Tuples

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.

- **Unchangeable**
- Ordered
- Written with round brackets.
- Allow duplicate values.
- Can contain different data types

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.

In [1]:
```
# Creating a tuple:
thistuple = ("apple",)
print(type(thistuple))
print(thistuple)

# NOT a tuple
# thistuple = ("apple")
# print(type(thistuple))
```

```
<class 'tuple'>
('apple',)
```

In [2]:
```
mytuple = ("apple", "banana", "cherry")
print(mytuple)
```

```
('apple', 'banana', 'cherry')
```

In [3]:
```
tuple1 = ("abc", 34, True, 40, "male")
print(tuple1)
```

```
('abc', 34, True, 40, 'male')
```

In [4]:
```
# Creating a tuple using tuple() Constructor
mytuple = tuple(("apple", "banana", "cherry"))
# mylist = list(("apple", "banana", "cherry"))
```

In [5]:
```
print(len(mytuple))
print(type(mytuple))
```

```
3
<class 'tuple'>
```

Access Tuple Items

In [6]:
```
thistuple = ("apple", "banana", "cherry", "orange", "kiwi", "melon", "mango")
print(thistuple[1])
```

```
banana
```

In [7]:
```
print(thistuple[-1])
```

```
mango
```

In [8]:
```
print(thistuple[2:5])
print(type(thistuple[2:5]))
```

```
('cherry', 'orange', 'kiwi')
<class 'tuple'>
```

In [9]:
```
print(thistuple[:4])
```

```
('apple', 'banana', 'cherry', 'orange')
```

In [10]:
```
print(thistuple[-4:-1])
```

```
('orange', 'kiwi', 'melon')
```

In [11]:
```
# in operator to check if item is in tuple or not:

thistuple = ("apple", "banana", "cherry")
if "apple" in thistuple:
  print("Yes")
```

```
Yes
```

In [12]:
```
if "orange" not in thistuple:
    print("No")
else:
    print("Yes")
```

```
No
```

Change Tuple Values

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.**

In [13]:
```
# Add items

thistuple = ("apple", "banana", "cherry")
thistuple.append("orange")
print(thistuple)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-13-93dbc271cb6a> in <module>
      2 
      3 thistuple = ("apple", "banana", "cherry")
----> 4 thistuple.append("orange")
      5 print(thistuple)

AttributeError: 'tuple' object has no attribute 'append'
```

In [14]:
```python
# Add items
thistuple = ("apple", "banana", "cherry")
mylist = list(thistuple)
print(mylist)
mylist.append("orange")
print(mylist)
thistuple = tuple(mylist)
print(thistuple)
```

```
['apple', 'banana', 'cherry']
['apple', 'banana', 'cherry', 'orange']
('apple', 'banana', 'cherry', 'orange')
```

In [15]:
```python
# Remove items
thistuple = ("apple", "banana", "cherry")
mylist = list(thistuple)
print(mylist)
mylist.remove("cherry")
print(mylist)
thistuple = tuple(mylist)
print(thistuple)
print(type(thistuple))
```

```
['apple', 'banana', 'cherry']
['apple', 'banana']
('apple', 'banana')
<class 'tuple'>
```

In [16]:
```python
# Delete tuple
thistuple = ("apple", "banana", "cherry")
del thistuple
```

Packing and Unpacking a tuple

When we create a tuple, we normally assign values to it. This is called "packing" a tuple. In Python, we are also allowed to extract the values back into variables. This is called "unpacking".

Note: The number of variables must match the number of values in the tuple, if not, you must use an asterix to collect the remaining values as a list.

In [17]:
```
# Packing
fruits = ("apple", "banana", "cherry")
```

In [18]:
```
# Unpacking

fruits = ("apple", "banana", "cherry")
(green, yellow, red) = fruits
print(green)
print(yellow)
print(red)
```

```
apple
banana
cherry
```

In [19]:
```
# Unpacking using asterisk(*): If the number of variables is less than the number of values, you
# an * to the variable name and the values will be assigned to the variable as a list

fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")
(green, yellow, *red) = fruits

print(green)
print(yellow)
print(red)
```

```
apple
banana
['cherry', 'strawberry', 'raspberry']
```

In [20]:
```
# If the asterix is added to another variable name than the last, Python will assign values to t
# variable until the number of values left matches the number of variables left.

fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")
(green, *yellow, red) = fruits

print(green)
print(yellow)
print(red)
```

```
apple
['banana', 'cherry', 'strawberry']
raspberry
```

Loop through a tuple

In [21]:
```
thistuple = ("apple", "banana", "cherry")
for x in thistuple:
    print(x)
```

```
apple
banana
cherry
```

In [22]:
```
thistuple = ("apple", "banana", "cherry")
for i in range(len(thistuple)):
    print(thistuple[i])
```

```
apple
banana
cherry
```

In [23]:
```
thistuple = ("apple", "banana", "cherry")
i = 0
while i < len(thistuple):
    print(thistuple[i])
    i = i + 1
```

```
apple
banana
cherry
```

enumerate(): A lot of times when dealing with iterators, we also get a need to keep a count of iterations. Python eases the programmers’ task by providing a built-in function enumerate() for this task. Enumerate() method adds a counter to an iterable and returns it in a form of enumerate object. This enumerate object can then be used directly in for loops or be converted into a list of tuples using list() method.

Signature: enumerate(iterable, start = 0)

In [24]:
```
l1 = ["eat","sleep","repeat"]
s1 = "geek"

# creating enumerate objects
obj1 = enumerate(l1)
obj2 = enumerate(s1)

print (type(obj1))
print (list(enumerate(l1)))

# changing start index to 2 from 0
print (list(enumerate(s1,2)))
```

```
<class 'enumerate'>
[(0, 'eat'), (1, 'sleep'), (2, 'repeat')]
[(2, 'g'), (3, 'e'), (4, 'e'), (5, 'k')]
```

In [25]:
```python
for x,y in enumerate(thistuple):
    print(x,y)
```
```
0 apple
1 banana
2 cherry
```

In [26]:
```python
for x,y in enumerate(thistuple, start = 5):
    print(x,y)
```
```
5 apple
6 banana
7 cherry
```

In [27]:
```python
for x,y in enumerate(thistuple, 5):
    print(x,y)
```
```
5 apple
6 banana
7 cherry
```

In [28]:
```python
for x in enumerate(l1):
    print(x)
```
```
(0, 'eat')
(1, 'sleep')
(2, 'repeat')
```

Join Tuples

In [29]:
```
# l1 = [1, 2, 3]
# l2 = ["hello", 4, 5]
# l1 = l1 + l2
# print(l1)

tuple1 = ("a", "b" , "c")
tuple2 = (1, 2, 3)

tuple3 = tuple1 + tuple2
print(tuple3)
```

```
('a', 'b', 'c', 1, 2, 3)
```

In [30]:
```
# fruits = ["apple", "banana", "cherry"]
# fruits = fruits*2
# print(fruits)

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

```
('apple', 'banana', 'cherry', 'apple', 'banana', 'cherry')
```

Tuple Methods

In [31]:
```
# count(): Returns the number of times a specified value occurs in a tuple

thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)
x = thistuple.count(5)
print(x)
```

```
2
```

In [32]:
```
# index(): finds the first occurrence of the specified value.

thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)
x = thistuple.index(8)
print(x)
```

```
3
```

In [ ]:
```

```