1 Assignment -2 (Data Structures)

1.1 1) What are data structures, and why are they important

Data structures are ways of organizing and storing data efficiently in a computer so that it can be accessed and modified effectively. They are important because the choice of data structure affects the performance of algorithms in terms of time and space. They affects the efficiency of operations like search, insertion, deletion ,etc.

1.2 2) Explain the difference between mutable and immutable data types with examples.

(i) Mutable : Mutable data types are those data types in which we can make changes after they are created . Exp : list,sets,etc.

(ii) Immutable : Immutable data types are those data types in which we cannot make changes once they are created . Exp : Tuple,Strings.

1.3 3) What are the main differences between lists and tuples in Python?

(i) Lists : Lists are ordered , mutable , indexed , duplicates allowed ,uses square brackets ('[]').

(ii) Tuple : Tuples are ordered, immutable, indexed , duplicates allowed, uses parentheses ('()').

The main difference between lists and tuples is that lists are mutable whereas tuples are not.

1.4 4) Describe how dictionaries store data.

Dictionaries is a unordered type of data structure , it stores the data in form of key-value pairs and no duplicate elements are allowed , where every keys have their specifi values and and the key's name is anything such as int, float ,string but not the special characters.

1.5 5) Why might you use a set instead of a list in Python?

I would use a set instead of a list in scenarios where:

(i) I need to eliminate duplicate values.

(ii) I need to perform set operations like union, intersection, and difference.

(iii I need fast membership testing (checking if an element is in the set), since sets have O(1) average time complexity for

lookups, unlike lists which have O(n) time complexity.

1.6 6)What is a string in Python, and how is it different from a list ?

String is an ordered sequence of characters and it stores the stores the data in single or double or triple quotoes.like 'ravi' or "ravi" ,etc and strings are immutable , whereas lists are mutable and it stores data in square brackets and separated by commas and lists can stores any data types .

1.7 7) How do tuples ensure data integrity in Python?

Tuples ensure data integrity by being immutable. Once a tuple is created it's values cannot be altered. This immutablitiy gurantees that the data remains constant throughout its existence .

1.8 8) What is a hash table, and how does it relate to dictionaries in Python?

A hash table is a data structure that stores key-value pairs using a hash function to compute an index into an array of buckets or slots, from which the desired value can be found. In Python, dictionaries are implemented using hash tables. Each key is hashed, and the value is stored at the resulting index in the table.

1.9 9) Can lists contain different data types in Python?

Yes , list can contain different data types in python.

1.10 10) Explain why strings are immutable in Python.

Strings are immutable in Python because:

(i) Hashing: Since strings are hashable (used as keys in dictionaries), their immutability ensures that their hash values remain consistent throughout their lifetime.

(ii) Thread-safety: Immutability ensures that strings are safe to use in multi-threaded applications, as they cannot be changed by one thread while another is accessing them.

(iii) Efficiency: Immutability allows for memory optimization. When multiple variables refer to the same string, they can share the same memory location, saving space.

1.11 11) What advantages do dictionaries offer over lists for certain tasks?

Advantages of dictionaries over lists:

(i)Fast lookups: You can retrieve a value by its key in O(1) time on average, whereas lists require O(n) time for searching.

(ii)Key-value pairs: You can store and retrieve data in a key-value format, which is more intuitive and efficient for certain tasks like counting frequencies or mapping items.

(iii) Fast insertion, deletion, and modification by key.

(iv) When we need to represent complex or nested data structures.

1.12 12) Describe a scenario where using a tuple would be preferable over a list.

A tuple is preferable when:

(i) When i need to store data should remain unchanged,such as coordinates(x,y),etc

(ii) You want to use it as a key in a dictionary because tuples are hashable, whereas lists are not.

1.13 13) How do sets handle duplicate values in Python?

In sets there are no duplicate values allowed , they just keep only one duplicate element and delete all the remaining duplicates.

1.14 14) How does the "in" keyword work differently for lists and dictionaries?

(i) Lists : In a list, the in keyword checks if an element exists in the list. It iterates through the list to find a match (O(n) time complexity).

(ii) Dictionaries : In a dictionary, the in keyword checks if a key exists in the dictionary. It uses a hash table for faster lookup (O(1) average time complexity).

1.15 15) Can you modify the elements of a tuple? Explain why or why not.

No , we cannot modify the elements of a tuple , because tuples are immutable and this immutability ensures data integrity and allows tuples to be used as keys in dictionaries.

1.16 16) What is a nested dictionary, and give an example of its use case?

A nested dictionary is a dictionary where the values themselves are dictionaries. This structure allows for representing more complex data.

Exp: Student's information:

students = {

"student1": {"name": "Ravi", "age": 18},

"student2": {"name": "Ayush", "age": 19}

}

1.17 17) Describe the time complexity of accessing elements in a dictionary.

The time complexity for accessing elements in a dictionary is O(1) on average.This is because dictionaries use a hash function to map keys to positions in an internal array (the hash table), allowing Python to directly compute the position where the value is stored.

1.18 18) In what situations are lists preferred over dictionaries?

Lists are preferred over dictionaries when:

(i) When we need to maintain the order of the elements.

(ii) When we need to store multiple elements of the same type and don't need to associate each element with a key.

(iii) We frequently iterate over the collection without needing fast lookups by key.

1.19 19) Why are dictionaries considered unordered, and how does that affect data retrieval?

Dictionaries are unordered because they are based on a hash table, which allows fast lookups but doesn't inherently preserve any specific order of elements in memory.

How Does This Affect Data Retrieval:

(i) Accessing Elements by Key (O(1)).

(ii) Iteration Over Keys/Values (Insertion Order in Python 3.7+).

1.19.1 (iv) Order-Based Operations.

1.20 20) Explain the difference between a list and a dictionary in terms of data retrieval.

(i) List: Data is accessed via an index. The index represents the position of an element, and retrieval takes O(1) time if you know the index.

(ii) Dictionary: Data is accessed via a key, and the value associated with the key is retrieved in O(1) average time due to hashing.

2 Practical questions

2.0.1 1) Write a code to create a string with your name and print it.

```
[259]: name = 'Ravi kumar yadav'
       name
```

```
[259]: 'Ravi kumar yadav'
```

2.0.2 2) Write a code to find the length of the string "Hello World"

```
[265]: length = len('Hello World')
       print("Lenght of \'Hello World\' is : ",length)
```

```
Lenght of 'Hello World' is :  11
```

2.0.3 3) Write a code to slice the first 3 characters from the string "Python Programming".

```
[275]: string = 'Python Programming'
       print('Slicing first 3 characters from \'Python Programming\' is  : ',string[:
        ↪3])
```

```
Slicing first 3 characters from 'Python Programming' is  :  Pyt
```

2.0.4 4) Write a code to convert the string "hello" to uppercase.

```
[281]: string2 = 'hello'
       print('Converting string \'hello\' to uppercase : ',string2.upper())
```

```
Converting string 'hello' to uppercase :  HELLO
```

2.0.5 5) Write a code to replace the word "apple" with "orange" in the string "I like apple".

```
[285]: string3 = 'I like apple'
       print('Replacing \'apple\' with \'orange\' : ',string3.
        ↪replace('apple','orange'))
```

```
Replacing 'apple' with 'orange' :  I like orange
```

2.0.6 6) Write a code to create a list with numbers 1 to 5 and print it.

```
[1]: list1 = [1,2,3,4,5]
     print('List with number 1 to 5 : ',list1)
```

```
List with number 1 to 5 :  [1, 2, 3, 4, 5]
```

2.0.7 7) Write a code to append the number 10 to the list [1, 2, 3, 4].

```
[12]: list2 = [1,2,3,4]
      list2.append(10)
      print('To append \'10\' in the list2 : ',list2)
```

```
To append '10' in the list2 :  [1, 2, 3, 4, 10]
```

2.0.8 8) Write a code to remove the number 3 from the list [1, 2, 3, 4, 5].

```
[14]: list3 = [1,2,3,4,5]
      list3.remove(3)
      print('To remove \'3\' from list3 : ',list3)
```

```
To remove '3' from list3 :  [1, 2, 4, 5]
```

2.0.9 9) Write a code to access the second element in the list ['a', 'b', 'c', 'd'].

```
[16]: list4 = ['a','b','c','d']
      print('To access second element of list4 : ',list4[1])
```

```
To access second element of list4 :  b
```

2.0.10 10) Write a code to reverse the list [10, 20, 30, 40, 50].

```
[18]: list5 = [10,20,30,40,50]
      print('To reverse the list5 : ',list5[::-1])
```

```
To reverse the list5 :  [50, 40, 30, 20, 10]
```

2.0.11 11) Write a code to create a tuple with the elements 10, 20, 30 and print it.

```
[20]: tuple1 = (10,20,30)
      print('Tuple have elements : ',tuple1)
```

```
Tuple have elements :  (10, 20, 30)
```

2.0.12 12) Write a code to access the first element of the tuple ('apple', 'banana', 'cherry').

```
[25]: tuple2 = ('apple','banana','cherry')
      print('To access \'first\' element of tuple2 : ',tuple2[0])
```

```
To access 'first' element of tuple2 :  apple
```

2.0.13 13) Write a code to count how many times the number 2 appears in the tuple (1, 2, 3, 2, 4, 2).

[31]:
```
tuple3 = (1,2,3,2,4,2)
print('Number of times \'2\' appears in tuple3 is  : ',tuple3.count(2))
```

```
Number of times '2' appears in tuple3 is  :  3
```

2.0.14 14) Write a code to find the index of the element "cat" in the tuple ('dog', 'cat', 'rabbit').

[35]:
```
tuple4 = ('dog','cat','rabbit')
print('Index of \'cat\' is : ',tuple4.index('cat'))
```

```
Index of 'cat' is :  1
```

2.0.15 15) Write a code to check if the element "banana" is in the tuple ('apple', 'orange', 'banana').

[43]:
```
tuple5 = ('apple','orange','banana')
print('\'banana\' is in tupple5 : ','banana' in tuple5)
```

```
'banana' is in tupple5 :  True
```

2.0.16 16) Write a code to create a set with the elements 1, 2, 3, 4, 5 and print it.

[45]:
```
set1 = {1,2,3,4,5}
print('Elements of set1 are : ',set1)
```

```
Elements of set1 are :  {1, 2, 3, 4, 5}
```

2.0.17 17) Write a code to add the element 6 to the set {1, 2, 3, 4}.

[49]:
```
set2 = {1,2,3,4}
set2.add(6)
print('Adding \'6\' to set2 : ',set2)
```

```
Adding '6' to set2 :  {1, 2, 3, 4, 6}
```

2.0.18 18) Write a code to create a tuple with the elements 10, 20, 30 and print it.

[51]:
```
tupl = (10,20,30)
print('Elements of tuple are : ',tupl)
```

```
Elements of tuple are :  (10, 20, 30)
```

2.0.19 19) Write a code to access the first element of the tuple (‘apple’, ‘banana’, ‘cherry’).

```
[53]: tuple2 = ('apple','banana','cherry')
      print('To access \'first\' element of tuple2 : ',tuple2[0])
```

```
To access 'first' element of tuple2 :  apple
```

2.0.20 20) Write a code to count how many times the number 2 appears in the tuple (1, 2, 3, 2, 4, 2).

```
[55]: tuple3 = (1,2,3,2,4,2)
      print('Number of times \'2\' appears in tuple3 is  : ',tuple3.count(2))
```

```
Number of times '2' appears in tuple3 is  :  3
```

2.0.21 21) Write a code to find the index of the element “cat” in the tuple (‘dog’, ‘cat’, ‘rabbit’).

```
[57]: tuple4 = ('dog','cat','rabbit')
      print('Index of \'cat\' is : ',tuple4.index('cat'))
```

```
Index of 'cat' is :  1
```

2.0.22 22) Write a code to check if the element “banana” is in the tuple (‘apple’, ‘orange’, ‘banana’).

```
[59]: tuple5 = ('apple','orange','banana')
      print('\'banana\' is in tupple5 : ','banana' in tuple5)
```

```
'banana' is in tupple5 :  True
```

2.0.23 23) Write a code to create a set with the elements 1, 2, 3, 4, 5 and print it.

```
[61]: set1 = {1,2,3,4,5}
      print('Elements of set1 are : ',set1)
```

```
Elements of set1 are :  {1, 2, 3, 4, 5}
```

2.0.24 24) Write a code to add the element 6 to the set {1, 2, 3, 4}.

```
[63]: set2 = {1,2,3,4}
      set2.add(6)
      print('Adding \'6\' to set2 : ',set2)
```

```
Adding '6' to set2 :  {1, 2, 3, 4, 6}
```

```
[ ]:
```