**WEB DEVELOPMENT USING PYTHON**

**LAB MANUAL**

**Year : 2024 - 2025**

**Course Code : 23SDCA02**

**Regulations : Y23**

**Semester : III**

**Branch : BCA**

**Prepared by**

**Mr.D VENKATESH,ASSISTNT PROFESSOR**



**Koneru Lakshmaiah Education Foundation**

**(Deemed to be University)**

**Bowrampet, Gandimysamma Road, Hyderabad-500043, Telangana ,India.**



1. **Program Outcomes:**

| **Program Outcomes (POs)** | |
| --- | --- |
| **PO-1** |  |
| **PO-2** |  |
| **PO-3** |  |
| **PO-4** |  |
| **PO-5** |  |
| **PO-6** |  |
| **PO-7** |  |
| **PO-8** |  |
| **PO-9** |  |
| **PO-10** |  |
| **PO-11** |  |
| **PO-12** |  |

**2. PROGRAM SPECIFIC OUTCOMES**

|  |  |
| --- | --- |
| **PROGRAM SPECIFIC OUTCOMES (PSO's)** | |
| **PSO-1** |  |
| **PSO-2** |  |
| **PSO-3** |  |

**3. ATTAINMENT OF PROGRAM OUTCOMES AND PROGRAM SPECIFIC OUTCOMES:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Experiment** | **Program Outcomes Attained** | **Program Specific Outcomes Attained** |
| **WEEK-l** |  | **PO-2,PO-3** | **PSO-1** |
| Valid Parentheses:Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.An input string is valid if: a. Open brackets must be closed by the same type of brackets. b. Open brackets must be closed in the correct order |
| **WEEK-2** |  | **PO-2,PO-3** | **PSO-1** |
| Binary Tree Inorder Traversal :Given the root of a binary tree, return the inorder traversal of its nodes' values. |
| **WEEK-3** |  | **PO-2** |  |
| Roman to Integer :Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M. Given a roman numeral, convert it to an integer. |
| **WEEK-4** |  | **PO-3** | **PSO-1** |
| Program Creating Classes/Objects and Inheritance |
| **WEEK-5** |  | **PO-3** | **PSO-1** |
| Program on Arithmetic Operations Using Modules in Python |
| **WEEK-6** |  | **PO-3** | **PSO-1** |
| Program on File Handling and Exception Handling Concepts |
| **WEEK-7** |  | **PO-2,PO-3** |  |
| Number of 1 Bits : Write a function that takes an unsigned integer and returns the number of '1' bits it has (also known as the Hamming weight). |
| **WEEK-8** |  | **PO-2,PO-3** |  |
| Write an algorithm to determine if a number n is happy. A happy number is a number defined by the following process: Starting with any positive integer, replace the number by the sum of the squares of its digits.Repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1.Those numbers for which this process ends in 1 are happy.Return true if n is a happy number, and false if not |
| **WEEK-9** |  | **PO-3** |  |
| Contains Duplicate Given an integer array nums, return true if any value appears at least twice in the array, and return false if every element is distinct |
| **WEEK-l0** |  | **PO-3,PO-12** | **PSO-1** |
| Reverse String Write a function that reverses a string. The input string is given as an array of characters s.You must do this by modifying the input array in-place with O(1) extra memory. |
| **WEEK-l1** |  | **PO-3** | **PSO-1** |
| Add Digits :Given an integer num, repeatedly add all its digits until theresult has only one digit, and return it. |
| **WEEK-l2** |  | **PO-3** | **PSO-1** |
|  |
| Create Program on JSON Application: User Data Management and Regular Expressions Application: Email Validator |
| **WEEK-l3** |  | **PO-3** | **PSO-1** |
| Program on Libraries SciPy Application: Compute Definite Integral and NumPy Application: Statistical Analysis & Matplotlib Application: Plotting Data |
| **WEEK-l4** | Make a two-player Rock-Paper-Scissors game. (Hint: Ask for player plays (using input), compare them, print out a message of congratulations to the winner, and ask if the players want to start a new game) |  |  |
| **WEEK-l5** | Write a password generator in Python. Be creative with how you generate passwords - strong passwords have a mix of lowercase letters, uppercase letters, numbers, and symbols. |  |  |
| **WEEK-l6** | Write a program (function!) that takes a list and returns a new list that contains all the elements of the first list minus all the duplicates. |  |  |
| **WEEK-l7** | Program to perform Python Connectivity with Databases using MySQL |  |  |
| **WEEK-l8** | MongoDB queries to create,project,insert,update and delete in a database. using MongoDB CRUD Operations |  |  |
| **WEEK-l9** | Write a program to print data in a web page using Django framework |  |  |
| **WEEK-20** | Write a program to demonstrate the screen scrapping application Programs in python. |  |  |
| **WEEK-21** | Write a program to demonstrate web crawling application in Python. |  |  |
| **WEEK-22** | Program to Creating a Simple Blog Application using Django |  |  |
| **WEEK-23** | Program Registration form student details and store into database perform Validations |  |  |
| **WEEK-24** | Building some Django Applications of Python Programming Language. |  |  |

**4. MAPPING COURSE OBJECTIVES LEADING TO THE ACHIEVEMENT OF PROGRAM OUTCOMES:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Course Objectives** | **Program Outcomes** | | | | | | | | | | | | **Program Specific Outcomes** | | |
| **PO1** | **PO2** | **PO3** | **PO4** | **PO5** | **PO6** | **PO7** | **PO8** | **PO9** | **PO**  **10** | **PO**  **11** | **PO**  **12** | **PSO1** | **PSO2** | **PSO3** |
| **I** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **II** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **III** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| **IV** |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**5. SYLLABUS:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | |
| **I Semester: BCA/** | | | | | | | | | | | |
| **Course Code** | | | | **Category** | **Hours / Week** | | | | **Maximum Marks** | | |
|  | | | | **Core** | **L** | **T** | **P** | **S** | **InSem** | **Extenal** | **Total** |
|  |  |  |  |  |  |  |
| **Contact Classes: Nil** | | | | **Tutorial Classes: Nil** | **Practical Classes:** | | | | **Total Classes:** | | |
| **OBJECTIVES:**  **The course should enable the students to:**  Learn how to analyze a problem and design the solution for the problem.   1. Design and implement efficient algorithms for a specified application. 2. Strengthen the ability to identify and apply the suitable algorithm for the given real world problem. | | | | | | | | | | | |
| **LIST OF EXPERIMENTS** | | | | | | | | | | | |
| **WEEK-1** |  | | | | | | | | | | |
| Valid Parentheses :Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.An input string is valid if: a. Open brackets must be closed by the same type of brackets. b. Open brackets must be closed in the correct order. | | | | | | | | | | | |
| **WEEK-2** |  | | | | | | | | | | |
| Binary Tree Inorder Traversal :Given the root of a binary tree, return the | | | | | | | | | | | |
| **WEEK-3** |  | | | | | | | | | | |
| Roman to Integer :Roman numerals are represented by seven different symbols: I, V, X, L, C, D and M. Given a roman numeral, convert it to an integer. | | | | | | | | | | | |
| **WEEK-4** |  | | | | | | | | | | |
| Program Creating Classes/Objects and Inheritance | | | | | | | | | | | |
| **WEEK-5** |  | | | | | | | | | | |
| Program on Arithmetic Operations Using Modules in Python | | | | | | | | | | | |
| **WEEK-6** |  | | | | | | | | | | |
| Find Minimum Cost Spanning Tree of a given undirected graph using Kruskal’s algorithm. | | | | | | | | | | | |
| **WEEK-7** |  | | | | | | | | | | |
| Number of 1 Bits : Write a function that takes an unsigned integer and returns the number of '1' bits it has (also known as the Hamming weight). | | | | | | | | | | | |
| **WEEK-8** | | |  | | | | | | | | |
| Write an algorithm to determine if a number n is happy. A happy number is a number defined by the following process: Starting with any positive integer, replace the number by the sum of the squares of its digits.Repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1.Those numbers for which this process ends in 1 are happy.Return true if n is a happy number, and false if not | | | | | | | | | | | |
| **WEEK-9** | | |  | | | | | | | | |
| Contains Duplicate Given an integer array nums, return true if any value appears at least twice in the array, and return false if every element is distinct | | | | | | | | | | | |
| **WEEK-10** | |  | | | | | | | | | |
| Reverse String Write a function that reverses a string. The input string is given as an array of characters s.You must do this by modifying the input array in-place with O(1) extra memory. | | | | | | | | | | | |
| **WEEK-11** | |  | | | | | | | | | |
| Add Digits :Given an integer num, repeatedly add all its digits until the result has only one digit, and return it. | | | | | | | | | | | |
| **WEEK-12** | | |  | | | | | | | | |
| Create Program on JSON Application: User Data Management and Regular Expressions Application: Email Validator | | | | | | | | | | | |
| **WEEK-13** | | | Program on Libraries SciPy Application: Compute Definite Integral and NumPy Application: Statistical Analysis & Matplotlib Application: Plotting Data | | | | | | | | |
| **WEEK-14** | | | Make a two-player Rock-Paper-Scissors game. (Hint: Ask for player plays (using input), compare them, print out a message of congratulations to the winner, and ask if the players want to start a new game) | | | | | | | | |
| **WEEK-15** | | | Write a password generator in Python. Be creative with how you generate passwords - strong passwords have a mix of lowercase letters, uppercase letters, numbers, and symbols. | | | | | | | | |
| **WEEK-16** | | | Write a program (function!) that takes a list and returns a new list that contains all the elements of the first list minus all the duplicates. | | | | | | | | |
| **WEEK-17** | | | Program to perform Python Connectivity with Databases using MySQL | | | | | | | | |
| **WEEK-18** | | | MongoDB queries to create,project,insert,update and delete in a database. using MongoDB CRUD Operations | | | | | | | | |
| **WEEK-19** | | | Write a program to print data in a web page using Django framework | | | | | | | | |
| **WEEK-20** | | | Write a program to demonstrate the screen scrapping application Programs in python. | | | | | | | | |
| **WEEK-21** | | | Write a program to demonstrate web crawling application in Python. | | | | | | | | |
| **WEEK-22** | | | Program to Creating a Simple Blog Application using Django | | | | | | | | |
| **WEEK-23** | | | Program Registration form student details and store into database perform Validations | | | | | | | | |
| **WEEK-24** | | | Building some Django Applications of Python Programming Language. | | | | | | | | |

**6. INDEX:**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Experiment** | **Page No** |
| **1** | **VALID PARANTHESIS** |  |
| **2** | **BINARY TREE INORDER TRAVERSAL** |  |
| **3** | **ROMAN TO INTEGER** |  |
| **4** | **CLASSES/ OBJECTS AND INHERITANCE** |  |
| **5** | **ARITHMETIC OPERATIONS USING MODULES** |  |
| **6** | **FILE HANDLING AND EXCEPTION HANDLING** |  |
| **7** | **NO OF ‘1’ BITS** |  |
| **8** | **HAPPY NUMBER** |  |
| **9** | **DUPLICATE** |  |
| **10** | **REVERSE STRING** |  |
| **11** | **ADD DIGITS** |  |
| **12** | **JSON APPLICATION PROGRAM** |  |
| **13** | **PROGRAM ON LIBRARIES** |  |
| **14** | **ROCK- PAPER-SCISSORS GAME** |  |
| **15** | **PASSWORD GENERATOR** |  |
| **16** | **FUNCTION TO RETURN A NEW LIST** |  |
| **17** | **PYTHON CONNECTIVITY WITH DATABASES USING MYSQL** |  |
| **18** | **MONGODB CRUD OPERATIONS** |  |
| **19** | **DATA IN WEBPAGE USING DJANGO** |  |
| **20** | **SCREEN SCRAPPING APPLICATION IN PYTHON** |  |
| **21** | **WEB CRAWLING APPLICATION IN PYTHON** |  |
| **22** | **SIMPLE BLOG APPLICATION USING DJANGO** |  |
| **23** | **REGISTRATION FORM** |  |
| **24** | **DJANGO APPLICATIONS** |  |

**WEEK-1**

**VALID PARANTHESIS**

* 1. **OBJECTIVE:**
  2. **RESOURCES:**

* 1. **PROGRAM LOGIC:**
  2. **PROCEDURE:**

1. **EXAMPLE:**
2. **SOURCE CODE:**

ol=["[","{","("]

cl=["]","}",")"]

def check(st):

    stk=[]

    for i in st:

        if i in ol:

          stk.append(i)

        elifi in cl:

            p=cl.index(i)

            if len(stk)>=0 and (ol[p]==stk[len(stk)-1]):

                stk.pop()

            else:

                return ("Unbalanced:")

    if len(stk)==0:

         return("Balanced")

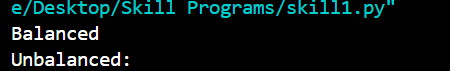
    else:

         return("Unbalanced")

print(check("[()]"))

print(check("[(]"))

* 1. **INPUT/ OUTPUT**

****

* 1. **LAB VIVA QUESTIONS:**

**WEEK-2**

**BINARY TREE INORDER TRAVERSAL**

**SOURCE CODE:**

rt1 = None

stack = []

#create node class

class TreeNode:

def \_\_init\_\_(self, val=0, left=None, right=None):

self.val = val

self.left = left

self.right = right

#Create a binary search Tree

def ctree(root, i):

if root == None:

root = TreeNode(i)

return root

else:

tmp = root

nd = TreeNode(i)

while tmp != None:

tmp1 = tmp

if i<tmp.val:

print(tmp.val)

tmp = tmp.left

else:

tmp = tmp.right

if i< tmp1.val:

tmp1.left = TreeNode(i)

else:

tmp1.right = TreeNode(i)

return root

#in-order traversal

def preorder(root):

if root != None:

preorder(root.left)

stack.append(root.val)

preorder(root.right)

return stack

#Pre-order traversal

def inorder(root):

if root != None:

inorder(root.left)

stack.append(root.val)

inorder(root.right)

return stack

#Post-order Traversal

def postorder(root):

if root != None:

postorder(root.left)

postorder(root.right)

stack.append(root.val)

return stack

lst1 = [1,2,3,4,5,6,7]

for i in lst1:

rt1 = ctree(rt1, i)

s1 =inorder(rt1)

print('inorder:',s1)

**INPUT/ OUTPUT:**

****

**VIVA QUESTIONS:**

**WEEK-3**

**ROMAN TO INTEGER**

**SOURCE CODE:**

def romanToInt(s):

       dict = {

        'I' : 1,

        'V' : 5,

        'X' : 10,

        'L' : 50,

        'C' : 100,

        'D' : 500,

        'M' : 1000

        }

       result  = 0

       tmp = 0

       i = 0

       while i<len(s):

           tmp = dict[s[i]]

           if (i +1) <len(s) and dict[s[i]] <dict[s[i + 1]]:

               tmp = dict[s[i + 1]] - dict[s[i]]

               i += 1

           i += 1

           result += tmp

       return (result)

st='XV'

x=romanToInt(st)

print(x)

**INPUT/ OUTPUT:**



**VIVA QUESTIONS:**

**WEEK-4**

**CLASSES/OBJECTS AND INHERITANCE**

**SOURCE CODE:**

class Animal:

    def sound(self):

        print("Some sound")

class Dog(Animal):

    def sound(self):

        print("Bark")

class Cat(Animal):

    def sound(self):

        print("Meow")

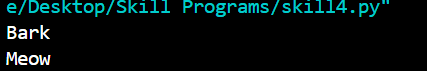
dog = Dog()

cat = Cat()

dog.sound()

cat.sound()

**INPUT/ OUTPUT:**



**VIVA QUESTIONS:**

**WEEK-5**

**ARITHMETIC OPERATIONS USING MODULES**

**SOURCE CODE:**

# arithmetic.py

def add(x, y):

return x + y

def subtract(x, y):

return x - y

def multiply(x, y):

return x \* y

def divide(x, y):

if y == 0:

return "Error! Division by zero."

return x / y

# main.py

# Import the arithmetic module

import arithmetic

# Test the arithmetic operations

a = 10

b = 5

print("Addition:", arithmetic.add(a, b)) # Output: 15

print("Subtraction:", arithmetic.subtract(a, b)) # Output: 5

print("Multiplication:", arithmetic.multiply(a, b)) # Output: 50

print("Division:", arithmetic.divide(a, b)) # Output: 2.0

**INPUT/ OUTPUT:**



**VIVA QUESTIONS:**

**WEEK-6**

**FILE HANDLING AND EXCEPTION HANDLING**

**SOURCE CODE:**

try:

a=int(input("enter a value"))

b=int(input("enter b value"))

c=a/b

print("a/b=",c)

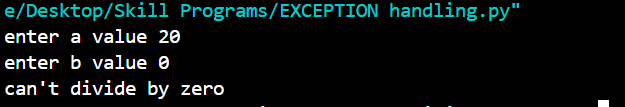
except:

print("can't divide by zero")

else:

print("hi i am else block")

**INPUT/ OUTPUT:**



**VIVA QUESTIONS:**

**WEEK-7**

**NO OF ‘1’ BITS**

**SOURCE CODE:**

def findones(n):

  count = 0

  a = 1

  bin\_num = 0

  while (n!=0):

    rem = n % 2

    n = n // 2

    if rem==1:

        count+=1

    b = (rem \* a)

    bin\_num = int(bin\_num + b)

    a = int(a \* 10)

  return bin\_num,count

#driver code

n1=int(input('Enter an integer : ' ))

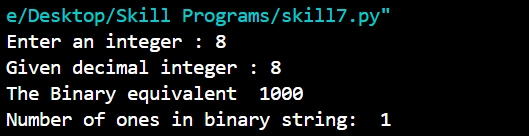
print(f'Given decimal integer : {n1}')

bin\_num,count=findones(n1)

print(f'The Binary equivalent  {bin\_num}')

print(f'Number of ones in binary string:  {count}')

**INPUT/ OUTPUT:**



**VIVA QUESTIONS:**

**WEEK-8**

**HAPPY NUMBER**

**SOURCE CODE:**

def happynumber(n):

   newn = list(str(n))

   s = 0

   loop\_chk = []

   while s != 1:

        for i in range(len(newn)):

            newn[i] = int(newn[i])\*\*2

            s += newn[i]

        newn = list(str(s))

        loop\_chk.append(str(newn))

        if loop\_chk.count(str(newn)) > 1:

            return False

        if s == 1:

            return True

        else:

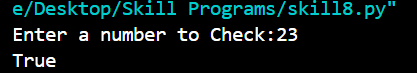
            s = 0

n1=int(input('Enter a number to Check:'))

x=happynumber(n1)

print(x)

**INPUT/ OUTPUT:**



**VIVA QUESTIONS:**

**WEEK-9**

**DUPLICATES**

**SOURCE CODE:**

def containduplicates(nums):

    n=set(nums)  #return the set of elements without duplicates. each ele is represented only once

    return False if len(n) == len(nums) else True

lst = []

# number of elements as input

n = int(input("Enter number of elements : "))

# iterating till the range

for i in range(0, n):

    ele = int(input())

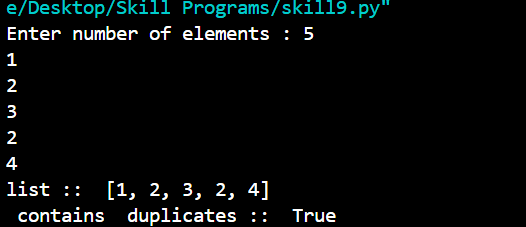
    # adding the element

    lst.append(ele)

print('list :: ',lst)

print(' contains  duplicates :: ', containduplicates(lst))

**INPUT/ OUTPUT:**



**VIVA QUESTIONS:**

**WEEK-10**

**REVERSE STRING**

**SOURCE CODE:**

def reverse\_string(s):

    left,right=0,len(s)-1

    while left<right:

        s[left],s[right]=s[right],s[left]

        left,right=left+1,right-1

s=['h','e','l','l','o']

reverse\_string(s)

print(s)

**INPUT/ OUTPUT:**



**VIVA QUESTIONS:**

**WEEK-11**

**ADD DIGITS**

**SOURCE CODE:**

def add\_digits(num):

if(num==0):

return 0

else:

return 1+(num-1)%9

num=38

result=add\_digits(num)

print(result)

**INPUT/ OUTPUT:**



**VIVA QUESTIONS:**

**WEEK-12**

**JSON APPLICATION PROGRAM**

**SOURCE CODE:**

import json

import re

# Email validator function

def is\_valid\_email(email):

    pattern = r'^[\w\.-]+@[\w\.-]+\.\w+$'

    if re.match(pattern, email):

        return True

    else:

        return False

# Function to save data to a JSON file

def save\_user\_data(data):

    with open('user\_data.json', 'w') as file:

        json.dump(data, file)

# Main program

name = input("Enter your name: ")

email = input("Enter your email: ")

if is\_valid\_email(email):

    user\_data = {"name": name, "email": email}

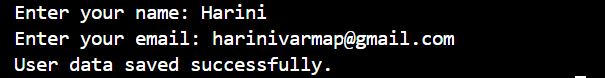
    save\_user\_data(user\_data)

    print("User data saved successfully.")

else:

    print("Invalid email address.")

**INPUT/ OUTPUT:**



**VIVA QUESTIONS:**

**WEEK-12**

**JSON APPLICATION PROGRAM**

**SOURCE CODE:**

import json

import re

# Email validator function

def is\_valid\_email(email):

    pattern = r'^[\w\.-]+@[\w\.-]+\.\w+$'

    if re.match(pattern, email):

        return True

    else:

        return False

# Function to save data to a JSON file

def save\_user\_data(data):

    with open('user\_data.json', 'w') as file:

        json.dump(data, file)

# Main program

name = input("Enter your name: ")

email = input("Enter your email: ")

if is\_valid\_email(email):

    user\_data = {"name": name, "email": email}

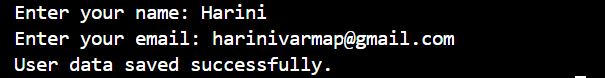
    save\_user\_data(user\_data)

    print("User data saved successfully.")

else:

    print("Invalid email address.")

**INPUT/ OUTPUT:**



**VIVA QUESTIONS:**

**WEEK-13**

**PROGRAM ON LIBRARIES**

**SOURCE CODE:**

import numpy as np

import matplotlib.pyplot as plt

from scipy.integrate import quad

# 1. SciPy: Compute Definite Integral of a function f(x) = x^2 from 0 to 1

def f(x):

    return x\*\*2

# Using scipy to compute the integral of f(x) from 0 to 1

result, error = quad(f, 0, 1)

print(f"Definite integral of x^2 from 0 to 1 is: {result}")

# 2. NumPy: Statistical Analysis

data = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

mean = np.mean(data)

std\_dev = np.std(data)

print(f"Mean of the data: {mean}")

print(f"Standard Deviation of the data: {std\_dev}")

# 3. Matplotlib: Plotting the data

plt.plot(data, label='Data')

plt.title('Simple Data Plot')

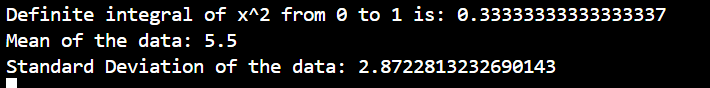
plt.xlabel('Index')

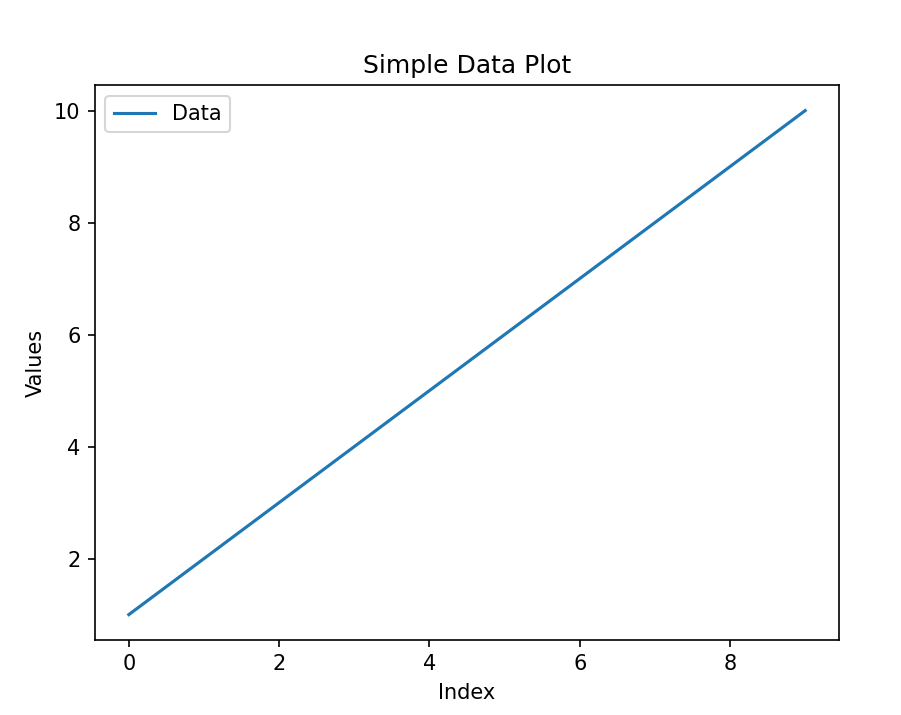
plt.ylabel('Values')

plt.legend()

plt.show()

**INPUT/ OUTPUT:**





**VIVA QUESTIONS:**

**WEEK-14**

**ROCK PAPER SCISSORS**

**SOURCE CODE:**

def rock\_paper\_scissors():

    while True:

        player1 = input("Player 1, enter your choice (rock, paper, or scissors): ").lower()

        player2 = input("Player 2, enter your choice (rock, paper, or scissors): ").lower()

        if player1 == player2:

            print("It's a tie!")

        elif (player1 == "rock" and player2 == "scissors") or \

             (player1 == "scissors" and player2 == "paper") or \

             (player1 == "paper" and player2 == "rock"):

            print("Player 1 wins!")

        else:

            print("Player 2 wins!")

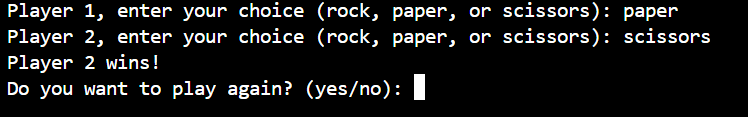
        play\_again = input("Do you want to play again? (yes/no): ").lower()

        if play\_again != "yes":

            break

rock\_paper\_scissors()

**INPUT/ OUTPUT:**



**VIVA QUESTIONS:**

**WEEK-15**

**PASSWORD GENERATOR**

**SOURCE CODE:**

import random

import string

def generate\_password(length):

    # Define possible characters

    lower = string.ascii\_lowercase

    upper = string.ascii\_uppercase

    digits = string.digits

    symbols = string.punctuation

    # Combine all character sets

    all\_chars = lower + upper + digits + symbols

    # Ensure password has at least one of each type of character

    password = [

        random.choice(lower),

        random.choice(upper),

        random.choice(digits),

        random.choice(symbols)

    ]

    # Fill the rest of the password with random characters

    password += random.choices(all\_chars, k=length-4)

    # Shuffle the password to make it more random

    random.shuffle(password)

    return ''.join(password)

# Input for desired password length

length = int(input("Enter the desired password length: "))

# Generate and print the password

print("Generated password:", generate\_password(length))

**INPUT/ OUTPUT:**



**VIVA QUESTIONS:**

**WEEK-16**

**FUNCTION TO RETURN A NEW LIST**

**SOURCE CODE:**

def remove\_duplicates(input\_list):

    return list(set(input\_list))

# Example usage:

my\_list = [1, 2, 2, 3, 4, 4, 5]

new\_list = remove\_duplicates(my\_list)

print("List without duplicates:", new\_list)

**INPUT/ OUTPUT:**



**VIVA QUESTIONS:**

**WEEK-17**

**PYTHON CONNECTIVITY WITH DATABASE USING MYSQL**

**SOURCE CODE:**

import mysql.connector  
  
mydb = mysql.connector.connect(  
  host="localhost",  
  user="*yourusername*",  
  password="*yourpassword*"  
)  
  
mycursor = mydb.cursor()  
  
mycursor.execute("CREATE DATABASE mydatabase")

Check if Database Exists

import mysql.connector  
  
  mydb = mysql.connector.connect(  
  host="localhost",  
  user="*yourusername*",  
  password="*yourpassword*"  
)  
  
mycursor = mydb.cursor()  
  
mycursor.execute("SHOW DATABASES")  
  
for x in mycursor:  
  print(x)

**INPUT/ OUTPUT:**

**VIVA QUESTIONS:**