1. Introduction to Python Third Party Libraries(Packages)

This content is designed to introduce new developers to managing third-party packages using the Python package index and a tool called Pip. This course is part of a series of content designed to help you learn to program with the Python programming language

**Learning Objectives**

* The Python Package Index
* The tool used to manage packages called pip
* The tool used to create isolated Python environments named venv
* Considerations to be made before installing a third-party package

**Third-Party Libraries**

Python ships with a lovely standard library, which is a collection of packages and modules that are available on every machine that runs Python. However, you'll soon find that it doesn't contain everything you need. When this happens, you have two options:

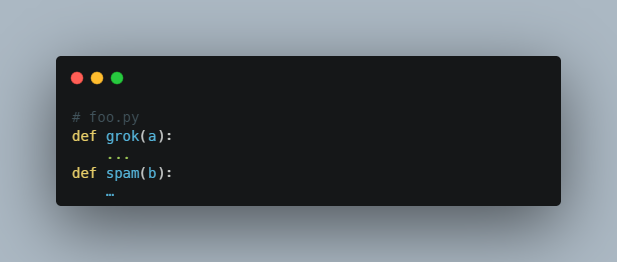
* Writing your own supporting package yourself
* Use someone’s else code

**Introduction to Packages(Before Third Parties)**

The subject of packaging is an ever-evolving, overly complex part of Python development. Rather than focus on specific tools, the main focus of this section is on some general code organization principles that will prove useful no matter what tools you later use to give code away or manage dependencies.

**Modules**

Any Python source file is a module.



An import statement loads and *executes* a module.



**Packages vs Modules**

For larger collections of code, it is common to organize modules into a package.

# From this

pcost.py

report.py

fileparse.py

# To this

porty/

    \_\_init\_\_.py

    pcost.py

    report.py

    Fileparse.py

You pick a name and make a top-level directory. porty in the example above (clearly picking this name is the most important first step).

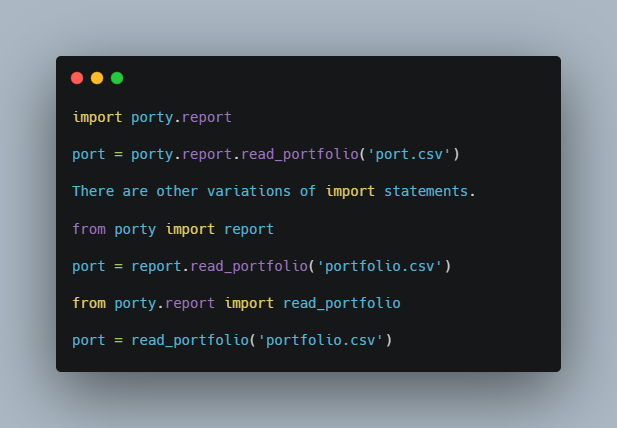
Add an \_\_init\_\_.py file to the directory. It may be empty.

Put your source files into the directory.

**Using a Package**

A package serves as a namespace for imports.

This means that there are now multilevel imports.



**Two Problems**

There are two main problems with this approach.

* imports between files in the same package break.
* Main scripts placed inside the package break.

So, basically everything breaks. But, other than that, it works

**Problem: Imports**

Imports between files in the same package *must now include the package name in the import*. Remember the structure.

porty/

    \_\_init\_\_.py

    pcost.py

    report.py

    fileparse.py

Modified import example.



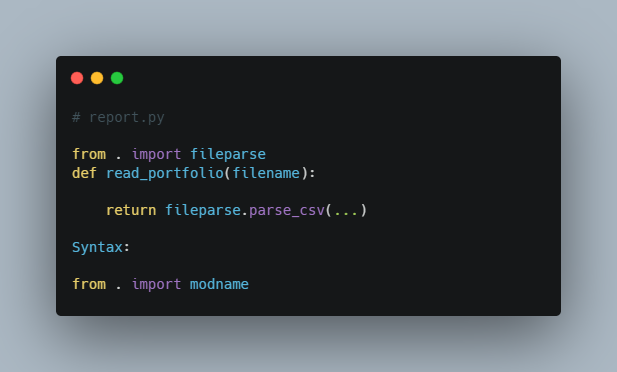
All imports are *absolute*, not relative.

# report.py

**import** fileparse  # BREAKS. fileparse not found

**Relative: Imports**

Instead of directly using the package name, you can use . to refer to the current package.



This makes it easy to rename the package.

**Problem: main scripts**

Running a package submodule as a main script breaks.

bash $ python porty/pcost.py # BREAKS

*Reason: You are running Python on a single file and Python doesn't see the rest of the package structure correctly (sys.path is wrong).*

All imports break. To fix this, you need to run your program in a different way, using the -m option.

bash $ python -m porty.pcost # WORKS

**\_\_init\_\_.py files**

The primary purpose of these files is to stitch modules together.

Example: consolidating functions



1. Introduction Third Party Libraries and Tools

Python has a large standard library, providing many useful modules right out of the box. But it will never have everything you need. Python also has a huge collection of third-party modules available for you to install.

**Pip**

Python comes with a package manager called pip. The name is a recursive acronym short for Pip Installs Packages. If you’re using a recent version of Python 3 (3.4 or higher), then you already have a pip3 command that will install packages.

Installing packages is as easy as:

pip install the\_package\_name

If you have more than one installation of Python on your machine, be sure you know which pip (or pip3) you are using. Pip will install packages into the Python installation you run it from.

**PyPI**

Python packages are available from the Python Package Index, or [PyPI](http://pypi.python.org/), pronounced Pie-Pee-Eye. Pip automatically knows to install from PyPI, but you can manually search or browse PyPI if you need to.

**Virtualenv**

Virtualenv is a third-party tool for isolating Python package installations from each other. If you are working on more than one project, it’s a good practice to keep their package requirements separate. Virtualenv lets you create a virtual Python environment for each project, and install packages separately for each.

**Popular third-party libraries**

PyPI has thousands of packages, it would be impossible to summarize them all. But there are a handful of go-to packages that many people use for common tasks:

* requests
* scrapy
* Twisted
* Pillow
* lxml
* PyYAML
* Django, Flask, Pyramid
* SQLAlchemy
* numpy, scipy, pandas
* pytest, tox, coverage, mock
* six
* Jinija2
* cryptography
* pylint, flake8, pep8
* pymongo, redis, MySQL-Python, psycopg2

**Additionally, here are a few more libraries that are more specific to a Big Data domain such as:**

**RedShift and S3:** Amazon services are used with their cloud services. S3 is a storage service and RedShift is a data warehousing service.

**BigQuery:** Developed by Google, BigQuery is a Cloud service library that is useful with RESTful APIs.

**PySpark:** This is an open-source framework used for large scale data processing and works with resilient distributed datasets.

**Kafka:** This is apublish-subscribe messaging system that receives logs in the form of packages and is stored in partitioned spaces.

**Pydoop:** Pydoop provides an interface between Hadoop and Python and support for handling its Hadoop distributed file systems.

**Resources:**

1. <https://www.netsolutions.com/insights/top-10-python-frameworks-for-web-development-in-2019/>
2. Python standard library

**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:

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. 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.
4. **Numpy:**The name “Numpy” stands for “Numerical Python”. It is the 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 TensorFlow 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 and high-level screen scraping. It can also be used for data mining and automated testing of data.
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 variously 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 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 really helpful for developers that are new in research fields.

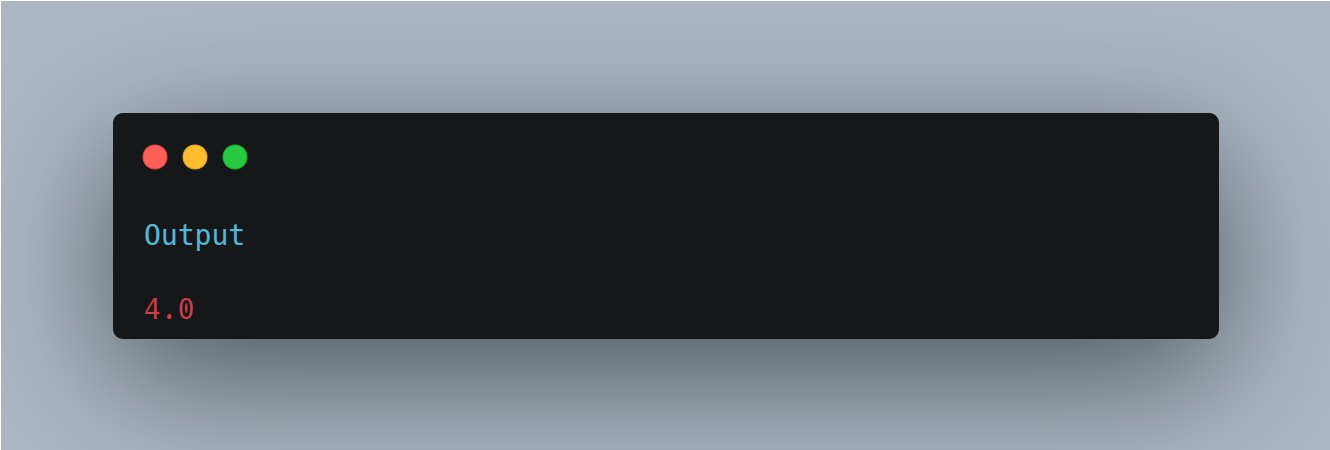
There are many more libraries in Python. We can use a suitable library for our purposes. Hence, Python libraries play a very crucial role and are very helpful to the developers.

**Use of Libraries in Python Program**

As we write large-size programs in Python, we want to maintain the code’s modularity. For the easy maintenance of the code, we split the code into different parts and we can use that code later ever we need it. In Python, *modules* play that part. Instead of using the same code in different programs and making the code complex, we define mostly used functions in modules and we can just simply import them in a program wherever there is a requirement. We don’t need to write that code but still, we can use its functionality by importing its module. Multiple interrelated modules are stored in a library. And whenever we need to use a module, we import it from its library. In Python, it’s a very simple job to do due to its easy syntax. We just need to use **import**.



**Output**



Here in the above code, we imported the math library and used one of its methods i.e. sqrt (square root) without writing the actual code to calculate the square root of a number. That’s how a library makes the programmers’ job easier. But here we needed only the sqrt method of math library, but we imported the whole library. Instead of this, we can also import specific items from a library module.

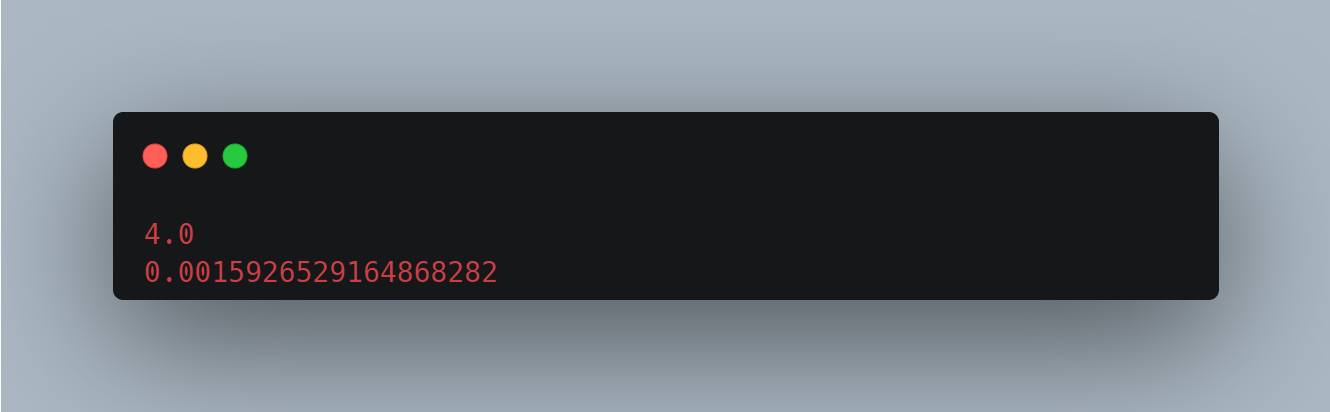
**Importing specific items from a library module**

As in the above code, we imported a complete library to use one of its methods. But we could have just imported “sqrt” from the math library. Python allows us to import specific items from a library.

Let’s look at an exemplar code :



**Output**



1. Python Modules

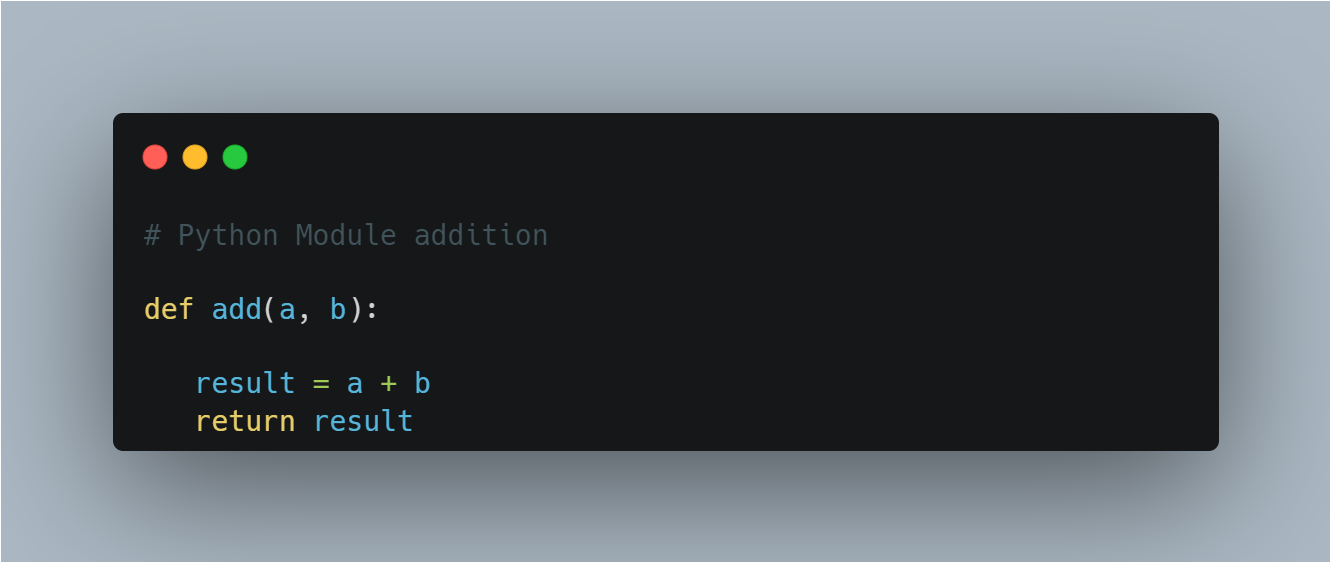
**Getting Started With Python Modules**

Python modules are reusable pieces of code that can be imported into a program to provide additional functionality. They are usually written in Python and can be either built-in modules or third-party modules that can be installed using tools like pip. Modules can contain functions, classes, variables, and constants that can be accessed by importing them into a Python script. Some popular built-in modules include math, DateTime, and random, while popular third-party modules include pandas, NumPy, and matplotlib.

As our program grows bigger, it may contain many lines of code. Instead of putting everything in a single file, we can use modules to separate codes in separate files as per their functionality. This makes our code organized and easier to maintain.

Module is a file that contains code to perform a specific task. A module may contain variables, functions, classes etc. Let's see an example,

Let us create a module. Type the following and save it as example.py.

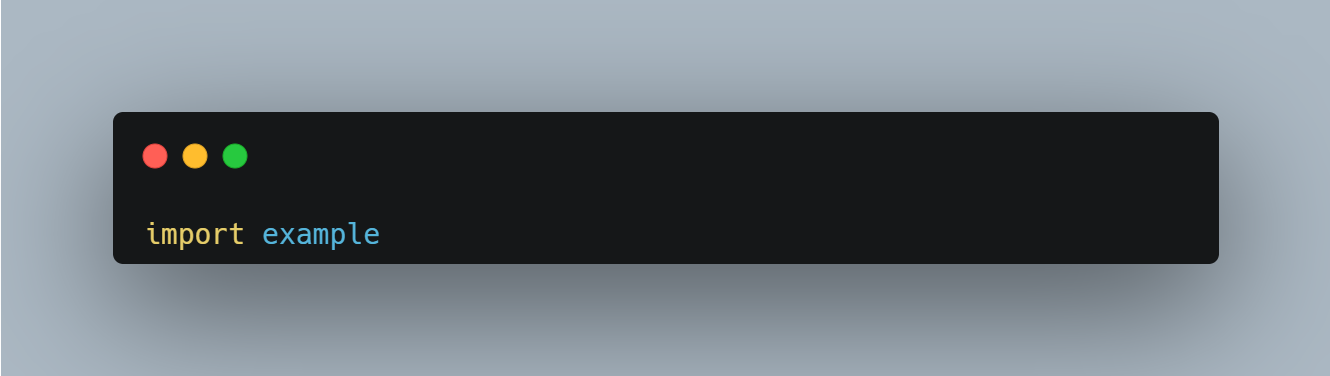


Here, we have defined a function add() inside a module named example. The function takes in two numbers and returns their sum.

**Import modules in Python**

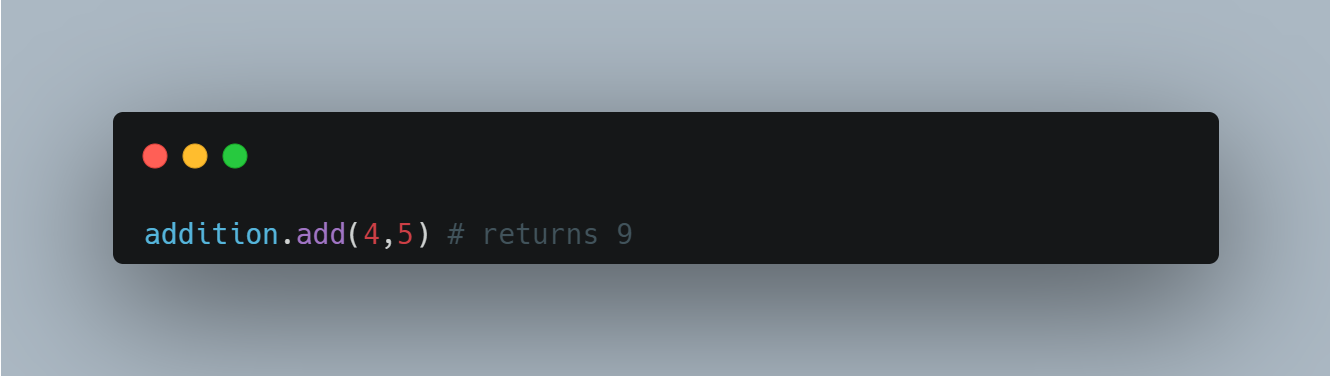
We can import the definitions inside a module to another module or the interactive interpreter in Python.

We use the import keyword to do this. To import our previously defined module example, we type the following in the Python prompt.



This does not import the names of the functions defined in example directly in the current symbol table. It only imports the module name example there.

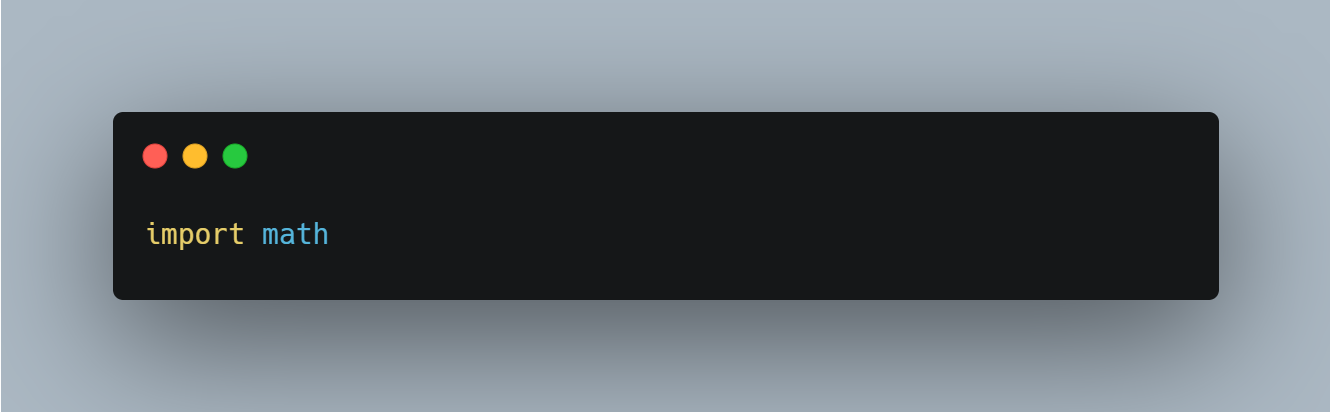
Using the module name we can access the function using the dot . operator. For example:



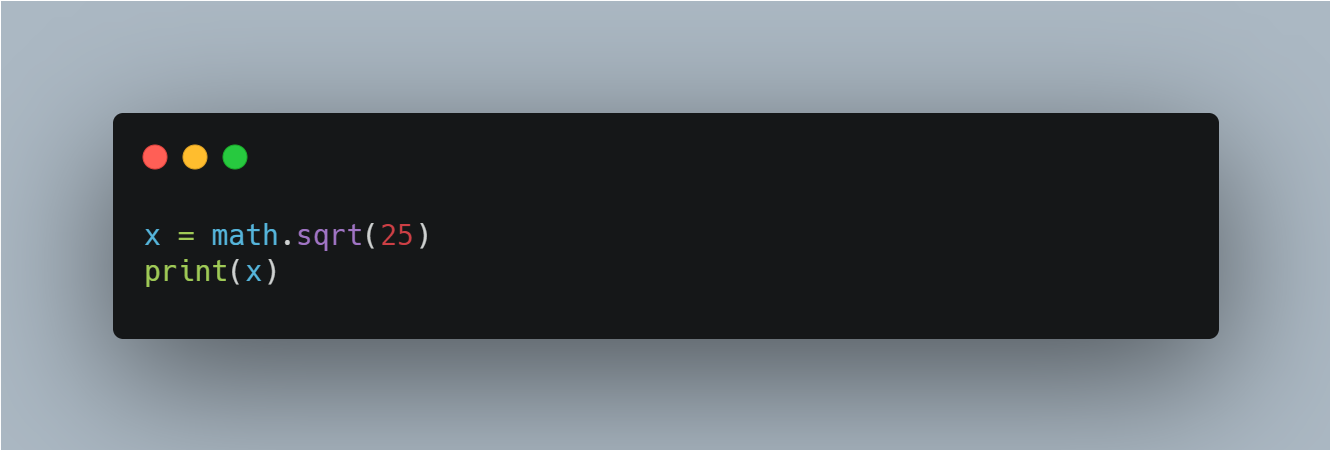
**Import Python Standard Library Modules**

The Python standard library contains well over **200** modules. We can import a module according to our needs.

To import a module from the Python Standard Library, you can use the import statement followed by the name of the module. Here's an example of how to import the math module:



You can then access the functions and constants defined in the math module using the . notation. For example, to use the sqrt function to find the square root of a number, you could write:



This would output 5.0, which is the square root of 25. There are many modules in the Python Standard Library, and you can find a list of them in the Python documentation.

**Python import with Renaming**

In Python, we can also import a module by renaming it. For example,

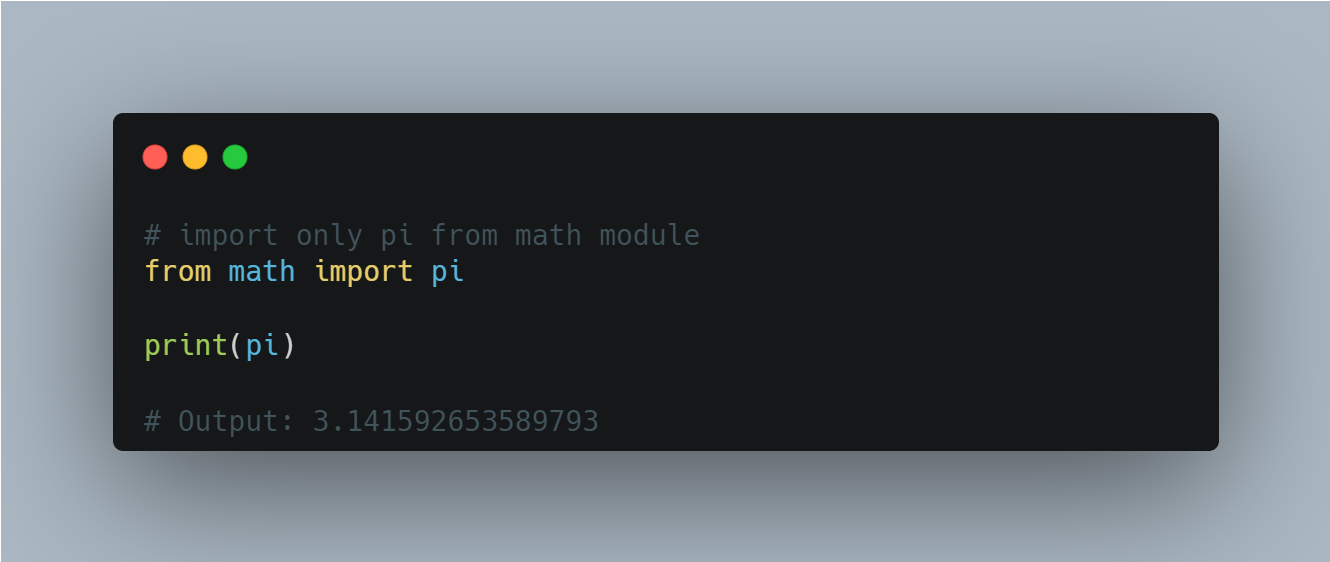


Here, We have renamed the math module as m. This can save us typing time in some cases.

Note that the name math is not recognized in our scope. Hence, math.pi is invalid, and m.pi is the correct implementation.

**Python from...import statement**

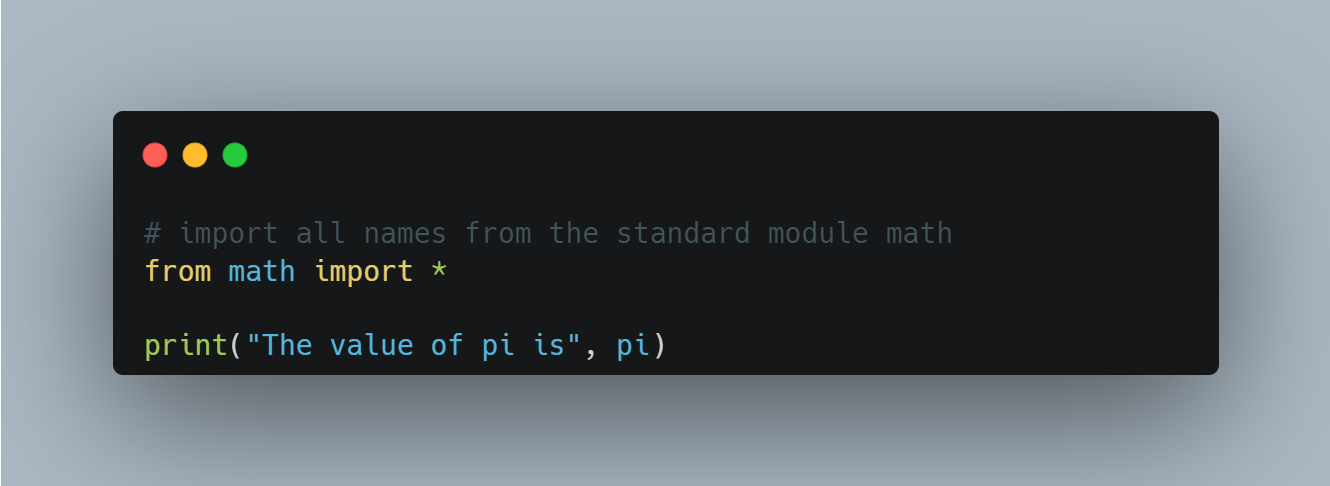
We can import specific names from a module without importing the module as a whole. For example,



Here, we imported only the pi attribute from the math module.

**Import all names**

In Python, we can import all names(definitions) from a module using the following construct:



Here, we have imported all the definitions from the math module. This includes all names visible in our scope except those beginning with an underscore(private definitions).

Importing everything with the asterisk (\*) symbol is not a good programming practice. This can lead to duplicate definitions for an identifier. It also hampers the readability of our code.

1. Understaning the use of python library NUMPY

**Here are some of the most commonly used Python libraries, along with a brief description of their purpose:**

**NumPy: Numerical computing library for working with arrays and matrices**

NumPy arrays are a fundamental data structure in the NumPy library, which is a powerful tool for numerical computing in Python. A NumPy array is similar to a Python list, but it has the following properties:

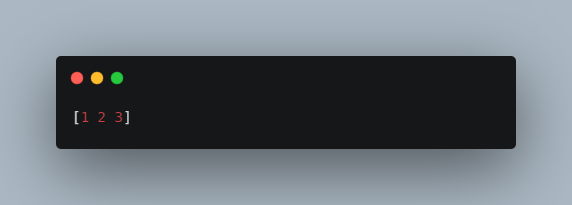
1. Homogeneous data type: All elements of a NumPy array must have the same data type, which is typically a numeric data type such as int or float.
2. Fixed size: The size of a NumPy array is fixed when it is created, and it cannot be changed without creating a new array.
3. Efficient: NumPy arrays are designed to be efficient for numerical operations, and they are implemented in C, which makes them faster than Python lists.

NumPy arrays can have one or more dimensions, and they are represented in memory as contiguous blocks of data. This makes it possible to perform mathematical operations on the entire array at once, rather than looping over individual elements

**Creating a NumPy array:**

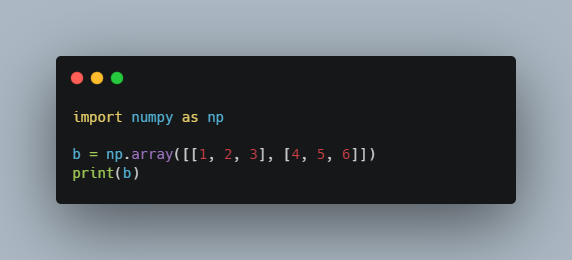


Output:

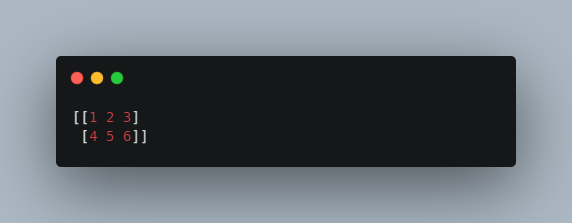


*In this example, we create a NumPy array with three elements and print it to the console. NumPy arrays can be used for a wide variety of numerical computing tasks, including linear algebra, statistics, and signal processing, among others.*

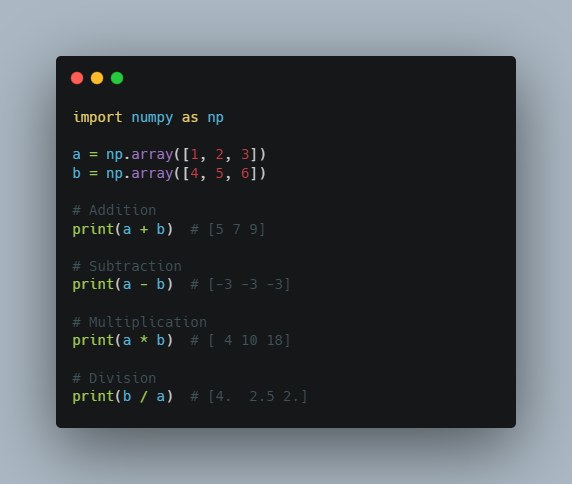
**Creating a 2D NumPy array:**



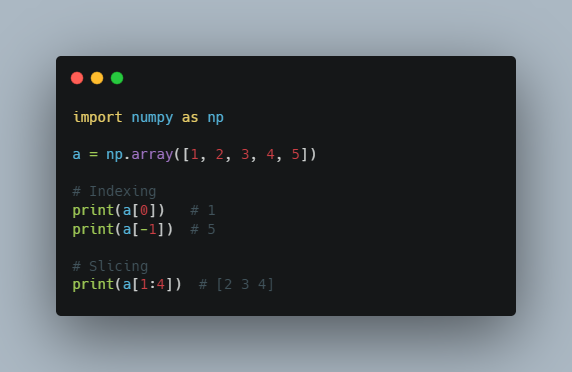
Output:



**Using NumPy functions to perform mathematical operations on arrays:**



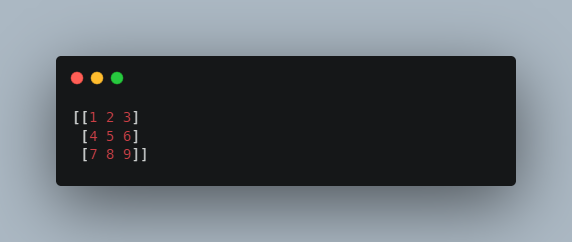
**Slicing and indexing NumPy arrays**:



**Reshaping NumPy arrays:**



Output:



These are just a few examples of the many things you can do with NumPy in Python. NumPy is a powerful library for scientific computing, and it is widely used in data analysis, machine learning, and other fields.

1. Understaning the use of python library PANDAS

**Data manipulation and analysis library for working with structured data.**

Pandas is a popular open-source library for data manipulation and analysis in Python. It provides data structures and functions for working with structured data, such as tables of numerical and textual data. Pandas is built on top of the NumPy library, which provides efficient array operations in Python.

Pandas provides two main data structures: Series and DataFrame. A Series is a one-dimensional array-like object that can hold any data type. It has an associated index, which labels each element in the Series. A DataFrame is a two-dimensional table-like structure that contains rows and columns of data. It is similar to a spreadsheet or a SQL table. Each column in a DataFrame is a Series.

Pandas provides a wide range of functions for reading and writing data to and from various file formats, including CSV, Excel, SQL databases, and more. It also provides functions for data cleaning, filtering, sorting, merging, and grouping.

Here are some examples of what you can do with Pandas:

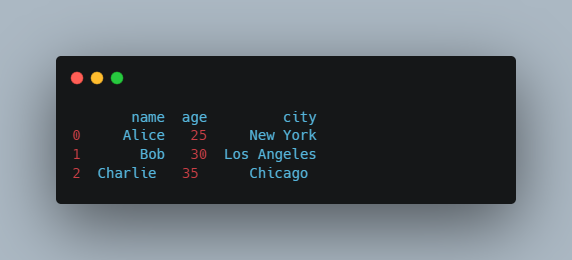
* Load data from a CSV file into a DataFrame
* Filter and sort data in a DataFrame based on certain criteria
* Compute summary statistics of the data, such as mean, median, and standard deviation
* Aggregate data using group-by operations
* Visualize data using built-in plotting functions

Pandas is widely used in data science, finance, and other fields where data analysis is an important part of the work. Its intuitive syntax and powerful functionality make it a popular choice for working with structured data in Python.

**Creating a Pandas DataFrame:**



Output:



**Reading data from a CSV file into a Pandas DataFrame:**



**Selecting data from a Pandas DataFrame:**



**Adding a new column to a Pandas DataFrame:**



Output:



**Grouping data in a Pandas DataFrame:**



Output:



These are just a few examples of the many things you can do with Pandas in Python. Pandas is a powerful library for data manipulation and analysis, and it is widely used in data science, finance, and other fields.

1. Understaning the use of python library Matplotlib

Matplotlib is a popular open-source data visualization library for Python. It provides a wide range of tools for creating static, animated, and interactive visualizations in Python.

Matplotlib is built on top of NumPy, a library for numerical computation in Python, and provides a high-level interface for creating common types of plots, such as line plots, scatter plots, bar charts, histograms, and more. It also provides tools for customizing the appearance of plots, including the colors, labels, titles, and axes.

One of the key strengths of Matplotlib is its flexibility. It provides a low-level interface for creating custom plots and visualizations, which can be used to create complex and sophisticated plots. Additionally, Matplotlib can be used in conjunction with other libraries for data analysis, such as Pandas, Seaborn, and Plotly, to create more advanced visualizations.

Matplotlib provides a wide range of functionality for creating different types of plots and visualizations. Some of the key features of Matplotlib include:

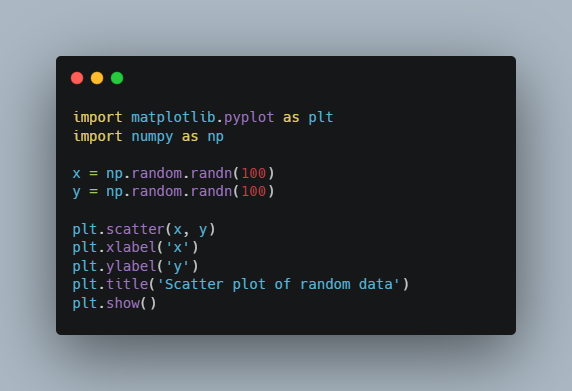
* Line plots: used to plot continuous data as a series of connected points.
* Scatter plots: used to plot data points as individual dots or markers.
* Bar charts: used to plot categorical data as bars.
* Histograms: used to plot the distribution of a set of continuous data.
* Heatmaps: used to plot data as a color-coded grid.
* Contour plots: used to plot data as a series of contours or lines.

Matplotlib is a powerful and flexible library for creating visualizations in Python. Its wide range of functionality, combined with its intuitive and easy-to-use interface, make it a popular choice for data scientists, researchers, and developers who need to create informative and visually appealing plots and visualizations.

**Line plot:**



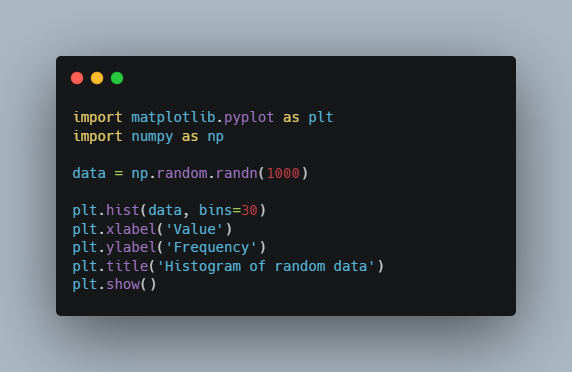
**Scatter plot:**



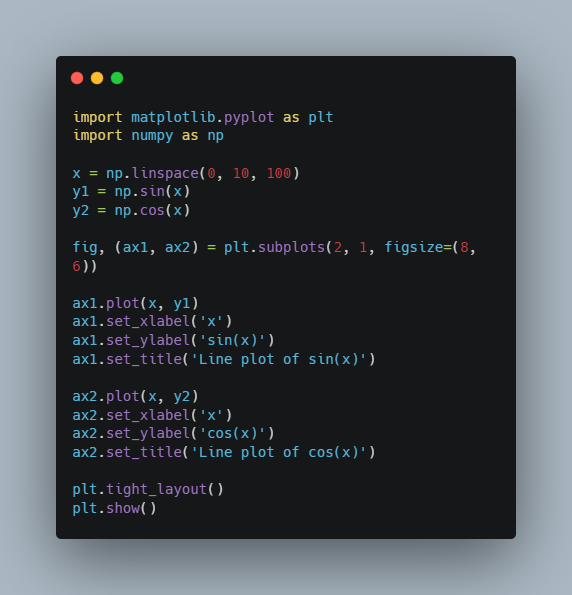
**Bar chart:**



**Histogram:**



**Subplots:**



1. Weekly Live Session Link

## **May 2024 Cohort Intro To Django Session**

<https://powerlearnproject-org.zoom.us/rec/share/KmQFMKz5Ic_0WLZW884qJ3F4mMKKalAkYPsrZmmnBGz_r9FT2LDuSidGWv5WIAsx.gXNUsYsDcs9E_Rdz>

## **Building a Weather App Recording**

<https://powerlearnproject-org.zoom.us/rec/share/ZY1ow65Ix4p-9eetMXzYxvyntTfh49O9_9LDLmPxYgs4WMMy9ZY-dmewqtExExI.LCSR1KCGHd3hqYNr>

## **Random API**

<https://powerlearnproject-org.zoom.us/rec/share/hYK86gdyD83B3psFh3VFFZfQl2ibOXViZ-BBMGvIes50weOXW-xr-IfT4_Gecwn0.yvWrRmC7ZRY2lRfU>

## Repository For reference

<https://github.com/Evans-mutuku/May-Cohort_Django.git>

<https://github.com/progskill/weatherApp.git>

1. Safaricom Hoot Class Recordings
2. [Morning Recording](https://powerlearnproject-org.zoom.us/rec/share/vkk95AXOodWChMTfw9fnjOnZHt5V4W5p_QwC3zAVRLMW8cdv-zdcq-np6OkEsE7e.mqoQLz4dI4zWe0g6?startTime=1736234029000)
3. Passcode: &=l0.=#8
4. [Afternoon Recording](https://powerlearnproject-org.zoom.us/rec/share/Pv2KoLG6o9y9lzEGZCbVYfQzULMCgc5SsHjFg4HFq7W3bFSwtueXUyejKphiGu_o.n-uGvaTwjmfNpXsb?startTime=1736248061000)
5. Passcode: =fR=9dKV
6. [Evening Recording](https://powerlearnproject-org.zoom.us/rec/share/DgjSjK6g-95Y-l0Tjx7Xts-9Ys3jAlpu39Rvm3o0xkuZUg9lCwDaUa0nFOh-0NMW.XpcGH94VsW4C8fky?startTime=1736265722000)
7. Passcode: k?sKEfB7
8. [Flask Recording](https://powerlearnproject-org.zoom.us/rec/share/8vAXJKtkvTLPKBonFPYkBSL7grDzcU-tcOv1IeoglHmbY1E3XecFFt9CfYqxxAKe.ZnXMvCzOoEZ4EoA6?startTime=1736853061000)
9. Passcode: v16Qw%\*z

### **Live Class Videos**

1. **Introduction to Django**

### **Linking MySQL Database in Django**

### **Extras**

1. [**Django MySQL - How to Set up Django to use a Database? - AskPython**](https://www.askpython.com/django/django-mysql)
2. [**Connect and Configure MySQL in Django Project**](https://studygyaan.com/django/how-to-use-mysql-database-with-django-project)
3. [**How to integrate Mysql database with Django? - GeeksforGeeks**](https://www.geeksforgeeks.org/how-to-integrate-mysql-database-with-django/)
4. [**How to use MySql with Django - For Beginners - DEV Community**](https://dev.to/sm0ke/how-to-use-mysql-with-django-for-beginners-2ni0)
5. [**Building a Task Manager: From Concept to Completion**](https://parsers.vc/news/241127-building-a-task-manager--from-concept-to/)
7. **GitHub Links**
8. [**https://github.com/chankjen/Django\_Blog.git**](https://github.com/chankjen/Django_Blog.git)
9. [**https://github.com/chankjen/Task\_Manager.git**](https://github.com/chankjen/Task_Manager.git)
11. **Deploying Django Project**