This notebook contains a collection of Python programs (Program 71 to Program 80) covering **dictionary sorting**, **list/array generation**, **regex validation**, and **generator functions**. These exercises are great for learning intermediate Python logic and flow control.

The noteboook can be accessed in the (https://amitvsuryavanshi04/SIC_programming_and_coding)

Program 71 to check order of character in string using OrderedDict()

```
from collections import OrderedDict
def check_order(string, reference):
  # Create OrderedDicts for both strings
  string_dict = OrderedDict.fromkeys(string)
  reference_dict = OrderedDict.fromkeys(reference)
  # Check if the OrderedDict for the string matches the OrderedDict f
  return string_dict == reference_dict
# Input strings
input_string = "hello world"
reference_string = "helo wrd"
# Check if the order of characters in input_string matches reference_st
if check_order(input_string, reference_string):
  print("The order of characters in the input string matches the reference_st")
else:
  print("The order of characters in the input string does not match the reference_st")
```

```
The order of characters in the input string matches the reference_st
```

Program 72 python dictonaries by key or value

```
# Sort by Keys:
sample_dict = {'apple': 3, 'banana': 1, 'cherry': 2, 'date': 4}
sorted_dict_by_keys = dict(sorted(sample_dict.items()))
print("Sorted by keys:")
for key, value in sorted_dict_by_keys.items():
  print(f"{key}: {value}")
```

```
Sorted by keys:
apple: 3
banana: 1
```

```
cherry: 2
date: 4
```

```
# Sort by values
sample_dict = {'apple': 3, 'banana': 1, 'cherry': 2, 'date': 4}
sorted_dict_by_values = dict(sorted(sample_dict.items(), key=lambda item: item[1]))

print("Sorted by values:")
for key, value in sorted_dict_by_values.items():
    print(f"{key}: {value}")
```

```
Sorted by values:
banana: 1
cherry: 2
apple: 3
date: 4
```

Program 73 Write a program that calculates and prints the value according to the given formula: Following are the fixed values of C and H: C is 50. H is 30. D is the variable whose values should be input to your program in a commaseparated sequence

```
import math
# Fixed values
C = 50
H = 30
# Function to calculate Q
def calculate_Q(D):
  return int(math.sqrt((2 * C * D) / H))
# Input comma-separated sequence of D values
input_sequence = input("Enter comma-separated values of D: ")
D_values = input_sequence.split(',')
# Calculate and print Q for each D value
result = [calculate_Q(int(D)) for D in D_values]
print(','.join(map(str, result)))
```

```
Enter comma-separated values of D: 100,150,1809
18,22,77
```

Program 74 Write a program which takes 2 digits, X,Y as input and generates a 2-dimensional array. The element value in the i-th row and j-th column of the array should be i*j. Note: i=0,1.., X-1; j=0,1,¡Y-1. Example Suppose the following inputs are given to the program: 3,5 Then, the output of the program should be: [[0, 0, 0, 0, 0], [0, 1, 2, 3, 4], [0, 2, 4, 6, 8]]

```
# Input two digits, X and Y
X, Y = map(int, input("Enter two digits (X, Y): ").split(','))
# Initialize a 2D array filled with zeros
array = [[0 for j in range(Y)] for i in range(X)]
# Fill the array with values i * j
for i in range(X):
  for j in range(Y):
    array[i][j] = i * j
# Print the 2D array
for row in array:
  print(row)
```

```
Enter two digits (X, Y): 3,5
[0, 0, 0, 0, 0]
[0, 1, 2, 3, 4]
[0, 2, 4, 6, 8]
```

Program 75 Write a program that accepts a comma separated sequence of words as input and prints the words in a comma-separated sequence after sorting them alphabetically.

Suppose the following input is supplied to the program: without,hello,bag,world

Then, the output should be:

bag,hello,without,world

```
# Accept input from the user
input_sequence = input("Enter a comma-separated sequence of words: ")
# Split the input into a list of words
words = input_sequence.split(',')
# Sort the words alphabetically
sorted_words = sorted(words)
# Join the sorted words into a comma-separated sequence
```

```
sorted_sequence = ','.join(sorted_words)
# Print the sorted sequence
print("Sorted words:", sorted_sequence)
```

```
Enter a comma-separated sequence of words: amit,zeb,vittal,yashoda
Sorted words: amit,vittal,yashoda,zeb
```

Program 76 Write a program that accepts a sequence of whitespace separated words as input and prints the words after removing all duplicate words and sorting them alphanumerically.

Suppose the following input is supplied to the program:

hello world and practice makes perfect and hello world again

Then, the output should be: again and hello makes perfect practice world

```
# Accept input from the user
input_sequence = input("Enter a sequence of whitespace-separated words:")
# Split the input into words and convert it into a set to remove duplic
words = set(input_sequence.split())
# Sort the unique words alphanumerically
sorted_words = sorted(words)
# Join the sorted words into a string with whitespace separation
result = ' '.join(sorted_words)
# Print the result
print("Result:", result)
```

```
Enter a sequence of whitespace-separated words:hello world this is amit suryavanshi
Result: amit hello is suryavanshi this world
```

Program 78 Write a program that accepts a sentence and calculate the number of letters and digits. Suppose the following input is supplied to the program: hello world! 123

Then, the output should be:

LETTERS 10

DIGITS 3

```
# Accept input from the user
sentence = input("Enter a sentence: ")
# Initialize counters for letters and digits
letter_count = 0
digit_count = 0
# Iterate through each character in the sentence
for char in sentence:
  if char.isalpha():
    letter_count += 1
  elif char.isdigit():
    digit_count += 1
# Print the results
print("LETTERS", letter_count)
print("DIGITS", digit_count)
```

```
Enter a sentence: AMIT SURYAVANSHI 10072002
LETTERS 15
DIGITS 8
```

Program 79

🔐 Password Validation Program

This program checks the validity of transaction passwords entered by users based on the following criteria:

✅ Password Criteria:

1. Must contain **at least one lowercase letter** (`a-z`)
2. Must contain **at least one uppercase letter** (`A-Z`)
3. Must contain **at least one digit** (`0-9`)
4. Must contain **at least one special character** from the set: `$`, `#`, `@`
5. Must be **6 to 12 characters** in length (inclusive)

📄 Input Format:

- A sequence of **comma-separated passwords** entered by the user.

💡 Example:

Input:

```
import re

# Function to check if a password is valid
def is_valid_password(password):
    # Check the length of the password
    if 6 <= len(password) <= 12:
        # Check if the password matches all the criteria using regex
        if re.search(r"[a-z]", password) and \
           re.search(r"[A-Z]", password) and \
           re.search(r"[0-9]", password) and \
           re.search(r"[$#@]", password):
            return True
    return False

# Accept input from the user as comma-separated passwords
passwords = input("Enter passwords separated by commas: ").split(',')

# Initialize a list to store valid passwords
valid_passwords = []

# Iterate through the passwords and check their validity
for psw in passwords:
    psw = psw.strip()  # Remove extra whitespace
    if is_valid_password(psw):
        valid_passwords.append(psw)

# Print the valid passwords separated by commas
print(','.join(valid_passwords))
```

```
Enter passwords separated by commas: abc123,@V$rt
```

Program 80 Define a class with a generator which can iterate the numbers, which are divisible by 7, between a given range 0 and n

```
class DivisibleBySeven:
```

```
class DivisibleBySeven:
    def __init__(self, n):
        self.n = n

    def generate_divisible_by_seven(self):
        for num in range(self.n + 1):
            if num % 7 == 0:
                yield num

# Take input from the user
n = int(input("Enter your desired range: "))

# Create an instance and call the generator method
divisible_by_seven_generator = DivisibleBySeven(n).generate_divisible_by_seven()

# Print all numbers divisible by 7 in the range
print("Numbers divisible by 7 from 0 to", n, ":")
for num in divisible_by_seven_generator:
    print(num)
```

```
Enter your desired range: 100
Numbers divisible by 7 from 0 to 100 :
0
7
14
21
28
35
42
49
56
63
70
77
84
91
98
```