**Batch: C4-1 Roll No.: 24**

**Experiment / assignment / tutorial No. 2**

**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)  for i in range(len(thislist)):  print(thislist[i])  thislist.sort()  print(thislist) | Yes, 'apple' is in the fruits list  orange  mango  kiwi  pineapple  apple  orange  mango  kiwi  pineapple  apple  ['apple', 'kiwi', 'mango', 'orange', 'pineapple'] |
| thislist=["apple","banana","cherry"] | thislist.pop(1)  thislist.insert(1, '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) | 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) | TypeError: 'tuple' object does not support item assignment |
| 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) | {40.5, True, 34, 'abc'}  4  <class 'set'>  True  {True, 34, 40.5, 'abc', 'orange'} |
| thisset = {"apple", "mango", "cherry"}  tropical={"papaya", "mango"} | thisset=thisset+tropical  print(thisset) **Corrected Syntax:**  thisset.update(tropical)  print(thisset) | TypeError: unsupported operand type(s) for +: 'set' and 'set' **Corrected Output:**  {'papaya', 'apple', 'cherry', 'mango'} |
| thisset.update(tropical)  print(thisset) | {'papaya', 'apple', 'cherry', 'mango'} |
| thisset.intersection\_update (tropical)  print(thisset) | {'mango'} |
| thisset.symmetric\_difference\_update(tropical)  print(thisset) | {'cherry', 'papaya', 'apple'} |

|  |  |  |
| --- | --- | --- |
| **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:**

**Question 2 :**

list1 = []

i = *int*(input("Enter the Number of Elements you want to Type: "))-1

while(i>=0):

  list1.append(input("Input parameters: "))

  i -= 1

set1 = *set*(list1)

list1 = *list*(set1)

print(list1)

**Question 3:**

a = *str*(input("Enter a String: ")).upper()

b = *list*(a)

c = *list*(*set*(a))

d = {}

i = 0

while(i < len(c)):

  d.update({c[i]:b.count(c[i])})

  i += 1

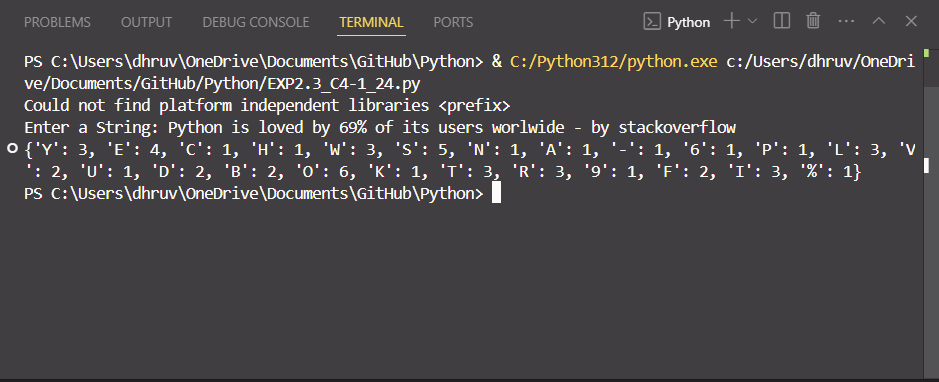
d.pop(" ")

print(d)

**Output(s):**

**Question 2 :**

**Question 3:**



**Conclusion:**

Learned to Declare a string and applied multiple methods and operations namely concatenation, slicing, length etc. Various string methods practiced like capitalize, replace, startswith, endswith, swapcase, zfill etc.

Coded a Python program to take list values as input parameters and return another list without any duplicates using the while loop to perform complex tasks on strings. Also coded a Python 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 using the data structures.

**Post Lab Descriptive Questions**

1. List out Mutable and Immutable Data Types in Python.

**Ans. Immutable Data Types**- String, int, tuple, float Boolean

**Mutable Data Types**-Lists, Dictionary, Set, and user-defined classes

1. What do you mean by indexed and ordered data type in Python?

**Ans.** These are a set of either mutable or immutable data items, not necessarily of same data types. Ordered data types are numerically indexed and ordered by the index of the element. We can access these elements just by knowing their positions. Index value data type shows the position at which the integer value data type is located. It also displays how many times the integer value is repeated.

**Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ Signature of faculty in-charge**