

Intro to Python — SMM692

Getting Started with Python

Simone Santoni

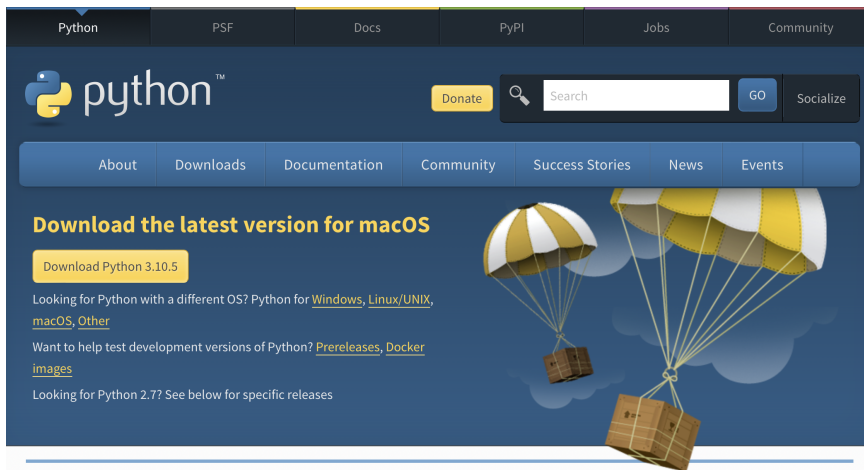
Bayes Business School

MSc Pre-Course Series

Outline

Outline

Option 1: Official Installer



The screenshot shows the Python.org homepage. At the top, there is a navigation bar with links to Python, PSF, Docs, PyPI, Jobs, and Community. Below this is the Python logo and a search bar. A yellow 'Donate' button is also visible. A secondary navigation bar contains links to About, Downloads, Documentation, Community, Success Stories, News, and Events. The main content area features a large banner for downloading the latest version for macOS, with a yellow button labeled 'Download Python 3.10.5'. To the right of the text is an illustration of two parachutes carrying crates. Below the banner, there are links for other operating systems and development versions.

Download the latest version for macOS

[Download Python 3.10.5](#)

Looking for Python with a different OS? Python for [Windows](#), [Linux/UNIX](#), [macOS](#), [Other](#)

Want to help test development versions of Python? [Prereleases](#), [Docker images](#)

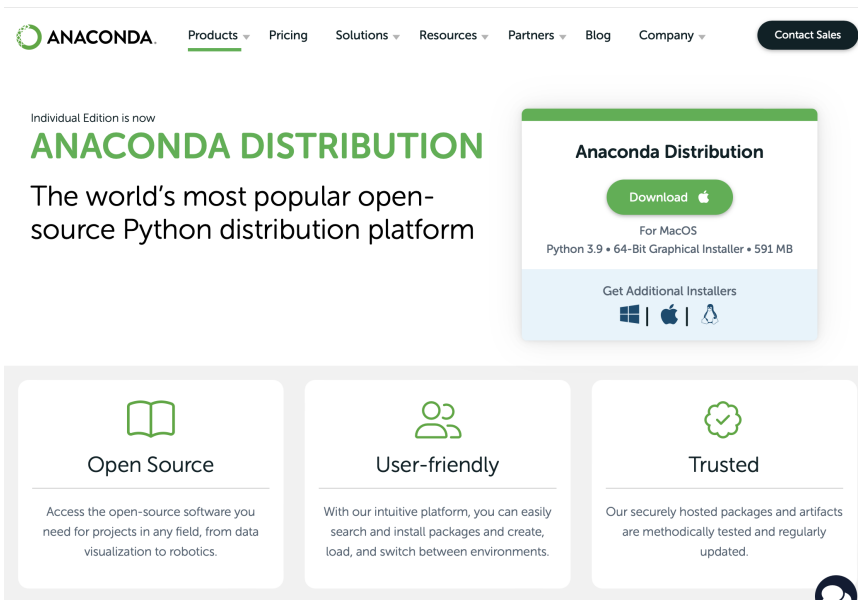
Looking for Python 2.7? See below for specific releases

Active Python Releases

For more information visit the [Python Developer's Guide](#).

Python version	Maintenance status	First released	End of support	Release schedule
3.10	bugfix	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596

Option 2: Anaconda Distro (Preferred Way)



The screenshot shows the Anaconda Distribution website. At the top is a navigation bar with the Anaconda logo, links for Products, Pricing, Solutions, Resources, Partners, Blog, and Company, and a Contact Sales button. The main content area features the text 'Individual Edition is now' followed by 'ANACONDA DISTRIBUTION' in large green letters. Below this is the tagline 'The world's most popular open-source Python distribution platform'. To the right is a download box for macOS with a 'Download' button and details about the installer. Below the main text are three feature boxes: 'Open Source', 'User-friendly', and 'Trusted', each with an icon and a description. At the bottom right is a chat bubble icon and a set of navigation icons.

ANACONDA


Products Pricing Solutions Resources Partners Blog Company Contact Sales

Individual Edition is now

ANACONDA DISTRIBUTION




The world's most popular open-source Python distribution platform


Anaconda Distribution


Download 


For MacOS
Python 3.9 • 64-Bit Graphical Installer • 591 MB

Get Additional Installers

 |  | 


Open Source
Access the open-source software you need for projects in any field, from data visualization to robotics.


User-friendly
With our intuitive platform, you can easily search and install packages and create, load, and switch between environments.


Trusted
Our securely hosted packages and artifacts are methodically tested and regularly updated.

Outline

Steps

- ❶ Download the installer for your operating system (unless you have a very old machine running Win, go for the 64-Bit version)
- ❷ Run the installer
 - For Linux: navigate to the folder where you have downloaded the installer as per step 1, open a shell session, then run `$ bash ./Anaconda3-XXXX.XX-Linux-x86-64.sh`
 - For Win and Mac OS: just run the graphical installer downloaded in step 1
- ❸ Accept the terms proposed by the Anaconda people to use their software, comprising Python, the conda package manager, and a bundle of modules for data science
- ❹ That's it!
 - For Linux users: if you accepted the default installation options, an environmental variable has been created either in your `.bashrc` or `.zshrc`. That means you can access the various pieces of software included in the Anaconda installation (e.g., Anaconda Navigator) from a shell session
 - For Win and Mac OS users: the various pieces of software included in the Anaconda installation are available from the menu of your system

Outline

How is the Anaconda Distro Different?

- Anaconda is 'battery-included' — it comes with a humongous number of modules for data science
 - If you use the official Python installation, then you have to install the modules you need on your own!
- Anaconda is a bundle of various pieces of software:
 - `conda` is the Swiss army knife to manage Python modules and environments
 - `anaconda-navigator` is the graphical interface from within to access Python IDEs and related desktop/web applications

Outline

What Software Can I Access from Anaconda Navigator?

[Connect](#)[Home](#)[Environments](#)[Learning](#)[Community](#)[End-to-end package security, guaranteed](#)[Documentation](#)[Anaconda Blog](#)Applications on base (root)[Channels](#)**DataSpell**

DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.

[Install](#)**DataLore**

Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.

[Launch](#)**IBM Watson Studio Cloud**

IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

[Launch](#)**JupyterLab**[3.0.14](#)

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

[Launch](#)**Notebook**[6.3.0](#)

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

**IPyT Console**[5.0.3](#)

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.

**Spyder**[4.2.5](#)

Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

**VS Code**[1.69.2](#)

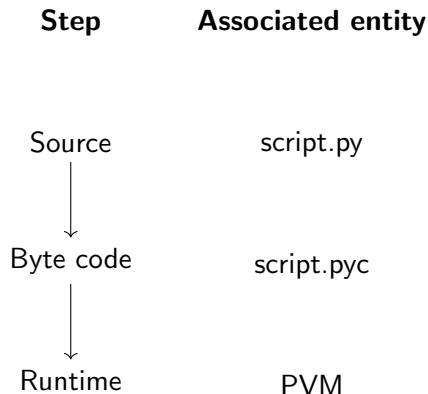
Streamlined code editor with support for development operations like debugging, task running and version control.

Python as a Language and Interpreter

- Typically, we refer to Python as a programming language
- However, Python is also a software package called an interpreter
 - An interpreter is a kind of program that executes other programs
 - When you write a Python program, the Python interpreter reads your program and carries out the instructions it contains
 - In effect, the interpreter is a layer of software logic between your code and the computer hardware on your machine

Python Execution Model

- When you instruct Python to run your script (e.g., 'script.py'), there are a few steps the interpreter carries out before your code actually starts crunching away
- First, the code is compiled to something called 'byte code'
 - This step happens behind the scenes (there is nothing to do for the programmer!)
 - A file called 'script.pyc', automatically generated, contains the translation of your code into lower-level code instructions
- Then, the compiled code is routed to something called a 'Python Virtual Machine' (PVM)
 - This step is hidden to the programmer like the previous one
 - Mainly, it iterates through your byte code instructions, one by one, to carry out their operations



Outline

Script Preparation → Script Run

Step 1: script preparation

The below displayed Python code achieves two things: i) it prints the string object “Bazinga!”, and ii) it prints the result of the algebraic operation $4 + 2$. Note that all lines starting with # are not considered Python code — instead, they are comments that illustrate/explain the logic of the script.

```
# Print a string object
print("Bazinga")
# Print the result of an algebraic operation
print(2 + 4)
```

Step 2: script run



```
(base) → ~ python simple_script.py
Bazinga
6
(base) → ~
```

Outline

Running a Python Shell in the Terminal

```
(base) → ~ python
Python 3.9.7 (default, Sep 16 2021, 08:50:36)
[Clang 10.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Bazinga")
Bazinga
>>> print(2 + 4)
6
>>> █
```

Running an IPython Shell in the Terminal

```
(base) → ~ ipython
Python 3.9.7 (default, Sep 16 2021, 08:50:36)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.29.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: print("Bazinga")
Bazinga

In [2]: print(2 + 4)
6

In [3]:
```

Running a Python IDE

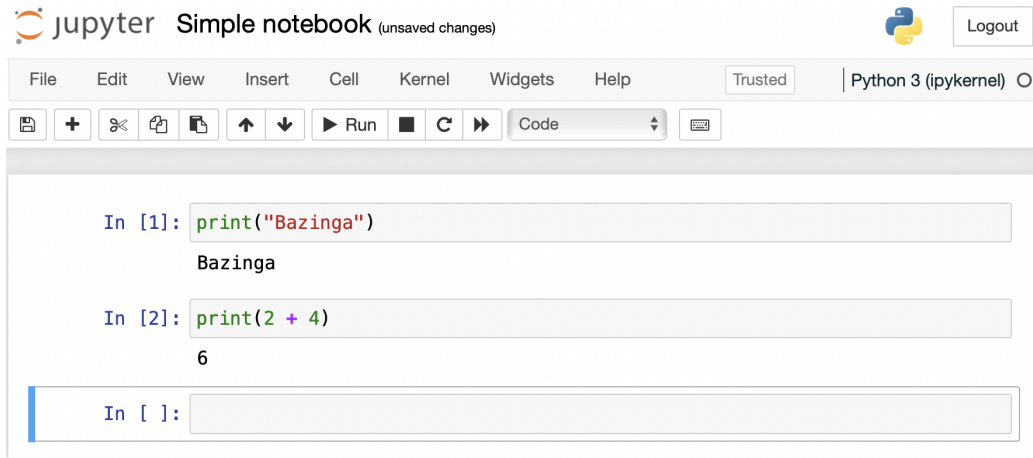
There are plenty of Python IDEs in the market, including:

- Colab (online)
- IDLE
- Datalore (online)
- Jupyter/Jupyterlab
- PyCharm
- Qt Console
- Spyder
- Thonny
- Wing

By installing a couple of plugins, the following (advanced) text editors can turn into Python IDEs:

- Emacs
- Vim/Neovim
- VSCode

Interactive Python Coding with Jupyter



The screenshot shows a Jupyter Simple notebook interface. At the top left is the Jupyter logo and the text "Simple notebook (unsaved changes)". At the top right is the Python logo and a "Logout" button. Below this is a menu bar with "File", "Edit", "View", "Insert", "Cell", "Kernel", "Widgets", and "Help". To the right of the menu bar are buttons for "Trusted" and "Python 3 (ipykernel)". Below the menu bar is a toolbar with icons for saving, adding, deleting, copying, pasting, undo, redo, and running code. The main area contains three code cells. The first cell has the input "In [1]: print('Bazinga')" and the output "Bazinga". The second cell has the input "In [2]: print(2 + 4)" and the output "6". The third cell is empty and has the input "In []:".

jupyter Simple notebook (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Save Add Delete Copy Paste Undo Redo Run Code

```
In [1]: print("Bazinga")
Bazinga
```

```
In [2]: print(2 + 4)
6
```

```
In [ ]:
```

Interactive Python Coding with VSCode

The screenshot shows the VS Code interface with a file named `simple_script.py` open. The code in the editor is:

```
1 print("Bazinga")
2 print(2 + 4)
```

The right-hand side of the interface shows the `Interactive - simple_script.py` console. It displays the output of the script execution:

```
Python 3.9.7 (default, Sep 16 2021, 08:50:36)
Type 'copyright', 'credits' or 'license' for more
information
IPython 7.29.0 -- An enhanced Interactive
Python. Type '?' for help.

✓ print("Bazinga") ...
... Bazinga

✓ print(2 + 4) ...
... 6
```

The console also shows the prompt `tech_sci (Python 3.9.7)` and buttons for `Clear All` and `Restart`.

Interactive Python Coding in Jupyter with VSCode

The screenshot displays the VS Code Jupyter interface for a notebook titled "simple_script.ipynb — intro-to-Python-SMM692". The interface includes a sidebar on the left with icons for Explorer, Search, Source Control, Run and Debug, Testing, Extensions, Python, Remote Explorer, and Docker. The main editor area shows the notebook content. At the top, there's a toolbar with icons for settings, play, refresh, and other actions. Below the toolbar, the notebook has a "Code" tab selected, and a "Run All" button is visible. The notebook contains two executed code cells. The first cell contains the code `print("Bazinga")` and shows the output "Bazinga" with a status of "[1] ✓ 0.3s". The second cell contains the code `print(2 + 4)` and shows the output "6" with a status of "[2] ✓ 0.2s". Both cells are labeled "Python" on the right. A third, empty code cell is visible at the bottom, with a status of "[]". A context menu is open over the empty cell, showing options like "Run", "Run and Debug", "Run and Test", "Run and Profile", "Run and Coverage", "Run and Coverage (Python)", "Run and Coverage (C/C++)", "Run and Coverage (Java)", "Run and Coverage (JavaScript)", "Run and Coverage (TypeScript)", "Run and Coverage (Go)", "Run and Coverage (Rust)", "Run and Coverage (C#)", "Run and Coverage (F#)", "Run and Coverage (VB)", "Run and Coverage (Swift)", "Run and Coverage (Kotlin)", "Run and Coverage (Scala)", "Run and Coverage (Java (Android))", "Run and Coverage (Java (Spring))", "Run and Coverage (Java (JUnit))", "Run and Coverage (Java (JUnit4))", "Run and Coverage (Java (JUnit5))", "Run and Coverage (Java (JUnit6))", "Run and Coverage (Java (JUnit7))", "Run and Coverage (Java (JUnit8))", "Run and Coverage (Java (JUnit9))", "Run and Coverage (Java (JUnit10))", "Run and Coverage (Java (JUnit11))", "Run and Coverage (Java (JUnit12))", "Run and Coverage (Java (JUnit13))", "Run and Coverage (Java (JUnit14))", "Run and Coverage (Java (JUnit15))", "Run and Coverage (Java (JUnit16))", "Run and Coverage (Java (JUnit17))", "Run and Coverage (Java (JUnit18))", "Run and Coverage (Java (JUnit19))", "Run and Coverage (Java (JUnit20))", "Run and Coverage (Java (JUnit21))", "Run and Coverage (Java (JUnit22))", "Run and Coverage (Java (JUnit23))", "Run and Coverage (Java (JUnit24))", "Run and Coverage (Java (JUnit25))", "Run and Coverage (Java (JUnit26))", "Run and Coverage (Java (JUnit27))", "Run and Coverage (Java (JUnit28))", "Run and Coverage (Java (JUnit29))", "Run and Coverage (Java (JUnit30))", "Run and Coverage (Java (JUnit31))", "Run and Coverage (Java (JUnit32))", "Run and Coverage (Java (JUnit33))", "Run and Coverage (Java (JUnit34))", "Run and Coverage (Java (JUnit35))", "Run and Coverage (Java (JUnit36))", "Run and Coverage (Java (JUnit37))", "Run and Coverage (Java (JUnit38))", "Run and Coverage (Java (JUnit39))", "Run and Coverage (Java (JUnit40))", "Run and Coverage (Java (JUnit41))", "Run and Coverage (Java (JUnit42))", "Run and Coverage (Java (JUnit43))", "Run and Coverage (Java (JUnit44))", "Run and Coverage (Java (JUnit45))", "Run and Coverage (Java (JUnit46))", "Run and Coverage (Java (JUnit47))", "Run and Coverage (Java (JUnit48))", "Run and Coverage (Java (JUnit49))", "Run and Coverage (Java (JUnit50))", "Run and Coverage (Java (JUnit51))", "Run and Coverage (Java (JUnit52))", "Run and Coverage (Java (JUnit53))", "Run and Coverage (Java (JUnit54))", "Run and Coverage (Java (JUnit55))", "Run and Coverage (Java (JUnit56))", "Run and Coverage (Java (JUnit57))", "Run and Coverage (Java (JUnit58))", "Run and Coverage (Java (JUnit59))", "Run and Coverage (Java (JUnit60))", "Run and Coverage (Java (JUnit61))", "Run and Coverage (Java (JUnit62))", "Run and Coverage (Java (JUnit63))", "Run and Coverage (Java (JUnit64))", "Run and Coverage (Java (JUnit65))", "Run and Coverage (Java (JUnit66))", "Run and Coverage (Java (JUnit67))", "Run and Coverage (Java (JUnit68))", "Run and Coverage (Java (JUnit69))", "Run and Coverage (Java (JUnit70))", "Run and Coverage (Java (JUnit71))", "Run and Coverage (Java (JUnit72))", "Run and Coverage (Java (JUnit73))", "Run and Coverage (Java (JUnit74))", "Run and Coverage (Java (JUnit75))", "Run and Coverage (Java (JUnit76))", "Run and Coverage (Java (JUnit77))", "Run and Coverage (Java (JUnit78))", "Run and Coverage (Java (JUnit79))", "Run and Coverage (Java (JUnit80))", "Run and Coverage (Java (JUnit81))", "Run and Coverage (Java (JUnit82))", "Run and Coverage (Java (JUnit83))", "Run and Coverage (Java (JUnit84))", "Run and Coverage (Java (JUnit85))", "Run and Coverage (Java (JUnit86))", "Run and Coverage (Java (JUnit87))", "Run and Coverage (Java (JUnit88))", "Run and Coverage (Java (JUnit89))", "Run and Coverage (Java (JUnit90))", "Run and Coverage (Java (JUnit91))", "Run and Coverage (Java (JUnit92))", "Run and Coverage (Java (JUnit93))", "Run and Coverage (Java (JUnit94))", "Run and Coverage (Java (JUnit95))", "Run and Coverage (Java (JUnit96))", "Run and Coverage (Java (JUnit97))", "Run and Coverage (Java (JUnit98))", "Run and Coverage (Java (JUnit99))", "Run and Coverage (Java (JUnit100))".

Outline

What is a Python Environment?

Step 1: What is a Python Environment?

Step 2: Why would I use a Python Environment?

Outline

Outline

Outline

