# Python basic

## Install Miniconda instead of Python

Install Miniconda rather than Python because Miniconda includes Python already.

In addition, when you create an environment, it always come with a python interpreter.

## Run miniconda and create/activate a conda environment

To run Miniconda, use anaconda prompt (**yes, anaconda**, not miniconda prompt), **not windows terminal**.

From the anaconda prompt:

Create a conda environment (**remember to indicate which version of python interpreter; otherwise Visual Studio Code does not recognize this conda environment**).

*[from anaconda command prompt]* conda create -n environment\_name **python=3.9**

Put the name of packages and versions at the end to install the packages with the environment:

*[from anaconda command prompt]* conda create -n environment\_name **python=3.9** flask

Activate a conda environment

*[from anaconda command prompt]* conda activate environment\_name

Deactivate a conda environment

*[from anaconda command prompt]* conda deactivate

List conda environment

*[from anaconda command prompt]* conda list

Install a new package (Jupyter Notebook) in the active environment

*[from anaconda command prompt]* conda install jupyter

When Visual Studio Code does not recognize an anaconda environment

Activate the anaconda env and then type

*[from anaconda command prompt]* code

## Visual Studio Code: activate an environment

### Create virtual environment

To create a virtual environment, use the following command, where ".venv" is the name of the environment folder:

# macOS/Linux

# You may need to run sudo apt-get install python3-venv first

python3 -m venv .venv

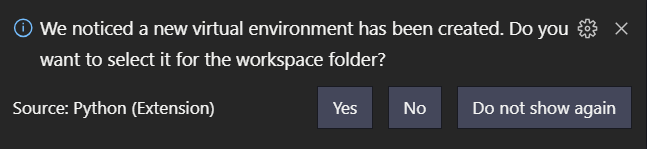
# Windows

# You can also use py -3 -m venv .venv

python -m venv .venv

**Note**: To learn more about the venv module, see [Creation of virtual environments](https://docs.python.org/3/library/venv.html) on Python.org.

When you create a new virtual environment, a prompt will be displayed to allow you to select it for the workspace.



### Create conda environment

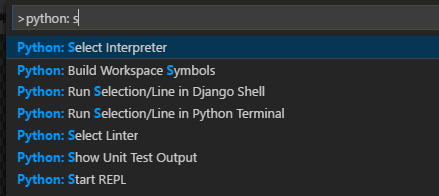
The Python extension automatically detects existing conda environments **provided that the environment contains a Python interpreter**. For example, the following command creates a conda environment with the Python 3.4 interpreter and several libraries, which VS Code then shows in the list of available interpreters:

conda create -n env-01 python=3.4 scipy=0.15.0 astroid babel

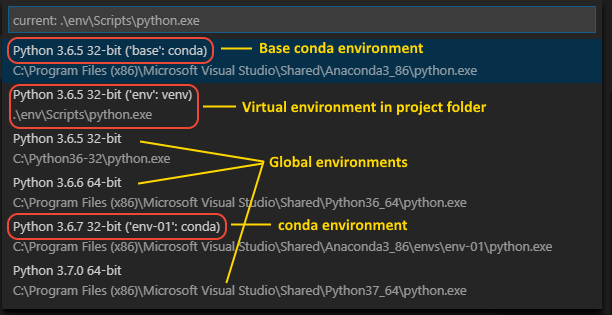
**In contrast, if you fail to specify an interpreter, as with conda create --name env-00, the environment won't appear in the list.**

### Select and activate an environment (conda or virtual)

By default, the Python extension looks for and uses the first Python interpreter it finds in the system path. To select a specific environment, use the **Python: Select Interpreter** command from the **Command Palette** (Ctrl+Shift+P).



The **Python: Select Interpreter** command displays a list of available global environments, conda environments, and virtual environments.



## Python interpreter

Use ‘**where python’** to find out the location of python interprerter

In Windows:

C:\Users\tranh\AppData\Local\Programs\Python\Python39\python.exe

C:\Users\tranh\AppData\Local\Microsoft\WindowsApps\python.exe

Note that an environment have to contain an interpreter otherwise Visual studio code does not recognize it.

## Global/virtual/conda environment

By default, any Python interpreter that you've installed runs in its own **global environment**, which is not specific to any one project. For example, if you just run python (Windows) or python3 (macOS/Linux) at a new command prompt, you're running in that interpreter's global environment. Accordingly, any packages that you install or uninstall affect the global environment and all programs that you run within that context.

For this reason, developers often create a **virtual environment** for a project. A virtual environment is a subfolder in a project that contains a copy of a specific interpreter. When you activate the virtual environment, any packages you install are installed only in that environment's subfolder.

A conda environment is a Python environment that's managed using the conda package manager (see [Getting started with conda](https://conda.io/projects/conda/en/latest/user-guide/getting-started.html) (conda.io)). Conda works well to create environments with interrelated dependencies as well as binary packages. Unlike virtual environments, which are scoped to a project, conda environments are available globally on any given computer. This availability makes it easy to configure several distinct conda environments and then choose the appropriate one for any given project.

## Virtual Environment

A virtual environment is a semi-isolated Python environment that allows packages to be installed for use by a particular application, rather than being installed system wide.

venv is the standard tool for creating virtual environments, and has been part of Python since Python 3.3. Starting with Python 3.4, it defaults to installing pip into all created virtual environments.

To create a virtual environment, go to your project’s directory and run venv.

py -m venv env

Before you can start installing or using packages in your virtual environment you’ll need to *activate* it.

.\env\Scripts\activate

## pip

pip (preferred installer program) is a program that manages packages for python.

C:\Users\tranh\AppData\Local\Programs\Python\Python39\Scripts\pip.exe

pip is included in the python standard package.

Install a package:

py -m pip install package\_name

Uninstall a package:

py -m pip uninstall package\_name

## Conda/miniconda/anaconda

conda is both a command line tool, and a python package.

Miniconda installer = Python + conda

Anaconda installer = Python + conda + meta package anaconda

meta Python pkg anaconda = about 160 Python pkgs for daily use in data science

Anaconda installer = Miniconda installer + conda install anaconda

## Python’s Interactive mode

Use **python** to enter and **quit()** to quit the Python interactive mode

## Command for python in Windows: py, in Linux: python

In Windows, it’s **py**, not python.