1. Installing python and setting up environment. Simple statements like printing the names (“Hello World”), numbers, mathematical calculations, etc.

```
print("HELLO WORLD")
print("ARITHEMETIC CALCULATIONS")
print(5+8)    #ADDITION
print(10-5)   #SUBTRACTION
print(4*6)    #MULTIPLICATION
print(8/2)    #DIVISION
print(8%2)    #MODULAS
print(3**2)   #EXPONENTIAL
print(23//2)  #FLOOR DIVISION
```

2. Write a program to find all prime numbers within a given range.

```
start=int(input("Enter the start number"))
end=int(input("Enter the end numbers"))
if start<=end:
    print("Prime numbers between",start,"end",end)
    for num in range(start,end + 1):
      count=0
      for i in range(1,num + 1):
            if num%i==0:
              count=count+1
            if count==2:
              print(num)
else:
    print("Start must be less than end")
```

3. Write a program to print "n" terms of Fibonacci Series using Iteration.

```
n=int(input("enter the number of terms:"))

#first two terms
a=0
b=1

#print fibonacci series
print("fibonacci series:")
for i in range(n):
    print(a, end=" ")
    c = a + b
    a = b
    b = c
```

4. Write a program to demonstrate the use of slicing in string.

```
#Demonstration of Slicing in String

#Original string
text = "Python Programming"

#Display original string
print("Original String:", text)

#Slice from index 0 to 6 (excluding 6)
print("Slice [0:6]:", text [0:6])       #Output: Python

#Slice from index 7 to end
print("Slice [7:]:", text [7:])         #Output: Programming

#Slice from beginning to index 6
print("Slice [:6]:", text[:6])          #Output: Python

#Slice using negative index
print("Slice [-11:-1]:", text [-11:-1])  #output: programmin


# Full string using slicing
print("Full string (:::", text[::])     #Output: Python Programming

#Reverse the string
```

```
print("Reversed String [::-1]:", text[::-1])  #Output: gnimmargorP nohtyP
```

 # Skip characters (step size)

```
print("Every second character [::2]:", text[::2]) #Output: Pto rgamn
```

 # Output: Pto rgamn

5. Write a Program related to Functions & Modules

#demonstrate the use of function and module in python

 # mymath.py

```
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply (a, b):
    return a * b

def divide(a, b):
   if b != 0:
      return a/b
   else:
      return "Division by zero not allowed"#main.py
import mymath
x=20
y=10
print("x=",x,"y=",y)
print("Addition:",mymath.add(x,y))
print("Multiplication:",mymath.multiply(x,y))
print("Subtraction:",mymath.subtract(x,y))
print("Division;",mymath.divide(x,y))
```

6. Write a program to demonstrate the use of list & related functions.

```
#Creating a list using list() constructor

t1=(10, 25, 5, 15, 30)

numbers=list(t1)

print("Original list of numbers:", numbers)

#len: Get the number of items in the list

print("Length of list:", len(numbers))

#max(): Get the maximum value

print("Maximum value:", max(numbers))

#min): Get the minimum value

print("Minimum value:", min(numbers))

#sum): Get the sum of all items

print("Sum of all elements:", sum(numbers))

#sorted(): Return a new sorted list (does not modify the original)

sorted_numbers = sorted(numbers)

print("Sorted list (ascending):", sorted_numbers)
```

```
#Show that original list is unchanged

print("Original list after sorted():", numbers)

#Now show sort) which modifies the list

numbers.sort()

print("List after sort():", numbers)

#Demonstrate reverse()

numbers.reverse()
print("List after reverse():", numbers)

#Demonstrate append(), extend(), insert()

numbers.append(40)

print("After append(40):", numbers)

numbers.extend([50, 60])

print("After extend([50, 60]):", numbers)

numbers.insert(2, 35)

print("After insert(2, 35):", numbers)

#Demonstrate remove(), pop(), index(), count()

numbers.remove(10) # Remove first occurrence of 10
```

```
print("After remove(10):", numbers)

popped = numbers.pop() # Remove and return last item

print("After pop():", numbers)

print("Popped item:", popped)

index_25=numbers.index(25)

print("Index of 25:", index_25)

count_30=numbers.count(30)

print("Count of 30:", count_30)

# Copy and clear

numbers_copy = numbers.copy()

print("Copy of list:", numbers_copy)

numbers.clear()

print("List after clear():", numbers)
```

7. Write a program to demonstrate the use of Dictionary & related functions.

```
#dictionary example i python

#creating a dictionary with your details

student={
    "name":"Harshal",
    "age":20,
    "course":"BCA",
    "grades":[95,92,95]
      }

#display the dictionary
print("original dictionary:")
print(student)


#Accessing values
print("\naccessing individual items:")
print("Name:",student["name"])
print("Course:",student.get("course"))  #safer way


#listing key values and items
print("\ndictionary keys:",list(student.keys()))
print("dictionary values :",list(student.values()))
print("dictionary items(key-valuepairs):",list(student.items()))
```

```
#adding or updating on item
student["age"]=21  #update
student["university"]="NMU UNIVERSITY"
print("\ndictionary after adding /updating items:")
print(student)

#using update()to merge another dictionary
additional_info={"graduated":False,"semester":3}
student.update(additional_info)
print("\nafter using update():")
print(student)


#removing items
student.pop("semester")  #remove by key
print("\nafter pop('semester'):",student)

student.popitem() #removes the last inserted item
print("\nafter popitem():",student)


#Iterating through dictionary
print("\niterating through bdictionary:")
for key,value in student.items():
   print(f"{key}=>{value}")
```

```
#copying the dictionry
student_copy=student.copy()
print("\ncopied dictonary:")
print(student_copy)

#clearing all items
student.clear()
print("\n after clear():",student)


#deleting the dictionary
del student
print("\n original dictionary deleted.")
```

8. Write a program to demonstrate the use of Tuple.

```
#=== Tuple Declaration ===

my_tuple=(10, 20, 30, 40, 50, 20)
print("my_tuple=", my_tuple)

print("\n=== Accessing Elements ===")
print("First element (index 0);", my_tuple [0])
print("Third element (index 2):", my_tuple [2])
print("Last element (index -1):", my_tuple[-1])

print("\n=== Slicing ===")
print("Elements from index 1 to 3:", my_tuple [1:4])
print("First 3 elements:", my_tuple[:3])
print("Last 3 elements:", my_tuple[-3:])

print("\n=== Tuple Methods ===")
print("Count of 20:", my_tuple.count(20))
print("Index of 30:", my_tuple.index(30))
print("\n=== Tuple Operations ===")

#Concatenation
tuple2=(60, 70)
```

 # how many times 20 appears #first index where 30 appears

```
concatenated= my_tuple + tuple2
print("Concatenated tuple (my_tuple + tuple2):", concatenated)
```

#Repetition

```
repeated= my_tuple * 2
print("Repeated, tuple (my_tuple * 2):", repeated)
```

#Membership test

```
print("Is 40 in my_tuple?", 40 in my_tuple)
print("Is 20 not in my_tuple?", 20 not in my_tuple)
```

9. Write a program to demonstrate the working of Class and Objects.

 # Example: Demonstrating Class and Objects in Python

```
#Define a class
class Student:

    def __init__(self, name, roll_no, marks):

        self.name = name
 self.roll_no = roll_no
self.marks = marks



    def display_info(self):

        print("Student Name:", self.name)

        print("Roll Number:", self.roll_no)

        print("Marks:", self.marks)



    def check_result(self):
        if self.marks >= 40:
```

```
            print(self.name, "has Passed")
        else:
            print(self.name, "has Failed")

# Create objects of the Student class
student1 = Student("Harshal", 101, 85)
student2 = Student("Kunal", 102, 35)

# Access methods using objects
print("-- Student 1 ----")
student1.display_info()
student1.check_result()

print("\n---- Student 2 ----")
student2.display_info()
student2.check_result()
```

10. Write a program to demonstrate the working of Overloading Methods.

```
import math
class Shape:
    def area (self):
      print("Area is not defined for generic shape.")


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


     def area (self):
        circle_area= math.pi*     self.radius ** 2
        print (f"Area of Circle with radius (self.radius):
(circle_area:.2f)")


class Rectangle (Shape):
     def __init__(self, width, height):
        self.width = width
        self.height = height
def area (self):
        rectangle_area = self.width* self.height
        print (f"Area of Rectangle ((self.width) x (self.height)):
(rectangle_area)")
        self.radius = radius
```

```
    def area (self):
        circle_area = math.pi * self.radius ** 2
        print (f"Area of Circle with radius (self.radius): (circle
area:.21)")
class Rectangle (Shape):
    def __init__(self,width,height):
        self.width=width
        self.height = height

    def area(self):
        rectangle_area= self.width* self.height
        print (f"Area of Rectangle ((self.width) x (self.height)):
(rectangle_area)")

shape= Shape ()
circle= Circle (5)
rectangle= Rectangle (4, 6)

#Call area method
shape.area ()              #Base class method

#Base class method
circle.area()              #Overridden in circle

# Overridden in Circle
rectangle.area()           #Overridden in rectangle
```

11..Write a program to demonstrate the working of libraries.

```
#numpy

import numpy as np

# Creating a 2D array

arr = np.array([[1, 2, 3], [4, 5, 6]])

# Sum of all elements

total = np.sum(arr)

print("Array:\n", arr)

print("Sum of elements:", total)


#matplotlib

import matplotlib.pyplot as plt

# Simple line plot

x = [2021, 2022, 2023, 2024, 2025]

y = [100, 150, 130, 170, 200]

plt.bar(x, y, color='skyblue')

plt.title('Simple Line Plot')
```

```
plt.xlabel('Years')

plt.ylabel('No of units sale')

for year, units in zip(x, y):

    plt.text(year, units + 3, str(units), ha='center')

plt.show()
```