1 > Aim: Introduce the Python fundamentals, data types, operators, flow control and exception handling in Python

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

```
# 1>A

m1=float(input("Enter 1st marks :"))
m2=float(input("Enter 2nd marks :"))
m3=float(input("Enter 3rd marks :"))
minimum=min(m1,m2,m3)
best_of_two=(m1+m2+m3)-minimum
avg=best_of_two/2
print(" Avg. of Best of two marks is:",avg)
```

Output:

Enter 1st marks :30
Enter 2nd marks :29
Enter 3rd marks :30
Avg. of Best of two marks is: 30.0

Enter 1st marks :30
Enter 2nd marks :29
Enter 3rd marks :28
Avg. of Best of two marks is: 29.5

1 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.

```
# 1>B
def palindrome(n):
    return n == n[::-1]

word = input("Enter word or Number to check palindrome: ")
is_palindrome = palindrome(word)

if is_palindrome:
    print(word, "is a palindrome word")
    for char in set(word):
        print(char, ":", word.count(char))
else:
    print(word, "is not a palindrome word")
```

Output:

Enter word or Number to check palindrome: madam
madam is a palindrome word
a : 2
m : 2
d : 1

Enter word or Number to check palindrome: 121
121 is a palindrome word
1 : 2
2 : 1

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

2 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.

```
# 2>A
def fibonacci(n):
    if n<=0:
        raise ValueError(" N Must be a positive Integer ")
    if n==1:
        return 0
    if n==2:
        return 1
    else:
        return fibonacci(n-1)+fibonacci(n-2)

try:
    n=int(input("Enter the Number :"))
    result=fibonacci(n)
    print("Fibonacci of", n,"is :",result)
except ValueError as error:
    print(error)
```

Output:

Enter the Number :5
Fibonacci of 5 is : 3

Enter the Number :3
Fibonacci of 3 is : 1

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

```
#2>B
# binary  to decimal
def bin2dec(binary):
    decimal=0
    power=0
    while binary!=0:
        last_digit=binary%10
        decimal+=last_digit*(2**power)
        binary//=10
        power+=1
    return decimal

# # octal to hexadecimal
def oct2hex(octal):
    decimal = 0
    power = 0
    while octal != 0:
        last_digit = octal % 10
        decimal += last_digit * (8 ** power)
        octal //= 10
        power += 1
    hexadecimal = hex(decimal).upper()
    return hexadecimal

b=int(input("Enter the binary Number :"))
result_dec=bin2dec(b)
print("IN Decimal :",result_dec)

octal=int(input("Enter the octal Number :"))
result_hex=oct2hex(octal)
print("IN Hexa-Decimal :",result_hex)
```

Output:

Enter the binary Number :1011
IN Decimal : 11
Enter the octal Number :13
IN Hexa-Decimal : 0XB

Note: we can implement oct2hex in another way as follows:

Conversion of Octal to hexadecimal using the following steps:

- Input the octal number.
- An octal number system is converted into a binary number system.
- Extract the 4 bits in a group from the right-side.
- Provide the hexadecimal number to the extracted 4 bits.

```
def oct2bin(octal):
    binary = ""
    while octal > 0:
        digit = octal % 10
        binary_digit = bin(digit)[2:].zfill(3)
        binary = binary_digit + binary
        octal //= 10
    return binary

def bin_to_hex(binary):
    hex_digits = "0123456789ABCDEF"
    hex_result = ""

    # Pad binary with zeros to make it a multiple of 4
    while len(binary) % 4 != 0:
        binary = "0" + binary

    # Process binary in groups of 4 bits and convert to hexadecimal
    for i in range(0, len(binary), 4):
        group = binary[i:i+4]
        hex_digit = hex(int(group, 2))[2:]
        hex_result = hex_digit + hex_result

    return hex_result.upper()

octal = int(input("Enter the octal Number: "))
binary_result = oct2bin(octal)
print("In Binary:", binary_result)

hexadecimal = bin_to_hex(binary_result)
print("In Hexadecimal:", hexadecimal[::-1])
```

Output:

Enter the octal Number: 13
In Binary: 001011
In Hexadecimal: 0B

Enter the octal Number: 123
In Binary: 001010011
In Hexadecimal: 053

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

3A) Write a Python program that accepts a sentence and find the number of words, digits, uppercase letters and lowercase letters

```
# 3>A
sentence=input("Enter the sentence:")
word =sentence.split()
print("Number of Words :",len(word))

digits=sum(char.isdigit() for char in sentence)
print("Number of Digits :",digits)

upercase=sum(char.isupper() for char in sentence)
print("Number of Uppercase letter :",upercase)

lowercase=sum(char.islower() for char in sentence)
print("Number of lowercase letter:",lowercase)
```

Output :

Enter the sentence:This is the Program no 3A
Number of Words : 6
Number of Digits : 1
Number of Uppercase letter : 3
Number of lowercase letter: 16

3B) Write a Python program to find the string similarity between two given strings
Sample Output:
Original string:
Python Exercises
Python Exercises
Similarity between two said strings:
1.0
Sample Output:
Original string:
Python Exercises
Python Exercise
Similarity between two said strings:
0.967741935483871

```
# 3>B

from difflib import SequenceMatcher

def string_similarity(str1,str2):
    similarity_ratio=SequenceMatcher(None,str1,str2).ratio()
    return similarity_ratio

str1=input("Enter the original string :")
str2=input("Enter the string to check similarity :")

similarity=string_similarity(str1,str2)
print("Similarity between Both string is ", similarity)
print("Similarity between Both string in percent : ", int(similarity*100),"%")
```

Output:

Enter the original string :Department AIML
Enter the string to check similarity :Department AI
Similarity between Both string is 0.9285714285714286
Similarity between Both string in percent : 92 %

4 > Aim: Demonstration of manipulation of strings using string methods

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

```
# 4>A
def merge_sort(lst):
    if len(lst) > 1:
        mid = len(lst) // 2
        left_half = lst[:mid]
        right_half = lst[mid:]

        merge_sort(left_half)
        merge_sort(right_half)

        i = j = k = 0

        while i < len(left_half) and j < len(right_half):
            if left_half[i] < right_half[j]:
                lst[k] = left_half[i]
                i += 1
            else:
                lst[k] = right_half[j]
                j += 1
            k += 1

        while i < len(left_half):
            lst[k] = left_half[i]
            i += 1
            k += 1

        while j < len(right_half):
            lst[k] = right_half[j]
            j += 1
            k += 1

    return lst


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

```
my_list = []

for i in range(5):
    n=int(input("Enter the elements to append in list for Sorting:"))
    my_list.append(n)

print("\nUnsorted List")
print(my_list)

print("Sorting using Merge Sort")
merge_sort(my_list)
print(my_list)

print("Sorting using Insertion Sort")
insertion_sort(my_list)
print(my_list)
```

Output:
Enter the elements to append in list for Sorting:96
Enter the elements to append in list for Sorting:56
Enter the elements to append in list for Sorting:12
Enter the elements to append in list for Sorting:68
Enter the elements to append in list for Sorting:7

Unsorted List
[96, 56, 12, 68, 7]
Sorting using Merge Sort
[7, 12, 56, 68, 96]
Sorting using Insertion Sort
[7, 12, 56, 68, 96]

4 B) Write a program to convert roman numbers in to integer values using dictionaries.

```
# 4>B
def roman_to_int(roman):
    roman_dict = {'I': 1,'V': 5,'X': 10,'L': 50,'C': 100, 'D': 500, 'M': 1000}

    total = 0
    prev_value = 0

    for char in roman:
        value = roman_dict[char]

        if value > prev_value:
            total += value - 2* prev_value
        else:
            total += value

        prev_value = value

    return total
roman=input("Enter the Roman number :")
result =roman_to_int(roman)
print("Roman Number ", roman, "in Integer is:",result)
```

Output:

Enter the Roman number :VI
Roman Number VI in Integer is: 6

Enter the Roman number :IV
Roman Number IV in Integer is: 4

Enter the Roman number :DCM
Roman Number DCM in Integer is: 1400

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

5 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.

```
#5>A
def isPhonenumber(number):
    if len(number)!=12:
        return False
    if number [3]!="-" or number[7]!="-":
        return False
    if not number[:3].isdigit() or not number[4:7].isdigit() or not
number[8:].isdigit() :
        return False
    return True

import re
def regex(number):
    pattern=('\d{3}\-\d{3}\-\d{4}')
    return bool (re.match(pattern,number))

n=input("Enter the number :")
result=isPhonenumber(n)
print("Number is valid without regular expression:",result)

result1=regex(n)
print("Number is valid with regular expression:",result1)
```

Output:

Enter the number :727-293-0302
Number is valid without regular expression: True
Number is valid with regular expression: True

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

```
#5>B

import re
f=input("Enter the file name or path:")
file=open(f,"r")
content=file.read()
Emailadd=re.findall('\S+@\S{9}',content)
for email in Emailadd:
    print("Email found :",email)
PhNo=re.findall('\+\d{12}',content)
for ph in PhNo:
    print("Phone Number found :",ph)
```

Output:
Enter the file name or path:sagar.txt
Email found : sagar181203@gmail.com
Phone Number found : +917272930302

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

6 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

```
#6>A
fname=input("Enter your file name or path:")
file=open(fname,"r")
n=int(input("Enter the number of Lines:"))
data=file.readlines()
for lines in data[:n]:
    print(lines,end="")
word=input("Enter the word to count:")
counting=0
for Each in data:
    counting+=Each.count(word)
print("Word Found :",counting,"times in file ")
```

Output:
Enter your file name or path:sagar.txt
Enter the number of Lines:10
sagar_kumar
4PM21AI033
sagar181203@gmail.com
+917272930302
sagar
1
2
3
4
1
Enter the word to count:sagar
Word Found : 3 times in file

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

```
#6>B
import shutil
# only for folder only
source_folder=input("Enter the path or folder name which to be Zipped : ")
output_folder=input("Set the Zip folder name :")
try:
 archived=shutil.make_archive(output_folder,"zip",source_folder)
 print("Zip file created as ",f"{output_folder}.zip")
except FileNotFoundError :
    print("Source folder not found")
```

Output:
Enter the path or folder name which to be Zipped : Test
Set the Zip folder name :Test
Zip file created as Test.zip

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

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

```
# 7>A
import math

class Shape:
    def area(self):
        pass

class Triangle(Shape):
    def __init__(self, base, height):
        self.base = base
        self.height = height

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

class Circle(Shape):
    def __init__(self, radius):
        self.radius = radius

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

class Rectangle(Shape):
    def __init__(self, length, width):
        self.length = length
        self.width = width

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

triangle = Triangle(5, 8)      # Example usage-give the parameter
circle = Circle(3)
rectangle = Rectangle(4, 6)

print("Area of Triangle:", triangle.area())
print("Area of Circle:", circle.area())
print("Area of Rectangle:", rectangle.area())
```

Output:
Area of Triangle: 20.0
Area of Circle: 28.274333882308138
Area of Rectangle: 24

7 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.

```
# 7>B

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

    def display_details(self):
        print("Employee Details")
        print("Name:", self.name)
        print("Employee ID:", self.emp_id)
        print("Department:", self.department)
        print("Salary:", self.salary)

    def update_salary(self, new_salary):
        self.salary = new_salary
        print("Updated Salary:", self.salary)

# Create a list to store employee objects
employees = []

# Function to update salary for employees in a given department
def update_salary_by_department(department, new_salary):
    for employee in employees:
        if employee.department == department:
            employee.update_salary(new_salary)

# Function to add a new employee
def add_employee():
    name = input("\nEnter Employee name: ")
    emp_id = input("Enter Employee ID: ")
    department = input("Enter Employee Department: ")
    salary = int(input("Enter Employee Salary: "))
    employee = Employee(name, emp_id, department, salary)
    employees.append(employee)

# Adding n employees
n=int(input("Enter the Number of Employee:"))
for i in range (n):
    add_employee()
```

```
# Display all employee details
for employee in employees:
    employee.display_details()

# Update salary for employees in a specific department
dept_to_update = input("Enter the department to update salaries: ")
new_salary = int(input(f"Enter the new salary for employees in the
{dept_to_update} department: "))
update_salary_by_department(dept_to_update, new_salary)

print("\nUpdated employee details")
for employee in employees:
    employee.display_details()
```

Output:
Enter the Number of Employee:2

Enter Employee name: sagar
Enter Employee ID: AI033
Enter Employee Department: AIML
Enter Employee Salary: 100000

Enter Employee name: Kumar
Enter Employee ID: AI37
Enter Employee Department: LIB
Enter Employee Salary: 100000
Employee Details
Name: sagar
Employee ID: AI033
Department: AIML
Salary: 100000
Employee Details
Name: Kumar
Employee ID: AI37
Department: LIB
Salary: 100000
Enter the department to update salaries: AIML
Enter the new salary for employees in the AIML department: 150000
Updated Salary: 150000

Updated employee details
Employee Details
Name: sagar
Employee ID: AI033

Department: AIML
Salary: 150000
Employee Details
Name: Kumar
Employee ID: AI37
Department: LIB
Salary: 100000

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

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.

```
# 8
class PalindromeChecker:
    def is_palindrome(self, value):
        # Convert value to a string to handle both strings and integers
        str_value = str(value)
        return str_value == str_value[::-1]

# SI_PalindromeChecker that inherits from PalindromeChecker
class SI_PalindromeChecker(PalindromeChecker):
    def is_palindrome(self, value):
        # Override the base class method for string input
        return super().is_palindrome(str(value))
# is_palindrome method varies depending on the type of input this shows the
# polymorphism

def main():
    input_value = input("Enter a string or an integer: ")
    checker = SI_PalindromeChecker()

    if checker.is_palindrome(input_value):
        print(f"'{input_value}' is a palindrome.")
    else:
        print(f"'{input_value}' is not a palindrome.")

if __name__ == "__main__":
    main()
```

Output:

Enter a string or an integer: madam
'madam' is a palindrome.

Enter a string or an integer: 121
'121' is a palindrome.

9 > Aim: Demonstration of working with excel spreadsheets and web scraping

9 A) Write a python program to download the all XKCD comics.

```
# 9>A
# Importing required modules
import requests as req
import os, bs4

# Storing website URL
url = 'https://xkcd.com/'
# Make Directory to store image
os.makedirs('xkcd', exist_ok=True)
# exist_ok prevent program from throwing an exception if the folder existed

n = input("Input the comic number ")
# Append the user comic number in the url
url += n
print('Downloading image from %s...' % url)
# Request the url from the web
res = req.get(url)

# Now Store the HTML page that is found in the url
soup = bs4.BeautifulSoup(res.text)

# Find the Element that contain the image tag
comicElem = soup.select('#comic img')

# Now get that source of the image and make it as a url
comicUrl = 'http:' + comicElem[0].get('src')

# Request the url from the web
res = req.get(comicUrl)

# Save the file in the directory  , " wb is for write binary"
imageFile = open(os.path.join('xkcd', os.path.basename(comicUrl)), 'wb')
for chunk in res.iter_content(1):
    # Writing the binary image file
    imageFile.write(chunk)
# Closing the binary image file
imageFile.close()
print('Successfully downloaded')
```

Output:

Input the comic number :2

Downloading image from https://xkcd.com/2..

Successfully downloaded

9B) Demonstrate python program to read the data from the spreadsheet and write the data in to the spreadsheet

```
# 9>B
import openpyxl

class ExcelHandler:
    def __init__(self, file_path):
        self.file_path = file_path
        self.workbook = openpyxl.load_workbook(file_path)
        self.sheet = self.workbook.active

    def read_data(self):
        try:
            for row in self.sheet.iter_rows(values_only=True):
                print(row)
        except Exception as e:
            print(f"Error reading data from {self.file_path}: {e}")

    def write_data(self, data):
        try:
            self.sheet.append(data)
            self.workbook.save(self.file_path)
            print(f"Data written to {self.file_path} successfully.")
        except Exception as e:
            print(f"Error writing data to {self.file_path}: {e}")

# which have taste and dish column
input_file = "Food_review.xlsx"

# Create an ExcelHandler instance for the input file
excel_handler = ExcelHandler(input_file)

# Read and display data from the input spreadsheet
excel_handler.read_data()

# Define data to write (e.g., a new row)
new_data = ["yummy", "burger"]

# Write data to the same Excel worksheet
excel_handler.write_data(new_data)

# Read and display the updated data
excel_handler.read_data()
```

Output:
('REVIEWS', 'DISHES')

('Good burger', 'burger ')
('burnt burger', 'burger ')
('salty burger', 'burger ')
Data written to Food_review.xlsx successfully.
('REVIEWS', 'DISHES')
('Good burger', 'burger ')
('burnt burger', 'burger ')
('salty burger', 'burger ')
('yummy', 'burger')

OR

Note : in another way we can implement the reading and writing the data in excel sheet as follows ---

```
from openpyxl import Workbook

wb = Workbook() # Create a new Excel workbook and add sheets
sheet = wb.active
sheet.title = "Language"
wb.create_sheet(title = "Capital")

# Data for the Excel sheet
lang = ["Kannada", "Telugu", "Tamil"]
state = ["Karnataka", "Telangana", "Tamil Nadu"]
code =['KA', 'TS', 'TN']

# Set the headers for the sheet
sheet.cell(row = 1, column = 1).value = "State"
sheet.cell(row = 1, column = 2).value = "Language"
sheet.cell(row = 1, column = 3).value = "Code"

# Populate the sheet with data
for i in range(3):
    sheet.cell(row = i+1, column = 1).value = state[i]
    sheet.cell(row = i+1, column = 2).value = lang[i]
    sheet.cell(row = i+1, column = 3).value = code[i]

wb.save("demo0.xlsx")# Save the workbook as "demo0.xlsx"
sheet = wb["Language"]

# Read and print the first row (headers), we can change the number of row
for row in sheet.iter_rows(min_row=1, max_row=1, values_only=True):
    print(row)

wb.close()# Close the workbook
```

Output : 'Karnataka', 'Kannada', 'KA')

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

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

```
#10>A
from PyPDF2 import PdfReader, PdfWriter

# Get the number of PDF documents to combine
n = int(input("Enter the number of PDF documents to combine: "))

# Initialize the PDF writer
pdf_writer = PdfWriter()

# Loop to input the page numbers for each PDF document
for i in range(1, n + 1):
    pdf_file_name = input(f"Enter the name of PDF file {i}: ")
    page_num = int(input(f"Enter the page number for PDF file {i}: "))

    # Open the PDF document
    pdf_file = open(pdf_file_name, 'rb')

    # Read and add the specified page from the document
    pdf_reader = PdfReader(pdf_file)
    if page_num <= len(pdf_reader.pages) and page_num > 0:
        page = pdf_reader.pages[page_num - 1]
        pdf_writer.add_page(page)
    else:
        print(f"Page {page_num} does not exist in {pdf_file_name}.")

# Save the combined PDF to a new file
output_pdf_name = input("Set the name of the output PDF file: ")
with open(output_pdf_name, 'wb') as output:
    pdf_writer.write(output)

print(f"Combined PDF saved as '{output_pdf_name}'")
```

Output:
Enter the number of PDF documents to combine: 2
Enter the name of PDF file 1: m1.pdf
Enter the page number for PDF file 1: 2
Enter the name of PDF file 2: m2.pdf
Enter the page number for PDF file 2: 4
Set the name of the output PDF file: merged.pdf
Combined PDF saved as 'merged.pdf'

10 B) Write a python program to fetch current weather data from the JSON file.

```
# 10 B

import json

# Specify the path to your JSON file
json_file_path = 'weather_data.json'

try:
    # Open and read the JSON file
    with open("weather_data.json", 'r') as json_file:
        # Load the JSON data
        weather_data = json.load(json_file)

    # Extract and display the current weather information
    if 'current_weather' in weather_data:
        current_weather = weather_data['current_weather']
        print("Current Weather:")
        print(f"Weather Condition: {current_weather.get('condition', 'N/A')}")
        print(f"Temperature: {current_weather.get('temperature', 'N/A')}°C")
        print(f"Humidity: {current_weather.get('humidity', 'N/A')}%")
        print(f"Wind Speed: {current_weather.get('wind_speed', 'N/A')} m/s")
        print(f"Pressure: {current_weather.get('pressure', 'N/A')} hPa")
    else:
        print("No current weather data found in the JSON file.")

except FileNotFoundError:
    print(f"File not found: {json_file_path}")
except json.JSONDecodeError as e:
    print(f"Error decoding JSON: {e}")
except Exception as e:
    print(f"An error occurred: {e}")
```

Output:
Current Weather:
Weather Condition: Sunny
Temperature: 25°C
Humidity: 60%
Wind Speed: 5 m/s
Pressure: 1013 hPa

This is the data for weather_data.json file

```
{
  "current_weather": {
     "condition": "Sunny",
     "temperature": 25,
     "humidity": 60,
     "wind_speed": 5,
     "pressure": 1013
  }
}
```

OR

Note : In another way to fetch current weather data from the JSON file we can implement as follows :

```
# 10>B
import json

# Load the JSON data from file
with open('weather_data.json') as f:
    data = json.load(f)

# Extract the required weather data
current_temp = data['main']['temp']
humidity = data['main']['humidity']
weather_desc = data['weather'][0]['description']

# Display the weather data
print(f"Current temperature: {current_temp}°C")
print(f"Humidity: {humidity}%")
print(f"Weather description: {weather_desc}")
```

Output:
Current temperature: 15.45°C
Humidity: 64%
Weather description: clear sky

This is the data for weather_data.json file

```
{
 "weather": [
  {
   "main": "Clear",
   "description": "clear sky"
  }
 ],

 "main": {
  "temp": 15.45,
  "pressure": 1017,
  "humidity": 64
 }
 }
```