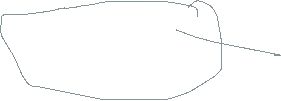
Functions

def add\_item(list\_x=[]):

list\_x += [3] Here list will be created only once after calling.



print(list\_x)

add\_item()

add\_item([1,2])

add\_item()

* Non-default arguments cannot follow default arguments.
  + def greet(arg\_2, arg\_1="Hi"):

print(arg\_1 + " " + arg\_2)

greeting = input()

name = input()

greet(arg\_2=name)

* Built in functions
  + len()
  + round(3.12212, 1) 🡪 3.1
  + min()
    - min([1,-2,4,2])
    - min(“python”, “Java”) 🡪 here p - 80 & j – 74 🡺 “Java”
  + max()
  + sum([])
  + sorted([])
  + sorted([list], reverse= True)
* Recursion
  + In recursion there will be base case where the condition matches and terminates.

# **SET**

* setOperations
  1. add (will add individual item into set only 1 arg)
  2. update(x,y,z or[list] takes multiple arg but arg should be iterable)
  3. copy( it clones the set ex:- s = {“sd”, “d”} 🡺 s1 = s.copy())
  4. pop(removes random element in set)
  5. remove( removes specified element if sepecified element is not present then it will gives the error)
  6. discard(removes the specified element if el is not present then it will not gives the error)
  7. clear(it will remove all the elements from the set)
* Mathematical operations on sets
  1. Union **(|)** (returns all the elements present in both sets)
  2. Intersection **(&)** (returns the common elements present in both sets)



* 1. Difference **(-)** (x-y returns the elements present in x but not in y (common removed))



* 1. Symmetric\_difference **(^)** (returns the uncommon elements present in both the elements)



* Set comprehension
  1. S = {x\*x for x in range(1,6)}
  2. Print(s)
  3. O/P 🡪 {1, 4, 6, 9, 16, 25}
* Set comparisi=ons
  1. issubset
  2. issuperset
  3. isdisjoint

# **Tuples**

* immutable
* tuple ex:-
  1. tup = () 🡪 empty tuple
  2. tup = 1,2,3,4 🡪 here it is packed and will give the result as a tuple.
  3. tup = (1,) 🡪keep comma for single tuple else it will convert to datatype.
  4. tup, = (1,) 🡪 here it is unpacked and will as an int.

# **Dictionaries**

* To create empty dictionary
  + dict\_a = {} / dict\_a = dict()
* Example of dictionary

dict\_a = {

‘name’ : ‘Rajesh’

‘age’ : 23

}

* In dictionaries keys must be immutable and unique and values can be any data type and can repeat.
* Accessing items
  + Accessing items with key and [] 🡺 This will return error if the key is not found.
    - Print(dict\_a[‘key’])
  + Accessing items with get() 🡺 This will return none if keyerror is not found.
    - Print(dict\_a.get(‘city’)) 🡪 returns none because there is no key as city.
    - Print(dict\_a.get(‘name’)) 🡪 returns value.
* Membership check
  + Result = ‘name’ in dict\_a 🡺 returns true if key exists else returns false.

## **Operations on dictionaries**

🡪Adding a key value pair.

🡪Modifying existing items.

🡪Deleting existing items.

* Adding a key value pair.

dict\_a = {'name': 'Teja','age': 15 }

dict\_a['city'] = 'Goa'

print(dict\_a) 🡺 {'name': 'Teja', 'age': 15, 'city’: ‘Goa’}

* Modifying an Existing Item
  + As dictionaries are mutable, we can modify the values of the keys.

dict\_a = {

'name': 'Teja',

'age': 15

}

dict\_a['age'] = 24

print(dict\_a) 🡪{'name': 'Teja', 'age': 24}

* Deleting an Existing Item
  + We can also use the del keyword to remove individual items or the entire dictionary itself.

dict\_a = {

'name': 'Teja',

'age': 15

}

del dict\_a['age']

print(dict\_a) 🡪 {'name': 'Teja'}

## **Dictionary Views**

They provide a dynamic view on the dictionary’s entries, which means that when the dictionary changes, the view reflects these changes.

* Dictionary Methods
* dict.keys() 🡪 print(dict\_a.keys())
  + returns dictionary Keys
* dict.values() 🡪 print(dict\_a.values())
  + returns dictionary Values
* dict.items() 🡪 print(dict\_a.items())
* returns dictionary items(key-value) pairs

The objects returned by keys () , values() & items() are View Objects .

#### Iterate over Dictionary Views

* Examples
  + for key in dict\_a.keys():

print(key) 🡪 name /n age

* + keys\_list = list(dict\_a.keys())

print(keys\_list) 🡪 ['name', 'age’]

* + for value in dict\_a.values():

print(value) 🡪 Raj /n 15

* + for key, value in dict\_a.items():

pair = "{} {}".format(key,value)

print(pair) 🡪 name Teja /n age 15

##### Dictionary View Objects

keys() , values() & items() are called Dictionary Views as they provide a dynamic view on the dictionary’s items.

view = dict\_a.keys()

print(view) 🡪 dict\_keys(['name', 'age'])

dict\_a['roll\_no'] = 10

print(view) 🡪 dict\_keys(['name', 'age', 'roll\_no’)

**OOPS**