UNIT -5: PYTHON PACKAGES
ONE SHOT

Today's Target

- MOST IMPORTANT QUESTIONS
- AKTU PYQs

By PRAGYA RAJVANSHI

B.Tech, M.Tech(C.S.E)

AKTU PYQs : Unit-5

Q1	What is python package ? List various step to create a python package
Q2	What is use of _int_.py in python
Q3	What is python module? how to create and import module in python
Q4	What is the significance of module in python? What are the methods of importing a module? Explain with suitable example
Q5	What is the python libraries? List some of the important python libraries
Q6	How can you create python file that can be imported as a library as well as a standalone script
Q7	Difference between import library and from library import*
Q8	Define matplotlib. List some built-in function
Q9	Write a program to plot a simple line graph using matplotlib
Q10	Create a bar chart using matplotlib to represent darta

AKTU PYQs : Unit-5

Q11	Create a histogram using matplotlib
Q12	Define Numpy . List some built-in function in NUMPY
Q13	Write a numpy program to create a 2 d array of sizw2*3 (composed of 4 byte integer elements), also print the shape, size, type and data type of the array
Q14	Define pandas. List some built-in function in Pandas
Q15	Define Pandas Data frame. How do you create a pandas Data frame
Q16	How do you select rows and columns from Panda data frame
Q17	What is tkinter and why it is used in python programming
Q18	How do you create basic window using tkinter?
Q19	What are the widgets in tkinter and can you give example of some commonly used widget

AKTU PYQs : Unit-5

Q20	Define function and write its advantages (2 Marks)
Q21	How to define and call function in python? explain different parts of a function
Q22	Discuss the different types of arguments passing methods in python.wxpalin the variable length argument with any suitable example
Q23	Discuss the scope rule in python (2 Mark)
Q24	Discuss the function in python with its parts and scope .Explain with example (take simple calculator with add, sub, division and multiplication)
Q25	Write apython function remvekth(s,k) that take s as a input string and a n integer k>=0 and removes the character at a index k. if k is beyond the length of s , the whole os string is returned removekth("PYTHON",1) return PTHON removekth("PYTHON",3) return PYTON removekth("PYTHON",20) return PYTHON

→ decorators allows us to wrap another function in order to extend the behaviour of the wrapped function

without permanently modify it

AKTU PYQs : Unit-5

Q26	Write a python code to find out occurrence of an elements in a list
Q27	Write a python function, searchMany(s,x,k) that takes an argument a sequence s and integersx,k(k>0) the function return true if there are at most k occurrence of x in s .otherwise it return false searchMany([10,17,15,12],15,1) return true, searchMany([10,12,12,12],12,2) return false
Q28	What is decorators? (2 marks)

Decorators are a very powerful & useful tool in python. Since it allow programmer to modify the behaviour of a function or class.

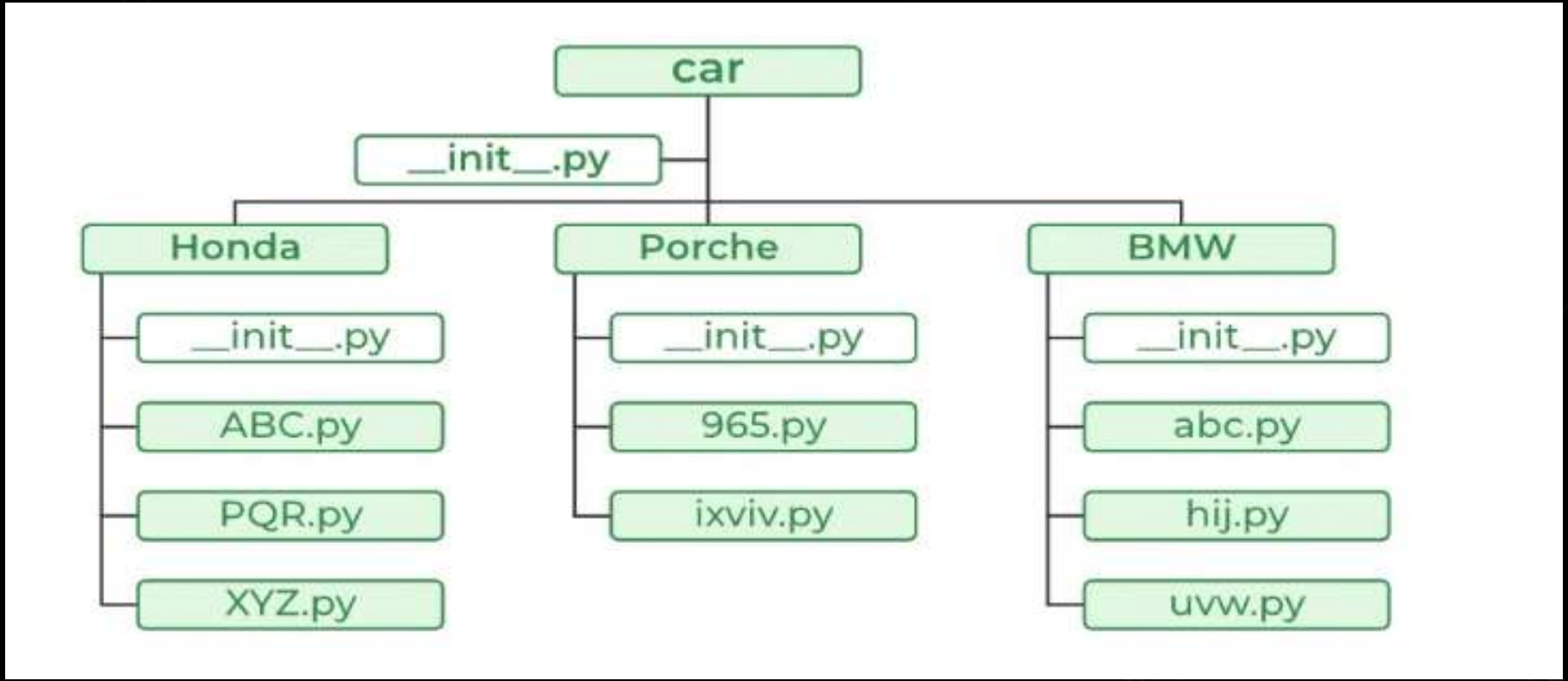
Python Packages

- We usually organize our files in different folders and subfolders based on some criteria, so that they can be managed easily and efficiently. For example, we keep all our games in a Games folder and we can even subcategorize according to the genre of the game or something like that. The same analogy is followed by the packages in Python.
- **What is a Python Package?**
- Python modules may contain several classes, functions, variables, etc. whereas Python packages contain several modules. In simpler terms, a Package in Python is a folder that contains various modules as files.

Creating Package

- Let’s create a package in Python named mypckg that will contain two modules mod1 and mod2. To create this module follow the below steps:
- Create a folder named mypckg.
- Inside this folder create an empty Python file i.e. __init__.py
- Then create two modules mod1 and mod2 in this folder.
- **mod1.py**
- def gfg():
- 	print("Welcome to GFG")
- **mod2.py**
- def sum(a, b):
- 	return a+b

The Hierarchy of our Python package looks like this:



Understanding __init__.py

- **__init__.py helps the Python interpreter recognize the folder as a package. It also specifies the resources to be imported from the modules. If the __init__.py is empty this means that all the functions of the modules will be imported. We can also specify the functions from each module to be made available.**
- **For example, we can also create the __init__.py file for the above module as:**
- **__init__.py**
- from .mod1 import gfg
- from .mod2 import sum
- This __init__.py will only allow the gfg and sum functions from the mod1 and mod2 modules to be imported

We will import the modules from the above-created package and will use the functions inside those modules

- from mypckg import mod1
- from mypckg import mod2
- mod1.gfg()
- res = mod2.sum(1, 2)
- print(res)
- OUTPUT
- Welcome to GFG
- 3

We can also import the specific function also using the same syntax

- from mypckg.mod1 import gfg
- from mypckg.mod2 import sum
- gfg()
- res = sum(1, 2)
- print(res)
- OUTPUT
- Welcome to GFG
- 3

Python Modules

- Python Module is a file that contains built-in functions, classes, and variables. There are many Python modules, each with its specific work,
- A Python module is a file containing Python definitions and statements. A module can define functions, classes, and variables. A module can also include runnable code.
- Grouping related code into a module makes the code easier to understand and use. It also makes the code logically organized
- **The significance of Modules is as follows:**
- It reduces the complexity of programs.
- Its content can be reused in other programs, without having to rewrite them.
- It creates a number of well-defined, documented boundaries within the program.

Create a Python Module

- To create a Python module, write the desired code and save that in a file with .py extension.

```
1  # A simple module, calc.py
2  def add(x, y):
3      return (x+y)
4
5  def subtract(x, y):
6      return (x-y)
7  )
8
```

Import module in Python

- We can import the functions, and classes defined in a module to another module using the import statement in some other Python source file.
- When the interpreter encounters an import statement, it imports the module if the module is present in the search path
- Note: A search path is a list of directories that the interpreter searches for importing a module.
- Syntax to Import Module in Python
- import module

Different methods of importing a module

- Importing modules
- Import Specific Attributes from a Python module
- Import the whole module and rename it, usually using a shorter variable name:
- import specific things from the module and rename them as you're importing them

1. Importing modules

```
1  # importing sqrt() and factorial from the
2  # module math
3  import math
4  print(math.sqrt(16))
5  print(math.factorial(6))
6
7
```

```
4.0
720
```

2.Import Specific Attributes from a Python module

```
1  # importing sqrt() and factorial from the
2  # module math
3  from math import sqrt, factorial
4  print(sqrt(16))
5  print(factorial(6))
6
7
```

```
4.0
720


...Program finished with exit code 0
Press ENTER to exit console.
```

3. Import the whole module and rename it, usually using a shorter variable name:

```
1  # importing sqrt() and factorial from the
2  # module math
3  import math as mt
4
5  # if we simply do "import math", then
6  # math.sqrt(16) and math.factorial()
7  # are required.
8  print(mt.sqrt(16))
9  print(mt.factorial(6))
10
11
```

```
4.0
720
```

4.import specific things from the module and rename them as you're importing them

```
1 from math import sqrt as sq
2 print(sq(3))
3
```

```
1.7320508075688772


...Program finished with exit code 0
Press ENTER to exit console.
```

Python Library

- Normally, a library is a collection of books or a room or place where many books are stored to be used later. Similarly, in the programming world, a library is a collection of precompiled codes that can be used later on in a program for some specific well-defined operations. Other than pre-compiled codes, a library may contain documentation, configuration data, message templates, classes, values, etc.
- A Python library is a collection of related modules. It contains bundles of code that can be used repeatedly in different programs. It makes Python Programming simpler and more convenient for the programmer. As we don't need to write the same code again and again for different programs. Python libraries play a very vital role in fields of Machine Learning, Data Science, Data Visualization, etc
- As stated above, a Python library is simply a collection of codes or modules of codes that we can use in a program for specific operations. We use libraries so that we don't need to write the code again in our program that is already available. But how it works? Actually, in the MS Windows environment, the library files have a DLL extension (Dynamic Load Libraries). When we link a library with our program and run that program, the linker automatically searches for that library. It extracts the functionalities of that library and interprets the program accordingly

Python standard library

- The Python Standard Library contains the exact syntax, semantics, and tokens of Python. It contains built-in modules that provide access to basic system functionality like I/O and some other core modules. Most of the Python Libraries are written in the C programming language. The Python standard library consists of more than 200 core modules. All these work together to make Python a high-level programming language. Python Standard Library plays a very important role. Without it, the programmers can’t have access to the functionalities of Python. But other than this, there are several other libraries in Python that make a programmer’s life easier. Let’s have a look at some of the commonly used libraries

Python standard library

1.TensorFlow: This library was developed by Google in collaboration with the Brain Team. It is an open-source library used for high-level computations(dot product, addition, exponent, division etc). It is also used in machine learning and deep learning algorithms. It contains a large number of tensor operations. Researchers also use this Python library to solve complex computations in Mathematics and Physics.

2.Matplotlib: This library is responsible for plotting numerical data. And that's why it is used in data analysis. It is also an open-source library and plots high-defined figures like pie charts, histograms, scatterplots, graphs, etc

3.Pandas: Pandas are an important library for data scientists. It is an open-source machine learning library that provides flexible high-level data structures and a variety of analysis tools. It eases data analysis, data manipulation, and cleaning of data. Pandas support operations like Sorting, Re-indexing, Iteration, Concatenation, Conversion of data, Visualizations, Aggregations, etc.

Python standard library

4. Numpy: The name “Numpy” stands for “Numerical Python”. It is the most commonly used library. It is a popular machine-learning library that supports large matrices and multi-dimensional data. It consists of in-built mathematical functions for easy computations. Even libraries like Tensor Flow use Numpy internally to perform several operations on tensors. Array Interface is one of the key features of this library

5.SciPy: The name “SciPy” stands for “Scientific Python”. It is an open-source library used for high-level scientific computations. This library is built over an extension of Numpy. It works with Numpy to handle complex computations. While Numpy allows sorting and indexing of array data, the numerical data code is stored in SciPy. It is also widely used by application developers and engineers.

6. Scrapy: It is an open-source library that is used for extracting data from websites. It provides very fast web crawling(automatically accessing a website and obtaining data via software program) and high-level screen scraping. It can also be used for data mining and automated testing of data.

Python standard library

7. Scikit-learn: It is a famous Python library to work with complex data. Scikit-learn is an open-source library that supports machine learning. It supports various supervised and unsupervised algorithms like linear regression, classification, clustering, etc. This library works in association with Numpy and SciPy.

8.PyGame: This library provides an easy interface to the Standard Directmedia Library (SDL) platform-independent graphics, audio, and input libraries. It is used for developing video games using computer graphics and audio libraries along with Python programming language.

9.PyTorch: PyTorch is the largest machine learning library that optimizes tensor computations. It has rich APIs to perform tensor computations with strong GPU acceleration. It also helps to solve application issues related to neural networks.

10.PyBrain: The name "PyBrain" stands for Python-based Based Reinforcement Learning, Artificial Intelligence, and Neural Networks library. It is an open-source library built for beginners in the field of Machine Learning. It provides fast and easy-to-use algorithms for machine-learning tasks. It is so flexible and easily understandable and that's why is helpful for developers that are new in research fields

DIFFERENCE BETWEEN-

S.NO	Import library	From library import*
1	Python import loads a python module/library into its own namespace	From loads a python module into the current name space
2	By using import , we include all the codes inside the module or library	We includes all the code or only the required function , classes present inside the module or library
3	Dot operator is used	Dot operator is not required
4	It is required to mention the module name followed by dot to access any names in the module	It is not needed to mention the module name to access any names in the module
5		

DIFFERENCE BETWEEN-

S.NO	Import library	From library import*

main.py

```
1
2 import math
3 print(math.sqrt(16))
4
5
6
7
```

```
4.0


...Program finished with exit code 0
Press ENTER to exit console.
```

main.py

```
1
2 from math import *
3 print(sqrt(16))
4
5
6
7
```

```
4.0


...Program finished with exit code 0
Press ENTER to exit console.
```

Python Numpy

- Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object and tools for working with these arrays. It is the fundamental package for scientific computing with Python
- Array in Numpy is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In Numpy, several dimensions of the array are called the rank of the array.
- A tuple of integers giving the size of the array along each dimension is known as the shape of the array. An array class in Numpy is called ndarray.
- Elements in Numpy arrays are accessed by using square brackets and can be initialized by using nested Python Lists.

Python Numpy

```python
# Python program for
# Creation of Arrays
import numpy as np

# Creating a rank 1 Array
arr = np.array([1, 2, 3])
print("Array with Rank 1: \n",arr)

# Creating a rank 2 Array
arr = np.array([[1, 2, 3],
                [4, 5, 6]])
print("Array with Rank 2: \n", arr)

# Creating an array from tuple
arr = np.array((1, 3, 2))
print("\nArray created using "
      "passed tuple:\n", arr)

```

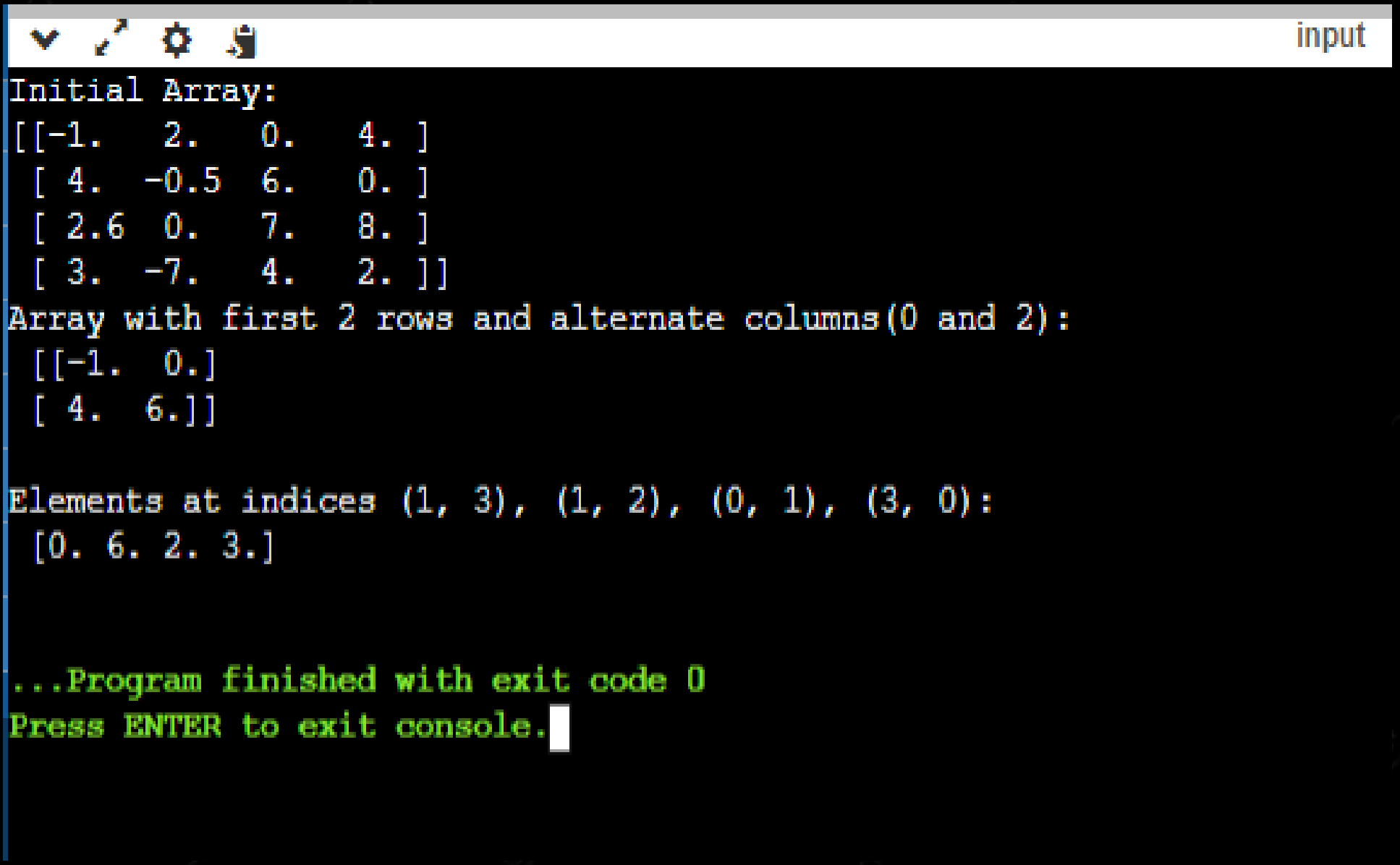
input

```
Array with Rank 1:
 [1 2 3]
Array with Rank 2:
 [[1 2 3]
 [4 5 6]]

Array created using passed tuple:
 [1 3 2]
```

main.py

```python
 #Python program to demonstrate
# indexing in numpy array
import numpy as np

# Initial Array
arr = np.array([[-1, 2, 0, 4],
                [4, -0.5, 6, 0],
                [2.6, 0, 7, 8],
                [3, -7, 4, 2.0]])
print("Initial Array: ")
print(arr)

# Printing a range of Array
# with the use of slicing method
sliced_arr = arr[:2, ::2]
print ("Array with first 2 rows and"
       " alternate columns(0 and 2):\n", sliced_arr)

# Printing elements at
# specific Indices
Index_arr = arr[[1, 1, 0, 3],
                [3, 2, 1, 0]]
print ("\nElements at indices (1, 3), "
       "(1, 2), (0, 1), (3, 0):\n", Index_arr)

```

GW
GATEWAY CLASSES



GW
GATEWAY CLASSES

Python PIP <2 Marks

- Package refers to a distribution of Python code that includes one or more modules or libraries. These packages are typically published on the Python Package Index (PyPI) and can be easily installed using pip.
- Python packages may contain modules, sub-packages, and additional resources such as documentation and data files
- Python PIP is the package manager for Python packages. We can use PIP to install packages that do not come with Python

Plotting with Matplotlib in Python

- Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002. One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram, etc.
- Installation: Windows, Linux, and macOS distributions have Matplotlib and most of its dependencies as wheel packages. Run the following command to install the Matplotlib package. But before that make sure Python and PIP are already installed on a system

Plotting with Matplotlib in Python

- After checking Python and PIP in your system, You need to run this command to install Matplotlib.
- python -m pip install -U matplotlib
- **Importing Matplotlib**
- After successfully installing Matplotlib, You can use this command to import Matplotlib on your system.
- import matplotlib

Types of Matplotlib

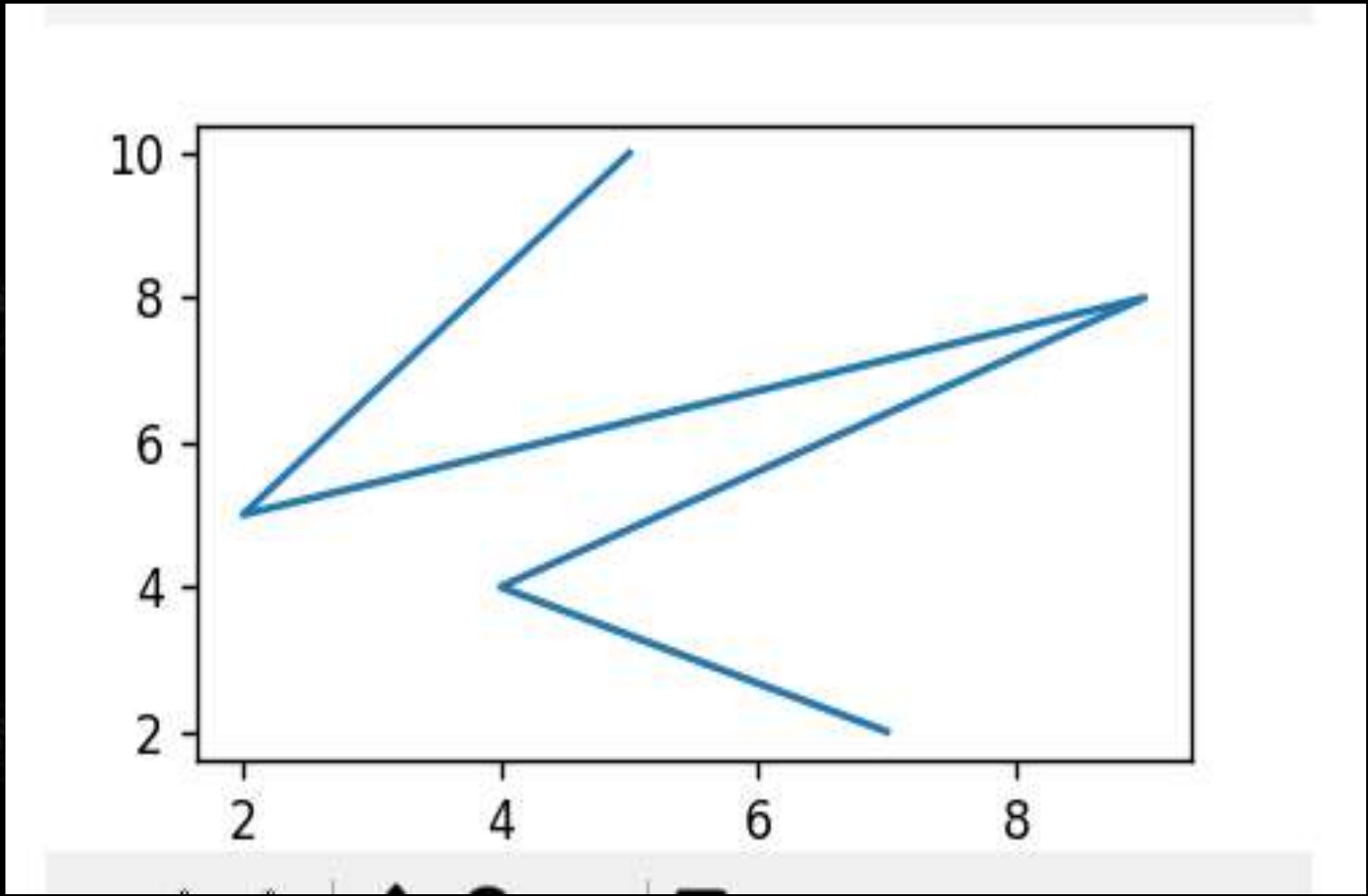
- Matplotlib comes with a wide variety of plots. Plots help to understand trends, and patterns, and to make correlations. They're typically instruments for reasoning about quantitative information. Some of the sample plots are covered here.
- Matplotlib Line Plot
- Matplotlib Bar Plot
- Matplotlib Histograms Plot
- Matplotlib Scatter Plot
- Matplotlib Pie Charts
- Matplotlib Area Plot

Matplotlib Line Plot

- By importing the matplotlib module, defines x and y values for a plotPython, plots the data using the plot() function and it helps to display the plot by using the show() function . The plot() creates a line plot by connecting the points defined by x and y values.

```
1  # importing matplotlib module
2  from matplotlib import pyplot as plt
3
4  # x-axis values
5  x = [5, 2, 9, 4, 7]
6
7  # Y-axis values
8  y = [10, 5, 8, 4, 2]
9
10 # Function to plot
11 plt.plot(x, y)
12
13 # function to show the plot
14 plt.show()
15
16
```

Matplotlib Line Plot

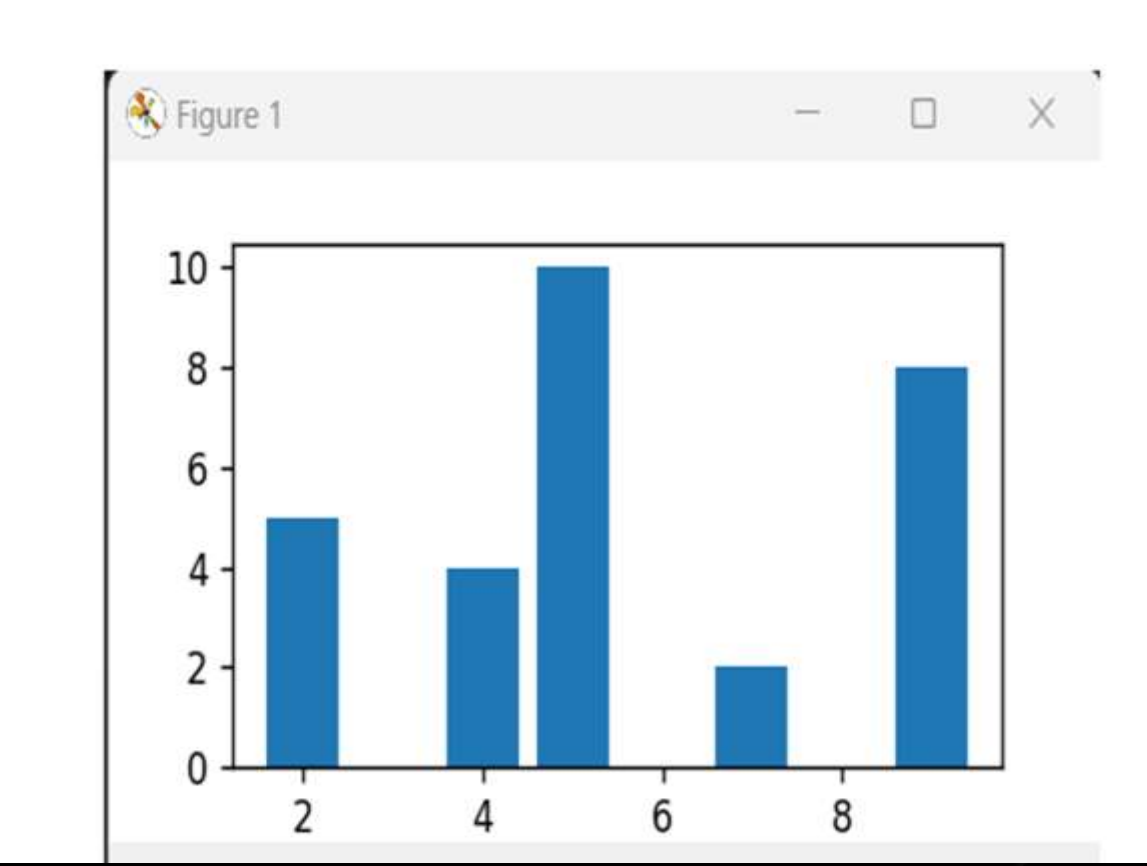


Matplotlib Bar Plot

- By using matplotlib library in Pythontwo, it allows us to access the functions and classes provided by the library for plotting. There are tow lists x and y are defined . This function creates a bar plot by taking x-axis and y-axis values as arguments and generates the bar plot based on those value.

```
1  # importing matplotlib module
2  from matplotlib import pyplot as plt
3
4  # x-axis values
5  x = [5, 2, 9, 4, 7]
6
7  # Y-axis values
8  y = [10, 5, 8, 4, 2]
9
10 # Function to plot the bar
11 plt.bar(x, y)
12
13 # function to show the plot
14 plt.show()
15
16
17
```

Matplotlib Bar Plot

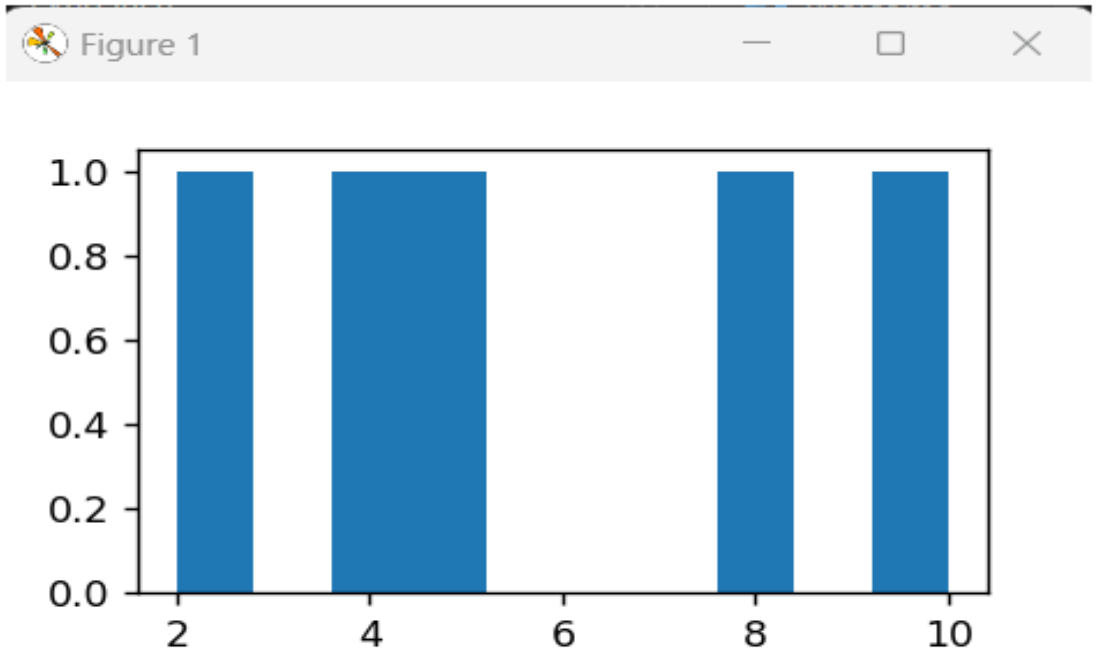


Matplotlib Histograms Plot

- By using the matplotlib module defines the y-axis values for a histogram plot. Plots in the ,histogram using the hist() function and displays the plot using the show() function. The hist() function creates a histogram plot based on the values in the y-axis list.

```
1  # importing matplotlib module
2  from matplotlib import pyplot as plt
3
4  # Y-axis values
5  y = [10, 5, 8, 4, 2]
6
7  # Function to plot histogram
8  plt.hist(y)
9
10 # Function to show the plot
11 plt.show()
12
13
14
15
16
```

Matplotlib Histograms Plot



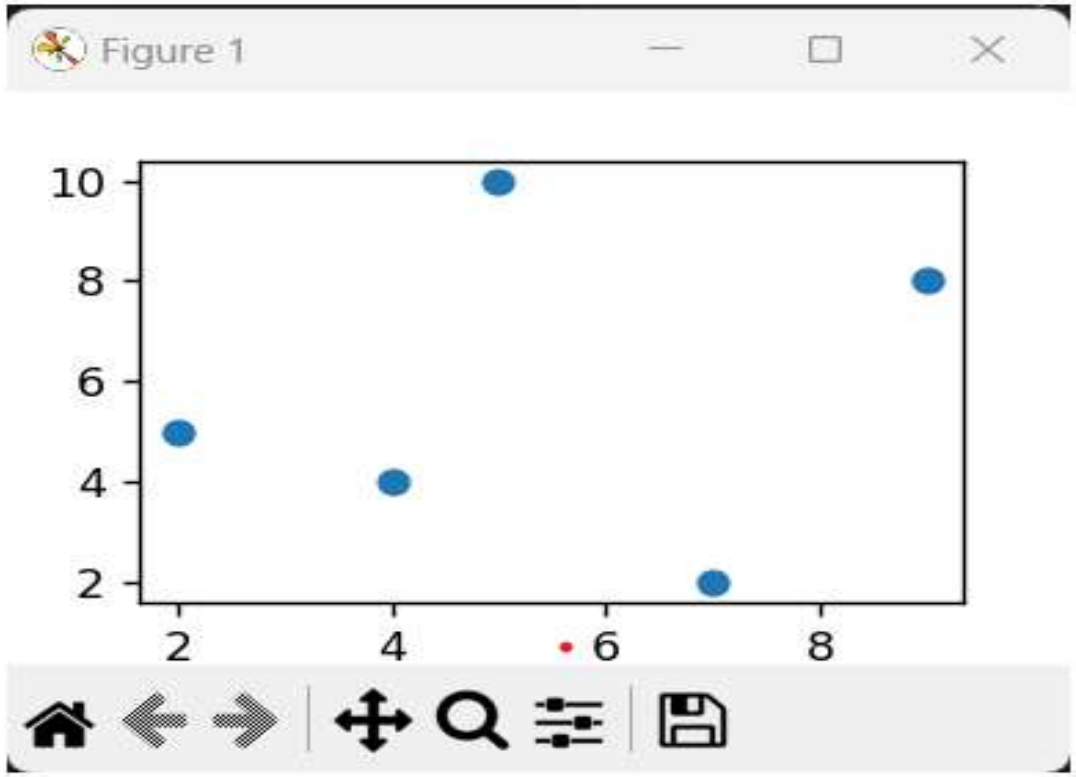
Histogram

Matplotlib Scatter Plot

- By imports, plot the matplotlib module, define x and y values for a scatter plot, plots the data using the scatter() function, and displays the plot using the show() function. The scatter() function creates a scatter plot by plotting individual data points defined by the x and y values

```
1  # importing matplotlib module
2  from matplotlib import pyplot as plt
3
4  # x-axis values
5  x = [5, 2, 9, 4, 7]
6
7  # Y-axis values
8  y = [10, 5, 8, 4, 2]
9
10 # Function to plot scatter
11 plt.scatter(x, y)
12
13 # function to show the plot
14 plt.show()
15
16
17
18
19
```

Matplotlib Scatter Plot



Scatter Plot

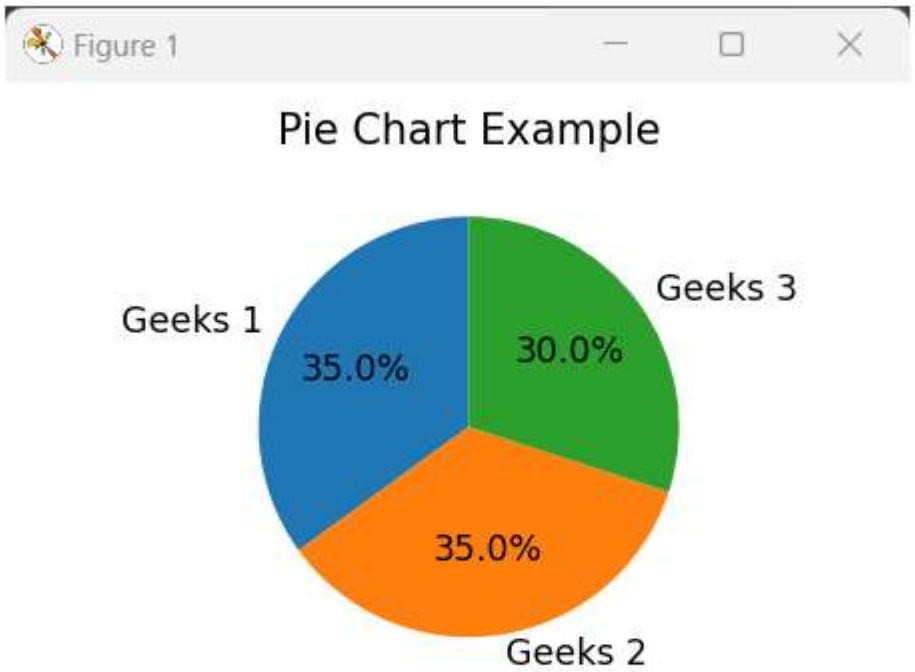
Matplotlib Pie Charts

- By importing the module Matplotlib in Python to create a pie chart with three categories and respective sizes. The plot .pie() function is used to generate the chart, including labels, percentage formatting, and a starting angle.
- A title is added with plt. title(), and plt. show() displays the resulting pie chart, visualizing the proportional distribution of the specified categories.

main.py

```
1  import matplotlib.pyplot as plt
2
3  # Data for the pie chart
4  labels = ['Geeks 1', 'Geeks 2', 'Geeks 3']
5  sizes = [35, 35, 30]
6
7  # Plotting the pie chart
8  plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=90)
9  plt.title('Pie Chart Example')
10 plt.show()
11
12
13
14
15
16
17
18
```

Matplotlib Pie Charts



Pie Chart

Matplotlib Area Plot

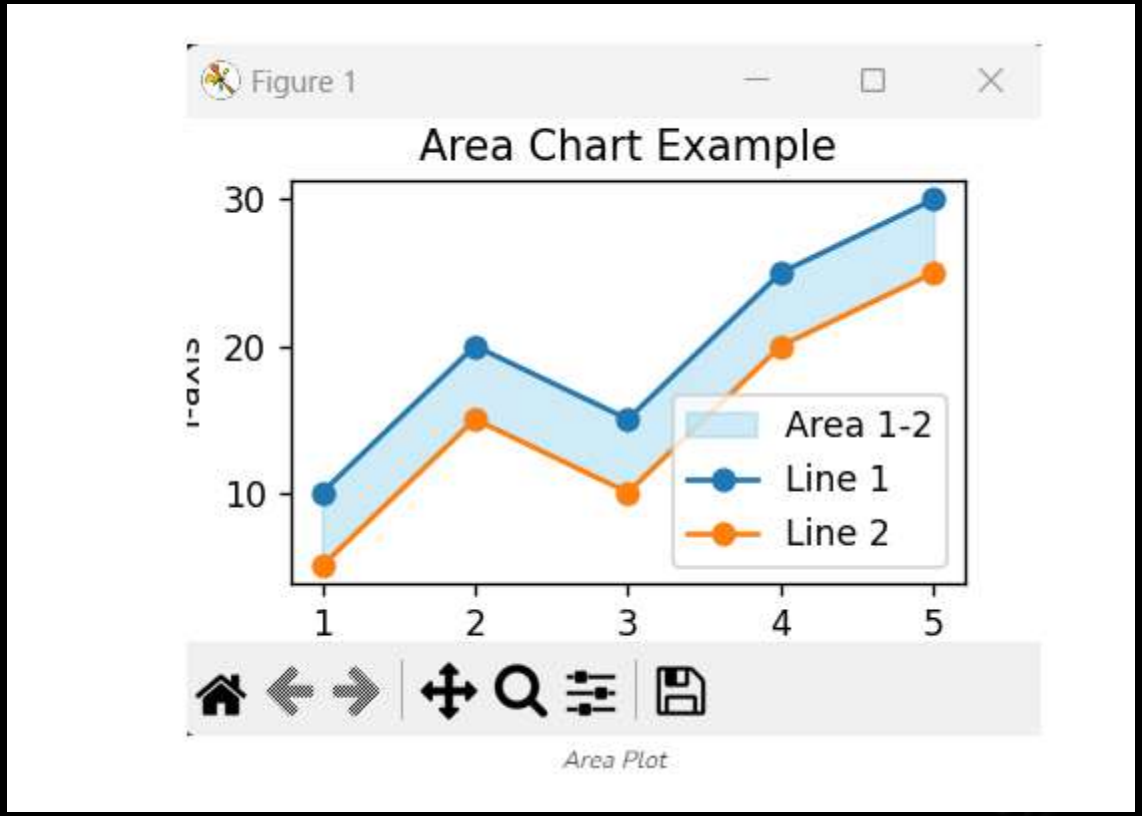
- By importing Matplotlib we sky blue generated an area chart with two lines (‘Line 1’ and ‘Line 2’). The area between the lines is shaded in a sky-blue color with 40% transparency.
- The x-axis values are in the list ‘x’, and the corresponding y-axis values for each line are in ‘y1’ and ‘y2’. Labels, titles legends, and legends are added, and the resulting area chart is displayed.

Matplotlib Area Plot

ain.py

```python
import matplotlib.pyplot as plt

# Data
x = [1, 2, 3, 4, 5]
y1, y2 = [10, 20, 15, 25, 30], [5, 15, 10, 20, 25]

# Area Chart
plt.fill_between(x, y1, y2, color='skyblue', alpha=0.4, label='Area 1-2')
plt.plot(x, y1, label='Line 1', marker='o')
plt.plot(x, y2, label='Line 2', marker='o')

# Labels and Title
plt.xlabel('X-axis'), plt.ylabel('Y-axis'), plt.title('Area Chart Example')

# Legend and Display
plt.legend(), plt.show()
.
```

Matplotlib Area Plot



Numpy Functions in Python

- **1.np.array():** This function is used to create an array in NumPy.

main.py

```
1  import numpy as np
2
3  arr = np.array([1, 2, 3])
4  print(arr)
5
6
```

```
[1 2 3]
```

Numpy Functions in Python

- **2.np.arange():** This function is used to create an array with a range of values.

```
1  import numpy as np
2
3  arr = np.arange(1, 6)
4  print(arr)
5
6
```

```
[1 2 3 4 5]
```

Numpy Functions in Python

- **3.np.zeros():** This function is used to create an array filled with zeros

main.py

```
1  import numpy as np
2
3  arr = np.zeros(3)
4  print(arr)
5
6
```

```
[0. 0. 0.]
```

Numpy Functions in Python

- **4.np.ones():** This function is used to create an array filled with ones.

```
1  import numpy as np
2  arr=np.ones(3)
3  print(arr)
4
```

```
[1. 1. 1.]
```

Numpy Functions in Python

- **5.Numpy.square:** This mathematical function helps user to calculate square value of each element in the array

```
1
2  import numpy as np
3  arr1 = [1, -3, 15, -466]
4  print ("Square Value of arr1 : \n", np.square(arr1))
5  arr2 = [23 ,-56]
6  print ("\nSquare Value of arr2 : ", np.square(arr2))
7
8
9
10
```

input

```
Square Value of arr1 :
 [     1      9    225 217156]

Square Value of arr2 :  [ 529 3136]


...Program finished with exit code 0
Press ENTER to exit console.
```

Numpy Functions in Python

- **6. np.random.rand():** This function is used to create an array with random values between 0 and 1.

main.py

```
1  import numpy as np
2
3  arr = np.random.rand(3)
4  print(arr)
5
6
7
8
```

```
[0.32498081 0.00193826 0.62767232]


...Program finished with exit code 0
Press ENTER to exit console.
```

Numpy Functions in Python

- **7. np.random.randint(): This function is used to create an array with random integer values between a specified range.**

```
1  import numpy as np
2
3  arr = np.random.randint(0, 10, 5)
4  print(arr)
5
6
7
8
```

```
[1 1 6 2 2]


...Program finished with exit code 0
Press ENTER to exit console.
```

Numpy Functions in Python

- **8. np.max():** This function is used to find the maximum value in an array

main.py

```
1  import numpy as np
2
3  arr = np.array([1, 2, 3])
4  max_value = np.max(arr)
5  print(max_value)
6
7
8
```

```
3


...Program finished with exit code 0
Press ENTER to exit console.
```

Numpy Functions in Python

- **9.np.min():** This function is used to find the minimum value in an array

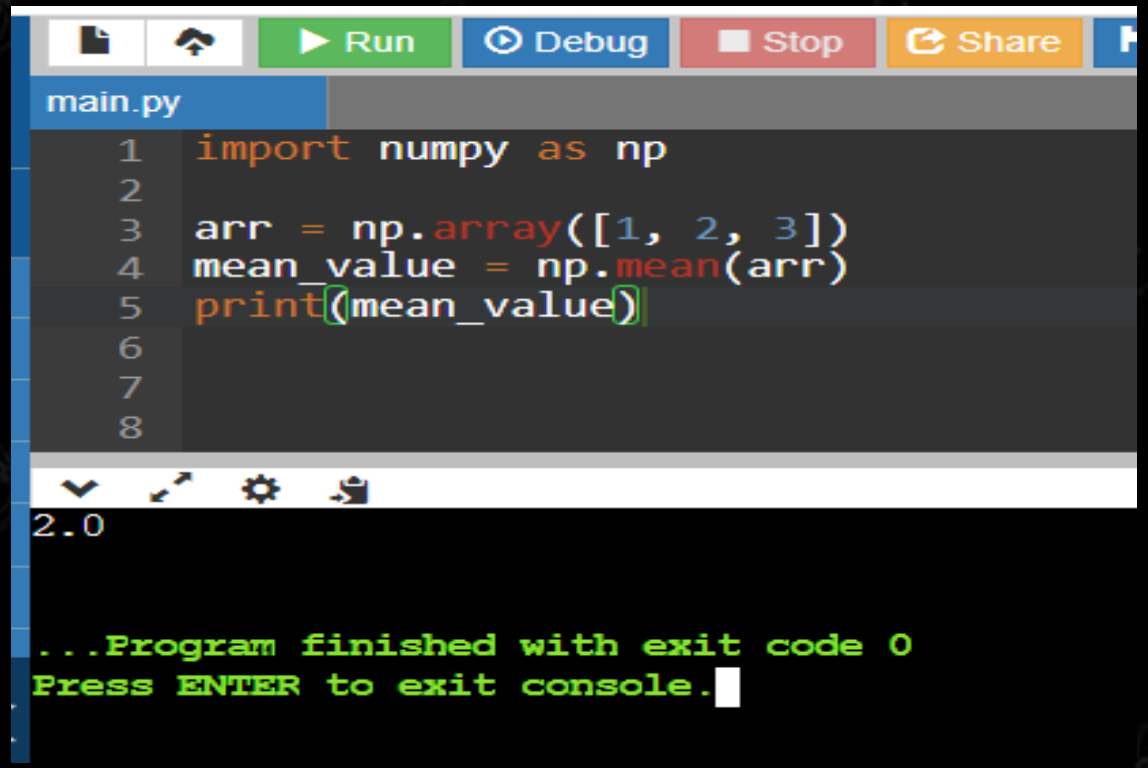
```
1  import numpy as np
2
3  arr = np.array([1, 2, 3])
4  min_value = np.min(arr)
5  print(min_value)
6
7
8
```

```
1
```

```
...Program finished with exit code 0
Press ENTER to exit console.
```

Numpy Functions in Python

- **10 np.mean():** This function is used to find the mean value of an array



Numpy Functions program

- Write a numpy program to create a 2 d array of size2*3 (composed of 4 byte integer elements), also print the shape, size, type and data type of the array

main.py

```
1  import numpy as np
2  x=np.array([[2,4,6],[6,8,9]],np.int32)
3  print(type(x))
4  print(x.shape)
5  print(x.dtype)
6
7
8
```

```
<class 'numpy.ndarray'>
(2, 3)
int32
```

What is Pandas?

- Pandas is a powerful and versatile library that simplifies tasks of data manipulation in Python.
- Pandas is built on top of the NumPy library and is particularly well-suited for working with tabular data, such as spreadsheets or SQL tables.
- Its versatility and ease of use make it an essential tool for data analysts, scientists, and engineers working with structured data in Python.
- What can you do using Pandas?
- Pandas are generally used for data science but have you wondered why? This is because pandas are used in conjunction with other libraries that are used for data science. It is built on the top of the NumPy library which means that a lot of structures of NumPy are used or replicated in Pandas.
- The data produced by Pandas are often used as input for plotting functions of Matplotlib, statistical analysis in SciPy, and machine learning algorithms in Scikit-learn. Here is a list of things that we can do using Panda Data set cleaning, merging, and joining.

What is Pandas?

- Easy handling of missing data (represented as NaN) in floating point as well as non-floating point data.
- Columns can be inserted and deleted from Data Frame and higher dimensional objects.
- Powerful group by functionality for performing split-apply-combine operations on data sets.
- Data Visualization

Getting Started with Pandas

Installing Pandas

- The first step of working with pandas is to ensure whether it is installed in the system or not. If not then we need to install it in our system using the pip command. Type the cmd command in the search box and locate the folder using the cd command where the python-pip file has been installed. After locating it, type the command:
- pip install pandas

Getting Started with Pandas

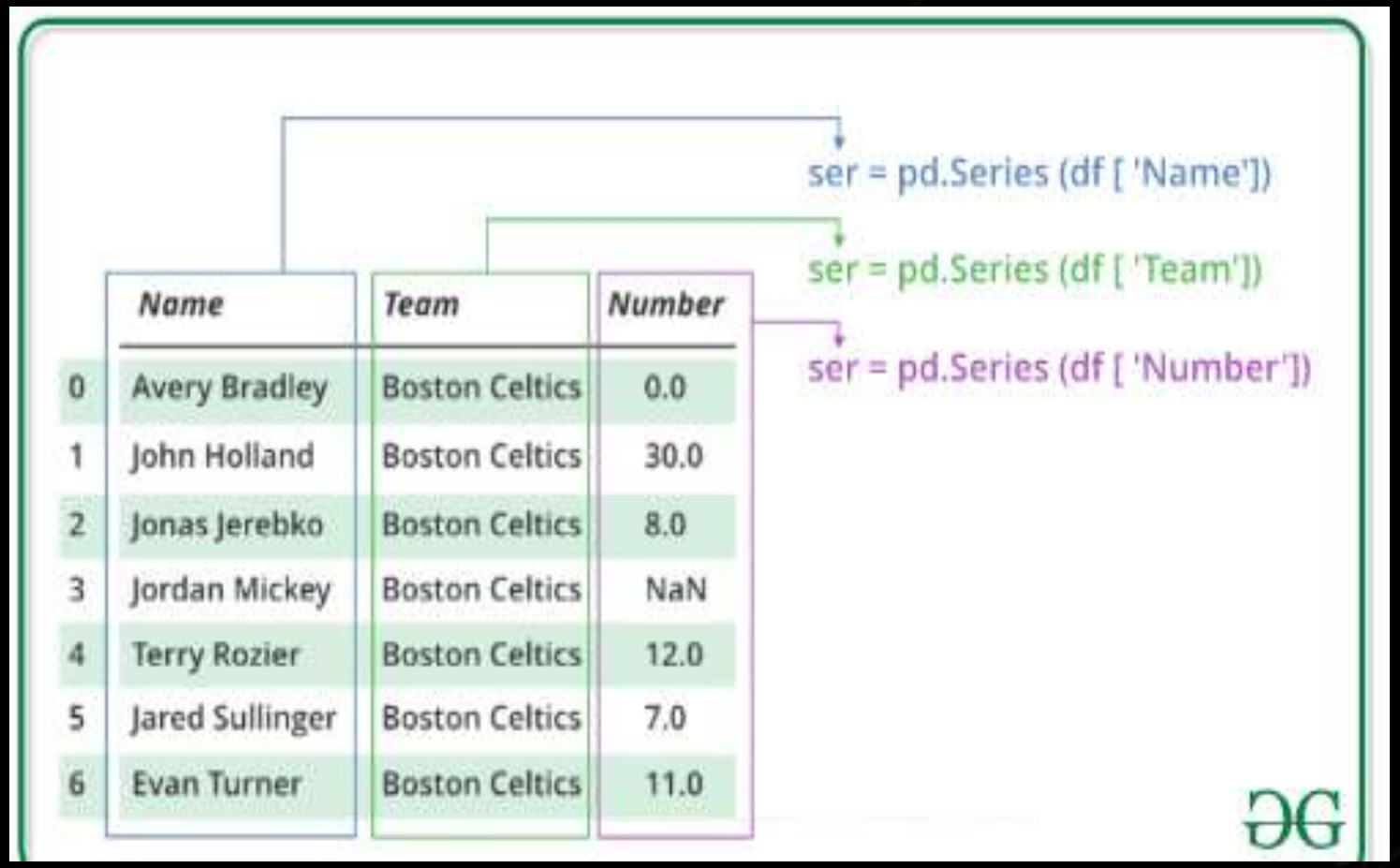
Importing Pandas

- After the pandas have been installed into the system, you need to import the library. This module is generally imported as follows:
- import pandas as pd
- Here, pd is referred to as an alias to the Pandas. However, it is not necessary to import the library using the alias, it just helps in writing less amount code every time a method or property is called.

Pandas Data Structures

- Pandas generally provide two data structures for manipulating data, They are:
- Series, DataFrame
- **Pandas Series**
- A Pandas Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called indexes.
- Pandas Series is nothing but a column in an Excel sheet. Labels need not be unique but must be a hashable type. The object supports both integer and label-based indexing and provides a host of methods for performing operations involving the index.

Pandas Data Structures



	Name	Team	Number
0	Avery Bradley	Boston Celtics	0.0
1	John Holland	Boston Celtics	30.0
2	Jonas Jerebko	Boston Celtics	8.0
3	Jordan Mickey	Boston Celtics	NaN
4	Terry Rozier	Boston Celtics	12.0
5	Jared Sullinger	Boston Celtics	7.0
6	Evan Turner	Boston Celtics	11.0

Creating a Pandas Series

- In the real world, a Pandas Series will be created by loading the datasets from existing storage, storage can be SQL Database, CSV file, and Excel file. Pandas Series can be created from the lists, dictionaries, and from a scalar value, etc. Series can be created in different ways, here are some ways by which we create a series:

- Creating a series from the array: In order to create a series from the array, we have to import a numpy module and have to use the array() function.

Creating a Pandas Series

main.py

```
# import numpy as np
import numpy as np

# simple array
data = np.array(['g','e','e','k','s'])

ser = pd.Series(data)
print(ser)
```

input

```
0    g
1    e
2    e
3    k
4    s
dtype: object
```

Accessing element of Series

main.py

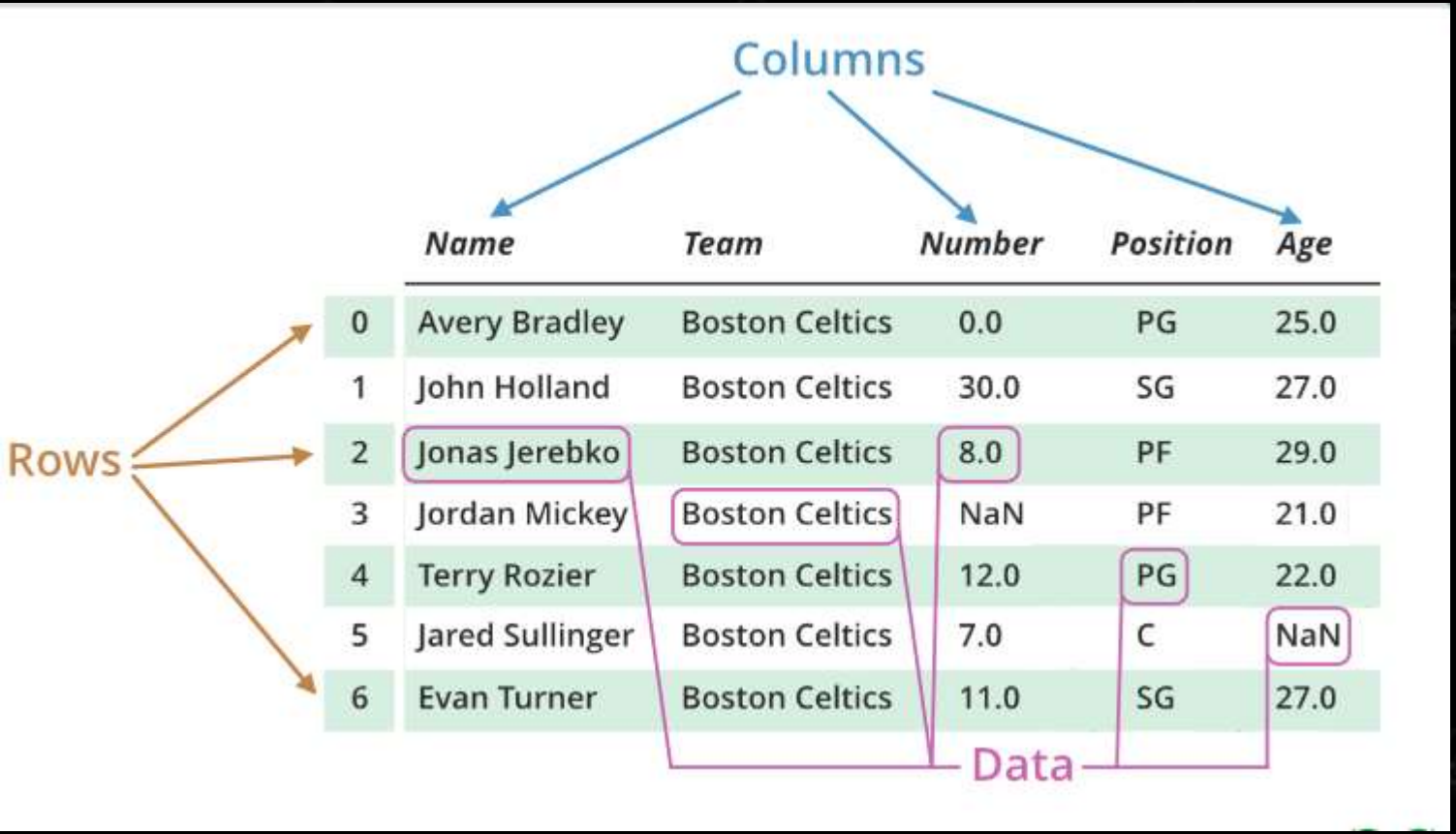
```
4  # import pandas and numpy
5  import pandas as pd
6  import numpy as np
7
8  # creating simple array
9  data = np.array(['g','e','e','k','s','f', 'o','r','g','e','e','k','s'])
10 ser = pd.Series(data)
11
12 #retrieve the first element
13 print(ser[:5])
14
15
```

input

```
0    g
1    e
2    e
3    k
4    s
dtype: object
```

Python Pandas DataFrame

. Pandas DataFrame is a 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 Data Frame 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

Creating a Pandas DataFrame

main.py

```
3  # import pandas as pd
4  import pandas as pd
5
6  # list of strings
7  lst = ['Geeks', 'For', 'Geeks', 'is',
8                'portal', 'for', 'Geeks']
9
10 # Calling DataFrame constructor on list
11 df = pd.DataFrame(lst)
12 print(df)
13
14
```

```
        0
0   Geeks
1     For
2   Geeks
3      is
4  portal
5     for
5   Geeks
```

Creating DataFrame from dict of ndarray/lists:

```python
# Python code demonstrate creating
# DataFrame from dict narray / lists
# By default addresses.

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)
```

input

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

...Program finished with exit code 0
Press ENTER to exit console.
```

Dealing with rows and column

- A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. We can perform basic operations on rows/columns like selecting, deleting, adding, and renaming.
- Column Selection: In Order to select a column in Pandas DataFrame, we can either access the columns by calling them by their column name.
- Row Selection: Pandas provide a unique method to retrieve rows from a Data frame. DataFrame.loc[] method is used to retrieve rows from Pandas DataFrame. Rows can also be selected by passing integer location to an iloc[] function

Dealing with column

```python
# 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)

# select two columns
print(df[['Name', 'Qualification']])

```

input

```
     Name Qualification
0     Jai           Msc
1  Princi            MA
2  Gaurav           MCA
3    Anuj           Phd


...Program finished with exit code 0
Press ENTER to exit console.
```

Dealing with rows

```python
# importing pandas package
import pandas as pd

# making data frame from csv file
data = pd.read_csv("nba.csv", index_col ="Name")

# retrieving row by loc method
first = data.loc["Avery Bradley"]
second = data.loc["R.J. Hunter"]


print(first, "\n\n\n", second)
```

Dealing with rows

```
Team            Boston Celtics
Number                       0
Position                    PG
Age                         25
Height                     6-2
Weight                     180
College                  Texas
Salary            7.73034e+06
Name: Avery Bradley, dtype: object


 Team            Boston Celtics
Number                       28
Position                     SG
Age                          22
Height                      6-5
Weight                      185
College         Georgia State
Salary            1.14864e+06
Name: R.J. Hunter, dtype: object
```

Dealing with rows and column

- A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. We can perform basic operations on rows/columns like selecting, deleting, adding, and renaming.
- Column Selection: In Order to select a column in Pandas DataFrame, we can either access the columns by calling them by their columns name.
- Row Selection: Pandas provide a unique method to retrieve rows from a Data frame. DataFrame.loc[] method is used to retrieve rows from Pandas DataFrame. Rows can also be selected by passing integer location to an iloc[] function

List of Important Pandas Functions

FUNCTION	DISCRIPTION
Pandas read_csv() Function	This function is used to retrieve data from CSV files in the form of a dataframe
Pandas head() Function	This function is used to return the top n (5 by default) values of a data frame or series
Pandas tail() Function	This method is used to return the bottom n (5 by default) rows of a data frame or series
Pandas sample() Function	This method is used to generate a sample random row or column from the data frame.
Pandas info() Function	This method is used to generate the summary of the DataFrame, this will include info about columns with their names, their datatypes, and missing values
Pandas dtypes() Function	This method returns a Series with the data type of each colum

List of Important Pandas Functions

FUNCTION	DISCRIPTION
Pandas shape() Function	It returns a tuple representing the dimensionality of the Pandas DataFrame.
Pandas size() Function	This method returns the number of rows in the Series. Otherwise, return the number of rows times the number of columns in the DataFrame.
Pandas ndim() Function	This function returns 1 if Series and 2 if DataFrame
Pandas describe() Function	Returns descriptive statistics about the data like mean, minimum, maximum, standard deviation, etc.
Pandas unique() Function	It returns all the unique values in a particular column.
Pandas nunique() Function	Returns the number of unique values in the column

List of Important Pandas Functions

FUNCTION	DISCRIPTION
Pandas isnull() Function	Returns the DataFrame/Series of the boolean values. Missing values gets mapped to True and non-missing value gets mapped to False.
Python isna() Function	Returns dataframe/series with bool values. Missing values gets mapped to True and non-missing gets mapped to False.
Pandas fillna() Function	This function is used to trim values at a specified input threshold.
Pandas clip() Function	Returns index information of the DataFrame.
Pandas columns() Function	Returns column names of the dataframe
Pandas sort_values() Function	This method sorts the data frame in ascending or descending order of passed Column

THESE TOPIC ALREADY COVERED IN UNIT1

What is python? How python in interpreted? What are the tools that helps to find the bugs or perform static analysis?
What is python IDE? Discuss some python IDE?
Programming cycle in python
Elements of python type conversion
Operator precedence and Boolean expression
How memory is managed in python? Explain PEP8

How can you create a python file that can be imported as library as well as run as a standalone script

- Module/ library are units that store code and data.
- It provided code reuse to python projects, and also useful in partitioning the system's namespace in self-contained packages
- They are self-contained because we can only access attributes of a module/library after importing it.
- Steps for importing python file as a library:

Step 1: create a file and name it test.py

Step 2: inside test.py create a function called display message().

For example

```
Def display_message(name):
return("welcome"+name +"hello")
```

How can you create a python file that can be imported as library as well as run as a standalone script

Step 3: now create another file display.py

Step 4: inside display .py import the module test.py file as shown below :

Import test

While importing we do not have to mention the test.py but just the name of the file

Step 5: then we can call the function display _message () from test.py inside displau.py, we need to make use of module _name, function.name

For example

```
Import test

Print(test.display_message(“Aditya))
```

Step6

When we execute display.py we will get the following output

```
Welcome Aditya  hello
```

FUNCTION

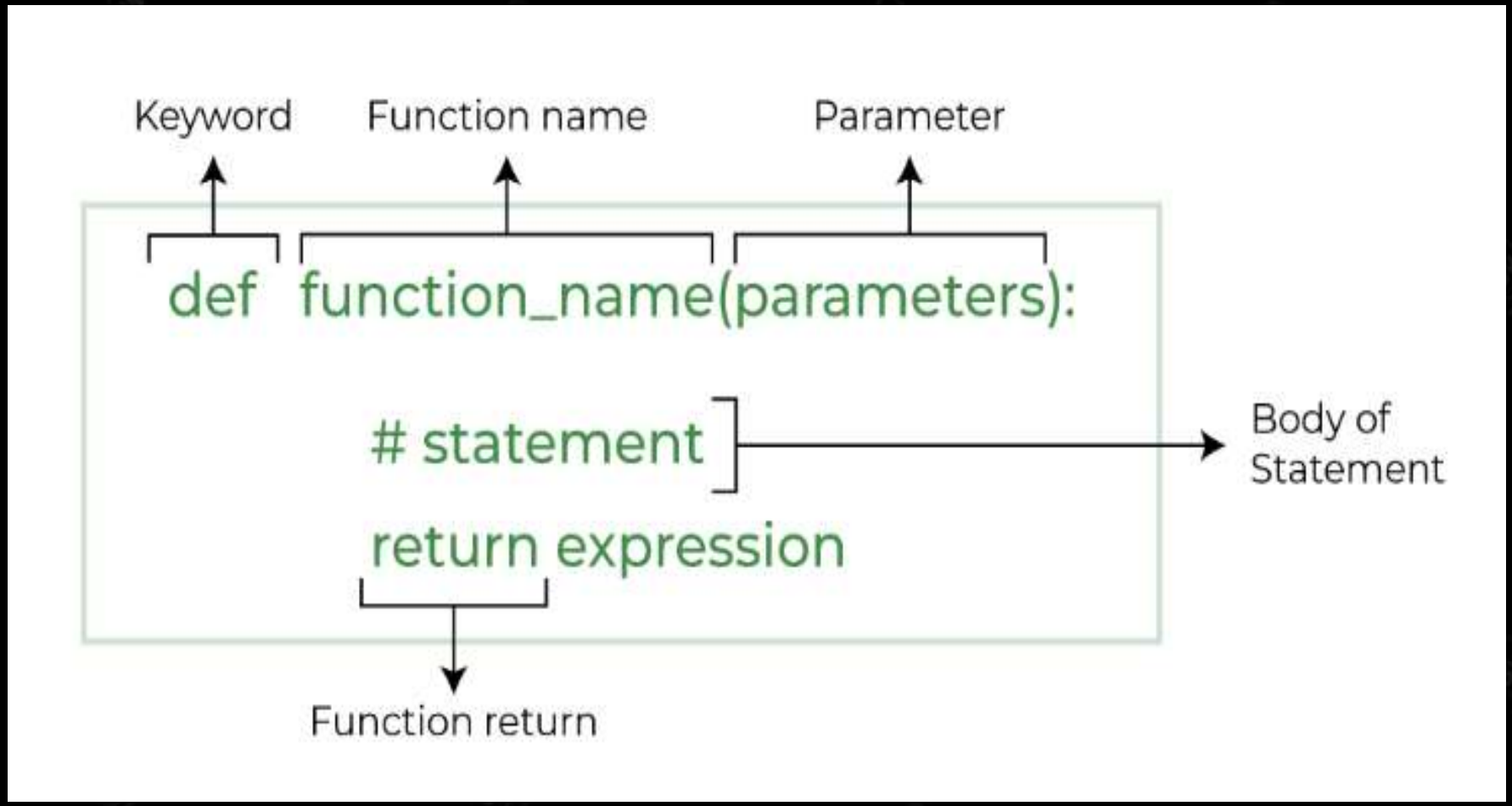
- **Python Functions is a block of statements that return the specific task. The idea is to put some commonly or repeatedly done tasks together and make a function so that instead of writing the same code again and again for different inputs, we can do the function calls to reuse code contained in it over and over again.**
- **Some Benefits of Using Functions**
- **Increase Code Readability**
- **Increase Code Reusability**

Types of Functions in Python

There are mainly two types of functions in Python

- Built-in library function: These are Standard functions in Python that are available to use.
- User-defined function: We can create our own functions based on our requirements.

Python Function Declaration



Python Function

- **Syntax of function definition :**
- def function name(arg 1, arg 2):

{

.....

}

Syntax of function call

Function_name(arg1,arg2)

Python Function

- **Keyword:** the keyword ‘def’ is used to define function header.
- **Function name:** we define the function name for identification or to uniquely identify the function.
- **(:)** a colon to mark the end of function header.
- **Arguments** these are the values passed to the function between parentheses
- **Body of the function**: the body process the argument to do something use ful
- An optional **return statement** to return the value from the function.
- **function call** To execute a function , we have to call it

Creating a Function in Python

- We can define a function in Python, using the def keyword. We can add any type of functionalities and properties to it as we require.

```
# A simple Python function
def fun():
    print("Welcome ")


# Driver code to call a function
fun()
```

```
Welcome

...Program finished with exit code 0
Press ENTER to exit console.
```

Python Function with Parameters

- If you have experience in C/C++ or Java then you must be thinking about the return type of the function and data type of arguments. That is possible in Python as well (specifically for Python 3.5 and above).
- Defining and calling a function with parameters
- def function_name(parameter: data_type) -> return_type:
- """Docstring"""
- # body of the function
- return expression.

Python Function with Parameters

```python
def add(num1: int, num2: int) -> int:
    """Add two numbers"""
    num3 = num1 + num2

    return num3

# Driver code
num1, num2 = 5, 15
ans = add(num1, num2)
print(f"The addition of {num1} and {num2} results {ans}.")

```

input

```
The addition of 5 and 15 results 20.


...Program finished with exit code 0
Press ENTER to exit console.
```

Types of Python Function Arguments

- Python supports various types of arguments that can be passed at the time of the function call. In Python, we have the following 4 types of function arguments.
- Default argument
- Keyword arguments (named arguments)
- Positional arguments
- Arbitrary arguments (variable-length arguments *args and **kwargs)

Default Arguments

- A default argument is a parameter that assumes a default value if a value is not provided in the function call for that argument

```
1  # Python program to demonstrate
2  # default arguments
3  def myFun(x, y=50):
4      print("x: ", x)
5      print("y: ", y)
6
7
8  # Driver code
9  myFun(10)
10 myFun(11,20)
11
12
```

input

```
:  10
:  50
:  11
:  20
```

Keyword Arguments

- The idea is to allow the caller to specify the argument name with values so that the caller does not need to remember the order of parameters.

```
1  # Python program to demonstrate Keyword Arguments
2  def student(firstname, lastname):
3      print(firstname, lastname)
4
5
6  # Keyword arguments
7  student(firstname='python', lastname='Practice')
8  student(lastname='Practice', firstname='python')
9
10
11
```

input

```
ython Practice
ython Practice


...Program finished with exit code 0
ress ENTER to exit console.
```

Keyword Arguments

- Keyword-only arguments mean whenever we pass the arguments(or value) by their parameter names at the time of calling the function in Python in which if you change the position of arguments then there will be no change in the output.
- Benefits of using Keyword arguments over positional arguments
- On using keyword arguments you will get the correct output because the order of argument doesn't matter provided the logic of your code is correct. But in the case of positional arguments, you will get more than one output on changing the order of the arguments..

Positional Arguments

- Position-only arguments mean whenever we pass the arguments in the order we have defined function parameters in which if you change the argument position then you may get the unexpected output. We should use positional Arguments whenever we know the order of argument to be passed.
- So now, we will call the function by using the position-only arguments in two ways, and In both cases, we will be getting different outputs from which one will be correct and another one will be incorrect

- We used the Position argument during the function call so that the first argument (or value) is assigned to name and the second argument (or value) is assigned to age. By changing the position, or if you forget the order of the positions, the values can be used in the wrong places, as shown in the Case-2 example below, where 27 is assigned to the name and Suraj is assigned to the age

Positional Arguments

```python
def nameAge(name, age):
    print("Hi, I am", name)
    print("My age is ", age)


# You will get correct output because
# argument is given in order
print("Case-1:")
nameAge("Suraj", 27)
# You will get incorrect output because
# argument is not in order
print("\nCase-2:")
nameAge(27, "Suraj")

```

input

```
Hi, I am Suraj
My age is  27

Case-2:
Hi, I am 27
My age is  Suraj
```

Arbitrary Keyword Arguments

- In Python Arbitrary Keyword Arguments, *args, and **kwargs can pass a variable number of arguments to a function using special symbols. There are two special symbols:
- *args in Python (Non-Keyword Arguments)
- **kwargs in Python (Keyword Arguments)

*args in Python (Non-Keyword Arguments)

- We can declare a variable length length argument with the*symbol
- The *args allows a function to accept any number of positional arguments
- The arguments are collected as a tuple within the function

*args in Python (Non-Keyword Arguments)

main.py

```
# beginning of the method
def sum(*args):
    resultfinal = 0
#beginning of for loop
    for num in args:
        resultfinal = resultfinal + num

    return resultfinal
#printing the values
print(sum(10, 20))                  # 30
print(sum(10, 20, 30))              # 60
print(sum(10, 20, 2))               # 32
```

```
30
60
32
```

***kwargs in Python (Keyword Arguments)

- The *kwargs parameter allows a function to accept any number of keyword arguments.
- The arguments are collected as a dictionary within the function

```python
# Let's write a Python program
# *kwargs for a variable number of keyword arguments

def myPrg(**kwargs):
    for key, value in kwargs.items():
        print (key, "==" , value)

# Driver code for kwargs in python
myPrg(first ='Hello', mid ='Welcome', last='Hello')

```

input

```
first == Hello
mid == Welcome
last == Hello


...Program finished with exit code 0
Press ENTER to exit console.
```

Calculator using function

main.py

```
# Python program for simple calculator

# Function to add two numbers
def add(num1, num2):
    return num1 + num2

# Function to subtract two numbers
def subtract(num1, num2):
    return num1 - num2

# Function to multiply two numbers
def multiply(num1, num2):
    return num1 * num2

# Function to divide two numbers
def divide(num1, num2):
    return num1 / num2

print("Please select operation -\n" \
        "1. Add\n" \
        "2. Subtract\n" \
        "3. Multiply\n" \
        "4. Divide\n")

```

Calculator using function

```
# Take input from the user
select = int(input("Select operations form 1, 2, 3, 4 :"))

number_1 = int(input("Enter first number: "))
number_2 = int(input("Enter second number: "))

if select == 1:
    print(number_1, "+", number_2, "=",
                    add(number_1, number_2))

elif select == 2:
    print(number_1, "-", number_2, "=",
                    subtract(number_1, number_2))

elif select == 3:
    print(number_1, "*", number_2, "=",
                    multiply(number_1, number_2))

elif select == 4:
    print(number_1, "/", number_2, "=",
                    divide(number_1, number_2))
else:
    print("Invalid input")
```

Calculator using function

```
Please select operation -
1. Add
2. Subtract
3. Multiply
4. Divide

Select operations form 1, 2, 3, 4 :2
Enter first number: 7
Enter second number: 8
7 - 8 = -1


...Program finished with exit code 0
Press ENTER to exit console.
```

Python Scope variable

- The location where we can find a variable and also access it if required is called the scope of a variable
- **Python Local variable**
- Local variables are those that are initialized within a function and are unique to that function. It cannot be accessed outside of the function

```
1
2  def f():
3
4      # local variable
5      s = "I love India"
6      print(s)
7
8
9  # Driver code
10 f()
11
```

```
I love India
```

Python Scope variable

- **Python Local variable**
- **If we will try to use this local variable outside the function**

```
1  def f():
2
3      # local variable
4      s = "I love python"
5      print("Inside Function:", s)
6
7  # Driver code
8  f()
9  print(s)
10
```

input

```
Inside Function: I love python
Traceback (most recent call last):
  File "/home/main.py", line 9, in <module>
    print(s)
NameError: name 's' is not defined


...Program finished with exit code 1
Press ENTER to exit console.
```

Python Scope variable

- **Python Global variables**
- Global variables are the ones that are defined and declared outside any function and are not specified to any function. They can be used by any part of the program.

```
1  # This function uses global variable s
2  def f():
3      print(s)
4
5
6  # Global scope
7  s = "I love python"
8  f()
9
10
```

input

```
I love python


...Program finished with exit code 0
Press ENTER to exit console.
```

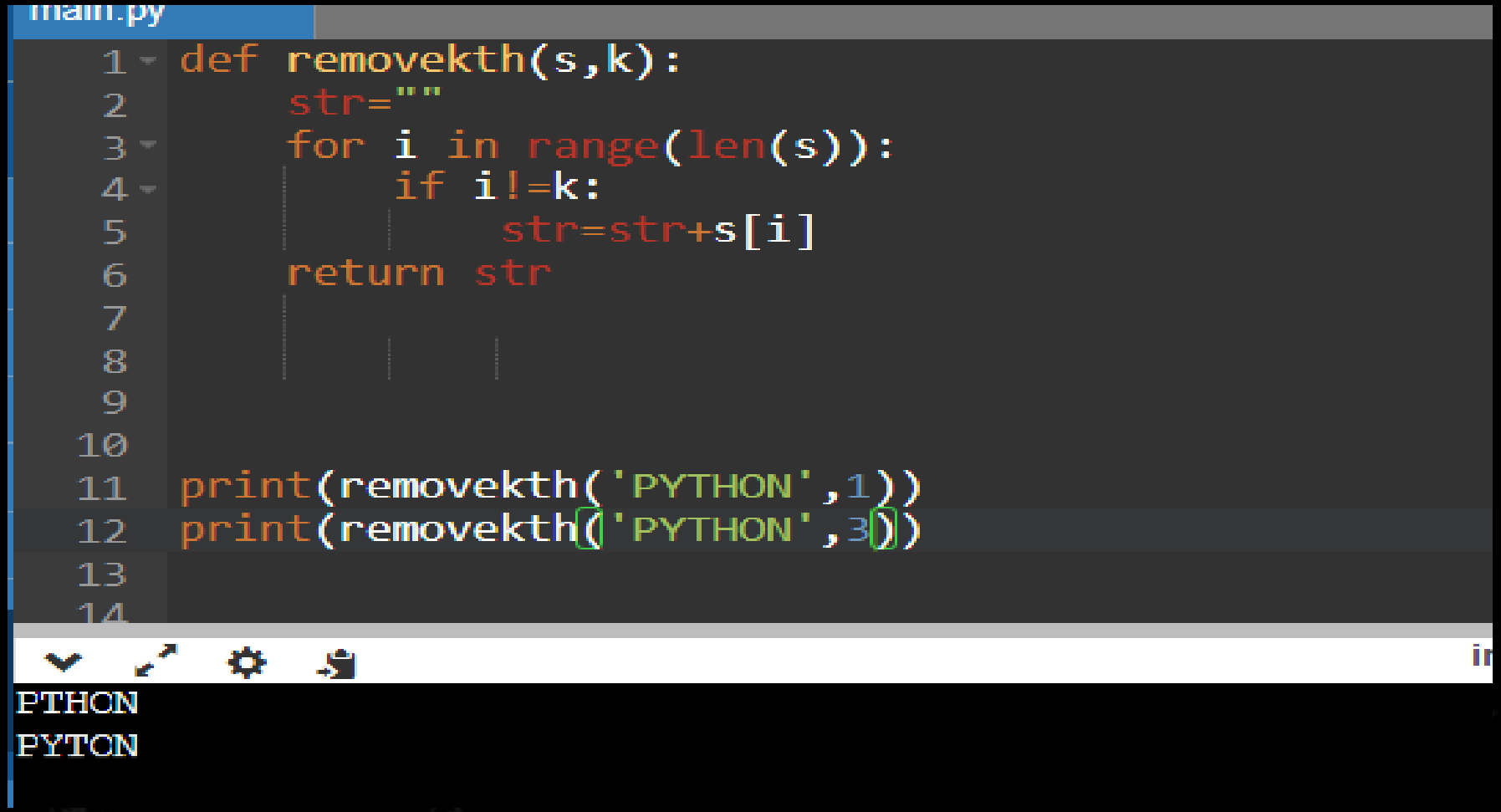
- Write a program factor(N) THAT RETURNS ALIST OF ALL POSITIVE DIVISOR OF N(N>=1)
- EXAMPLE FACTOR(6) RETURN [1,2,3,6]

```python
def factor(num):
    flist=[]
    if num>=1:
        for i in range(1,num+1):
            if  num%i==0:
                flist.append(i)
        return flist

print(factor(6))
print(factor(13))
```

```
[1, 2, 3, 6]
[1, 13]
```

GATEWAY CLASSES

- Write a python function remvekth(s,k) that take s as a input string and a n integer k>=0 and removes the character at a index k. if k is beyond the length of s , the whole os string is returned
- removekth(“PYTHON”,1) return PTHON
- removekth(“PYTHON”,3) return PYTON
- removekth(“PYTHON”,20) return PYTHON


main.py
def removekth(s,k):
str=""
for i in range(len(s)):
if i!=k:
str=str+s[i]
return str
print(removekth('PYTHON',1))
print(removekth('PYTHON',3))
PTHON
PYTON

- **Write a python code to find out occurrence of an elements in a list**

main.py

```
1  vowels=['a','a','e','i','a','o','u','u']
2  print("count of a",vowels.count('a'))
3  print("count of e",vowels.count('e'))
4  print("count of i",vowels.count('i'))
5  print("count of o",vowels.count('o'))
6  print("count of u",vowels.count('u'))
```

input

```
count of a 3
count of e 1
count of i 1
count of o 1
count of u 2


...Program finished with exit code 0
Press ENTER to exit console.
```

Ques Write a python function, search Many(s,x,k) that takes an argument a sequence s and integersx,k(k>0) the function returns true if there are at most k occurrences of x in s .otherwise it returns false

- searching([10,17,15,12],15,1) return true, searching([10,12,12,12],12,2) return false

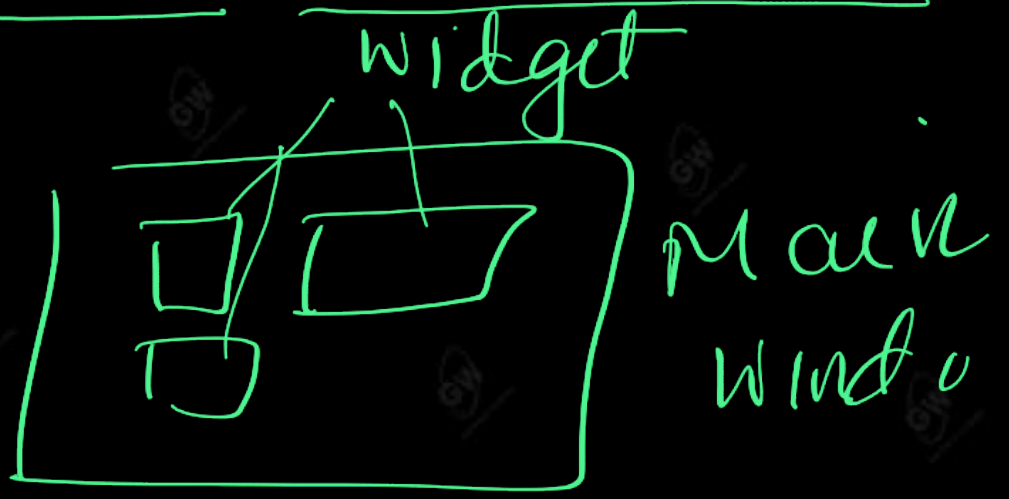
main.py

```
1  def searchMany(s,x,k):
2      count=0
3      for i in s:
4          if i==x:
5              count=count+1
6      if count==k:
7          return True
8      else:
9          return False
10
11 print(searchMany([10,17,15,12],15,1))
12 print(searchMany([10,17,15,12],17,3))
```

```
True
False
```

Python GUI – tkinter

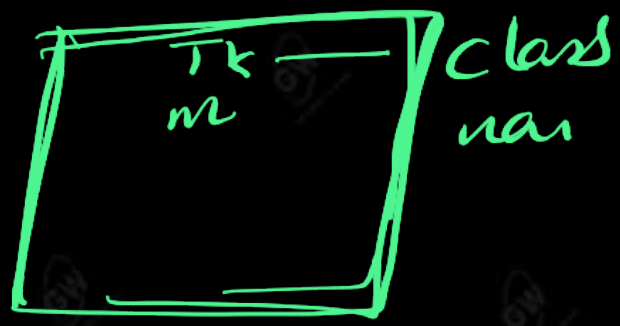
- Python offers multiple options for developing GUI (Graphical User Interface). Out of all the GUI methods, tkinter is the most commonly used method. It is a standard Python interface to the Tk GUI toolkit shipped with Python. Python tkinter is the fastest and easiest way to create GUI applications. Creating a GUI using tkinter is an easy task.
- To create a tkinter Python app:
- Importing the module – tkinter
- Create the main window (container)
- Add any number of widgets to the main window
- Apply the event Trigger on the widgets



Python GUI – tkinter

- Importing a tkinter is the same as importing any other module in the Python code. Note that the name of the module in Python 2.x is 'Tkinter' and in Python 3.x it is 'tkinter'.
- import tkinter
- There are two main methods used which the user needs to remember while creating the Python application with GUI.

Tk() → It help to display the root window & manages all other components of the tkinter application.



Python GUI – tkinter

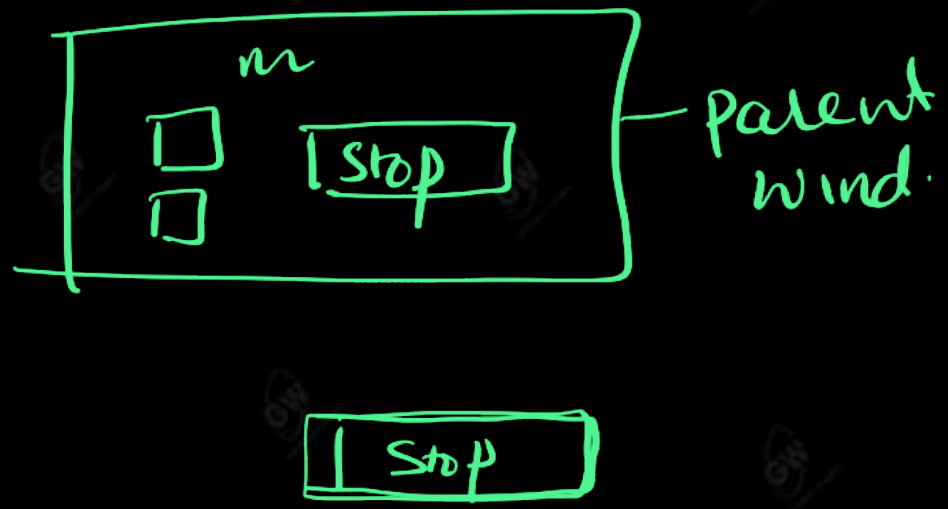
- There are two main methods used which the user needs to remember while creating the Python application with GUI.
- Tk(screenName=None, baseName=None, className='Tk', useTk=1): To create a main window, tkinter offers a method 'Tk(screenName=None, baseName=None, className='Tk', useTk=1)'. To change the name of the window, you can change the class name to the desired one. The basic code used to create the main window of the application is:
- m=tkinter.Tk() where m is the name of the main window object

- Main loop (): There is a method known by the name main loop () that is used when your application is ready to run. Main loop () is an infinite loop used to run the application, wait for an event to occur, and process the event as long as the window is not closed.
- m.mainloop()

Python GUI – tkinter

main.py

```
1  import tkinter
2  m = tkinter.Tk()
3  '''
4  widgets are added here
5  '''
6  m.mainloop()
7
8
```



Tkinter also offers access to the geometric configuration of the widgets which can organize the widgets in the parent windows. There are mainly three geometry manager classes class.

pack() method:It organizes the widgets in blocks before placing in the parent widget.

grid() method:It organizes the widgets in grid (table-like structure) before placing in the parent widget.

place() method:It organizes the widgets by placing them on specific positions directed by the programmer

Button:

- w=Button(master, option=value)
- master is the parameter used to represent the parent window. Several options are used to change the format of the Buttons. Several options can be passed as parameters separated by commas. Some of them are listed below
- **active background:** to set the background color when the button is under the cursor.
- **Active foreground:** to set the foreground color when the button is under the cursor.
- **bg:** to set the normal background color.
- **command:** to call a function.
- **font:** to set the font on the button label.
- **image:** to set the image on the button.
- **width:** to set the width of the button.
- **height:** to set the height of the button

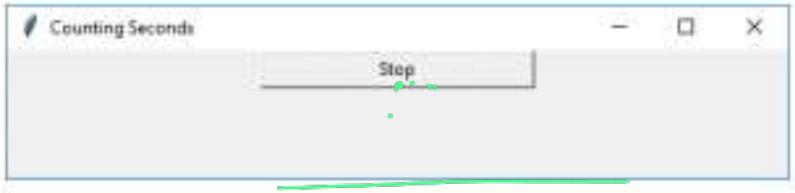
Button:

main.py

```
1  import tkinter as tk
2  r = tk.Tk()
3  r.title('Counting Seconds')
4  button = tk.Button(r, text='Stop', width=25, command=r.destroy)
5  button.pack()
6  r.mainloop()
7
```

r is the name of parent window

Output:



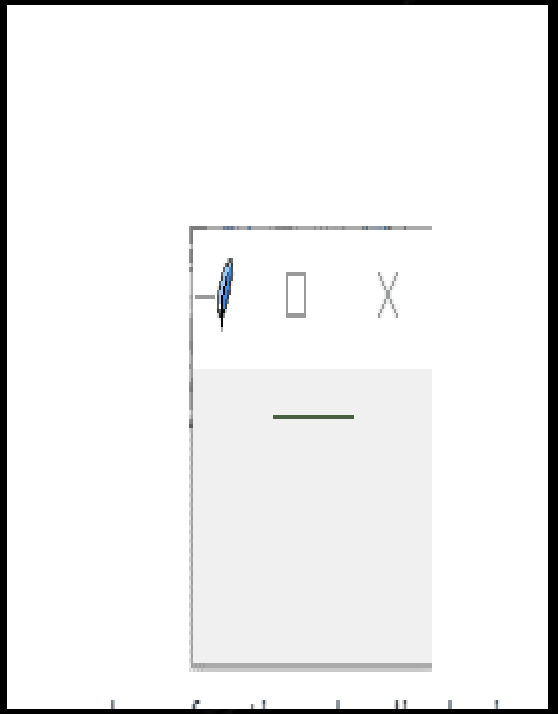
Canvas

- It is used to draw pictures and other complex layout like graphics, text, and widgets. The general syntax is:
- w = Canvas(master, option=value)
- master is the parameter used to represent the parent window.
- There are a number of options that are used to change the format of the widget. A number of options can be passed as parameters separated by commas. Some of them are listed below.
- **bd:** to set the border width in pixels.
- **bg:** to set the normal background color.
- **cursor:** to set the cursor used in the canvas.
- **Highlight color:** to set the color shown in the focus highlight.
- **width:** to set the width of the widget.
- **height**: to set the height of the wedge

Canvas

main.py

```
1  from tkinter import *
2  master = Tk()
3  w = Canvas(master, width=40, height=60)
4  w.pack()
5  canvas_height=20
6  canvas_width=200
7  y = int(canvas_height / 2)
8  w.create_line(0, y, canvas_width, y )
9  mainloop()
10
11
12
```



Check Button

- Select any number of options by displaying several options to a user as toggle buttons. The general syntax is:
- w = Check Button(master, option=value)
- Several options are used to change the format of this widget. Several options can be passed as parameters separated by commas. Some of them are listed below.
- Title: To set the title of the widget.
- **Active background:** to set the background color when widget is under the cursor.
- **Active foreground:** to set the foreground color when widget is under the cursor.
- **bg:** to set the normal background color.
- **command:** to call a function.
- **font:** to set the font on the button label.
- **image:** to set the image on the widget.

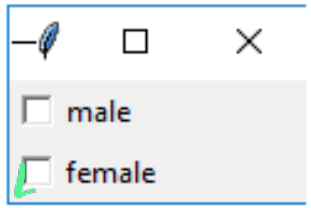
Check Button

Python

```
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()
```

W → the widget will stick to the left of the cell

Output:



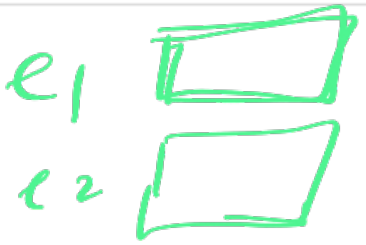
Entry

- It is used to input the single-line text entry from the user. For multi-line text input, a Text widget is used. The general syntax is:
- w=Entry(master, option=value)
- master is the parameter used to represent the parent window. Several options are used to change the format of the widget. Several options can be passed as parameters separated by commas. Some of them are listed below.
- **bd:** to set the border width in pixels.
- **bg:** to set the normal background color.
- **cursor:** to set the cursor used.
- **command:** to call a function.
- **Highlight color:** to set the color shown in the focus highlight.
- **width:** to set the width of the button.
- **height:** to set the height of the button..

Entry

Python

```
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()
```



Output:

GATEWAY CLASSES

GATEWAY CLASSES

AKTU Full Courses (Paid)

Download Gateway Classes Application

From Google Play store

All Subjects

Link in Description

Thank You