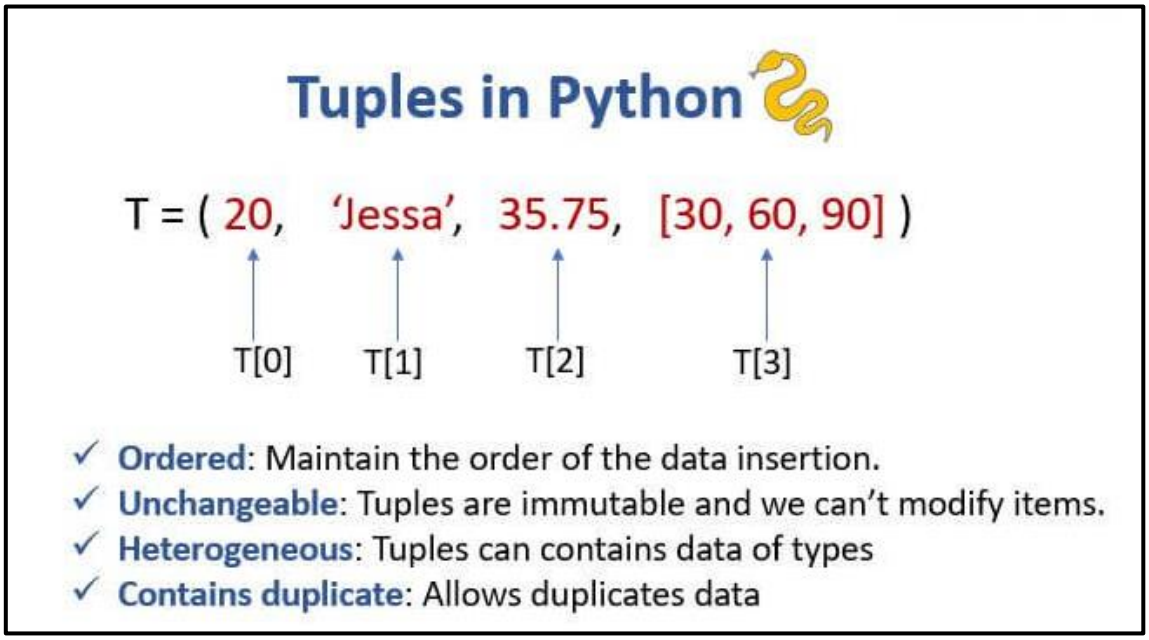
Table of Contents

Chapter 5

Python Collections (Arrays)
There are four collection data types in the Python programming language:
-List is a collection which is ordered and changeable. Allows duplicate members.
-Tuple is a collection which is ordered and unchangeable. Allows duplicate members.
-Set is a collection which is unordered, unchangeable*, and unindexed. No duplicate members.
-Dictionary is a collection which is ordered** and changeable. No duplicate members.

Tuple



Tuples are used to store multiple items in a single variable.
Tuple is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Set, and Dictionary, all with different qualities and usage.
A tuple is a collection which is ordered and unchangeable.
Tuples are written with round brackets.
Tuple items are ordered, unchangeable, and allow duplicate values.
When we say that tuples are ordered, it means that the items have a defined order, and that order will not change.
Tuple items are indexed, the first item has index [0], the second item has index [1] etc.
A tuple can contain different data types:
Ex.

```
fruit = ("apple", "banana", "cherry")
print(fruit) # ('apple', 'banana', 'cherry')
```

```
#Tuple with on Item
# Create Tuple With One Item
# To create a tuple with only one item, you have to add a comma after the item, otherwise Python will not recognize it as a tuple.
# One item tuple, remember the comma:
oneTpl = ("apple",)
print(type(oneTpl)) # <class 'tuple'>

#NOT a tuple
ntTpl = ("apple")
print(type(ntTpl)) # <class 'str'>
#Join Tuples:
# To join two or more tuples you can use the + operator:
# Join two tuples:

tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)
tuple3 = tuple1 + tuple2
print(tuple3) # ('a', 'b', 'c', 1, 2, 3)

# Multiply Tuples
# If you want to multiply the content of a tuple a given number of times, you can use the * operator:
# Multiply the fruits tuple by 2:

fruits = ("apple", "banana", "cherry")
mytuple = fruits * 2
print(mytuple) # ('apple', 'banana', 'cherry', 'apple', 'banana', 'cherry')

# Once a tuple is created, you cannot change its values. Tuples are unchangeable, or immutable as it also is called.
# But there is a workaround. You can convert the tuple into a list, change the list, and convert the list back into a tuple.

#Add Items
# *********Add Items*********
print("*********Add Items*********")
x = ("apple", "banana", "cherry")
```

```
y = list(x) # type cast into list
y[1] = "kiwi" # perform list operation
x = tuple(y) # type case back to tuple
print(x) # ("apple", "kiwi", "cherry")

a = ("apple", "banana", "cherry")
b = ("orange",)
a += b
print(a)
```

```
#Remove Items
# *********Remove Items*********
print("*********Remove Items*********")
# Tuples are unchangeable, so you cannot remove items from it, but you can use the same workaround as we used for changing and adding tuple items
tupleRem = ("apple", "banana", "cherry")
y = list(tupleRem)
y.remove("apple")
tupleRem = tuple(y) # ('banana', 'cherry')

# When we create a tuple, we normally assign values to it. This is called "packing" a tuple:
# Packing a tuple:
fruits = ("apple", "banana", "cherry")

# But, in Python, we are also allowed to extract the values back into variables. This is called "unpacking":
fruits = ("apple", "banana", "cherry")
(green, yellow, red) = fruits
print(green) # apple
print(yellow) # banana
print(red) # cherry

# If the number of variables is less than the number of values, you can add an * to the variable name and the values will be assigned to the variable as a list:
# Assign the rest of the values as a list called "red":

fruits = ("apple", "mango", "papaya", "pineapple", "cherry")

(green, *tropic, red) = fruits

print(green) # apple
print(tropic) # ['mango', 'papaya', 'pineapple']
print(red) # cherry
```