NGI
NAGARJUNA
GROUP OF INSTITUTIONS

NAGARJUNA
COLLEGE OF ENGINEERING & TECHNOLOGY

NAGARJUNA COLLEGE OF ENGINERING AND TECHNOLOGY

(An Autonomous Institute under VTU)

Mudugurki, Venkatgirikote, Devanahalli, Karnataka, Bengaluru-562164.

Department of CSE (Data Science)

Laboratory Manual

IV Semester – 2023

PYTHON PROGRAMMING LAB

(21CDL46)

Prepared by

Subhakar M
Assistant Professor
Department of CSE (Data Science)

1a) Introduce the Python fundamentals, data types, operators, flow control and exception handling in Python

1a). Students test marks for each course is considered as the best of two test average marks out of three test's marks, implement a python program to find the test average marks, take input from the user.

Solution:

```
m1 = int(input("Enter the marks in the first test: "))
m2 = int(input("Enter the marks in second test: "))
m3 = int(input("Enter the marks in third test: "))

if (m1 > m2):
    if (m2 > m3):
        total = m1 + m2
    else:
        total = m1 + m3
elif (m1 > m3):
    total = m1 + m2
else:
    total = m2 + m3

Avg = total / 2
print("The average of the best two test marks is: ", Avg)
```

output:

Case 1:

```
Enter the marks in the first test: 20

Enter the marks in the second test: 15

Enter the marks in the third test: 22

The average of the best two test marks is: 21.0
```

Case 2:

```
Enter the marks in the first test: 20

Enter the marks in the second test: 23

Enter the marks in the third test: 18

The average of the best two test marks is: 21.5
```

Case 3:

```
Enter the marks in the first test: 15

Enter the marks in the second test: 20

Enter the marks in the third test: 21

The average of the best two test marks is: 20.5
```

b). Implement 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.

Solution:

```
temp=num
rev=0
while(num!=0):
    rem=num%10
    rev=rev*10+rem
    num=num//10
    if rem == digit:
        count += 1

if temp==rev:
    print("The number is a palindrome!")

else:
    print("The number isn't a palindrome!")
print("{} occurred {} times in {}".format(digit, count, temp))
```

output:

Case1:

```
Enter number:2315132

Enter a Digit3

The number is a palindrome!

3 occurred 2 times in 2315132
```

Case2:

```
Enter number:1234356

Enter a Digit3

The number isn't a palindrome!

3 occurred 2 times in 1234356
```

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.

Solution

```
nterms = int(input("How many terms? "))

# first two terms
n1, n2 = 0, 1
count = 0

# check if the number of terms is valid
if nterms <= 0:
   print("Please enter a positive integer")
# if there is only one term, return n1
elif nterms == 1:
   print("Fibonacci sequence upto",nterms,":")
   print(n1)
# generate fibonacci sequence
else:
   print("Fibonacci sequence:")
   while count < nterms:
       print(n1)
       nth = n1 + n2
       # update values
       n1 = n2
       n2 = nth
       count += 1
```

output:

Case1:

```
How many terms? 5

Fibonacci sequence:

0

1

1

2

3
```

Case2:

```
How many terms? 5
Fibonacci sequence:
0
1
1
2
3
Process finished with exit code 0
```

Case3:

```
How many terms? -3
Please enter a positive integer
```

2b) Demonstrate how to implement a python program to convert binary to decimal, octal to hexadecimal using functions.

Solution:

```
def decimal_into_binary(decimal_1):
    decimal = int(decimal_1)

    print("The given decimal number", decimal, "in Binary number is: ", bin(decimal))

def decimal_into_octal(decimal_1):
    decimal = int(decimal_1)

    print("The given decimal number", decimal, "in Octal number is: ", oct(decimal))

def decimal_into_hexadecimal(decimal_1):
    decimal = int(decimal_1)

    print("The given decimal number", decimal, " in Hexadecimal number is: ", hex(decimal))

decimal_1 = int(input(" Enter the Decimal Number: "))
decimal_into_binary(decimal_1)
decimal_into_octal(decimal_1)
decimal_into_hexadecimal(decimal_1)
```

output:

Case1:

```
Enter the Decimal Number: 54

The given decimal number 54 in Binary number is:  0b110110

The given decimal number 54 in Octal number is:  0o66

The given decimal number 54  in Hexadecimal number is:  0x36
```

Case2:

```
Enter the Decimal Number: 124

The given decimal number 124 in Binary number is:  0b1111100

The given decimal number 124 in Octal number is:  0o174

The given decimal number 124 in Hexadecimal number is:  0x7c
```

3. Aim: Demonstration of manipulation of strings using string methods

a) When an interpreter reads a line/ sentence from user, find the number of words, digits, uppercase letters and lowercase letters in that sentence; demonstrate with the help of python programming

Solution:

```
s = input("Enter a sentence: ")
w, d, u, l = 0, 0, 0, 0
l_w = s.split()
w =  len(l_w)
for c in s:
    if c.isdigit():
        d = d + 1
    elif c.isupper():
        u = u + 1
    elif c.islower():
        l = l + 1

print ("No of Words: ", w)
print ("No of Digits: ", d)
print ("No of Uppercase letters: ", u)
print ("No of Lowercase letters: ", l)
```

output:

case1:

```
Enter a sentence: My name is Ram

No of Words:  4

No of Digits:  0

No of Uppercase letters:  2

No of Lowercase letters:  9
```

Case2:

```
Enter a sentence: I am 12 years old

No of Words:  5

No of Digits:  2

No of Uppercase letters:  1

No of Lowercase letters:  10
```

b) Let us take two strings compare and find the string similarity between two given strings with the help of python programming

solution:

```
import difflib
def string_similarity(str1, str2):
    result =  difflib.SequenceMatcher(a=str1.lower(), b=str2.lower())
    return result.ratio()
str1 = 'Python Exercises'
str2 = 'Python Exercises'
print("Original string:")
print(str1)
print(str2)
print("Similarity between two said strings:")
print(string_similarity(str1,str2))
str1 = 'Python Exercises'
str2 = 'Python Exercise'
print("\nOriginal string:")
print(str1)
print(str2)
print("Similarity between two said strings:")
print(string_similarity(str1,str2))
str1 = 'Python Exercises'
str2 = 'Python Ex.'
print("\nOriginal string:")
print(str1)
print(str2)
print("Similarity between two said strings:")
print(string_similarity(str1,str2))
str1 = 'Python Exercises'
str2 = 'Python'
print("\nOriginal string:")
print(str1)
print(str2)
print("Similarity between two said strings:")
print(string_similarity(str1,str2))
str1 = 'Python Exercises'
str1 = 'Java Exercises'
print("\nOriginal string:")
print(str1)
print(str2)
print("Similarity between two said strings:")
print(string_similarity(str1,str2))
```

output:

```
Original string:

Python Exercises

Python Exercises

Similarity between two said strings:

1.0


Original string:
```

```
Python Exercises
Python Exercise
Similarity between two said strings:
0.967741935483871

Original string:
Python Exercises
Python Ex.
Similarity between two said strings:
0.6923076923076923

Original string:
Python Exercises
Python
Similarity between two said strings:
0.5454545454545454

Original string:
Java Exercises
Python
Similarity between two said strings:
0.0
```

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

a) User enters a list of random numbers, the programmer need to arrange these random numbers in ascending order with sorting techniques such as insertion sort and merge sort using lists in python.

Solution:

Merge sort

```
def merge(arr, l, m, r):
    n1 = m - l + 1
    n2 = r - m

    # create temp arrays
    L = [0] * (n1)
    R = [0] * (n2)

    # Copy data to temp arrays L[] and R[]
    for i in range(0, n1):
        L[i] = arr[l + i]

    for j in range(0, n2):
        R[j] = arr[m + 1 + j]

    # Merge the temp arrays back into arr[l..r]
    i = 0  # Initial index of first subarray
    j = 0  # Initial index of second subarray
    k = l  # Initial index of merged subarray

    while i < n1 and j < n2:
        if L[i] <= R[j]:
            arr[k] = L[i]
            i += 1
        else:
            arr[k] = R[j]
            j += 1
        k += 1

    # Copy the remaining elements of L[], if there
    # are any
    while i < n1:
        arr[k] = L[i]
        i += 1
        k += 1

    # Copy the remaining elements of R[], if there
    # are any
    while j < n2:
        arr[k] = R[j]
        j += 1
        k += 1


# l is for left index and r is right index of the
# sub-array of arr to be sorted
```

```
def mergeSort(arr, l, r):
    if l < r:
        # Same as (l+r)//2, but avoids overflow for
        # large l and h
        m = l + (r - l) // 2

        # Sort first and second halves
        mergeSort(arr, l, m)
        mergeSort(arr, m + 1, r)
        merge(arr, l, m, r)


# Driver code to test above
arr = [12, 11, 13, 5, 6, 7]
n = len(arr)
print("Given array is")
for i in range(n):
    print("%d" % arr[i], end=" ")

mergeSort(arr, 0, n - 1)
print("\n\nSorted array is")
for i in range(n):
    print("%d" % arr[i], end=" ")
```

output:

```
Given array is

12 11 13 5 6 7



Sorted array is

5 6 7 11 12 13
```

Insertion sort:

```
def insertionSort(arr):
    if (n := len(arr)) <= 1:
        return
    for i in range(1, n):

        key = arr[i]

        # Move elements of arr[0..i-1], that are
        # greater than key, to one position ahead
        # of their current position
        j = i - 1
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key


# sorting the array [12, 11, 13, 5, 6] using insertionSort
arr = [12, 11, 13, 5, 6]
insertionSort(arr)
print(arr)
```

output:

```
Sorted array is:

[5, 6, 11, 12, 13]
```

4b). Demonstrate with a python program to convert roman numbers into integer values using dictionaries, by taking inputs from user.

Solution:

```
class Solution(object):
  def romanToInt(self, s):

    roman =
{'I':1,'V':5,'X':10,'L':50,'C':100,'D':500,'M':1000,'IV':4,'IX':9,'XL':40,'XC':90,'CD':400,'CM':900}
    i = 0
    num = 0
    while i < len(s):
      if i+1<len(s) and s[i:i+2] in roman:
        num+=roman[s[i:i+2]]
        i+=2
      else:
        num+=roman[s[i]]
        i+=1
    return num
ob1 = Solution()
print(ob1.romanToInt("III"))
print(ob1.romanToInt("CDXLIII"))
```

output:

```
3

443
```

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

5a) Implement 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.

Solution:

Without regular expression

```
def isPhoneNumber(text):
  if len(text) != 12:
     return False
     for i in range(0, 3):
      if not text[i].isdecimal():
             return False
  if text[3] != '-':
     return False
     for i in range(4, 7):
      if not text[i].isdecimal():
             return False
  if text[7] != '-':
     return False
     for i in range(8, 12):
      if not text[i].isdecimal():
             return False
  return True

print('Is 415-555-4242 a phone number?')
print(isPhoneNumber('415-555-4242'))
print('Is Moshi moshi a phone number?')
print(isPhoneNumber('Moshi moshi'))
```

output:

```
Is 415-555-4242 a phone number?
True
Is Moshi moshi a phone number?
False
```

With regular expression:

```
import re
phoneNumRegex = re.compile(r'\d\d\d-\d\d\d-\d\d\d\d')
mo = phoneNumRegex.search('My number is 415-555-4242.')
print('Phone number found: ' + mo.group())
```

output:

```
Phone number found: 415-555-4242
```

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

```
import pyperclip, re    # Importing the libraries(this case: regex and pyperclip)

# Create phone regex.
phoneRegex = re.compile(r'''(
   (\d{3}|\(\d{3}\))? # area code
   (\s|-|\.)? # separator
   (\d{3}) # first 3 digits
   (\s|-|\.) # separator
   (\d{4}) # last 4 digits
   (\s*(ext|x|ext.)\s*(\d{2,5}))?                    # extension
   )''', re.VERBOSE)

# TODO: Create email regex.
# TODO: Find matches in clipboard text.
# TODO: Copy results to the clipboard.

# Create email regex.
emailRegex = re.compile(r'''(
   [a-zA-Z0-9._%+-] + #username
   @                # @symbole
   [a-zA-Z0-9.-] +     # domain
   (\.[a-zA-Z]{2,4})   # dot-something
   )''', re.VERBOSE)


# Find matches in the clipboard text.

text = str(pyperclip.paste())

matches = []

for groups in phoneRegex.findall(text):

   phoneNum = '-'.join([groups[1], groups[3], groups[5]])

   if groups[8] != '':

      phoneNum += ' x' + groups[8]

   matches.append(phoneNum)

for groups in emailRegex.findall(text):

   matches.append(groups[0])


if len(matches) > 0:

   pyperclip.copy('\n'.join(matches))

   print('Copied to clipboard: ')
```

```
    print('\n'.join(matches))
# TODO: Pasting the content --> txt file.
    s = pyperclip.paste()
    with open('phone&emailfinder.txt','w') as g:
        g.write(s)
    g.close()
else:
    print('No phone numbers or email addresses found.')
```

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

a) Demonstrate how files are read in python by considering myfile.txt as an example file name which is entered by the user to 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

1. Display the first N line of the file

```
# Get the file name from the user
file_name = input("Enter the file name (e.g., myfile.txt): ")

try:
   # Open the file in read mode
   with open(file_name, 'r') as file:
      # Read all the lines into a list
      lines = file.readlines()

      # Get the number of lines to display from the user
      num_lines = int(input("Enter the number of lines to display: "))

      # Display the first N lines
      for line in lines[:num_lines]:
         print(line.strip())

except FileNotFoundError:
   print("File not found. Please make sure the file exists in the specified path.")
```

Output:

```
Enter the file name (e.g., myfile.txt): myfile.txt
Enter the number of lines to display: 2
this is my file
Display the first N line of the file

Enter the file name (e.g., myfile.txt): myfile.txt
Enter the number of lines to display: 1
this is my file
```

6 a)

2. Find the frequency of occurrence of the word accepted from the user in the file

```
# Get the file name from the user

file_name = input("Enter the file name (e.g., myfile.txt): ")

try:
    # Open the file in read mode
    with open(file_name, 'r') as file:
        # Get the word from the user
        word = input("Enter the word to find its frequency: ")

        # Initialize a counter variable
        frequency = 0

        # Loop through each line of the file
        for line in file:
            # Split the line into words
            words = line.strip().split()

            # Count the occurrences of the word
            frequency += words.count(word)

        # Display the frequency of the word
        print(f"The word '{word}' appeared {frequency} times in the file.")

except FileNotFoundError:
    print("File not found. Please make sure the file exists in the specified path.")
```

Output:

```
Enter the file name (e.g., myfile.txt): myfile.txt
Enter the word to find its frequency: file
The word 'file' appeared 2 times in the file.


Enter the file name (e.g., myfile.txt): myfile.txt
Enter the word to find its frequency: my
The word 'my' appeared 1 times in the file.
```

6 b) Python is termed as secure language, demonstrate a simple method of securing data by creating a ZIP file of a particular folder which contains several files inside it.

```
import zipfile
import os

# Get the folder path from the user
folder_path = input("Enter the folder path to zip: ")

# Get the ZIP file name from the user
zip_file_name = input("Enter the name of the ZIP file to create: ")

try:
    # Create a new ZIP file
    with zipfile.ZipFile(zip_file_name, 'w', zipfile.ZIP_DEFLATED) as zipf:
        # Walk through each file in the folder
        for root, _, files in os.walk(folder_path):
            for file in files:
                # Get the absolute path of the file
                file_path = os.path.join(root, file)

                # Add the file to the ZIP archive
                zipf.write(file_path, arcname=file)

    print(f"The folder '{folder_path}' has been securely zipped into the file '{zip_file_name}'.")

except FileNotFoundError:
    print("Folder not found. Please make sure the folder exists in the specified path.")
```

Output:

```
Enter the folder path to zip: C:\Users\SAM_LALI\Desktop\Temp22062023
Enter the name of the ZIP file to create: tt

The folder 'C:\Users\SAM_LALI\Desktop\Temp22062023' has been securely zipped into the file 'tt'.
```

7 a) Inheritance is one of the main pillars of OOPs concept. By using inheritance, a child class acquires all properties and behaviors of parent class. Referring the above inheritance concept write a python program to find the area of triangle, circle and rectangle.

```
import math


# Parent class
class Shape:
    def __init__(self):
        pass

    def calculate_area(self):
        pass

# Child class - Triangle inherits from Shape
class Triangle(Shape):
    def __init__(self, base, height):
        super().__init__()
        self.base = base
        self.height = height

    def calculate_area(self):
        return 0.5 * self.base * self.height

# Child class - Circle inherits from Shape
class Circle(Shape):
    def __init__(self, radius):
        super().__init__()
        self.radius = radius

    def calculate_area(self):
        return math.pi * self.radius**2

# Child class - Rectangle inherits from Shape
class Rectangle(Shape):
    def __init__(self, length, width):
        super().__init__()
        self.length = length
        self.width = width

    def calculate_area(self):
        return self.length * self.width

# Creating instances of each class
triangle = Triangle(5, 7)
circle = Circle(3)
rectangle = Rectangle(4, 6)

# Calculating and printing the area of each shape
print("Area of Triangle:", triangle.calculate_area())
print("Area of Circle:", circle.calculate_area())
print("Area of Rectangle:", rectangle.calculate_area())
```

Output:

```
Area of Triangle: 17.5
Area of Circle: 28.274333882308138
Area of Rectangle: 24
```

7 b) Implement 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.

```
class Employee:
    def __init__(self, name, employee_id, department, salary):
        self.name = name
        self.employee_id = employee_id
        self.department = department
        self.salary = salary

    def update_salary(self, new_salary, department):
        if self.department == department:
            self.salary = new_salary

    def __str__(self):
        return f"Name: {self.name}\nEmployee_ID: {self.employee_id}\nDepartment:
         {self.department}\nSalary: {self.salary}"


# Create employees
employee1 = Employee("John Doe", 1, "Finance", 5000)
employee2 = Employee("Jane Smith", 2, "Engineering", 6000)
employee3 = Employee("Tom Wilson", 3, "Engineering", 5500)

# Print employee details before updating salary
print("Before salary update:")
print(employee1)
print(employee2)
print(employee3)

# Update salary of employees belonging to "Engineering" department
new_salary = 6500
department = "Engineering"
employee2.update_salary(new_salary, department)
employee3.update_salary(new_salary, department)

# Print employee details after updating salary
print("\nAfter salary update:")
print(employee1)
print(employee2)
print(employee3)
```

Output:

```
Before salary update:
Name: John Doe
Employee_ID: 1
Department: Finance
Salary: 5000
```

```
Name: Jane Smith
Employee_ID: 2
Department: Engineering
Salary: 6000
Name: Tom Wilson
Employee_ID: 3
Department: Engineering
Salary: 5500

After salary update:
Name: John Doe
Employee_ID: 1
Department: Finance
Salary: 5000
Name: Jane Smith
Employee_ID: 2
Department: Engineering
Salary: 6500
Name: Tom Wilson
Employee_ID: 3
Department: Engineering
Salary: 6500
```

8. a) Inheritance applies to classes, whereas polymorphism applies to methods, using these concepts implement a python program to find whether the given input is palindrome or not (forboth string and integer).

```
# Parent class
class PalindromeChecker:
    def __init__(self, input_str):
        self.input_str = input_str

    def is_palindrome(self):
        pass

# Child class for string palindrome
class StringPalindromeChecker(PalindromeChecker):
    def __init__(self, input_str):
        super().__init__(input_str)

    def is_palindrome(self):
        reversed_str = self.input_str[::-1]
        return self.input_str.lower() == reversed_str.lower()

# Child class for integer palindrome
class IntegerPalindromeChecker(PalindromeChecker):
    def __init__(self, input_int):
        super().__init__(str(input_int))

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

# Getting user input
user_input = input("Enter a string or number to check palindromicity: ")

# Checking if the input is a string palindrome
str_palindrome = StringPalindromeChecker(user_input)
if str_palindrome.is_palindrome():
    print("The entered string is a palindrome.")
else:
    print("The entered string is not a palindrome.")

# Checking if the input is an integer palindrome (if it is a valid integer)
try:
    int_input = int(user_input)
    int_palindrome = IntegerPalindromeChecker(int_input)
    if int_palindrome.is_palindrome():
        print("The entered number is a palindrome.")
    else:
        print("The entered number is not a palindrome.")
except ValueError:
    pass
```

Output:

```
Enter a string or number to check palindromicity: noon
The entered string is a palindrome.

Enter a string or number to check palindromicity: 124
The entered string is not a palindrome.

Enter a string or number to check palindromicity: jam
The entered string is not a palindrome.
```

9. a) XKCD is a webcomic website consists of many curious comics and sometimes user wants to save that comic image on their local devices, a user has to visit every page of the comic website. instead implement a python program to download the all XKCD comics.

```
import requests
import os

# Define the folder path to save the comics
folder_path = 'xkcd_comics'

# Create the folder if it doesn't exist
os.makedirs(folder_path, exist_ok=True)

# Fetch the latest comic number
response = requests.get('https://xkcd.com/info.0.json')
latest_comic_number = response.json()['num']

# Iterate through all the comics and download them
for comic_number in range(1, latest_comic_number + 1):
    response = requests.get(f'https://xkcd.com/{comic_number}/info.0.json')
    comic_info = response.json()

    # Extract the image URL
    image_url = comic_info['img']

    # Download the image
    response = requests.get(image_url)
    file_path = os.path.join(folder_path, f'comic{comic_number}.png')

    # Save the image to the local folder
    with open(file_path, 'wb') as file:
        file.write(response.content)

    print(f"Downloaded comic {comic_number}.")

print("All XKCD comics downloaded successfully!")
```

first need to install requests for importing

```
pip install requests
```

9. b) In python programming how to read the data from the spreadsheet and write the data in to the spreadsheet , by using the load_workbook() method, demonstrate with code snippet.

```
import openpyxl

# Load the workbook
workbook = openpyxl.load_workbook('example.xlsx')

# Select the worksheet
worksheet = workbook.active

# Read data from a cell
cell_value = worksheet.cell(row=1, column=1).value

# Write data to a cell
worksheet.cell(row=1, column=2).value = 'New Value'

# Save the changes
workbook.save('example.xlsx')
```

10. a) Demonstrate with a python program the possible ways to combine select pages from many PDFs.

```
from PyPDF2 import PdfFileMerger

# Create a merger object
merger = PdfFileMerger()

# Add PDFs to the merger object
merger.append(open('file1.pdf', 'rb'))
merger.append(open('file2.pdf', 'rb'))

# Select pages from the PDFs and add them to the output file
merger.addBookmark('Page 1', 0)
merger.addBookmark('Page 2', 1)
merger.write(open('output.pdf', 'wb'))

# Close the merger object
merger.close()
```

10. b) Accessing current weather data for any location on Earth, We collect and process weather data from different sources such as global and local weather models, satellites, radars and a vast network of weather stations and in different format. Implement a python program to fetch current weather data from the JSON file format.

```
import json

def fetch_weather_data(file_path):
    with open(file_path, 'r') as file:
        data = json.load(file)
        return data

# Replace 'weather_data.json' with the actual file path of your JSON file
file_path = 'weather_data.json'

try:
    weather_data = fetch_weather_data(file_path)
    # Process the weather data as per your requirements
    print(weather_data)
except FileNotFoundError:
    print(f"File '{file_path}' not found.")
except json.JSONDecodeError:
    print(f"Invalid JSON format in '{file_path}'.")
```