Python Programming

Tuples

- A tuple is a standard data type of Python that can store a sequence of values belonging to any type.
  - (1,2,'a','b',3.4)
- Tuples are immutable- Elements of a tuple cannot be changed in place
- Empty Tuple
  - t=()
  - l=tuple()
- Nested Tuple
  - t=(3,4,(5,6),7)

Accessing values in a tuple

- The first element in a tuple will be at index 0
- You can use a negative number to index backwards
- The last element in the tuple will be at index -1
- Common method used to input tuple is eval(input())
- Tuples can be created from existing sequences.
  - t=tuple("hello") will return the tuple ('h','e','l','l','o')

Tuple Operations

- Joining tuples + operator
- Replicating tuples * operator
- Slicing a tuple seq=T[start:stop:step]
- Membership operators – IN and Not IN

- Creating a tuple from a set of values is called packing and creating individual values from a tuple's elements is called unpacking
- <v1>,<v2>,<v3>,….=t where the number of variables on the left must match the number of elements in the tuple
- eg: t=(1,2,'a','b')

  w,x,y,z=t

- Deleting an element of a tuple is not possible using del statement as tuples are immutable
- An entire tuple can be deleted using

  del <tuple_name>

Dictionaries

- Dictionaries are mutable, unordered collections with elements in the form of key:value pairs that associate keys to values
- To create a dictionary:
- <dictionary>={key1:value1,key2:value2….}
- dict1={}
- The keys of a dictionary must be of immutable types such as strings, numbers, tuple etc

Accessing elements

- <dictionaryname>[<key>]
- Attempting to access a key that doesn't exist , causes an error.
- To traverse a dictionary

```
for<k> in <dictionaryname>
    print<dictionaryname[k]>
```

<dictionary>.keys() will return all the keys as a sequence

<dictionary>.values() will return all the values as a sequence

Characteristics of a Dictionary

- Unordered Set, Not a sequence
- Indexed by Keys
- Keys must be unique
- Dictionaries are mutable

Creating a dictionary

- Initializing a Dictionary
  - Employee={'name':'John','salary':10000,'age':24}
- Adding key:value pairs to an empty dictionary
  - Employee={} or Employee=dict()
  - <dictionary>[<key>]=<value>
- Creating a dictionary from name and value pairs
  - Employee=dict(name='John',salary=10000,age=24)
  - Employee=dict({'name':'John','salary':10000,'age':24})
  - Employee=dict(zip(('name','salary','age'),('John',10000,24)))

- To add elements to a dictionary

  <dictionary>[<key>]=<value> where

  <key >should be a new unique value

- To update an existing value

  <dictionary>[<key>]=<value> where

  <key >should be an existing one

- To delete a dictionary element:
  - del<dictionary>[<key>]
  - The key should exist otherwise Python raises Exception
  - del<dictionary>

- To check the existence of a key in dictionary we can use in or not in operator.
  - <key> in <dictionary>
  - <key> not in <dictionary>
- To check whether a value is there in the dictionary we can use <value> in <dictionary>.values()

Dictionary functions

- len(<dictionary>) returns the count of key-value pairs in the dictionary
- <dictionary>.clear() removes all the elements from the dictionary
- del <dictionaryname> removes the dictionary and elements in it
- <dictionary>.get(<key>[,default]) will return the value associated with the key. If the key is not present and default is given that will be printed otherwise the error will be returned

- <dictionary>.items() returns all the items of the dictionary as a sequence of (key,value) tuples
- eg:
  employee={“name”:”John”,”salary”:10000,”age”:24
  }
- seq=employee.items()
- for x in seq:

```
    print (x)
 for k,v in mylist:
    print(k,v)
```

- fromkeys() method is used to create a new dictionary from a sequence containing all the keys and a common value which will be assigned to all the keys.
  - dict.fromkeys(<key sequence>,[<value>])
    - <key sequence > is a Python sequence containing the keys for the new dictionary
    - <value> is the common value that is assigned to all the keys; if skipped, value None is assigned to all the keys.

- setdefault() method inserts a new key:value pair only if the key doesn't already exist.
  - dict.setdefault(<key>,[<value>])
  - <key> and <value> are the key and value to be added to the dictionary. If <value> is not passed as input, the default None value is used as the value

- update method merges key:value pairs from the new dictionary into the original dictionary adding or replacing as needed
- <dictionary>.update(<otherdictionary>)

- pop() method removes and returns the dictionary element associated to passed key.
  - <dict>.pop(<key>,[<value>])
  - <dict> is the dictionary in which a key is to be deleted
  - <key> is the key to be deleted. If the key is found, it is deleted and its value is returned
  - <value> is the return value / message which will be returned by the method , if the given key is not found in the dictionary. If <value> is skipped and given <key> is not found, Python will raise error

- popitem() method removes and returns a <key,value> pair from the dictionary
  - <dict>.popitem()
  - It returns the items which was the last item entered in the dictionary.
  - It returns the deleted <key,value> pair in the form of a tuple.
  - If the dictionary is empty, calling popitem() raises a KeyError

- sorted(<dict>,[reverse=False])
  - <dict> is the dictionary whose keys are to be sorted
  - reverse argument is optional and when set to True, it returns the keys of the dictionary sorted in descending order.
  - returns a list containing the sorted keys of the dictionary

Thank you