LAB 2

NumPy in Python:

NumPy is a powerful library in Python used for numerical computing. It stands for Numerical Python, and it provides support for **arrays.**

- Arrays in NumPy are like lists in Python, but they are faster and more efficient. • NumPy makes it easier to perform mathematical operations on large sets of data.

Installing NumPy:

To use NumPy, you first need to install it. You can install NumPy using **pip** if you don't have it already:

```
!pip install numpy
```

After successfully installing using pip then we will move towards **Importing**

NumPy: import numpy as np

Here, “np “ is just an alias (nickname for usage within the code) that makes it easier to refer to NumPy in the code. Instead of writing numpy.array(), you can write np.array().

1. Creating NumPy Arrays:

```
import numpy as np

# Creating a 1D array (like a list)
arr = np.array([1, 20, 19, 21, 4])

print(arr) # Output :[1, 20, 19, 21, 4]
```

2. Creating Multi-dimensional Arrays:

```
# Creating a 2D array (like a matrix)
2DArray = np.array([[1, 2, 3], [4, 5, 6]])
print(2DArray) # Output : [ [1 2 3]
 [4 5 6] ]
```

3. Slicing NumPy Arrays:

Slicing is an essential concept that allows you to extract a portion of the array. It works similarly to Python lists but has more flexibility.

- **Slicing a 1D Array:**

```
arr = np.array([10, 20, 30, 40, 50])
```

```
# Getting elements from index 1 to 3 (not including index 3)
arr = arr[1:4]
print(arr) # Output: [20 30 40]
```

- **Slicing a 2D Array (Matrix):**

```
arr_2d = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
# Get the first two rows, and the first two columns
arr = arr_2d[:2, :2]
print(arr) # Output : [[1 2]
 [4 5]]
```

- **Using Negative Indexing:**

```
arr = np.array([10, 20, 30, 40, 50])

# Get the last 3 elements
Arr1 = arr[-3:]
print(Arr1) # Output: [30 40 50]
```

Matplotlib in Python:

Matplotlib is a popular Python library for creating **visualizations** such as **plots**, **graphs**, and **charts**. It is commonly used for data visualization and helps in presenting complex data in an easy-to-understand, graphical format.

- **Matplotlib** helps you plot data, make graphs, and create visualizations.
- It is extremely useful for **data analysis** and **communication**.

Installing Matplotlib:

Similarly as previously discussed, we need to install matplotlib with pip command if you don’t have It already.

```
! pip install matplotlib
```

After Installing we will be importing **Importing Matplotlib**:

```
import matplotlib.pyplot as plt
```

Note “plt” is referred to as alias here.

1. **Creating a Simple Line Plot:**

```
import matplotlib.pyplot as plt

# Sample data
x = [1, 2, 3, 4, 5]
```

```
y = [1, 4, 9, 16, 25]

# Creating a line plot
plt.plot(x, y)
# Display the plot
plt.show()
```

2. Adding Titles and Labels:

You can add a title, labels to the axes, and a grid to make the graph more readable: `plt.plot(x, y)`

```
plt.title('Simple Line Plot')
plt.xlabel('X-axis Label')
plt.ylabel('Y-axis Label')
plt.grid(True)
plt.show()
```

3. Creating a Bar Chart:

```
categories = ['A', 'B', 'C', 'D']
values = [5, 7, 3, 8]

plt.bar(categories, values)
plt.title('Bar Chart Example')
plt.xlabel('Category')
plt.ylabel('Value')
plt.show()
```

LAB 2 TASKS

Question 1 (Estimated Time: 15 min, Marks: 15)
You are provided with the following two datasets:
# Measurements for each group

```
group_A = [12, 15, 14, 13, 16, 18, 19, 15, 14, 20, 17, 14, 15,40,45,50,62]
group_B = [12, 17, 15, 13, 19, 20, 21, 18, 17, 16, 15, 14, 16, 15]
```

> Create two separate Box plots using **Matplotlib** and **Subplots**, one for **Group A** and one for **Group B**.
> Ensure that each plot is properly labeled, including:

- Titles for both individual plots.
- Y-axis labels for measurement values.
- An overall figure title.

Subplot: https://www.w3schools.com/python/matplotlib_subplot.asp
Boxplot: https://www.geeksforgeeks.org/box-plot-in-python-using-matplotlib

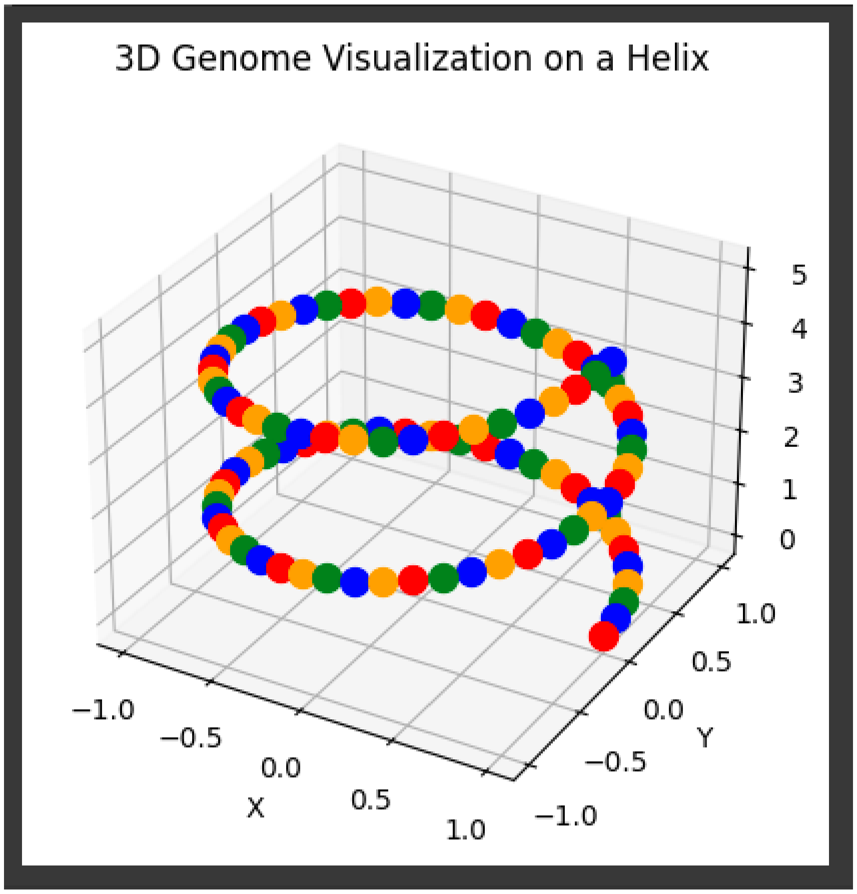
Question 2 (Estimated Time: 30 min, Marks: 30) **:**

Given a text file having helix-sequences, read that file into a genome_seqeunce, create a list and find its length using **list() and len()** built-in functions. Then use this code for computation of helix structure.

```
# We'll use the parametric equations for a helix:
#   x = cos(t), y = sin(t), z = t (or a scaled version of t)
# We want to span a range so that the helix makes a few turns.
t = np.linspace(0, 4 * np.pi, genome_length)  # 4*pi gives about
2 turns
x = np.cos(t)
y = np.sin(t)
z = np.linspace(0, 5, genome_length)  # z increases linearly to
spread out the helix vertically
# Combine the coordinates into a (genome_length x 3) array
coordinates = np.column_stack((x, y, z))
```

> assign colour of your choice to each molecule like A=red,etc
> create a 3D scatter plot as made in the guide.
Read file: https://www.geeksforgeeks.org/how-to-read-from-a-file-in-python/

> your resultant helix will look like this:



Question 3 (Estimated Time: 20 min, Marks: 20)
>Take any image from internet and convert that into numpy array
> plot that numpy array using matplot lib pyplot.
> rot and flip that image using np.rot90 and np.fliplr respectively and plot it
> also apply this grayscale filter to your original numpy image and plot it.

```
# Grayscale conversion formula: Y = 0.299*R + 0.587*G + 0.114*B
gray_img = np.dot (img_array[..., :3], [0.299, 0.587, 0.114])
```

Question 4 (Estimated Time: 30min, Marks: 20)

Downloading the Iris dataset:

Go to Google Collab and paste this code in the cell :

```
from sklearn.datasets import load_iris

import numpy as np

# Load the Iris dataset

iris = load_iris()

# Accessing the features (data) using NumPy array

X = np.array(iris.data) # (Features (sepal length, sepal width, petal length, petal

width) #Accessing the target labels (species)

Y = np.array(iris.target) # Target variable (species: 0 for setosa, 1 for versicolor, 2
for  virginica)
```

You are given the built-in **Iris dataset**, which contains information about different flower species and their features. Your task is to:

1. Use **NumPy** to:
   o Calculate the **mean, median, and standard deviation** for each feature. o Find the minimum and maximum values for each feature.
   o Extract only the **sepal length and sepal width** as a NumPy array. 2. Use **Matplotlib** to visualize the data:
   o Create a **scatter plot** of sepal length vs sepal width.
   o Plot a **histogram** showing the distribution of sepal length.
   o Create a **line plot** to visualize the relationship between petal length and petal width.