# AMITY UNIVERSITY MAHARASHTRA

Established vide Maharashtra Act No. 13 of 20 14, of Government of Maharashtra, and recognized under Section 2(t) of UGC Act 1956

# CERTIFICATE

## This is to certify that ANSARI ZEESHAN ALI Enrollment No.

A71045024017 of class **MCA, Semester I** has satisfactorily completed the **Python (Django & Kivy) Practical lab Manual** prescribed by Amity University Maharashtra during the academic year 2023-2024.

Sign of Faculty Sign of Dept. Coordinator

INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No** | **Index** | **Page No** | **Remark** |
| 1 | WAP on Basic Syntax, Mathematical Operators, and Data Types | 1 |  |
| 2 | WAP to implement Decision making using If else statement | 3 |  |
| 3 | WAP for implementation of Lists-CRUD operations | 5 |  |
| 4 | WAP for implementation of Tuples-Accessing and Operations | 8 |  |
| 5 | WAP on Dictionaries – Adding, Accessing, and Deleting Elements | 10 |  |
| 6 | WAP to find factorial of number using normal function and Lambda function | 12 |  |
| 7 | WAP to perform Simple Data Visualization with Matplotlib | 14 |  |
| 8 | Installation and Setup of Django; Creating and Running a Simple Django Project | 16 |  |
| 9 | Installing Kivy and Running a Simple "Hello World" Program | 19 |  |

**PRACTICAL-1**

**WAP ON BASIC SYNTAX, MATHEMATICAL OPERATORS, AND DATA TYPES**

**Steps to Write and Run the Program:**

**Step 1**: Open the Python IDE or text editor. Launch **Python IDLE**, **PyCharm**, **VS Code**, or any other Python editor. Alternatively, open a text editor like Notepad if you're using a command-line interpreter.

**Step 2**: Create a new file. In the editor, create a new file and save it with a .py extension (e.g., basic\_program.py).

**Step 3**: Write the code. Type the program you want to execute. For example, a simple program is given below.

**Step 4**: Save the file. After writing the code, save it to a location where you can easily find it.

**Step 5**: Run the program. If using an IDE, click the Run button. If using a command line, navigate to the file location using the terminal or command prompt, and run the program with the command:

* python basic\_program.py

**Program Code:**

# Basic Python Program with Data Types

# Integer

b = 3.5 # Float

name = "Python" # String

sum\_result = a + b

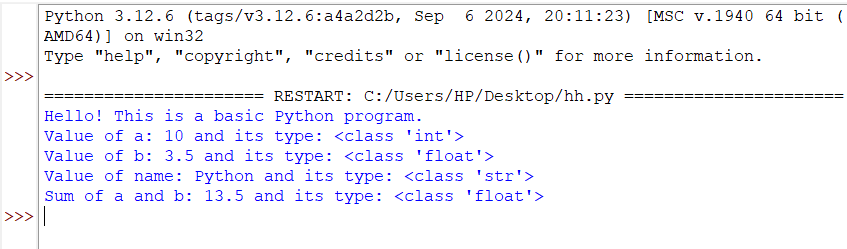
print("Hello! This is a basic Python program.")

print("Value of a:", a, "and its type:", type(a))

print("Value of b:", b, "and its type:", type(b))

print("Value of name:", name, "and its type:", type(name))

print("Sum of a and b:", sum\_result, "and its type:", type(sum\_result))

**Output:**

**PRACTICAL-2**

**WAP TO IMPLEMENT DECISION MAKING USING IF ELSE STATEMENT**

**Steps to Write and Run the Program:**

**Step 1:** Open the Python IDE or text editor.  
Launch Python IDLE, PyCharm, VS Code, or any other Python editor. Alternatively, open a text editor like Notepad if you're using a command-line interpreter.

**Step 2:** Create a new file.  
In the editor, create a new file and save it with a .py extension (e.g., decision\_making\_program.py).

**Step 3:** Write the code.  
Type the program you want to execute.

**Step 4:** Save the file.  
After writing the code, save it to a location where you can easily find it.

**Step 5:** Run the program.  
If using an IDE, click the **Run** button.  
If using a command line, navigate to the file location using the terminal or command prompt, and run the program with the command:

* python decision\_making\_program.py

**Program Code:**

# Decision Making using if-else

number = int(input("Enter a number: "))

if number > 0:

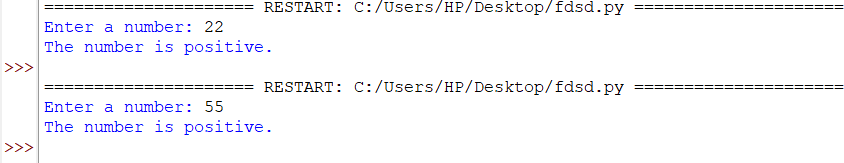
print("The number is positive.")

elif number < 0:

print("The number is negative.")

else:

print("The number is zero.")

**Output:**

**PRACTICAL-3**

**WAP FOR IMPLEMENTATION OF LISTS-CRUD OPERATIONS**

**Steps to Write and Run the Program:**

**Step 1:** Open the Python IDE or text editor (e.g., PyCharm, VS Code, or Python IDLE).

**Step 2:** Create a new file. Save it with a .py extension, such as list\_crud\_operations.py.

**Step 3:** Write the code that demonstrates CRUD (Create, Read, Update, Delete) operations.

**Step 4:** Save the file.

**Step 5:** Run the program:

* If using an IDE, click the Run button.
* If using the command line, navigate to the file's location and run:
  + - python list\_crud\_operations.py

**Program Code:**

# Simple List CRUD Operations

# Initialize an empty list

my\_list = []

# Create operation: Adding elements to the list

def create\_element():

element = input("Enter an element to add: ")

my\_list.append(element)

# Read operation: Display the list

def read\_list():

print("Current List:", my\_list)

# Update operation: Modify an element in the list

def update\_element():

index = int(input("Enter index to update: "))

if index < len(my\_list):

new\_value = input("Enter new value: ")

my\_list[index] = new\_value

else:

print("Invalid index!")

# Delete operation: Remove an element from the list

def delete\_element():

index = int(input("Enter index to delete: "))

if index < len(my\_list):

my\_list.pop(index)

else:

print("Invalid index!")

while True:

print("\n1. Add (Create)")

print("2. Display (Read)")

print("3. Update")

print("4. Delete")

print("5. Exit")

choice = input("Choose an operation: ")

if choice == '1':

create\_element()

elif choice == '2':

read\_list()

elif choice == '3':

update\_element()

elif choice == '4':

delete\_element()

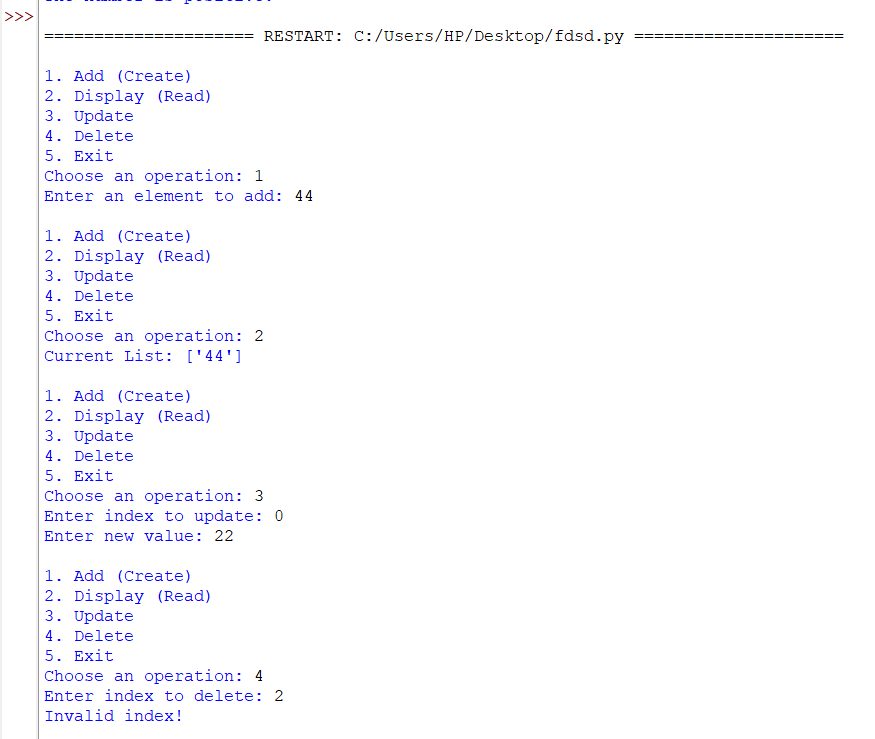
elif choice == '5':

break

else:

print("Invalid choice! Try again.")

OUTPUT –

****

**PRACTICAL-4**

**WAP FOR IMPLEMENTATION OF TUPLES-ACCESSING AND OPERATIONS**

**Steps to Write and Run the Program:**

**Step 1:** Open the Python IDE or text editor (e.g., PyCharm, VS Code, or Python IDLE).

**Step 2:** Create a new file. Save it with a .py extension, such as tuple\_operations.py.

**Step 3:** Write the code. Here's the Python program that demonstrates accessing elements and performing operations on tuples:

**Step 4:** Save the file.

**Step 5:** Run the program:

* If using an IDE, click the **Run** button.
* If using the command line, navigate to the file's location and run:
  + - python tuple\_operations.py

**Program Code:**

# Tuple Operations: Accessing and Performing Operations

# Define a tuple

my\_tuple = (1, 2, 3, 4, 5)

# Accessing elements from the tuple

print("Accessing elements from the tuple:")

print("First element:", my\_tuple[0])

print("Last element:", my\_tuple[-1])

print("Elements from index 1 to 3:", my\_tuple[1:4])

# Tuple operations

# 1. Concatenation (Joining two tuples)

tuple2 = (6, 7, 8)

concatenated\_tuple = my\_tuple + tuple2

print("\nConcatenated Tuple:", concatenated\_tuple)

# 2. Repetition (Repeating the tuple)

repeated\_tuple = my\_tuple \* 2

print("Repeated Tuple:", repeated\_tuple)

# 3. Checking if an element exists in the tuple

element = 3

print(f"\nIs {element} in the tuple?", element in my\_tuple)

# 4. Length of the tuple

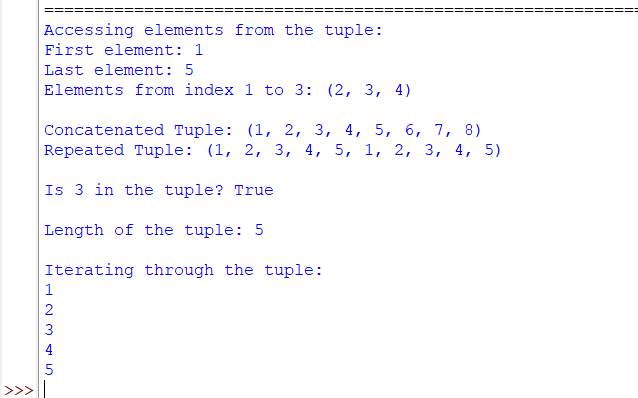
print("\nLength of the tuple:", len(my\_tuple))

# 5. Iterating through the tuple

print("\nIterating through the tuple:")

for item in my\_tuple:

print(item)

**Output:**

**PRACTICAL-5**

**WAP ON DICTIONARIES – ADDING, ACCESSING, AND DELETING ELEMENTS**

**Steps to Write and Run the Program:**

**Step 1:** Open the Python IDE or text editor (e.g., PyCharm, VS Code, or Python IDLE).

**Step 2:** Create a new file. Save it with a .py extension, such as dictionary\_operations.py.

**Step 3:** Write the code. Here is the Python program that demonstrates how to add, access, and delete elements in a dictionary:

**Step 4:** Save the file.

**Step 5:** Run the program:

* If using an IDE, click the Run button.
* If using the command line, navigate to the file's location and run

**Program Code:**

# Simple Dictionary Operations: Adding, Accessing, and Deleting Elements

# Initialize a dictionary

my\_dict = {}

# Adding elements to the dictionary

my\_dict["name"] = "John"

my\_dict["age"] = 25

my\_dict["city"] = "New York"

# Accessing elements

print("Accessing elements:")

print("Name:", my\_dict["name"])

print("Age:", my\_dict["age"])

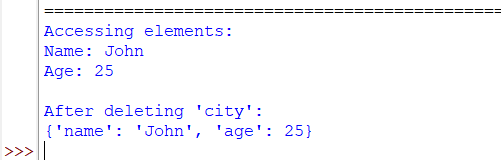
# Deleting an element

del my\_dict["city"]

print("\nAfter deleting 'city':")

print(my\_dict)

**Output:**

****

**PRACTICAL-6**

**WAP TO FIND FACTORIAL OF NUMBER USING NORMAL FUNCTION AND LAMBDA FUNCTION**

**Steps to Write and Run the Program:**

**Step 1:** Open the Python IDE or text editor (e.g., PyCharm, VS Code, or Python IDLE).

**Step 2:** Create a new file. Save it with a .py extension, such as factorial\_program.py.

**Step 3:** Write the code. Below is the Python program that demonstrates finding the factorial of a number using both a normal function and a lambda function.

**Step 4:** Save the file.

**Step 5:** Run the program:

* If using an IDE, click the **Run** button.
* If using the command line, navigate to the file's location and run

**Program Code:**

# Normal function to find factorial of a number

def factorial\_normal(n):

if n == 0 or n == 1:

return 1

result = 1

for i in range(2, n + 1):

result \*= i

return result

# Lambda function to find factorial of a number

factorial\_lambda = lambda n: 1 if n == 0 or n == 1 else n \* factorial\_lambda(n - 1)

# Input number

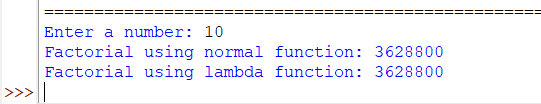
number = int(input("Enter a number: "))

# Using normal function

print("Factorial using normal function:", factorial\_normal(number))

# Using lambda function

print("Factorial using lambda function:", factorial\_lambda(number))

**Output:**

**PRACTICAL-7**

**WAP TO PERFORM SIMPLE DATA VISUALIZATION WITH MATPLOTLIB**

**Step 1:** Install Matplotlib. Open a terminal or command prompt and run:

* pip install matplotlib

**Step 2:** Open your Python IDE or text editor and create a new file (e.g., simple\_plot.py).

**Step 3:** Write the code. Below is a program to create a basic line plot, bar chart, and pie chart using Matplotlib.

**Step 4:** Save the file.

**Step 5:** Run the program:

* If using an IDE, click the **Run** button.
* If using the command line, navigate to the file's location and run

**Program Code:**

import matplotlib.pyplot as plt

# Line Plot

x = [1, 2, 3, 4, 5]

y = [2, 4, 6, 8, 10]

plt.plot(x, y, marker='o') # Plot the line with circular markers

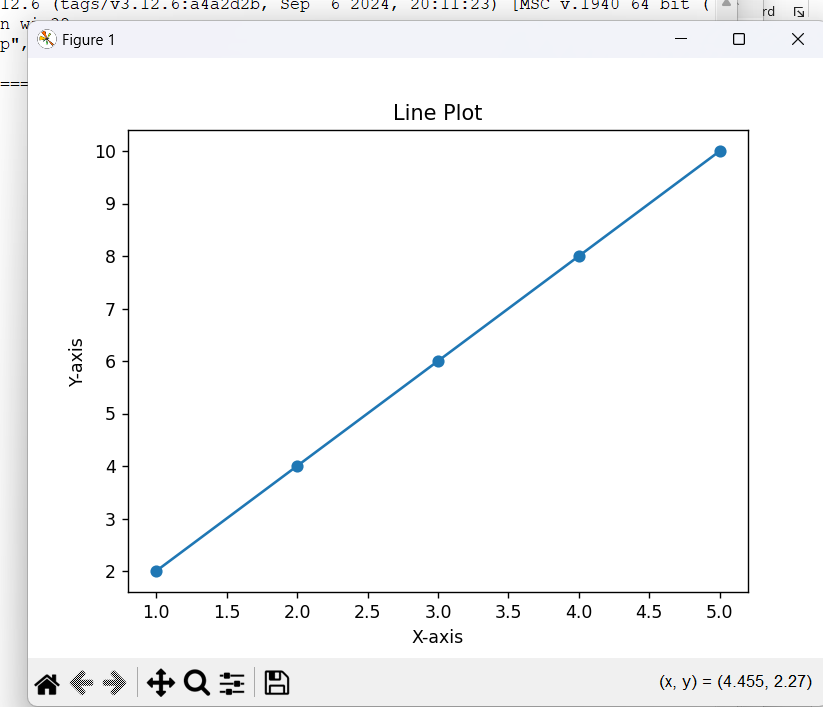
plt.title("Line Plot") # Add a title

plt.xlabel("X-axis") # Label for X-axis

plt.ylabel("Y-axis") # Label for Y-axis

plt.show() # Display the plot

**OuTput:**

****

**PRACTICAL-8**

**INSTALLATION AND SETUP OF DJANGO; CREATING AND RUNNING A SIMPLE DJANGO PROJECT**

**Step 1:** Install Django. Open a terminal or command prompt and run

* pip install Django

**Step 2:** Create a project folder. Navigate to the desired location and create a new folder for your Django project (e.g., myproject).

* mkdir myproject

cd myproject

**Step 3:** Start a new Django project. Run the following command to create a new Django project:

* django-admin startproject mysite

**Step 4:** Navigate into the project folder. Move into the newly created mysite directory.

* cd mysite

**Step 5:** Run the Django development server. Start the server to confirm the setup is working:

* python manage.py runserver

**Step 6:** Open your web browser. Navigate to http://127.0.0.1:8000/ to see the default Django welcome page.

Program Code: **Creating a Simple Django App**

**Step 1:** Create a Django app. Stop the server by pressing Ctrl+C and run:

* python manage.py startapp myapp

**Step 2:** Add the app to the project. Open mysite/settings.py and add 'myapp', under the INSTALLED\_APPS list:

INSTALLED\_APPS = [

...

'myapp',

]

**Step 3:** Create a view. Open myapp/views.py and add:

from django.http import HttpResponse

def hello\_world(request):

return HttpResponse("Hello, World!")

**Step 4:** Map the view to a URL. Create a urls.py file in the myapp folder and add:

from django.urls import path

from . import views

urlpatterns = [

path('', views.hello\_world, name='hello\_world'),

]

**Step 5:** Connect the app's URL to the project. Open mysite/urls.py and include the app's URLs

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

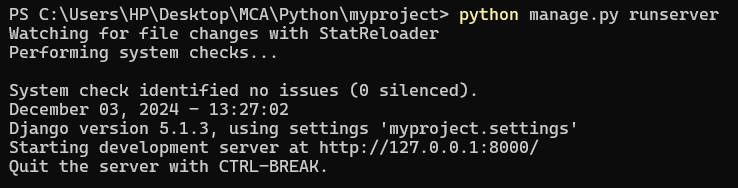
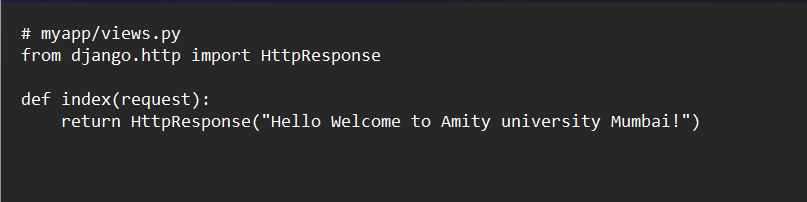
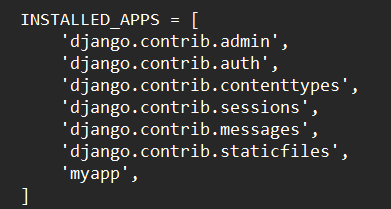
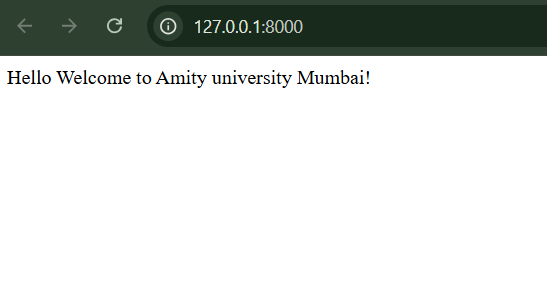
path('', include('myapp.urls')), # Include myapp's URLs

]

**Step 6:** Run the server. Start the server again:

python manage.py runserver

**Step 7:** Open your browser. Navigate to http://127.0.0.1:8000/ to see the **"** **Hello Welcome to Amity university Mumbai!”** message.



**PRACTICAL-9**

**INSTALLING KIVY AND RUNNING A SIMPLE "HELLO WORLD" PROGRAM**

**Step 1:** Install Kivy. Open a terminal or command prompt and run the following command to install Kivy:

* pip install kivy

**Step 2:** Create a Python script. Navigate to your desired project location and create a new Python script file (e.g., hello\_world.py).

* touch hello\_world.py

**Step 3:** Write a Simple Kivy Program. Open the hello\_world.py file in any text editor or IDE and add the code.

**Step 4:** Run the Kivy Program. In the terminal, navigate to the directory where the hello\_world.py file is located, and run.

* python hello\_world.py

**Step 5:** Verify the Application. A window should appear displaying the text "Hello, World!". This confirms that Kivy is installed and your program is working correctly

**Program Code:**

from kivy.app import App

from kivy.uix.label import Label

class HelloWorldApp(App):

def build(self):

return Label(text="Hello, World!")

if \_\_name\_\_ == "\_\_main\_\_":

HelloWorldApp().run()

