**Aim: Introduce the Python fundamentals, data types, operators and flow control 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

1b). 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**

def F(N):  
 if N <= 0:  
 print("Error: Number of terms must be a positive integer.")  
 return  
 elif N == 1:  
 return [0]  
 elif N == 2:  
 return [0, 1]  
 else:  
 f\_n = [0,1] *# fn = [fn-1,fn-2]* while len(f\_n) < N:  
 f\_n\_next = f\_n[-1] + f\_n[-2] *#Next term fn = fn-1+fn-2* f = f\_n.append(f\_n\_next) *#Creating a list of terms* return f\_n  
  
  
*# Accept input from the user*n = int(input("Enter a positive integer (N > 0): "))  
  
*# Call the Fibonacci function and display the result or error message*result = F(n)  
if result is not None:  
 print(f"The numbers till N are: {result}")

**output:**

case1:

Enter a positive integer (N > 0): 0

Error: Number of terms must be a positive integer

case2:

Enter a positive integer (N > 0): 1

The numbers till N are: [0]

case3:

Enter a positive integer (N > 0): 2

The numbers till N are: [0, 1]

Case4:

Enter a positive integer (N > 0): 5

The numbers till N are: [0, 1, 1, 2, 3]

2b) Demonstrate how to implement a python program to convert decimal to binary, decimal to octal and decimal 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

Simila.rity 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 mergeSort(arr):

if len(arr) > 1:

# Create sub\_array2 ← A[start..mid] and sub\_array2 ← A[mid+1..end]

mid = len(arr)//2

sub\_array1 = arr[:mid]

sub\_array2 = arr[mid:]

# Sort the two halves

mergeSort(sub\_array1)

mergeSort(sub\_array2)

# Initial values for pointers that we use to keep track of where we are in each array

i = j = k = 0

# Until we reach the end of either start or end, pick larger among

# elements start and end and place them in the correct position in the sorted array

while i < len(sub\_array1) and j < len(sub\_array2):

if sub\_array1[i] < sub\_array2[j]:

arr[k] = sub\_array1[i]

i += 1

else:

arr[k] = sub\_array2[j]

j += 1

k += 1

# When all elements are traversed in either arr1 or arr2,

# pick up the remaining elements and put in sorted array

while i < len(sub\_array1):

arr[k] = sub\_array1[i]

i += 1

k += 1

while j < len(sub\_array2):

arr[k] = sub\_array2[j]

j += 1

k += 1

arr = [10, 9, 2, 4, 6, 13]

mergeSort(arr)

print(arr)

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