Practical 1

#WAP to read and display the following information. Name, Address, Phone no.

name = input("Name: ")

address = input("Address: ")

phone\_no = input("Phone No: ")

print(name)

print(address)

print(phone\_no)

Practical 2

#WAP to read two numbers from the keyboard and display the larger one on the screen.

num1 = *int*(input("Enter First Number:"))

num2 = *int*(input("Enter Second Number:"))

if (num1 > num2):

    print(num1, " is greater!")

else:

    print(num2, " is greater!")

Practical 3

#WAP to find, a given number is PRIME or NOT.

num = *int*(input("Enter Number: "))

is\_prime = True

for i in range(2, num):

    if(num%i==0):

        is\_prime = False

        print(i,"\*",num//i,"=",num)

        print(num,"is not prime!")

        break

if(is\_prime):

    print(num,"is prime!")

Practical 4

#Write a Function to swap values of a pair of integers.

*def* swap(*num1*, *num2*):

    temp = num1

    num1 = num2

    num2 = temp

    return num1, num2

num1 = *int*(input("Enter the First Number: "))

num2 = *int*(input("Enter the Second Number: "))

print("Before Swap:", num1,num2)

num1,num2 = swap(num1,num2)

print("After Swap:", num1,num2)

Practical 5

#WAP to print Fibonacci series of n numbers, where n is given by the programmer.

*def* fibonacci(*num*):

    if(num<=1):

        return 1

    return fibonacci(num-1) + fibonacci(num-2)

*def* fibonacci\_series(*num*):

    for i in range(num+1):

        print(fibonacci(i), *end*=" ")

num = *int*(input("Enter the Term: "))

fibonacci\_series(num)

Practical 6

#WAP to print Fibonacci series of n numbers, where n is given by the programmer.

*def* fibonacci(*num*):

    if(num<=1):

        return 1

    return fibonacci(num-1) + fibonacci(num-2)

*def* fibonacci\_series(*num*):

    for i in range(num+1):

        print(fibonacci(i), *end*=" ")

num = *int*(input("Enter the Term: "))

fibonacci\_series(num)

Practical 7

#WAP to read a set of numbers in an array & to find the largest of them.

*def* largest(*arr*):

    i = 0

    for elm in arr:

        if i < elm:

            i = elm

    return i

arr = [3,5,6,18,-11,75]

print(largest(arr))

Practical 8

#WAP to sort a list of names in ascending order

names = input("Enter: ")

names = names.split()

names.sort()

for name in names:

    print(name)

Practical 9

#WAP to read a set of numbers from keyboard & to find the sum of all elements of the given array using a function.

*def* sum\_of\_all(*arr*):

    result = 0

    for elm in arr:

        result+=elm

    return result

size = *int*(input("Size: "))

arr = []

for i in range(size):

    arr.append(*int*(input("Enter Value: ")))

print(sum\_of\_all(arr))

Practical 10

"""Calculate area of different

geometrical figures (circle,

rectangle, square, and triangle).

"""

import math

*def* area\_of\_circle(*radius*):

    return math.pi\*radius\*radius

*def* area\_of\_rect(*length*, *width*):

    return length\*width

*def* area\_of\_square(*side*):

    return side\*side

*def* area\_of\_triangle(*base*, *height*):

    return base\*height/2

option = *int*(input("Whose area do you want to calculate?\n1 Circle\n2 Rectangle\n3 Square\n4 Triangle\n"))

if(option == 1):

    print(area\_of\_circle(*int*(input("Radius:"))))

elif(option == 2):

    print(area\_of\_rect(*int*(input("Length:")), *int*(input("Width:"))))

elif(option == 3):

    print(area\_of\_square(*int*(input("Side:"))))

elif(option == 4):

    print(area\_of\_triangle(*int*(input("Base:")), *int*(input("Height:"))))

else:

    print("Error: Input between 1-4")

Practical 11

"""WAP to increment the employee

salaries on the basis of their

designation. Use employee name,

id, designation and salary as data

member and inc\_sal as member

function"""

*class* Employee:

*def* \_\_init\_\_(*self*, *name*, *id*, *designation*, *salary*):

        self.name = name

        self.id = id

        self.designation = designation

        self.salary = salary

*def* \_\_str\_\_(*self*):

        return *f*'{self.name} is {self.designation}'

*def* inc\_sal(*self*):

        print("Salary Increment")

        if(self.designation == "Manager"):

            self.salary\*=1.5

        elif(self.designation == "Senor Manager"):

            self.salary\*=2

        else:

            self.salary\*=1.1

*def* get\_salary(*self*):

        print(self.name,"has",self.salary)

e = Employee("Ravi", 1, "Manager", 110000)

print(e)

e.get\_salary()

e.inc\_sal()

e.get\_salary()

Practical 12

"""Create two classes namely Employee and Qualification. Using

multiple inheritance derive two classes Scientist and Manager. Take

suitable attributes & operations.

WAP to implement this class

hierarchy.

"""

*class* Employee:

*def* get\_data(*self*, *name*, *designation*):

        self.name = name

        self.designation = designation

*def* data\_info(*self*):

        return *f*'{self.name} is {self.designation}'

*class* Qualification:

*def* get\_degree(*self*, *degree*):

            self.degree = degree

*def* degree\_info(*self*):

            return *f*'Completed {self.degree}'

*class* Scientist(*Employee*,*Qualification*):

*def* get\_data(*self*, *name*):

        self.name = name

        self.designation = "Scientist"

    pass

*class* Manager(*Employee*,*Qualification*):

*def* get\_data(*self*, *name*):

        self.name = name

        self.designation = "Manager"

    pass

s = Scientist()

s.get\_data("Ravi")

s.get\_degree("PhD")

print(s.data\_info(), s.degree\_info())

m = Manager()

m.get\_data("Rajiv")

m.get\_degree("MBA")

print(m.data\_info(), m.degree\_info())

Practical 13

"""WAP to read data from keyboard &

write it to the file. After writing is

completed, the file is closed. The

program again opens the same file

and reads it.

"""

text\_file = open("readme.txt","w")

text\_file.write("Hello\n")

text\_file.write("I'm Learning Python\t")

text\_file.write("Bye!")

text\_file.close()

file = open("readme.txt","r")

for line in file:

    print(line)