

Diagram illustrating the first example: The format string `'{0} {1}'` has two placeholders. A blue arrow labeled `10000000` points from the `{1}` placeholder to the `10**7` argument. A red arrow labeled `7` points from the `{0}` placeholder to the `7` argument. The result is `'7 10000000'`.

Diagram illustrating the second example: The format string `'a{0}b{1}c{0}d'` has three placeholders. A blue arrow labeled `'y'` points from the `{1}` placeholder to the `'y'` argument. Two red arrows, both labeled `'x'`, point from the two `{0}` placeholders to the `'x'` argument. The result is `'axbycxd'`.

2.5. Các kiểu xuất dữ liệu

Muốn xuất căn phải ta có thể làm như sau:

```
print('-'*15)
print('{0:>2} {1:>11}'.format('STT', 'Giá trị'))
print('-'*15)
print('{0:>2} {1:>11}'.format(1, 10**10))
print('{0:>2} {1:>11}'.format(2, 10**9))
print('{0:>2} {1:>11}'.format(3, 10**8))
print('{0:>2} {1:>11}'.format(4, 10**7))
print('{0:>2} {1:>11}'.format(5, 10**6))
print('{0:>2} {1:>11}'.format(6, 10**5))
print('{0:>2} {1:>11}'.format(7, 10**4))
print('{0:>2} {1:>11}'.format(8, 10**3))
print('{0:>2} {1:>11}'.format(9, 10**2))
print('{0:>2} {1:>11}'.format(10, 10**1))
print('-'*15)
```

STT	Giá trị
1	10000000000
2	1000000000
3	100000000
4	10000000
5	1000000
6	100000
7	10000
8	1000
9	100
10	10

2.6. Các loại lỗi trong Python

Bất cứ người nào lập trình cũng đều sinh ra lỗi, tùy vào và khả và kinh nghiệm của từng người mà gặp các loại lỗi khác nhau. Có 3 loại lỗi thường gặp là:

2.6.1. Lỗi cú pháp(Syntax Errors)

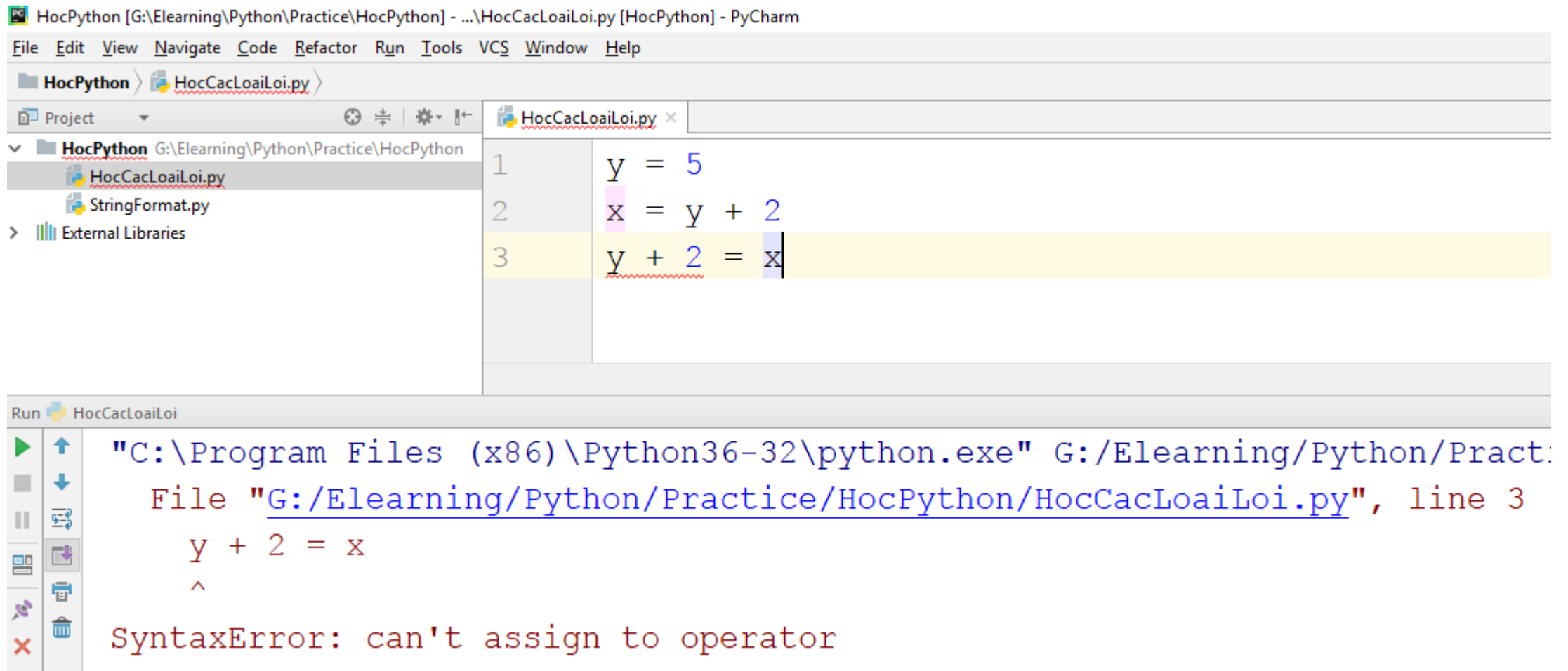
2.6.2. Lỗi thực thi(Run-time Exceptions)

2.6.3. Lỗi nghiệp vụ(Logic Errors)

Python cũng giống như các ngôn ngữ khác là cho phép bắt lỗi bằng khối lệnh try...catch

2.6.1. Lỗi cú pháp(Syntax Errors)

Lỗi thường gặp với những Programmer mới bắt đầu học, viết đâu sai đó, không thể thông dịch được phần mềm.



The screenshot shows the PyCharm IDE interface. The top menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The project name is HocPython, and the current file is HocCacLoaiLoi.py. The code editor displays three lines of Python code:

```
1 y = 5
2 x = y + 2
3 y + 2 = x
```

Line 3 is highlighted in yellow, and the expression `y + 2 = x` is underlined in red, indicating a syntax error. The Run window at the bottom shows the command used to execute the program:

```
"C:\Program Files (x86)\Python36-32\python.exe" G:/Elearning/Python/Pract:
```

Below the command, the file path and line number are shown:

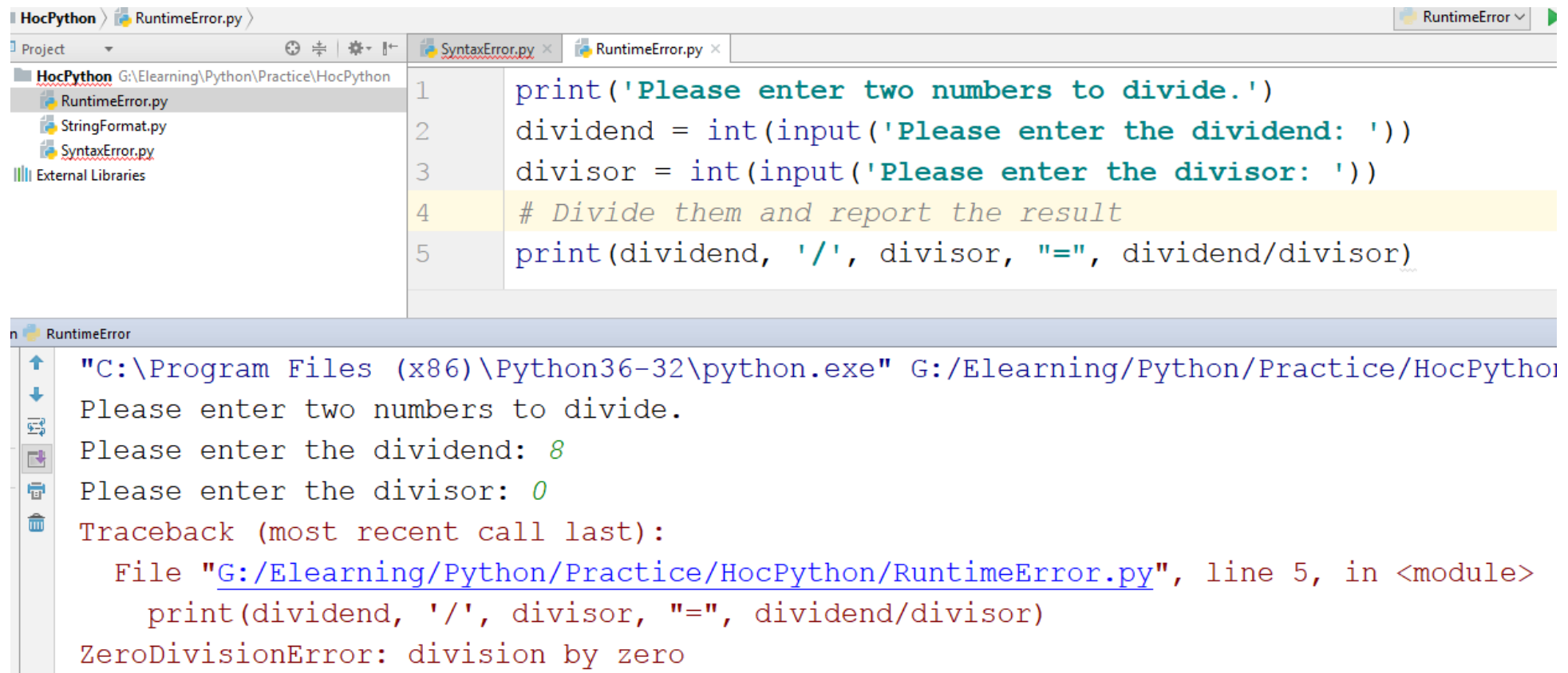
```
File "G:/Elearning/Python/Practice/HocPython/HocCacLoaiLoi.py", line 3
```

The error message is displayed in red text:

```
y + 2 = x
^
SyntaxError: can't assign to operator
```

2.6.2. Lỗi thực thi(Run-time Exceptions)

Thường lỗi này thông dịch hoàn tất, nhưng trong quá trình thực thi lại phát sinh ra lỗi. Các lỗi thường gặp như chia cho 0, ép kiểu sai, đọc file lỗi, kết nối mạng lỗi....



```
HocPython > RuntimeError.py
Project
HocPython G:\Elearning\Python\Practice\HocPython
  RuntimeError.py
  StringFormat.py
  SyntaxError.py
  External Libraries

1 print('Please enter two numbers to divide.')
2 dividend = int(input('Please enter the dividend: '))
3 divisor = int(input('Please enter the divisor: '))
4 # Divide them and report the result
5 print(dividend, '/', divisor, "=", dividend/divisor)

RuntimeError
"C:\Program Files (x86)\Python36-32\python.exe" G:/Elearning/Python/Practice/HocPython
Please enter two numbers to divide.
Please enter the dividend: 8
Please enter the divisor: 0
Traceback (most recent call last):
  File "G:/Elearning/Python/Practice/HocPython/RuntimeError.py", line 5, in <module>
    print(dividend, '/', divisor, "=", dividend/divisor)
ZeroDivisionError: division by zero
```

2.6.3. Lỗi nghiệp vụ (Logic Errors)

Lỗi này rất nghiêm trọng, thông dịch và chạy ngon lành. Nhưng kết quả sai với yêu cầu nghiệp vụ của Khách hàng → Đứt bóng đời Cô Lựu

Các lỗi này thường rất hiếm khi xảy ra, nhưng khi xảy ra rồi thì rất khó tìm thấy, và thấy rồi cũng thường khó fix bug.

2.6.4. Bắt lỗi

Python hỗ trợ try...catch để bắt lỗi Runtime, giúp báo rõ loại lỗi chương trình đang gặp và vẫn tiếp tục hoạt động khi gặp lỗi. Có rất nhiều trường hợp sử dụng, bài học trình bày 1 cách tổng quát nhất:

```
HandleException.py x
1  try:
2      x=5
3      y=0
4      z=x/y
5  except: # Catch absolutely any other type of exception
6      print('This program has encountered a problem')
```



THANK YOU

028 37244555 www.uel.edu.vn

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KINH TẾ - LUẬT

Số 669, đường Quốc lộ 1, khu phố 3, phường Linh Xuân,
quận Thủ Đức, Thành phố Hồ Chí Minh.