Batch: C2-2	Roll No.: 047
Experiment / assignment / tutorial No.	
Grade: AA / AB / BB / BC / CC / CD /DD	
Signature of the Staff In-charge with date	

TITLE: Basic Data Structure in Python
AIM: Use suitable methods to get output for a given input.
OUTCOME: Students will be able to use basic data structures in Python.

Resource Needed: Python IDE

Theory:

Python Collections (Arrays)

There are four collection data types in the Python programming language:

- **A list** is a collection that is ordered and changeable. Allows duplicate members.
- Tuple is an unchangeable collection that has been ordered. Allows duplicate members.
- A set is a collection that is unordered and unindexed. There are no duplicate members.
- A dictionary is a collection that is unordered and changeable. There are no duplicate members.

When choosing a collection type, it is useful to understand the properties of that type. Choosing the right type for a particular data set could mean retention of meaning, and it could mean an increase in efficiency or security.

List: Lists are used to store multiple items in a single variable. Lists are created using square brackets. e.g. mylist = ["apple", "banana", "cherry"]

List Methods

Python has a set of built-in methods that you can use on lists. L:list, e:element, i:index

Method	Description
L.append(e)	Adds an element at the end of the list
L.clear()	Removes all the elements from the list
L.copy()	Returns a copy of the list
L.count(e)	Returns the number of elements with the specified value
L.extend(L2)	Add the elements of a list (or any iterable), to the end of the current list
L.index(e)	Returns the index of the first element with the specified value
L.insert(i,e)	Adds an element at the specified position
L.pop(i)	Removes the element at the specified position
L.remove(e)	Removes the item with the specified value
L.reverse()	Reverses the order of the list

L.sort()	Sorts the list

Tuple

Tuples are used to store multiple items in a single variable. A tuple is a collection that is ordered and **unchangeable**. Tuples are written with round brackets.
e.g. mytuple = ("apple", "banana", "cherry")

Tuple Methods

Python has two built-in methods that you can use on tuples. T:tuple, e:element

Method	**Description**
T.count(e)	Returns the number of times a specified value occurs in a tuple
T.index(e)	Searches the tuple for a specified value and returns the position of where it was found

Set

Sets are used to store multiple items in a single variable. A set is a collection which is both ***unordered*** and ***unindexed***. Sets are written with curly brackets.
e.g. myset = {"apple", "banana", "cherry"}

Set Methods

Python has a set of built-in methods that you can use on sets.

Method	**Description**
S.add(e)	Adds an element to the set
S.clear()	Removes all the elements from the set
S.copy()	Returns a copy of the set
S1.difference(S2)	Returns a set containing the difference between two or more sets
S1.difference_update(S2)	Removes the items in this set that are also included in another, specified set
S1.discard(e)	Remove the specified item
S1.intersection(S2)	Returns a set, that is the intersection of two other sets
S1.intersection_update(S2)	Removes the items in this set that are not present in other, specified set(s)
S1.isdisjoint(S2)	Returns whether two sets have a intersection or not
S1.issubset(S2)	Returns whether another set contains this set or not
S1.issuperset(S2)	Returns whether this set contains another set or not
S.pop()	Removes an element from the set
S.remove(e)	Removes the specified element
S1.symmetric_difference(S2)	Returns a set with the symmetric differences of two sets
S1.symmetric_difference_update(S2)	inserts the symmetric differences from this set and another

S1.union(S2)	Return a set containing the union of sets
S1.update(L1)	Update the set with the union of this set and others

Dictionary

Dictionaries are used to store data values in key:value pairs. A dictionary is a collection which is **ordered (3.7 version onward)**, **changeable** and **does not allow duplicates**. Dictionaries are written with curly brackets, and have keys and values.
e.g. thisdict = {"brand": "Ford", "model": "Mustang", "year": 1964}

Dictionary Methods

Python has a set of built-in methods that you can use in dictionary.

Method	Description
D.clear()	Removes all the elements from the dictionary
D.copy()	Returns a copy of the dictionary
D.get(k)	Returns the value of the specified key
D.items()	Returns a list containing a tuple for each key value pair.
D.keys()	Returns a list containing the dictionary's keys.
D.pop(k)	Removes the element with the specified key
D.popitem()	Removes the last inserted key-value pair
D.setdefault(k,v)	Returns the value of the specified key. If the key does not exist: insert the key, with the specified value.
D.update({k:v})	Updates the dictionary with the specified key-value pairs
D.values()	Returns a list of all the values in the dictionary.

Problem Definition:

1. In the below table, the input variable, Python code, and output column are given. You have to complete a blank cell in every row.

List		
Input	**Python Code**	**Output**
thislist=["apple","banana","cherry","orange","kiwi","melon","mango"]	print(len(thislist)) print(type(thislist)) print(thislist[1]) print(thislist[-1]) print(thislist[2:5]) print(thislist[:4]) print(thislist[2:])	7 <class 'list'> banana mango ['cherry', 'orange', 'kiwi'] ['apple', 'banana', 'cherry', 'orange'] ['cherry', 'orange', 'kiwi', 'melon', 'mango']

thislist = ["orange", "mango", "kiwi", "pineapple", "apple"]	if "apple" in thislist:
	  print("Yes, 'apple' is in the fruits list")
	for x in thislist:
	  print(x)
		Yes, 'apple' is in the fruits list
		orange
		mango
		kiwi
		pineapple
		apple

	for i in range(len(thislist)): print(thislist[i]) thislist.sort() print(thislist)	orange mango kiwi pineapple apple ['apple', 'kiwi', 'mango', 'orange', 'pineapple']
thislist=["apple","banana","cherry"]	i = thislist.index("banana") thislist[i]="blackcurrant" print(thislist)	['apple','blackcurrant','cherry']
thislist=["apple", "banana", "cherry"]	thislist.insert(2,"watermelon") print(thislist)	['apple','banana','watermelon', 'cherry']
thislist=["apple","banana","cherry"]	thislist.append("orange") print(thislist)	['apple', 'banana', 'cherry', 'orange']
thislist=["apple", "banana", "cherry"] tropical=["mango", "pineapple"]	thislist.extend(tropical) print(thislist)	['apple', 'banana', 'cherry', 'mango', 'pineapple']
thislist = ["apple", "banana", "cherry"]	thislist.pop(1) print(thislist)	['apple', 'cherry']
thislist = ["apple", "banana", "cherry"]	del thislist print(thislist)	print(thislist) ^^^^^^^^ NameError: name 'thislist' is not defined
thislist = ["apple", "banana", "cherry"]	thislist.clear() print(thislist)	[]
thislist = ["apple", "banana", "cherry"]	x=thislist y= thislist.copy() thislist.clear() print(x) print(y)	[] ['apple', 'banana', 'cherry']
list1 = [5, 6, 7] list2 = [1, 2, 3]	list3 = list1 + list2 print(list3)	[5, 6, 7, 1, 2, 3]

Tuple		
Input	**Python Code**	**Output**
x = ("apple",) y = ("apple")	print(type(x)) print(type(y))	<class 'tuple'> <class 'str'>
thistuple=("apple","banana","cherry")	print(thistuple[-1])	cherry
x = ("apple", "banana", "cherry")	x[1] = "kiwi" print(x)	x[1] = "kiwi" ^^^^ peError: 'tuple' object does not suppo
x = ("apple", "banana", "cherry")	y = list(x) y[1] = "kiwi" x = tuple(y) print(x)	('apple', 'kiwi', 'cherry')
fruits = ("apple", "banana", "cherry", "strawberry", "raspberry")	(green, yellow, *red) = fruits print(green) print(yellow) print(red) print(type(red))	apple banana ['cherry', 'strawberry', 'raspberry'] <class 'list'>

fruits = ("apple", "banana", "cherry")	mytuple = fruits * 2 print(mytuple.count("apple")) print(mytuple.index("banana"))	2 1

Set		
Input	**Python Code**	**Output**
myset = {"abc", 34, True, 40.5}	print(myset) print(len(myset)) print(type(myset)) print(34 in myset) myset.add("orange") print(myset)	{'abc', True, 34, 40.5} 4 <class 'set'> True {True, 34, 40.5, 'abc', 'orange'}
thisset = {"apple", "mango", "cherry"} tropical={"papaya", "mango"}	thisset=thisset+tropical print(thisset)	thisset=thisset+tropical TypeError: unsupported operand type(s) for +: 'set'
	thisset.update(tropical) print(thisset)	{'papaya', 'apple', 'cherry', 'mango'} PS C:\Users\kjsce_comp275\Documents\pyt
	thisset.intersection_update (tropical) print(thisset)	{'mango'}
	thisset.symmetric_difference_update(tropical) print(thisset)	{'papaya', 'apple', 'cherry'}

Dictionaries		
Input	**Python Code**	**Output**
thisdict={"brand":"Ford","model": "Mustang","year": 1964, "year": 2020}	print(thisdict) print(type(thisdict)) print(len(thisdict)) print(thisdict["brand"]) print(thisdict["year"]) x = thisdict.get("model") print(x) y = thisdict.keys() print(y) z = thisdict.values() print(z) thisdict["color"] = "white" print(thisdict) if "model" in thisdict: print("Yes")	{'brand': 'Ford', 'model': 'Mustang', 'year': 2020} <class 'dict'> 3 Ford 2020 Mustang dict_keys(['brand', 'model', 'year']) dict_values(['Ford', 'Mustang', 2020]) {'brand': 'Ford', 'model': 'Mustang', 'year': 2020, 'color': 'white' Yes
	thisdict["year"] = 2018 print(thisdict)	{'brand': 'Ford', 'model': 'Mustang', 'year': 2018}
	thisdict.pop("model") print(thisdict)	{'brand': 'Ford', 'year': 2020}

	for x in thisdict:
	  print(x)
	  print(thisdict[x])
		brand
		Ford
		model
		Mustang
		year
		2020

	for x, y in thisdict.items(): print(x, y)	brand Ford model Mustang year 2020

2. Write a Python program to take list values as input parameters and return another list without any duplicates.

3. Write a program that takes a string as input from the user and computes the frequency of each letter. Use a variable of dictionary type to maintain the count.

Books/journals/websites referred:

1. Reema Thareja, *Python Programming: Using Problem-Solving Approach*, Oxford University Press, First Edition 2017, India
2. Sheetal Taneja and Naveen Kumar, *Python Programming: A Modular Approach*, Pearson India, Second Edition 2018,

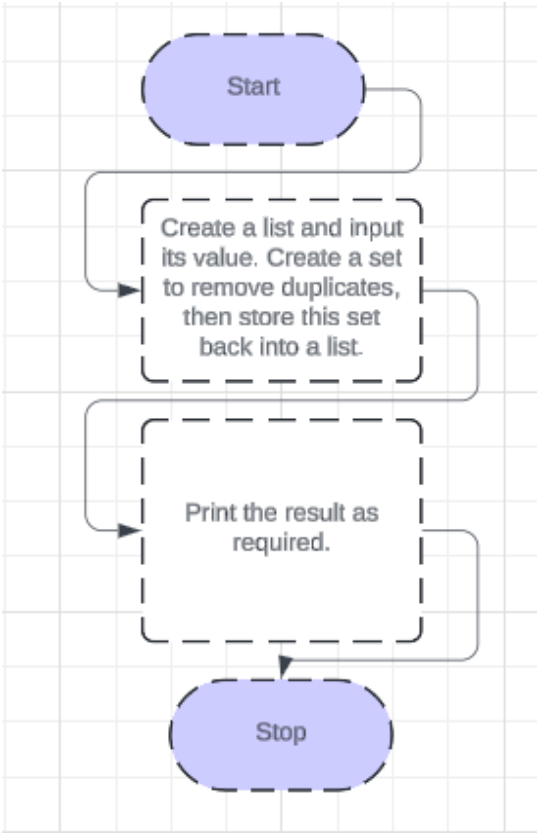
Implementation details:

Problem 1:

Algorithm:

1. Start
2. Create a list variable named list2, typecast it to type “list” and input value of the list.
3. Create a set called noDupe and typecast the list to the set. This step removes any duplicates automatically.
4. Now, store back this noDupe set into a list datatype by typecasting again.
5. Print the result as required.
6. Stop.

Flowchart:



Code:

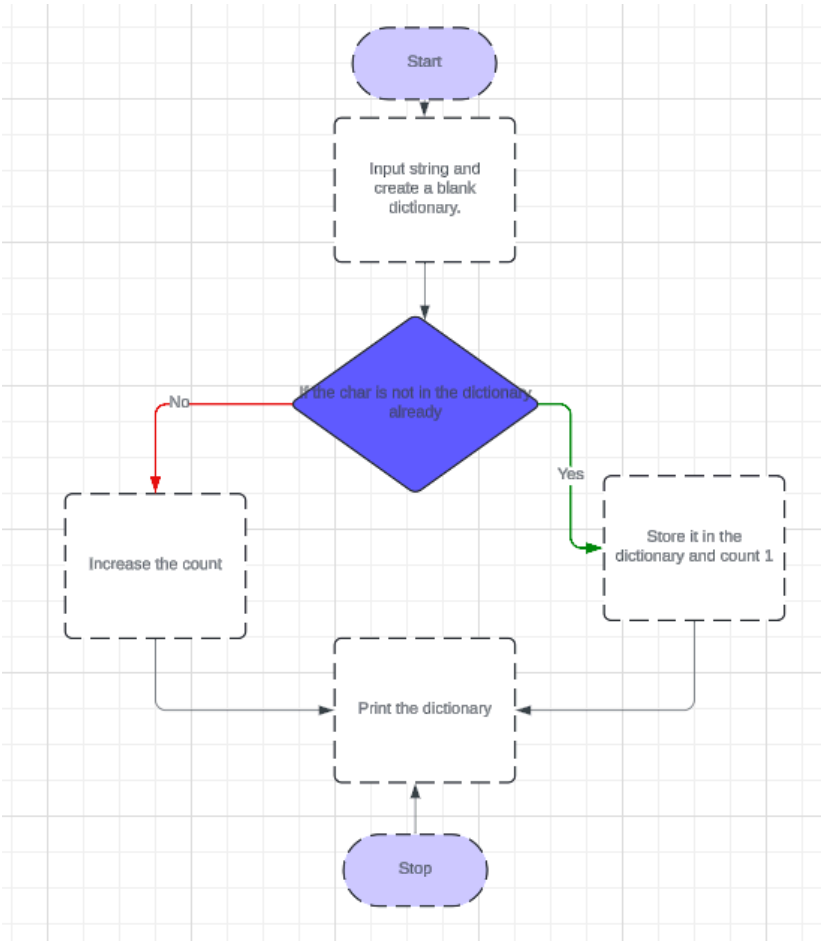
```
# String and frequency.py    nodupe.py
nodupe.py > ...
1   list2 = list(input("Enter your list: "))
2   noDupe = set(list2)
3   list1 = list(noDupe)
4   print(list1)
```

Problem 2:

Algorithm:

1. Start
2. Input a string.
3. Create a blank dictionary.
4. If the char is not in the dictionary already, store it in the dictionary and count 1.
5. If the char is in dictionary already, increase the count.
6. Print the dictionary.
7. Stop.

Flowchart:



Code:

```
# String and frequency.py
C: > Users > syeda > OneDrive > Desktop > ashwera > # String and frequency.py > ...
1   # String and frequency
2   string1 = str(input("Enter string: "))
3   d1={}
4   for i in string1:
5       if i not in d1:
6           d1[i]=string1.count(i)
7   print (d1)
```

Output(s):

1.

```
Enter your list: 4392742974
['4', '3', '2', '7', '9']
PS C:\Users\syeda\OneDrive\Desktop\Python>
```

2.

```
Enter string: ashwera
{'a': 2, 's': 1, 'h': 1, 'w': 1, 'e': 1, 'r': 1}
PS C:\Users\syeda\OneDrive\Desktop\Python>
```

Conclusion:

1. A list is used to store multiple values in one variable. These values can repeat.
2. A set is a collection of values that are totally unique. Duplicate values are automatically deleted from sets.
3. A dictionary is a datatype that stores key:value entries. The dictionary does not allow duplicates either, but we can manipulate it to count number of appearances like done in problem 2.

Post Lab Descriptive Questions

1. List out Mutable and Immutable Data Types in Python.
   Mutable data types in python are:
   a. List
   b. Dictionary
   Immutable data types in python include:
   a. Tuple
   b. Set
2. What do you mean by indexed and ordered data type in Python?
   When a data type is ordered and indexed, the entities in that data type have an accessible location (called the index) and the elements are in a certain order. Set is an unindexed data type in python. It is also unordered: the elements are not arranged in any order.
   Dictionary, tuple, and list are indexed and ordered data types. Their entities have an index where they can be called from, and they are also arranged in a proper ordered form inside that data type. This order can be ascending, descending, or random based on the need.

Date: 6th October, 2024 **Signature of faculty in-charge**