Modules

 A module is a piece of software that has a specific functionality. For example, when building a ping pong game, one module would be responsible for the game logic, and another module would be responsible for drawing the game on the screen. Each module is a different file, which can be edited separately.

Modularizing your code is one of the best ways to reuse your code. So let's look at an example, say you want to write a program, and you have to use a square function like more than 100x, remember this is just an example to show you how to create modules in python. I know we have been using Jupyter notebook and colab but as you move forward you will have to use some other editors, but for this simple program we are going to use a text editor.

1. First create a file and name it square_module.py
2. Open jupyter notebook and create a square function, Note that the notebook must be saved in the same folder as the python file. Here is the code to create a square function;

```
def Square (x):

    result = x * x;

    return result
```

1. Call the function and pass an argument to make sure the function works as expected. For example you can run this code in a new cell,

Square(5)

Note that the expected output is 25.

2. If the code produces the desired result then copy the function code and paste it in the square_module.py file, save and exit.

You have successfully created a python module, i.e the square_module.py file.

Importing a Module

Modules are imported in python using the import statement. In our example above we can import the square.py module by simple running the import square command as shown below;

import square_module

Here the command imports the entire module, but what we are interested in, is the function in the module. To import a function or class from a module we use the from module import statement, here is how we import the square function;

from square_module import square

square(8)

#prints 64

It is good to know how to create and import modules, but python has some modules that we can use to perform some tasks without having to create them ourselves.

Python has lots of modules that we can use to perform different tasks. Python built in Modules such as the math module has built in functions that can be used to perform math related tasks. For example to find the square root of a number, we don't need to write a function to do that we just need to import the sqrt function in the math module. Here is an example;
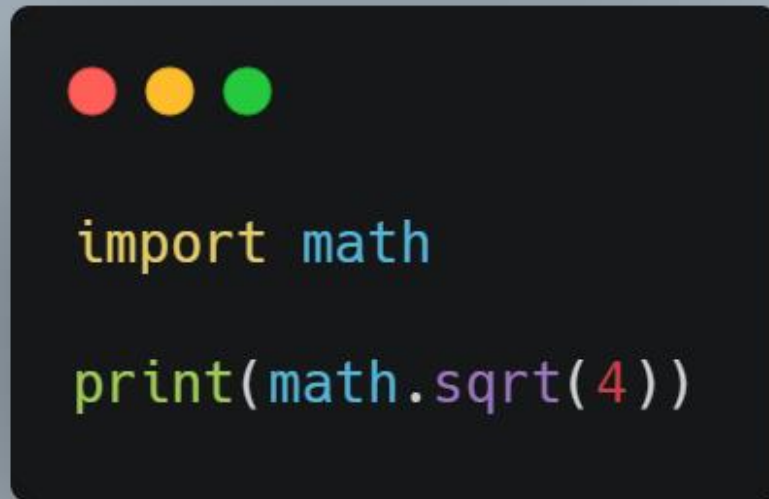
Before the example here are some thing to note, there are two ways to import the function

1. You can import the entire math module and then get the sqrt function from it, or
2. You can import just the sqrt function.

The second option is great for when you need just one function from the module, the first is great for when you need to use more than one function from a module.

Here is how you can do it in code

First option;

```
import math

print(math.sqrt(4))
```

we can also use another function like the power function

print(math.pow(2,2))

Notice the second example, we used the power function from the math module, the math.pow takes two arguments, the number and its power, so for this example we get 4 because 2 to the power of 2 is 4.
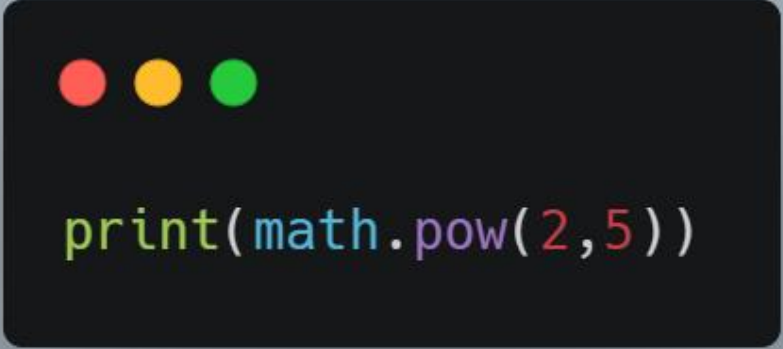
Second option:

from math import sqrt

print(sqrt(4))

Restart the runtime , if you are using colab go to runtime and choose restart runtime, and if you are running it locally you go to kernels and choose restart.

Run this in a new cell,

This will give an error because we did not import the entire math module.

There are other modules in python you will get to know them as we move forward.

Python Libraries

Python libraries are different from python modules, a python module is a file which contains python functions , global variables etc. It is nothing but a .py file which has python executable code or statement.

A Python library is a reusable chunk of code that you may want to include in your programs/ projects. Compared to languages like C++ or C, Python libraries do not pertain to any specific context in Python. Here, a 'library' loosely describes a collection of core modules. Essentially, then, a library is a collection of modules. A package is a library that can be installed using a package manager like pip

What is pip?

What is pip? pip is the standard package manager for Python. It allows you to install and manage additional packages that are not part of the Python standard library.

Important libraries in python for data science.

Here are libraries that will take you places in your journey with Python. These are also the Python libraries for Data Science.

1. **Pandas**

Like we've said before, Pandas is a must for data-science. It provides fast, expressive, and flexible data structures to easily (and intuitively) work with structured (tabular,multidimensional, potentially heterogeneous) and time-series data.

2. NumPy

It has advanced math functions and a rudimentary scientific computing package.

3. Matplotlib

Matplotlib helps with data analyzing, and is a numerical plotting library.

4. SciPy

SciPy has a number of user-friendly and efficient numerical routines. These include routines for optimization and numerical integration.

5. SciKit-Learn

This is an industry-standard for data science projects based in Python. Scikits is a group of packages in the SciPy Stack that were created for specific functionalities – for example, image processing. Scikit-learn uses the math operations of SciPy to expose a concise interface to the most common machine learning algorithms.

Data scientists use it for handling standard machine learning and data mining tasks such as clustering, regression, model selection, dimensionality reduction, and classification. Another advantage? It comes with quality documentation and offers high performance.

6. TensorFlow

TensorFlow is a popular Python framework for machine learning and deep learning, which was developed at Google Brain. It's the best tool for tasks like object identification, speech recognition, and many others. It helps in working with artificial neural networks that need to handle multiple data sets.

7. PyTorch

PyTorch is a framework that is perfect for data scientists who want to perform deep learning tasks easily. The tool allows performing tensor computations with GPU acceleration. It's also used for other tasks – for example, for creating dynamic computational graphs and calculating gradients automatically.

Installing Libraries in python

Now we have seen python libraries and what they can be used for, lets see how we can install and use them. Here we are going to look at two different ways to install libraries in python.

1. Installing libraries in colab

One of the reasons colab is widely used is because it comes with some libraries for data science installed. There are times when you need to instal some libraries too.

To import a library that's not in a Colab by default, you can use !pip install or !apt-get install.

Pandas is installed in colab by default but i will use this to show you how to install libraries.

!pip install pandas

2. **Installing libraries in Anaconda**

You can also use the graphical interface Anaconda Navigator to install conda packages with just a few clicks.

Open an Anaconda Prompt (terminal on Linux or macOS) and follow these instructions.

Enter the command:

conda install package-name

To install pandas you can enter the command;

Conda install pandas

Note that you can also install packages/ libraries on a jupyter notebook using the pip too,

It uses the same syntax as colab

!pip install pandas

Importing libraries

To import a library you use the import statement. For example to import the pandas library you run the command;

import pandas

In python you can import a library and rename it. What this means is that, when you have a library and its name is complex or too long for you to remember you can rename it to something simple or short.

Syntax:

Import package_name as new_name

 Here is an example,

import pandas as pd

You will notice as we go forward that this is the way most libraries will be imported.