Applying all functions in Tuples, List, Dictionaries & Sets

---

1. Tuple ()

In [1]:
```
tuple1 =(23,56.8, 23, 23, True, 'Runner Up', True)
tuple1
```

In [2]:
```
tuple2= (34.67, False, 9,9,9,9, 'Gone')
tuple2
```

count() returns the number of times element appears in the tuple.

In [3]:
```
tuple1.count(23)
```

Out[3]: 3

index() method returns the index of the specified element in the tuple.

In [4]:
```
tuple2.index('Gone')
```

Out[4]: 6

---

2. List []

In [5]:
```
list1 =[23,56.8, 23, 23, True, 'Runner Up', True]
list1
```

Out[5]: [23, 56.8, 23, 23, True, 'Runner Up', True]

In [6]:
```
list2= [34.67, False, 9,9,9,9, 'Gone']
list2
```

Out[6]: [34.67, False, 9, 9, 9, 9, 'Gone']

append() method appends an element to the end of the list.

In [7]:
```
list1.append(45)
list1
```

Out[7]: [23, 56.8, 23, 23, True, 'Runner Up', True, 45]

clear() method removes all items from the list

In [8]:
```
list1.clear()
list1
```

Out[8]: []

copy() method returns a shallow copy of the list.

In [9]:
```
list1=list2.copy()
list1
```

Out[9]: [34.67, False, 9, 9, 9, 9, 'Gone']

count() method returns the number of times the specified element appears in the list.

In [10]:
```
list1.count(9)
```

Out[10]: 4

extend() method adds all the elements of an iterable (list, tuple, string etc.) to the end of the list.

In [11]:
```
list3=[34,58,6,79,68]
list1.extend(list3)
list1
```

Out[11]: [34.67, False, 9, 9, 9, 9, 'Gone', 34, 58, 6, 79, 68]

index() method returns the index of the specified element in the list.

In [12]:
```
list1.index('Gone')
```

Out[12]: 6

insert() method inserts an element to the list at the specified index.

In [13]:
```
list1.insert(5, 'Stadium')
list1
```

Out[13]: [34.67, False, 9, 9, 9, 'Stadium', 9, 'Gone', 34, 58, 6, 79, 68]

pop() method removes the item at the given index from the list and returns the removed item.

In [14]:
```
list1.pop()
list1
```

Out[14]: [34.67, False, 9, 9, 9, 'Stadium', 9, 'Gone', 34, 58, 6, 79]

In [15]:
```
list1.pop(3)
list1
```

Out[15]: [34.67, False, 9, 9, 'Stadium', 9, 'Gone', 34, 58, 6, 79]

remove() method removes the first matching element (which is passed as an argument) from the list.

In [16]:
```
list1.remove(9)
```

```
list1
```

Out[16]: [34.67, False, 9, 'Stadium', 9, 'Gone', 34, 58, 6, 79]

reverse() method reverses the elements of the list.

In [17]:
```
list1.reverse()
list1
```

Out[17]: [79, 6, 58, 34, 'Gone', 9, 'Stadium', 9, False, 34.67]

sort() method sorts the elements of a given list in a specific ascending or descending order.

In [18]:
```
list3.sort()
list3
```

Out[18]: [6, 34, 58, 68, 79]

---

3. Dictionary { key : value}

In [19]:
```
dict1 = {'Ball':50, 'Bat':500,'Wicket':200}
dict1
```

Out[19]: {'Ball': 50, 'Bat': 500, 'Wicket': 200}

In [20]:
```
dict2= {'Pen':10,'Marker':20,'Pencil':5,'Sharpner':5}
dict2
```

Out[20]: {'Pen': 10, 'Marker': 20, 'Pencil': 5, 'Sharpner': 5}

clear() method removes all items from the dictionary.

In [21]:
```
dict1.clear()
dict1
```

Out[21]: {}

copy() method returns a copy (shallow copy) of the dictionary.

In [22]:
```
dict1= dict2.copy()
dict1
```

Out[22]: {'Pen': 10, 'Marker': 20, 'Pencil': 5, 'Sharpner': 5}

fromkeys() method returns a new dictionary with the given sequence of elements as the keys of the dictionary.

In [23]:
```
fruit= {'Apple', 'Banana','Grape','Gauva'}
price={50}

dict3=dict.fromkeys(fruit,price)
dict3
```

Out[23]: {'Gauva': {50}, 'Apple': {50}, 'Banana': {50}, 'Grape': {50}}

get() method returns the value for the specified key if the key is in the dictionary.

In [24]:
```
dict2.get('Pen')
```

Out[24]: 10

items() method returns a view object that displays a list of dictionary's (key, value) tuple pairs.

In [25]:
```
dict2.items()
```

Out[25]: dict_items([('Pen', 10), ('Marker', 20), ('Pencil', 5), ('Sharpner', 5)])

keys() method returns a view object that displays a list of all the keys in the dictionary

In [26]:
```
dict3.keys()
```

Out[26]: dict_keys(['Gauva', 'Apple', 'Banana', 'Grape'])

values() method returns a view object that displays a list of all the values in the dictionary.

In [27]:
```
dict3.values()
```

Out[27]: dict_values([{50}, {50}, {50}, {50}])

pop() method removes and returns an element from a dictionary having the given key.

In [28]:
```
dict3.pop('Apple')
```

Out[28]: {50}

In [29]:
```
dict3
```

Out[29]: {'Gauva': {50}, 'Banana': {50}, 'Grape': {50}}

popitem() method removes and returns the last element (key, value) pair inserted into the dictionary.

In [30]:
```
dict3.popitem()
```

Out[30]: ('Grape', {50})

In [31]:
```
dict3
```

Out[31]: {'Gauva': {50}, 'Banana': {50}}

setdefault() method returns the value of a key. If not, it inserts key with a value to the dictionary.

```
```

In [32]:
```
# Mango not in the dict3
dict3.setdefault('Mango')
dict3
```

Out[32]:
```
{'Gauva': {50}, 'Banana': {50}, 'Mango': None}
```

In [33]:
```
# Grape in dict3
dict3.setdefault('Grape')
```

In [34]:
```
dict3
```

Out[34]:
```
{'Gauva': {50}, 'Banana': {50}, 'Mango': None, 'Grape': None}
```

update() method updates the dictionary with the elements from another dictionary object

or from an iterable of key/value pairs.

In [35]:
```
dict4={'orange':90}

dict3.update(dict4)
dict3
```

Out[35]:
```
{'Gauva': {50}, 'Banana': {50}, 'Mango': None, 'Grape': None, 'orange': 90}
```

---

4. Sets {}

In [36]:
```
set1= {2,56,4.5, 'Salt','Sugar', True}
set2= {2,56,567,'Apple','Banana', True}
```

In [37]:
```
#Making a blank set
set3=set()
set3
```

Out[37]:
```
set()
```

In [38]:
```
set4= set([3,4,67,'Green'])

set4
```

Out[38]:
```
{3, 4, 67, 'Green'}
```

add() method adds a given element to a set. If the element is already present, it doesn't add any element.

In [39]:
```
set3.add(12)
set3
```

Out[39]:
```
{12}
```

clear() method removes all elements from the set.

In [40]:
```
set3.clear()
```

```
In [ ]: set3
```

```
Out[40]: set()
```

remove() method removes the specified element from the set.

```
In [41]: set4.remove(67)
         set4
```

```
Out[41]: {3, 4, 'Green'}
```

copy() method returns a shallow copy of the set.

```
In [42]: set3= set1.copy()
         set3
```

```
Out[42]: {2, 4.5, 56, 'Salt', 'Sugar', True}
```

difference() method returns the set difference of two sets.

```
In [43]: print (set2.difference(set1))
         set2
```

```
{'Apple', 'Banana', 567}
Out[43]: {2, 56, 567, 'Apple', 'Banana', True}
```

difference_update() updates the set calling difference_update() method with the difference of sets.

```
In [44]: print (set2.difference_update(set1))

         set2
```

```
None
Out[44]: {567, 'Apple', 'Banana'}
```

discard() method removes a specified element from the set (if present).

```
In [45]: set3.discard('Salt')

         set3
```

```
Out[45]: {2, 4.5, 56, 'Sugar', True}
```

intersection() method returns a new set with elements that are common to all sets.

```
In [46]: print (set3.intersection(set1))
         set3
```

```
{True, 2, 4.5, 'Sugar', 56}
Out[46]: {2, 4.5, 56, 'Sugar', True}
```

intersection_update() method allows an arbitrary number of arguments (sets).

```
In [47]: print (set3.intersection_update(set1))
```

```
set3
```

```
None
```

Out[47]: {2, 4.5, 56, 'Sugar', True}

isdisjoint() method returns True if two sets are disjoint sets. If not, it returns False.

In [48]:
```
print (set3.isdisjoint(set1))
set3
```

```
False
```

Out[48]: {2, 4.5, 56, 'Sugar', True}

issubset() method returns True if all elements of a set are present in another set (passed as an argument).

If not, it returns False.

In [49]:
```
print(set3.issubset(set1))
set3
```

```
True
```

Out[49]: {2, 4.5, 56, 'Sugar', True}

issuperset() method returns True if a set has every elements of another set (passed as an argument).

If not, it returns False.

In [50]:
```
print(set3.issuperset(set1))
set3
```

```
False
```

Out[50]: {2, 4.5, 56, 'Sugar', True}

pop() method removes an arbitrary element from the set and returns the element removed.

In [51]:
```
set3.pop()
set3
```

Out[51]: {2, 4.5, 56, 'Sugar'}

remove() method removes the specified element from the set.

In [52]:
```
set3.remove('Sugar')
set3
```

Out[52]: {2, 4.5, 56}

Symmetric_difference() method returns the symmetric difference of two sets.

In [53]:
```
print(set3.symmetric_difference(set1))
set3
```

```
{True, 'Salt', 'Sugar'}
```

Out[53]: {2, 4.5, 56}

symmetric_difference_update() method finds the symmetric difference of two sets and updates the set calling it.

In [54]:
```
print(set3.symmetric_difference_update(set1))
set3
```

```
None
```

Out[54]: {'Salt', 'Sugar', True}

union() methord combines two sets

In [55]:
```
print (set3.union(set4))
print(set3)
print(set4)
```

```
{True, 'Salt', 3, 4, 'Sugar', 'Green'}
{True, 'Salt', 'Sugar'}
{'Green', 3, 4}
```

update() methord bring update/change in set

In [56]:
```
print(set3.update(set4))
print(set3)
print(set4)
```

```
None
{True, 'Salt', 3, 4, 'Sugar', 'Green'}
{'Green', 3, 4}
```