WEEK#5

Objectives

- To generate random data and perform operations on it.
- To manipulate numpy arrays and perform sorting operations.

Outcomes

After completing this week, the students would be able to:

- Generate random data like OTPs and passwords.
- Perform operations on numpy arrays and sort them.

Problems

1. Write a program to generate a 6-digit random secure OTP.

```
import numpy as np

print(np.random.randint(10000,99999))
```

```
29835
PS C:\Users\hp\Documents\Suhel\3rd - Sem\Lab-Manual-III>
```

2. Write a program to pick a random character from a user-supplied string.

```
# Choosing a random character from a string

import random as rd

my_string = input("Enter a string: ")

str_len = len(my_string)

index_flt = rd.random() * str_len

index_int = int(index_flt)

print("Random string is: ", my_string[index_int])
```

```
Enter a string: this is a string
Random string is:  s
PS C:\Users\hp\Documents\Suhel\3rd - Sem\Lab-Manual-III>
```

3. Write a program to generate a random password that meets the following conditions: a. Password length must be 10 characters long. b. It must contain at least 2 uppercase letters, 1 digit, and 1 special symbol.

```
# Generating a random password

import random

import string
```

```
def generate_random_password():
    # Define character sets
    uppercase_letters = string.ascii_uppercase
    lowercase_letters = string.ascii_lowercase
    digits = string.digits
    special_symbols = string.punctuation
    print(uppercase_letters)
    # Combine all characters
    all_characters = uppercase_letters + lowercase_letters + digits + special_symbols
    # Generate a random password
    password = ''.join(random.choice(all_characters) for _ in range(10))
    return password
# Generate and print the password
random_password = generate_random_password()
print(f"Random password: {random_password}")
```

```
Random password: >DxV$DSu#)
PS C:\Users\hp\Documents\Suhel\3rd - Sem\Lab-Manual-III>
```

4. Given two lists of numbers, write a program to create a new list containing odd numbers from the first list and even numbers from the second list.

```
# List problem
list = [] # new list
list1 = [11, 22, 33, 45, 65, 78, 99, 101] # list 1
list2 = [11001, 2021, 330, 450, 655, 7658, 909, 205] # list 2
for i in list1:
    if i % 2 !=0:
        list.append(i)
for i in list2:
    if i % 2==0:
        list.append(i)
print("New List is : ",list)
```

```
New List is :  [11, 33, 45, 65, 99, 101, 330, 450, 7658]
PS C:\Users\hp\Documents\Suhel\3rd - Sem\Lab-Manual-III>
```

5. Write a program to create a numpy array and return an array of odd rows and even columns from the numpy array.

```
import numpy as np

def create_and_extract_array():
    # Create a random 5x5 NumPy array (you can adjust the size as needed)
    rows, cols = 5, 5
    random_array = np.random.randint(1, 100, size=(rows, cols))
    print(random_array)
    # Extract odd rows and even columns
    odd_rows = random_array[::2]  # Select every 2nd row (odd rows)
    even_columns = random_array[:, 1::2]  # Select every 2nd column (even columns)
    return odd_rows, even_columns
# Example usage
odd_rows_result, even_columns_result = create_and_extract_array()
print("Odd rows:")
print(odd_rows_result)
print("\nEven columns:")
print(even_columns_result)
```

```
[[14 55 74 47  2]
 [84 88  5 48 75]
 [42 77 97 29  2]
 [61  6 36 44 80]
 [99 68  7 72 45]]
Odd rows:
[[14 55 74 47  2]
 [42 77 97 29  2]
 [99 68  7 72 45]]

Even columns:
[[55 47]
 [88 48]
 [77 29]
 [ 6 44]
 [68 72]]
PS C:\Users\hp\Documents\Suhel\3rd - Sem\Lab-Manual-III>
```

6. Write a program to create a numpy array and sort it as per the following cases:

a. Case 1: Sort the array by the second row.

b. Case 2: Sort the array by the second column.

```
import numpy as np

# Create a sample 2D NumPy array

array = np.array([[5, 2, 9],
```

```
                [1, 6, 4],
                [3, 8, 7]])
print("Original Array:")
print(array)
# Case 1: Sort the array by the second row
# Get the indices that would sort the second row
sorted_indices_row = np.argsort(array[1])
# Use those indices to sort the entire array
sorted_array_row = array[:, sorted_indices_row]
print("\nSorted Array by the second row:")
print(sorted_array_row)
# Case 2: Sort the array by the second column
# Get the indices that would sort the second column
sorted_indices_col = np.argsort(array[:, 1])
# Use those indices to sort the entire array
sorted_array_col = array[sorted_indices_col]
print("\nSorted Array by the second column:")
print(sorted_array_col)
```

```
Original Array:
[[5 2 9]
 [1 6 4]
 [3 8 7]]

Sorted Array by the second row:
[[5 9 2]
 [1 4 6]
 [3 7 8]]

Sorted Array by the second column:
[[5 2 9]
 [1 6 4]
 [3 8 7]]
PS C:\Users\hp\Documents\Suhel\3rd - Sem\Lab-Manual-III>
```