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)
```