Practical No:1

```
# Demonstrating Basic Data Types
integer_num = 20
float_num = 3.42
string_text = "Hello, Python!"
boolean_value = True

# Displaying the types of variables
print("Integer:", integer_num, "Type:", type(integer_num))
print("Float:", float_num, "Type:", type(float_num))
print("String:", string_text, "Type:", type(string_text))
print("Boolean:", boolean_value, "Type:", type(boolean_value))

# Demonstrating Data Structures
li = [10, 20, 30, 40, "Python"] # List
tup = (1, 2, 3, "Immutable") # Tuple
s = {10, 20, 30, 10} # Set (duplicates removed)
dict = {"name": "Tanmay", "age": 20, "city": "Vasai"} # Dictionary

# Displaying Data Structures
print("List:", li)
print("Tuple:", tup)
print("Set:", s)
print("Dictionary:", dict)

# Arithmetic Operators
a = 25
b = 5
sum = a + b
diff = a - b
prod = a * b
```

```
div = a / b
mod = a % b

print("Sum:", sum)
print("Difference:", diff)
print("Product:", prod)
print("Division:", div)
print("Modulus:", mod)

# Logical Operators
x = True
y = False
print("x and y:", x and y)
print("x or y:", x or y)
print("not x:", not x)

# Taking User Input
name = input("Enter your name: ")
age = int(input("Enter your age: ")) # Converting input to integer
# Displaying Output
print("Hello,", name, "You are", age, "years old.")
```

Output:

Integer: 20 Type: <class 'int'>

Float: 3.42 Type: <class 'float'>

String: Hello, Python! Type: <class 'str'>

Boolean: True Type: <class 'bool'>

List: [10, 20, 30, 40, 'Python']

Tuple: (1, 2, 3, 'Immutable')

Set: {10, 20, 30}

Dictionary: {'name': 'Tanmay', 'age': 20, 'city': 'Vasai'}

Sum: 30

Difference: 20

Product: 125

Division: 5.0

Modulus: 0

x and y: False

x or y: True

not x: False

Enter your name: Tanmay

Enter your age: 20

Hello, Tanmay You are 20 years old.

Practical No:2

```
# Example 1: Checking if a number is positive
num1 = int(input("Example 1 - Enter a number : "))
if num1 > 0:
 print("Example 1 Output: The number is positive")

# Example 2: Checking even or odd using if-else
num2 = int(input("Example 2 - Enter a number : "))
if num2 % 2 == 0:
 print("Example 2 Output: The number is even")
else:
 print("Example 2 Output: The number is odd")

# Example 3: Nested if statement (positive even, positive odd, negative, zero)
num3 = int(input("Example 3 - Enter a number: "))
if num3 > 0:
 if num3 % 2 == 0:
  print("Example 3 Output: It is a positive even number")
 else:
  print("Example 3 Output: It is a positive odd number")
elif num3 < 0:
 print("Example 3 Output: It is a negative number")
else:
 print("Example 3 Output: The number is zero")

# Example 4: Checking age group using if-elif-else
age = int(input("Example 4 - Enter your age: "))
if age < 13:
 print("Example 4 Output: You are a child")
elif age >= 13 and age < 20:
 print("Example 4 Output: You are a teenager")
```

```
elif age >= 20 and age < 60:
 print("Example 4 Output: You are an adult")
else:
 print("Example 4 Output: You are a senior citizen")

# Example 5: Checking grade based on marks
marks = int(input("Example 5 - Enter your marks: "))
if marks >= 90:
 print("Example 5 Output: Grade A")
elif marks >= 80:
 print("Example 5 Output: Grade B")
elif marks >= 70:
 print("Example 5 Output: Grade C")
elif marks >= 60:
 print("Example 5 Output: Grade D")
else:
 print("Example 5 Output: Grade F (Fail)")
```

Output:

Example 1 - Enter a number : 2

Example 1 Output: The number is positive

Example 2 - Enter a number : 4

Example 2 Output: The number is even

Example 3 - Enter a number: 6

Example 3 Output: It is a positive even number

Example 4 - Enter your age: 20

Example 4 Output: You are an adult

Example 5 - Enter your marks: 72

Example 5 Output: Grade C

Practical No:3

```
x = 1
while x <= 5:
 print("While Loop Output:", x)
 x += 1
for num in range(1, 8):
 print("For Loop Output:", num)
#Sum of first n natural numbers
x = 1
sum_natural = 0
while x <= 10:
  sum_natural += x
  x += 1
print("Sum of first 10 natural numbers:", sum_natural)
#Iterating through list
fruits = ["Mango", "Graper", "Apple"]
for fruit in fruits:
 print("For Loop (List) Output:", fruit)
# Using break
for num in range(1, 10):
 if num == 7:
  break # Stops loop when num == 4
 print("Break Example Output:", num)
# Using continue
for num in range(1, 10):
 if num == 7:
  continue # Skips printing 3
 print("Continue Example Output:", num)
```

Output:

While Loop Output: 1

While Loop Output: 2

While Loop Output: 3

While Loop Output: 4

While Loop Output: 5

For Loop Output: 1

For Loop Output: 2

For Loop Output: 3

For Loop Output: 4

For Loop Output: 5

For Loop Output: 6

For Loop Output: 7

Sum of first 10 natural numbers: 55

For Loop (List) Output: Mango

For Loop (List) Output: Graper

For Loop (List) Output: Apple

Break Example Output: 1

Break Example Output: 2

Break Example Output: 3

Break Example Output: 4

Break Example Output: 5

Break Example Output: 6

Continue Example Output: 1

Continue Example Output: 2

Continue Example Output: 3

Continue Example Output: 4

Continue Example Output: 5

Continue Example Output: 6

Continue Example Output: 8

Continue Example Output: 9

Practical No:4

```
# **List Operations**
print("=== List Operations ===")
my_list = [1, 2, 3, 4]
my_list.append(7)
print("After append:", my_list)

my_list.insert(2, 29)
print("After insert at index 2:", my_list)

my_list.remove(3)
print("After removing 3:", my_list)

print("Popped element:", my_list.pop())
print("List after pop:", my_list)

my_list.sort()
print("Sorted List:", my_list)

my_list.reverse()
print("Reversed List:", my_list)
print("Length of List:", len(my_list))

# **Tuple Operations**
print("\n=== Tuple Operations ===")
my_tuple = (7, 14, 21, 28, 35)
print("Tuple:", my_tuple)
print("Count of 21:", my_tuple.count(21))
print("Index of 28:", my_tuple.index(28))
print("Maximum element:", max(my_tuple))
print("Minimum element:", min(my_tuple))
```

```
print("Length of Tuple:", len(my_tuple))

# **Set Operations**
print("\n=== Set Operations ===")
set1 = {7, 8, 9, 10}
set2 = {9, 10, 11, 12}
set1.add(6)
print("Set after adding 6:", set1)

set1.remove(8)
print("Set after removing 8:", set1)

print("Union of sets:", set1.union(set2))
print("Intersection of sets:", set1.intersection(set2))
print("Difference (set1 - set2):", set1.difference(set2))
print("Is set1 a subset of set2?:", set1.issubset(set2))

# **String Operations**
print("\n=== String Operations ===")
my_string = " Hello, Python! "
print("Original String:", my_string)
print("Uppercase:", my_string.upper())
print("Lowercase:", my_string.lower())
print("Title Case:", my_string.title())
print("Stripped String:", my_string.strip())
print("Replace 'Python' with 'World':", my_string.replace("Python", "World"))
print("Find 'Python':", my_string.find("Python"))
split_string = my_string.split(",")
print("Split String:", split_string)
joined_string = " - ".join(split_string)
print("Joined String:", joined_string)
```

Output:

=== List Operations ===

After append: [1, 2, 3, 4, 7]

After insert at index 2: [1, 2, 29, 3, 4, 7]

After removing 3: [1, 2, 29, 4, 7]

Popped element: 7

List after pop: [1, 2, 29, 4]

Sorted List: [1, 2, 4, 29]

Reversed List: [29, 4, 2, 1]

Length of List: 4

=== Tuple Operations ===

Tuple: (7, 14, 21, 28, 35)

Count of 21: 1

Index of 28: 3

Maximum element: 35

Minimum element: 7

Length of Tuple: 5

=== Set Operations ===

Set after adding 6: {6, 7, 8, 9, 10}

Set after removing 8: {6, 7, 9, 10}

Union of sets: {6, 7, 9, 10, 11, 12}

Intersection of sets: {9, 10}

Difference (set1 - set2): {6, 7}

Is set1 a subset of set2?: False

=== String Operations ===

Original String: Hello, Python!

Uppercase: HELLO, PYTHON!

Lowercase: hello, python!

Title Case: Hello, Python!

Stripped String: Hello, Python!

Replace 'Python' with 'World': Hello, World!

Find 'Python': 8

Split String: [' Hello', ' Python! ']

Joined String: Hello - Python!

Practical No:5

#WAP: To understand basic array operations(1-D and Multidimensional)using NumPy

```
import array as arr
a = arr.array('i', [10, 20, 30, 40])
print(type(a))
for i in a:
 print(i)
print(a)
a = arr.array('f', [7, 8, 9, 10])
print(a)
a.append(2.0)
print(a)
a.reverse()
print(a)
```

#Output:

```
10
20
30
40
array('i', [10, 20, 30, 40])
array('f', [7.0, 8.0, 9.0, 10.0])
array('f', [7.0, 8.0, 9.0, 10.0, 2.0])
array('f', [2.0, 10.0, 9.0, 8.0, 7.0])
```

```
import numpy as np
l1=[[10,20,30],[40,50,60],[70,80,90]]
a1=np.array(l1)
print(a1[1:3,1])
print(a1[1:3,:1])
```

#Output

[50 80]

[[40]

[70]]

```
import array as arr
a=arr.array('i',[])
n=int(input("Enter the array size:"))
for x in range(n):
 e=int(input("enter the element:"))
a.append(e)
print(a)
```

```
import array as arr
a=arr.array('i',[])
n=int(input("Enter the array size:"))
for x in range(n):
 e=int(input("enter the element:"))
 a.append(e)
 print(sum(a))
```

#output

```
Enter the array size:5
enter the element:20
enter the element:52
enter the element:36
enter the element:73
enter the element:84
array('i', [20, 52, 36, 73, 84])
```

```
import array as arr
a=arr.array('i',[])
```

```
n=int(input("Enter the array size:"))
for x in range(n):
e=int(input("enter the element:"))
a.append(e)
print(sum(a))
```

#Output:

```
Enter the array size:5
enter the element:13
enter the element:17
enter the element:35
enter the element:16
enter the element:82
163
```

```
import numpy as np
l1=[[10,20,30],[40,50,60],[70,80,90]]
a1=np.array(l1)
print(a1[:,1])
```

#Output:

```
[20 50 80]
import numpy as np
l1=[[10,20,30],[40,50,60],[70,80,90]]
a1=np.array(l1)
print(a1)
a2=np.arange(11,17)
print(a2)
a1=np.ones((3,4))
print(a1)
```

#Output:

[[10 20 30]

[40 50 60]

[70 80 90]]

[11 12 13 14 15 16]

[[1. 1. 1. 1.]

[1. 1. 1. 1.]

[1. 1. 1. 1.]]

```
import numpy as np
x=np.array([[1,2],
[3,4]])
y=np.array([[11,12],
[13,14]])
print(x,y)
z=x+y
print(z)
```

#Output:

[[1 2]

[3 4]] [[11 12]

[13 14]]

[[12 14]

[16 18]]

```
import numpy as np
n=int(input("Enter size of an array"))
l1=[]
for x in range(n):
e=int(input("Enter the value"))
l1.append(e)
```

```
a3=np.array(l1)
print(np.flip(a3))
```

#Output:

```
Enter size of an array: 4
Enter the value: 2
Enter the value: 6
Enter the value: 8
Enter the value: 9
[9 8 6 2]
```

Practical No:6

```
from functools import reduce

num = [11, 12, 13, 14, 15, 16, 17, 18, 19, 20]


power4 = list(map(lambda x: x ** 4, num))

print("Numbers^4:", power4)


multiple3 = list(filter(lambda x: x % 3 == 0, num))

print("Numbers Multiple of Three:", multiple3)


sum = reduce(lambda x, y: x + y, num)

print("Sum of Numbers:", sum)
```

Output

```
Numbers^4: [14641, 20736, 28561, 38416, 50625, 65536, 83521, 104976, 130321, 160000]

Numbers Multiple of Three: [12, 15, 18]

Sum of Numbers: 155
```

Practical No:7

```
class Employee:
    def __init__(self, Eid, Name, Address, MobileNo):
        self.Eid = Eid
        self.Name = Name
        self.Address = Address
        self.MobileNo = MobileNo
    def info(self):
        print("Employee Details:")
        print(f"Employee ID: {self.Eid}")
        print(f"Name: {self.Name}")
        print(f"Address: {self.Address}")
        print(f"Mobile Number: {self.MobileNo}")
# Creating an object for Employee
Emp1 = Employee(416, "Tanmay", "Vasai", "9451476325")
# Displaying employee info
Emp1.info()
```

Output:

```
Employee Details:
Employee ID: 416
Name: Tanmay
Address: Vasai
Mobile Number: 9451476325
```

Practical No: 8

Single inheritance:

```
class Parent:
    def display(self):
        print("This is the parent class.")

class Child(Parent):
    def show(self):
        print("This is the child class.")

obj = Child()
obj.display()
obj.show()
```

Output:

This is the parent class.
This is the child class.

Multiple Inheritance:

```
class Father:
    def skills(self):
        print("Father: Gardening")

class Mother:
    def skills(self):
        print("Mother: Cooking")

class Child(Father, Mother):
    def skills(self):
        Father.skills(self)
        Mother.skills(self)
        print("Child: Drawing")

obj = Child()
obj.skills()
```

Output:

Father: Gardening
Mother: Cooking
Child: Drawing

Multilevel Inheritance:

```
class Grandparent:
    def house(self):
        print("Grandparent has a house.")

class Parent(Grandparent):
    def car(self):
        print("Parent has a car.")

class Child(Parent):
    def bike(self):
```

```
        print("Child has a bike.")

obj = Child()
obj.house()
obj.car()
obj.bike()
```

Output:

```
Grandparent has a house.
Parent has a car.
Child has a bike.
```

Hierarchical Inheritance:

```
class Parent:
    def show(self):
        print("This is the parent class.")

class Child1(Parent):
    def child1_func(self):
        print("This is child 1.")

class Child2(Parent):
    def child2_func(self):
        print("This is child 2.")

obj1 = Child1()
obj1.show()
obj1.child1_func()

obj2 = Child2()
obj2.show()
obj2.child2_func()
```

Output:

```
This is the parent class.
This is child 1.
This is the parent class.
This is child 2.
```

Hybrid Inheritance:

```
class A:
    def methodA(self):
        print("Method A")

class B(A):
    def methodB(self):
        print("Method B")

class C(A):
    def methodC(self):
        print("Method C")
```

```
class D(B, C):  # Hybrid: Multilevel + Multiple
    def methodD(self):
        print("Method D")

obj = D()
obj.methodA()
obj.methodB()
obj.methodC()
obj.methodD()
```

Output:

```
Method A
Method B
Method C
Method D
```

Practical No:9

Method Overloading:

```
class Calculator:
   def add(self, a=None, b=None, c=None):
      if a is not None and b is not None and c is not None:
         print("Sum of 3 numbers:", a + b + c)
      elif a is not None and b is not None:
         print("Sum of 2 numbers:", a + b)
      else:
         print("Insufficient arguments")

calc = Calculator()
calc.add(5, 10)
calc.add(3, 6, 9)
calc.add(7)
```

Output:

```
Sum of 2 numbers: 15
Sum of 3 numbers: 18
Insufficient arguments
```

Method Overriding:

```
class Animal:
   def speak(self):
      print("Animal speaks")

class Dog(Animal):
   def speak(self):
      print("Dog barks")

animal = Animal()
animal.speak()

dog = Dog()
dog.speak()
```

Output:

```
Animal speaks
Dog barks
```

Practical No:10

User-Defined Module

```
#creating User-Defined Module(mymath.py)

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

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

#Main Program
import mymath

x = 10
y = 5

print("Addition:", mymath.add(x, y))
print("Subtraction:", mymath.subtract(x, y))
```

Output:

```
Addition: 15
Subtraction: 5
```

User-Defined Package

```
#creating User-Defined Package
clac.py
def multiply(a, b):
   return a * b

def divide(a, b):
   return a / b

greeting.py
def say_hello(name):
   return f"Hello, {name}!"

__init__.py
# This can be left empty or used to import specific functions

main.py

from my_package import calc, greeting

print("Multiplication:", calc.multiply(4, 5))
print("Division:", calc.divide(20, 4))

print(greeting.say_hello("Alice"))
```

Output:

```
Multiplication: 20
Division: 5.0
Hello, Alice!
```

Practical No:11

```
import threading
import time
counter = 0
lock = threading.Lock()
def increment(name):
   global counter
   for i in range(5):
      time.sleep(1)
      lock.acquire()
      try:
         temp = counter
         print(f"[{name}] Read counter: {temp}")
         temp += 1
         print(f"[{name}] Updated counter to: {temp}")
         counter = temp
      finally:
         lock.release()

thread1 = threading.Thread(target=increment, args=("Thread-1",))
thread2 = threading.Thread(target=increment, args=("Thread-2",))

thread1.start()
thread2.start()

thread1.join()
thread2.join()
print(f"\nFinal counter value: {counter}")
```

Output:

```
[Thread-1] Read counter: 0
[Thread-1] Updated counter to: 1
[Thread-2] Read counter: 1
[Thread-2] Updated counter to: 2
[Thread-1] Read counter: 2
[Thread-1] Updated counter to: 3
[Thread-2] Read counter: 3
[Thread-2] Updated counter to: 4
[Thread-1] Read counter: 4
[Thread-1] Updated counter to: 5
[Thread-2] Read counter: 5
[Thread-2] Updated counter to: 6
[Thread-1] Read counter: 6
[Thread-1] Updated counter to: 7
[Thread-2] Read counter: 7
[Thread-2] Updated counter to: 8
[Thread-1] Read counter: 8
[Thread-1] Updated counter to: 9
[Thread-2] Read counter: 9
[Thread-2] Updated counter to: 10

Final counter value: 10
```

```
import threading
import time

def printTable(n):
    for i in range(1,6):
        # print(n,"x",i, "=",(n*i))
        print(f"{n} x {i} = {n*i}")
        time.sleep(0.5)

print('-'*20)

t1 = threading.Thread(target=printTable, args=(10,))
t2 = threading.Thread(target=printTable, args=(7,))

t1.start()
t2.start()
print("Multiplication table printed without synchonisation : ")
```

Output:

```
10 x 1 = 10
7 x 1 = 7
Multiplication table printed without synchonisation :
7 x 2 = 14
10 x 2 = 20
10 x 3 = 30
7 x 3 = 21
10 x 4 = 40
7 x 4 = 28
7 x 5 = 35
10 x 5 = 50
```

Practical No:12

```
def divide_numbers(a, b):
    try:
        result = a / b
        print("Result:", result)
    except ZeroDivisionError:
        print("Error: Cannot divide by zero.")
    except TypeError:
        print("Error: Please enter numbers only.")
    else:
        print("Division successful.")
    finally:
        print("Execution completed.\n")

divide_numbers(10, 2)
divide_numbers(5, 0)
divide_numbers("10", 2)
```

Output:

```
Result: 5.0
Division successful.
Execution completed.

Error: Cannot divide by zero.
Execution completed.

Error: Please enter numbers only.
Execution completed.
```

Practical No:13

```
# 1. Writing to a file
def write_to_file():
    with open("sample.txt", "w") as file:
        file.write("Hello, this is a sample file.\n")
        file.write("Writing some text here.")
    print("Data written to sample.txt")

# 2. Reading entire file
def read_file():
    with open("sample.txt", "r") as file:
        content = file.read()
    print("Reading entire file:")
    print(content)

# 3. Reading file line by line
def read_lines():
    with open("sample.txt", "r") as file:
        print("Reading line by line:")
        for line in file:
            print(line.strip())

# 4. Appending to a file
def append_to_file():
    with open("sample.txt", "a") as file:
        file.write("\nAppending new data.")
    print("Data appended to sample.txt")

# 5. Reading and writing (r+ mode)
def read_and_write():
    with open("sample.txt", "r+") as file:
        content = file.read()
        print("Before modification:")
        print(content)
        file.seek(0)
        file.write("Modified content.\n")
    print("File modified using r+ mode")

# Main execution
if __name__ == "__main__":
    print("File Handling")
    print("-" * 20)

    # Execute operations
    write_to_file()
    read_file()

    print("-" * 20)
    read_lines()

    print("-" * 20)
    append_to_file()
```

```
    read_file()

    print("-" * 20)
    read_and_write()
    read_file()
```

Output:

```
File Handling
--------------------
Data written to sample.txt
Reading entire file:
Hello, this is a sample file.
Writing some text here.
--------------------
Reading line by line:
Hello, this is a sample file.
Writing some text here.
--------------------
Data appended to sample.txt
Reading entire file:
Hello, this is a sample file.
Writing some text here.
Appending new data.
--------------------
Before modification:
Hello, this is a sample file.
Writing some text here.
Appending new data.
File modified using r+ mode
Reading entire file:
Modified content.
mple file.
Writing some text here.
Appending new data.
```

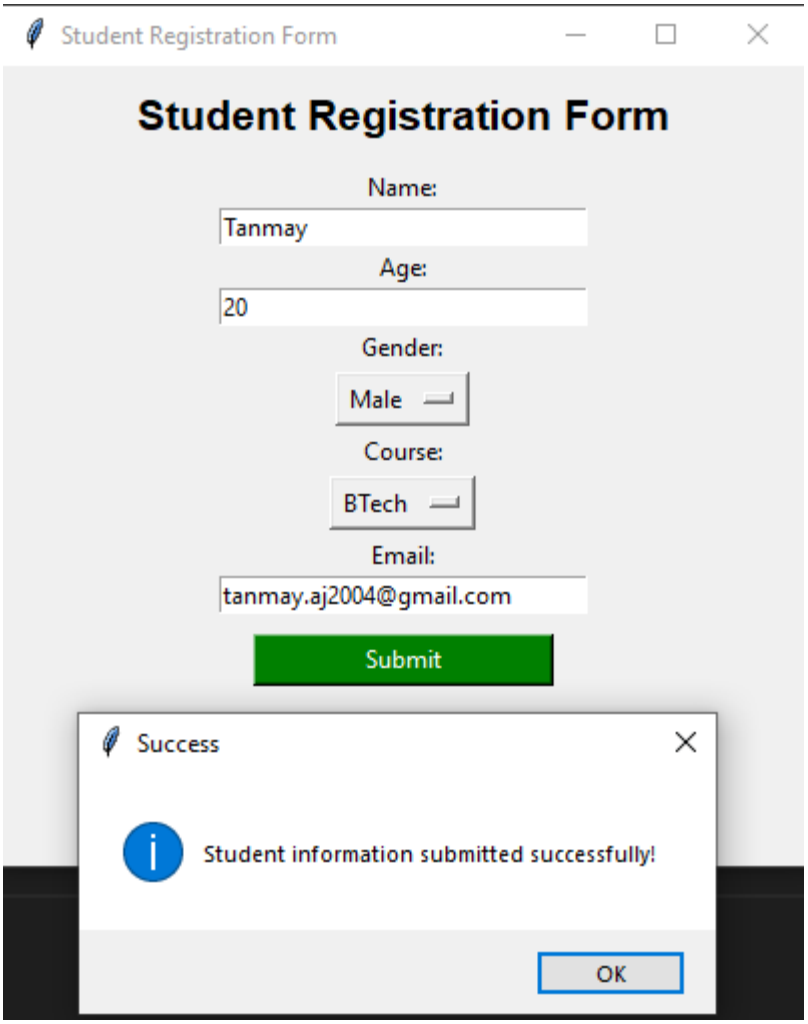
Practical No:14

```
import tkinter as tk
from tkinter import messagebox

def submit_form():
    name = entry_name.get()
    age = entry_age.get()
    gender = gender_var.get()
    course = course_var.get()
    email = entry_email.get()

    if name == "" or age == "" or email == "":
        messagebox.showwarning("Incomplete Data", "Please fill all required fields!")
    else:
        info = f"""
        Name   : {name}
        Age    : {age}
        Gender : {gender}
        Course : {course}
        Email  : {email}
        """
        print(info)
        messagebox.showinfo("Success", "Student information submitted successfully!")
        clear_form()

def clear_form():
    entry_name.delete(0, tk.END)
    entry_age.delete(0, tk.END)
    entry_email.delete(0, tk.END)
    gender_var.set("Select")
    course_var.set("Select")

root = tk.Tk()
root.title("Student Registration Form")
root.geometry("400x400")

tk.Label(root, text="Student Registration Form", font=("Arial", 16, "bold")).pack(pady=10)

tk.Label(root, text="Name:").pack()
entry_name = tk.Entry(root, width=30)
entry_name.pack()

tk.Label(root, text="Age:").pack()
entry_age = tk.Entry(root, width=30)
entry_age.pack()

tk.Label(root, text="Gender:").pack()
gender_var = tk.StringVar()
gender_var.set("Select")
gender_menu = tk.OptionMenu(root, gender_var, "Male", "Female", "Other")
gender_menu.pack()
```

```
tk.Label(root, text="Course:").pack()
course_var = tk.StringVar()
course_var.set("Select")
course_menu = tk.OptionMenu(root, course_var, "BCA", "BSc", "BCom", "BA", "BTech")
course_menu.pack()

tk.Label(root, text="Email:").pack()
entry_email = tk.Entry(root, width=30)
entry_email.pack()

tk.Button(root, text="Submit", command=submit_form, bg="green", fg="white",
width=20).pack(pady=10)

root.mainloop()
```

Output:

Name : Tanmay
Age : 20
Gender : Male
Course : BTech
Email : tanmay.aj2004@gmail.com



Practical No:15

```
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4, 5]
y = [10, 20, 15, 25, 30]

# Create a line plot
plt.plot(x, y, color='blue', marker='o', linestyle='--')

# Add labels and title
plt.title("Simple Line Plot")
plt.xlabel("X-Axis (Time)")
plt.ylabel("Y-Axis (Values)")

# Add grid
plt.grid(True)

# Display the plot
plt.show()
```

Output: