GFMI

GLOBAL FINANCIAL MARKETS INSTITUTE

Article **2021**

Getting Started with Python: A Popular Tool for Finance Professionals

by Cara Marshall, Ph.D.

Getting Started with Python: A Popular Tool for Finance Professionals

Why Python?

Financial professionals are expected to be able to work with increasingly large datasets. Whether you are analyzing the bank's trade positions or working tirelessly to find exploitable anomalies in the plight for alpha, you must learn to efficiently clean up, sift through, and interpret vast amounts of data.

Python is quickly becoming the tool of choice for working with big data. It is fairly user-friendly and since Python is open-source, programmers have written and shared many libraries (also called modules or packages) that enable us to quickly develop sophisticated code without having to reinvent the wheel.

Python offers us the ability to import and work with Excel, CSV, or similar files. Then, from within Python, we can use libraries to perform statistical analysis or visualize data. Later, we can export files in various formats.

Python is already widely used in quantitative finance for processing and analyzing large financial datasets. Certain libraries, for example SciKit or PyBrain, offer machine learning algorithms which enable predictive and real time analytics and can be used for such tasks as automating fraud detection, analyzing consumer behavior, and algorithmic trading. Additionally, portfolio managers, analysts, and risk managers may find that Microsoft Excel crashes when working with CUSIP-level data. Python can more efficiently handle data aggregation while performing calculations such as weighted averages within asset classes or across desks.

In this article we teach you how to get started writing code to gather inputs from a user, and how to work with variables. You should then be able to write your own simple future value calculator in Python!

Getting Started in Python

To get started, you must install the latest version of Python. Since it is open-source, Python is free. The best way to install Python is through the Anaconda distribution, which can be used to download and install everything that you will need to get started.

Visit https://www.anaconda.com/products/individual to install the latest version for Windows, MacOS, or Linux.

Once installed, you will find that the Anaconda distribution comes with Spyder and Jupyter Notebook. Both programs can be used to write, test, and debug Python code.

Write a Simple Program

When you open Spyder, you will write code within the left window. The code can be run using the console in the right window. You can also run any Python file directly from the Anaconda Prompt.

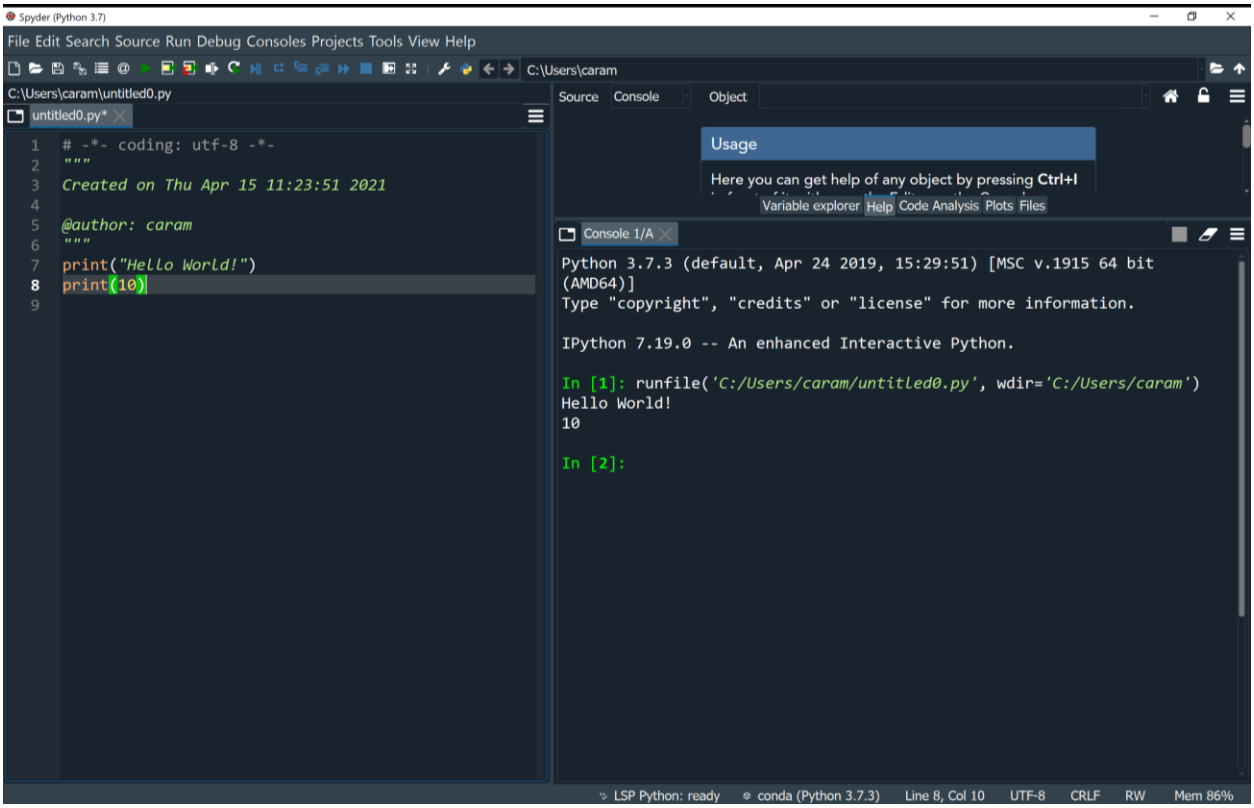
A simple Python command is print, which is used to output something. Much like Excel, you write your command and then list the data in parentheses. While numeric data does not require any characters around it, text should always go in quotes.

Try coding the following:

```
print("Hello World!")

print(10)
```

Then use the Run File (F5 key) play button on Spyder's toolbar, to watch your program run in the console on the right.



You can perform mathematical operations and output the results this way as well, for example:

```
print(10+5)
```

This should output 15.

Variables

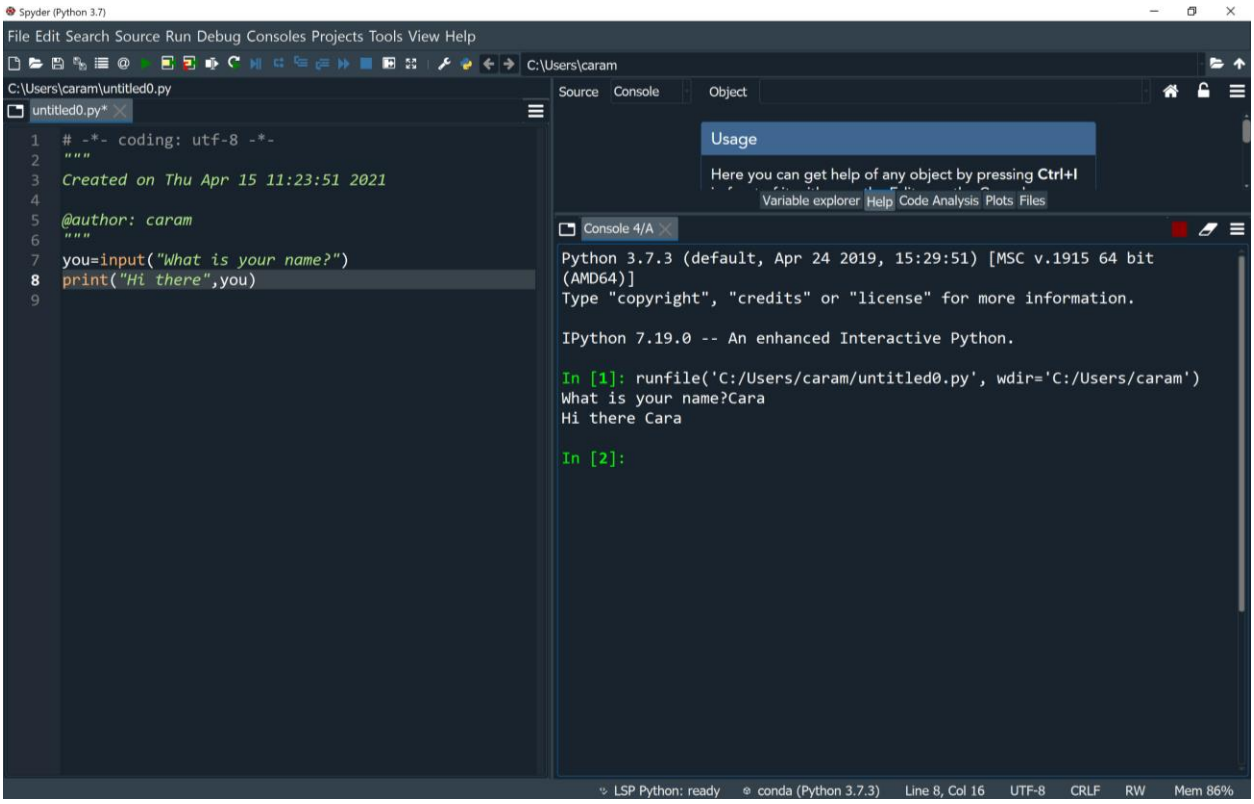
We can create variables to store information inside the program. The information stored in variables can be manipulated and outputted, using code. The type of information that will be stored can be specified (variable types include: integers, numbers with decimals, dates, strings of text, and more).

To create a variable called you and get information to store inside it, we can use the input command:

```
you=input("What is your name?")

print("Hi there",you)
```

When we run this program, we should see the question: "What is your name?" at which point, we must enter something in the console. Once entered, the program should respond with "Hi there Cara" (or whatever name you provided).



Create a Simple Future Value Calculator in Python

To give this some financial context, let's set up a simple future value calculator in Python using variables to make it interactive. Let's say we want the user to provide us with an amount to invest, an annual interest rate, and a number of years to invest. Assuming annual compounding and a simple interest rate, we can write up an equation that Python can use to solve:

$$\text{Future Value} = \text{Present Value} * (1 + \text{Annual Interest Rate})^{\text{Years}}$$

How can we translate this to Python? Begin by asking yourself how many variables will be needed? We should create one variable to hold each piece of information (each input):

```
presentvalue=input("How much do you want to invest?")

rate=input("What is your annual interest rate?")

years=input("How many years do you plan to invest?")
```

Next, we need to calculate the result based on the inputs that have been stored in each of the three variables. We can calculate and store the result in a fourth variable, called futurevalue (or whatever you would like to name it).

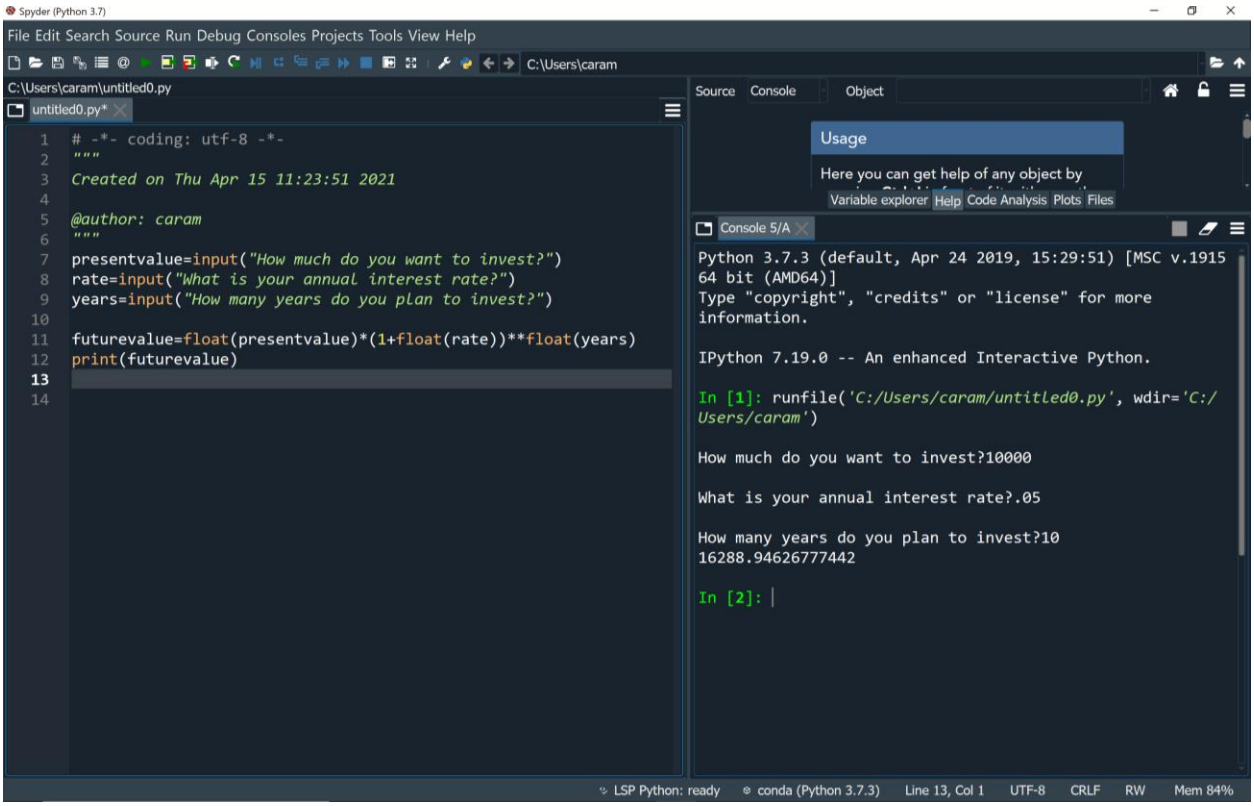
Importantly, the numeric operator for exponentiation is ** in Python. Also, the input command automatically treats the data that was input as if it is a string of text. We need to convert the inputs to numbers so that we can use the numbers in our future value calculation. To convert each input to a number, we can use the float command. Float is a variable type that is used for numbers that can contain decimals.

```
futurevalue=float(presentvalue)*(1+float(rate))**float(years)
```

Then we can output the result:

```
print(futurevalue)
```

This will output the future value on an investment based on the inputs you provided.



Other Types of Code

Our future value calculator is what programmers would refer to as sequential code, code that performs one action and then another. In the future value calculator, it is necessary to gather the inputs before calculating the result.

Conditional code is code that uses logic to do something if it meets a certain criteria. If it does not meet the criteria, then the code is designed to do something else. Another type of code is code that repeats. Repeat code uses looping to perform the same actions over and over again. Looping is very useful for churning through large datasets, such as working through the data row by row, or file by file.

Python Libraries

One of the greatest benefits of Python is the ability to import and immediately use the many free open-source libraries. Python libraries that are frequently used by financial professionals include Math, NumPy, SciPi, Pandas, and MatPlotLib. Math contains some basic mathematical functions (natural log, square root, etc.). NumPy and SciPi focus on numerical programming specializing in working with matrices. Pandas contains tools related to data analytics. MatPlotLib allows you to generate charts, histograms and the like, so that you can visualize your data.

There are also many very specialized libraries that will perform backtesting, price options, analyze markets, perform algorithmic trading, run technical analysis, pull in financial data from the web, and the like. These libraries can be imported and utilized directly in your Python code. Once you have imported the library, you have access to many new and powerful commands that can immediately be put to use on your datasets.

It is these vast and powerful libraries which makes Python a particularly popular choice for financial professionals.

About the Author: Cara Marshall, Ph.D.

Cara Marshall has the unique background of the academics (graduate and undergraduate levels) combined with a financial markets career. With over fifteen years of classroom experience, she knows the art of designing, developing and delivering an engaging learning program, keeping the students actively participating in the process. She also has financial industry experience, starting as an intern with Merrill Lynch, and working her way up to an independent consultant to some of the largest investment banks, federal agencies and hedge funds and other banks for the past sixteen years. In addition, she is a published author, writing journal articles and a book that focus on her areas of special interest, such as financial engineering; derivatives; pricing and risk management, amongst others. This dual-experience background – of the markets as well as the academics – enables Cara to apply hands-on practical experience and in-depth knowledge to beginners as well as to more advanced students.

Cara's specific topics of expertise include:

- Microsoft Excel for Financial Professionals (Introductory/Intermediate/Advanced)
- Programming in Visual Basic for Applications (VBA) in Excel
- Programming with Python
- Financial Modeling

23 Maytime Ct
Jericho, NY 11753
+1 516 935 0923
www.GFMI.com