**LIST**

* Mutable.
* Can have different data types
* Most used Data Structure in Python – heap, linked list, queue, etc.
* **Methods**

- **list.append()**: Adds the value at the end of the list

-**list.insert(index,value):** inserts a <value> at the given <index>

-**list.pop(index):** removes and return the value at a given index

-**list.remove(value):** removes the first occurrence of a given <value>

-**list.count(value):** counts the no. of occurrence of the given value

-**list.index(value):** finds the first occurrence of the value and returns it’s index

Else, an error.

-**list.sort():** sorts the list value in ascending order

- **list.sort(reverse=TRUE):** this will sort the list Descending order.

- **list.reverse():** This will reverse the string. (Index n to index 0)

**TUPLE**

* Immutable
* Denoted by ()
* Similar to List (Just immutable)
* **Methods:**

**- tuple.count(value):** This is similar to the list optn, this will count the occurrence of <value>

**-tuple.index(value):** This will return the index of the <value>

- tu.sort() – the in-place sort will not work on tuples as tuples are immutable.

**DICTIONARY**

* Key-Value pair data type
* **Nesting is available**
* Similar to JSON.
* Are orderless[**unlike list() & tuple()**] – value can be placed in any order, can access data using the key so what’s the need of order.
* **Methods:**

**- dict.keys():** returns a list called dict\_keys([<KEYS>])

**TYPE is <class 'dict\_keys'>**

**-key = list(myDict.keys()):** This will type cast the dict\_keys(obvio!), and will return just the list.

**-dict.values():** This will return <class dict\_value> type list.

**-dict.items():** This will return the <class dict\_items> type list. The values inside the list is of type **tuple()**

**-dict.get(key): This will return None** if the the key is not present in the dict, else the v alue.

myDict = {

    "name": "Sahil Singh",

    "age": 23,

    "About": ["Dev", {"Exp": 3}, [22, 3, 1998]]

}

print(myDict.keys())

>> dict\_keys(['name', 'age', 'About', 'Loves to Play'])

dili = myDict.keys()

print(f"this is dili= {dili}")

>> this is dili= dict\_keys(['name', 'age', 'About', 'Loves to Play'])

print(f"Type of dili= {type(dili)}")

>> Type of dili= <class 'dict\_keys'>

print(type(myDict.keys()))

# Important

key = list(myDict.keys())

print(f"Type of key is {type(key)}")

>> **Type of key is <class 'list'>**

print(key)

**>> ['name', 'age', 'About', 'Loves to Play']**

#Items in dict

items = myDict.items()

print(f"This is dict.items(): {items}")

**>> This is dict.items(): dict\_items([('name', 'Sahil Singh'), ('age', 23), ('About', ['Dev', {'Exp': 3}, [22, 3, 1998]]), ('Loves to Play', 'Fifa')])**

print(f"Type of dict.items(): {type(items)}")

**>> Type of dict.items(): <class 'dict\_items'>**

items = list(myDict.items())

print(f"Type of items of dict.item(): {type(items[0])}")

**>> Type of items of dict.item(): <class 'tuple'>**

#The values inside the dict\_item list are tuples, (key,values) tuples

**SET**

* Mutable data type which **removes duplicate values, will keep only one occurrence of a given value.**
* **Unordered- Sets are unordered in the sense that they will return values in any random order.**
* **Supports multiple datatypes in a single set(similar to list & tuples)**
* **Sets are unindexed**
* **Methods:**

- **set.add(value):** This will add <value> in a given set.

- **Union and Intersection: you can use any iterable**

# UNION

newSet = s.union({7, 13, 15})

print(f"UNION is {newSet}")

# >> {(11, 23), 1, 4, 6, 7, 13, 15}

# INTERSECTION

inter = newSet.intersection({1, 5, 7, 10})

print(f"INTERSECTION of newSet and other set is: {inter}")

# >> INTERSECTION of newSet and other set is: {1, 7}

s = {1, 4, 6}

# printing any value on a given index

print(s[0])

**>> TypeError: 'set' object is not subscriptable**

# You cannot access set values the way access <list> values.

**\*\*Important**

**(i) You cannot add list() in the set, because list() are unhashable and “Mutable”**

s.add([7, 8, 9])

**>> s.add([7, 8, 9])**

**TypeError: unhashable type: 'list'**

**(ii) You can add tuple() in set as they are Immutable.**

# Can add tuples in the set as they tuple() are Immutable

s.add((11, 23))

print(s)

**>> {(11, 23), 1, 4, 6, 'sahil'}**

**(iii) Dict in set()?- No**

# Can we add dict() in set? No we cannot as dict() are unhashable and mutable

s.add({"Role": "Python Dev"})

print(s)

**>> s.add({"Role": "Python Dev"})**

**TypeError: unhashable type: 'dict'**

**(iv) Set in set()? - No**

# Can we add set() in a set? No we cannot as set() are unhashable and mutable

s.add({11, 56, 70})

print(s)

**>> s.add({11, 56, 70})**

**TypeError: unhashable type: 'set'**