# Python

Python is a general-purpose programming language, aimed at striking a balance between performance and code interpretability. Python has some high-performance numerical libraries, but is not as performant as languages like C++. However, it is widely used, the code can be easy to read, and it has a large collection of packages.

## Installation

You should install Python 3 (preferably version 3.6 or greater) --- not Python 2. Rather than using the default version installed on whatever system you're using, you should use conda. This allows you to manage multiple installations, as different packages might not be compatible with the same Python version. conda is great at automatically upgrading or downgrading the version as needed.

Some packages may not be available through conda, but in that case, use `pip` to install them from within your conda environment.

## Usage

A good place to start if you don't know any Python is the Python Fundamentals chapter from Earth and Environmental Data Science. There are also a number of other resources available online, and many specific questions can be or already are answered on Stack Overflow.

Python is a dynamically-typed language, meaning that it automatically assigns types to variables and functions. This means that, by default, it lacks type checking, which can result in bugs. Dynamic typing is also one reason why Python is more user-friendly than a statically-typed language like C++, but is also slower.

## Development Environments

There are a number of good IDEs for Python. A few options:

- Spyder: Lightweight, free, open-source, natively-coded in Python.
- PyCharm: Fully-functional IDE, though it costs money. We can get you a license if this is the way you want to go.

- Atom: Text editor that can be turned into a Python IDE using several packages.

## Recommended Packages

You'll usually want to include the following in any research conda environment:

- `numpy`: high-performance library for numerical computing.
- `scipy`: contains functionality for a wide variety of scientific computing tasks, including optimization and statistics.
- `matplotlib`: library for visualization and plotting.

Other packages may make sense depending on your workflow and tasks:

- `pandas`: data analysis and manipulation.
- `xarray`: allows for structured datasets and interfacing with netCDF.
- `seaborn`: adds some additional plotting functionality on top of `matplotlib`, and integrates well with `pandas`.
- `pyMC3`: probabilistic programming using a variety of methods, including Hamiltonian Monte Carlo and Metropolis-Hastings.
- `pyStan`: Similar to `pyMC3`, but uses the Stan backend.
- `TensorFlow`: machine learning tools and libraries.
- `mpi4py`: bindings for parallelization across multiple processors and compute nodes.
- `mypy`: optional static type-checking, can help reduce errors by clarifying expected inputs and outputs and make it easier for others to work with your code.
- `black`: automates code formatting.

## Learn More

- Beyond PEP 8 -- Best Practices for Beautiful Intelligible Code by Raymond Hettinger from PyCon 2015.
- Livecoding Madness - Let's Build a Deep Learning Library by Joel Grus
- Software Testing and Testing Automation with Python
- Python posts from Water Programming.

Last update: June 26, 2023