Experiment No.: 1

Title: Implementation of tuples, lists & dictionaries using python.

Objectives: 1. To learn tuples, lists & dictionaries

Theory:

Tuples, lists, and dictionaries:

Lists are what they seem - a list of values. Each one of them is numbered, starting from zero - the first one is numbered zero, the second 1, the third 2, etc. You can remove values from the list, and add new values to the end.

Example: cats' names.

```
cats = ['Tom', 'Snappy', 'Kitty', 'Jessie', 'Chester']
```

Lists are extremely similar to tuples. Lists are modifiable (or 'mutable', as a programmer may say), so their values can be changed. Most of the time we use lists, not tuples, because we want to easily change the values of things if we need to.

Creating lists:	

list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5];
list3 = ["a", "b", "c", "d"];
	# list1, list2,list3 created

Accessing/Updating/Slicing Values in Lists	
print ("list1[0]: ", list1[0]) print ("list2[1:]: ", list2[1:]) list2[2] = 6	# prints **list1[0]: 'physics'** # prints sliced list2 **list2[1:]: [2,3,4,5]** # updates value at index 2 as **6**
Deleting list elements:	
del list[2]	# deletes element at index 2
Basic List Operations:	

len([1, 2, 3])
 	
[1, 2, 3] + [4, 5, 6]
 	
['Hi!'] * 4
 	
3 in [1, 2, 3]
 	
for x in [1,2,3] :
 	
    print (x,end = ' ')
	# Length
	
	# Concatenation
	
	# Repetition
	
	# Membership
	
	# Iteration

Built-in List Functions and Methods:	

Sr.No.	**Function & Description**
1. **cmp(list1, list2)**	No longer available in Python 3.
2. **len(list)**	Gives the total length of the list.
3. **max(list)**	Returns item from the list with max value.
4. **min(list)**	Returns item from the list with min value.
5. **list(seq)**	Converts a tuple into list.
Sr.No.	**Methods & Description**
1. **list.append(obj)**	Appends object obj to list
2. **list.count(obj)**	Returns count of how many times obj occurs in list
3. **list.extend(seq)**	Appends the contents of seq to list
4. **list.index(obj)**	Returns the lowest index in list that obj appears
5. **list.insert(index, obj)**	Inserts object obj into list at offset index
6. **list.pop(obj = list[-1])**	Removes and returns last object or obj from list
7. **list.remove(obj)**	Removes object obj from list
8. **list.reverse()**	Reverses objects of list in place
9. **list.sort([func])**	Sorts objects of list, use compare func if given

Tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The main difference between the tuples and the lists is that the tuples cannot be changed unlike lists. Tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally, you can put these comma-separated values between parentheses also.

Creating Tuples:	

tuple1 = ('physics', 'chemistry', 1997, 2000)
tuple2 = (1, 2, 3, 4, 5)
tuple3 = "a", "b", "c", "d"
tuple4 = ();
tuple5 = (50,)
	# 5 tuples are created
	
	# tuple4 is an **empty** tuple
	#tuple5 : single value **50 ", is must after it"**

Accessing/Slicing Values in Tuples:	
print ("tuple1[0]: ", tuple1[0]) print ("tuple2[1:]: ", tuple2[1:]) tuple2[2] = 6 # Invalid	# prints **tuple1[0]: 'physics'** # prints sliced tuple2 # Invalid since tuples are immutable.
Deleting Tuple elements:	

del tuple1	# deletes tuple1
Basic Tuple Operations:	
len((1, 2, 3)) (1, 2, 3) + (4, 5, 6) ('Hi!',) * 4 3 in (1, 2, 3) for x in (1,2,3) : print (x, end = ' ')	# Length 3 # Concatenation (1, 2, 3, 4, 5, 6) # Repetition ('Hi!', 'Hi!', 'Hi!', 'Hi!') # Membership True # Iteration 1 2 3
Built-in Tuple Functions and Methods:	
Sr.No.	**Function & Description**
1. **cmp(tuple1, tuple2)**	Compares elements of both tuples.
2. **len(tuple)**	Gives the total length of the tuple.
3. **max(tuple)**	Returns item from the tuple with max value.
4. **min(tuple)**	Returns item from the tuple with min value.
5. **tuple(seq)**	Converts a list into tuple.

Dictionaries are similar to what their name suggests - a dictionary. In a dictionary, you have an 'index' of words, and for each of them a definition. In python, the word is called a 'key', and the definition a 'value'. The values in a dictionary aren't numbered - tare similar to what their name suggests - a dictionary. In a dictionary, you have an 'index' of words, and for each of them a definition. In python, the word is called a 'key', and the definition a 'value'. The values in a dictionary aren't numbered - they aren't in any specific order, either - the key does the same thing. You can add, remove, and modify the values in dictionaries. Example: telephone book.

Creating Dictionaries:	
dict ={'Name':'Zara', 'Age':7, 'Name': 'Manni'} print ("dict['Name']: ", dict['Name'])	# Dictionary dict created
Accessing/Updating/Slicing Values in Dictionaries	
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'} print ("dict['Name']: ", dict['Name']) dict['Age'] = 8	# prints **dict['Name']: Zara** # updates **dict['Age']: 8**
Deleting Dictionary elements:	

del dict['Name'] dict.clear() del dict	# remove entry with key 'Name' # remove all entries in dict # delete entire dictionary
Keys must be immutable. This means you can use strings, numbers or tuples as dictionary keys but something like **['key']** is not allowed.	example − dict = {['Name']: 'Zara', 'Age': 7} print ("dict['Name']: ", dict['Name']) **Error: list objects are unhashable**
More than one entry per key is not allowed. This means no duplicate key is allowed. When duplicate keys are encountered during assignment, the last assignment wins. For example − dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'} print ("dict['Name']: ", dict['Name'])	# produces dict['Name']: Manni

Built-in List Functions and Methods:

Sr.No.	Function & Description
len(dict)	Gives the total length of the dictionary. This would be equal to the number of items in the dictionary.
str(dict)	Produces a printable string representation of a dictionary
type(variable)	Returns the type of the passed variable. If passed variable is dictionary, then it would return a dictionary type.

Sr.No.	Methods & Description
dict.clear()	Removes all elements of dictionary *dict*
dict.copy()	Returns a shallow copy of dictionary *dict*
dict.fromkeys()	Create a new dictionary with keys from seq and values *set* to *value*.
dict.get(key, default=None)	For *key* key, returns value or default if key not in dictionary
dict.has_key(key)	Removed, use the *in* operation instead.
dict.items()	Returns a list of *dict*'s (key, value) tuple pairs
dict.keys()	Returns list of dictionary dict's keys
dict.setdefault(key, default = None)	Similar to get(), but will set dict[key] = default if *key*is not already in dict
dict.update(dict2)	Adds dictionary *dict2*'s key-values pairs to *dict*
dict.values()	Returns list of dictionary *dict*'s values