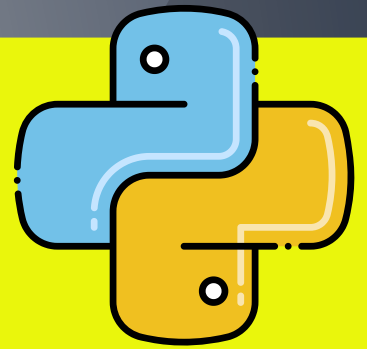
AKTU

B.Tech III-Sem

GATEWAY CLASSES

Python

Crash Course



Unit-5 in One Shot

Pragya Ma'am

Gateway classes

IMPORTANT QUESTION OF UNIT 5

1.	What is numpy , pandas, matplotlib
2.	Explain different types of plot(line, bar, scatter plot with example)
3.	Write a program transpose a matrix(see numpy program)
4.	Write a program to add /sub two matrices (see numpy program
5.	Write a program to perform matrix multiplication(see numpy program)
6	Write a program to reverse an array
7	Explain panda data series and explain its operation
8.	What is panda dataframe. Write a program to create dataframe from dictionary
9.	Write a program to sort the any series
10.	What is python? How python is interpreted? What are the toolsthat help to find bugs or perform static analysis? What are python decorators?

Gateway classes

IMPORTANT QUESTION OF UNIT 5

11.	Describe how to generate random number using numpy. writea python m program to create an array of 5 random number integer between 10 and 50
12.	What is trinket? Explain some of its widget with example

Gateway classes

SYLLABUS OF UNIT 5

Python packages: Simple programs using the built-in functions of packages matplotlib, numpy, pandas etc. GUI Programming: Tkinter introduction, Tkinter and PythonProgramming, Tk Widgets, Tkinter examples. Python programming with IDE.

Gateway classes

AKTU B.Tech Second Year

PYTHON PROGRAMMING

Unit 5

Python packages

LEC-1

Today's Target

- **NUMPY**
- **AKTU PYQs**

By
Pragya Rajvanshi

Gateway classes

Numpy

- NumPy is an open-source Python library that provides support for large, multi-dimensional arrays and matrices.
- It also have a collection of high-level mathematical functions to operate on arrays.

```
Array is of type:  <class 'numpy.ndarray'>
No. of dimensions:  2
Shape of array:  (2, 3)
Size of array:  6
Array stores elements of type:  int64
```

main.py

```python
import numpy as np
# Creating array object
arr = np.array( [[ 1, 2, 3],
                 [ 4, 2, 5]] )
# Printing type of arr object
print("Array is of type: ", type(arr))

# Printing array dimensions (axes)
print("No. of dimensions: ", arr.ndim)

# Printing shape of array
print("Shape of array: ", arr.shape)

# Printing size (total number of elements) of array
print("Size of array: ", arr.size)

# Printing type of elements in array
print("Array stores elements of type: ", arr.dtype)
```

Newway classes

NumPy Basic Operations

1. Operations on a single NumPy array

main.py

```
1  # Python program to demonstrate
2  # basic operations on single array
3  import numpy as np
4  a = np.array([1, 2, 5, 3])
5  # add 1 to every element
6  print ("Adding 1 to every element:", a+1)
7  # subtract 3 from each element
8  print ("Subtracting 3 from each element:", a-3)
9  # multiply each element by 10
10 print ("Multiplying each element by 10:", a*10)
11 # square each element
12 print ("Squaring each element:", a**2)
13 # modify existing array
14 a *= 2
15 print ("Doubled each element of original array:", a)
16 # transpose of array
17 a = np.array([[1, 2, 3], [3, 4, 5], [9, 6, 0]])
18
19 print ("\nOriginal array:\n", a)
20 print ("Transpose of array:\n", a.T)
21
```

Gateway classes

```
Adding 1 to every element: [2 3 6 4]
Subtracting 3 from each element: [-2 -1  2  0]
Multiplying each element by 10: [10 20 50 30]
Squaring each element: [ 1  4 25  9]
Doubled each element of original array: [ 2  4 10  6]

Original array:
 [[1 2 3]
 [3 4 5]
 [9 6 0]]
Transpose of array:
 [[1 3 9]
 [2 4 6]
 [3 5 0]]
```

NumPy – Unary Operators

Many unary operations are provided as a method of ndarray class. This includes sum, min, max, etc. These functions can also be applied row-wise or column-wise by setting an axis parameter

main.py

```python
# Python program to demonstrate
# unary operators in numpy
import numpy as np

arr = np.array([[1, 5, 6],
                [4, 7, 2],
                [3, 1, 9]])
# maximum element of array
print ("Largest element is:", arr.max())
print ("Row-wise maximum elements:",
                    arr.max(axis = 1))
# minimum element of array
print ("Column-wise minimum elements:",
                        arr.min(axis = 0))
# sum of array elements
print ("Sum of all array elements:",
                            arr.sum())
                            # cumulative sum along each row
print ("Cumulative sum along each row:\n",
                        arr.cumsum(axis = 1))

```

```
Largest element is: 9
Row-wise maximum elements: [6 7 9]
Column-wise minimum elements: [1 1 2]
Sum of all array elements: 38
Cumulative sum along each row:
 [[ 1  6 12]
 [ 4 11 13]
 [ 3  4 13]]
```

Gateway classes

NumPy – Binary Operators

These operations apply to the array elementwise and a new array is created. You can use all basic arithmetic operators like +, -, /, etc. In the case of *+=, -=,* = operators, the existing array is modified

main.py

```
1  # Python program to demonstrate
2  # binary operators in Numpy
3  import numpy as np
4
5  a = np.array([[1,2],
6                [2,4]])
7  b = np.array([[4, 3],
8                [2, 1]])
9
10 # add arrays
11 print ("Array sum:\n", a + b)
12
13 # multiply arrays (elementwise multiplication)
14 print ("Array multiplication:\n", a*b)
15
16 # matrix multiplication
17 print ("Matrix multiplication:\n", a.dot(b))
18
```

```
Array sum:
 [[5 5]
 [4 5]]
Array multiplication:
 [[4 6]
 [4 4]]
Matrix multiplication:
 [[ 8  5]
 [16 10]]
```

Gateway classes

Numpy sorting array

main.py

```
1  import numpy as np
2  a = np.array([[1, 4, 2],
3                [3, 4, 6],
4                [0, -1, 5]])
5  # sorted array
6  print ("Array elements in sorted order:\n",
7                  np.sort(a, axis = None))
8  # sort array row-wise
9  print ("Row-wise sorted array:\n",
10                 np.sort(a, axis = 1))
11 # specify sort algorithm
12 print ("Column wise sort by applying merge-sort:\n",
13                 np.sort(a, axis = 0, kind = 'mergesort'))
```

```
Array elements in sorted order:
 [-1  0  1  2  3  4  4  5  6]
Row-wise sorted array:
 [[ 1  2  4]
 [ 3  4  6]
 [-1  0  5]]
Column wise sort by applying merge-sort:
 [[ 0 -1  2]
 [ 1  4  5]
 [ 3  4  6]]
```

Gateway Classes

Write a NumPy program to convert a list of numeric values into a one-dimensional NumPy array.

main.py

```
1  import numpy as np
2  # Original list
3  original_list = [12.23, 13.32, 100, 36.32]
4
5  # Convert to one-dimensional NumPy array
6  numpy_array = np.array(original_list)
7
8  # Display results
9  print("Original List:", original_list)
10 print("One-dimensional NumPy array:", numpy_array)
```

```
Original List: [12.23, 13.32, 100, 36.32]
One-dimensional NumPy array: [ 12.23  13.32 100.    36.32]
```

Gateway classes

Write a NumPy program to create a 3x3 matrix with values ranging from 2 to 10.

main.py

```
1  import numpy as np
2  # Create a 3x3 matrix with values from 2 to 10
3  matrix = np.arange(2, 11).reshape(3, 3)
4  # Display the matrix
5  print("3x3 Matrix with values ranging from 2 to 10:")
6  print(matrix)
7
```

```
3x3 Matrix with values ranging from 2 to 10:
[[ 2  3  4]
 [ 5  6  7]
 [ 8  9 10]]
```

Gateway Classes

Write a NumPy program to reverse an array (the first element becomes the last)

main.py

```python
import numpy as np

# Create an example array
original_array = np.array([1, 2, 3, 4, 5])

# Reverse the array
reversed_array = original_array[::-1]

# Display the result
print("Original array:", original_array)
print("Reversed array:", reversed_array)

```

```
Original array: [1 2 3 4 5]
Reversed array: [5 4 3 2 1]
```

Gateway classes

main.py

```python
# Write a NumPy program to append values to the end of an array.
import numpy as np

# Original array
original_array = np.array([1, 2, 3])

# Values to append
values_to_append = [4, 5, 6]

# Append values to the array
result_array = np.append(original_array, values_to_append)

# Display the result
print("Original array:", original_array)
print("Appended array:", result_array)
```

```
Original array: [1 2 3]
Appended array: [1 2 3 4 5 6]
```

Gateway classes

Write a NumPy program to find common values between two arrays.

main.py

```
1  import numpy as np
2  # Define two arrays
3  array1 = np.array([1, 2, 3, 4, 5])
4  array2 = np.array([3, 4, 5, 6, 7])
5  # Find common values
6  common_values = np.intersect1d(array1, array2)
7  # Display the result
8  print("Array 1:", array1)
9  print("Array 2:", array2)
10 print("Common values:", common_values)
```

```
Array 1: [1 2 3 4 5]
Array 2: [3 4 5 6 7]
Common values: [3 4 5]
```

Gateway classes

Write a NumPy program to get the unique elements of an array.

main.py

```
1  import numpy as np
2  # Define an array
3  array = np.array([1, 2, 2, 3, 4, 4, 5, 5, 5])
4  # Get unique elements
5  unique_elements = np.unique(array)
6  # Display the result
7  print("Original array:", array)
8  print("Unique elements:", unique_elements)
9
```

```
Original array: [1 2 2 3 4 4 5 5 5]
Unique elements: [1 2 3 4 5]
```

Gateway classes

Write a NumPy program to find the set difference between two arrays. The set difference will return sorted, distinct values in array1 that are not in array2

main.py

```
1  import numpy as np
2
3  # Define two arrays
4  array1 = np.array([1, 2, 3, 4, 5])
5  array2 = np.array([3, 4, 5, 6, 7])
6
7  # Find the set difference
8  set_difference = np.setdiff1d(array1, array2)
9
10 # Display the result
11 print("Array 1:", array1)
12 print("Array 2:", array2)
13 print("Set difference", set_difference)
```

```
Array 1: [1 2 3 4 5]
Array 2: [3 4 5 6 7]
Set difference (elements in Array 1 not in Array 2): [1 2]
```

Gateway Classes

Union of two array

main.py

```
1  import numpy as np
2
3  # Define two arrays
4  array1 = np.array([0, 10, 20, 40, 60, 80])
5  array2 = np.array([10, 30, 40, 50, 70])
6
7  # Find the union of the two arrays
8  union_array = np.union1d(array1, array2)
9
10 # Display the result
11 print("Array 1:", array1)
12 print("Array 2:", array2)
13 print("Union of Array 1 and Array 2:", union_array)
14
```

```
Array 1: [ 0 10 20 40 60 80]
Array 2: [10 30 40 50 70]
Union of Array 1 and Array 2: [ 0 10 20 30 40 50 60 70 80]
```

Gateway Classes

AKTU B.Tech Second Year

PYTHON PROGRAMMING

Unit 5

LEC-2

Python packages

Today's Target

- **NUMPY program**
- **AKTU PYQs**

By
Pragya Rajvanshi

Gateway classes

Write a python program to create an array of 5 random integer between 10 and 50(inclusive 10 and 50)

main.py

```
1  import numpy as np
2  # Create an array of 5 random integers between 10 and 50
3  random_integers = np.random.randint(10, 51, size=5)
4  # Print the array
5  print(random_integers)
```

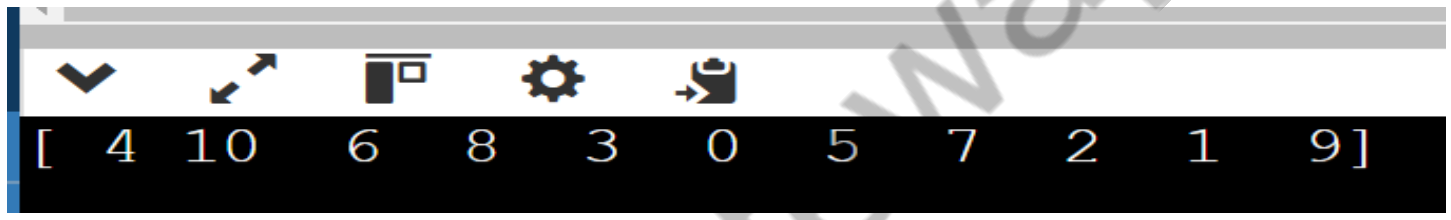


Gateway classes

Write a NumPy program to shuffle numbers between 0 and 10 (inclusive).

main.py

```
1  import numpy as np
2  # Create an array of numbers between 0 and 10 (inclusive)
3  numbers = np.arange(11)
4  # Shuffle the numbers
5  np.random.shuffle(numbers)
6  print(numbers)
```

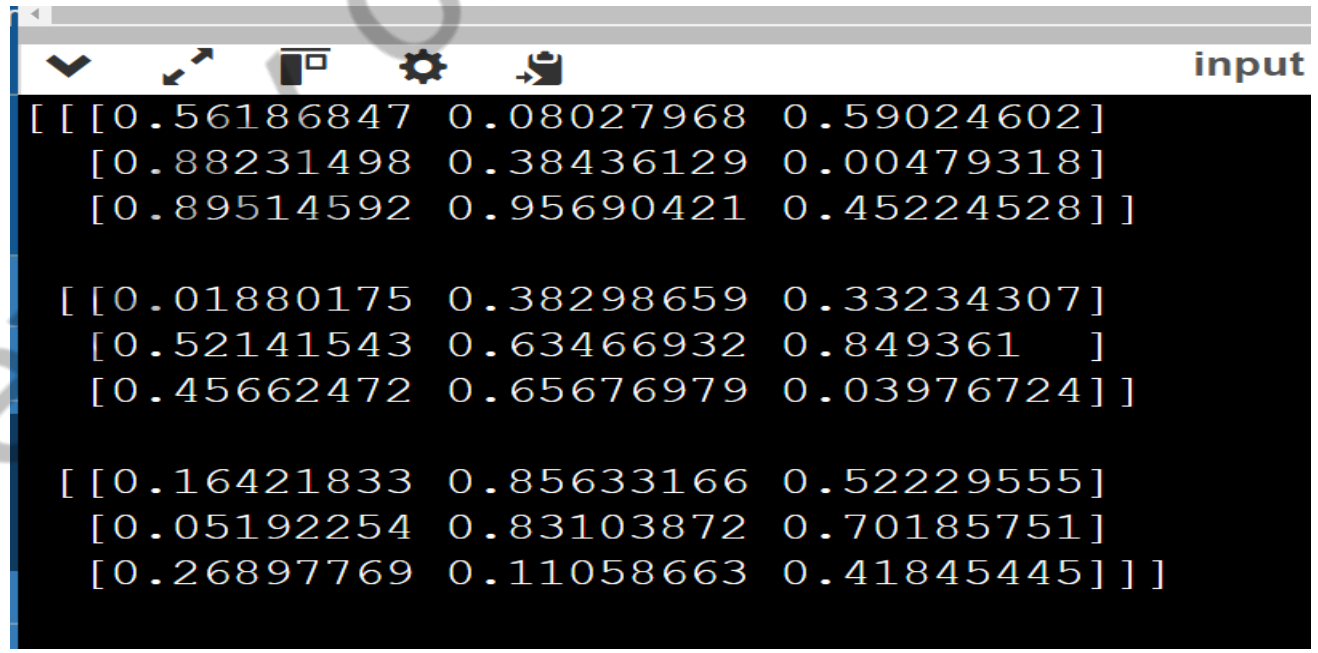


Gateway classes

Write a NumPy program to create a 3x3x3 array with random values

main.py

```
1  import numpy as np
2  # Create a 3x3x3 array with random values
3  random_array = np.random.random((3, 3, 3))
4  # Print the array
5  print(random_array)
```



Gate Classes

Panda

- Pandas is a powerful and open-source Python library. The Pandas library is used for data manipulation and analysis.
- Pandas is well-suited for working with tabular data, such as spreadsheet.
- Data Structures in Pandas Library
  - series and frames

Panda data series

Pandas Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.).

main.py

```
1  # import pandas as pd
2  import pandas as pd
3   # simple array
4  data = [1, 2, 3, 4]
5  ser = pd.Series(data)
6  print(ser)
```

```
0    1
1    2
2    3
3    4
dtype: int64
```

The axis labels are collectively called *index*. Pandas Series is nothing but a column in an excel sheet.

Gateway Classes

Creating series from list

main.py

```
1  import pandas as pd
2  # a simple list
3  list = ['h', 'e', 'l', 'l', 'o']
4  # create series form a list
5  ser = pd.Series(list)
6  print(ser)
```

```
0      h
1      e
2      l
3      l
4      o
dtype: object
```

Accessing Element from Series with Position

```
# import pandas and numpy
import pandas as pd
import numpy as np
# creating simple array
data = np.array(['l','e','e','m'])
ser = pd.Series(data)
#with the help of index operator
#retrieve the first element
print(ser[:2])
```

```
0    l
1    e
dtype: object
```

Gateway classes

Accessing Element Using Label (index)

main.py

```
1  # import pandas and numpy
2  import pandas as pd
3  import numpy as np
4   # creating simple array
5  data = np.array(['p','y','t','h','o','n', 'o','u','n','i','t','5','s'])
6  ser = pd.Series(data,index=[10,11,12,13,14,15,16,17,18,19,20,21,22])
7  # accessing a element using index element
8  print(ser[16])
```



Gateway classes

Binary operator

main.py

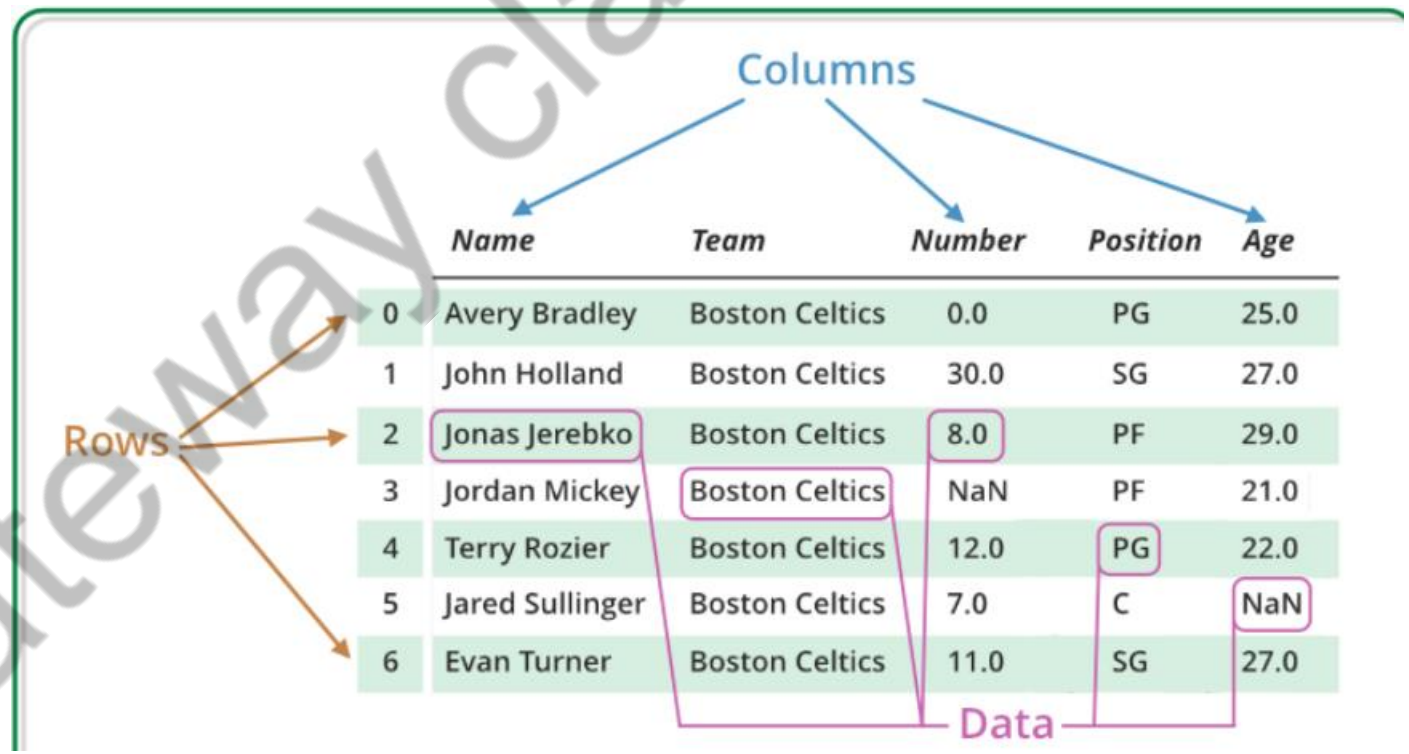
```
1  # importing pandas module
2  import pandas as pd
3  # creating a series
4  data = pd.Series([5, 2, 3,7], index=['a', 'b', 'c', 'd'])
5  # creating a series
6  data1 = pd.Series([1, 6, 4, 9], index=['a', 'b', 'd', 'e'])
7  print(data.add(data1, fill_value=0))
8  print(data.sub(data1, fill_value=0))
```

```
a      6.0
b      8.0
c      3.0
d     11.0
e      9.0
dtype: float64
a     4.0
b    -4.0
c     3.0
d     3.0
e    -9.0
dtype: float64
```

Gateway classes

Pandas DataFrame

Pandas DataFrame is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labeled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Pandas DataFrame consists of three principal components, the data, rows, and columns.



	Name	Team	Number	Position	Age
0	Avery Bradley	Boston Celtics	0.0	PG	25.0
1	John Holland	Boston Celtics	30.0	SG	27.0
2	Jonas Jerebko	Boston Celtics	8.0	PF	29.0
3	Jordan Mickey	Boston Celtics	NaN	PF	21.0
4	Terry Rozier	Boston Celtics	12.0	PG	22.0
5	Jared Sullinger	Boston Celtics	7.0	C	NaN
6	Evan Turner	Boston Celtics	11.0	SG	27.0

Gateway classes

Create a Pandas DataFrame from dictionary

```python
import pandas as pd
# intialise data of lists.
data = {'Name':['Tom', 'nick', 'krish', 'jack'],
        'Age':[20, 21, 19, 18]}
# Create DataFrame
df = pd.DataFrame(data)
# Print the output.
print(df)
```

```
    Name  Age
0    Tom   20
1   nick   21
2  krish   19
3   jack   18
```

Gateway classes

Create a Pandas DataFrame from dictionary

```
# Import pandas package
import pandas as pd

# Define a dictionary containing employee data
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}

# Convert the dictionary into DataFrame
df = pd.DataFrame(data)
print(df[['Name','Age']])
```

```
     Name  Age
0     Jai   27
1  Princi   24
2  Gaurav   22
3    Anuj   32
```

Gateway classes

Write a Pandas program to compare the elements of the two Pandas Series. Sample Series: [2, 4, 6, 8, 10], [1, 3, 5, 7, 10]

```
import pandas as pd
# Sample Series
series1 = pd.Series([2, 4, 6, 8, 10])
series2 = pd.Series([1, 3, 5, 7, 10])
# Compare the elements of the two Series
comparison_result = series1 == series2
print(comparison_result)
```

```
Comparison result:
0    False
1    False
2    False
3    False
4     True
dtype: bool
```

Gateway classes

Write a Pandas program to sort a given Series.

```
import pandas as pd
# Sample Series
series = pd.Series([10, 2, 5, 8, 1, 7])
# Sort the Series in ascending order
sorted_series = series.sort_values()
print("Sorted Series in ascending order:")
print(sorted_series)
```

```
Sorted Series in ascending order:
4     1
1     2
2     5
5     7
3     8
0    10
dtype: int64
```

G classes

AKTU B.Tech Second Year

PYTHON PROGRAMMING

Unit 5

Python file operations

LEC-3

Today's Target

- **PYTHON WITH IDE**
- **AKTU PYQs**

By
Pragya Rajvanshi

Gateway classes

Syllabus

Python packages: Simple programs using the built-in functions of packages matplotlib, numpy, pandas etc. GUI Programming: Tkinter introduction, Tkinter and PythonProgramming, Tk Widgets, Tkinter examples. Python programming with IDE.

Gateway classes

Python

- Python is a high-level, interpreted programming language known for its simplicity and readability
- You can use it to create websites, analyze data, automate tasks, or build apps.
- It's beginner-friendly and lets you write code that's easy to read and understand.

Features:

Easy to Learn and Use: Simple syntax like plain English.

Cross-Platform: Works on Windows, macOS, Linux, etc.

Large Library: Ready-to-use tools for tasks like math, web, and data.

Dynamic Typing: No need to define variable types explicitly.

Community Support: Lots of resources and help available online.

Gateway classes

How python is interpreted?

Python is interpreted because its code is executed line by line by the Python interpreter.

Here's how it works:

1.Source Code:

- You write Python code in a file (e.g., example.py).
- This is the human-readable version of your program.

2.Python Interpreter:

- When you run the code (e.g., using python example.py), the Python interpreter starts working.
- The interpreter reads the code line by line.

3.Bytecode Conversion:

- The interpreter translates your source code into an intermediate form called bytecode.
- Bytecode is a simplified version of your code that is easier for the computer to understand.
- This bytecode is stored temporarily in memory or as a .pyc file (if caching is enabled).

.

Gateway classes

4.Python Virtual Machine (PVM):

- The bytecode is executed by the Python Virtual Machine.
- The PVM takes each instruction from the bytecode and executes it on your computer.

5.Output:

- The interpreter runs the code step by step, producing results as it goes.
- If there's an error in your code, Python stops and shows an error message.

Static analysis

It is the process of analyzing code without executing it. It checks for bugs, errors, security vulnerabilities, and code quality issues by examining the source code or bytecode.

Key Points:

- **How it works:** Tools analyze the code structure, syntax, and logic to detect potential problems.
- **Purpose:** To catch errors early in the development process before the code is run or tested.
- **Examples:** Finding unused variables, type mismatches, or missing imports.

Gateway classes

Tools that help to find bugs or perform static analysis

- PyLint is a popular Python tool for analyzing code to ensure it adheres to coding standards, finds errors, and improves code quality.
- It performs static code analysis and provides suggestions for improvement.

Features of PyLint:

1.Code Quality Checks:

- Detects issues like unused variables, undefined variables, and unreachable code.
- Ensures your code follows the Python style guide (PEP 8).

2.Coding Standards:

- Highlights deviations from best practices.
- Enforces consistent naming conventions for variables, classes, and methods.

3. Error Detection:

- Finds common errors like missing imports, syntax issues, or potential bugs.

Gateway classes

Code Refactoring:

- Provides suggestions to refactor and simplify code to make it cleaner and more readable.

Custom Rules:

- Allows users to define custom rules for analysis or disable specific checks as needed.

- **PyChecker**
- It is a static analysis tool for Python that helps find bugs, errors, and inconsistencies in code.
- It was one of the earlier tools for Python linting, though it is less commonly used today compared to tools like PyLint.

Features:

1.Bug Detection:

- Identifies syntax errors and logical mistakes in code.
- Flags unused variables, modules, or imports.

2.Error Analysis:

- Detects type mismatches and missing function arguments.
- Highlights potential runtime errors.

Gateway classes

3. Code Warnings:

- Issues warnings for deprecated Python features or bad practices.

4.Cross-Version Compatibility:

- Checks for compatibility issues across different Python versions.

Python decorators

- A decorator is a special type of function in Python that allows you to modify or enhance the behavior of other functions or methods without changing their code.
- Decorators are often used to add functionality like logging, timing, or access control to functions.

Gateway classes

main.py

```python
def decorator_function(original_function):
    def wrapper_function():
        print("Before function execution")
        original_function()  # Call the original function
        print("After function execution")
    return wrapper_function

@decorator_function  # Apply the decorator
def greet():
    print("Hello, World!")

greet()
```

```
Before function execution
Hello, World!
After function execution
```

Gateway classes

What is IDE?

- An IDE (Integrated Development Environment) is a software application that provides tools to help programmers write, debug, and manage code efficiently.
- It integrates several features into one interface, making the development process easier and faster.

1.Code Editor: A text editor with features like syntax highlighting, autocomplete, and code formatting.

2.Debugger:Helps identify and fix errors in your code by running it step by step.

3.Compiler/Interpreter: Converts your source code into executable code (or interprets it for languages like Python).

4.Project Management: Allows you to organize files and folders within a project.

5.Integrated Tools: Includes tools for version control (e.g., Git), testing, and deployment.

.

Gateway classes

SOME OF IDE

1. PyCharm

- It is a popular Integrated Development Environment (IDE) specifically designed for Python development.
- It is developed by JetBrains and provides many features to help developers write, debug, and manage Python projects efficiently.

Key features :

1.Code Editor:

- Smart code completion and suggestions.
- Syntax highlighting and formatting.

2.Debugger:

- Powerful tools to identify and fix errors in Python code.

3.Code Analysis:

- Highlights coding issues and suggests improvements.

4.Project Management:

- Helps organize files and dependencies in large projects.

5.Support for Web Development:

- Includes frameworks like Django and Flask.

Gateway classes

6.Testing Tools:

- Includes support for running unit tests directly from the IDE.

Why Use PyCharm?

- Simplifies Python development with smart tools.
- Boosts productivity with features like code analysis and debugging.
- Ideal for both beginners and experienced developers.

2. VS Code

- VS Code (Visual Studio Code) is a lightweight, powerful, and free code editor developed by Microsoft.
- It is widely used for writing, debugging, and managing code in multiple programming languages, including Python, JavaScript, C++ etc.

Why Use VS Code?

- **Free:** Fully free and open-source.
- **Lightweight:** Ideal for both small scripts and large projects.
- **Customizable**: Extensions let you tailor it to your needs.
- **Community:** Huge user base with plenty of tutorials and resources.

Gateway classes

Key Features of VS Code:

1.Lightweight and Fast:

Designed to be minimalistic and run efficiently on most systems.

2.Code Editor:

Syntax highlighting, autocompletion, and IntelliSense (smart code suggestions)

3.Debugger:

Debug code with breakpoints, variable inspection, and call stack analysis.

4.Cross-Platform:

Available for Windows, macOS, and Linux.

5.Customizable:

Personalize the editor with themes, fonts, and keyboard shortcuts.

6.Multi-Language Support:

Supports many languages (Python, Java, JavaScript, C#, etc.) with extensions.

Gateway classes

3.Eclipse

- It is a powerful, open-source Integrated Development Environment (IDE) primarily used for developing Java applications.
- It is also versatile and supports other programming languages.

Key Features of Eclipse:

1.Java-Focused:

- Built specifically for Java development with tools like auto-completion, debugging, and refactoring.

2.Project Management:

- Organize large projects easily with a structured interface.

3.Integrated Debugger:

- Set breakpoints, inspect variables, and step through your code for debugging.

4.Cross-Platform:

- Available for Windows, macOS, and Linux.

5.Version Control:

- Built-in support for Git and other version control systems.

Gateway classes

4.Spyder

- Spyder (Scientific Python Development Environment) is an open-source Integrated Development Environment (IDE) specifically designed for data science and scientific computing in Python.
- It is widely used for tasks like data analysis, machine learning, and visualization.

Key features of spyder

1.Data Visualization:

- Supports inline plotting using libraries like Matplotlib and Seaborn.

2.Integrated Debugger:

- Step through your code to debug and fix errors.

3.Project Management:

- Organize multiple scripts and files in a structured manner.

4.Code Analysis:

- Includes tools like Pylint to detect errors and enforce coding standards.

5.Support for Scientific Libraries:

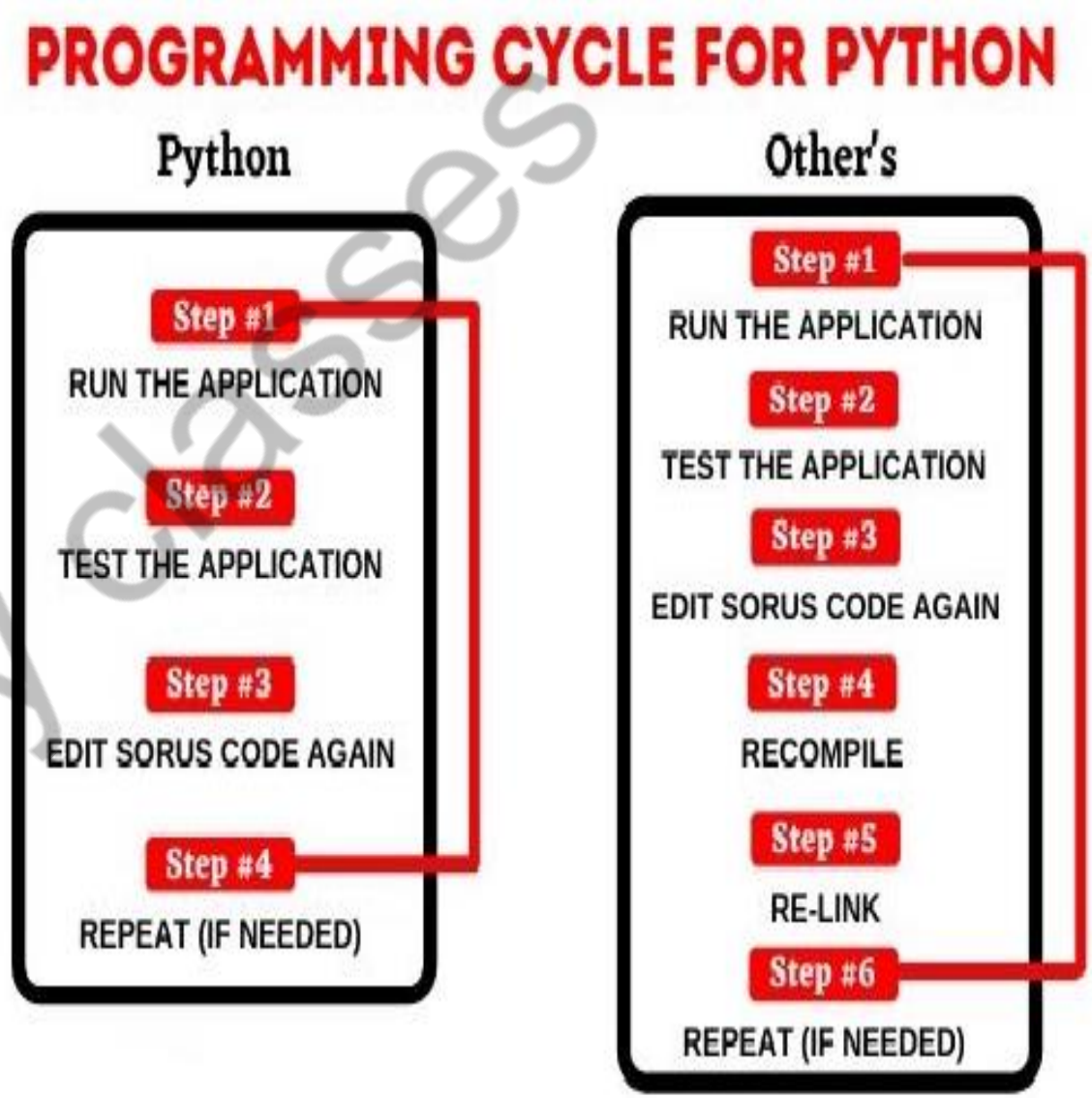
- Seamless integration with NumPy, Pandas, SciPy, and Matplotlib.

Gateway classes

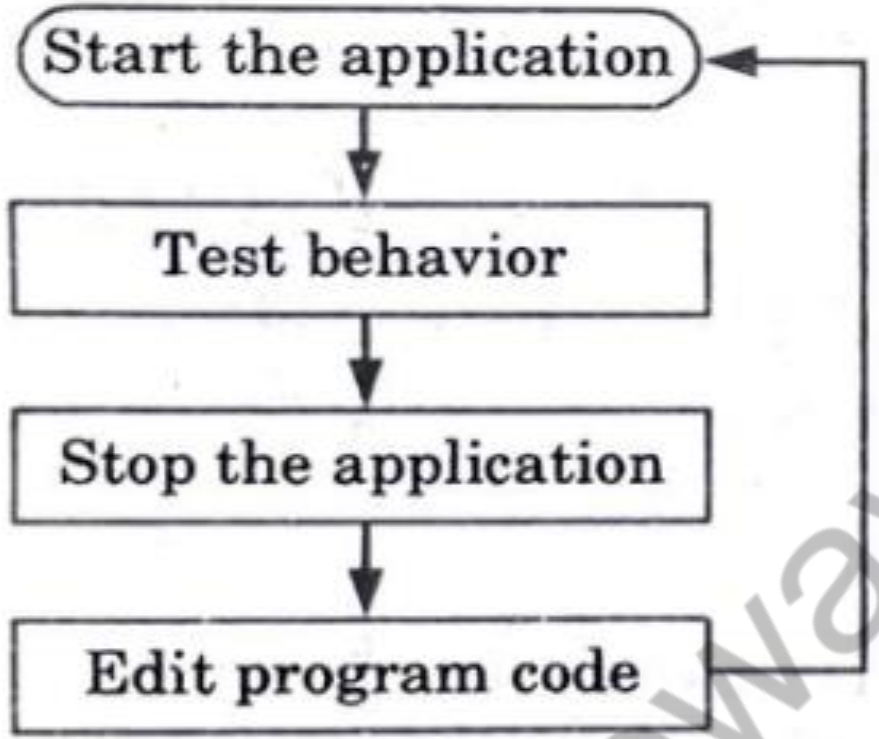
Programming Cycle for python

- The development cycle of Python is considerably shorter than that of traditional tools.
- There is no compilation or linking steps in Python.
- Python programs simply import modules at runtime and use the objects they contain. Because of this, Python programs run immediately after changes are made.

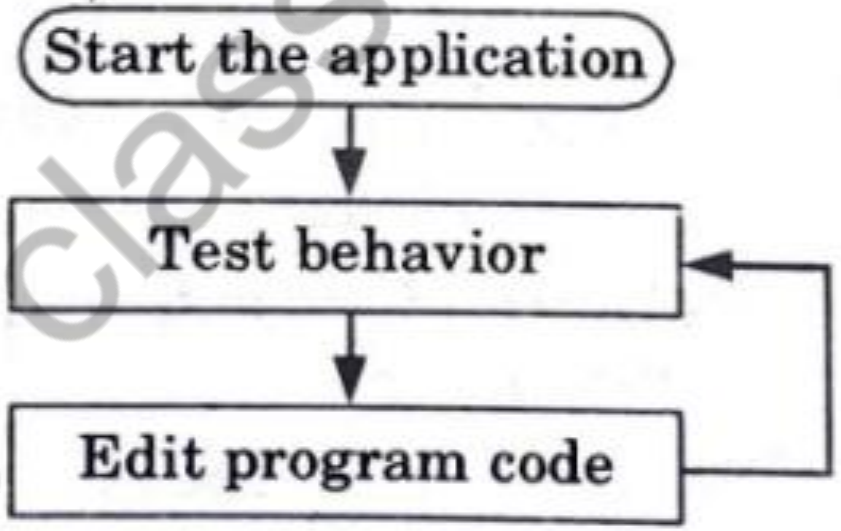
.



Gateway classes



a) Python's programming cycle



(b) Python's programming cycle with module reloading

Gateway classes

What is PEP 8?

- **PEP 8 (Python Enhancement Proposal 8) is the official style guide for Python code.**
- **It provides guidelines and best practices to help developers write clean, readable, and consistent Python code.**

Why Follow PEP 8?

- **Readability**: Makes code easier to read and understand.
- **Consistency**: Helps teams maintain a uniform coding style.
- **Best Practices**: Encourages efficient code.

Key Guidelines of PEP 8:

1.Indentation:

Use **4 spaces per indentation level**.

2. Line length

PEP 8 recommends limiting each line of code to a maximum of **79 characters** and comments or docstrings to **72 characters.**

Gateway classes

Spaces

- Use spaces around operators and after commas.
- Don't add spaces inside parentheses or brackets.

Imports

- Put imports at the top of the file.
- Import one library per line.

Comments

- Write meaningful comments to explain *why* something is done.
- Use # and keep comments short and clear python.

Functions

- Keep your functions small and focused on one
- Name them clearly to show what they do.

Gateway classes

AKTU QUESTIONS		
Q.1	What is python? How python is interpreted? What are the tools that help to find bugs or perform static analysis? What are python decorators?	AKTU 2019-20 AKTU 2020-21 AKTU 2022-23
Q.2	Describe how to generate random number using numpy. write a python m program to create an array of 5 random number integer between 10 and 50	AKTU 2023-24
Q.3	Explain the dataFrame in pandas. Write a python program to create an dataFrame from dictionary and print it	AKTU 2023-24

Gateway classes

AKTU B.Tech Second Year

PYTHON PROGRAMMING

Unit 5

Python file operations

LEC-4

Today's Target

- **matplotlib**
- **AKTU PYQs**

By
Pragya Rajvanshi

Gateway classes

Matplotlib

- Visualizing data helps us understand and share information more effectively.
- In Python , **Matplotlib** is one of the best tools for creating visualizations.

Different types of plolt

1.Line plot

2.Bar plot

3.Pie chart

4.Scatter plot

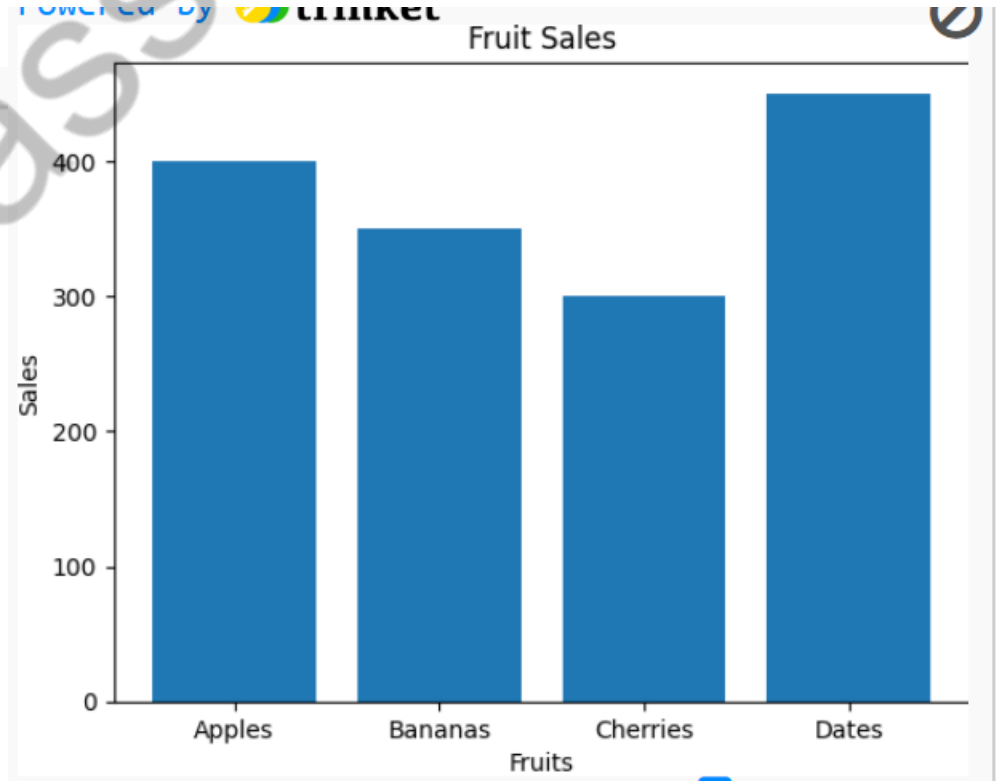
5.Histogram

Gateway classes

Bar plot

A **bar plot** or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent.

```
import matplotlib.pyplot as plt
fruits = ['Apples', 'Bananas', 'Cherries', 'Dates']
sales = [400, 350, 300, 450]
plt.bar(fruits, sales)
plt.title('Fruit Sales')
plt.xlabel('Fruits')
plt.ylabel('Sales')
plt.show()
```



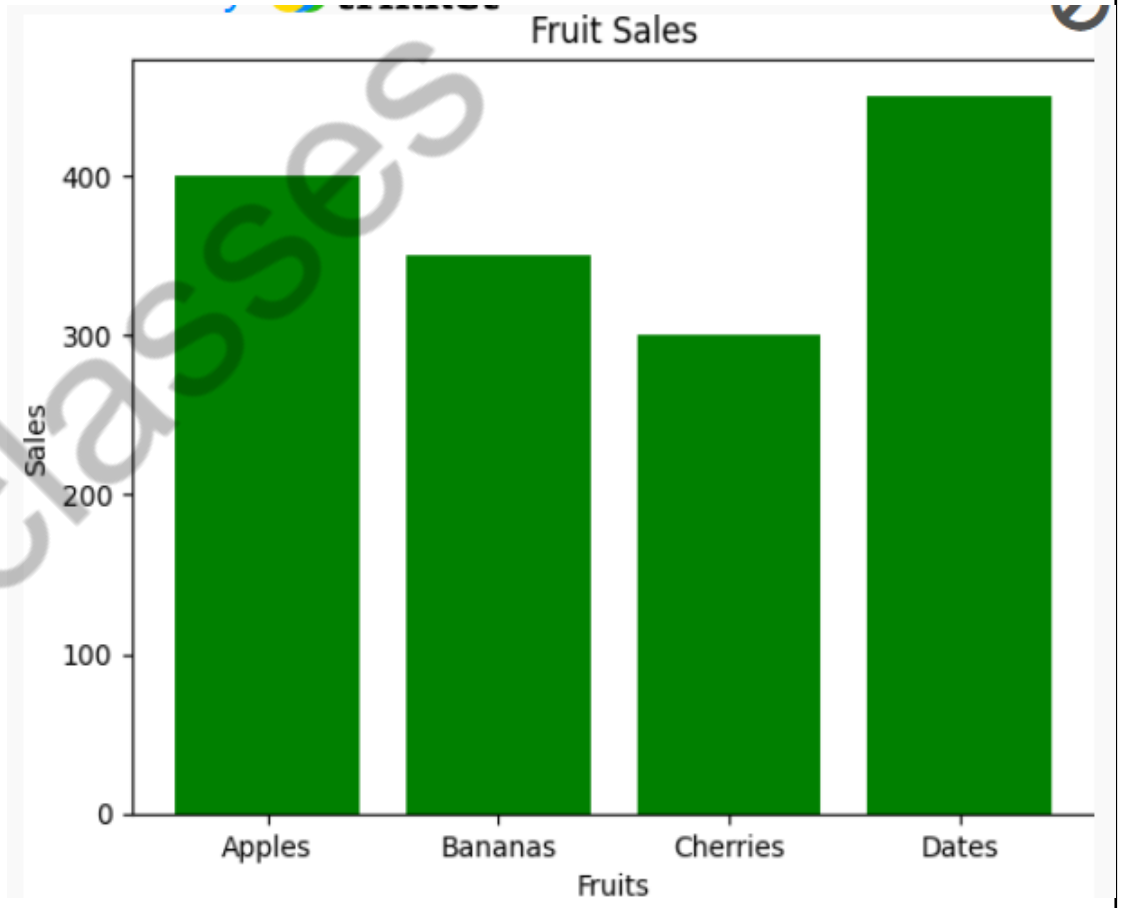
Gateway classes

Bar plot

Customizing Bar Colors

```
import matplotlib.pyplot as plt
fruits = ['Apples', 'Bananas', 'Cherries', 'Dates']
sales = [400, 350, 300, 450]

plt.bar(fruits, sales, color='green')
plt.title('Fruit Sales')
plt.xlabel('Fruits')
plt.ylabel('Sales')
plt.show()
```

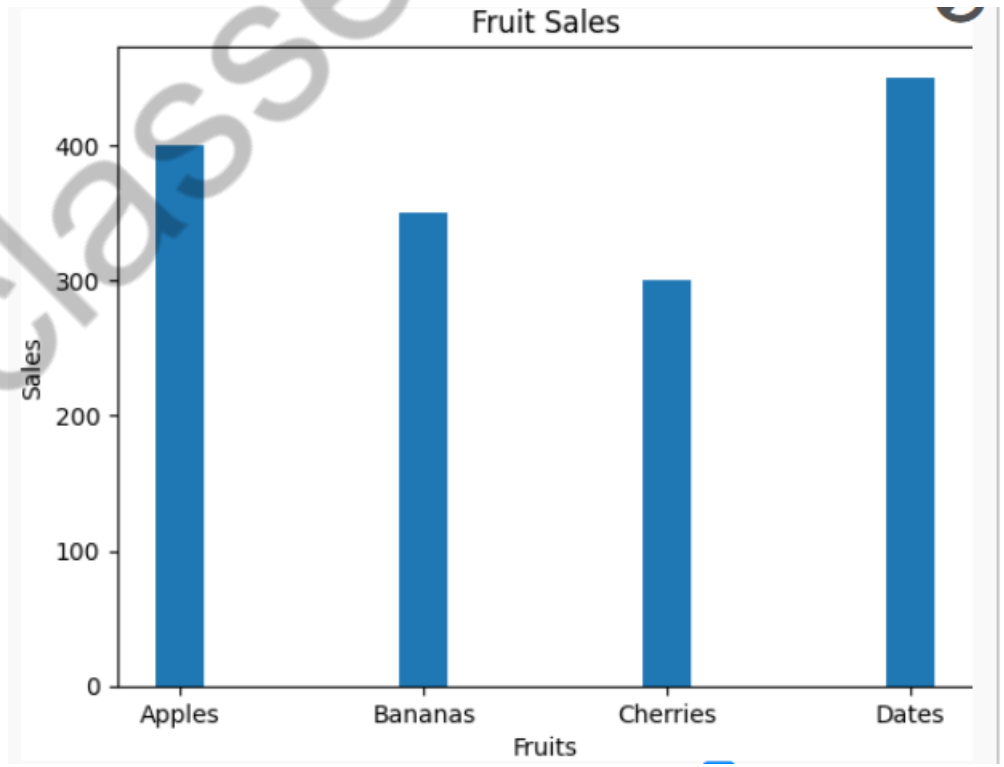


Gateway classes

Bar plot

Adjusting Bar Width

```
import matplotlib.pyplot as plt
fruits = ['Apples', 'Bananas', 'Cherries', 'Dates']
sales = [400, 350, 300, 450]
plt.bar(fruits, sales, width=0.2)
plt.title('Fruit Sales')
plt.xlabel('Fruits')
plt.ylabel('Sales')
plt.show()
```

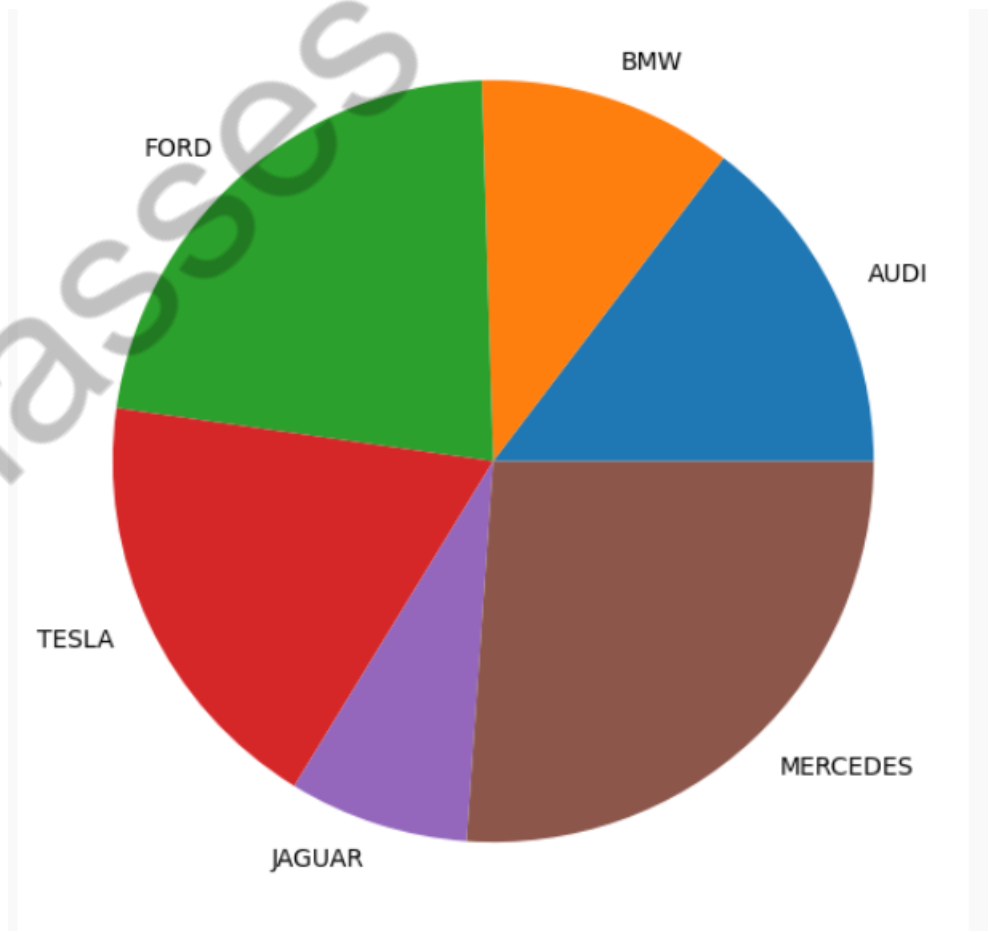


Gateway Classes

piechart

```
# Import libraries
from matplotlib import pyplot as plt
# Creating dataset
cars = ['AUDI', 'BMW', 'FORD',
        'TESLA', 'JAGUAR', 'MERCEDES']

data = [23, 17, 35, 29, 12, 41]
# Creating plot
fig = plt.figure(figsize=(10, 7))
plt.pie(data, labels=cars)
# show plot
plt.show()
```



Gateway classes

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4]

e  =(0.1, 0, 0, 0)

plt.pie(x, explode = e)

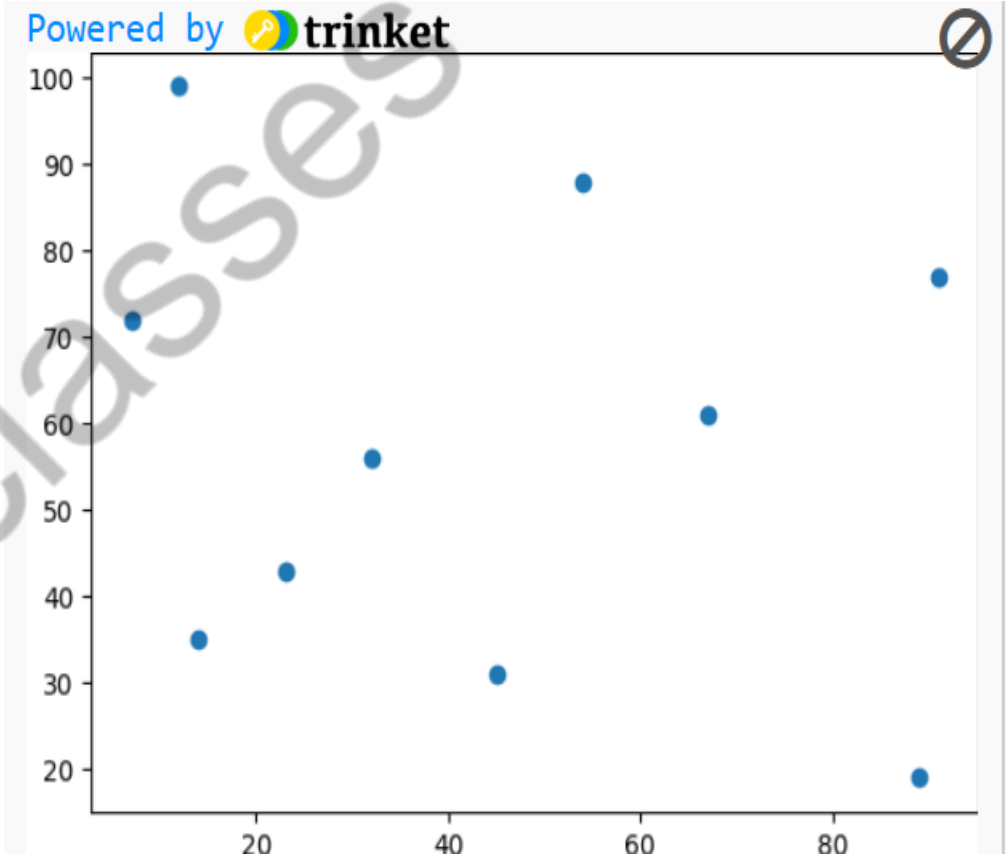
plt.title("Pie chart")
plt.show()
```



Gateway classes

Matplotlib Scatter

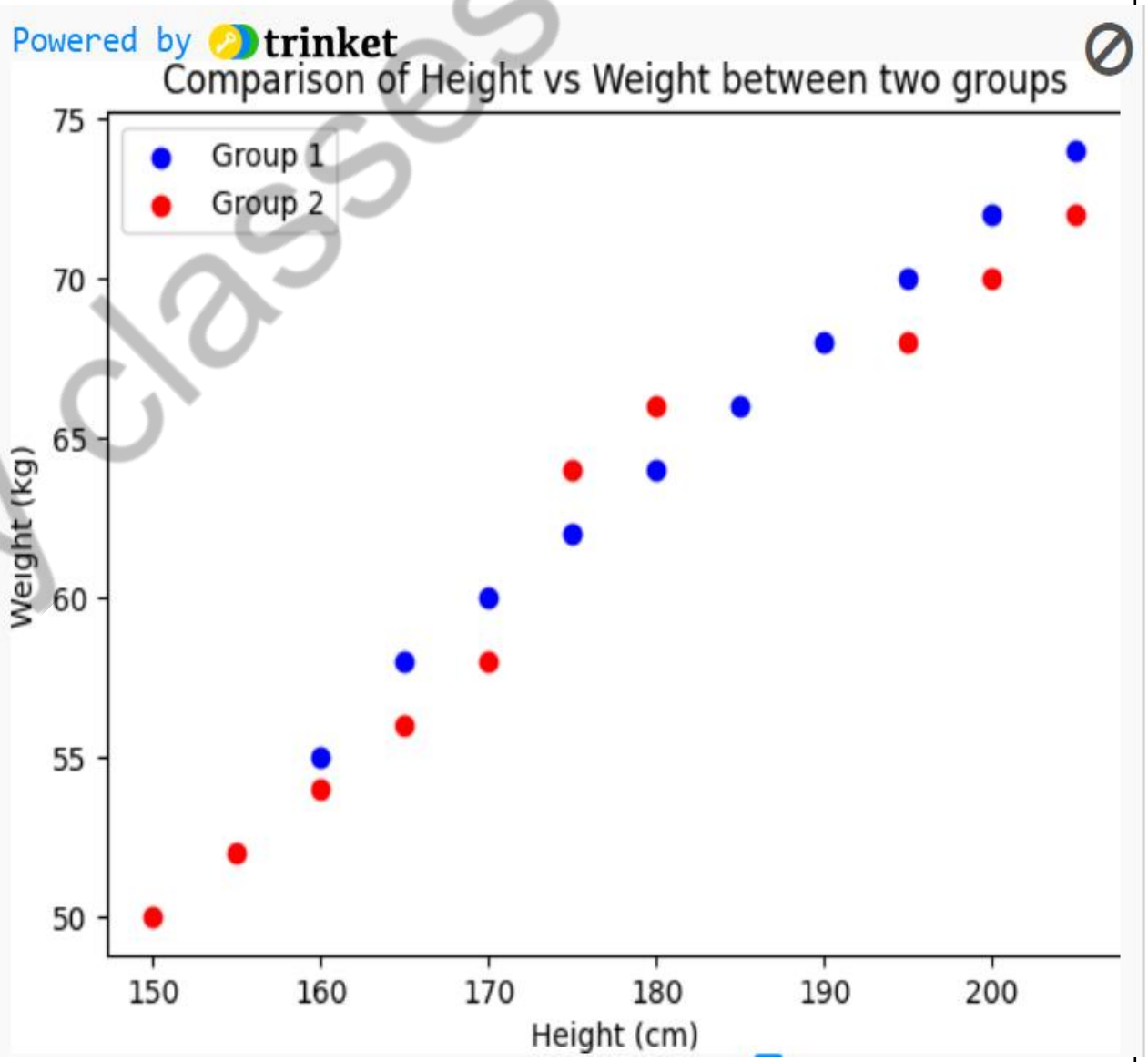
```
import matplotlib.pyplot as plt
import numpy as np
x = np.array([12, 45, 7, 32, 89, 54, 23, 67, 14, 91])
y = np.array([99, 31, 72, 56, 19, 88, 43, 61, 35, 77])
plt.scatter(x, y)
plt.show()
```



Gateway classes

Scatter Plot with Multiple Datasets

```
import matplotlib.pyplot as plt
import numpy as np
x1 = np.array([160, 165, 170, 175, 180, 185, 190, 195, 200, 205])
y1 = np.array([55, 58, 60, 62, 64, 66, 68, 70, 72, 74])
x2 = np.array([150, 155, 160, 165, 170, 175, 180, 195, 200, 205])
y2 = np.array([50, 52, 54, 56, 58, 64, 66, 68, 70, 72])
plt.scatter(x1, y1, color='blue', label='Group 1')
plt.scatter(x2, y2, color='red', label='Group 2')
plt.xlabel('Height (cm)')
plt.ylabel('Weight (kg)')
plt.title('Comparison of Height vs Weight between two groups')
plt.legend()
plt.show()
```



Gateway classes

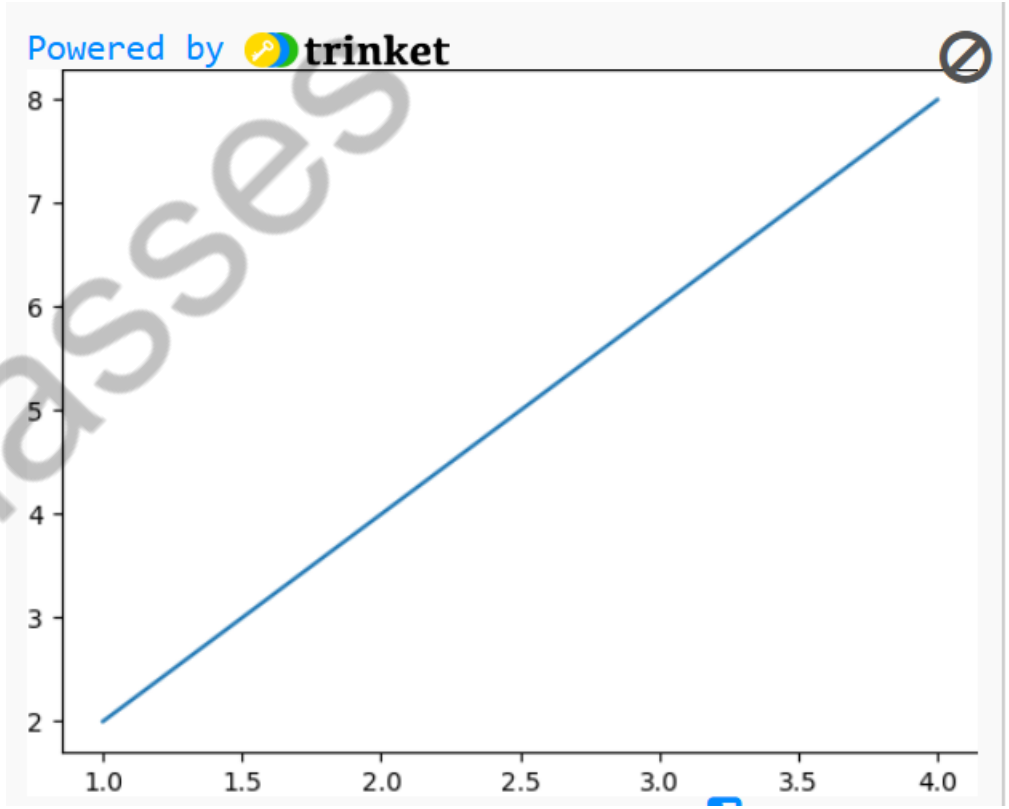
Different Types of Plots

1. Line chart

```
# importing the required libraries
import matplotlib.pyplot as plt
import numpy as np

# define data values
x = np.array([1, 2, 3, 4])  # X-axis points
y = x*2  # Y-axis points

plt.plot(x, y)  # Plot the chart
plt.show()  # display
```



Gateway classes

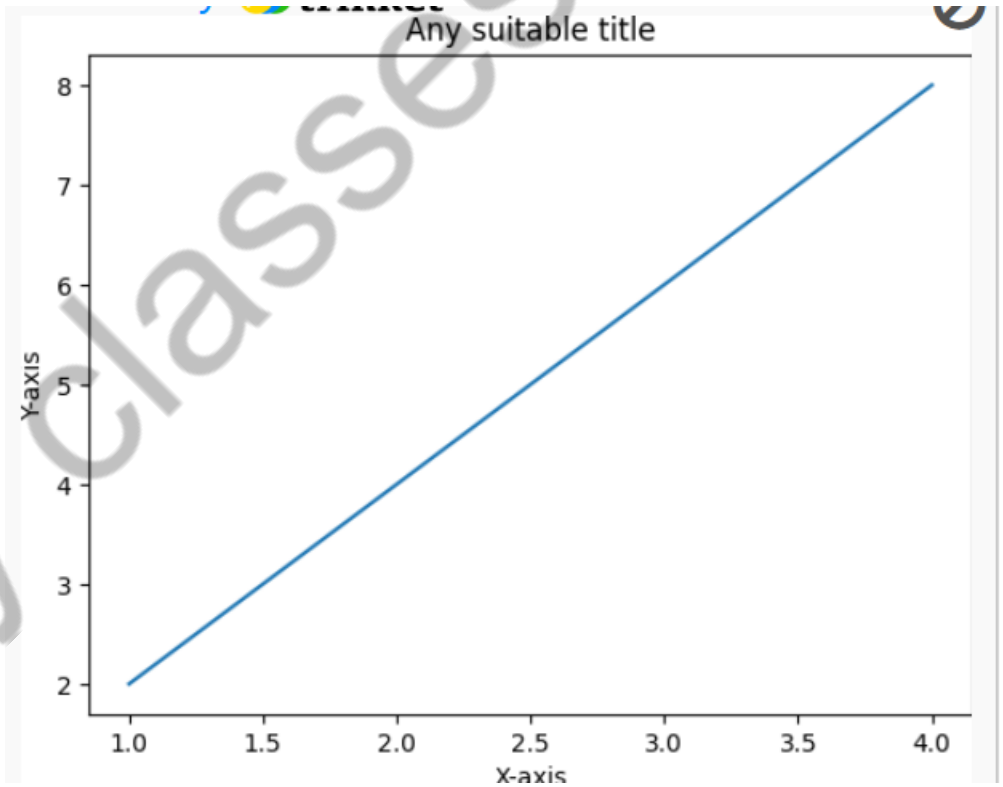
Different Types of Plots

1. Line chart

```
import matplotlib.pyplot as plt
import numpy as np


# Define X and Y variable data
x = np.array([1, 2, 3, 4])
y = x*2

plt.plot(x, y)
plt.xlabel("X-axis")  # add X-axis label
plt.ylabel("Y-axis")  # add Y-axis label
plt.title("Any suitable title")  # add title
plt.show()
```



Gateway Classes

Different Types of Plots
Multiple Plots on the Same Axis

```
import matplotlib.pyplot as plt
import numpy as np

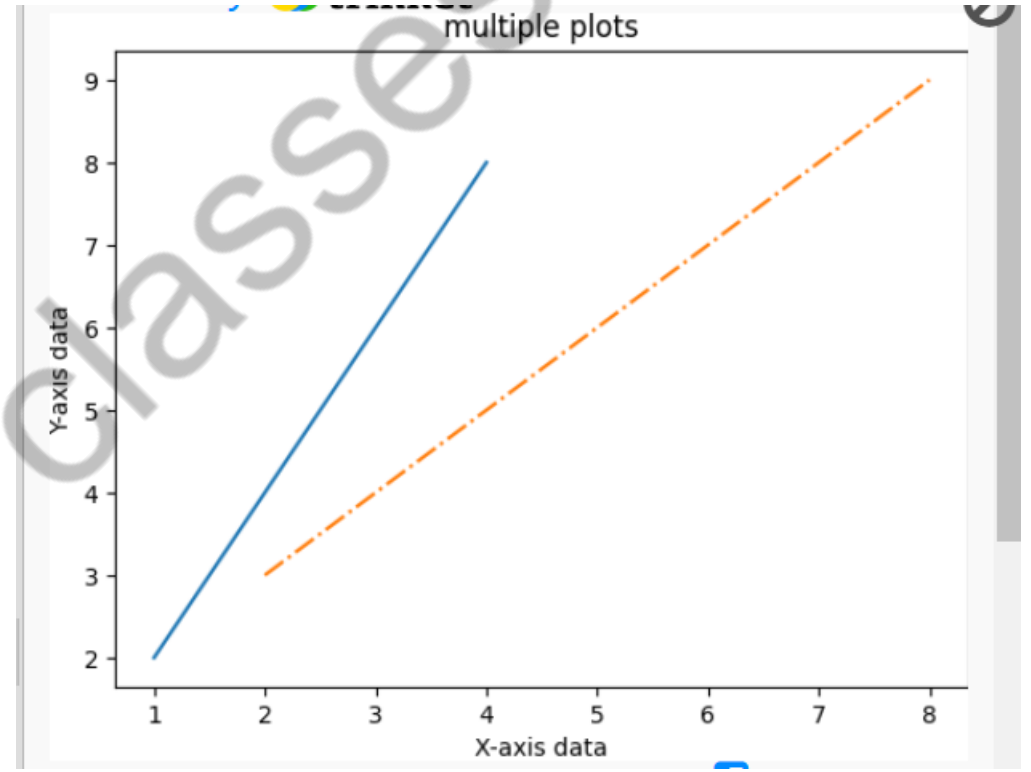
x = np.array([1, 2, 3, 4])
y = x*2

# first plot with X and Y data
plt.plot(x, y)

x1 = [2, 4, 6, 8]
y1 = [3, 5, 7, 9]

# second plot with x1 and y1 data
plt.plot(x1, y1, '-.')

plt.xlabel("X-axis data")
plt.ylabel("Y-axis data")
plt.title('multiple plots')
plt.show()
```

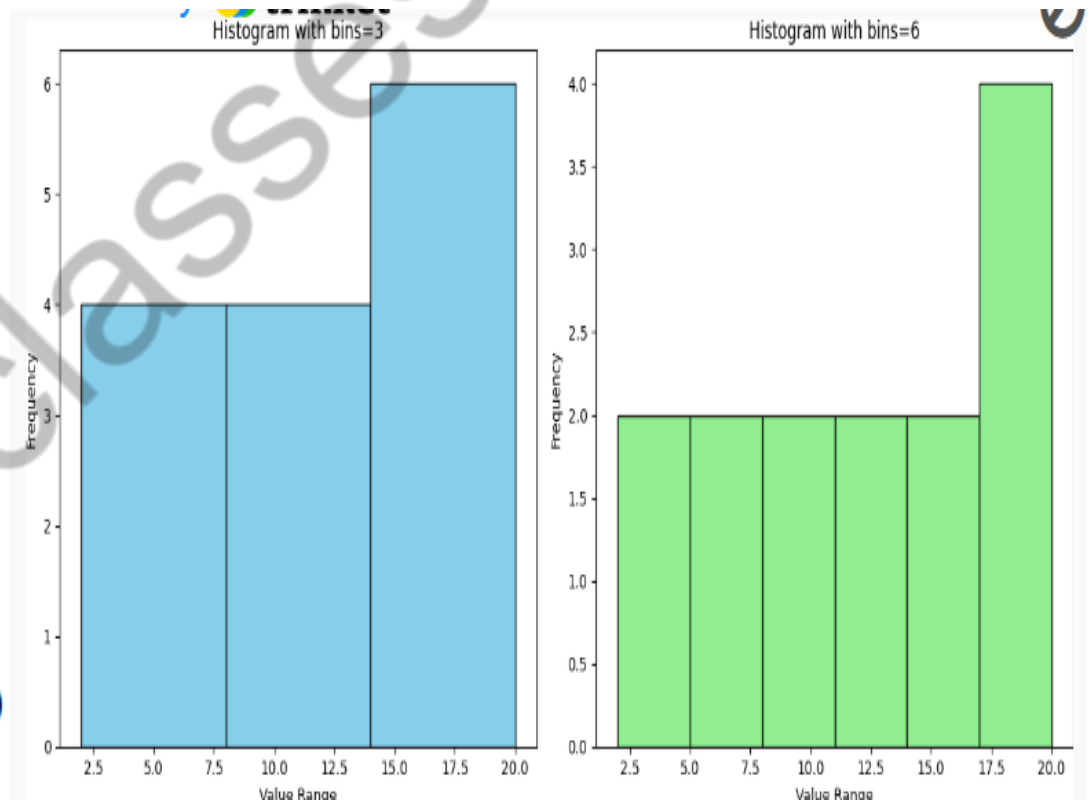


Gateway Classes

Histogram

```python
import matplotlib.pyplot as plt
# Example data
data = [2, 3, 5, 7, 8, 9, 11, 13, 14, 15, 17, 18, 19, 20]
# Plot histogram with different bin sizes
plt.figure(figsize=(12, 6))
# Using bins=3
plt.subplot(1, 2, 1)  # First subplot
plt.hist(data, bins=3, color='skyblue', edgecolor='black')
plt.title('Histogram with bins=3')
plt.xlabel('Value Range')
plt.ylabel('Frequency')
# Using bins=6
plt.subplot(1, 2, 2)  # Second subplot
plt.hist(data, bins=6, color='lightgreen', edgecolor='black')
plt.title('Histogram with bins=6')
plt.xlabel('Value Range')
plt.ylabel('Frequency')

# Show the plots
plt.tight_layout()
plt.show()
```



Gateway classes

AKTU B.Tech Second Year

PYTHON PROGRAMMING

Unit 5

Python file operations

LEC-5

Today's Target

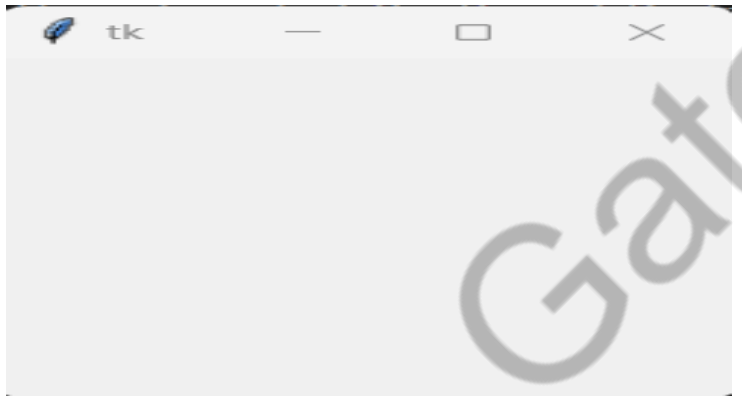
- **Tkinter**
- **module**

By
Pragya Rajvanshi

Gateway classes

Tkinter

- Python Tkinter is a standard GUI (Graphical User Interface) library for Python which provides a fast and easy way to create desktop applications.
- Tkinter provides a variety of widgets like buttons, labels, text boxes, menus and more that can be used to create interactive user interfaces.



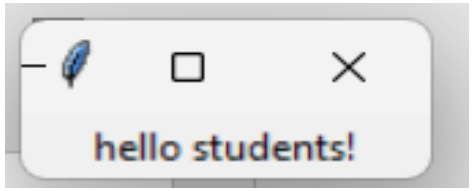

```
import tkinter
m = tkinter.Tk()
'''
widgets are added here
'''
m.mainloop()
```

Gateway classes

Tkinter Widget

- **1. Label**

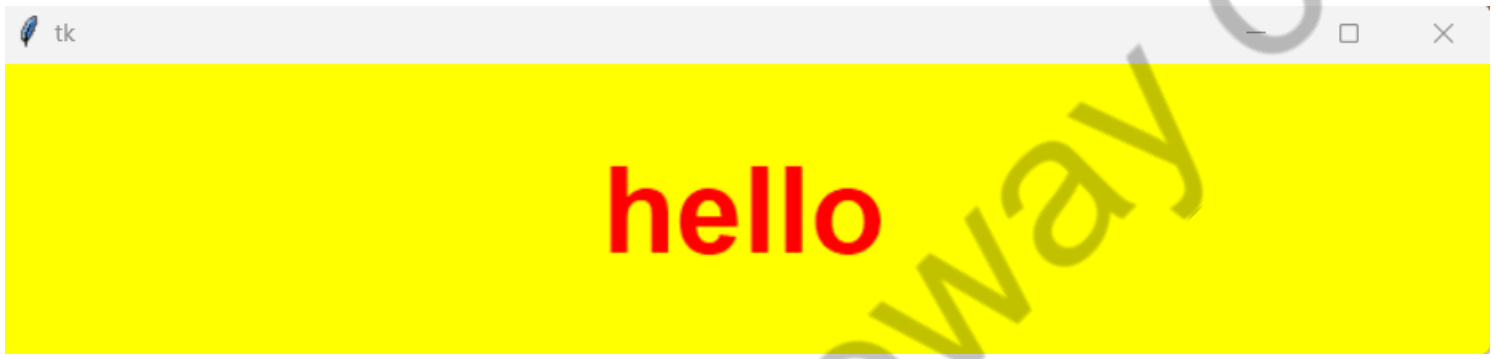
```
from tkinter import *
root = Tk()
w = Label(root, text='hello students!')
w.pack()
root.mainloop()
```



Gateway Classes

Tkinter Widget

```
from tkinter import *
root = Tk()
w = Label(root, text="hello", font=("Arial", 45, "bold"), fg="red",
          bg="yellow",width=20, height=2)
w.pack()
root.mainloop()
```



Gateway Classes

Tkinter Widget

➢ 1. Button

a **Button** is a widget that users can click to perform an action, such as closing a window, running a function, or opening another window

```
import tkinter as tk
r = tk.Tk()
r.title('Counting Seconds')
button = tk.Button(r, text='Stop', width=12, command=r.destroy)
button.pack()
r.mainloop()
```



Gateway Classes

Tkinter Widget

```
import tkinter as tk
r = tk.Tk()
r.title('Counting Seconds')
button = tk.Button(r, text='Stop', width=12, command=r.destroy,
                   fg="white", bg="blue",font=("Arial", 14, "bold"))
button.pack()
r.mainloop()
```



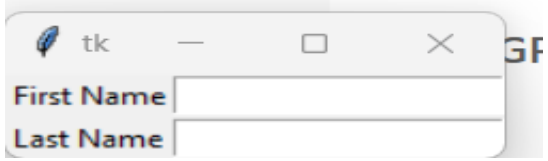
Gateway Classes

3.Entry

The **Entry** widget in Tkinter is used to create a single-line text input field where users can type and edit text. It's commonly used in forms or dialogs to capture user input.

Entry(master, options)

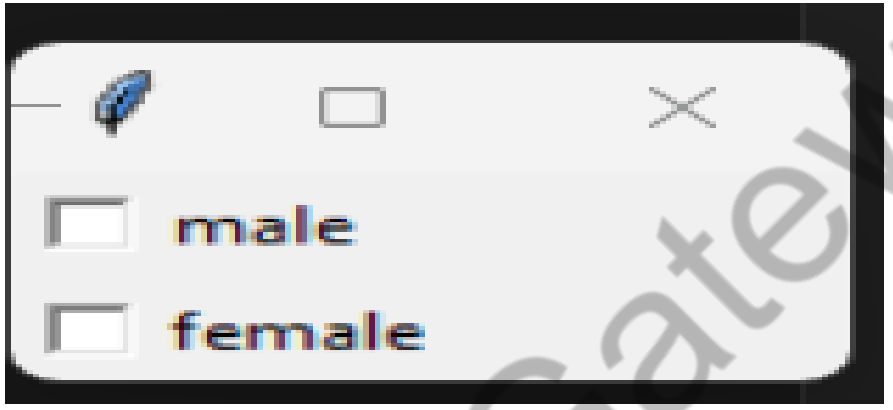
```
from tkinter import *
master = Tk()
Label(master, text='First Name').grid(row=0)
Label(master, text='Last Name').grid(row=1)
e1 = Entry(master)
e2 = Entry(master)
e1.grid(row=0, column=1)
e2.grid(row=1, column=1)
mainloop()
```



Gateway Classes

Checkbutton

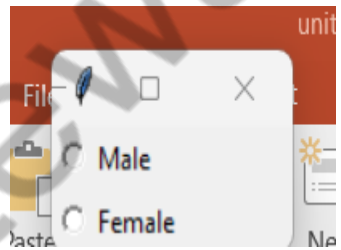
```
from tkinter import *
master = Tk()
var1 = IntVar()
Checkbutton(master, text='male', variable=var1).grid(row=0, sticky=W)
var2 = IntVar()
Checkbutton(master, text='female', variable=var2).grid(row=1, sticky=W)
mainloop()
```



Gateway classes

Radiobutton

```
from tkinter import *
master = Tk()
gender = IntVar()
Radiobutton(master, text='Male', variable=gender, value=1).grid(row=0, sticky=W)
Radiobutton(master, text='Female', variable=gender, value=2).grid(row=1, sticky=W)
mainloop()
```



Gateway classes

What is python module? How to create and import a module in python?

A module in Python is a file containing Python functions, classes, or variables that can be used in other programs. It helps in organizing and reusing code. A module typically has a .py extension.

How to Create a Python Module?

main.py | mymodule.py

```
1  def greet(name):
2      return f"Hello, {name}!"
3
4  pi = 3.14159
```

main.py | mymodule.py

```
1  from mymodule import greet, pi
2  print(greet("rani"))
3  print(pi)
4
```

```
Hello, rani!
3.14159
```

Gateway classes

Different ways to import module

1. Import specific functions/variables

```
from mymodule import greet, pi
print(greet("Bob"))
print(pi)
```

2. Import with an alias

```
import mymodule as mm
print(mm.greet("Charlie"))
print(mm.pi)
```

3.Import everything(*)

```
from mymodule import *
print(greet("David"))
print(pi)
```

Gateway classes

Significance of module in python

1. Code Reusability

Instead of rewriting the same code, you can import and use existing modules

2. Better Code Organization

Helps divide large programs into smaller, manageable files.

3. Avoiding Code Duplication

Saves time by defining functions once and reusing them across different projects.

4.Built-in Modules

Python provides many useful built-in modules like math

Gateway classes

1. Import library

- **Imports the entire module** but requires **module name** when accessing functions or variables.

```
import math
print(math.sqrt(25))
```

- Avoids **name conflicts** (functions/variables are accessed with the module name).
- Makes code more **readable and organized**.
- Need to type **module name** every time

from library import *

Imports everything from the module **directly** into the current namespace (no need to use the module name).

```
from math import *
print(sqrt(25))
```

- Shorter and easier to use functions directly.
- **Risk of name conflicts** (e.g., if another module has sqrt(), it may overwrite).
- Reduces **code clarity** (harder to track which function belongs to which module).

Gateway Classes

Built in function matplotlib

Plot(): used to create line plots.

Bar(): used to create bar chart.

Pie(): used to create pie chart.

hist(): used to create histogram.

Scatter(): use to create scatter plot

Title(): add title to the plot.

Legend(): add a legend.

xlabels() and ylabels(): add label to the axis.

Gateway classes


Thank
you