Data Types and Structures

Question1 : What are data structures, and why are they important?

Answer: Data structures are ways of organizing and storing data efficiently to facilitate access, modification, and management. They are crucial in software development as they optimize operations like searching, sorting, and data retrieval.

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

Answer:Mutable: Can be changed after creation (e.g., lists - my_list = [1, 2, 3]).

Immutable: Cannot be modified after creation (e.g., strings - my_string = "hello").

Question 3. What are the main differences between lists and tuples in Python?

Answer:Lists: Mutable, slower, defined using square brackets ([]).

Tuples: Immutable, faster, defined using parentheses (()).

Question 4. Describe how dictionaries store data.
Answer:Dictionaries store data in key-value pairs using a hash table. Keys are unique, and each key maps to a specific value.

Question 5. Why might you use a set instead of a list in Python?
Answer:Sets are used when you need to store unique elements and perform operations like union, intersection, or difference.

Question 6. What is a string in Python, and how is it different from a list?
Answer:A string is a sequence of characters, immutable, and enclosed in quotes. A list is mutable and can store elements of any data type.

Question 7. How do tuples ensure data integrity in Python?
Answer:Tuples are immutable, preventing accidental modification and ensuring data remains constant.

Question 8. What is a hash table, and how does it relate to dictionaries in Python?
Answer:A hash table is a data structure that maps keys to values using a hashing function. Dictionaries in Python are implemented using hash tables.

Question 9. Can lists contain different data types in Python?
Answer:Yes, lists can store mixed data types, e.g., [1, "hello", 3.5].

Question 10. Explain why strings are immutable in Python.
Answer:Strings are immutable to optimize memory usage, improve performance, and ensure thread safety.

Question 11. What advantages do dictionaries offer over lists for certain tasks?
Answer:Dictionaries provide faster lookups, data retrieval by keys, and allow key-value mapping.

Question 12. Describe a scenario where using a tuple would be preferable over a list.
Answer:Tuples are preferable when storing constant data like geographic coordinates ((40.7128, -74.0060)).

Question 13. How do sets handle duplicate values in Python?
Answer:Sets automatically eliminate duplicate values, ensuring all elements are unique.

Question 14. How does the “in” keyword work differently for lists and dictionaries?

Answer:In lists, it checks for the presence of a value.

In dictionaries, it checks for the presence of a key.

Question 15. Can you modify the elements of a tuple? Explain why or why not.
Answer:No, tuples are immutable, so their elements cannot be changed after creation.

Question 16. What is a nested dictionary, and give an example of its use case?
Answer:A nested dictionary contains dictionaries as values. Example:

student = {'name': 'John', 'grades': {'math': 90, 'science': 85}}

Question 17. Describe the time complexity of accessing elements in a dictionary.
Answer:Accessing elements in a dictionary has an average time complexity of O(1) due to hashing.

Question 18. In what situations are lists preferred over dictionaries?
Answer:Lists are preferred when order is important or when storing sequential data.

Question 19. Why are dictionaries considered unordered, and how does that affect data retrieval?
Answer:Dictionaries were unordered prior to Python 3.7. In earlier versions, data retrieval did not follow insertion order.

Question 20. Explain the difference between a list and a dictionary in terms of data retrieval.
Answer:Lists retrieve elements by index, while dictionaries retrieve values using keys, making dictionary lookups faster.

Practical Questions

```python
# 1. Create a string with your name and print it
name = "Dharmendra Sharma"
print(name)

# 2. Find the length of the string "Hello World"
print(len("Hello World"))

# 3. Slice the first 3 characters from "Python Programming"
print("Python Programming"[:3])

# 4. Convert "hello" to uppercase
print("hello".upper())

# 5. Replace "apple" with "orange" in "I like apple"
print("I like apple".replace("apple", "orange"))
```

```
# 6. Create a list with numbers 1 to 5 and print it
numbers = [1, 2, 3, 4, 5]
print(numbers)

# 7. Append the number 10 to the list [1, 2, 3, 4]
list = [1, 2, 3, 4]
lst.append(10)
print(list)

# 8. Remove the number 3 from the list [1, 2, 3, 4, 5]
lst2 = [1, 2, 3, 4, 5]
lst2.remove(3)
print(lst2)

# 9. Access the second element in the list ['a', 'b', 'c', 'd']
letters = ['a', 'b', 'c', 'd']
print(letters[1])

# 10. Reverse the list [10, 20, 30, 40, 50]
reverse_list = [10, 20, 30, 40, 50]
reverse_list.reverse()
print(reverse_list)

# 11. Create a tuple with the elements 100, 200, 300 and print it
my_tuple = (100, 200, 300)
print(my_tuple)

# 12. Access the second-to-last element of the tuple ('red',
'green', 'blue', 'yellow')
colors = ('red', 'green', 'blue', 'yellow')
print(colors[-2])

# 13. Find the minimum number in the tuple (10, 20, 5, 15)
numbers_tuple = (10, 20, 5, 15)
print(min(numbers_tuple))

# 14. Find the index of "cat" in the tuple ('dog', 'cat', 'rabbit')
animals = ('dog', 'cat', 'rabbit')
print(animals.index('cat'))
```

```
# 15. Create a tuple with fruits and check if "kiwi" is in it
fruits = ('apple', 'banana', 'orange')
print('kiwi' in fruits)

# 16. Create a set with elements 'a', 'b', 'c' and print it
my_set = {'a', 'b', 'c'}
print(my_set)

# 17. Clear all elements from the set {1, 2, 3, 4, 5}
clear_set = {1, 2, 3, 4, 5}
clear_set.clear()
print(clear_set)

# 18. Remove element 4 from the set {1, 2, 3, 4}
set_with_elements = {1, 2, 3, 4}
set_with_elements.remove(4)
print(set_with_elements)

# 19. Find the union of two sets {1, 2, 3} and {3, 4, 5}
set1 = {1, 2, 3}
set2 = {3, 4, 5}
print(set1.union(set2))

# 20. Find the intersection of two sets {1, 2, 3} and {2, 3, 4}
print(set1.intersection(set2))

# 21. Create a dictionary with keys "name", "age", "city" and print
it
my_dict = {"name": "John", "age": 30, "city": "New York"}
print(my_dict)

# 22. Add a new key-value pair "country": "USA" to the dictionary
my_dict["country"] = "USA"
print(my_dict)

# 23. Access the value associated with the key "name" in the
dictionary
print(my_dict["name"])
```

```python
# 24. Remove the key "age" from the dictionary {'name': 'Bob',
'age': 22, 'city': 'New York'}
del my_dict["age"]
print(my_dict)

# 25. Check if the key "city" exists in the dictionary {'name':
'Alice', 'city': 'Paris'}
print("city" in my_dict)

# 26. Create a list, a tuple, and a dictionary, and print them all
my_list = [1, 2, 3]
my_tuple2 = (4, 5, 6)
my_dict2 = {"key1": "value1", "key2": "value2"}
print(my_list, my_tuple2, my_dict2)

# 27. Create a list of 5 random numbers between 1 and 100, sort it,
and print the result
import random
random_numbers = sorted(random.sample(range(1, 101), 5))
print(random_numbers)

# 28. Create a list with strings and print the element at the third
index
string_list = ["apple", "banana", "cherry", "date", "fig"]
print(string_list[3])

# 29. Combine two dictionaries into one and print the result
dict1 = {"a": 1, "b": 2}
dict2 = {"c": 3, "d": 4}
combined_dict = {**dict1, **dict2}
print(combined_dict)

# 30. Convert a list of strings into a set
string_list2 = ["apple", "banana", "cherry", "apple"]
string_set = set(string_list2)
print(string_set)
```

Name: Dharmendra sharma
Email:sandysharma0783@gmail.com
Assignment Name:Data Types and Structures
Github link : code