**Data Structure:** In computer science, a data structure is a data organization, management, and storage format that enables efficient access and modification.

Data structures are two types.

1. Primitive

* Integer numbers.
* Floating-point numbers
* Fixed-point numbers.
* Booleans.
* Characters and strings.
* Numeric data type ranges.

1. Non-Primitive

* Array
* List
* Tuple
* Set
* Dictionary
* File

Here, We will talk about Non-Primitive data types mainly.

**List**

A list is an ordered and mutable Python container. To create a list, the elements are placed inside square brackets “ [ ] ”, separated by commas. Index starts from 0!

**EG:** list =[“A”, ”B”, 3, 5, 6]

print(list(2)) = 3 ; print(list(0)) = A

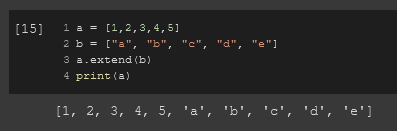
• To sort a list, use “.sort()” method.

list.sort() => list =[3, 5, 6, “A”, ”B” “P”]

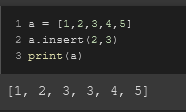
• To insert an element in a list, list.append(“P”)

list =[“A”, ”B”, 3, 5, 6, “P”]

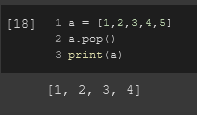
• To add one list with another list, We use list.extend(list\_2)



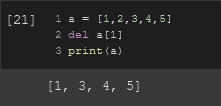
• To insert an element at a specific indexed place of a list, We use list.insert(new\_item,index\_number)



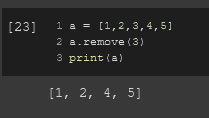
• If we want to delete an element from the last point of a list, We can use pop method.



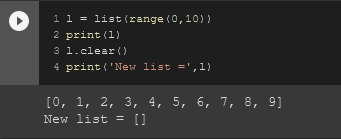
• If we want to delete an element from the last point of a list following index, We use del



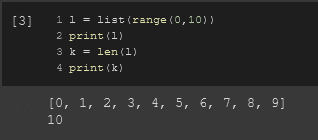
• If we want to delete an element from the last point of a list element wise, We use remove



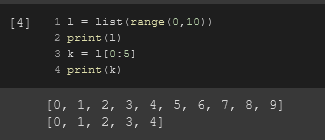
• to clear a list, We use l.clear() [l = list name]



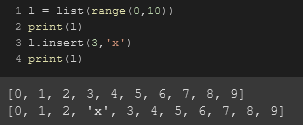
• to calculate the length of the list, we use len(list)



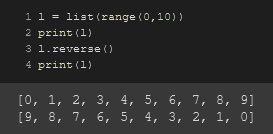
• to slice a list, we use list\_name(start\_index : finishing\_index)



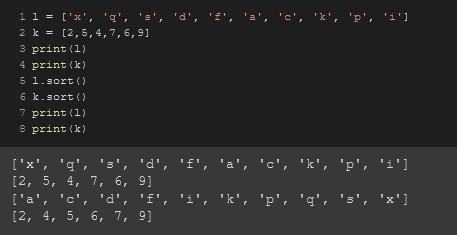
• to insert an element in a specific position or index, we use list\_name(index, “element”)



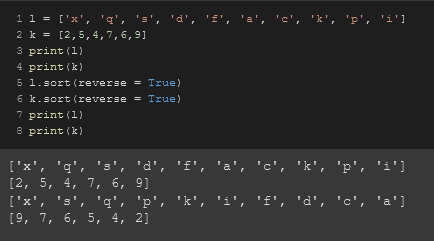
• to reverse a list, we use list\_name.reverse()



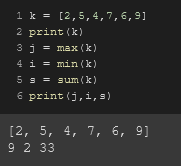
• to sort a list, we use list\_name.sort()



• to sort a list in reverse order, we use list\_name.sort(reverse = True)



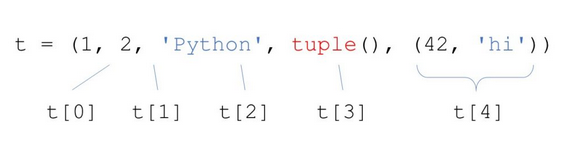
• to find the maximum value, minimum value and sum of all the elements in a list, we respectively use max(list\_name), min(list\_name), sum(list\_name)



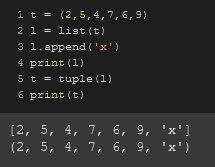
**Tuple**

Python tuples are a data structure that store an ordered sequence of values. Tuples are immutable. This means you cannot change the values in a tuple. Tuples are defined with parenthesis. Tuples are a core data structure in Python.

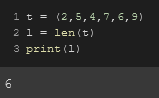
Tuples placed inside round brackets “ ( ) ”, separated by commas. Index starts from 0!



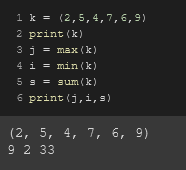
• Tuples are immutable, to insert a new element in a tuple, convert it into a list first and then add the element in the list and convert it back to tuple again.



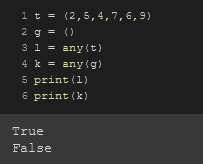
• to calculate the length of a tuple, we use len(tuple)



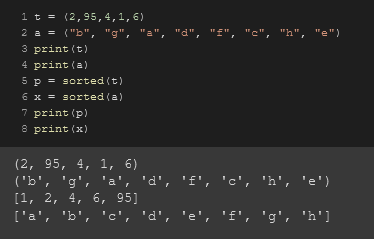
• to find the maximum value, minimum value and sum of all the elements in a tuple, we respectively use max(tuple\_name), min(tuple\_name), sum(tuple\_name)



• any() function tells if a tuple is filled or empty. Prints “True” & “False” for respectively filled and empty tuples.



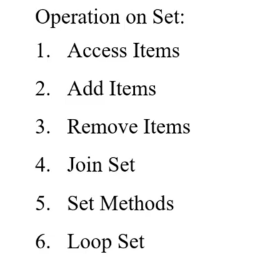
• to sort a tuple, we use sorted(tuple)



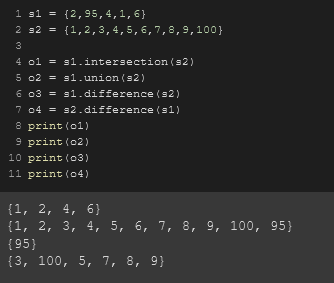
**SET**

Set does not support duplicate elements. Set placed inside curly brackets “ { } ”, separated by commas. Index starts from 0!

These are operations we can do using set.



In between 2 sets, We can do intersection, union and difference operation using “s1.intersection(s2)”, “s1.union(s2)”, “s1.difference(s2)” respectively.



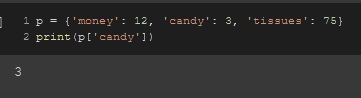
**Dictionary**

Dictionary is basically set with keys along with values. Same as set, dictionary placed inside curly brackets “ { } ”, separated by commas. Index starts from 0!

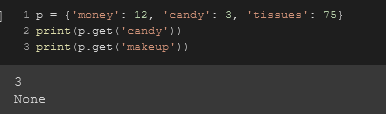
A dictionary has a key and a corresponding value which are stored in {key1:value1, key2:value2} format

****

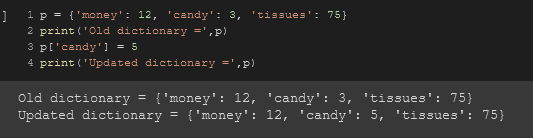
• To recall a value from the dictionary, We use the key name to call out the value.



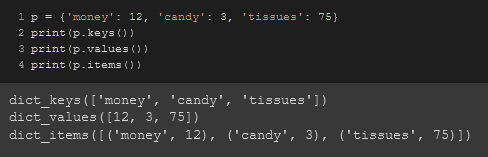
• To see if a key exists in a dictionary, we can use the “get()” function.



• To keywise update a dictionary, we use dictionary\_name[‘key’] = new\_value

****

• To see all the keys, values & key-values pairs of a dictionary, we print keys(), values(), items functions respectively

.****

purse = dict()

purse['money'] = 12

purse['candy'] = 3

purse['tissues'] = 75

print(purse)

**Output:-**

****

**Code:-**

purse = dict()

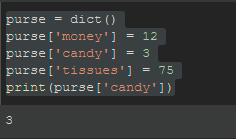
purse['money'] = 12

purse['candy'] = 3

purse['tissues'] = 75

print(purse['candy'])

**Output:-**

****

**Code:-**

purse = dict()

purse['money'] = 12

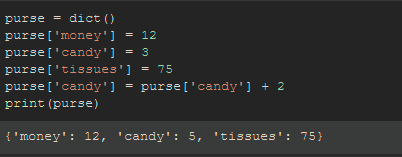
purse['candy'] = 3

purse['tissues'] = 75

purse['candy'] = purse['candy'] + 2

print(purse)

**Output:-**

****

**(Using Dictionaries)**

**Code:-**

c = dict()

n = ['csev', 'cwen', 'csev', 'zquian', 'cwen']

for i in n:

if i not in c:

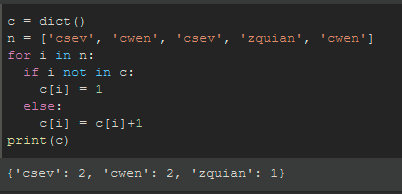
c[i] = 1

else:

c[i] = c[i]+1

print(c)

**Output:-**

****

**#4 (Clearing a dictionary)**

Let there be an already occupied dictionary which needs to be cleared, Using the **CLEAR()** method, We can empty a previously dictionary list and use it again.

**Code:-**

# Python program to demonstrate working of

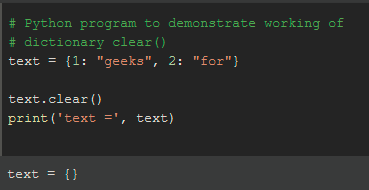
# dictionary clear()

text = {1: "geeks", 2: "for"}

text.clear()

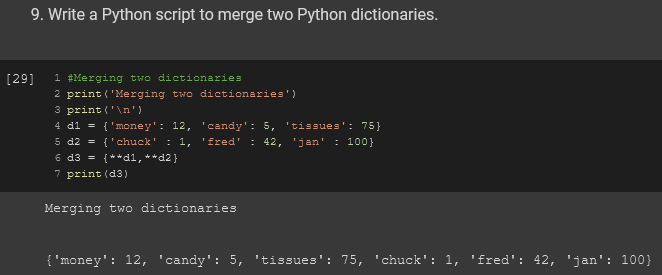
print('text =', text)

**Output:-**

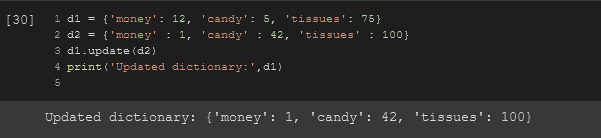


**#5 (Merging two dictionaries)**

**Code & Output:-**



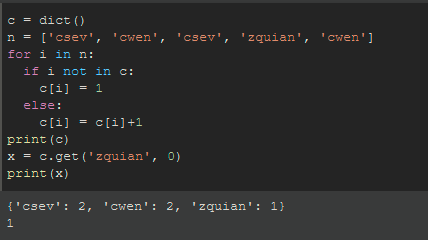
**#6 (Updating a dictionaries)**

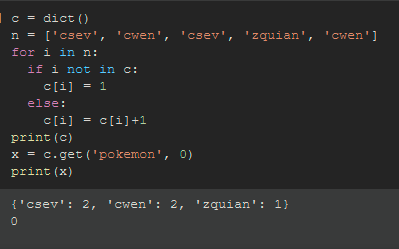
****

**#7 (Using get method)**

The **GET method** basically simplifies count. Two examples with explanations are given below.

**Code:-**

****

****

**Explanation:-**

Here in the line **“x = c.get(‘zquian’, 0)”** the **get()** method states that go to the **c** dictionary, Use the key **‘zquian’** to look up the dictionary **c** how many times is that there. And the **0** is set as the default value if the answer if the given **KEYWORD** is not there like for the second case where the keyword **‘pokemon’** is given.

It is showing **0** as the x’s answer because there s no keyword named **pokemon** in the dictionary **c**.

**#6 (Counting pattern)**

**Code:-**

c = dict()

i = input('Enter a line of text: ')

w = i.split()

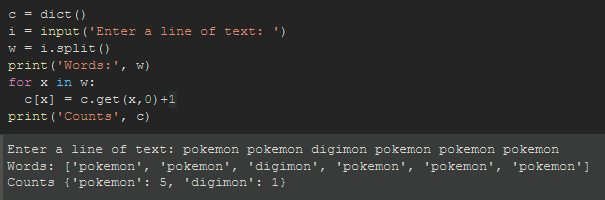
print('Words:', w)

for x in w:

c[x] = c.get(x,0)+1

print('Counts', c)

**Output:-**

****

**#7 (Looking up a dictionary using a dictionary)**

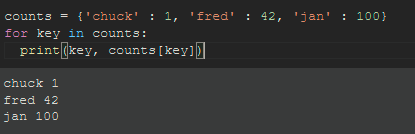
**Code:-**

counts = {'chuck' : 1, 'fred' : 42, 'jan' : 100}

for key in counts:

print(key, counts[key])

**Output:-**

****

**#8 (Retrieving lists of keys and values)**

**Code:-**

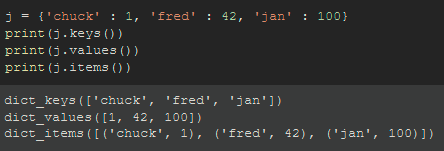
j = {'chuck' : 1, 'fred' : 42, 'jan' : 100}

print(j.keys())

print(j.values())

print(j.items())

**Output:-**

****

**#9 (Two iteration variables)**

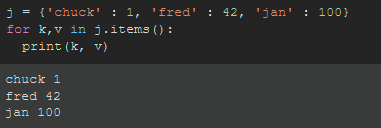
**Code:-**

j = {'chuck' : 1, 'fred' : 42, 'jan' : 100}

for k,v in j.items():

print(k, v)

**Output:-**

****

**Explanation:-**

Here, the two iteration variables K & V goes for respectively KEY & VALUES pairs.

**#10 (Using two iteration variables in dictionary)**

**Code:-**

f = input('Enter File: ')

o = open(f)

count = dict()

for i in o:

words = i.split()

for word in words:

count[word] = count.get(word,0)+1

bigcount = None

bigword = None

for word, count in count.items():

if bigcount is None or count > bigcount:

bigcount = count

bigword = word

print(bigword, bigcount)

**Output:-**

