**Python Tuples**

**Introduction**

|  |
| --- |
| A tuple is an ordered sequence of elements of different data types, such as integer, float, string, list or even a tuple. Elements of a tuple are enclosed in parenthesis (round brackets) and are separated by commas. Like list and string, elements of a tuple can be accessed using index values, starting from 0. Tuples can also contain elements of different data types. |

**Tuple declaration**

Numbers=(1,2,3,4,5)  
print(Numbers)

**Output**

**(1, 2, 3, 4, 5)**

**Note:** If you want to store one element in the tuple for instance 35, its declared as Numbers=(35,). If declared without a comma, the numbers will be treated as variable storing value 35. As mentioned see a tuple of different data types below.

Characters=(1,2, **"Andy"**, 3,4, **"xyz"**, 5)  
print(Characters)

**Output**

(1, 2, 'Andy', 3, 4, 'xyz', 5)

You can access an element by use of an index as shown below:

Characters=(1,2, **"Andy"**, 3,4, **"xyz"**, 5)  
print(Characters[2])

**Output**

Andy

A tuple declaration can have a nested list within as an element as shown below.

Characters=(1,2, **"Andy"**, 3,4, **"xyz"**, 5, [6,7])  
print(Characters[7])

The result of the above code would be



The same output can be obtained using negative indexing. Eg print(Characters[-1])

Tuple is an **immutable data type**. It means that the elements of a tuple cannot be changed after it has been created. An attempt to do this would lead to an error.

Consider the following code below:

Characters=(1,2, **"Andy"**, 3,4, **"xyz"**, 5, [6,7])  
Characters[1]=77  
print(Characters)

The above code will result in the following error below:

**TypeError: 'tuple' object does not support item assignment**

**However, you can change as follows:**

Characters=(1,2, 3,4, 5, [6,7])  
Characters[5][1]=77  
print(Characters)

**Output**



Tuple operations

Concatenation

Tuples can be concatenated using + sign

Characters=(1,2, 3,4, 5, [6,7])  
Names=(**"Andrew"**, **"Mary"**)  
print(Characters+ Names)

**Output**



Do not confuse this with changing contents of the tuple.

**Repetition of elements.**

This is more or less similar to multiplication and Python uses \* symbol for replication

**Membership**

The in operator checks if the element is present in the tuple and returns True, else it returns False. The opposite is not in. Let us check if the value 99 exists in the Characters tuple.

Characters=(1,2, 3,4, 5, [6,7])  
Result= **"Timothy" in** Characters  
print(Result)

**Output**

False

**Slicing**

Like string and list, slicing can be applied to tuples also.

print(Characters[2:5])

The output will be:

(3, 4, 5)

**Common Built in functions and methods for tuples**

|  |  |  |
| --- | --- | --- |
| **Method** | **Description** | **Example** |
| Len() | Returns the length or the number of elements of the tuple passed as the argument | Len(Characters) |
| Tuple() | Creates an empty tuple if no argument is passed Creates a tuple if a sequence is passed as argument | Characters=(**"Pthyon"**) a=tuple(Characters) print(a)  Output  ('P', 't', 'h', 'y', 'o', 'n') |
| count | Returns the number of times the given element appears in the tuple | Characters=(**"Pthyon"**) a=tuple(Characters) print(a.count(**"P"**))  Result=1 |
| sorted | Returns a sorted list | Characters=(**"pthyon"**) a=tuple(Characters) print(sorted(a))  Output  ['h', 'n', 'o', 'p', 't', 'y'] |
| Min, Max, sum | Resturns the minimum value. Max- returns maximum value. Sum returns the sum of all elements |  |

**Exercise**

Write a program to input n numbers from the user. Store these numbers in a tuple. Print the maximum, sum and minimum number from this tuple. Add some values to the tuple.

Print the initial tuple and the resulting tuple