```
                                        ##LIST

#Q1. How to declare a list?

#you can use square brackets [] and separate the elements by commas.
my_list = [1, 2, 3, 4, 5]

#my_list containing integers from 1 to 5. Lists in Python can contain
elements of different types, including integers, strings, or even
other lists
mixed_list = [1, "hello", 3.14, True]

nested_list = [[1, 2], [3, 4], [5, 6]]

#You can also create an empty list by simply using empty square
brackets:
empty_list = []

#Q2. Declare a list containing different fruit names?

fruits = ["apple", "banana", "orange", "grape", "kiwi"]

#This creates a list named fruits containing five different fruit
names: "apple", "banana", "orange", "grape", and "kiwi".

#Q3. Declare a list containing different data-types in it?

mixed_list = [1, "hello", 3.14, True]

1 is an integer.
"hello" is a string.
3.14 is a float.
True is a boolean value.

#This mixed_list contains elements of different data types: integer,
string, float, and boolean.

#Q4. write down the code to check a particular element in the list?

my_list = ["apple", "banana", "orange", "grape", "kiwi"]

if "banana" in my_list:
    print("The element 'banana' is in the list.")
else:
    print("The element 'banana' is not in the list.")

 #This code checks if the element "banana" is present in the my_list.
If it is, it prints "The element 'banana' is in the list."; otherwise,
it prints "The element 'banana' is not in the list."
```

```
The element 'banana' is in the list.
```

```
#Q5. create a list of 5 elements and print the middle element which is in the
list?

my_list = [1, 2, 3, 4, 5]

middle_index = len(my_list) // 2  # Using floor division to get the middle index
middle_element = my_list[middle_index]

print("The middle element is:", middle_element)
```

```
The middle element is: 3
```

```
#Q6. A list containing 10 elements slice it from index 3 to 4?

my_list = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

sliced_list = my_list[3:5]  # Slicing from index 3 to index 4

print("Sliced list:", sliced_list)
```

```
Sliced list: [3, 4]
```

```
#Q7. Declare a list and add a new element into the list using the append function?

my_list = [1, 2, 3, 4, 5]

# Adding a new element to the list using append
my_list.append(6)

print("Updated list:", my_list)
```

```
Updated list: [1, 2, 3, 4, 5, 6]
```

```
#Q8. Declare a list store multiple elements and access the elements using negative indexing?

my_list = ["apple", "banana", "orange", "grape", "kiwi"]

# Accessing elements using negative indexing
last_element = my_list[-1]
second_last_element = my_list[-2]

print("Last element:", last_element)
print("Second-to-last element:", second_last_element)
```

```
Last element: kiwi
Second-to-last element: grape
```

```
## Tuple
```

#Q1. How tuples are different from the list [theory type question ?]

Tuples

1. Are immutable, meaning once they are created, you cannot change their elements. You can't add, remove, or modify elements in a tuple.
2. Tuples are defined using parentheses ( ).
3.Tuples are used for collections of items that should not be changed, such as coordinates, database records, or configuration settings.
4.Tuples are generally faster than lists because they are immutable. Once created, their contents cannot be changed, which makes them more memory efficient and potentially faster to access.
5.Iterating over a list can be slightly faster than iterating over a tuple because lists use a dynamic array structure, while tuples use a fixed-size array.

List

1.Iterating over a list can be slightly faster than iterating over a tuple because lists use a dynamic array structure, while tuples use a fixed-size array.
2.Lists are defined using square brackets [ ].
3.Lists are used for collections of items that may need to be modified, sorted, or iterated over.

#Q2. Write a Python program to create a tuple?

```
# Creating a tuple
my_tuple = (1, 2, 3, 'hello', 'world')

# Printing the tuple
print("Tuple:", my_tuple)
```

```
Tuple: (1, 2, 3, 'hello', 'world')
```

 # Q3. Write a Python program to create a tuple with different data types?

```
# Creating a tuple with different data types
mixed_tuple = (1, 'hello', 3.14, True)

# Printing the tuple
print("Mixed Tuple:", mixed_tuple)
```

```
Mixed Tuple: (1, 'hello', 3.14, True)
```

#Q4. Write a Python program to create a tuple with numbers and print one item?

```python
# Creating a tuple with numbers
number_tuple = (10, 20, 30, 40, 50)

# Printing one item from the tuple
print("Item at index 2:", number_tuple[2])
```

```
Item at index 2: 30
```

```python
#Q5. Write a Python program to add an item in a tuple?

# Original tuple
original_tuple = (1, 2, 3, 4, 5)

# Item to add
new_item = 6

# Create a new tuple by concatenating the original tuple and the new item as a tuple
new_tuple = original_tuple + (new_item,)

# Printing the new tuple
print("New Tuple:", new_tuple)
```

```
New Tuple: (1, 2, 3, 4, 5, 6)
```

```python
#Q6. Write a Python program to get the 4th element and 4th element from the last of a tuple?

# Sample tuple
my_tuple = (1, 2, 3, 4, 5, 6, 7, 8, 9)

# Getting the 4th element (indexing starts from 0)
fourth_element = my_tuple[3]

# Getting the 4th element from the last
fourth_from_last = my_tuple[-4]

# Printing the results
print("4th Element:", fourth_element)
print("4th Element from the Last:", fourth_from_last)
```

```
4th Element: 4
4th Element from the Last: 6
```

```python
#Q7. Write a Python program to check whether an element exists within a tuple?

# Sample tuple
my_tuple = (1, 2, 3, 4, 5)
```

```python
# Element to check
element_to_check = 3

# Checking if the element exists in the tuple
if element_to_check in my_tuple:
    print(f"The element {element_to_check} exists in the tuple.")
else:
    print(f"The element {element_to_check} does not exist in the
tuple.")
```

```
The element 3 exists in the tuple.
```

```python
#Q8. Write a Python program to remove an item from a tuple?

# Original tuple
original_tuple = (1, 2, 3, 4, 5)

# Item to remove
item_to_remove = 3

# Create a new tuple without the item to remove
new_tuple = tuple(item for item in original_tuple if item !=
item_to_remove)

# Printing the new tuple
print("New Tuple:", new_tuple)
```

```
New Tuple: (1, 2, 4, 5)
```

```python
#Q9. Write a Python program to slice a tuple?

# Sample tuple
my_tuple = (1, 2, 3, 4, 5)

# Slicing the tuple from index 1 to index 4 (excluding index 4)
sliced_tuple = my_tuple[1:4]

# Printing the sliced tuple
print("Sliced Tuple:", sliced_tuple)
```

```
Sliced Tuple: (2, 3, 4)
```

```python
#Q10. Write a Python program to find the length of a tuple?

# Sample tuple
my_tuple = (1, 2, 3, 4, 5)

# Finding the length of the tuple
tuple_length = len(my_tuple)

# Printing the length of the tuple
print("Length of the tuple:", tuple_length)
```

```
Length of the tuple: 5
```

```
#Q11. Write a Python program to print a tuple with string formatting
Sample tuple : (100, 200, 300)?

# Sample tuple
sample_tuple = (100, 200, 300)

# Printing the tuple with string formatting using f-string
print(f"Sample tuple: {sample_tuple}")
```

```
Sample tuple: (100, 200, 300)
```