Paola Celio | Pietro Corsi | Sergio Lins

PYTHON BASICS

Dipartimento di Matematica e Fisica - Dipartimento di Scienze

- Python is easy to learn, highly readable, and simple to use. It has a clean and english-like syntax which requires less coding and let the programmer focus on the business logic rather than thinking about the nitty-gritty of the language.
- We've organized this course to provide depth, detail, and degree. Even a beginner can refer to it and learn Python with least efforts, without investing a lot of time.
- The below sections cover Python history, features, domains, why to learn Python, how to install and run Python on platforms like Windows, Linux, and Mac OS X.

History of Python

- It was a Dutch programmer, **Guido Van Rossum**, who wrote Python as a hobby programming project back in the late 1980s. Since then, it has grown to become one of the most polished languages of the computing world. [...]

Silent Features of Python

- ☛ **Code Quality**

- Python code is highly readable, which makes it more reusable and maintainable. It has broad support for advanced software engineering paradigms such as object-oriented (OO) and functional programming.

- ☛ **Developer Productivity**

- Python has a clean and elegant coding style. It uses an english-like syntax and is dynamically-typed. So, you never declare a variable. A simple assignment binds a name to an object of any type. Python code is significantly smaller than the equivalent C++/Java code. It implies there is less to type, limited to debug, and fewer to maintain. Unlike compiled languages, Python programs don't need compiling and linking, which further boosts the developer's productivity.

- ☛ **Code Portability**

- Since Python is an interpreted language, so the interpreter has to manage the task of portability. Also, Python's interpreter is smart enough to execute your program on different platforms to produce the same output. So, you never need to change a line in your code.

- ☛ **Built-in and External Libraries**

- Python packages a large no. of the prebuilt and portable set of libraries. You can load them as and when needed to use the desired functionality.

- ☛ **Component Integration**

- Some applications require interaction across different components to support the end to end workflows. Such component could be a Python script while others be a program written in languages like Java/C++ or any other technology.

- Python has several ways to support the cross-application communication. It allows mechanisms like loading of C and C++ libraries or vice-versa, integration with Java and DotNET components, communication using COM/Silverlight, and interfacing with USB devices over serial ports. It can even exchange data over networks using protocols like SOAP, XML-RPC, and CORBA.

- ☛ **Free to Use, Modify and Redistribute**

- Python is an OSS (http://ossfoundation.us/). You are free to use it, make amends in the source code, and redistribute, even for commercial interests. It is because of such openness that Python has garnered a vast community base that is continually growing and adding value.

- ☛ **Object-oriented from the Core**

- Python primarily follows the object-oriented programming (OOP) design. OOP provides an intuitive way of structuring your code, and a solid understanding of the concepts behind it can let you make the most out of your coding. With OOP, it is easy to visualize the complex problem into smaller flows by defining objects and how they correlate. And then, we can form the actual logic to make the program work.

Python Programming Domains

- ☛ **Web Application Development**

- Python has the lion's share in the field of web development. Many employers look for full-stack programmers who know Python. And you can become one of them by learning frameworks (WAF) like Django, Flask, CherryPy, and Bottle, which give extensive support for web development. All of these are developed using Python. These frameworks deliver essential features to simplify tasks related to content management, accessing a backend database, and handling network protocols like HTTP, SMTP, XML-RPC, FTP, and POP. Some of the known online products created in Python are Plone (Content Management System), Zope application server, Quixote web framework, and ERP5, an open-source enterprise solution used in the aerospace field.

- ☛ **Data Science and Machine Learning**

- Data science, analytics, and machine learning are evolving at a prolific rate. Many companies are now seeking machine learning engineers who can filter through the stacks of data and support them in making the right business decisions. And Python has now become the first language for everyone entering into the ML domain. It provides all kinds of tools and models to programmers for tasks like web scraping, data collection, cleaning, and algorithms. So, if you are apt in Python programming, then it is the right time to enter into this domain.

- ☛ **Scientific and Numeric Computing**

- Python has become the obvious choice for working in Scientific and Numeric Applications. And there are multiple reasons for this advancement. First and foremost is that Python is a free and open-source language. It even allows you to modify its source code and redistribute.

- Next, it is getting the support of a growing number of specialized modules like NumPy, SciPy, Pandas, matplotlib, and IPython. All of these are available for free and provide a reasonable alternative to paid products like Matlab. Hence, it is one of the reasons for it to become more dominant in the field of Scientific and Numeric.

- Hence, Python is becoming a leader in this field. The focus of Python language is to bring more productivity and increase readability.

☛ GUI Programming

- Python has some inherent qualities like clean and straightforward coding syntax as well as dynamic typing support. These work as the catalyst while developing complex GUI and image processing applications.

- Python's clean syntax and tremendous support of many GUI libraries (like wxWidgets, pyqt, or pyside) made Programmers deliver graphics software like Inkscape, Scribus, Paint Shop Pro, and GIMP.

- In addition to the 2D imaging solutions given above, Python is even propelling many 3D animation software like 3ds Max, Blender, Cinema 4D, Houdini, and Maya. These applications integrate with Python for automation to speed up their workflows and eliminate the need for doing them manually.

☛ Software Prototyping

- Python has many qualities that make it a natural choice for prototyping. The first is being an open-source programming language, a massive no. of users follow and contribute to its development. Further, the lightness, versatility, scalability, and flexibility of refactoring code in Python speed up the development process from the initial prototype.

- Hence, Python gives you an easy-to-use interface to create prototypes. For example, with Pygame (a multimedia library), you can prototype a game in different forms, test, and tailor it to match your requirements. Finally, you can take clues from the selected prototype and develop it using languages like C++/Java.

- ☛ **Professional Training**

- Python is indeed the right programming language for teaching and training purposes. It can be a stepping stone for beginners to enter into vocational training. They can even cover overlapping areas like Data Analytics and Machine Learning.

- Hence, there is a massive demand for professional trainers who can teach both basic and advanced level Python programming. You can impart training offline in a classroom or use tools like Skype or hangout to do it online.

Why Should You Learn Python Programming?

- Writing Python code is so much fun that you won't feel it like a routine programming task. Here are a bunch of compelling reasons for you to learn Python and read this Python tutorial.

- ☛ **Nonrestrictive Programming Syntax**

- Python is the language that even a non-programmer can understand. It is a highly readable, problem-oriented language that lets you focus on writing the logic instead of getting into the syntax nitty-gritty.

- ☛ **No Explicit Declaration**

- In Python, don't use type specifiers for declaring variables. You can do so without mentioning any data types in Python. Also, there is no need to use any separator like a semicolon to mark the end of a statement.

- In Python, indentation replaces brackets to group a block of instructions. And you can use either tabs or spaces to indent the code.

- However, Python enforces some rules (PEP 8), such as to use four spaces per indentation level. So, there are many such facets in Python which make learning simpler for beginners.

- ☛ **State of the Art OOP Support**

- Object-Oriented Programming (OOP) comes built into the Python language. It lays down a model that derives a solution by creating objects, defining relationships, and binding data. However, the procedural programming methodology takes on a top-down approach and solves one problem at a time while splitting it into smaller ones. On the other hand, OOP is a bottom-up problem-solving technique that seeks a blueprint of the solution at the onset and leaves implementation for the later.

- Using classes, building inheritance, and constructing objects are keys to developing yourself as a professional programmer. And you can quickly master these concepts in Python and produce better quality projects, quicker than in other languages.

- Python lets you minimize the size of code you write and accomplishes more.

- ☛ **Powerful Debugging**

- If you have a product in dev and QA logged a defect, then debugging is what you should be doing. However, if a customer has reported an issue, then debugging gets far more critical. Here also, Python leads the way by providing tools like pdb, pudb, and PyDebug that makes debugging experience more comfortable.

- For example, the pudb tool can quickly guide programmers to dig through the code and nail down the problem.

- It is a fact that most languages are working to improve their debugging tools. But with Python, they are only getting better.

How to Get Python Running on Your System?

- Python programming is platform-independent. It means that you write the code once, and it should run on all supported platforms. Here, you can find the steps to install Python on three major platforms: Mac OS X, Linux, and Windows. If you already have Python installed, then you can proceed to the next section of this Python tutorial.

- **Install Python on Mac OS X**

- ☛ The latest version of **Mac OS X** is **10.13 (High Sierra),** which has **Python 2.7** pre-installed. Beginners can start to use **Python 2** and learn to program. However, professional programmers should upgrade to **Python 3.7** (and like you can find on our download section we'll use Python 3.x).

- ☛ For installing Python on Mac, go to the **Download Python for Mac OS X** page, select the desired package, and click to download. Next, launch the downloaded package, follow the steps, and finish the installation.

- ☛ If you are a command-line geek and love to use a console for routine system tasks, then run the following commands to install Python.

- Firstly, open the terminal and install the **HomeBrew** package manager for **Mac OS X**.

- $ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

- Enter the below command to check the Brew installation status.

- $ brew --version

- Now, use the Brew package manager to install Python 3.x.

- $ brew install python3

- ☛ Installing Python with Brew also installs the **PIP tool (pip3)**. It is a quick update manager to check out the available modules in the Python repository.

- **Run Python on Mac OS X**

- ☛ Since now you have Python installed, it's time to start coding. But you would need an IDE for professional development. It would turn you more efficient and productive. We recommend the community version of **PyCharm** and **Eclipse with PyDev**. Both of these are free and full of features. However, you can read our reviews of the world's **best Python IDEs** and choose one for yourself.

- ☛ However, you can also begin with IDLE, the default IDE that comes with Python. Or even use the Python console to run your first few lines of code. In Python, you can do a lot with a single line of code. For example, you can write a function which checks for odd no, call it with input and print the outcome, but all of that in one single line. Please see the code below.

- print((lambda isOdd: isOdd(3))(lambda x: x % 2 != 0))

- If you type the above line on a **Python CLI**, then it would just take an enter key to execute. However, if you are using **IDLE** to write this code in a script, then press **F5** to run it. In either way, this simple one line of code would print **"True"** in its output.

- **Install Python on Linux (Ubuntu)**

- ☛ **Ubuntu 16.04** has both **Python 2** and **Python 3** installed by default. However, you can double-check if they are up to date or not using the **apt-get** commands.

- $ sudo apt-get update

- $ sudo apt-get -y upgrade

- ☛ Once the above process completes, verify the Python version with the following command.

- $ python3 -V
- **# Output.**
- Python 3.6.1
- ☛ To manage available modules and libraries for Python, let's **install PIP**.
- $ sudo apt-get install -y python3-pip
- ☛ The **PIP tool** downloads new and updates existing packages you may like to use during the development. Here is the command to install the Python packages.
- $ pip3 install <package_to_install>
- **# For example.**
- $ pip3 install numpy
- ☛ Now is the time to check out a few more libs and dev tools that could be pretty useful later. Run the below command to add them to your Python installation.
- $ sudo apt-get install build-essential libssl-dev libffi-dev python-dev
- ☛ You can also set up an isolated space on your system for Python projects. For this, install the **venv** module. It is a component of the standard Python 3 library.
- $ sudo apt-get install -y python3-venv

- **Run Python on Ubuntu**

- ☛ You now first need to create environments to run Python. Go ahead and execute the following commands.
- $ mkdir environments
- $ cd environments
- $ python3 -m venv test_env

- ☛ Before you can use the **<test_env>**, you have to activate it first. The following command will do the activation for you.
- $ source test_env/bin/activate
- Your prompt will now a little different than the standard one.
- (test_env) techbeamers@techbeamers:~/environments$
- This prefix indicates that the environment **test_env** is currently active. And you can create programs to use the environment's settings and packages.
- ☛ We have our virtual env set up, let's write a simple **"Hello, World!"** script. To do this, open up a command-line text editor such as **vi** and create a new file.
- (test_env) techbeamers@techbeamers:~/environments$ vi world.py
- The vi editor will open the text file in the terminal. Write the code given below. Press **":wq"** to save and exit from the editor.
- print("Hello, World!")
- ☛ Once you exit out of vi and return to the shell, let's run the script.
- (test_env) techbeamers@techbeamers:~/environments$ python3 hello.py
- **# Output**
- Hello, World!
- To log out of the environment, type the command **"deactivate,"** and it'll return to your original directory.

- ## Install Python on Windows

- ☛ For **installing Python on Windows**, go to the [Download Python for Windows]() page, select the desired package, and click to download. We recommend Python

3.7 as it is one of the most stable packages. Next, launch the downloaded package, follow the steps, and finish the installation.

- ☛ During installation, select the option **"Install for all users"** and use the destination directory **(C:\Python37)** as default.
- ☛ Next, open the **"Start"** menu and type **"cmd"** into the search box. Right-click on the **"cmd.exe"** link and choose to run as an administrator.
- ☛ Change directory to **"C:\Python37"** and run the following command to set Python on the system's path.
- setx PATH "%cd%;%path%;"
- pause
- The above command **(setx)** will set the Python path for all future instances of the **cmd.exe**, but not for the current one. So, you'll need to reopen the command window to use Python.

Run Python on Windows

- ☛ Now, you are ready to write your first Python program on Windows. As we stated above, you must use a professional IDE for better development. However, here, we'll tell you to use **IDLE** that comes as the default editor for Python.
- ☛ From the **"Start"** menu, open **"All Programs"** and select **"Python 3.7"**. Click on the **"IDLE (Python GUI)"** link to launch the editor.
- ☛ Once the **IDLE** window appears, press **CTRL+N** to create a new file. Then, name it **"world.py"** while saving using the **CTRL+S**.
- ☛ Place the following code in **"world.py."**
- print("Hello, World!")
- Save the file and go to **[Run >> Run Module]** or press **F5** to execute your first Python script.

Create Your First Program in Python

- Usually, it is the **“Hello, World!”** program which every language recommends as a starting point to learn Programming. It is a simple program that prints the **“Hello, World!”** message on the standard output.
- It obviously would be very trivial if we write it in Python, just one statement as the **print(“Hello, World!”)**.