Tutorial

Beginner's Guide to Python

Introduction

Python is a simple, powerful, and highly adaptable programming language. Here's an official description from Python.org:

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

Let's define some of those terms to understand Python better:

- **High-level data structures** mean you don't have to worry about low-level details like managing your hard drive's memory.
- **Object-oriented programming** refers to Python's ability to combine data and functionality since it is built around objects rather than procedures. This is especially useful for computer-aided design programs or mapping software
- **Python's syntax** is one of the easiest to learn and is used by everyone from school children to Artificial Intelligence programmers and NASA.

This guide introduces you to Python and some foundational programming concepts to enable you to write simple Python commands. Here, you will:

- Learn about the different Python interfaces
- Select an interface you are comfortable with
- Understand the basics of Python syntax and the ecosystem
- Troubleshoot common problems and interpret error messages
- Equip yourself with sufficient skills to explore Python on your own

In the Next Steps section, you can explore practice exercises and ideas for using Python with your learners.

What Is a Program?

A program is a sequence of instructions for a computer to read and interpret. An algorithm is a sequence of instructions that performs a meaningful task. Gaming software, Word documents are both examples of programs.

As well as Python, popular programming languages include C, C++, Python, Javascript, and Ruby.

Here's an introduction to Python to get you started.

Python in Schools

Python is a simple language, which makes it ideal for learning in schools. Unlike other programming languages, reading Python is similar to reading English. This makes Python easy to understand and lets you focus on the task at hand rather than the syntax. Technology is a significant aspect of our world nowadays, so learning how to code prepares learners with critical professional skills for the future. Furthermore, building programs develop learners' creative and critical thinking skills. In this respect, Python's popularity and adaptability make it especially useful.

Learning mindset

Learning how to code is like learning a new language. While Python is relatively easy, learning any new language can be challenging. Don't let this intimidate you. Unleash your curiosity and enthusiasm, and you will learn better. Start slowly, pay attention, and practice, practice, practice!

Python Interfaces

The versatility of the Python language means developers use it on several platforms. Here, we will look at these platforms, their key features and interface, and learn how to install Python on them.

The four platforms we will look at are IDLE, Jupyter notebook, Google Colab, and Pydroid:

	IDLE	Jupyter Notebooks	Google Colab	Pydroid
What is it?	IDLE is Python's own interface or IDE, which comes pre-installed.	Jupyter Notebooks is a web-based application you can use to create and share documents that contain live code, equations, visualizations, and text.	Colaboratory, or "Colab" for short, allows you to write and execute Python in your browser. Similar to Jupyter, it requires no setup and runs entirely in the cloud.	Pydroid is an Android app that users can download to learn and run Python commands on their phones.

Key features	It's a simple interface, is excellent for new users and learners.	Jupyter is useful for scientific research and data analysis. It also supports Markdown, an HTML language.	Notebooks are saved to your Google Drive. It is a great interface for learners and researchers alike	Designed for educational use, you can use all the features of Python - on your phone.
Video Introduction	Watch intro	Watch intro	Watch intro	Watch intro
Used on	PC, Mac, Linux, others	Web-browser application, all platforms	Web-browser application	Android phone
Download Link	Download Python	Download Anaconda	Open Colab	Download Pydroid

Have a look at each of these platforms and see which works best for you, although we would suggest Jupyter Notebooks when using this guide as it will focus on the features and function of Python using Jupyter Notebooks. However, the basic commands and syntax are the same; you can use this guide with other interfaces.

Installing Python

Python is free and open-source software, which means you won't pay expensive license fees and won't cost you anything to download and install. You can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new programs.

Python programs work on multiple platforms, including GNU/Linux, Windows, Mac, and Notebooks. If you avoid using system-dependent features, you can even get the programs you create to run on any of these platforms. We will guide you through the installation of Python with IDLE and Jupyter Notebooks.

Disclaimer: Depending on the version you are using, there may be slight differences in the design or features shown in this guide.

Installing Python (IDLE)

The IDLE interface downloads automatically when you install Python on your computer. While Python regularly releases updates, each version release is stable and is supported for at least five years, so you don't have to worry about downloading the most stable version. Once you've installed Python, you might want to create a shortcut on your Desktop to access the program quickly.

You can download Python here.

This beginner's guide is relevant for version 3.9.1 and above.

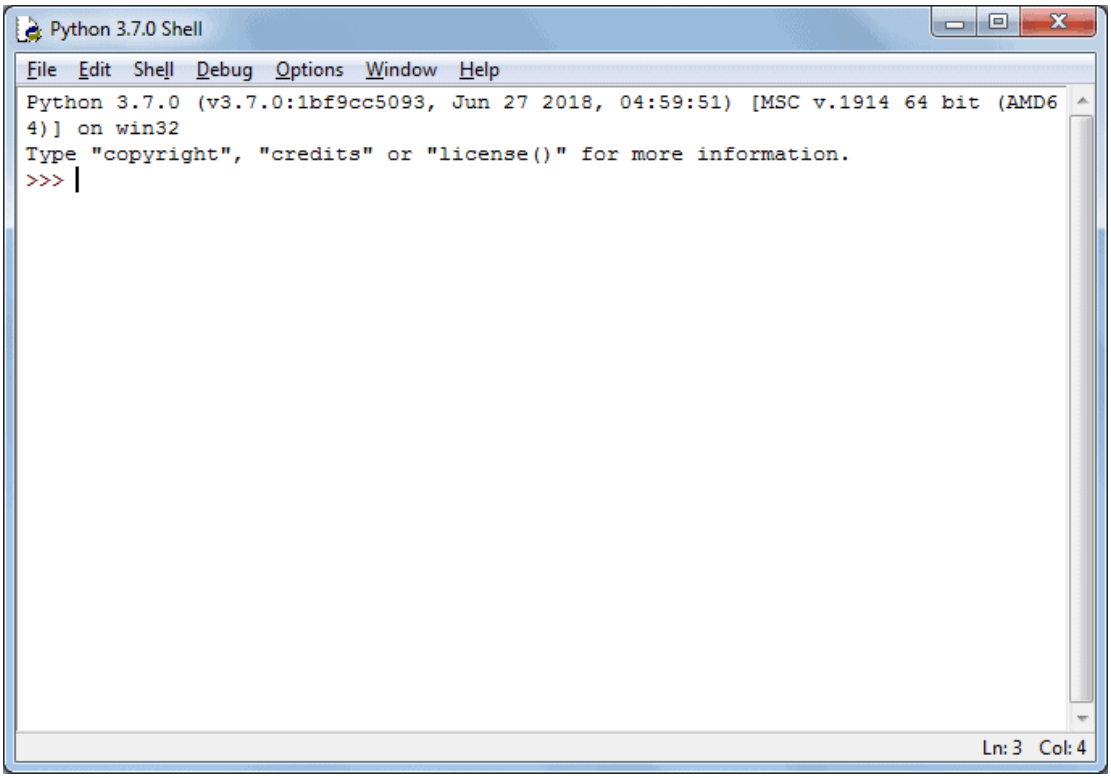
Installing Jupyter Notebooks

Jupyter was created for use with notebooks, so you can easily use this on your Chromebook or other notebooks. If you're using it for the first time, we suggest installing Anaconda, which includes Python, Jupyter Notebooks, and other data science packages, and is easy to install. If you need more help to install Anaconda, please watch this introductory video.

Navigating User Interface

Let's begin by looking at the user interfaces of IDLE, Jupyter, Colab, and Pydroid.

IDLE



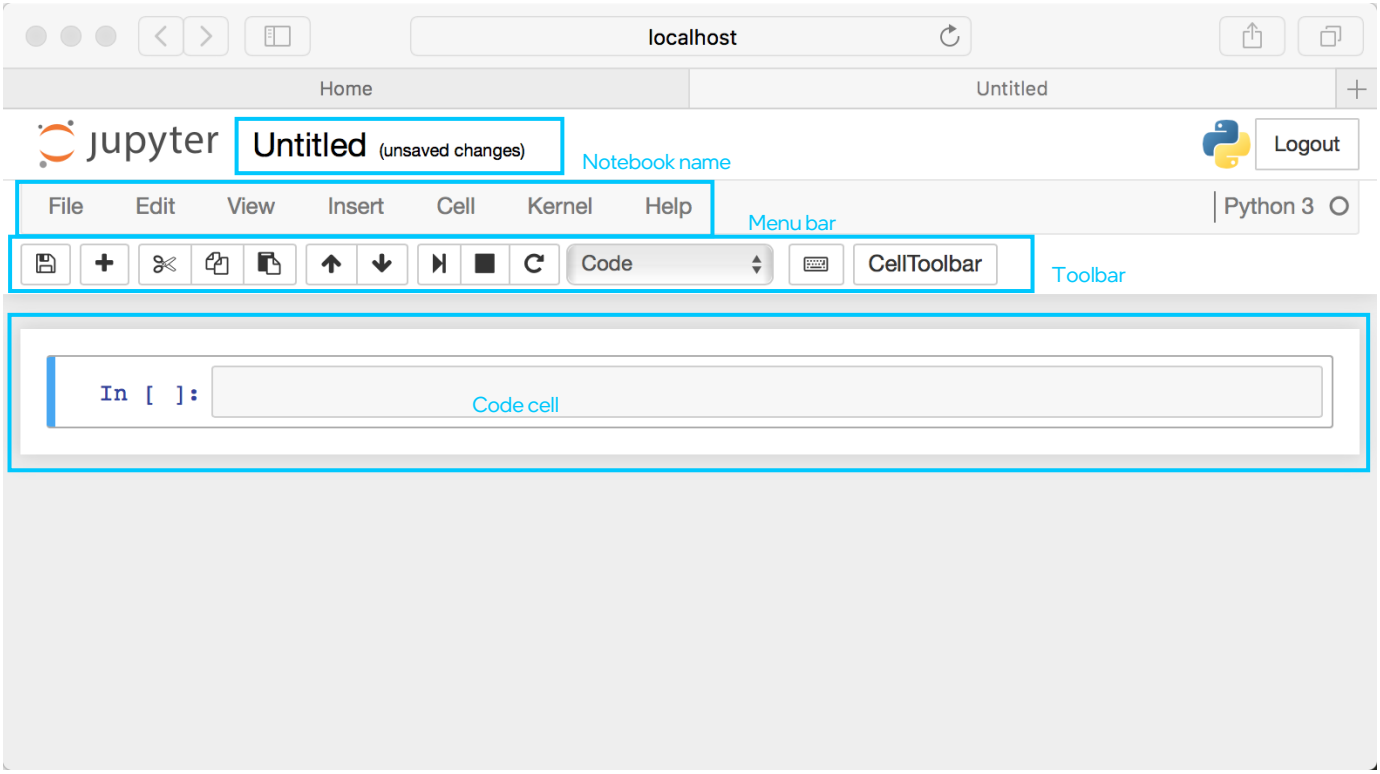
Idle has a simple interface where you can write and execute Python commands. Simply open it and start writing your code after the '>>> 'prompt. However, if you want to write programs, open a new file under File and begin.

Jupyter Notebooks

When you first open Jupyter, this dashboard appears. To open a Notebook, select New and Python 3.

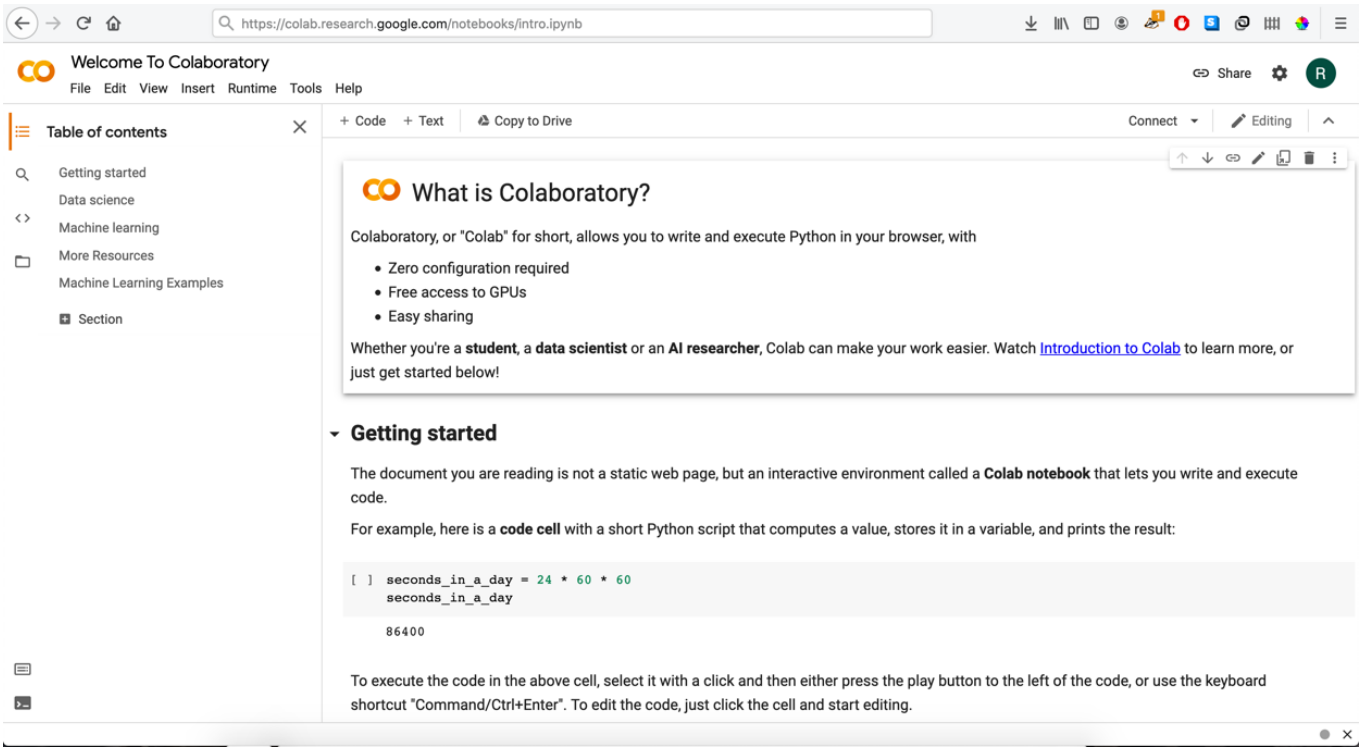
Notebook interface

After you open the Jupyter notebook, this is what you will see:



For a detailed review of all the menu options in Jupyter Notebooks, click here. You can run Jupyter in Firefox, Chrome, and Safari.

Google Colab



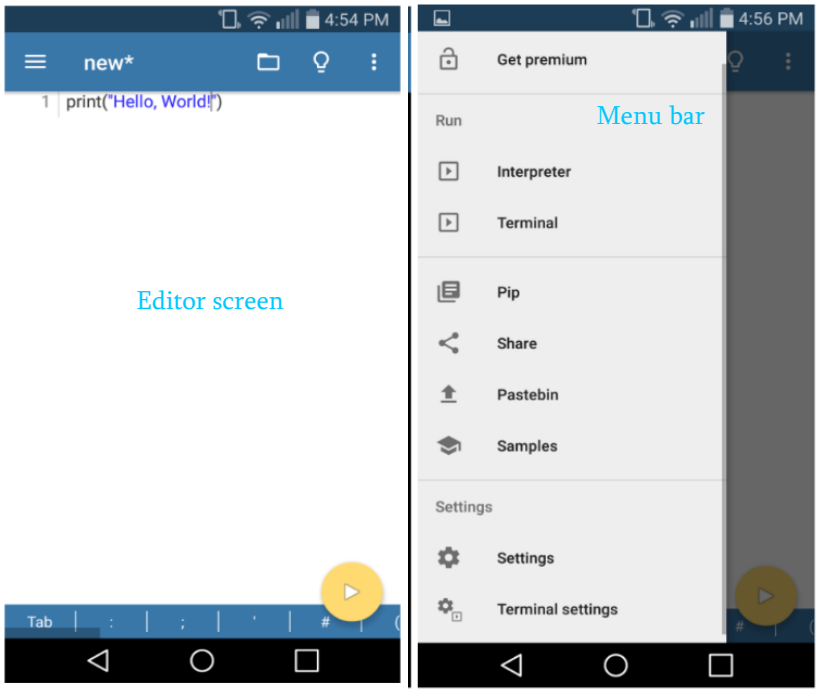
Colab Notebooks are Jupyter Notebooks hosted by Colab, so the interface is very similar. For more information, [read this overview of Colab.]

Pydroid Interface

When you launch Pydroid, it opens into an editor with a large play button to execute scripts.

The menu bar on the side panel lets you open the Python terminal and share your code.

Note: This application runs on an Android phone.



Basic Tools/Features of Python

In this section, we'll help you execute a few simple commands using Python syntax. You can enter these commands in any of the interfaces shown above. The links in this guide contain space for you to practice entering the commands.

Click on the following links to learn about each function and practice them:

- Your first command
- Variables, strings, and numbers
- Lists
- Basic operators
- Conditions
- Loops
- Functions
- Classes and objects
- Dictionaries

Jupyter Interface

Once you have grasped these basic commands, watch this video to get an overview of Jupyter Notebooks' features.

For extra practice and a fun game to play with your learners, try Cows and Bulls.

Troubleshooting

Writing code is a process that requires a lot of attention to detail. Small mistakes can cause your code to break and create errors when running the program, so pay close attention.

There are three main types of error in Python:

- **Compile-time errors** - These are errors that occur when you ask Python to run the application.
- **Run-time errors** - These occur after you have compiled the code and the program is running.
- **Syntax errors** – These happen when you have entered a code wrongly.

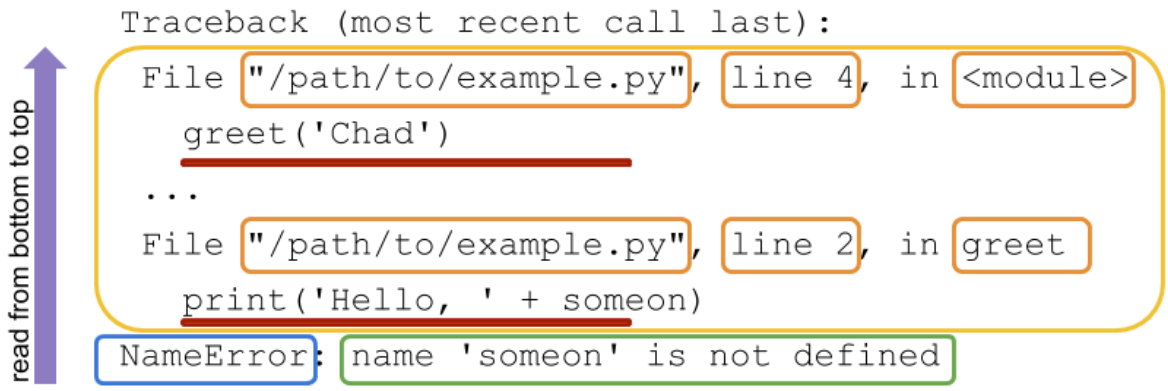
Python will usually dislay the error type, so you can easily look up the error code online to learn how to fix it.

Here is a complete list of error types. You can also look up errors here:

- The Jupyter Discourse Forum
- The Jupyter-Notebook tag on Stackoverflow

Troubleshooting Tips:

- If Jupyter doesn't start, try opening a terminal or command prompt and running the command Jupyter notebook.
- If it still doesn't open or shows a PATH error, reinstall Anaconda with the default settings.
- If Jupyter hangs, try opening it in another browser or disable browser extensions.
- Alternatively, try changing between localhost and 127.0.0.1. in the address bar. They should be the same, but it makes a difference in some cases.
- If you get a "permission denied" error when you attempt to run a Python script from the command line, it's most likely that the permissions are wrong. To check permissions, list the files in the directory with the command: sudo ls -la, and you should be able to read the permissions for the file.
- If you receive a "Bad interpreter: No such file or directory" error, find the correct path by typing in the command which Python and check the file location.
- Python is sensitive to white space, so if you copy and paste a script from a web page, you may accidentally paste in whitespace. You need to delete this to run the script correctly. The Python error message will often tell you exactly where the extra whitespace is located.
- "Type Error" messages mean that you are treating one type of data as if it were another. You can either change the data type or read the data as an integer.
- Use Python traceback to spot errors and understand how to fix your code. When a program results in an exception, Python prints a traceback to help you find what went wrong. Here's how to read a simple traceback. (You read from the bottom → top



1. **Blue box** -The last line of the traceback is the error message line. It contains the exception name that was raised.
2. **Green box** -This is the error message. It usually contains helpful information to understanding why the exception was raised.
3. **Yellow box** - Further up the traceback are various function calls moving from bottom to top (most recent to least recent). These calls are represented by two-line entries for each call. The first line of each call contains information like the file name, line number, and module name, all specifying where the code is found.
4. **Red underline** - The second line for these calls contains the executed code.

Source: Real Python

Other common omissions and syntax errors include:

- Using a single = when you meant ==
- Forgetting to put a : at the end of a statement like for, while, if, etc.
- A missing] or) bracket
- Using a ; semicolon when you meant a : colon
- Accidentally replacing one (parenthesis or [square bracket with a { curly bracket
- Spelling errors such as fro instead of for
- Leaving out a ' quote mark when working with a string, like print('widgets) or print(widgets')

You can learn more about some of the common mistakes users make with Python here.

Tips for Educators

- As with any language, learning Python requires commitment. Set aside some time every day to practice your Python skills and become fluent!
- Use Jupyter's code completion feature to learn about possible commands. Simply type in the first few characters of the function (or file) and press the Tab key. This will bring up a list of commands you can use.
- All the lines in a block must use the same indentation, either space or a tab.
- Python recommends **four spaces** as indentation to make the code more readable. Do not mix space and tab in the same block.
- You cannot write multi-line comments. For longer comments, each line should have the # symbol at the start.
- **Here are some helpful keyboard shortcuts:**
  - 'Esc' - Command mode, which enables you to navigate using keyboard shortcuts
  - 'Shift+Enter' - Run the program
  - 'Alt+Enter' - Run the current cell, insert below
  - 'Enter' - Edit mode that lets you edit text in cells
  - 'Space' - Scroll down notebook
  - 'C' - Copy selected cells
  - 'P' - Open the command palette

    - 'H' - Show all shortcuts
- Don't be afraid to experiment with Python. Click on this link for more tips.

Next Steps

Once you're confident using the Python's basic features, check out these resources to take your learning to the next level.

Practice Exercises

Practice makes perfect! Follow these practice exercises yourself or set them as assignments in class:

- Over 300 exercises for you to practice, from simple to advanced: https://www.hackinscience.org/exercises/
- Use these lessons to practice yourself, or introduce them to your learners: https://checkio.org/

Teaching tools:

- Guido van Robot: A teaching tool where learners write simple programs using a Python-like language to control a simulated robot. Field-tested at Yorktown High School, the project includes a lesson plan.
- Create your own Desktop Pet: Learn programming fundamentals in Python while building a digital pet.
- Online Python Tutor: Shows learners what happens as a computer executes each line of code. Educators and learners can use this tool to visualize what the computer does, step-by-step.

Further learning

- Python tutorials for further learning: https://www.askpython.com/
- Free interactive Python textbooks: https://runestone.academy/ns/books/published/overview/index.html
- E-book on the basics of Python: https://python.swaroopch.com/about_python.html
- Comprehensive Python course by Codeacademy: https://www.codecademy.com/learn/learn-python-3
- Python libraries where you can find and install packages you need: https://pypi.org/ (Intermediate usage)
- Glossary of python terms: https://docs.python.org/3/glossary.html

