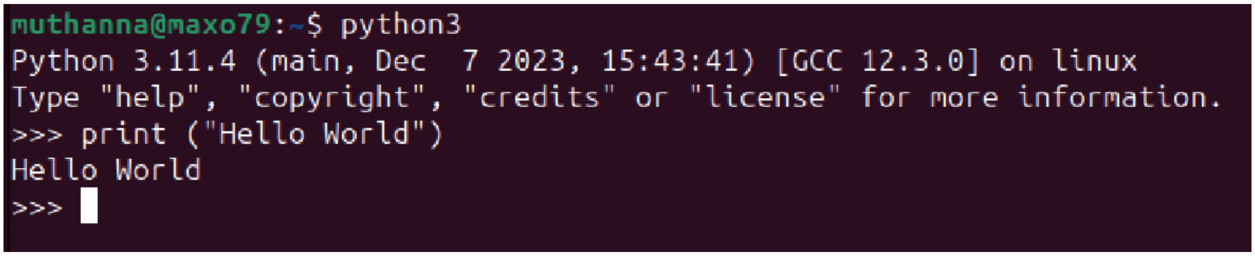
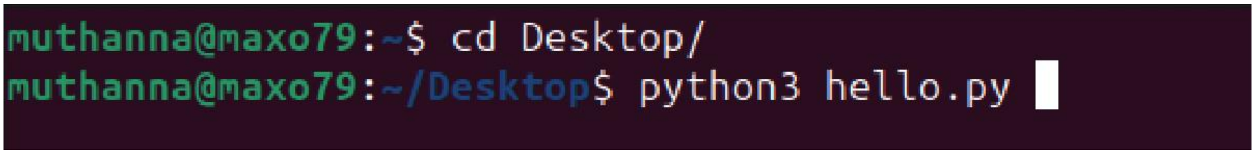
2. Python Syntax

As we learned in the previous, Python syntax can be executed by writing directly in the Command Line:



```
muthanna@maxo79:~$ python3
Python 3.11.4 (main, Dec  7 2023, 15:43:41) [GCC 12.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print ("Hello World")
Hello World
>>> 
```

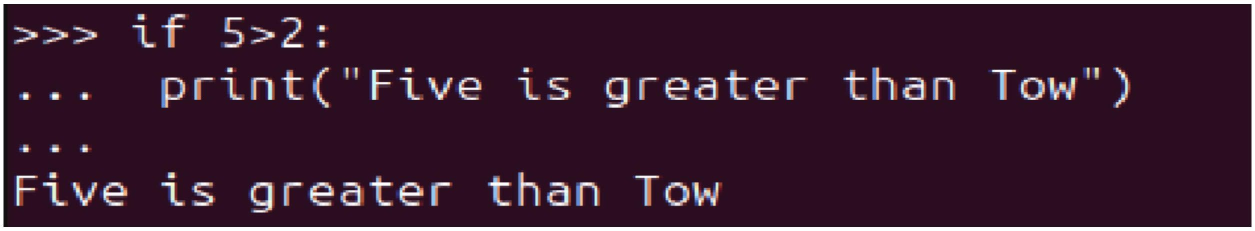
Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:



```
muthanna@maxo79:~$ cd Desktop/
muthanna@maxo79:~/Desktop$ python3 hello.py 
```

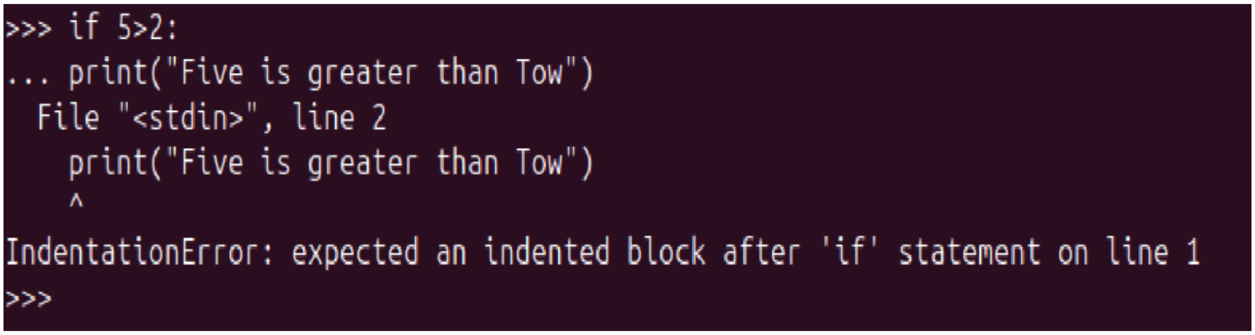
2.1 Python Indentation

Indentation refers to the spaces at the beginning of a code line. Where in other programming languages the indentation in code is for readability only, the indentation in Python is very important. Python uses indentation to indicate a block of code.



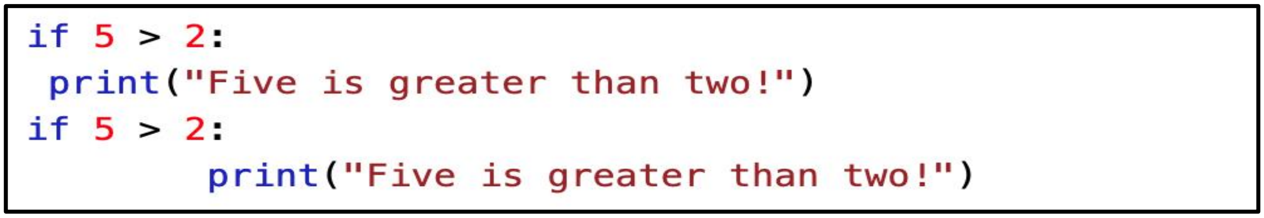
```
>>> if 5>2:
...   print("Five is greater than Tow")
...
Five is greater than Tow
```

Python will give you an error if you skip the indentation:

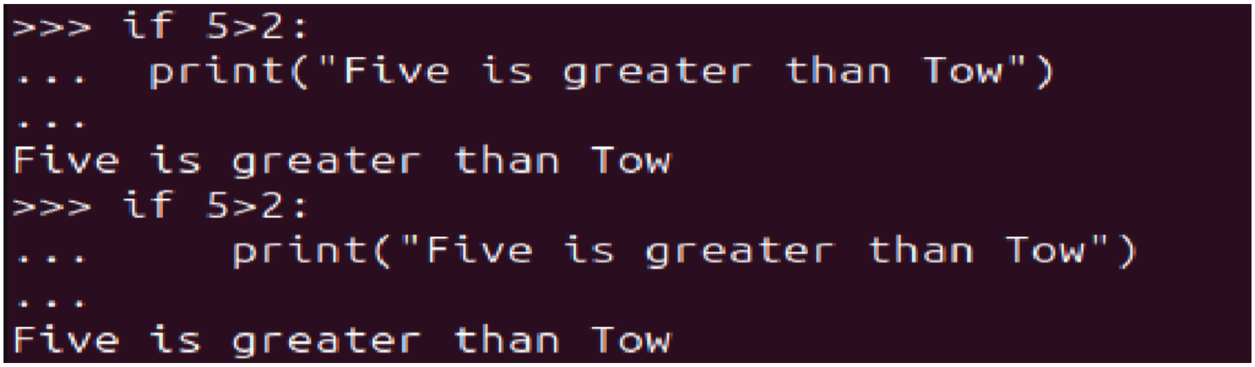


```
>>> if 5>2:
... print("Five is greater than Tow")
  File "<stdin>", line 2
    print("Five is greater than Tow")
    ^
IndentationError: expected an indented block after 'if' statement on line 1
>>>
```

The number of spaces is up to you as a programmer, the most common use is four, but it has to be at least one.

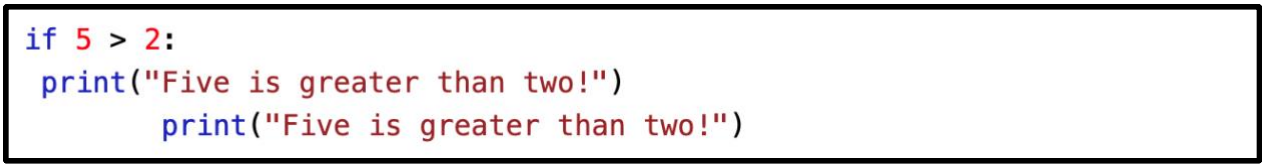


```
if 5 > 2:
 print("Five is greater than two!")
if 5 > 2:
        print("Five is greater than two!")
```

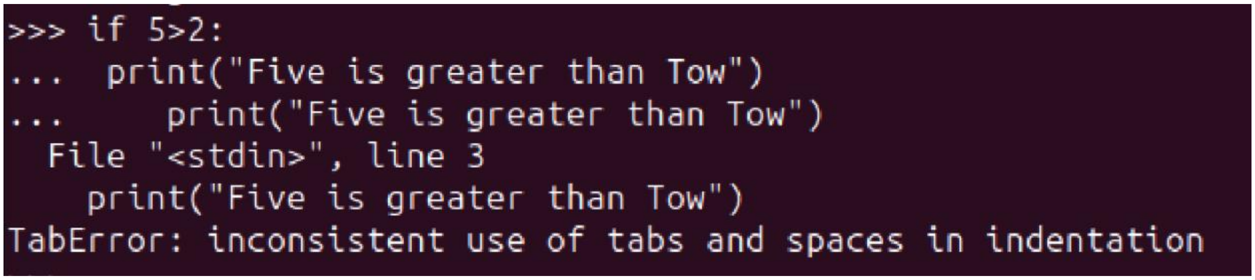


```
>>> if 5>2:
...  print("Five is greater than Tow")
...
Five is greater than Tow
>>> if 5>2:
...      print("Five is greater than Tow")
...
Five is greater than Tow
```

You have to use the same number of spaces in the same block of code, otherwise Python will give you an error:

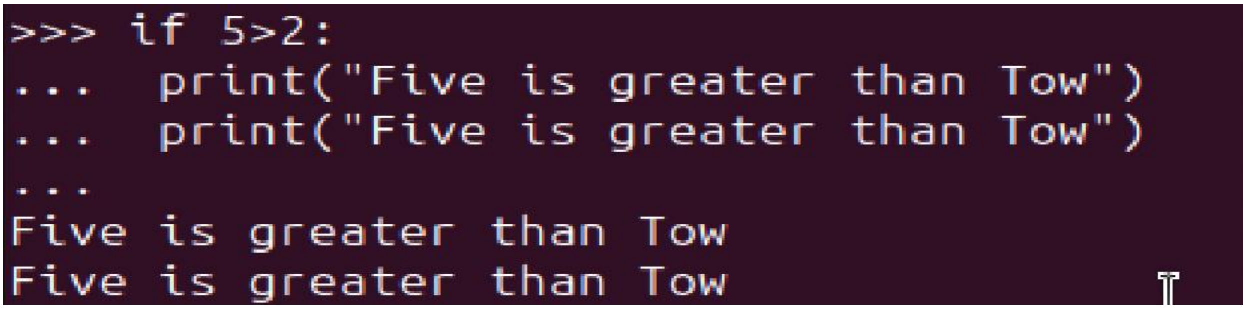


```
if 5 > 2:
 print("Five is greater than two!")
        print("Five is greater than two!")
```



```
>>> if 5>2:
...  print("Five is greater than Tow")
...     print("Five is greater than Tow")
  File "<stdin>", line 3
    print("Five is greater than Tow")
TabError: inconsistent use of tabs and spaces in indentation
```

See the correct:



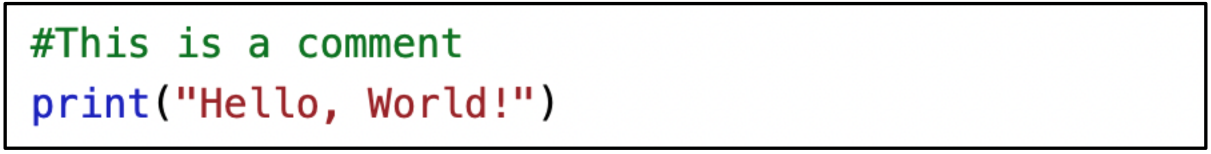
```
>>> if 5>2:
...  print("Five is greater than Tow")
...  print("Five is greater than Tow")
...
Five is greater than Tow
Five is greater than Tow
```

2.2 Python Comments

- Comments can be used to explain Python code.
- Comments can be used to make the code more readable.
- Comments can be used to prevent execution when testing code.

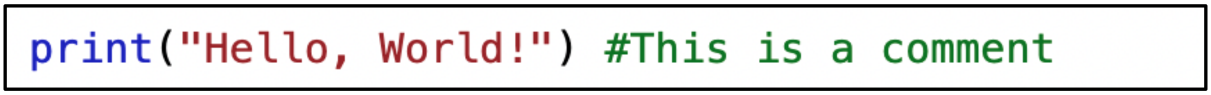
Creating a Comment

Comments starts with a #, and Python will ignore them:



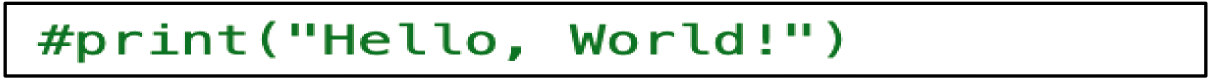
```
#This is a comment
print("Hello, World!")
```

Comments can be placed at the end of a line, and Python will ignore the rest of the line:



```
print("Hello, World!") #This is a comment
```

A comment does not have to be text that explains the code, it can also be used to prevent Python from executing code:

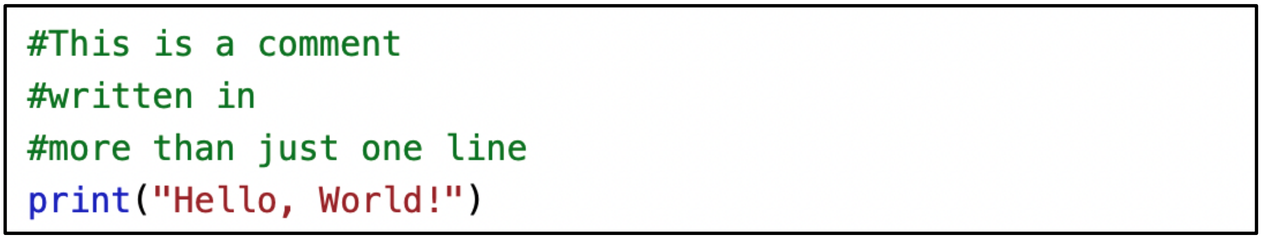


```
#print("Hello, World!")
```

Multiline Comments

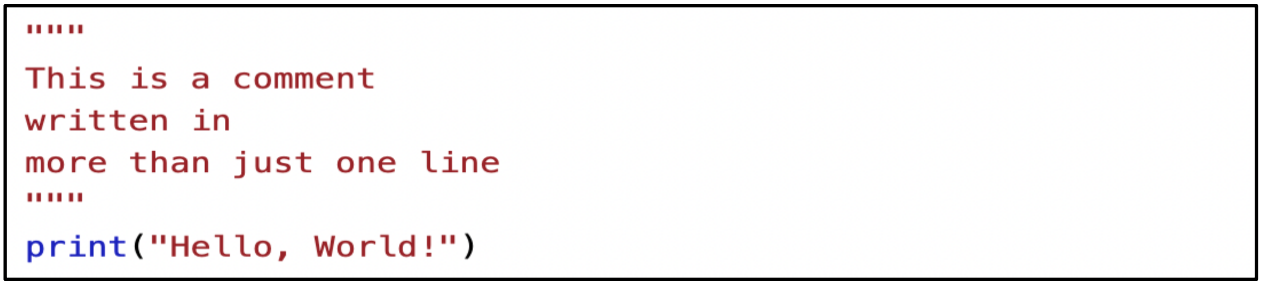
Python does not really have a syntax for multiline comments.

To add a multiline comment you could insert a # for each line:



```
#This is a comment
#written in
#more than just one line
print("Hello, World!")
```

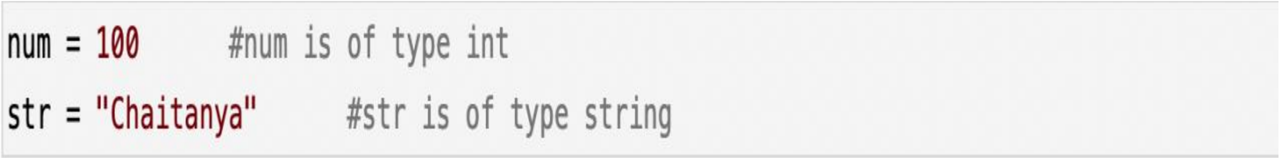
Or, not quite as intended, you can use a multiline string.
Since Python will ignore string literals that are not assigned to a variable, you can add a multiline string (triple quotes) in your code, and place your comment inside it:



```
"""
This is a comment
written in
more than just one line
"""
print("Hello, World!")
```

2.3 Python Variables

Variables are used to store data, they take memory space based on the type of value we assigning to them. Creating variables in Python is simple, you just have write the variable name on the left side of = and the value on the right side, as shown below. You do not have to explicitly mention the type of the variable, python infer the type based on the value we are assigning.



```
num = 100      #num is of type int
str = "Chaitanya"      #str is of type string
```

Variable name – Identifiers

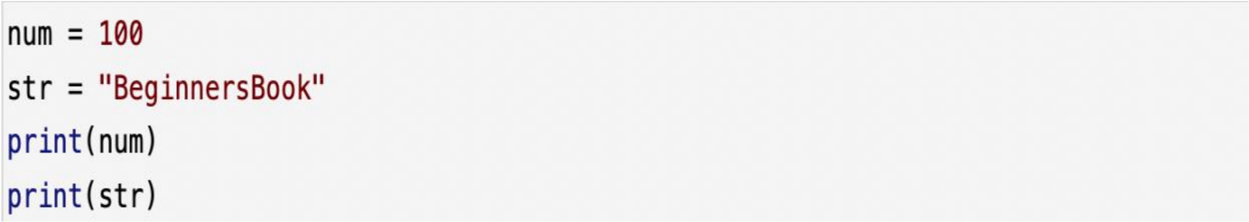
Variable name is known as identifier. There are few rules that you have to follow while naming the variables in Python.

1. The name of the variable must always start with either a letter or an underscore (_). For example: _str, str, num, _num are all valid name for the variables.

2. The name of the variable cannot start with a number. For example: 9num is not a valid variable name.

3. The name of the variable cannot have special characters such as %, $, # etc, they can only have alphanumeric characters and underscore (A to Z, a to z, 0-9 or _).

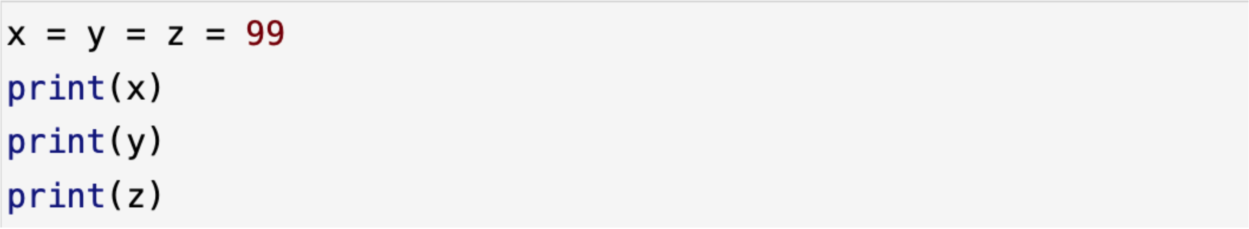
4. Variable name is case sensitive in Python which means num and NUM are two different variables in python.



```
num = 100
str = "BeginnersBook"
print(num)
print(str)
```

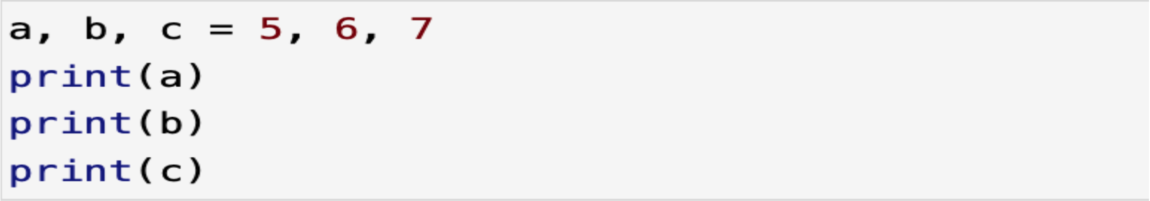
Python multiple assignment

We can assign multiple variables in a single statement like this in Python.



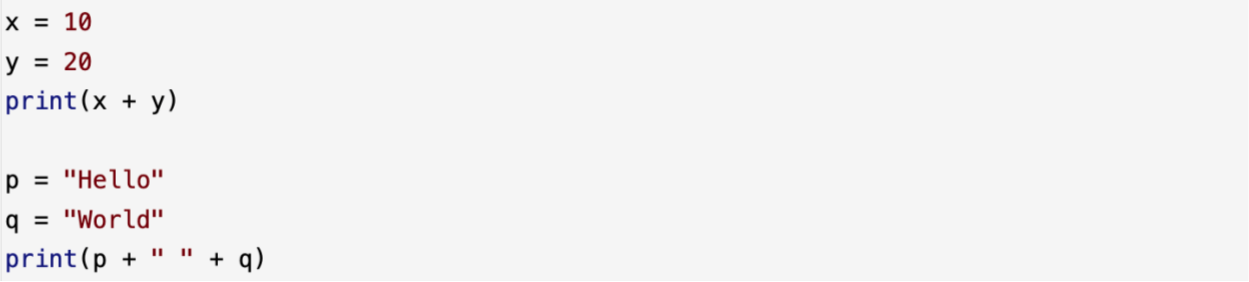
```
x = y = z = 99
print(x)
print(y)
print(z)
```

Or

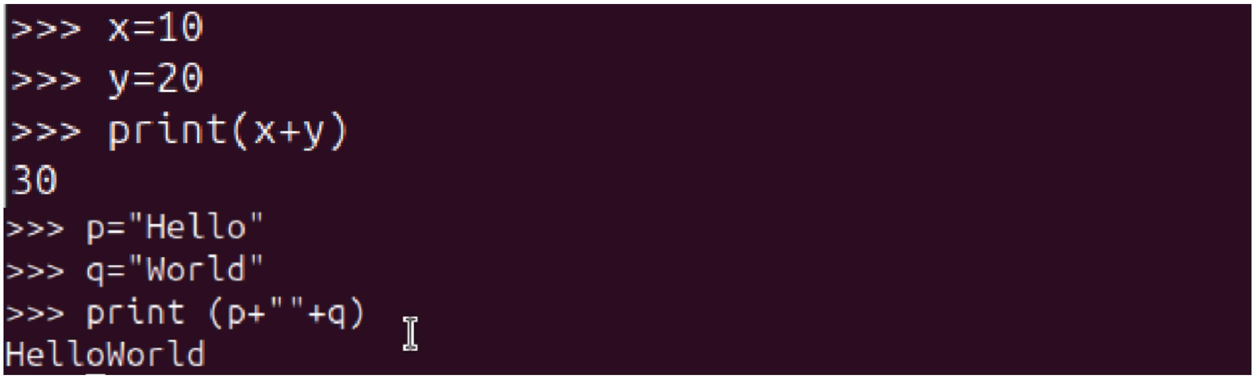


```
a, b, c = 5, 6, 7
print(a)
print(b)
print(c)
```

Plus and concatenation operation on the variables



```
x = 10
y = 20
print(x + y)

p = "Hello"
q = "World"
print(p + " " + q)
```



```
>>> x=10
>>> y=20
>>> print(x+y)
30
>>> p="Hello"
>>> q="World"
>>> print (p+""+q)
HelloWorld
```

2.4 Data Types

A data type defines the type of data, for example 123 is an integer data while "hello" is a String type of data. The data types in Python are divided in two categories:

1. Immutable data types – Values cannot be changed.

- Numbers
- String
- Tuple

2. Mutable data types – Values can be changed

- List
- Dictionaries
- Sets

1)Python Numbers

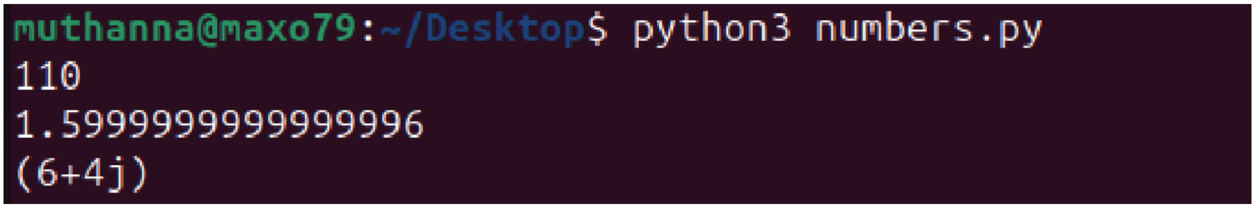
In this guide, we will see how to work with numbers in Python. Python supports integers, floats and complex numbers.

An integer is a number without decimal point for example 5, 6, 10 etc.

A float is a number with decimal point for example 6.7, 6.0, 10.99 etc.

A complex number has a real and imaginary part for example 7+8j, 8+11j etc.

```
# Python program to display numbers of
# different data types

# int
num1 = 10
num2 = 100
print(num1+num2)

# float
a = 10.5
b = 8.9
print(a-b)

# complex numbers
x = 3 + 4j
y = 9 + 8j
print(y-x)
```

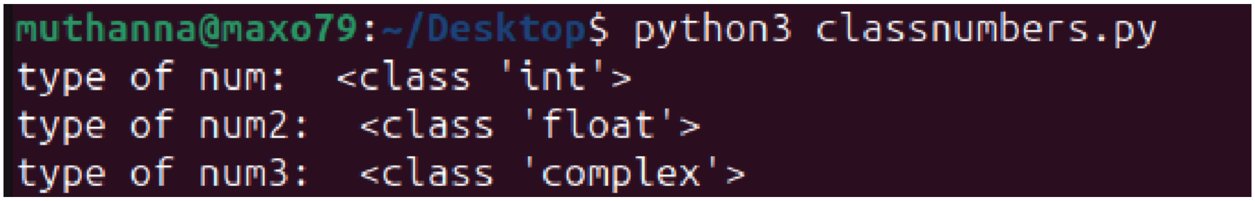


```
muthanna@maxo79:~/Desktop$ python3 numbers.py
110
1.5999999999999996
(6+4j)
```

Python example to find the class(data type) of a number

We can use the `type()` function to find out the class of a number. An integer number belongs to `int` class, a float number belongs to `float` class and a complex number belongs to `complex` class.

```
# program to find the class of a number

# int
num = 100
print("type of num: ",type(num))

# float
num2 = 10.99
print("type of num2: ",type(num2))

# complex numbers
num3 = 3 + 4j
print("type of num3: ",type(num3))
```

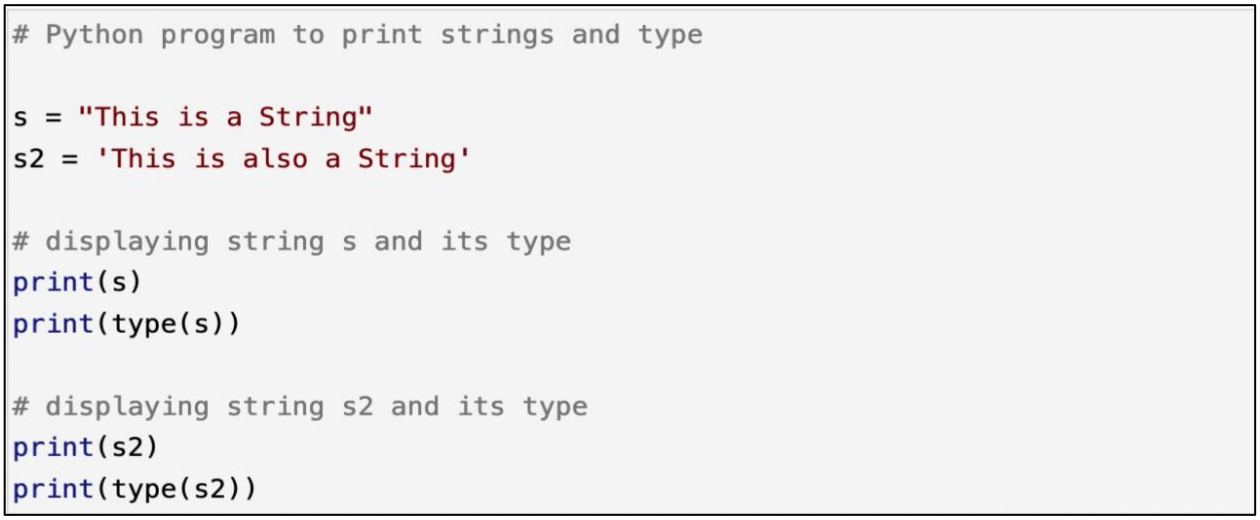


```
muthanna@maxo79:~/Desktop$ python3 classnumbers.py
type of num:  <class 'int'>
type of num2:  <class 'float'>
type of num3:  <class 'complex'>
```

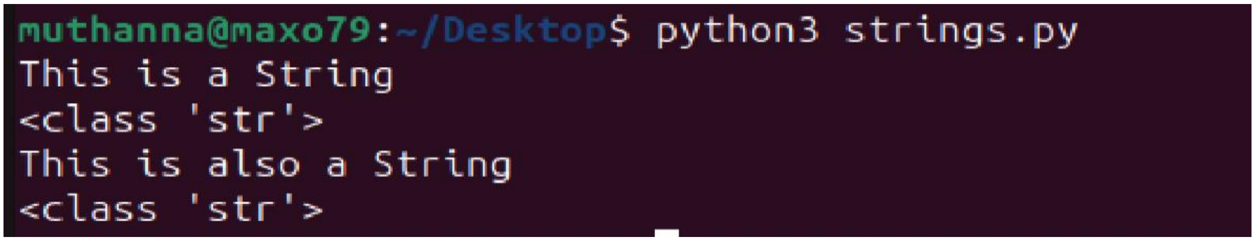
2. Python Strings

String is a sequence of characters in Python. The data type of String in Python is called “str”.

Strings in Python are either enclosed with single quotes or double quotes. In the following example we have demonstrated two strings one with the double quotes and other string s2 with the single quotes.



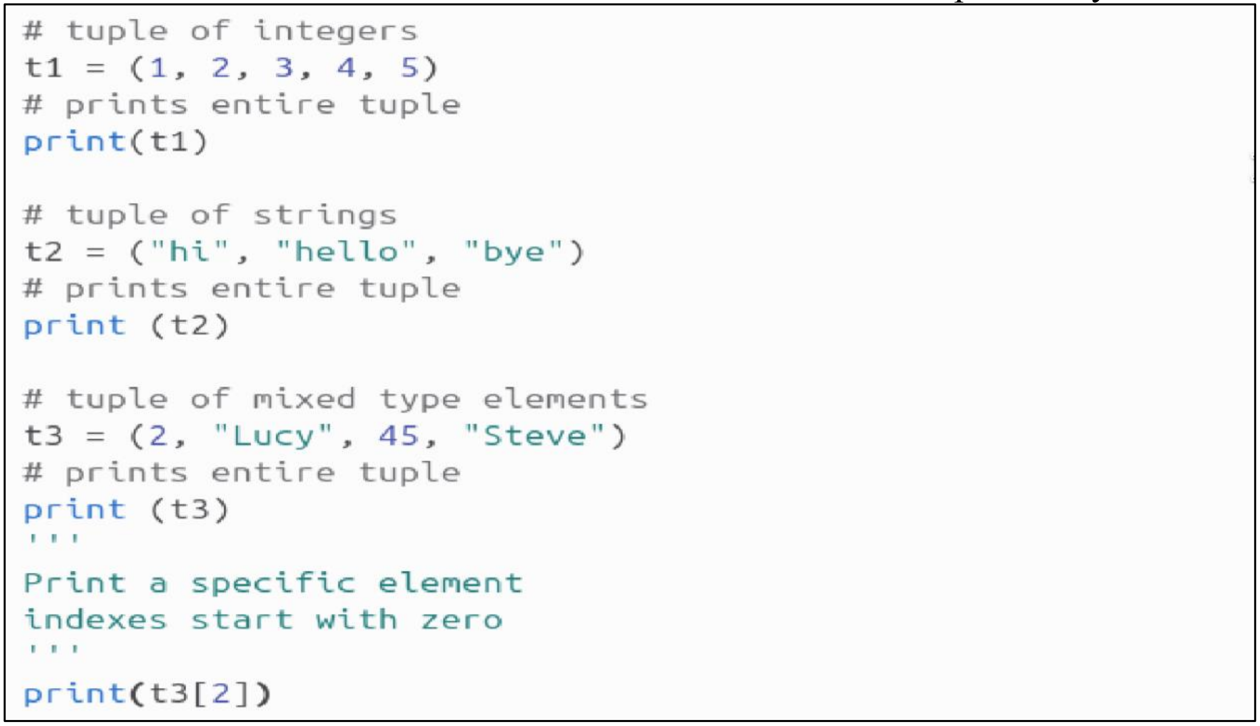
```
# Python program to print strings and type

s = "This is a String"
s2 = 'This is also a String'

# displaying string s and its type
print(s)
print(type(s))

# displaying string s2 and its type
print(s2)
print(type(s2))
```



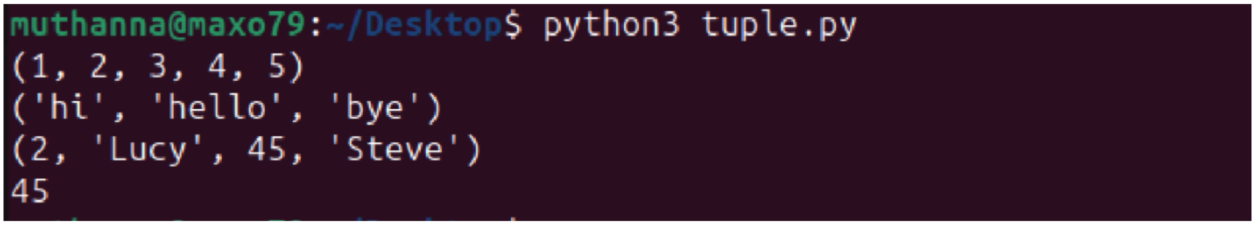
```
muthanna@maxo79:~/Desktop$ python3 strings.py
This is a String
<class 'str'>
This is also a String
<class 'str'>
```

3. Python Tuple

Tuple is immutable data type in Python which means it cannot be changed. It is an ordered collection of elements enclosed in round brackets and separated by commas.



```
# tuple of integers
t1 = (1, 2, 3, 4, 5)
# prints entire tuple
print(t1)

# tuple of strings
t2 = ("hi", "hello", "bye")
# prints entire tuple
print (t2)

# tuple of mixed type elements
t3 = (2, "Lucy", 45, "Steve")
# prints entire tuple
print (t3)
'''
Print a specific element
indexes start with zero
'''
print(t3[2])
```

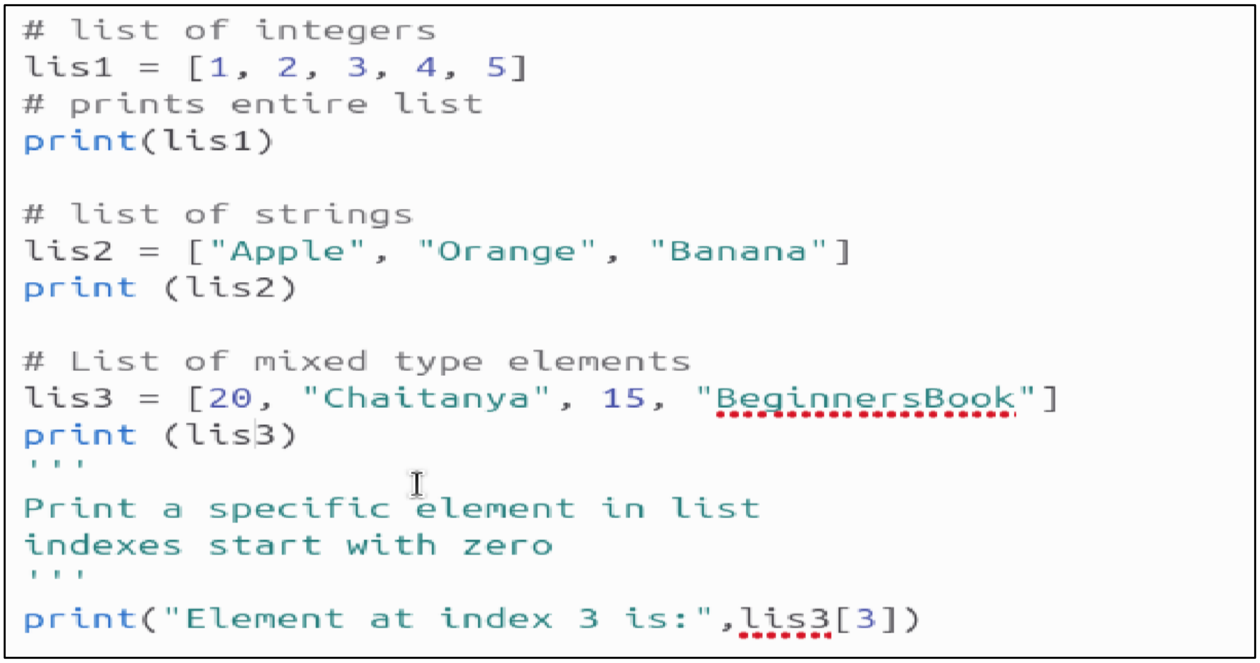


```
muthanna@maxo79:~/Desktop$ python3 tuple.py
(1, 2, 3, 4, 5)
('hi', 'hello', 'bye')
(2, 'Lucy', 45, 'Steve')
45
```

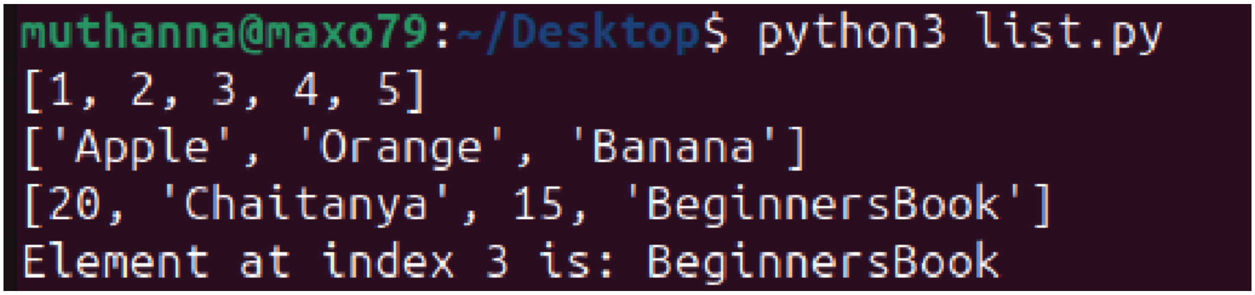
4. Python List

List is similar to tuple, it is also an ordered collection of elements, however list is a mutable data type which means it can be changed unlike tuple which is an immutable data type.

A list is enclosed with square brackets and elements are separated by commas.



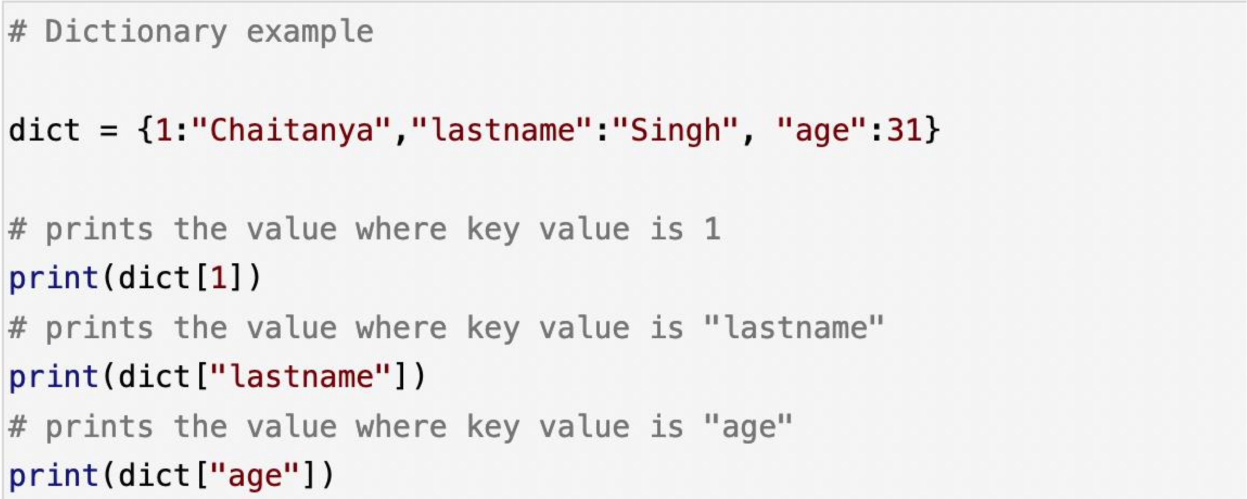
```
# list of integers
lis1 = [1, 2, 3, 4, 5]
# prints entire list
print(lis1)

# list of strings
lis2 = ["Apple", "Orange", "Banana"]
print (lis2)

# List of mixed type elements
lis3 = [20, "Chaitanya", 15, "BeginnersBook"]
print (lis3)
'''
Print a specific element in list
indexes start with zero
'''
print("Element at index 3 is:",lis3[3])
```



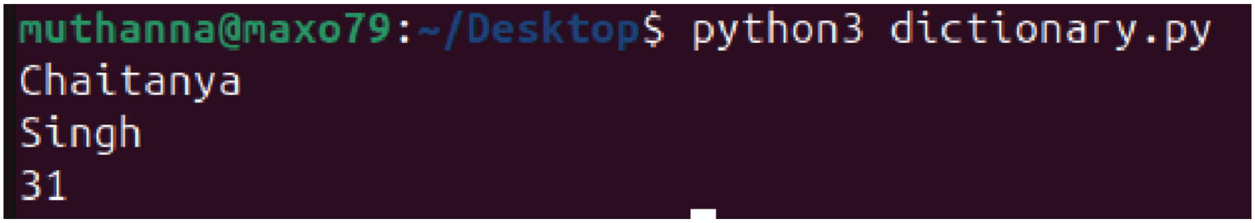
```
muthanna@maxo79:~/Desktop$ python3 list.py
[1, 2, 3, 4, 5]
['Apple', 'Orange', 'Banana']
[20, 'Chaitanya', 15, 'BeginnersBook']
Element at index 3 is: BeginnersBook
```

5. Python Dictionary

Dictionary is a collection of key and value pairs. A dictionary doesn’t allow duplicate keys but the values can be duplicate. It is an ordered, indexed and mutable collection of elements.
The keys in a dictionary doesn’t necessarily to be a single data type, as you can see in the following example that we have 1 integer key and two string keys.



```python
# Dictionary example

dict = {1:"Chaitanya","lastname":"Singh", "age":31}

# prints the value where key value is 1
print(dict[1])
# prints the value where key value is "lastname"
print(dict["lastname"])
# prints the value where key value is "age"
print(dict["age"])
```

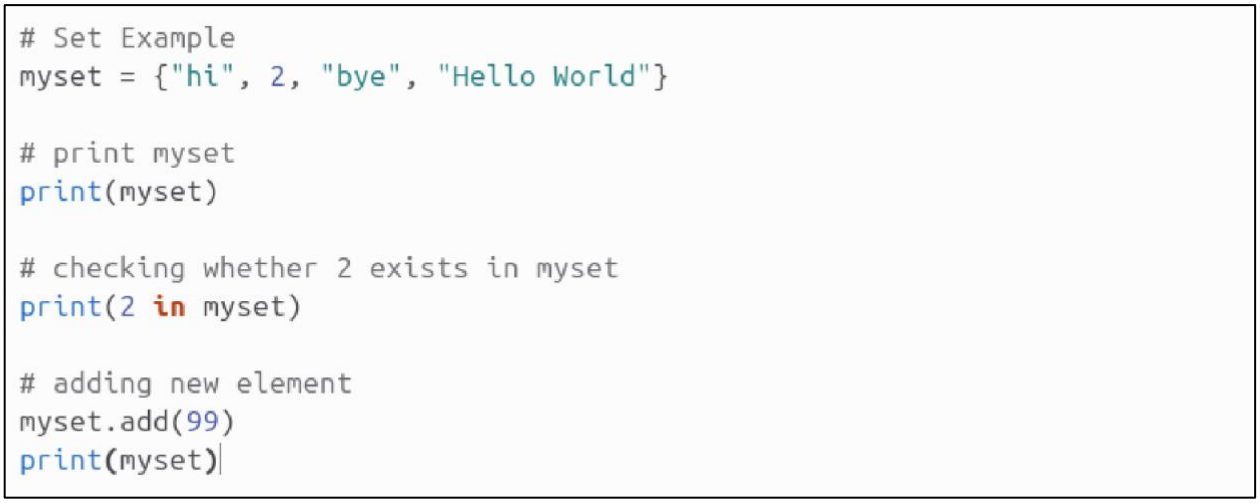


```
muthanna@maxo79:~/Desktop$ python3 dictionary.py
Chaitanya
Singh
31
```

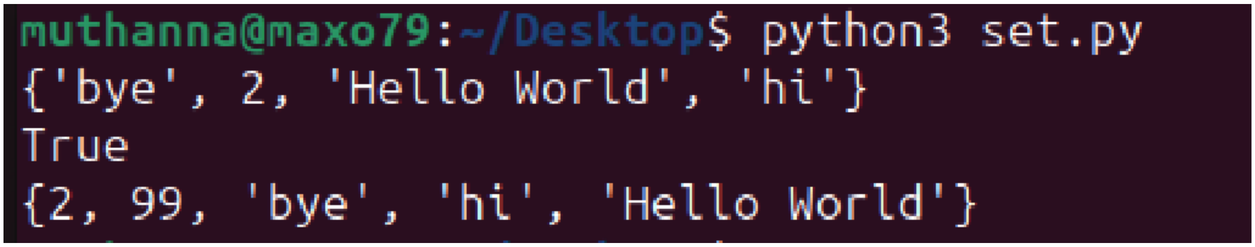
6. Python Set

A set is an unordered and unindexed collection of items. This means when we print the elements of a set they will appear in the random order and we cannot access the elements of set based on indexes because it is unindexed.

Elements of set are separated by commas and enclosed in curly braces. Lets take an example to understand the sets in Python.



```
# Set Example
myset = {"hi", 2, "bye", "Hello World"}

# print myset
print(myset)

# checking whether 2 exists in myset
print(2 in myset)

# adding new element
myset.add(99)
print(myset)
```



```
muthanna@maxo79:~/Desktop$ python3 set.py
{'bye', 2, 'Hello World', 'hi'}
True
{2, 99, 'bye', 'hi', 'Hello World'}
```