9·2·23

# APP ASSIGNMENT - 1

NAME: SHAURYA SRINET
REG NO: RA2111032010006
SECTION: T2.
DEPT: NWC (CSE IOT)

Q1. In python, lists & arrays are manipulated through "slicing", "dicing" and "splicing" methods.

→ **Slicing:**

By providing start and end indices, slicing is a technique used to remove a piece of a list or array.
List [start: end] is the syntax for slicing in Python, where start is the index of the first element to be included in the slice and end is the index of the first element to be ~~executed~~ excluded from the slice.
For instance:

```
numbers = [0,1,2,3,4,5,6,7,8,9]
print(numbers[2:7])
output → [2,3,4,5,6]
```

→ **Dicing:**

Using a Step Value, dicing is analogous to slicing. It enables us to take the nth element from a list or array & extract it
List [start: end: step] is the syntax for dicing in Python.
Here step is the amount of elements to skip between slices.
For instance:

```
numbers = [0,1,2,3,4,5,6,7,8,9]
print (numbers[::2])
output → [0, 2, 4, 6, 8]
```

→ Splicing:

Splicing is a method for adding, removing or replacing elements in a list or array. You can achieve this by giving a portion of the list new values.

For instance:

```
numbers = [0,1,2,3,4,5,6,7,8,9]
numbers[2:5] = [20,30,40]
print(numbers)

output → [0,1,20,30,40,5,6,7,8,9]
```

Q2:

→ Structured Programming Technique:

```
import random
def lottery_structured():
    lottery = random.randint(100,999)
    user_input = int(input("Enter a three digit number:"))
    lottery_str = str(lottery)
    user_input_str = str(user_input)
    if lottery_str == user_input_str:
        print("You won $10,000!")
    else:
        match = True
        for i in range(3):
            if lottery_str[i] != user_input_str[i]:
                match = False
                break
        if match:
            print("You won $3,000!")
        else:
            match = False
            for i in range(3):
```

```
            if lottery-str.find(user-input-str[i])!= -1 :
                match = True
                break
        if match:
            print(" You won $1,000!")
        else:
            print("Sorry, you lost")
lottery_structured()
```

Output:

```
Enter a three-digit number : 696
You won $1,000!
```

→ Procedural Programming Technique:

```
import random
def lottery-procedural():
        lottery=random.randint(100,999)
        user-input = int(input("Enter a three-digit number:"))
        lottery-str= str(lottery)
        user-input-str = str(user-input)
        if lottery-str == user-input-str:
            print("You won $10,000!")
        elif (lottery-str[0]== user-input-str[0]) and
            (lottery-str[1]== user-input-str[1] and
            (lottery-str[2] == user-input-str[2]):
            print("You won $3,000!")
        elif (lottery-str-find(user-input-str[0])!= -1) or
            (lottery-str.find(user-input-str[1])!= 1) or
            (lottery-str.find(user-input-str[2])!= 1):
            print("You won $1,000!)
        else:
            print("Sorry, you lost")
lottery-procedural()
```

Output:

```
Enter a three digit number: 345
You won $1,000!
```

→ The difference between the two techniques is that the Structured Programming Technique uses a series of 'if' statements to check each condition in a logical & organised manner, ~~while~~ it uses functions to encapsulate logic, it can be more efficient as it uses fewer function calls & makes it easier to reuse code. However, procedural programming implementation uses separate functions to perform each specific task. This can make the procedural implementation easier to understand, modify & maintain as each function has a well-defined purpose.

——— X ———