

## Question 1

What are data structures, and why are they important ? Data structures are specialized formats for organizing, processing, retrieving, and storing data within a computer system. Efficient Algorithms: Data structures are the foundation for designing efficient algorithms, which are sets of instructions used to solve problems. Problem Solving: Many computer science problems require the manipulation of data, and data structures provide a systematic way to approach these problems. Code Reusability: Well-designed data structures can be reused across different projects, saving time and effort in software development. Memory Management: Data structures help manage memory efficiently, reducing memory leaks and optimizing resource allocation. Scalability: As data sizes grow, the choice of data structure becomes critical. A well-chosen data structure can handle increases in data volume gracefully, ensuring the application remains efficient and scalable. Performance: Data structures can significantly impact the performance of a program. Choosing the right data structure for a task can reduce the amount of time and memory required to perform specific operations. Readability: Well-organized data structures can make code easier to understand and maintain.

## # Question 2

Explain the difference between mutable and immutable data types with examples?

## ✓ Answer of question 2

Mutable Data Types: Definition: Mutable data types allow modifications to their values after they are created. Changes are made directly to the object itself, without creating a new one.

example Examples of mutable data types include lists and dictionaries.

Immutable Data Types: Definition: Immutable data types do not allow changes to their values once created. Any attempt to modify them results in creating a new object.

examples of immutable data types include strings and tuples.

## # Question 3

What are the main differences between lists and tuples in Python

## # Answer of question 3

The primary difference between tuples and lists is that tuples are immutable instead of mutable lists. Therefore, it is possible to change a list but not a tuple.

Due to tuples' immutability, the contents of a tuple cannot change once it has been created in Python.

## # Questionn 4

Describe how dictionaries store data ?

## # Answer of question 4

Dictionaries store data as key-value pairs. Each key in a dictionary is unique and used to access its corresponding value. Dictionaries are like lookup tables, allowing for fast retrieval of data using the key. Keys can be of various immutable types like strings, numbers, or tuples, while values can be any data type.

## # Question 5

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

```
# Answer of question 5
```

Because sets cannot have multiple occurrences of the same element, it makes sets highly useful to efficiently remove duplicate values from a list or tuple and to perform common math operations like unions and intersections.

```
# Question 6
```

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

```
# Answer of question 6
```

In Python, a string is a sequence of characters enclosed in single or double quotes. Lists are mutable sequences of items, enclosed in square brackets. The key difference lies in their mutability: strings are immutable (cannot be changed after creation), while lists are mutable (can be modified).

```
# Question 7
```

How do tuples ensure data integrity in Python ?

```
# Answer of Question 7
```

Tuples are immutable to ensure that their contents remain constant throughout their lifecycle, guaranteeing data integrity and reliability. This immutability allows tuples to be used as keys in dictionaries and elements in sets, as they can be hashed.

```
# Question of 8
```

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

```
# Answer of Question 8
```

A hash table is a data structure that uses a hash function to map keys to their corresponding values in an array.

Dictionaries in Python:-

In Python, dictionaries are a built-in data type that is implemented using hash tables. When you create a dictionary, you're essentially creating a hash table where keys are hashed and used to find the corresponding values.

```
# Question 9
```

Can lists contain different data types in Python ?

```
# Answer of Question 9
```

In Python, lists can contain heterogeneous data types and objects. For instance, integers, strings, and even functions can be stored within the same list.

```
# Question 10
```

Explain why strings are immutable in Python ?

```
# Answer of Question 10
```

Python strings are "immutable" which means they cannot be changed after they are created.

# Question 11

What advantages do dictionaries offer over lists for certain tasks ?

# Answer of question 11

Dictionaries offer several advantages over lists, primarily for tasks involving fast data retrieval and key-value association. Dictionaries use hash tables for quick access to values based on keys, while lists require sequential traversal, making dictionary lookups significantly faster.

# Question 12

Describe a scenario where using a tuple would be preferable over a list ?

# Answer of Question 12

**Immutability:** Tuples are immutable, meaning their contents cannot be changed after creation. This makes them suitable for representing fixed collections of items where you want to ensure that the data remains constant. For example, using a tuple to represent coordinates (x, y) ensures that these values cannot be altered accidentally. **Performance:** Tuples can be more memory-efficient and faster than lists for certain operations since they have a smaller memory overhead due to their immutability. If you need a collection of items that doesn't change, using a tuple can lead to better performance. **Hashability:** Tuples can be used as keys in dictionaries or elements in sets because they are hashable (as long as their elements are hashable). Lists, on the other hand, cannot be used in these contexts due to their mutability. **Semantic Meaning:** When the data represents a fixed structure or a record, tuples can provide clearer semantics. For example, using a tuple to represent a database record (like (id, name, age)) can convey that the data structure is intended to be a fixed, non-changing set of values. **Multiple Return Values:** Tuples are commonly used to return multiple values from a function. This makes it clear that the function is returning a fixed number of related values.

# Question 13

How do sets handle duplicate values in Python ?

# Answer of Question 13

Sets store unique elements, meaning that duplicate values are automatically removed. Unordered.

# Question 14

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

The in operator for dictionaries ( dict ) checks for the presence of a key. Use the values() and items() methods to test for the presence of values or key-value pairs.

In keyword can be used to test whether certain characters are present in a string or a certain value present in a list or tuple.

# Question 15

Can you modify the elements of a tuple? Explain why or why not ?

# Answer of Question 15

No, you cannot directly modify the elements of a tuple in Python once it's created. Tuples are immutable, meaning their elements cannot be changed, added, or removed after creation. Trying to modify a tuple will result in a `TypeError`.

A nested dictionary is a dictionary where one or more of its values are themselves dictionaries. It's like having a dictionary inside another dictionary, allowing you to store hierarchical or multi-layered data.

# Question 16

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


# Answer of question 16

A nested dictionary is a dictionary where one or more of its values are themselves dictionaries. It's like having a dictionary inside another dictionary, allowing you to store hierarchical or multi-layered data.

example of its use case:-

```
people = {1: {'name': 'John', 'age': '27', 'sex': 'Male'},
          2: {'name': 'Marie', 'age': '22', 'sex': 'Female'}}

print(people)
```



```
{1: {'name': 'John', 'age': '27', 'sex': 'Male'}, 2: {'name': 'Marie', 'age': '22', 'sex': 'Female'}}
```

# Question 17

Describe the time complexity of accessing elements in a dictionary ?

# Answer of question 17

Accessing an element in a dictionary typically has a time complexity of  $O(1)$  (constant time) on average. This means the time it takes to retrieve a value using its key remains the same, regardless of the dictionary's size.

# Question 18

In what situations are lists preferred over dictionaries ?

# Answer of question 18

For quick data look-ups, configurations, or caches, favor dictionaries. For ordered collections and sequence operations, such as maintaining a stack or queue, lists are more suitable.

# Question 19

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

# Answer of Question 19

Dictionaries are considered unordered because they store data as key-value pairs without maintaining any inherent order of insertion. This means that when you retrieve data from a dictionary, the order of the elements returned will not necessarily match the order in which they were added.

# Question 20

Explain the difference between a list and a dictionary in terms of data retrieval ?

# Answer of Question 20

In terms of data retrieval, lists use integer indices (starting from 0) to access elements, while dictionaries use unique keys to access values. Lists offer sequential access,

while dictionaries provide fast, direct access based on the key.

#####

Practical Questions

# Question 1

Write a code to create a string with your name and print it ?

# Answer of Question 1

```
#strings
string = "Harshita dube"
```

```
string
```

```
↩ 'Harshita dube'
```

```
type(string)
```

```
↩ str
```

# Question 2

Write a code to find the length of the string "Hello World" ?

# Answer of Question 2

```
string = "Hello world"
```

```
string
```

```
↩ 'Hello world'
```

```
len(string)
```

```
↩ 11
```

# Question 3

Write a code to slice the first 3 characters from the string "Python Programming"?

# Answer of Question 3

```
text = "python programming"
sliced_text = text[:3]
print(sliced_text)
```


 pyt

#Question 4

Write a code to convert the string "hello" to uppercase ?

# Answer of Question 5

```
txt = "Hello"
x = txt.upper()
print(x)
```

 HELLO

# Question 5

Write a code to replace the word "apple" with "orange" in the string "I like apple?"

# Answer of question 5

```
text = "I like apple"
new_text = text.replace("apple", "orange")
print(new_text)
```


 I like orange

#Question 6

Write a code to create a list with numbers 1 to 5 and print it ?

# Answer of question 6

```
mylist = [1,2,3,4,5]
print(mylist)
```


 [1, 2, 3, 4, 5]

# Question 7

Write a code to append the number 10 to the list [1, 2, 3, 4] ?

# Answer of Question 7

```
my_list = [1, 2, 3, 4]
my_list.append(10)
print(my_list)
```

 [1, 2, 3, 4, 10]

# Question 8

Write a code to remove the number 3 from the list [1, 2, 3, 4, 5] ?

# Answer of Question 8

```
my_list = [1, 2, 3, 4, 5]
my_list.remove(3)
print(my_list)
```

➦ [1, 2, 4, 5]

# Question 9

Write a code to access the second element in the list ['a', 'b', 'c', 'd']?

# Answer of Question 9

```
my_list = ['a', 'b', 'c', 'd']
second_element = my_list[1]
print(second_element)
```

➦ b

# Question 10

Write a code to reverse the list [10, 20, 30, 40, 50] ?

# Answer of Question 10

```
my_list = [10, 20, 30, 40, 50]
my_list.reverse()
print(my_list)
```

➦ [50, 40, 30, 20, 10]

# Question 11

Write a code to create a tuple with the elements 100, 200, 300 and print it.?

# Answer of Question 11

```
my_tuple = (100, 200, 300)
print(my_tuple)
```

➦ (100, 200, 300)

# Question 12

Write a code to access the second-to-last element of the tuple ('red', 'green', 'blue', 'yellow') ?

```
my_tuple = ('red', 'green', 'blue', 'yellow')
second_to_last_element = my_tuple[-2]
print(second_to_last_element)
```

➦ blue

#Question 13

Write a code to find the minimum number in the tuple (10, 20, 5, 15). ?

# Answer of Question 13

```
my_tuple = (10, 20, 5, 15)
min_number = min(my_tuple)
print(min_number)
```

 5

# Question 14

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

# Answer of Question 14


```
my_tuple = ('dog', 'cat', 'rabbit')
index_of_cat = my_tuple.index('cat')
print(index_of_cat)
```

 1

# Question 15

Write a code to create a tuple containing three different fruits and check if "kiwi" is in it.


```
this_tuple = ("apple", "banana", "kiwi")
if "apple" in this_tuple:
    print("yes, 'kiwi' is in the fruits tuple")
```

 yes, 'kiwi' is in the fruits tuple

# Question 16

Write a code to create a set with the elements 'a', 'b', 'c' and print it ?

```
my_set = {'a', 'b', 'c'}
print(my_set)
```


 {'a', 'b', 'c'}

# Question 17

Write a code to clear all elements from the set {1, 2, 3, 4, 5}.

# Answer of Question 17

```
my_set = {1, 2, 3, 4, 5}
my_set.clear()
print(my_set)
```


 set()

# Question 18

Write a code to remove the element 4 from the set {1, 2, 3, 4} ?

# Answer of Question 18

```
my_set = {1, 2, 3, 4}
my_set.remove(4)
print(my_set)
```

 {1, 2, 3}

# Question 19



Write a code to find the union of two sets {1, 2, 3} and {3, 4, 5} ?

# Answer of Question 19

```
A = {1, 2, 3}
B = {3, 4, 5}
print('A U B = ', A.union(B))
```

↩ A U B = {1, 2, 3, 4, 5}

# Question 20

. Write a code to find the intersection of two sets {1, 2, 3} and {2, 3, 4}.?

# Answer of Question 20

```
A = {1, 2, 3}
B = {2, 3, 4}
print(A.intersection(B))
```

↩ {2, 3}

# Question 21

Write a code to create a dictionary with the keys "name", "age", and "city", and print it. ?

# Answer of Question 21

```
my_information = {'name': 'Rohit', 'age': 28, 'City': 'Indore'}
print(my_information)
```

↩ {'name': 'Rohit', 'age': 28, 'City': 'Indore'}

#Question 22

Write a code to add a new key-value pair "country": "USA" to the dictionary {'name': 'John', 'age': 25}. ?

# Answer of Question 22

```
my_dictionary = {'name': 'John', 'age': 25}
my_dictionary['country'] = 'USA'
print(my_dictionary)
```

↩ {'name': 'John', 'age': 25, 'country': 'USA'}

# Question 23

Write a code to access the value associated with the key "name" in the dictionary {'name': 'Alice', 'age': 30} ?

# Answer of Question 23

```
my_dict = {'name': 'Alice', 'age': 30}
print(my_dict['name'])
```

↩ Alice

# Question 24

Write a code to remove the key "age" from the dictionary {'name': 'Bob', 'age': 22, 'city': 'New York'} ?

# Answer of Question 24

```
my_dict = {"name": "Bob", "age": 22, "city": "New York"}
del my_dict["age"]
print(my_dict) # Output: {'name': 'Alice'}
```

```
➦ {'name': 'Bob', 'city': 'New York'}
```

# Question 25

Write a code to check if the key "city" exists in the dictionary {'name': 'Alice', 'city': 'Paris'}.

# Answer of Question 25

```
my_dict = {'name': 'Alice', 'city': 'Paris'}
if 'city' in my_dict:
    print("The key 'city' exists in the dictionary.")
else:
    print("The key 'city' does not exist in the dictionary.")
```

```
➦ The key 'city' exists in the dictionary.
```

# Question 26

Write a code to create a list, a tuple, and a dictionary, and print them all.?

```
#List
my_list = [1, 2, 3, 4, 5]
print(my_list)

#Tuple
my_tuple = (1, 2, 3, 4, 5)
print(my_tuple)

#Dictionary
my_dict = {'name': 'Alice', 'age': 25}
print(my_dict)
```

```
➦ [1, 2, 3, 4, 5]
  (1, 2, 3, 4, 5)
  {'name': 'Alice', 'age': 25}
```

# Question 27

Write a code to create a list of 5 random numbers between 1 and 100, sort it in ascending order, and print the result.(replaced)

# Answer of Question 27

```
import random
random_numbers = [random.randint(1, 100) for _ in range(5)]
random_numbers.sort()
print(random_numbers)
```


```
➦ [5, 8, 24, 54, 58]
```

# Question 28

Write a code to create a list with strings and print the element at the third index ?

# Answer of Question 28

```
my_list = ["apple", "banana", "cherry", "date"]
print(my_list[3])
```


 date

# Question 29

Write a code to combine two dictionaries into one and print the result ?

# Answer of Question 29

```
my_list1 = [1, 2, 3, 4, 5]
my_list2 = [6, 7, 8, 9, 10]
combined_list = my_list1 + my_list2
print(combined_list)
```


 [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# Question 30

Write a code to convert a list of strings into a set. ?

# Answer of Question 30

```
my_list = ["apple", "banana", "cherry", "date"]
my_set = set(my_list)
print(my_set)
```

 {'date', 'banana', 'cherry', 'apple'}

#####

Start coding or [generate](#) with AI.

Double-click (or enter) to edit

Double-click (or enter) to edit

Start coding or [generate](#) with AI.

Double-click (or enter) to edit

Start coding or [generate](#) with AI.

Double-click (or enter) to edit

Start coding or [generate](#) with AI.

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Start coding or [generate](#) with AI.

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Double-click (or enter) to edit

**string**

Double-click (or enter) to edit

Double-click (or enter) to edit

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

>

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Double-click (or enter) to edit

Double-click (or enter) to edit

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Double-click (or enter) to edit

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Double-click (or enter) to edit

Start coding or [generate](#) with AI.

Double-click (or enter) to edit

Double-click (or enter) to edit

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.

Start coding or [generate](#) with AI.