COL1000
Introduction to Programming

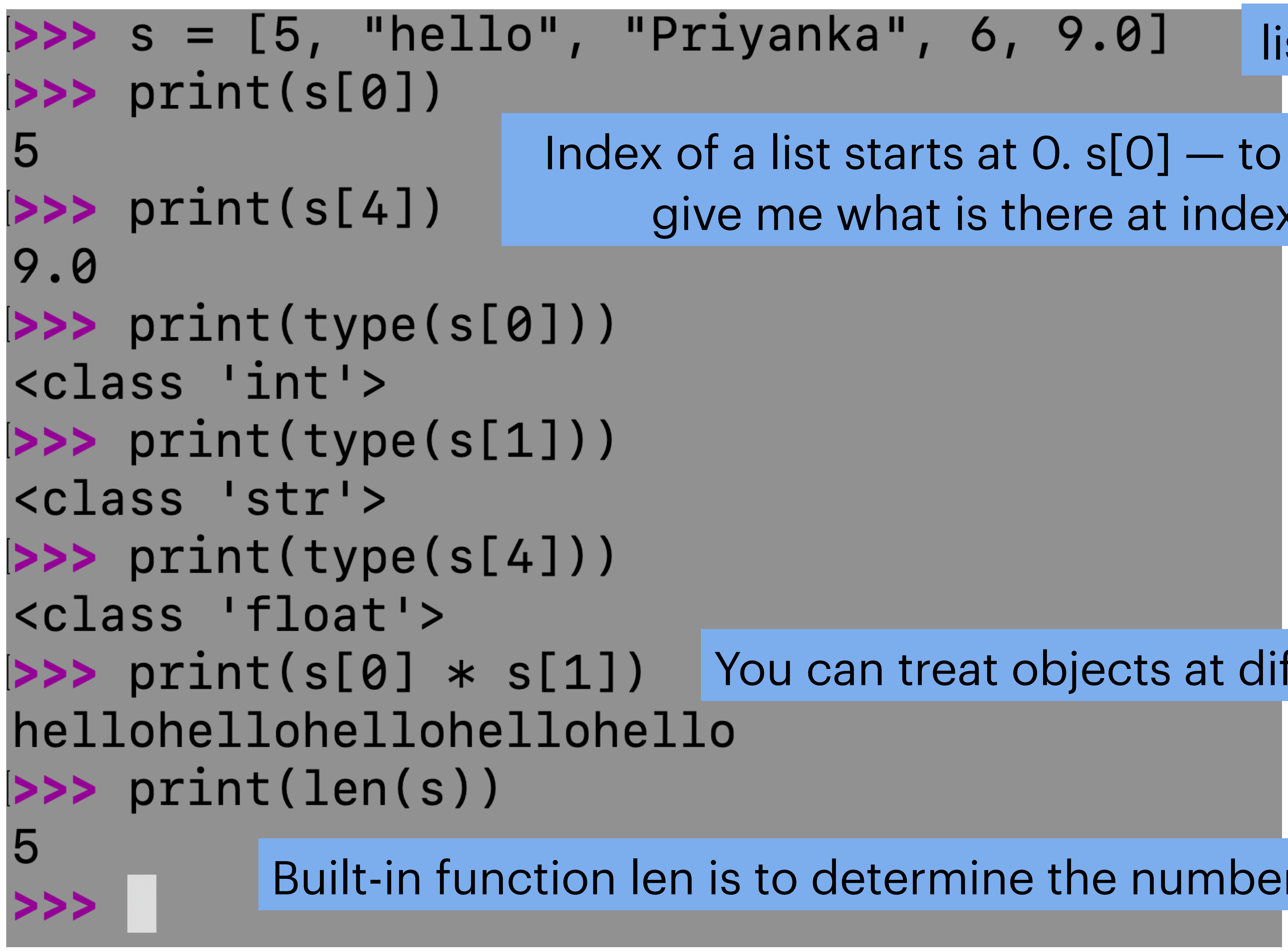
Priyanka Golia

Most (if not all) of the content is borrowed from Prof. Subodh Kumar's slides

Key Concepts

Lists

[] is used to indicate that it is a list



```
>>> s = [5, "hello", "Priyanka", 6, 9.0]
>>> print(s[0])
5
>>> print(s[4])
9.0
>>> print(type(s[0]))
<class 'int'>
>>> print(type(s[1]))
<class 'str'>
>>> print(type(s[4]))
<class 'float'>
>>> print(s[0] * s[1])
hellohellohellohellohello
>>> print(len(s))
5
>>> 
```

list s — Collection of different objects.

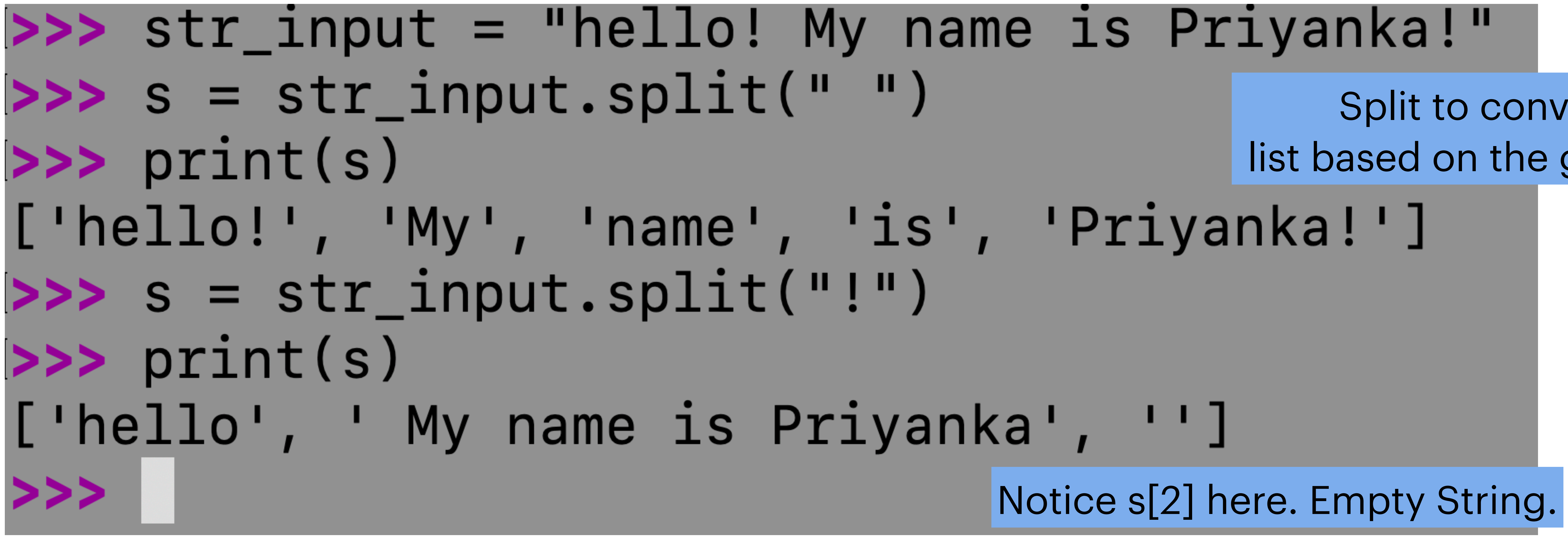
Index of a list starts at 0. s[0] — to tell python that give me what is there at index 0 in list s.

You can treat objects at different index individually

Built-in function len is to determine the number of objects in the list.

Key Concepts

Lists



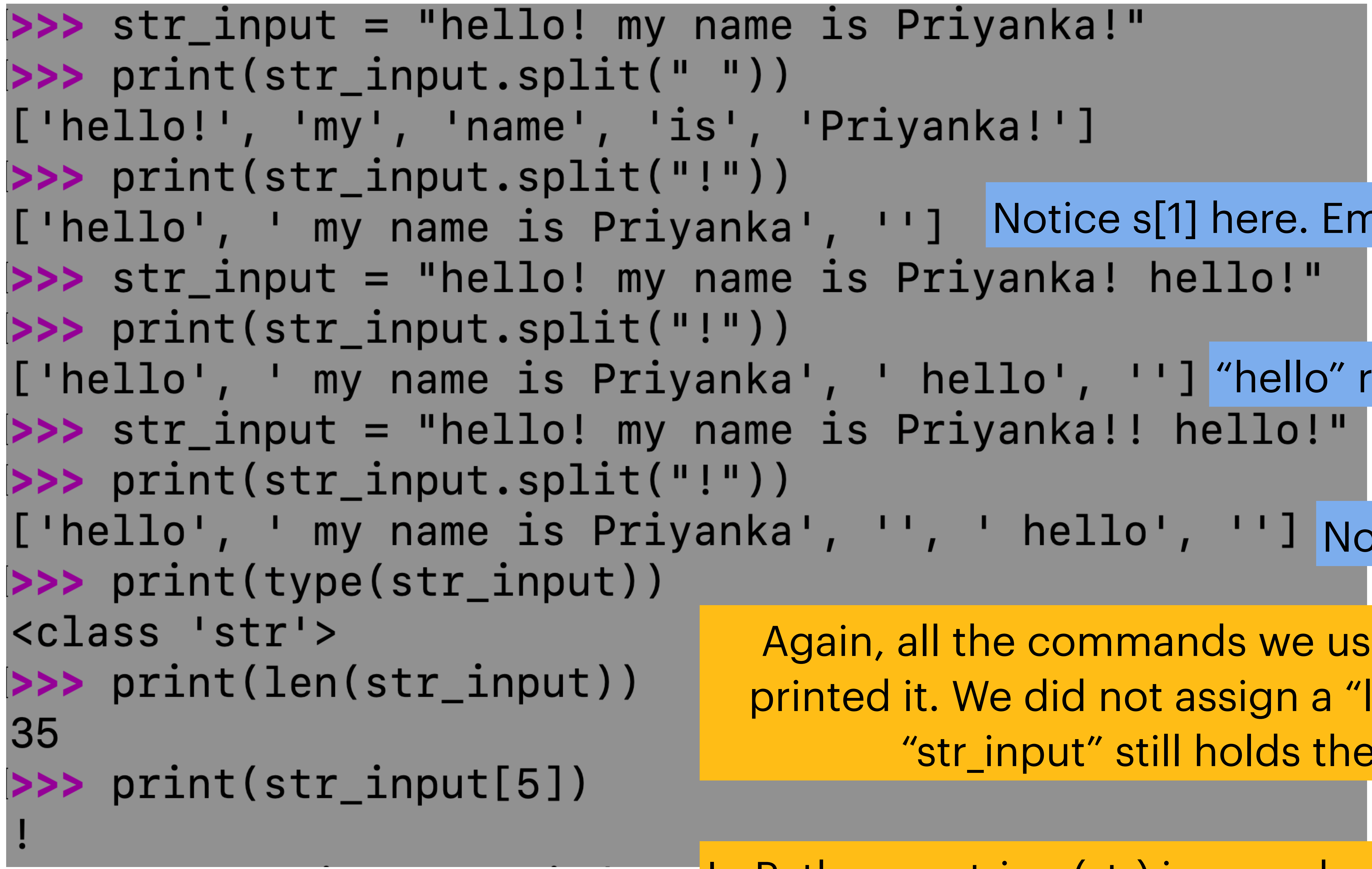
```
>>> str_input = "hello! My name is Priyanka!"
>>> s = str_input.split(" ")
>>> print(s)
['hello!', 'My', 'name', 'is', 'Priyanka!']
>>> s = str_input.split("!")
>>> print(s)
['hello', ' My name is Priyanka', '']
>>> 
```

Split to convert string into a list based on the given parameter (“ “)

Notice s[2] here. Empty String.

Key Concepts

Split is a built-in function, just like “type”, or type conversion.



```
>>> str_input = "hello! my name is Priyanka!"
>>> print(str_input.split(" "))
['hello!', 'my', 'name', 'is', 'Priyanka!']
>>> print(str_input.split("!"))
['hello', ' my name is Priyanka', '']
>>> str_input = "hello! my name is Priyanka! hello!"
>>> print(str_input.split("!"))
['hello', ' my name is Priyanka', ' hello', '']
>>> str_input = "hello! my name is Priyanka!! hello!"
>>> print(str_input.split("!"))
['hello', ' my name is Priyanka', '', ' hello', '']
>>> print(type(str_input))
<class 'str'>
>>> print(len(str_input))
35
>>> print(str_input[5])
!
```

Notice s[1] here. Empty String.

“hello” repeated.

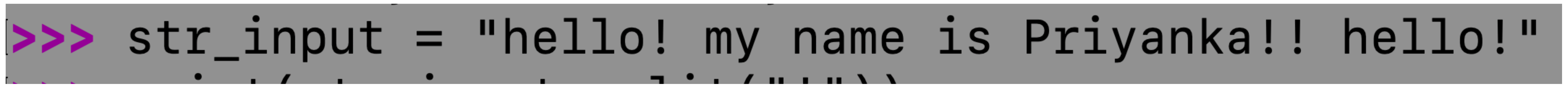
Notice s[1] here. Empty String. We had !! In the string.

Again, all the commands we used so far created a new object and printed it. We did not assign a “label” (or variable) to that object, so “str_input” still holds the value we originally defined

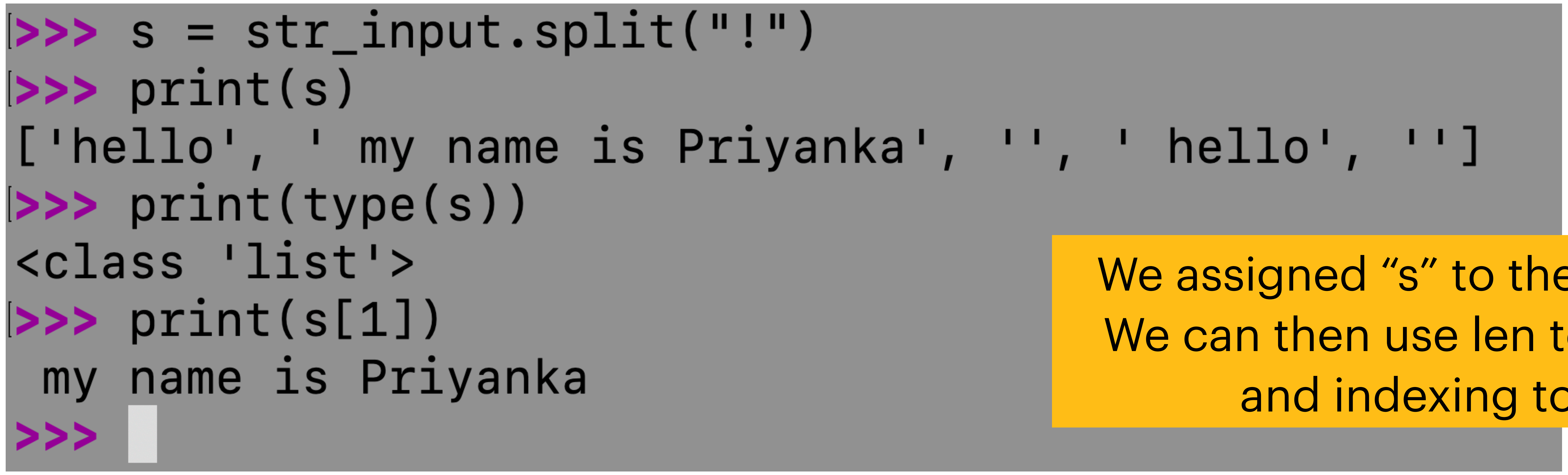
In Python, a string (str) is an ordered collection of characters, much like a list, but immutable. You can access each character using indexing. Remember, we classified strings as a compound data type in Lecture 3.

Highlevel- immutable means that once an object is created, its contents cannot be changed.

Key Concepts

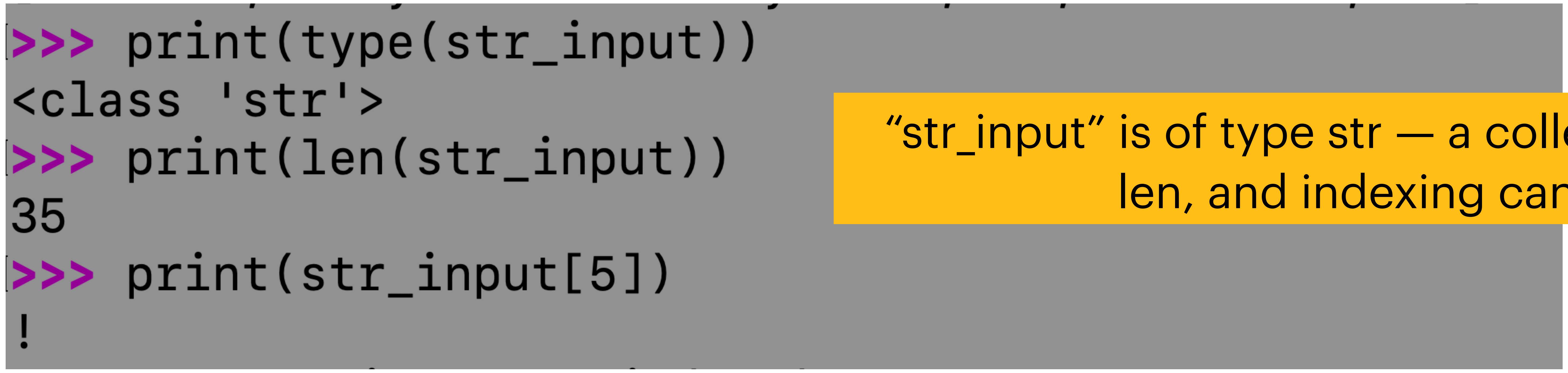


```
>>> str_input = "hello! my name is Priyanka!! hello!"
```



```
>>> s = str_input.split("!")
>>> print(s)
['hello', ' my name is Priyanka', '', ' hello', '']
>>> print(type(s))
<class 'list'>
>>> print(s[1])
 my name is Priyanka
>>> 
```

We assigned “s” to the list returned by the split() method. We can then use len to find the number of elements in s, and indexing to access individual elements.



```
>>> print(type(str_input))
<class 'str'>
>>> print(len(str_input))
35
>>> print(str_input[5])
!
```

“str_input” is of type str — a collection of characters. len, and indexing can be used.

Conditional Execution

Conditionals let a program make decisions based on tests.

For example: "If it's raining, take an umbrella. Otherwise, don't."

The decision depends on the condition: "Is it raining?"

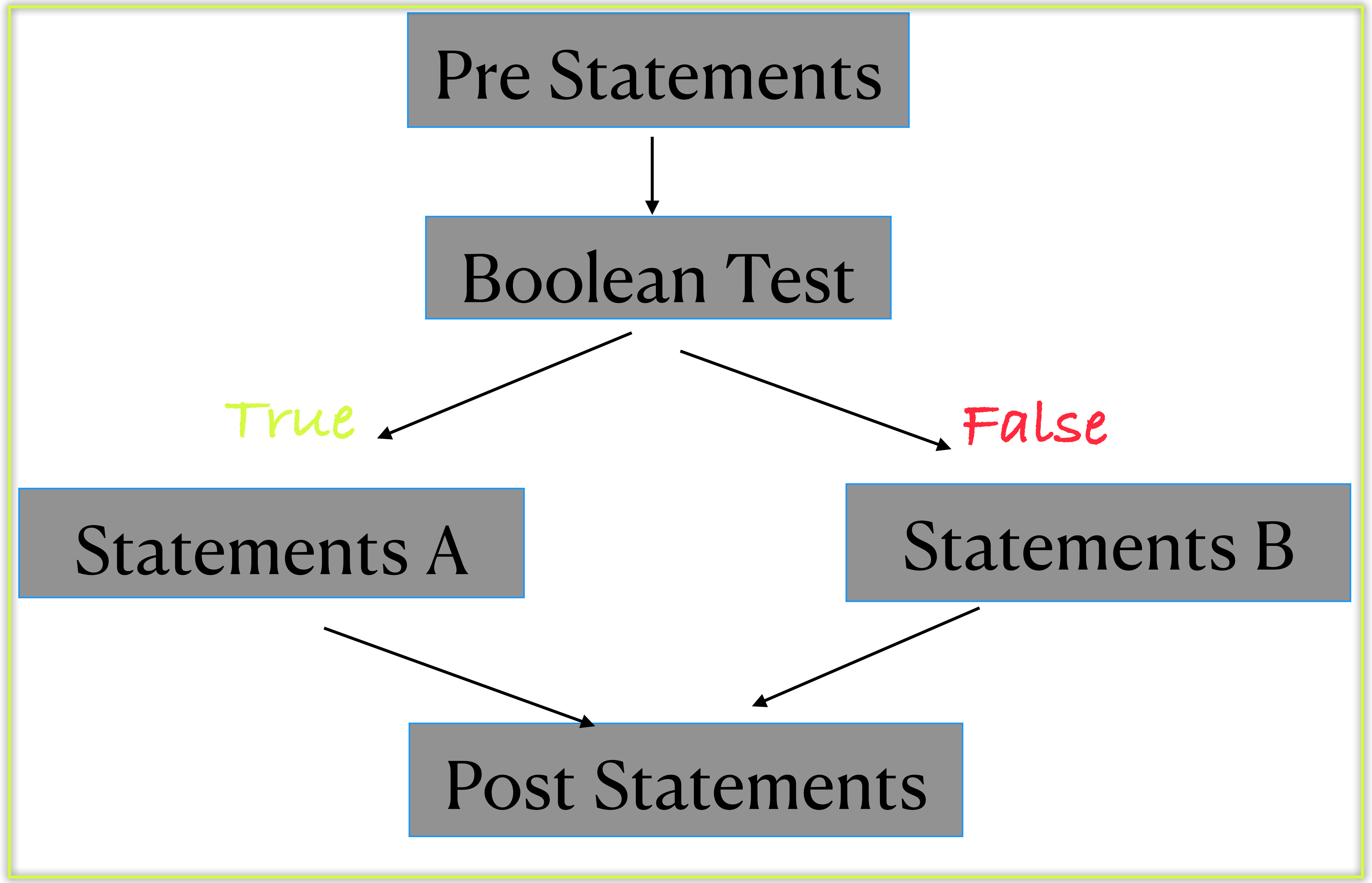
Every conditional is based on a **Boolean** test — something that evaluates to either: True or False

Python uses these True or False values to decide which part of the code to run.

Conditional Execution

Every conditional is based on a **Boolean** test — something that evaluates to either: True or False

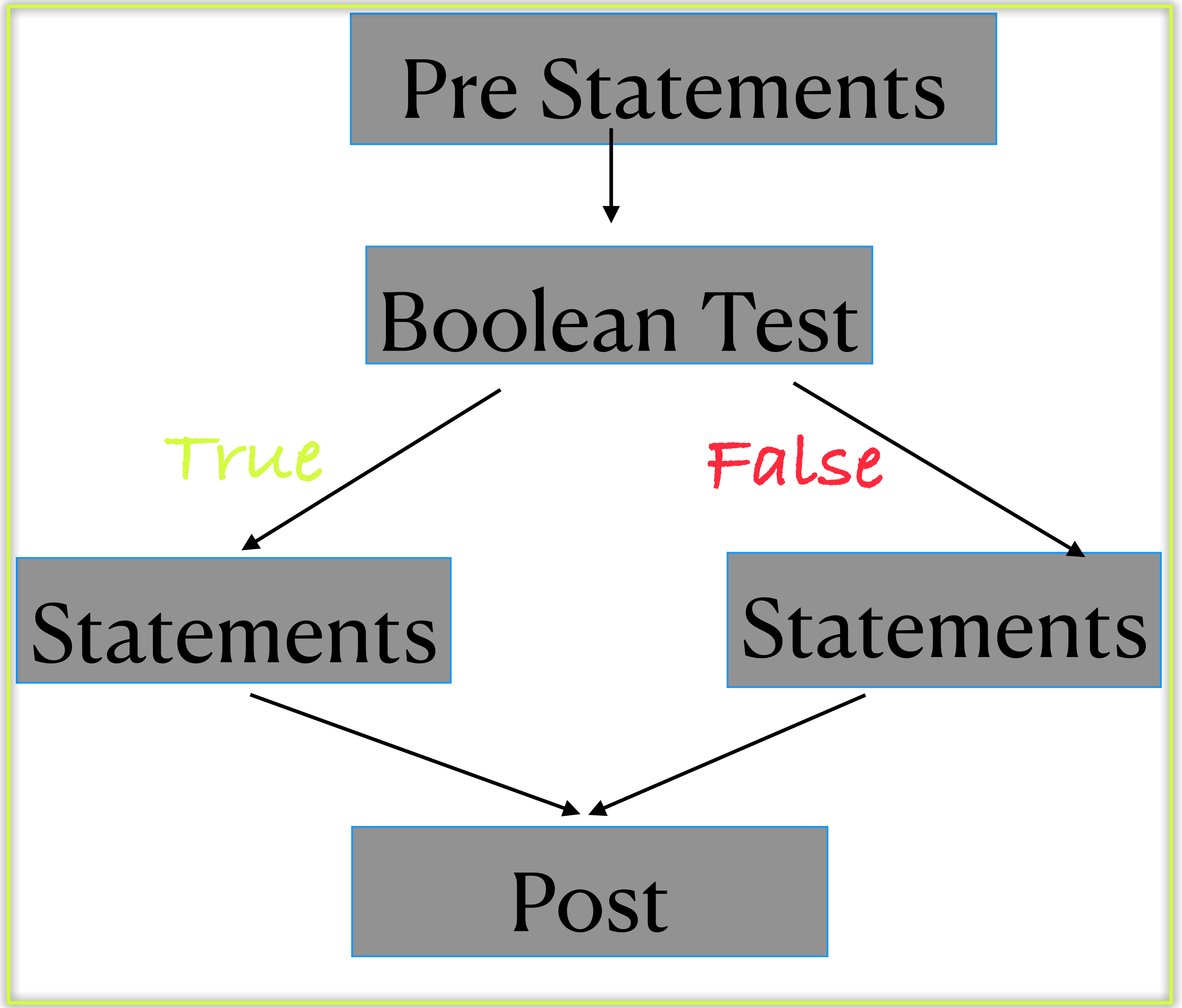
Python uses these True or False values to decide which part of the code to run.

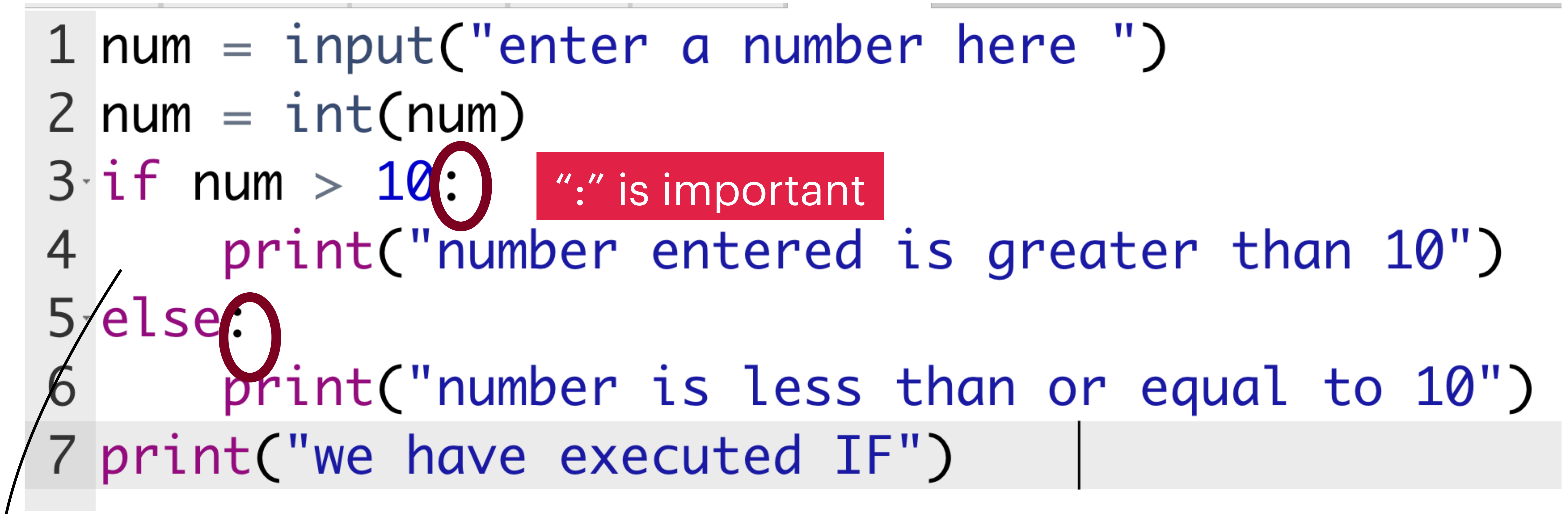


Conditional Execution

Every conditional is based on a **Boolean** test — something that evaluates to either: True or False

Python uses these True or False values to decide which part of the code to run.





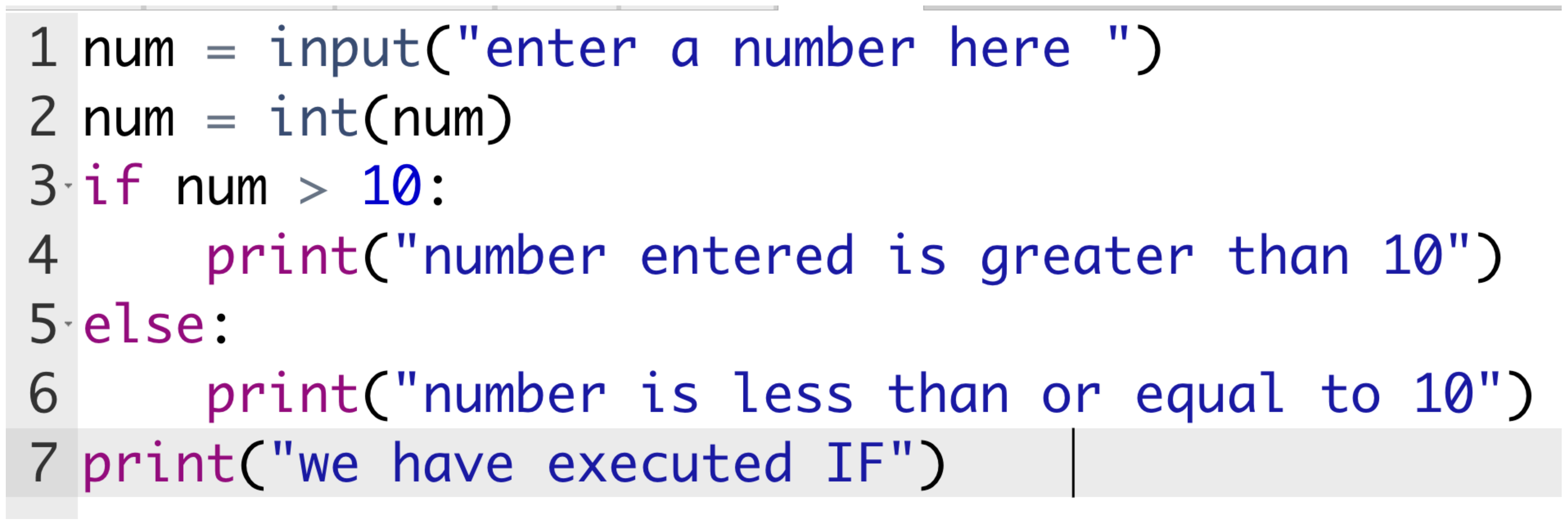
```
num = input("enter a number here ")
num = int(num)
if num > 10:
    print("number entered is greater than 10")
else:
    print("number is less than or equal to 10")
print("we have executed IF")
```

":" is important

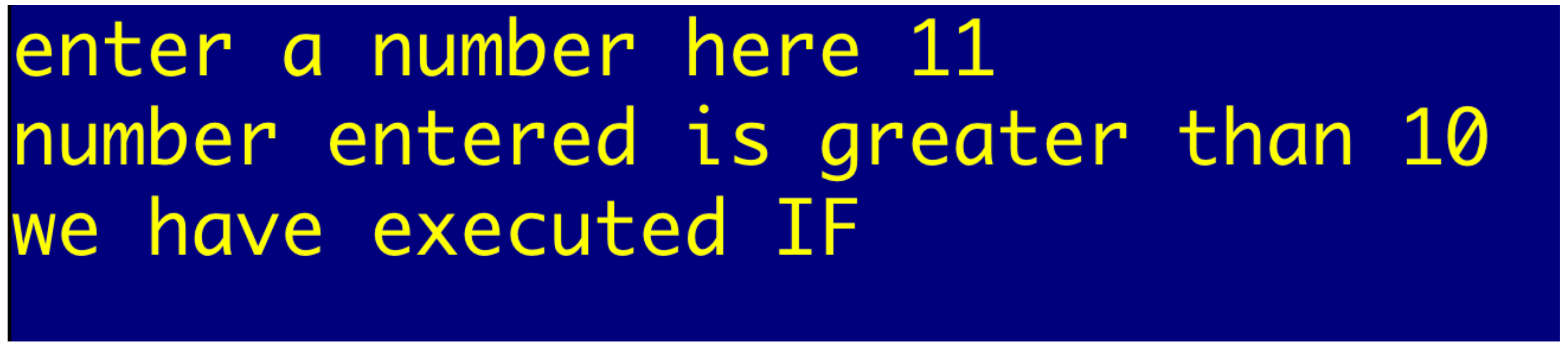
Colon (:) tells that a block of code is coming next, and it belongs to this statement.

Indentation (tabs or spaces) is used to define blocks of code

Conditional Execution

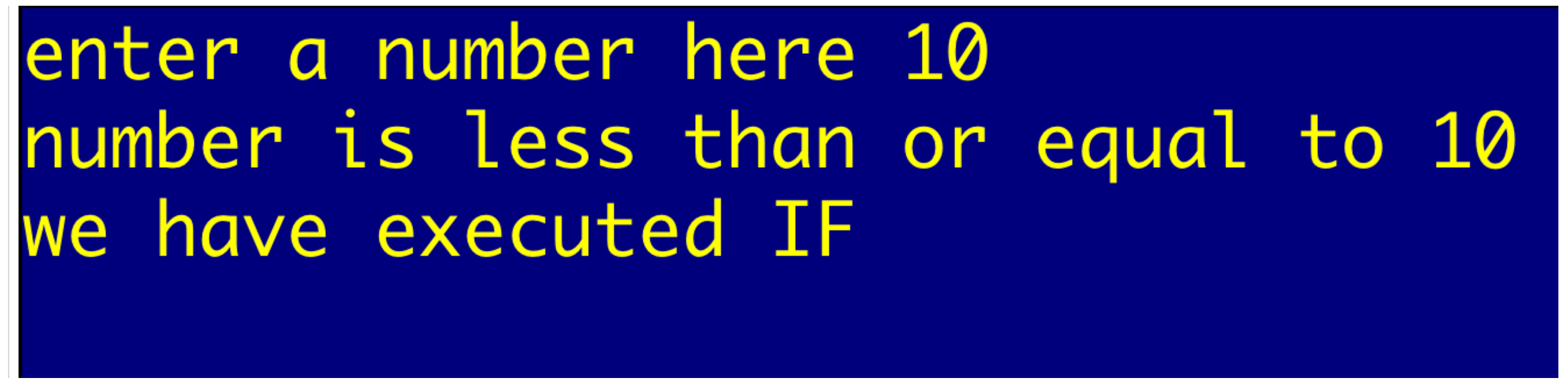


```python
num = input("enter a number here ")
num = int(num)
if num > 10:
    print("number entered is greater than 10")
else:
    print("number is less than or equal to 10")
print("we have executed IF")
```



```
enter a number here 11
number entered is greater than 10
we have executed IF
```

Number entered is 11, it executed the pre statement(s), checked for condition, and executed IF part, and then post statement(s).



```
enter a number here 10
number is less than or equal to 10
we have executed IF
```

Number entered is 10, it executed the pre statement(s), checked for condition, and executed else part, and then post statement(s).

Conditional Execution

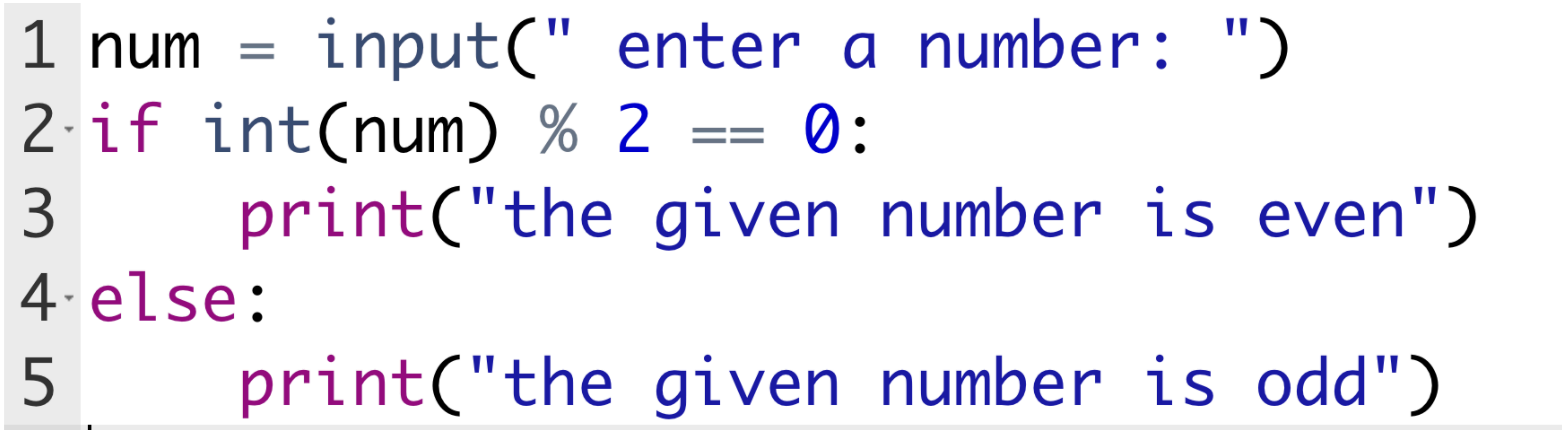
If Without else — that’s totally fine!

if block without an else — do something only if the condition is True. Otherwise, do nothing.

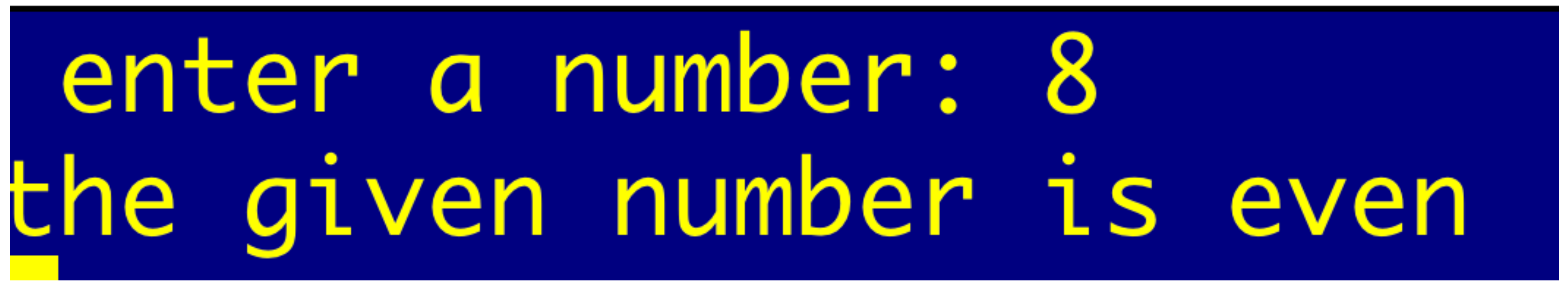
```
marks = int(input(“enter your marks:  ”))
if marks >= 90:
        print(“you did a good job)
```

No else is needed!

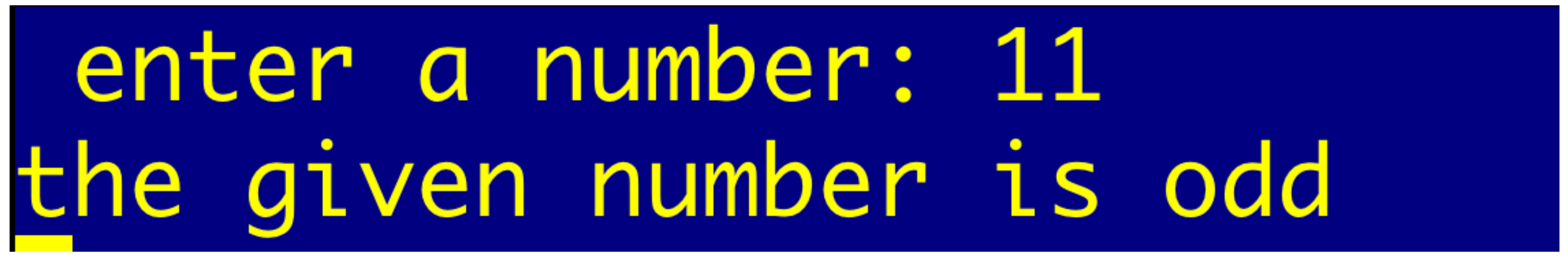
It is okay not to have ELSE



```python
num = input(" enter a number: ")
if int(num) % 2 == 0:
    print("the given number is even")
else:
    print("the given number is odd")
```



```
 enter a number: 8
the given number is even
```



```
 enter a number: 11
the given number is odd
```