

## 1. Introducere. Scurt istoric

- **Python** (<https://www.python.org/>) este un limbaj de programare cu interpretare de uz-general, interactiv, orientat pe obiecte și de nivel înalt.
- A fost creat de către olandezul Guido van Rossum (<https://gvanrossum.github.io/>), în anul 1989, la Institutul Național de Cercetare pentru Matematică și Informatică, fiind un derivat al mai multor limbaje de programare, printre care: ABC, Modula-3, C, C++, Unix shell.
- Codul sursă de **Python** este disponibil sub Licența Publică Generală (General Public License - GPL) GNU (<https://www.gnu.org/licenses/gpl-3.0.html>).
- **Python** este un *limbaj dinamic* (interpretorul îl transformă în cod de octeți în timpul execuției) precum PHP sau Perl, *multifuncțional* (a fost astfel proiectat încât să poată fi folosit pentru a dezvolta diverse tipuri de software) și de *nivel înalt* (prezintă un grad mare de abstractizare față de codul-mașină - astfel încât se aseamănă mai mult cu limbajul natural decât cu codul-mașină - ceea ce-i conferă eficiență și citibilitate sporite). De asemenea este un limbaj cu *tipizare dinamică* (dynamically typed) - interpretorul încearcă să stabilească tipul variabilelor și obiectelor în timpul execuției (at runtime). Aceasta înseamnă că tipul variabilelor nu trebuie declarat înainte de a le atribui o valoare.
- **Python** este un limbaj de programare în totalitate orientat pe obiecte, și nu “tastat static”. Spre deosebire de alte limbaje de programare, nu trebuie să declari variabilele sau tipul lor înainte de a le utiliza. Fiecare variabilă în Python este considerată un obiect.
- În multe limbaje - PHP, C++, Java - pentru a arăta sfârșitul unei instrucțiuni se folosește caracterul ;, iar pentru a arăta că mai multe instrucțiuni fac parte din același bloc instrucțional se încadrează acestea între acolade ({ }), în **python** fiecare instrucțiune este pusă singură pe un rând (deși se poate prelungi o instrucțiune pe mai multe rânduri, din motive estetice, punându-se \ când se trece la rândul următor), iar indentarea este cea care arată/stabilește ce instrucțiuni formează un bloc instrucțional: instrucțiunile succesive, care sunt indentate cu același număr de spații (de obicei patru, dar uneori și două), formează un bloc instrucțional.
- Python has maintained its grip on the No. 1 spot (<https://spectrum.ieee.org/at-work/innovation/the-2018-top-programming-languages>)

Language Rank	Types	Spectrum Ranking
1. Python	  	100.0
2. C++	  	99.7
3. Java	  	97.5
4. C	  	96.7
5. C#	  	89.4
6. PHP		84.9
7. R		82.9
8. JavaScript	 	82.6
9. Go	 	76.4
10. Assembly		74.1

- Python Mobile SIG (<https://www.python.org/community/sigs/current/mobile-sig/>): The Mobile-SIG exists to improve the usability of Python on mobile devices, as mobile platforms have their own unique requirements and constraints.

## 2. Anaconda (<https://www.anaconda.com/download/> )

Anaconda este o distributie python “completa” (<https://docs.anaconda.com/anaconda/> ), cu surse deschise , cu o comunitate de peste 6 milioane de utilizatori. Este usor de descarcat si instalat si este acceptat pe Linux, MacOS si Windows. Distributia vine cu mai mult de 1.000 de pachete de date, precum si pachetul Conda si managerul de mediu virtual.

### 2.1. Installation (<https://docs.anaconda.com/anaconda/install/>)

TIP: If you don’t want the hundreds of packages included with Anaconda, you can [install Miniconda](#), a mini version of Anaconda that includes just conda, its dependencies and Python.

TIP: Looking for Python 3.5 or 3.6? See our [FAQ](#).

#### System requirements

- License: Free use and redistribution under the terms of the [Anaconda End User License Agreement](#).
- Operating system: Windows Vista or newer, 64-bit macOS 10.10+, or Linux, including Ubuntu, RedHat, CentOS 6+, and others.
- Windows XP supported on Anaconda versions 2.2 and earlier. See [Old package lists](#). Download it from our [archive](#).
- System architecture: 64-bit x86, 32-bit x86 with Windows or Linux, Power8 or Power9.
- Minimum 3 GB disk space to download and install.

#### Detailed installation information

- [Installing on Windows](#)
- [Installing on macOS](#)
- [Installing on Linux](#)
- [Installing on Linux POWER8 or POWER9](#)
- [Verifying your installation](#)
- [Anaconda installer file hashes](#)
- [Updating from older versions](#)
- [Uninstalling Anaconda](#)

#### General installation information

On Windows, macOS, and Linux, it is best to install Anaconda for the local user, which does not require administrator permissions and is the most robust type of installation. However, if you need to, you can install Anaconda system wide, which does require administrator permissions.

## Silent mode install

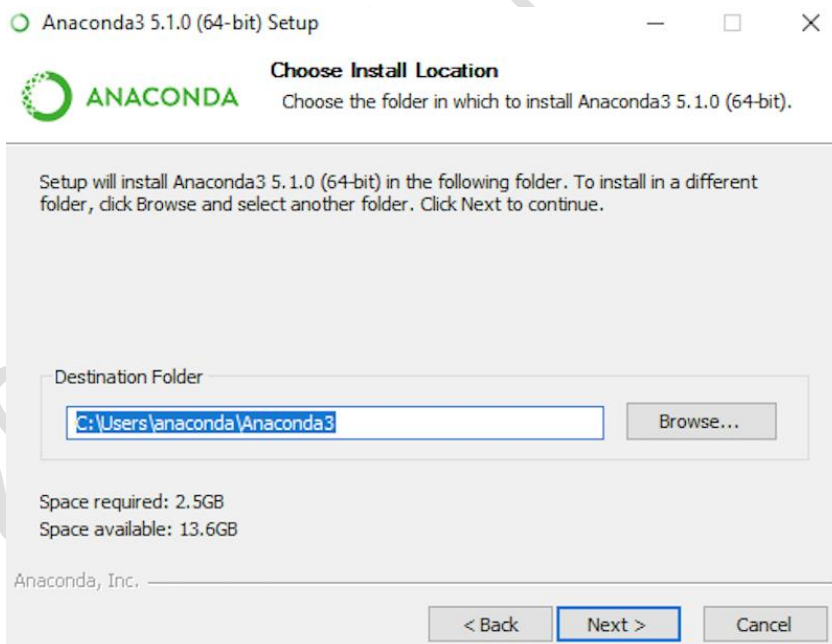
You can use [silent mode](#) to automatically accept default settings and have no screen prompts appear during installation.

## Installing Anaconda on a non-networked machine (air gap)

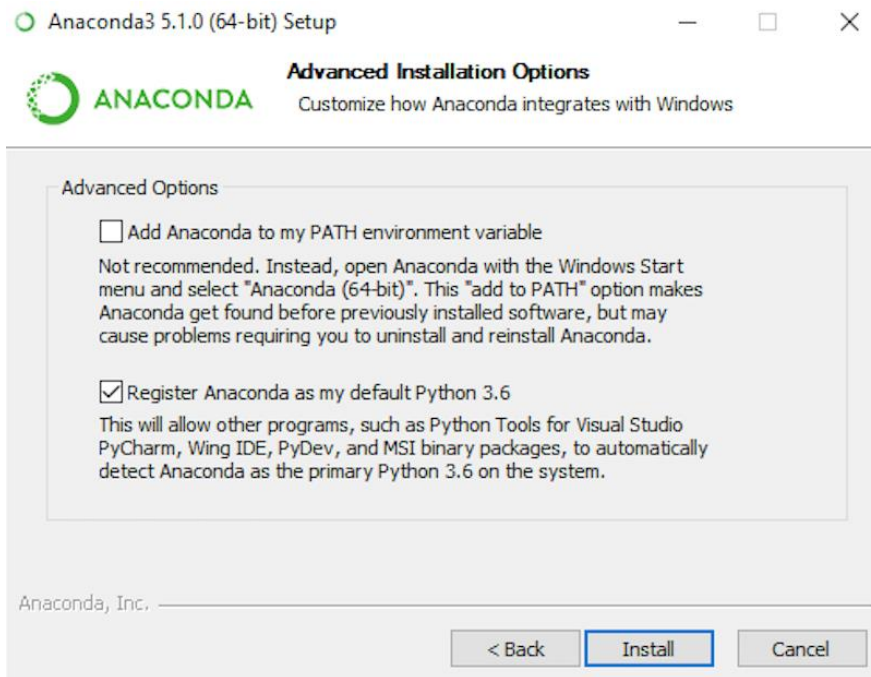
1. Obtain a local copy of the appropriate Anaconda installer for the non-networked machine. You can copy the Anaconda installer to the target machine using many different methods including a portable hard drive, USB drive or CD.
2. After copying the installer to the non-networked machine, follow the detailed installation instructions for your operating system.

## Installing on Windows (<https://docs.anaconda.com/anaconda/install/windows/>)

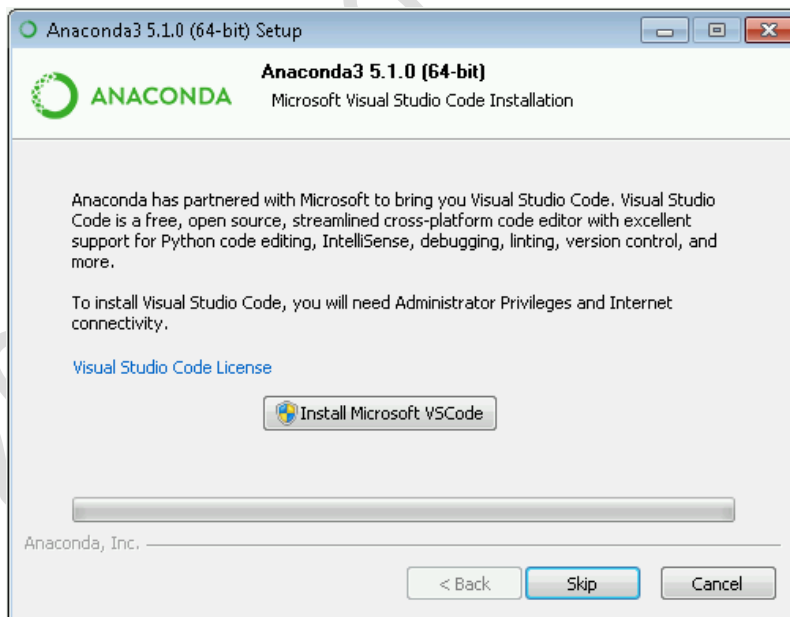
1. [Download the Anaconda installer.](#)
2. Optional: [Verify data integrity with MD5 or SHA-256.](#) [More info on hashes](#)
3. Double click the installer to launch.  
NOTE: To prevent permission errors, do not launch the installer from the [Favorites folder](#).  
NOTE: If you encounter issues during installation, temporarily disable your anti-virus software during install, then re-enable it after the installation concludes. If you installed for all users, uninstall Anaconda and re-install it for your user only and try again.
4. Click Next.
5. Read the licensing terms and click “I Agree”.
6. Select an install for “Just Me” unless you’re installing for all users (which requires Windows Administrator privileges) and click Next.
7. Select a destination folder to install Anaconda and click the Next button. See [FAQ](#).  
NOTE: Install Anaconda to a directory path that does not contain spaces or unicode characters.  
NOTE: Do not install as Administrator unless admin privileges are required.



8. Choose whether to add Anaconda to your PATH environment variable. We recommend not adding Anaconda to the PATH environment variable, since this can interfere with other software. Instead, use Anaconda software by opening Anaconda Navigator or the Anaconda Prompt from the Start Menu.



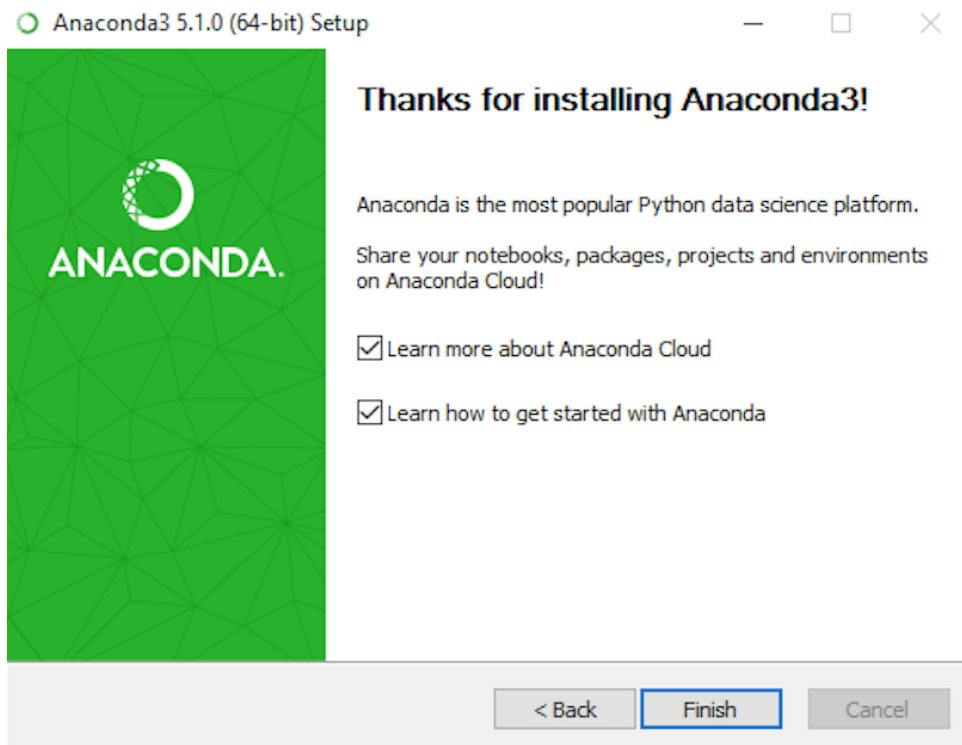
9. Choose whether to register Anaconda as your default Python 3.6. Unless you plan on installing and running multiple versions of Anaconda, or multiple versions of Python, accept the default and leave this box checked.
10. Click the Install button. If you want to watch the packages Anaconda is installing, click Show Details.
11. Click the Next button.
12. Optional: To [install VS Code](#), click the Install Microsoft VS Code button. After the install completes click the Next button.



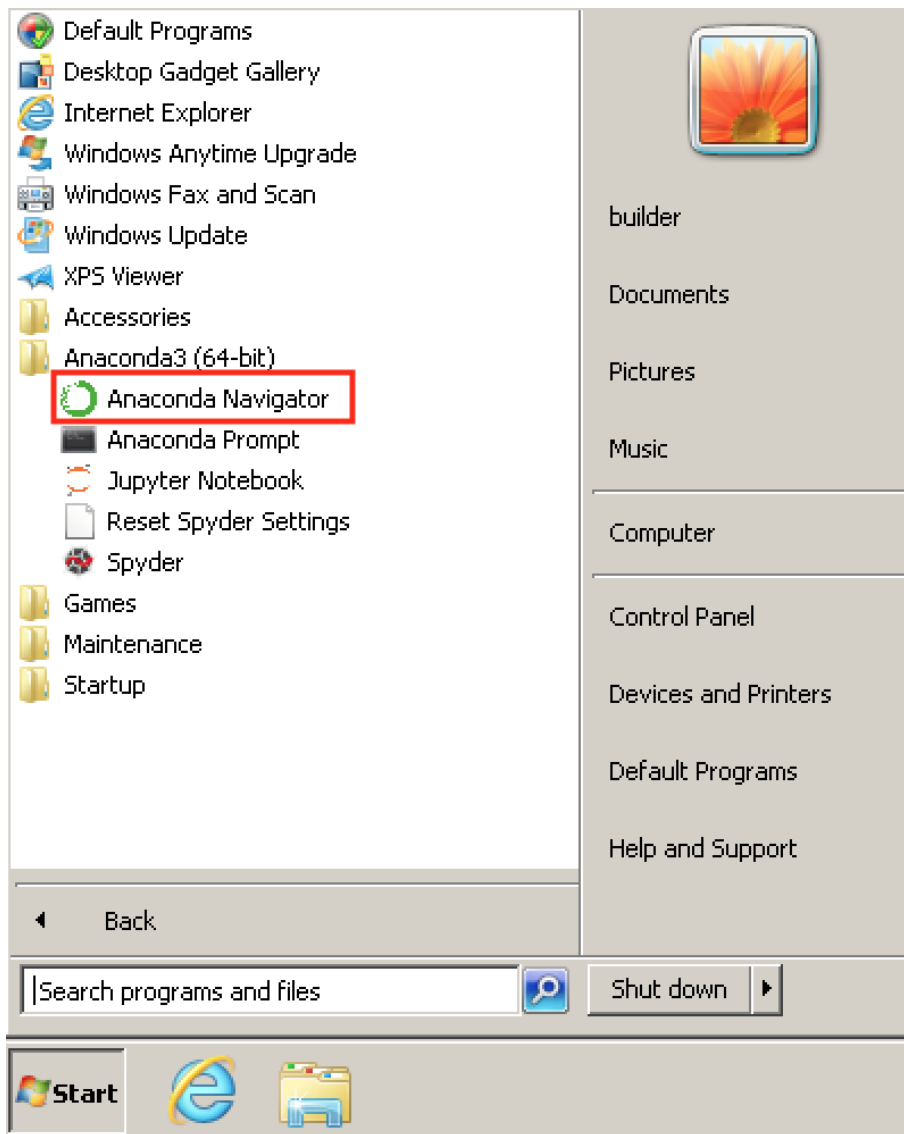
Or to install Anaconda without VS Code, click the Skip button.

NOTE: Installing VS Code with the Anaconda installer requires an internet connection. Offline users may be able to find an offline VS Code installer from Microsoft.

13. After a successful installation you will see the “Thanks for installing Anaconda” dialog box:



14. If you wish to read more about Anaconda Cloud package management service and Anaconda support, check the boxes “Learn more about Anaconda Cloud” and “Learn how to get started with Anaconda”. Click the Finish button.
15. After your install is complete, verify it by opening Anaconda Navigator, a program that is included with Anaconda: from your Windows Start menu, select the shortcut Anaconda Navigator. If Navigator opens, you have successfully installed Anaconda. If not, check that you completed each step above, then see our [Help page](#).



### Problems?

See [troubleshooting](#).

### What's next?

Get started programming quickly with Anaconda in the [Getting started with Anaconda](#) guide.

**Installing on macOS** (<https://docs.anaconda.com/anaconda/install/mac-os/>)

**Installing on Linux** (<https://docs.anaconda.com/anaconda/install/linux/>)

## 2.2. Verifying your installation

You can confirm that Anaconda is installed and working with Anaconda Navigator or conda.

### Anaconda Navigator

Open Anaconda Navigator, which is automatically installed when you install Anaconda.

- Windows: Click Start - then from the shortcuts, select Anaconda Navigator. If it opens, you have successfully installed Anaconda.
- macOS: Click Launchpad - then select Anaconda Navigator. If it opens, you have successfully installed Anaconda.
- Linux: See next section.

### Conda

You can also use conda in an Anaconda prompt (Terminal on Linux or macOS).

To open Anaconda Prompt (or Terminal on Linux or macOS):

- Windows: Open the Anaconda Prompt (Click Start, select Anaconda Prompt)
- macOS: Open Launchpad, then open Terminal or iTerm.
- Linux–CentOS: Open Applications - System Tools - Terminal.
- Linux–Ubuntu: Open the Dash by clicking the upper left Ubuntu icon, then type “terminal”.

After opening Anaconda prompt (Terminal on Linux or macOS), choose any of the following methods:

- Enter a command such as `conda list`. If Anaconda is installed and working, this will display a list of installed packages and their versions.
- Enter the command `python`. This command runs the Python shell. If Anaconda is installed and working, the version information it displays when it starts up will include “Anaconda”. To exit the Python shell, enter the command `quit()`.
- Open Anaconda Navigator with the command `anaconda-navigator`. If Anaconda is installed properly, the graphical program Anaconda Navigator will open.

If you find any problems, see the [troubleshooting guide](#) and the [Help and support](#) page for resources such as free community support and bug reports.

## 3. Getting started with Anaconda

Anaconda Distribution contains **conda** and **Anaconda Navigator**, as well as Python and hundreds of scientific [packages](#). When you installed Anaconda, you installed all these too. You can try both conda and Navigator to see which is right for you to manage your packages and environments. You can even switch between them, and the work you do with one can be viewed in the other.

Now, try this simple programming exercise two ways, with Navigator and a terminal, to help you decide which approach is right for you.

### Your first Python program: Hello, Anaconda!

Write and run a Python program using Anaconda Navigator.



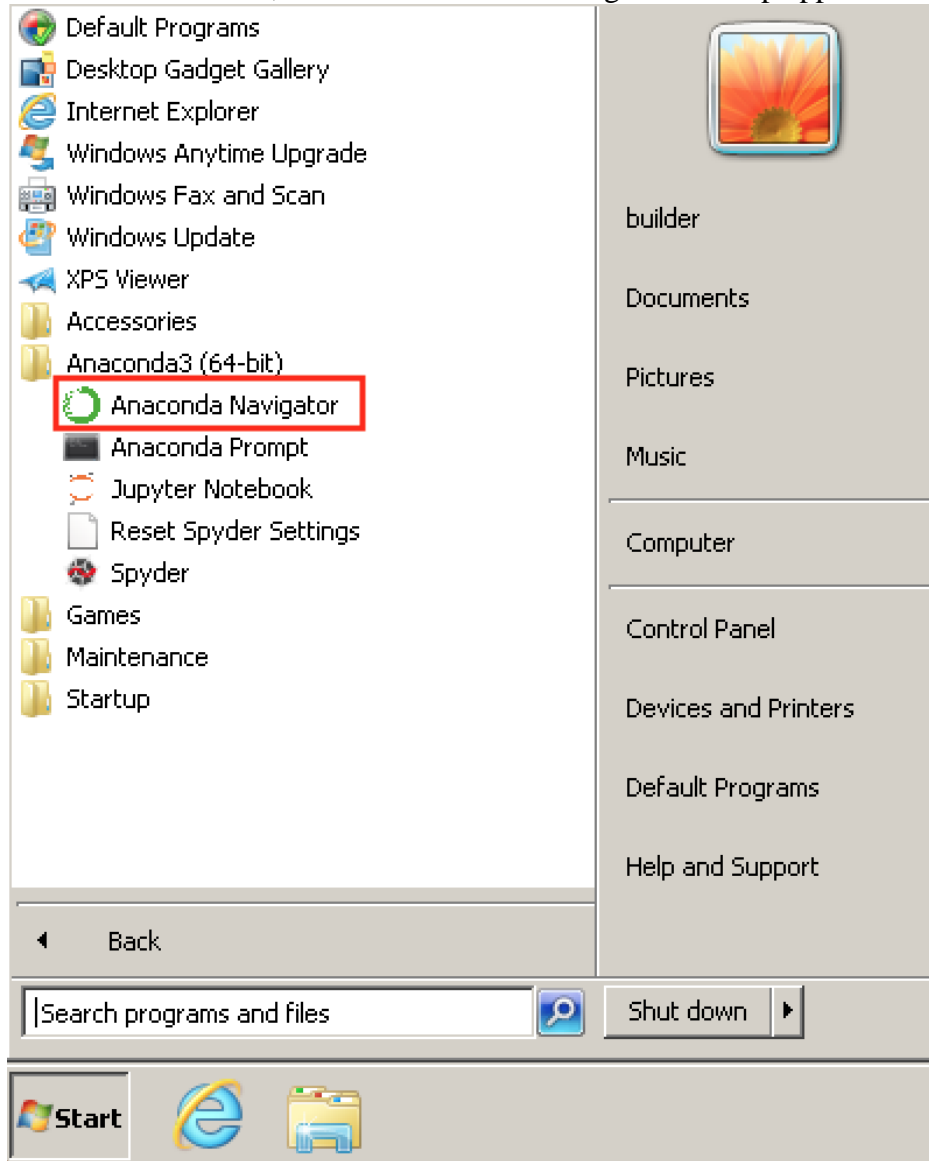
## 1. Open Navigator

Choose the instructions for your operating system.

- [Windows](#)
- [macOS](#)
- [Linux](#)

### Windows

From the Start menu, click the Anaconda Navigator desktop app.



### macOS

Open Launchpad, then click the Anaconda Navigator icon.

### Linux

Open a Terminal window and type `anaconda-navigator`.



## 2. Run Python in Spyder IDE (integrated development environment)

TIP: Navigator's Home screen displays several applications for you to choose from. For more information, see links at bottom of this page.

On Navigator's Home tab, in the Applications pane on the right, scroll to the Spyder tile and click the Install button to install Spyder.

NOTE: If you already have Spyder installed, you can jump right to the Launch step.

Launch Spyder by clicking Spyder's Launch button.

In the new file on the left, delete any placeholder text, then type or copy/paste `print("Hello Anaconda")`.

In the top menu, click File - Save As and name your new program `hello.py`.

Run your new program by clicking the triangle Run button.

You can see your program's output in the bottom right Console pane.

## 3. Close Spyder

From Spyder's top menu bar, select Spyder - Quit Spyder (In macOS, select Python - Quit Spyder).

## 4. Close Navigator

From Navigator's top menu bar, select Anaconda Navigator - Quit Anaconda-Navigator.

## Write a Python program using Anaconda Prompt or Terminal

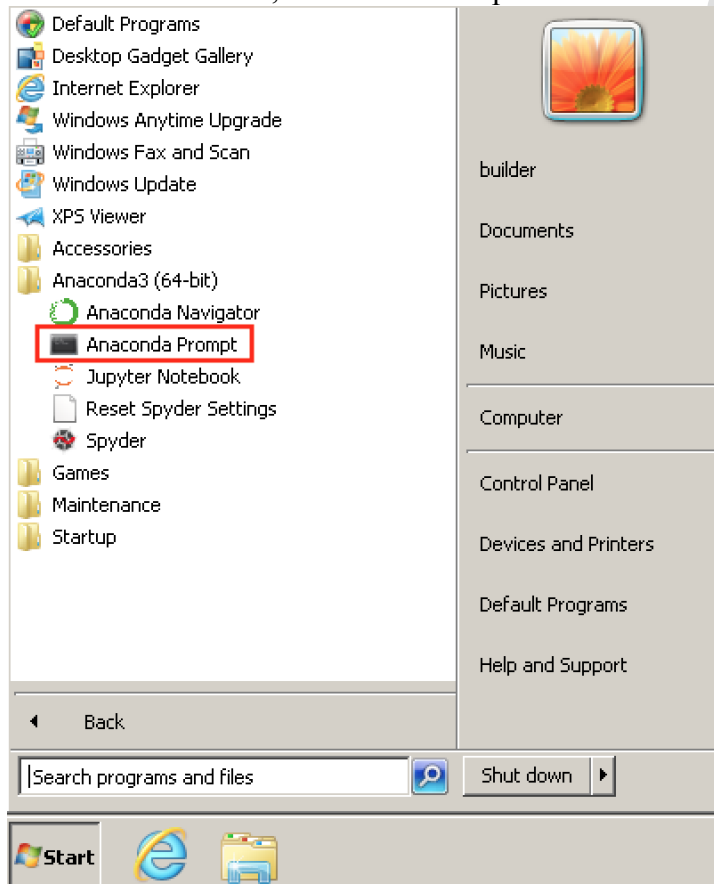
### 1. Open Anaconda Prompt

Choose the instructions for your operating system.

- [Windows](#)
- [macOS](#)
- [Linux](#)

### Windows

From the Start menu, search for and open "Anaconda Prompt":



## macOS

Open Launchpad, then click the Terminal icon.

## Linux

Open a Terminal window.

### 2. Start Python

At Anaconda Prompt (Terminal on Linux or macOS), type python and press Enter.

The >>> means you are in Python.

### 3. Write a Python program

At the >>>, type print("Hello Anaconda!") and press Enter.

When you press enter, your program runs. The words "Hello Anaconda!" print to the screen. You're programming in Python!

### 4. Exit Python

On Windows press CTRL-Z and press Enter. On macOS or Linux type exit() and press Enter.

**What's next?** (<https://docs.anaconda.com/anaconda/user-guide/getting-started/>)

#### Using Navigator

- [Getting started with Navigator \(10 minutes\)](#)
- [Navigator user guide](#)

#### Using conda

- [Getting started with conda \(20 minutes\)](#)
- [Conda cheat sheet \(pdf\)](#)
- [Conda user guide](#)

#### Links to IDE documentation

- [Eclipse and PyDev](#)
- [IDLE](#)
- [Sublime Text](#)
- [Ninja IDE](#)
- [Python Tools for Visual Studio \(PTVS\)](#)
- [Python for Visual Studio Code](#)
- [Spyder](#)
- [Wing IDE](#)
- [IntelliJ](#)

#### Use pip for Installing

[pip](#) is the recommended installer. Below, we'll cover the most common usage scenarios. For more detail, see the [pip docs](#), which includes a complete [Reference Guide](#).

## Upgrade pip in anaconda python

You can do so from the console editor of Anaconda using the command:

```
conda update pip
```

Or you can check this too. Just open **Anaconda Navigator** then scroll to **Environments** on the left pane. Then **search** for **pip** if there is any update available it will show you in the **version** tab. Then you can just click on to it and update your pip

```
Anaconda Prompt

(base) C:\Users\EP>pip install netaddr
Collecting netaddr
  Downloading https://files.pythonhosted.org/packages/ba/97/ce14451a9fd7bdb5a397abf99b24a1a6bb7a1a440b019bebd2e9a0dbec74/netaddr-0.7.19-py2.py3-none-any.whl (1.6MB)
    100% |#####| 1.6MB 6.6MB/s
distributed 1.21.8 requires msgpack, which is not installed.
Installing collected packages: netaddr
Successfully installed netaddr-0.7.19
You are using pip version 10.0.1, however version 18.0 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

(base) C:\Users\EP>python -m pip install --upgrade pip
Collecting pip
  Downloading https://files.pythonhosted.org/packages/5f/25/e52d3f31441505a5f3af41213346e5b6c221c9e086a166f3703d2ddaf940/pip-18.0-py2.py3-none-any.whl (1.3MB)
    100% |#####| 1.3MB 2.2MB/s
distributed 1.21.8 requires msgpack, which is not installed.
Installing collected packages: pip
  Found existing installation: pip 10.0.1
    Uninstalling pip-10.0.1:
      Successfully uninstalled pip-10.0.1
Successfully installed pip-18.0
```

### Exemplu – Anaconda Prompt python

```
Anaconda Prompt - python

>>> from netaddr import*
>>> ip=IPAddress('172.17.2.45')
>>> ip.version
4
>>>
```

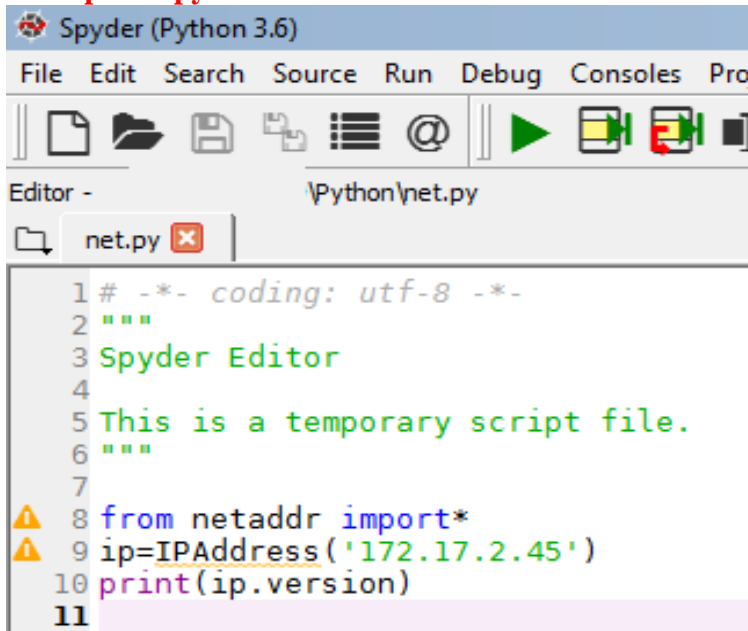
### Anaconda Prompt - spyder

```
Anaconda Prompt - spyder

>>> ^Z

(base) C:\Users\EP>spyder
Warning: QT_DEVICE_PIXEL_RATIO is deprecated. Instead use:
  QT_AUTO_SCREEN_SCALE_FACTOR to enable platform plugin controlled per-screen factors.
  QT_SCREEN_SCALE_FACTORS to set per-screen factors.
  QT_SCALE_FACTOR to set the application global scale factor.
```

### Exemplu - Spyder



```
1 # -*- coding: utf-8 -*-
2 """
3 Spyder Editor
4
5 This is a temporary script file.
6 """
7
8 from netaddr import*
9 ip=IPAddress('172.17.2.45')
10 print(ip.version)
11
```

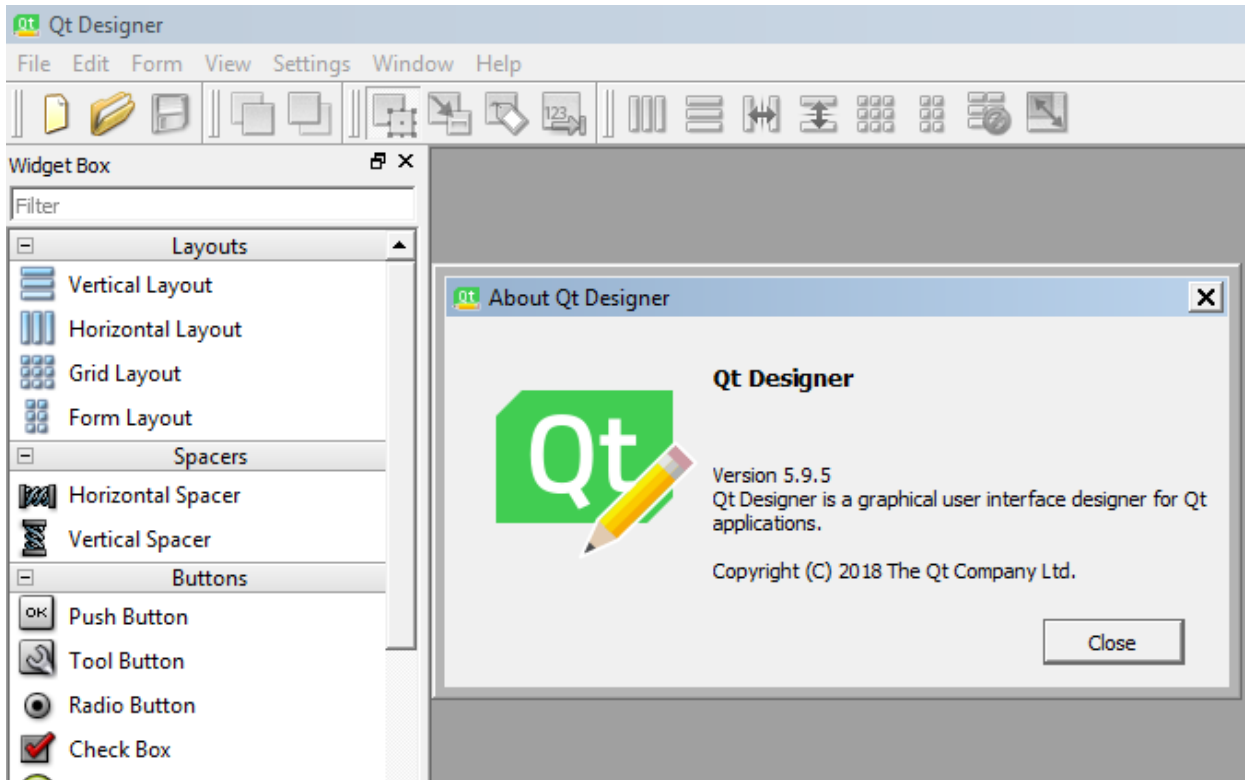
⇒ 4

**Qt Designer** (<http://doc.qt.io/qt-5/qt designer-manual.html>) is the Qt tool for designing and building graphical user interfaces (GUIs) with Qt Widgets. You can compose and customize your windows or dialogs in a what-you-see-is-what-you-get (WYSIWYG) manner, and test them using different styles and resolutions.

### Anaconda Prompt - designer



```
<base> C:\Users\EP>
<base> C:\Users\EP>designer
```



---

## Important differences between Python 2.x and Python 3.x with examples

(<https://www.geeksforgeeks.org/important-differences-between-python-2-x-and-python-3-x-with-examples/>)

- [Division operator](#)
- [print function](#)
- [Unicode](#)
- [xrange](#)
- [Error Handling](#)
- [\\_future\\_ module](#)

## Conclusion (<https://wsvincent.com/python2-vs-python3/>)

While there are several notable differences, migrating an existing project from Python 2 to Python 3 is quite manageable on legacy projects. But if you are learning Python or starting a new project, **you should use Python 3 to take advantage of continued advancements as well as better ongoing support within the broader Python community.**

## References:

Using Python on Windows - <https://docs.python.org/3/using/windows.html>

The Hitchhiker's Guide to Python - <http://docs.python-guide.org/en/latest/intro/learning/>

A Byte of Python - <https://www.gitbook.com/book/swaroopch/byte-of-python/details>

GUI Programming in Python - <https://wiki.python.org/moin/GuiProgramming>

# Spyder is the Scientific PYthon Development EnviRonment

(<https://pythonhosted.org/spyder/>):

- a powerful interactive development environment for the Python language with advanced editing, interactive testing, debugging and introspection features
- and a numerical computing environment thanks to the support of IPython (enhanced interactive Python interpreter) and popular Python libraries such as NumPy (linear algebra), SciPy (signal and image processing) or matplotlib (interactive 2D/3D plotting).

- **SpyderOverview**

<https://www.youtube.com/watch?v=pIQEh2H11s>

- **Basics of SPYDER IDE for Python Programmers**

[https://www.youtube.com/watch?v=a1P\\_9fGrfnU](https://www.youtube.com/watch?v=a1P_9fGrfnU)

- **Developing Inside Spyder**

<https://www.youtube.com/watch?v=HjdK4HS58m4>

- **Python With Spyder 1: First Steps**

<https://www.youtube.com/watch?v=J5GevIHNctM>

- **Python With Spyder 2: Basic Arithmetic and Variable Assignment**

<https://www.youtube.com/watch?v=OPuDjdRjtyY>

- **Python With Spyder 3: Functions and Scoping**

<https://www.youtube.com/watch?v=GT1UfkLIeZ4>

- **Python With Spyder 4: Strings, Indexing, and Slicing**

<https://www.youtube.com/watch?v=OKelK8-NGg8>

- **Python With Spyder 5: Lists Part 1**

<https://www.youtube.com/watch?v=1qeKsuhww8g>

- **Python With Spyder 6: Lists Part 2**

<https://www.youtube.com/watch?v=Ghb4wMW0YX4>

- **Python With Spyder 7: The Idea of Objects**

<https://www.youtube.com/watch?v=G6U5fQ8Siwo>

- **Python With Spyder 8: Python Objects Part 1**

<https://www.youtube.com/watch?v=JJPM7hI4fjE>

- **Python With Spyder 9: Objects Part 2 - Attributes and Methods and a Few Spyder "Tricks"**

<https://www.youtube.com/watch?v=oj4bWBFptt4>

- **Python With Spyder 10: Objects Part 3 - Private Data and Encapsulation**

<https://www.youtube.com/watch?v=VstfQaQjSBc>

- **Python With Spyder 11: Objects Part 4 -- Inheritance**  
<https://www.youtube.com/watch?v=ueDdQ8sx6SM>
- **Python With Spyder 12: Dictionaries**  
<https://www.youtube.com/watch?v=FzzYUbSuOSU>
- **Python With Spyder 13: For Loops**  
<https://www.youtube.com/watch?v=w7sfSkQqdgw>
- **Python With Spyder 14: If Statements**  
<https://www.youtube.com/watch?v=1bDSv18zGTU>