DATE: 12/03/2024

Name	:	Junaid Asif
Roll No	:	BSAI-144
Class	:	BSAI ~~3~~ 4th -Sem
Submitted To	:	Ms. Anab Batool

{BSAI-144 ASSIGNMENT #01}

Qs.1 Differentiate between mutable and immutable data structures in python.

Ans

	Mutable	Immutable
i.	Mutable data structures allow for modification after creation. This means you can change the values of individual elements or the structure itself.	Immutable data structures once created, cannot be changed. Any ~~attempt~~ attempt to modify their values results in the creation of a new object.
ii.	Examples of mutable data structures in ~~p~~ python include lists, dictionaries and sets.	Examples of immutable data structures in python include tuples, strings and integers.
iii.	Mutable objects can be modified in-place without creating a new object. This can lead to more efficient memory usage.	Immutable objects are hashable, which means they can be used ~~as~~ as keys in dictionaries and elements in sets.

DATE: __/__/__

Question # 02

Ans **Nested List:-**

Nested lists in python are lists that contain other lists as elements. They are used to represent two-dimensional arrays or ~~a~~ matrices, where each element of the outer list represents a row and each element of the inner list represents an element with in that row.

Creating an Empty 2D list:-

We can create an empty 2D list using list comprehension or by multiplying an empty list.

Initializing a 2D list:-

To initialize a 2D list with specific values, we can use nested list comprehensions or loops.

Accessing elements from a 2D list:-

You can access elements from a 2D list using indexing. The first ~~in~~ index represents the row number, and the second index represents the column number.

DATE: __/__/____

Available Methods for 2D lists:

1. append()	4. pop()	7. count()
2. extend()	5. remove()	8. copy()
3. insert()	6. index()	9. clear()
10. reverse()	11. sort()	

Question # 03

Ans Enumerate () function:-

The "enumerate ()" function in python is a built-in-function that allows you to iterate over a sequence (such as list, typle or string) ~~with also~~ while also keeping track of the index of each item in the sequence. It returns an enumerate object, which yeild paris of indices and corresponding values from the iterable.

Example:

```
fruits = ["apple", "banana", "orange"]
for index, fruit in enumerate(fruits):
        print(index, fruit)
```

DATE : __/__/__

Question # 04

Ans Output of Given Codes:-

3·1 Output:

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Explanation:

The code generates a list of squares of numbers ranging from 1 to 10 using list comprehension and then prints the resulting list.

3·2 Output:

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30,
 32, 34, 36, 38, 40]
```

Explanation:

The code ~~genet~~ generates a list of even numbers by multiplying each number from 1 to 20 by 2 using list comprehension.

3·3 Output:

```
['HELLO', 'WORLD', 'PYTHON']
```

Explanation:

The code iterates over each word in the 'words' list, converts each word to uppercase using the 'upper()' method, and stores the uppercase words in a new list called 'uppercase words'. Finally, it prints the 'uppercase_words' list.

DATE: 12/03/2024

3.4 **Output:**

```
[4, 16, 36, 64, 100, 144, 196, 256, 324, 400]
```

Explanation:

~~The~~ This code generates and prints a list of squares for even numbers ranging from 1 to 20.

3.5 **Output:**

```
[[1, 1], [2, 4], [3, 9], [4, 16], [5, 25]]
```

Explanation:

The code uses list comprehension to create a list containing pairs of numbers and their squares from 1 to 5.

3.6 **Output:**

```
[5, 6, 6, 4]
```

Explanation:

The code utilizes: list comprehension to generate a list of lengths of strings from the original list and then print it out.

3.7 **Output:**

```
[3, 4, 9, 8, 15, 12]
```

Explanation:

This list comprehension multiplies even numbers in a list by 2 and odd numbers by 3.

DATE : ___/___/______

3.8 **Output:**

```
[1, 2, 3, 4, 5]
```

Explanation:

This code uses a list comprehension to convert an integer to a list of digits by extracting them from the string representation.

3.9 **Output:**

```
['Positive', 'Positive', 'Zero', 'Negative', 'Positive', 'Negative']
```

Explanation:

The code categorizes numbers in a list as positive, zero, or negative using list comprehension.

3.10 **Output:**

```
[1, 2, 3, 4, 5]
```

Explanation:

List comprehension finds unique elements in ~~the~~ original list by adding only those not seen before the ~~c~~ current index.

3.11 **Output:**

```
['apple', 'date', 'fig']

['banana', 'cherry']
```

Explanation:

The code sorts words by length into short words (<=5 characters) and long words (>5 characters).

DATE : __/__/____

3.12 ~~Co~~ Output:

```
[32.0, 50.0, 68.0, 86.0, ~~40.0~~ 104.0]
```

Explanation:

This list comprehension converts celcius temperatures to fahrenheit and stores them in a new list.