PYTHON PROGRAMMING LABORATORY

Program 1

Aim: Introduce the Python fundamentals, data types, operators, flow control and exception handling in Python a) Write a python program to find the best of two test average marks out of three test's marks accepted from the user.

a) Write a python program to find the best of two test average marks out of three test's marks accepted from the user.

Algorithm:

Step 1: Request user input for the marks of three tests and convert to integers.
Step 2: Identify the lowest score.
Step 3: Compute the total of the top two scores by summing all and subtracting the lowest.
Step 4: Compute the average of the top two scores by dividing the total by 2.
Step 5: Output the average.
Step 6: Handle exceptions from invalid input by displaying an error message.

Code:

```
try:
    test1_marks = int(input("Enter Test 1 Marks: "))
    test2_marks = int(input("Enter Test 2 Marks: "))
    test3_marks = int(input("Enter Test 3 Marks: "))

    min_marks = min(test1_marks, test2_marks, test3_marks)
    best_two_sum = test1_marks + test2_marks + test3_marks - min_marks
    best_two_avg = best_two_sum / 2

    print("Average of Best Two Tests:", best_two_avg)
except ValueError:
    print("Invalid input! Please enter a number.")
```

Result: The program successfully calculates and displays the best of two test averages out of three tests.

PYTHON PROGRAMMING LABORATORY

b) Develop a Python program to check whether a given number is palindrome or not and also count the number of occurrences of each digit in the input number.

Algorithm:

Step 1: Accept user input for a number.
Step 2: Check if the number is the same when reversed:
   a) If it is, print that the number is a palindrome.
   b) If not, print that the number is not a palindrome.
Step 3: Create a set of unique digits from the number.
Step 4: Iterate through each unique digit.
Step 5: Count the occurrences of the current digit in the number using the count () method.
Step 6: Print the current digit and its count of occurrences.

Code:

```
try:
    num = int(input("Enter a number: "))
    if str(num) == str(num)[::-1]:
        print(num, "is a palindrome.")
    else:
        print(num, "is not a palindrome.")
    for digit in set(str(num)):
        print(f"Digit {digit} occurs {str(num).count(digit)} times.")
except Exception as e:
    print("An error occurred:", str(e))
```

Result: The program successfully checks for a palindrome and counts digit occurrences.

PYTHON PROGRAMMING LABORATORY

Program 2:

Aim: Demonstrating creation of functions, passing parameters and return values

a) Defined as a function F as Fn = Fn-1 + Fn-2. Write a Python program which accepts a value for N (where N >0) as input and pass this value to the function. Display suitable error message if the condition for input value is not followed.

Algorithm:

Step 1: Define a function fn that takes an integer n as input.

a) If n is equal to 1 or 2, return 1.
b) Otherwise, recursively call fn with n - 1 and n - 2 as input and add the results.

Step 2: Prompt the user to enter a positive integer greater than 0.
Step 3: Convert the user input to an integer.
Step 4: If the converted value is greater than 0, call fn with the converted value as input and print the result.
Step 5: Otherwise, raise a ValueError exception.

Code:

```
def fn(n):
    if n == 1 or n == 2:
        return 1
    else:
        return fn(n - 1) + fn(n - 2)


try:
    n = int(input("Enter an integer greater than zero: "))
    if n > 0:
        print(f"fn({n}) = {fn(n)}")
    else:
        raise ValueError
except ValueError:
    print("Invalid input. The number must be greater than 0")
```

Result: The program successfully calculates the Fibonacci sequence and validates input using functions.

PYTHON PROGRAMMING LABORATORY

b) Develop a python program to convert binary to decimal, octal to hexadecimal using functions.

Algorithm:

Step 1: Create a function to change binary to decimal.
Step 2: Create a function to change octal to hexadecimal.
Step 3: Ask for a binary number.
Step 4: Use first function to change the binary to decimal.
Step 5: Show the decimal number.
Step 6: Ask for an octal number.
Step 7: Use second function to change the octal to hexadecimal.
Step 8: Show the hexadecimal number.

Code:

```
def binary_to_decimal(binary):
    decimal = 0
    for i, digit in enumerate(binary[::-1]):
        decimal += int(digit) * 2 ** i
    return decimal


def octal_to_hexadecimal(octal):
    decimal = 0
    for i, digit in enumerate(octal[::-1]):
        decimal += int(digit) * 8 ** i

    hexadecimal, hex_values = "", "0123456789ABCDEF"
    while decimal > 0:
        remainder = decimal % 16
        hexadecimal = hex_values[remainder] + hexadecimal
        decimal //= 16
    return hexadecimal


binary = input("Enter a binary number to convert to decimal: ")
print(f"Binary {binary} is equal to decimal", binary_to_decimal(binary))

octal = input("Enter an octal number to convert to hexadecimal: ")
print(f"Octal {octal} is equal to hexadecimal", octal_to_hexadecimal(octal))
```

Result: The program successfully converts binary to decimal and octal to hexadecimal using functions.

PYTHON PROGRAMMING LABORATORY

Program 3:

Aim: Demonstration of manipulation of strings using string methods

a) Write a Python program that accepts a sentence and find the number of words, digits, uppercase letters, and lowercase letters

Algorithm:

Step 1: Get a sentence from the user.
Step 2: Use the split() method to count words.
Step 3: Use the isdigit(), isupper() and islower() methods to count digits, uppercase and lowercase letters.
Step 4: Display all the counts.

Code:

```
sentence = input("Please enter a sentence: ")

words = len(sentence.split())
digits = sum(char.isdigit() for char in sentence)
uppercase = sum(char.isupper() for char in sentence)
lowercase = sum(char.islower() for char in sentence)

print(f"Words: {words}")
print(f"Digits: {digits}")
print(f"Uppercase letters: {uppercase}")
print(f"Lowercase letters: {lowercase}")
```

Result: The program successfully counts words, digits, uppercase, and lowercase letters in a sentence using Python's string methods.

PYTHON PROGRAMMING LABORATORY

b) Write a Python program to find the string similarity between two given strings

Algorithm:

Step 1: Import the difflib library.
Step 2: Define a function to find the similarity between two strings.
Step 3: Get two strings from the user.
Step 4: Display the original strings.
Step 5: Use the function to calculate and display the similarity between the strings.

Code:

```
import difflib
def string_similarity(str1, str2):
    return difflib.SequenceMatcher(None, str1.lower(), str2.lower()).ratio()

str1 = input("Enter first string: ")
str2 = input("Enter second string: ")
print("Original strings:")
print(str1)
print(str2)
print("Similarity between two said strings:")
print(string_similarity(str1, str2))
```

Result: The program successfully measures the string similarity between two given strings.

PYTHON PROGRAMMING LABORATORY

Program 4:

Aim: Discuss different collections like list, tuple, and dictionary

a) Write a python program to implement insertion sort and merge sort using lists

Algorithm:

Insertion Sort:

Step 1: For each element in the list except the first one:
   a) Compare it with the previous elements and shift them right if they are larger.
   b) Otherwise, keep it in place.
Step 2: Continue this process for every element in the list.
Step 3: Stop when no more moves are necessary, indicating the list is sorted.
Step 4: Return the sorted list.

Merge Sort:

Step 1: Create a function to merge two sorted lists.
Step 2: Create a function for merge sort that splits the list, sorts each half, and merges them.
Step 3: If a list has 1 or 0 items, it's already sorted; return it.
Step 4: Merge the two sorted element into one sorted list.
Step 5: Return the sorted list.

Code:

```
def insertion_sort(list):
    for i in range(1, len(list)):
        key, j = list[i], i-1
        while j >= 0 and key < list[j]:
            list[j+1] = list[j]
            j -= 1
        list[j+1] = key
    return list


def merge(left, right):
    result = []
    i = j = 0
    while i < len(left) and j < len(right):
        if left[i] < right[j]:
            result.append(left[i])
            i += 1
        else:
            result.append(right[j])
            j += 1
    result += left[i:]
```

```
    result += right[j:]
    return result


def merge_sort(list):
    if len(list) <= 1:
        return list
    mid = len(list) // 2
    left = merge_sort(list[:mid])
    right = merge_sort(list[mid:])
    return merge(left, right)


data = [12, 11, 13, 5, 6]
print("Sorted list using insertion sort:", insertion_sort(data.copy()))
data = [18, 9, 15, 55, 4, 14]
print("Sorted list using merge sort:", merge_sort(data.copy()))
```

Result: The program successfully implements insertion sort and merge sort using lists.

PYTHON PROGRAMMING LABORATORY

b) Write a program to convert roman numbers into integer values using dictionaries.

Algorithm:

Step 1: Map Roman numerals to integers in a dictionary.
Step 2: Create a function to convert Roman to integer.
Step 3: Get Roman numeral from user.
Step 4: Call the function with user input.
Step 5: Print the converted integer.

Code:

```
roman_dict = {'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000}


def roman_to_int(s):
    result = 0
    for i in range(len(s)):
        current = roman_dict[s[i]]
        if i < len(s) - 1 and current < roman_dict[s[i+1]]:
            result -= current
        else:
            result += current
    return result


user_input = input("Enter a Roman numeral: ").upper()
print("The integer representation is:", roman_to_int(user_input))
```

Result: The program successfully converts Roman numbers to integer values using dictionaries.

PYTHON PROGRAMMING LABORATORY

Program 5:

Aim: Demonstration of pattern recognition with and without using regular expressions

a) Write a function called isphonenumber () to recognize a pattern 415-555-4242 without using regular expression and also write the code to recognize the same pattern using regular expression.

Algorithm:

Step 1: Import the regular expressions (re) module.
Step 2: Define a function using regex for phone number validation.
Step 3: Define another function without using regex for the same purpose.
Step 4: Create a function to display whether a given phone number is valid.
Step 5: Take the user input.
Step 6: Validate the phone number by using the functions.
Step 7: Print the results.

Code:

```
import re


def isphonenumber_regex(phone_number):
    return bool(re.fullmatch(r"\d{3}-\d{3}-\d{4}", phone_number))


def isphonenumber_string(phone_number):
    return phone_number.replace('-', '').isdigit() and [3, 3, 4] == [len(part)
for part in phone_number.split('-')]


def validate_phone_number(phone_number, validation_function, method):
    if validation_function(phone_number):
        print(f"{method}: This phone number '{phone_number}' is valid.")
    else:
        print(f"{method}: This phone number '{phone_number}' is not valid.")


phone_number = input("Enter a phone number to validate (format XXX-XXX-XXXX):
")
validate_phone_number(phone_number, isphonenumber_regex,"Using Regular
Expression")
validate_phone_number(phone_number, isphonenumber_string,"Without using
Regular Expression")
```

Result: The program successfully matches the pattern using regular expressions and non-regular expressions.

PYTHON PROGRAMMING LABORATORY

b) Develop a python program that could search the text in a file for phone numbers (+919900889977) and email addresses (sample@gmail.com)

Algorithm:

Step 1: Import the 're' module.
Step 2: Define the regular expression patterns for phone numbers and email addresses.
Step 3: Prompt the user to input the path of a file.
Step 4: Open and read the file specified by the user.
Step 5: Search for phone numbers in the text and print them.
Step 6: Search for email addresses in the text and print them.

Code:

```
import re
phone_pattern = re.compile(r'\+91\d{10}')
email_pattern = re.compile(r'\w+@\w+\.\w+')
file_path = input("Enter the path of the file: ").strip('\'"')
with open(file_path) as f:
    text = f.read()
print('Phone numbers found:', phone_pattern.findall(text))
print('Email addresses found:', email_pattern.findall(text))
```

Result: The program successfully searches for phone numbers and email addresses within a text file.

PYTHON PROGRAMMING LABORATORY

Program 6:

Aim: Demonstration of reading, writing, and organizing files.

a) Write a python program to accept a file name from the user and perform the following operations

1. Display the first N line of the file.
2. Find the frequency of occurrence of the word accepted from the user in the file

Algorithm:

Step 1: Take user input for the file path.
Step 2: Read lines and get user inputs for word and the first 'N' lines.
Step 3: Display first 'N' lines from file.
Step 4: Count and display frequency of given word.

Code:

```
with open(input('Enter the file name: ').strip('\'"')) as f:
    lines = f.readlines()
    word = input('Enter the word to search: ').lower()
    N = int(input('Enter the number of lines to display: '))
    print(f'Displaying the first {N} lines of the file:')
    print(''.join(lines[:N]))
    word_count = sum(word in line.lower() for line in lines)
    print(f'The word {word} occurs {word_count} times.')
```

Result: The program successfully displays the first N lines of a file and finds the frequency of a user-accepted word.

PYTHON PROGRAMMING LABORATORY

b) Write a python program to create a ZIP file of a particular folder which contains several files inside it.

Algorithm:

Step 1: Import zipfile and os modules.
Step 2: Ask the user for an input folder path and an output zip file path.
Step 3: Create a new zip file at the output path.
Step 4: Add each file from the input folder to the zip file.
Step 5: Close the zip file and display a success message.

Code:

```
import zipfile
import os
input_path = input("Enter the input folder path: ").strip('\'"')
output_path = input("Enter the output zip file path: ").strip('\'"')
zip_file = zipfile.ZipFile(output_path, 'w')
for file in os.listdir(input_path):
    zip_file.write(os.path.join(input_path, file), arcname=file)
zip_file.close()
print("ZIP file created successfully!")
```

Result: The program successfully creates a ZIP file of a folder containing multiple files.

PYTHON PROGRAMMING LABORATORY

Program 7:

Aim: Demonstration of the concepts of classes, methods, objects, and inheritance

a) By using the concept of inheritance write a python program to find the area of triangle, circle, and rectangle.

Algorithm:

Step 1: Import the math module.
Step 2: Define a base class 'Shape' with an area method.
Step 3: Define Triangle, Circle, and Rectangle classes that inherit from Shape and override the area method.
Step 4: Instantiate each shape and print their respective areas.

Code:

```
import math
class Shape:
    def area(self):
        pass


class Triangle(Shape):
    def area(self, base, height):
        return 0.5 * base * height


class Circle(Shape):
    def area(self, radius):
        return math.pi * (radius**2)


class Rectangle(Shape):
    def area(self, length, width):
        return length * width


triangle = Triangle()
circle = Circle()
rectangle = Rectangle()
print(f"The area of the triangle is: {triangle.area(10, 5)} units")
print(f"The area of the circle is: {circle.area(5)} units")
print(f"The area of the rectangle is: {rectangle.area(8, 4)} units")
```

Result: The program successfully calculates the area of a triangle, circle, and rectangle using inheritance.

PYTHON PROGRAMMING LABORATORY

b) Write a python program by creating a class called Employee to store the details of Name, Employee_ID, Department and Salary, and implement a method to update salary of employees belonging to a given department.

Algorithm:

Step 1: Create an Employee class with name, id, dept, sal, and a method to update sal.
Step 2: Input the number of employees and their details, and store them in a list.
Step 3: Input the dept, sal change, and change type for salary update.
Step 4: Loop through the list and update sal for matching dept employees.
Step 5: Loop through the list again and display the updated employee details.

Code:

```
class Employee:
    def __init__(self, name, id, dept, sal):
        self.name = name
        self.id = id
        self.dept = dept
        self.sal = sal

    def update_sal(self, change, change_type='fixed'):
        if change_type == 'fixed':
            self.sal += change
        else:
            self.sal *= (1 + change / 100)


n = int(input("Enter the number of employees: "))
emps = []
for i in range(n):
    name = input("Enter name: ")
    id = input("Enter ID: ")
    dept = input("Enter department: ")
    sal = int(input("Enter salary: "))
    emps.append(Employee(name, id, dept, sal))
dept = input("Enter department for salary update: ")
change = int(input("Enter salary change: "))
change_type = input("Enter change type (fixed/percent): ")
for emp in emps:
    if emp.dept == dept:
        emp.update_sal(change, change_type)
for emp in emps:
    print(f"{emp.name}'s ID: {emp.id}, Dept: {emp.dept}, Salary: {emp.sal}")
```

Result: The program successfully manages employee details and updates the salary of specific department.

PYTHON PROGRAMMING LABORATORY

Program 8:

Aim: Demonstration of classes and methods with polymorphism and overriding

a) Write a python program to find the whether the given input is palindrome or not (for both string and integer) using the concept of polymorphism and inheritance.

Algorithm:

Step 1: Define the base class Palindrome.
Step 2: Define the child classes StringPalindrome and IntegerPalindrome.
Step 3: Take string and integer inputs from the user.
Step 4: Check whether the inputs are palindromes or not.
Step 5: Display the results.

Code:

```
class Palindrome:
    def is_palindrome(self):
        pass


class StringPalindrome(Palindrome):
    def __init__(self, string):
        self.string = string

    def is_palindrome(self):
        return self.string == self.string[::-1]


class IntegerPalindrome(Palindrome):
    def __init__(self, number):
        self.number = number

    def is_palindrome(self):
        return str(self.number) == str(self.number)[::-1]


string_input = input("Enter String to Check: ")
string_palindrome = StringPalindrome(string_input)
print(f"Given string is a Palindrome: {string_palindrome.is_palindrome()}")
int_input = int(input("Enter Int to Check: "))
int_palindrome = IntegerPalindrome(int_input)
print(f"Given integer is a Palindrome: {int_palindrome.is_palindrome()}")
```

Result: The program successfully detects the palindromes using polymorphism and inheritance.

PYTHON PROGRAMMING LABORATORY

Program 9:

Aim: Demonstration of working with excel spreadsheets and web scraping.

a) Write a python program to download the all XKCD comics

Algorithm:

Step 1: Import the requests, os and bs4 modules.
Step 2: Create a directory named 'xkcd' to store the comics.
Step 3: Loop and construct comic URLs.
Step 4: Get the comic image URL and download the image to the 'xkcd' directory.
Step 5: Print success message after all comics are downloaded.

Code:

```
import requests
import os
import bs4
os.makedirs('xkcd', exist_ok=True)
for i in range(1, 10):
    url = f'https://xkcd.com/{i}'
    res = requests.get(url)
    res.raise_for_status()
    soup = bs4.BeautifulSoup(res.text, 'html.parser')
    comic_elem = soup.select_one('#comic img')
    comic_url = 'https:' + comic_elem['src']
    print(f'Downloading comic {comic_url}...')
    comic_name = os.path.basename(comic_url)
    with open(os.path.join('xkcd', comic_name), 'wb') as f:
        f.write(requests.get(comic_url).content)
print('All comics downloaded successfully!')
```

Result: The program successfully downloads all XKCD comics using web scraping.

PYTHON PROGRAMMING LABORATORY

b) Demonstrate python program to read the data from the spreadsheet and write the data in to the spreadsheet

Algorithm:

Step 1: Import the openpyxl module and create a new Workbook.
Step 2: Create and title two sheets: 'Language' and 'Capital'.
Step 3: Append headers and data to each sheet using a loop.
Step 4: Save workbook as 'demo.xlsx'.
Step 5: Search and display state details based on user input.

Code:

```
import openpyxl

wb = openpyxl.Workbook()
ls, cs = wb.active, wb.create_sheet('Capital')
ls.title = 'Language'

data = [
    {'State': 'Karnataka', 'Language': 'Kannada','Capital': 'Bengaluru',
'Code': 'KA'},
    {'State': 'Telangana', 'Language': 'Telugu','Capital': 'Hyderabad',
'Code': 'TS'},
    {'State': 'Tamil Nadu', 'Language': 'Tamil', 'Capital': 'Chennai', 'Code':
'TN'}
]

ls.append(['State', 'Language', 'Code'])
cs.append(['State', 'Capital', 'Code'])

for row in data:
    ls.append([row['State'], row['Language'], row['Code']])
    cs.append([row['State'], row['Capital'], row['Code']])
wb.save('demo.xlsx')

code = input('Enter State Code: ').upper()
for r in ls.iter_rows(min_row=2):
    if r[2].value == code:
        state, language = r[0].value, r[1].value
for r in cs.iter_rows(min_row=2):
    if r[2].value == code:
        capital = r[1].value

print(f'State: {state}\nCapital: {capital}\nLanguage: {language}')
wb.close()
```

Result: The program successfully reads data from a spreadsheet and writes data into it.

PYTHON PROGRAMMING LABORATORY

Program 10:

Aim: Demonstration of working with PDF, word, and JSON files

a) Write a python program to combine select pages from many PDFs

Algorithm:

Step 1: Import the PyPDF2 library.
Step 2: Create a PDF Writer instance.
Step 3: Extract user-defined pages from each PDF.
Step 4: Add the selected pages to the PDF Writer instance.
Step 5: Write the contents of the PDF Writer to a new PDF file.

Code:

```
from PyPDF2 import *
num_pdfs = int(input("Enter the number of PDFs you want to combine: "))
pdf_writer = PdfWriter()
for i in range(num_pdfs):
    path = input(f"Enter the full path to PDF {i+1}: ").strip('\'"')
    with open(path, 'rb') as pdf:
        pdf_reader = PdfReader(pdf)
        pages = input(f"Enter specific page ranges for PDF {i+1} (e.g. 2-
5,7,9-11): ")
        page_nums = [int(x) for x in pages.split(',')]
        for page_num in page_nums:
            page = pdf_reader.pages[page_num - 1]
            pdf_writer.add_page(page)
with open('output.pdf', 'wb') as output:
    pdf_writer.write(output)
```

Result: The program successfully combines select pages from multiple PDFs.

PYTHON PROGRAMMING LABORATORY

b) Write a python program to fetch current weather data from the JSON file

Algorithm:

Step 1: Import the json module to handle JSON data.
Step 2: Open and load data from the 'example.json' file.
Step 3: Print the current temperature from the loaded JSON data.
Step 4: Print the humidity level from the loaded JSON data.
Step 5: Print the weather description from the loaded JSON data.

Code:

```
import json

with open('example.json') as f:
    data = json.load(f)

print(f"Current temperature: {data['main']['temp']}°C")
print(f"Humidity: {data['main']['humidity']}%")
print(f"Weather description: {data['weather'][0]['description']}")
```

Result: The program successfully fetches current weather data from a JSON file