PYTHON BASICS

Abdelrahman A. Mohamed

Python Basics

- Data Types
- Variables
- Comments
- Arithmetic operations
- Range() and for loop
- Print() , input() and type()

Data Types

- Python needs to know how to set aside memory in your computer based on what kind of information you want to store

There are three basic types of data:

- Numeric Data Types
- Strings (character-based data)
- Boolean

Numeric Data Types:

- Integers:
  - Whole numbers that do not contain a decimal point
  - Abbreviated as “int” in Python
  - Example: 5, -5, 100, 10032
- Floating Point Numbers:
  - Numbers that contain a decimal point
  - Abbreviated as “float” in Python
  - Example: 5.0, -5.0, 100.99, 0.232132234

Numeric Data Types:

- complex (complex numbers):

  - Contain only “j” OR “J” letters and nothing else

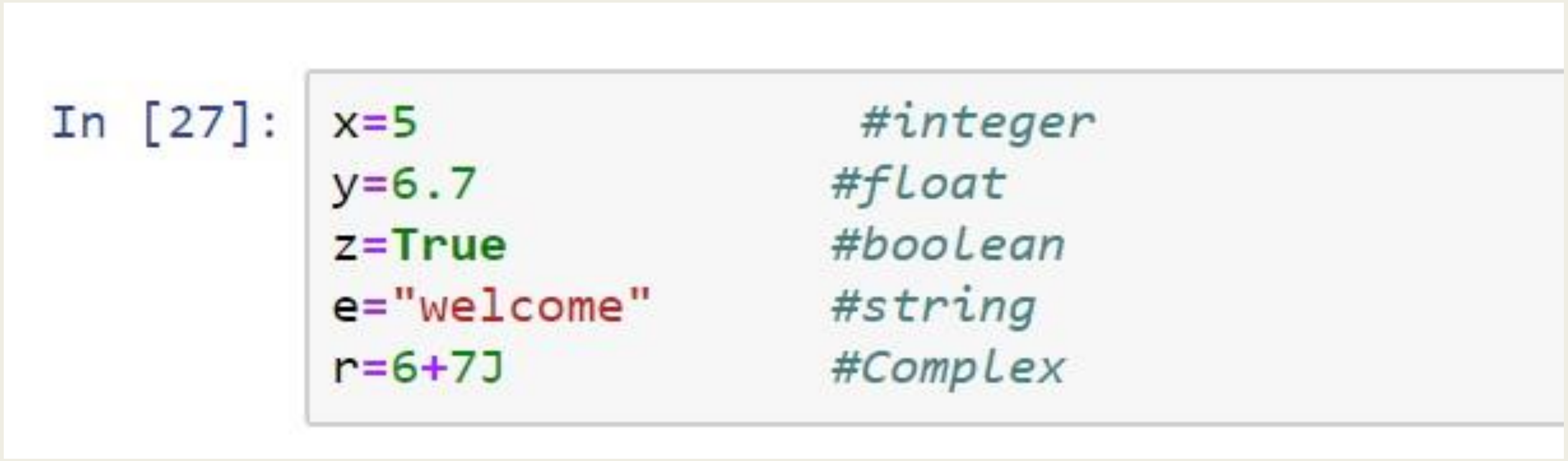
  - Abbreviated as “Complex” in Python

  - Examples 7+6j , 8j , 7.9j ,3J

Variables:

-Variables are nothing but reserved memory locations to store values according to their data types.

Examples:



```
In [27]: x=5                       #integer
         y=6.7                     #float
         z=True                    #boolean
         e="welcome"               #string
         r=6+7J                    #Complex
```

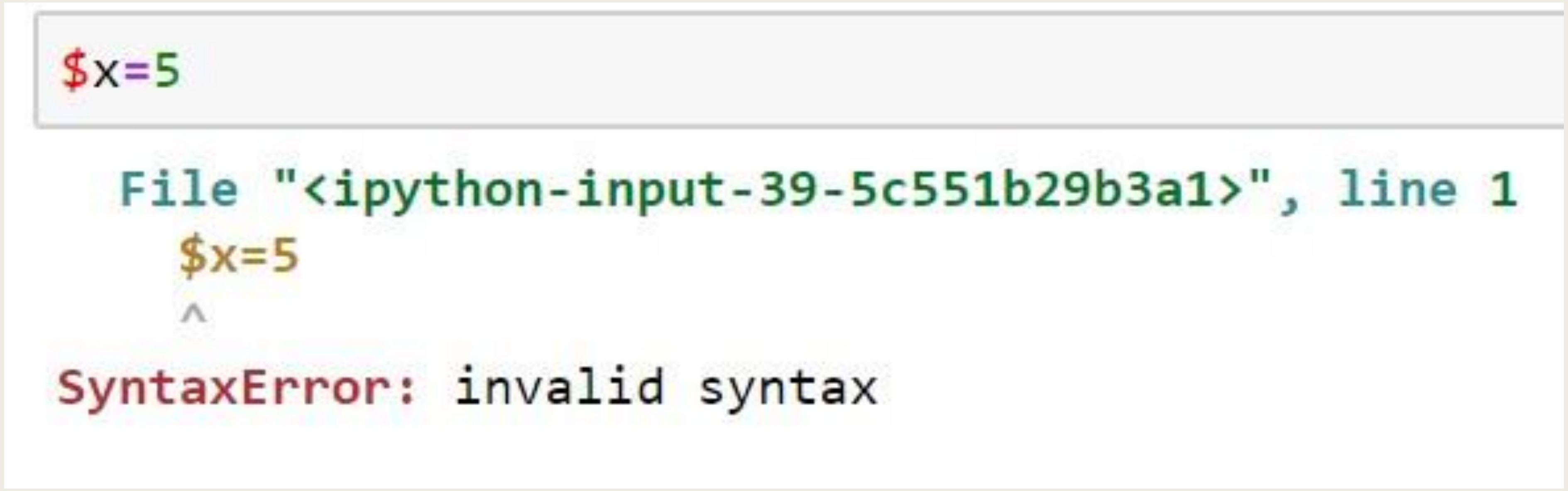
Variables:

Python sentence case language :

- X “capital case” not equal x “small case” .
- Print not equal print

The variable name can’t be the following:

- Can’t start or contain symbols

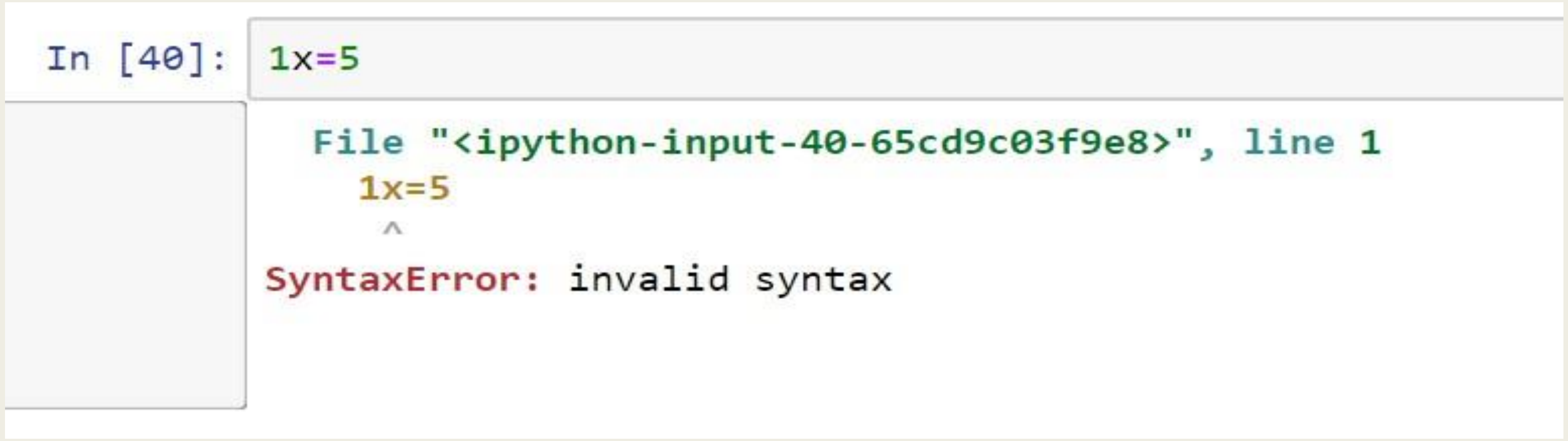


```
$x=5
```

```
  File "<ipython-input-39-5c551b29b3a1>", line 1
    $x=5
    ^
SyntaxError: invalid syntax
```

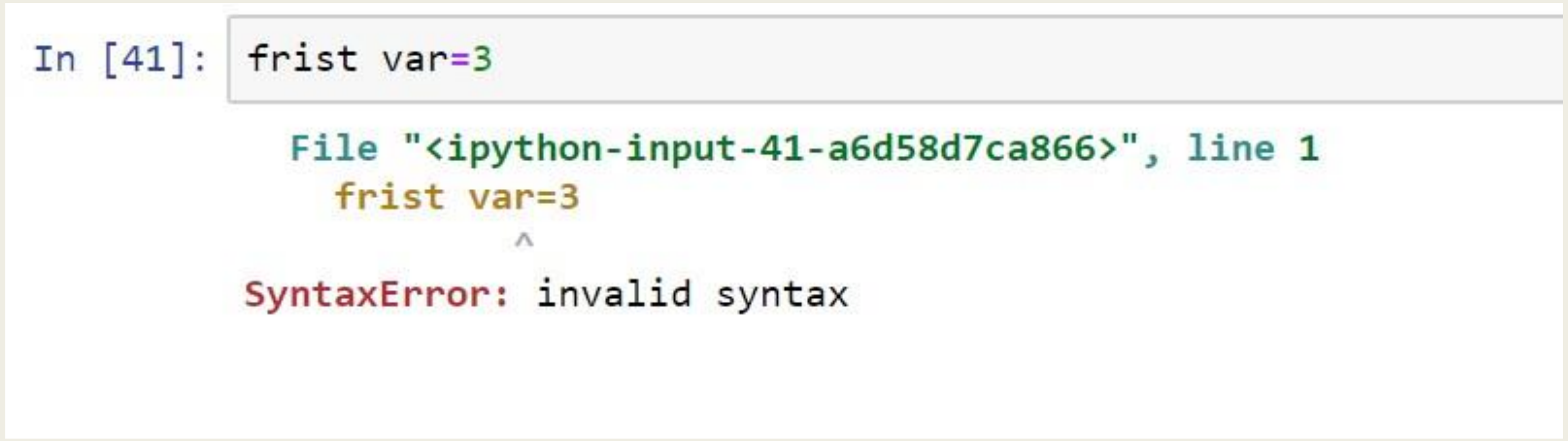
Variables:

- Can’t start with numbers



```
In [40]: 1x=5

  File "<ipython-input-40-65cd9c03f9e8>", line 1
    1x=5
     ^
SyntaxError: invalid syntax
```

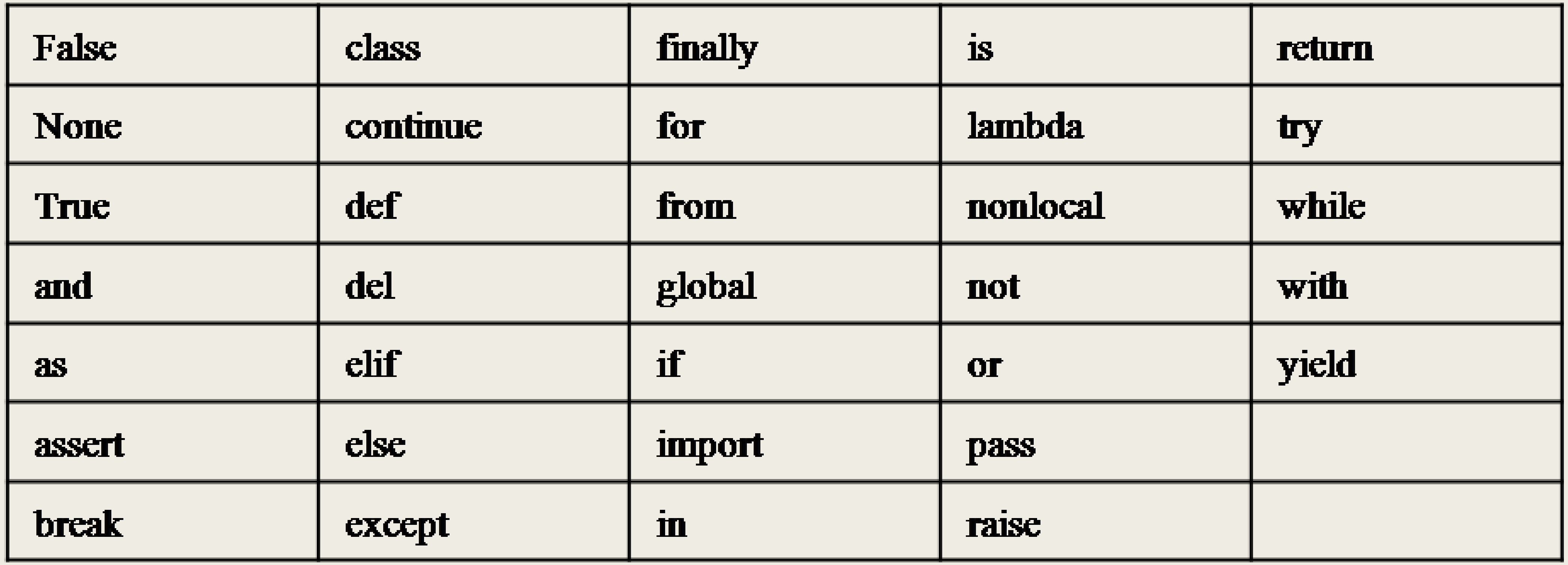
- Can’t contain space



```
In [41]: frist var=3

  File "<ipython-input-41-a6d58d7ca866>", line 1
    frist var=3
            ^
SyntaxError: invalid syntax
```

Variables:

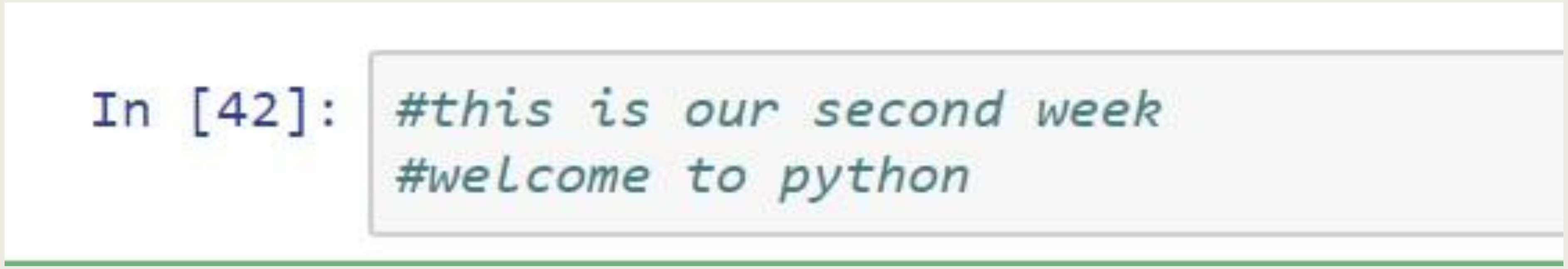
- <u>- Can’t be one of python keywords</u>



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	

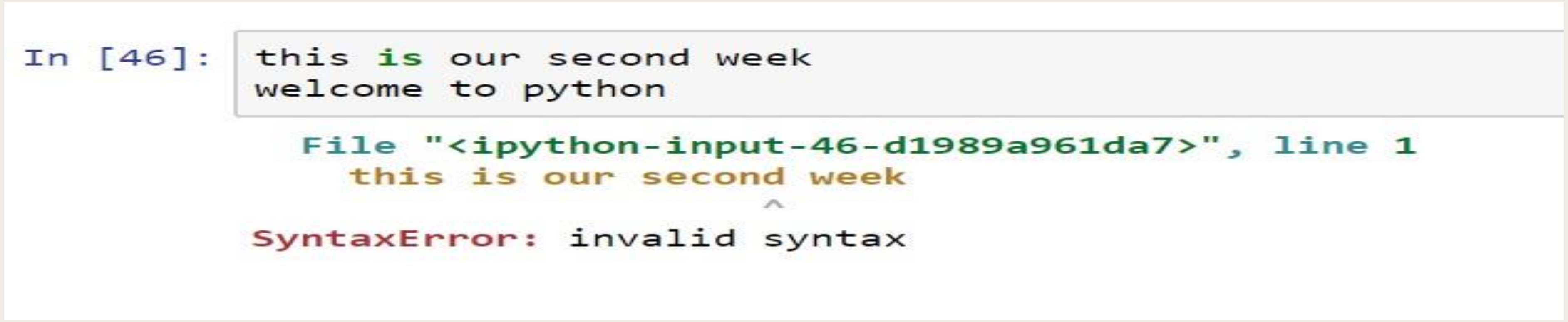
Comments

-To make a comment we use # before the line



```
In [42]: #this is our second week
         #welcome to python
```

-Without using # will give us an error



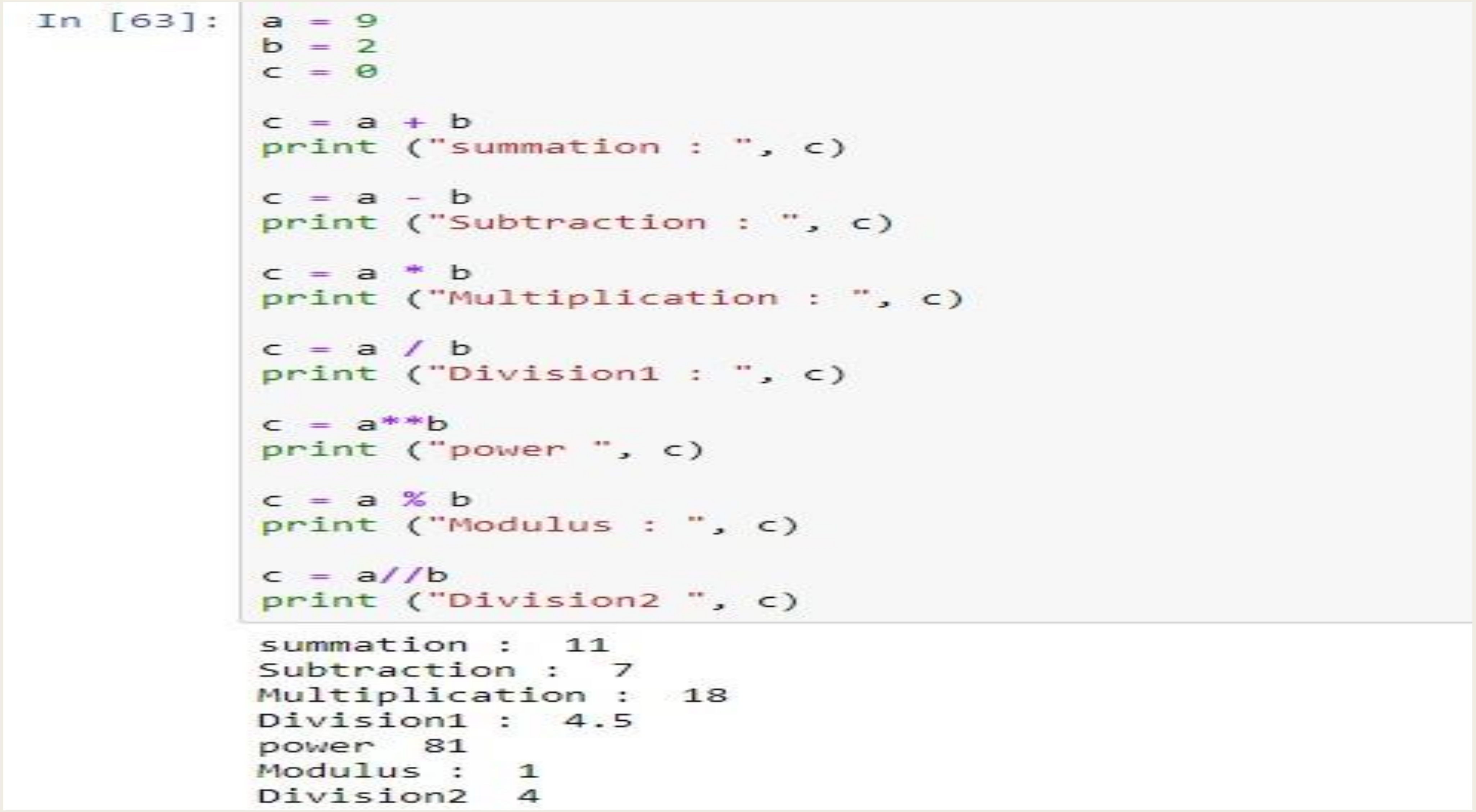
```
In [46]: this is our second week
         welcome to python

  File "<ipython-input-46-d1989a961da7>", line 1
    this is our second week
                ^
SyntaxError: invalid syntax
```

Abdelrahman A. Mohamed

Arithmetic operations

- (+) Addition
- (-)Subtraction
- (*)Multiplication
- (/) Division1
- (//) Division2
- (%) Modulus
- (**) power

Arithmetic operations

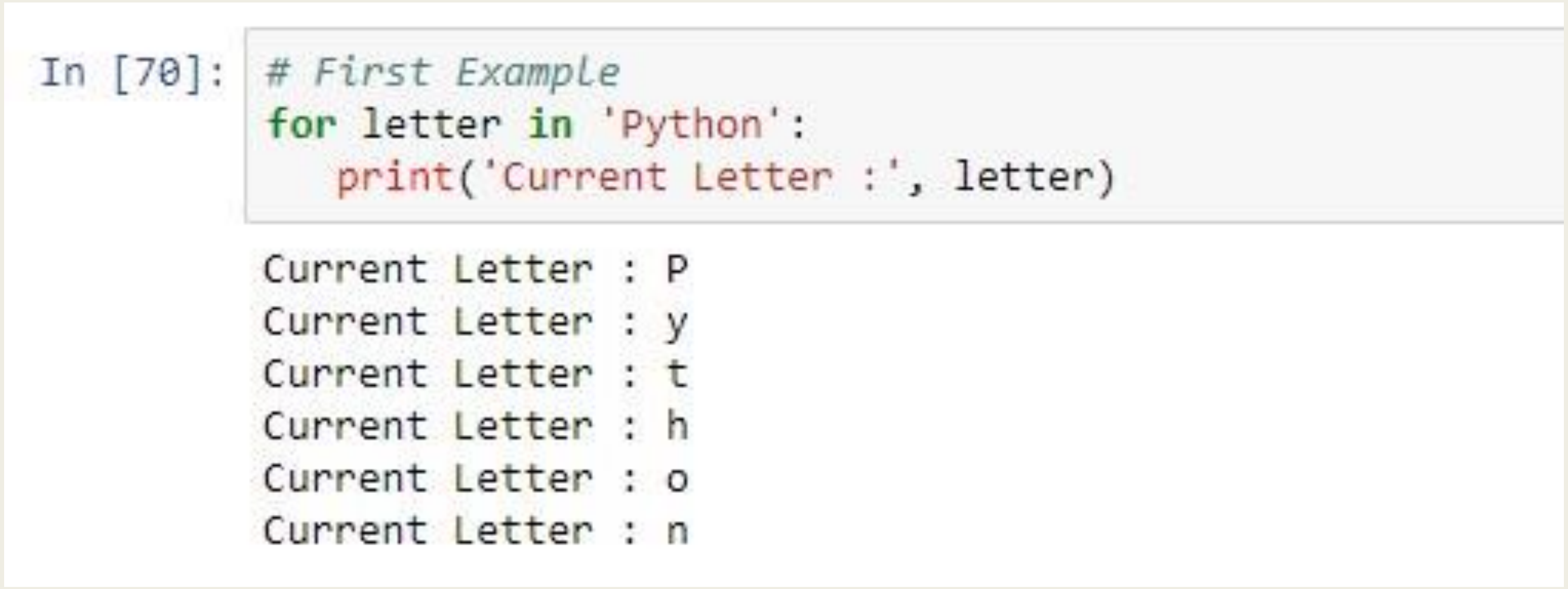


```
In [63]: a = 9
         b = 2
         c = 0

         c = a + b
         print ("summation : ", c)

         c = a - b
         print ("Subtraction : ", c)

         c = a * b
         print ("Multiplication : ", c)

         c = a / b
         print ("Division1 : ", c)

         c = a**b
         print ("power ", c)

         c = a % b
         print ("Modulus : ", c)

         c = a//b
         print ("Division2 ", c)
```

```
summation :  11
Subtraction :  7
Multiplication :  18
Division1 :  4.5
power  81
Modulus :  1
Division2  4
```

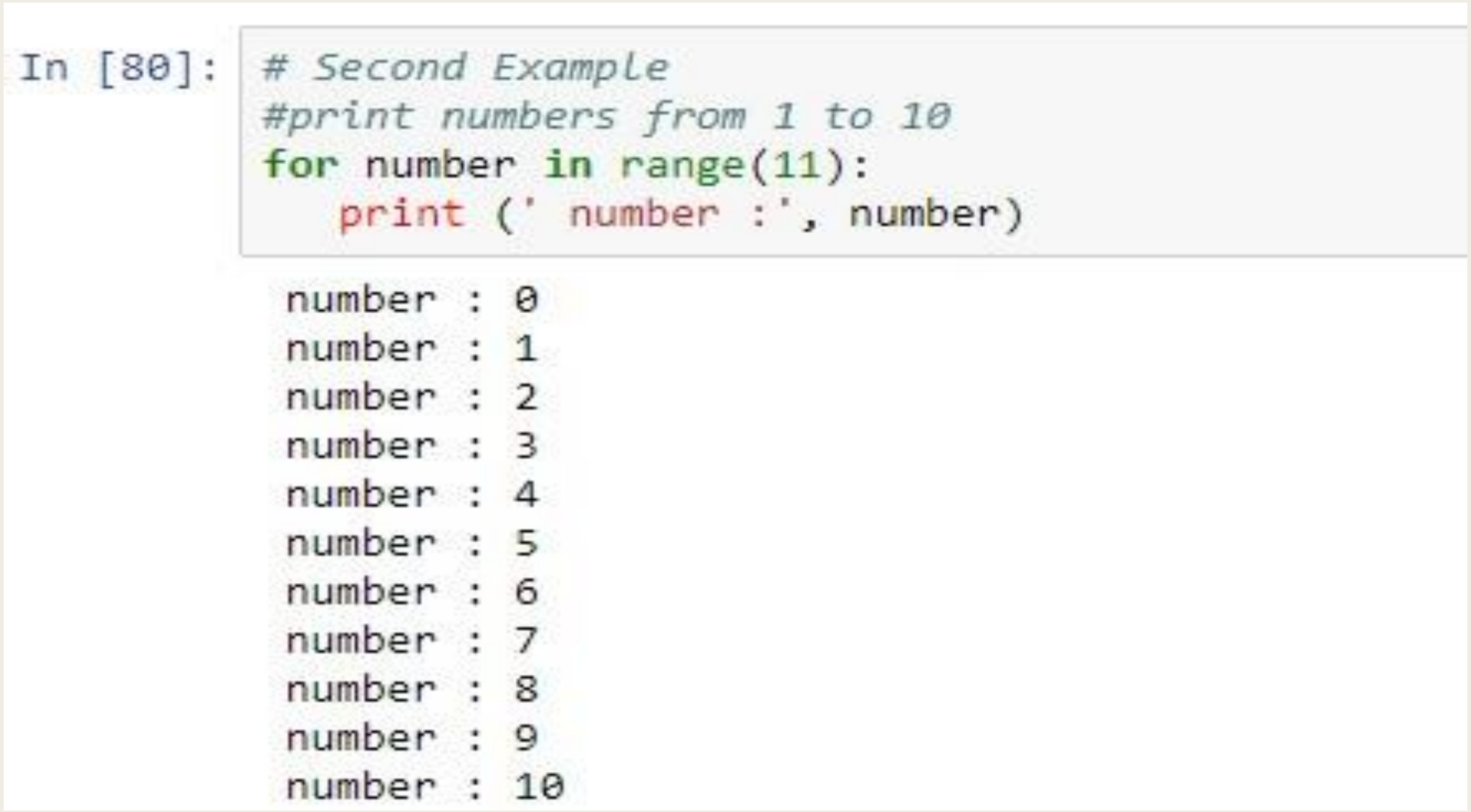
For loop

- Executes a sequence of statements multiple times.



```
In [70]: # First Example
         for letter in 'Python':
            print('Current Letter :', letter)

         Current Letter : P
         Current Letter : y
         Current Letter : t
         Current Letter : h
         Current Letter : o
         Current Letter : n
```

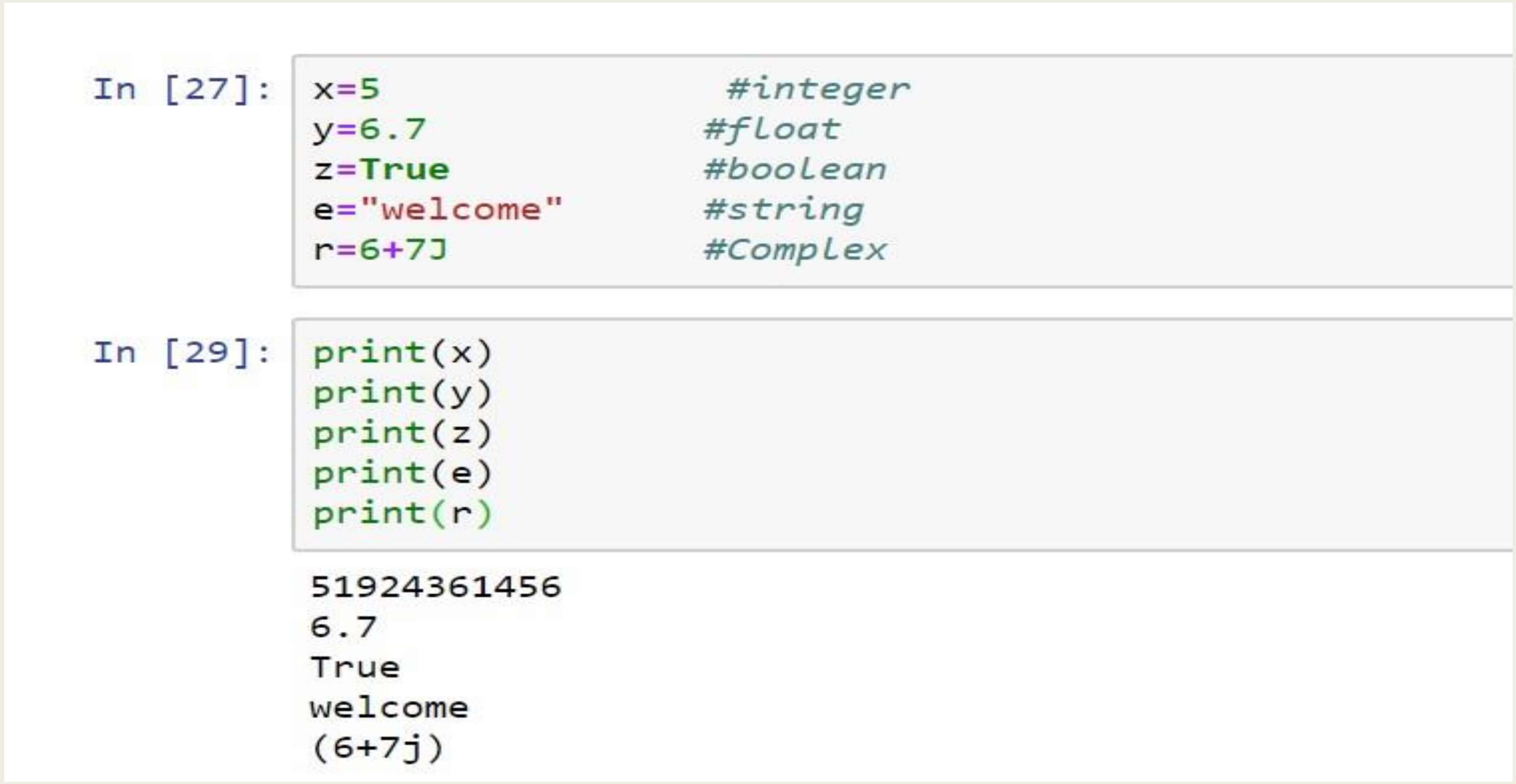
For loop



```
In [80]: # Second Example
         #print numbers from 1 to 10
         for number in range(11):
            print (' number :', number)
```

```
 number : 0
 number : 1
 number : 2
 number : 3
 number : 4
 number : 5
 number : 6
 number : 7
 number : 8
 number : 9
 number : 10
```

Print()

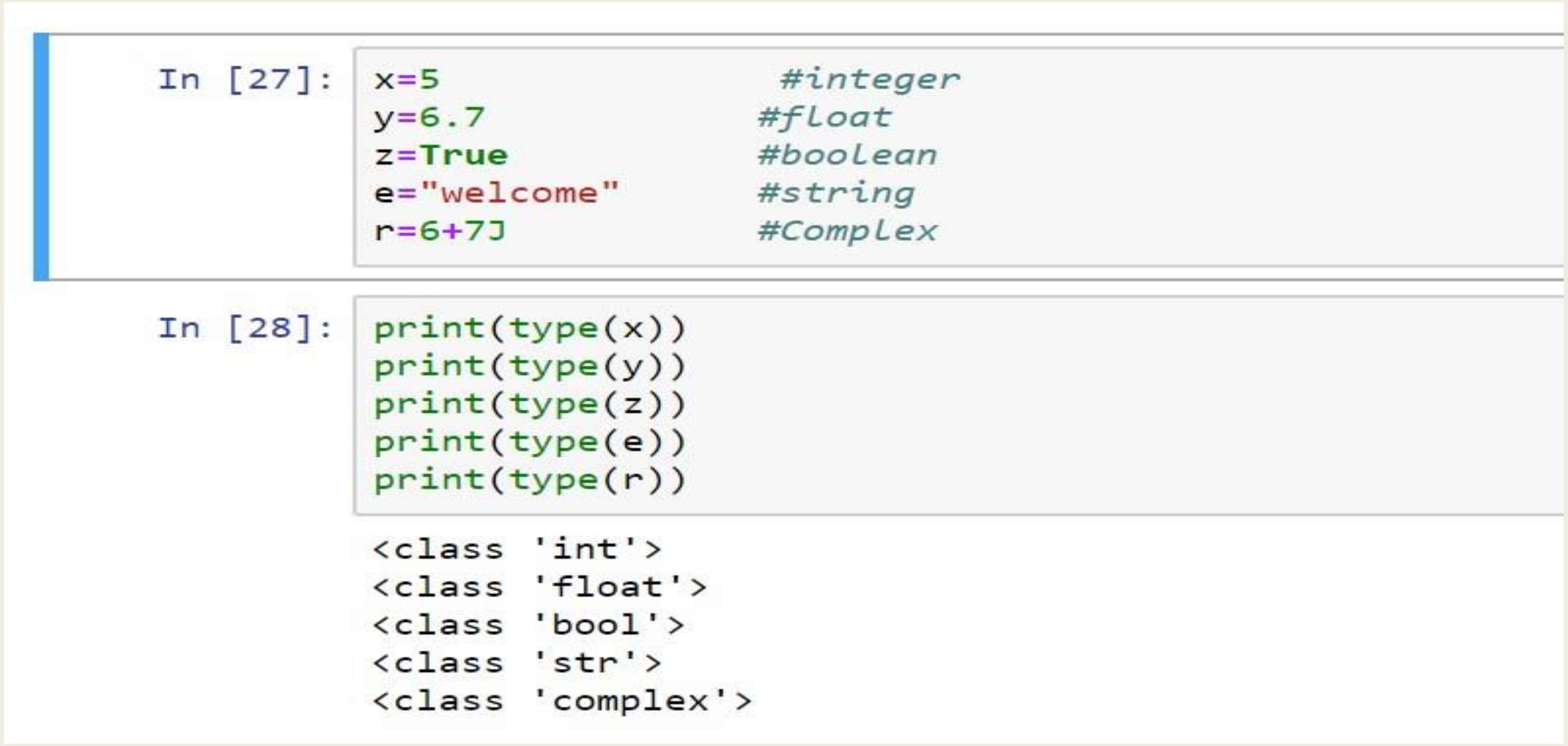


```
In [27]: x=5                          #integer
         y=6.7                       #float
         z=True                      #boolean
         e="welcome"                 #string
         r=6+7J                      #Complex
```

```
In [29]: print(x)
         print(y)
         print(z)
         print(e)
         print(r)
```

```
51924361456
6.7
True
welcome
(6+7j)
```

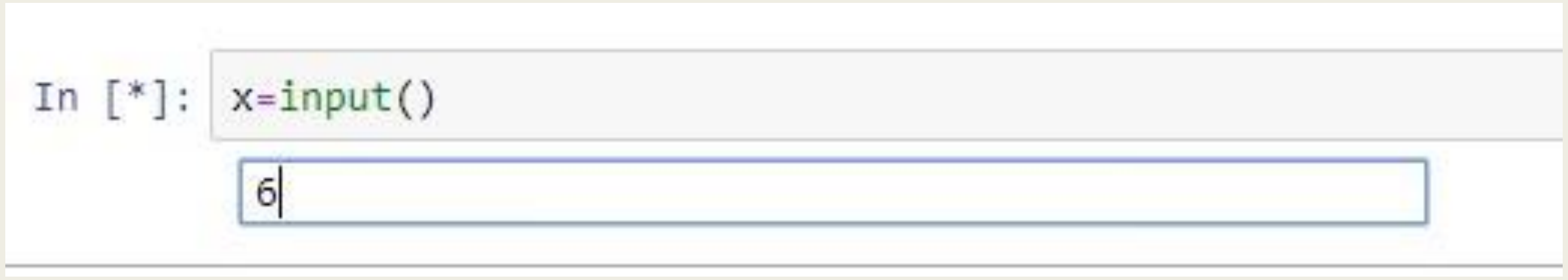
type()



```
In [27]: x=5                      #integer
         y=6.7                   #float
         z=True                  #boolean
         e="welcome"             #string
         r=6+7J                  #Complex
```

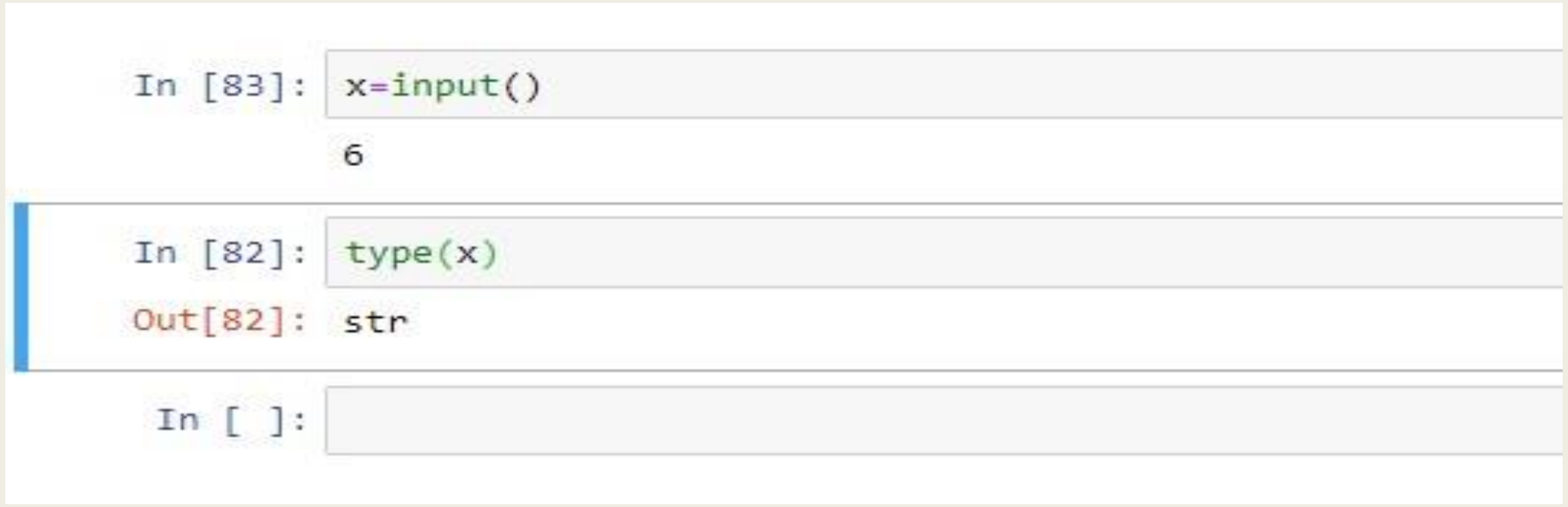
```
In [28]: print(type(x))
         print(type(y))
         print(type(z))
         print(type(e))
         print(type(r))

         <class 'int'>
         <class 'float'>
         <class 'bool'>
         <class 'str'>
         <class 'complex'>
```

input()



```
In [*]: x=input()
        6
```

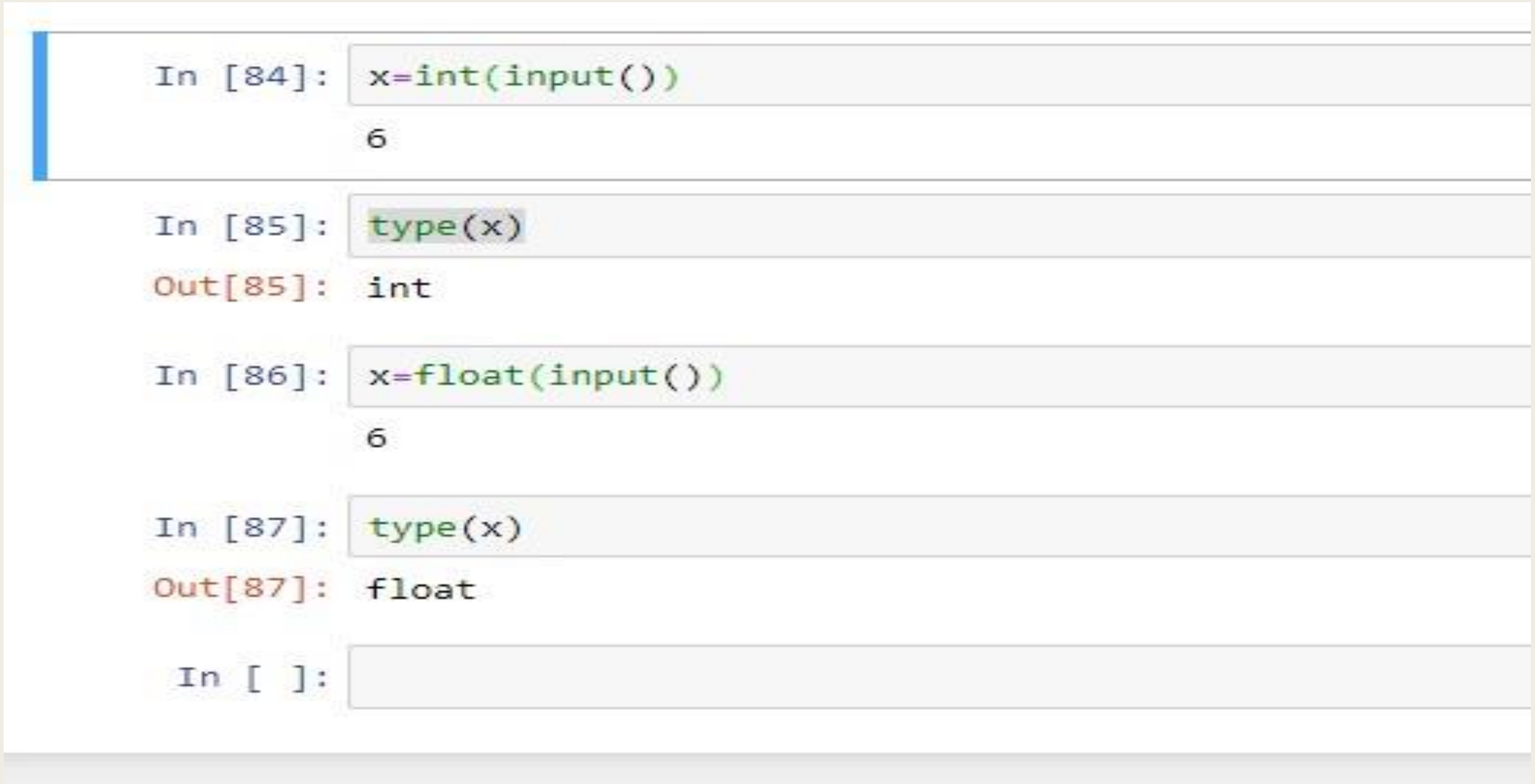
-Only take string data type:



```
In [83]: x=input()
         6

In [82]: type(x)
Out[82]: str

In [ ]:
```

input()

- To make the **input()** take an integer or float values:



```
In [84]: x=int(input())
         6

In [85]: type(x)
Out[85]: int

In [86]: x=float(input())
         6

In [87]: type(x)
Out[87]: float

In [ ]:
```