Python Crash Course

For Data Scientist

Notice

Please note, this is not meant to be a comprehensive overview of Python or programming in general.

Outline – Part 1

- Introduction
- Data types
  - Numbers
  - String
  - Printing
  - Lists
  - Dictionaries
  - Booleans
  - Tuples
  - Sets
- Python Comparison Operators
- Python statements
  - if, elif, else Statements
  - for Loops
  - while Loops
- Methods and functions
  - lambda expressions
  - map and filter
- Error handling
- Modules and Packages
- Object Oriented Programming

Outline – Part 2

- **Most popular data Python Data Science Libraries**
  - **NumPy**
  - **SciPy**
  - **Pandas**
  - **MetplotLab**
  - **Seaborn**
  - **Scikit-learn**
  - **And much more (Dask, …)**

Outline – Part 3

- **Data Science project**
  - **Cross-industry standard process for data mining**
  - **Exploratory data analysis**
  - **Data preparation**
    - **Missing data imputation**
    - **Data Encoding and scaling**
    - **Feature selection**
  - **Model construction**
    - **Cross-validation**
    - **Grid-search**
    - **Ensemble methods**
  - **Model deployment**

Python version

Python 2 or Python 3 ? Definitely 3

- https://www.python.org/
- https://pythonclock.org/

Python version

- Why Python ?

The rich ecosystem of libraries and tooling, and the convenience of the language itself, make Python an excellent choice.

- Many distributions of Python, such as
  - WinPython,
  - ActivePython,
  - **Anaconda**,
  - Enthought Canopy,
  - Python(x,y),
  - Pyzo

Python version

- Why Python ?

For this course, let's install Anaconda for python 3

  - https://www.anaconda.com/download

Python version

- Virtual Environment
  - Virtual Environments allow you to set up virtual installations of Python and libraries on your computer
  - You can have multiple versions of Python or libraries and easily activate or deactivate these environments
  - Virtual environment = a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages

Python version

- Why Virtual Environment?
  - Sometimes you'll want to program in different versions of a library
  - For example:
    - You develop a program with SciKit-Learn 0.17
    - SciKit-Learn 0.18 is released
    - You want to explore 0.18 but don't want you old code to break

Python version

- Virtual Environment

Several ways to create a virtual environment

- Using venv module of python (python-venv)
  - python3 -m venv tutorial-env
  - source tutorial-env/bin/activate
  - pip3 search astronomy
  - pip3 install novas or pip install novas=2.1.0
  - source deactivate
  - Virtual Environments allow you to set up virtual installations of Python and libraries on your computer
- Using Virtualenv Environment
- Using Pipenv Environment

Python version

- Virtual Environment

Several ways to create a virtual environment

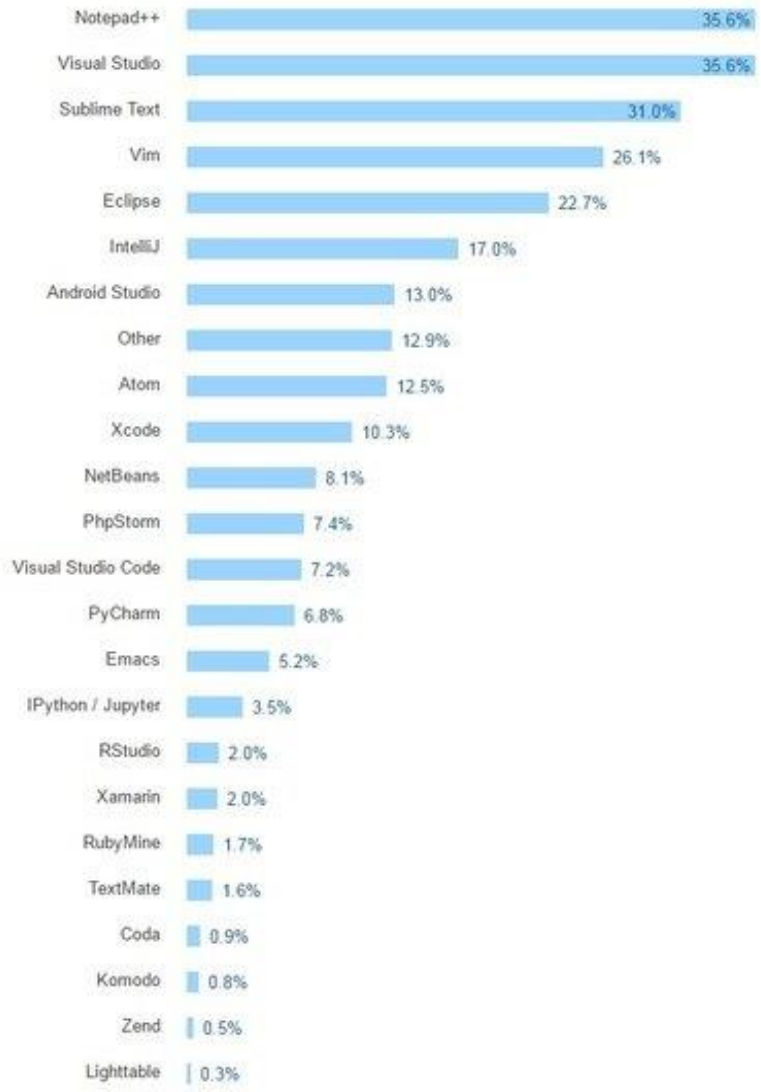
- Using Anaconda (Conda) a Environment

https://conda.io/docs/user-guide/tasks/manage-environments.html

  - conda create --name myenv
  - conda install numpy
  - conda install anaconda
  - source deactivate

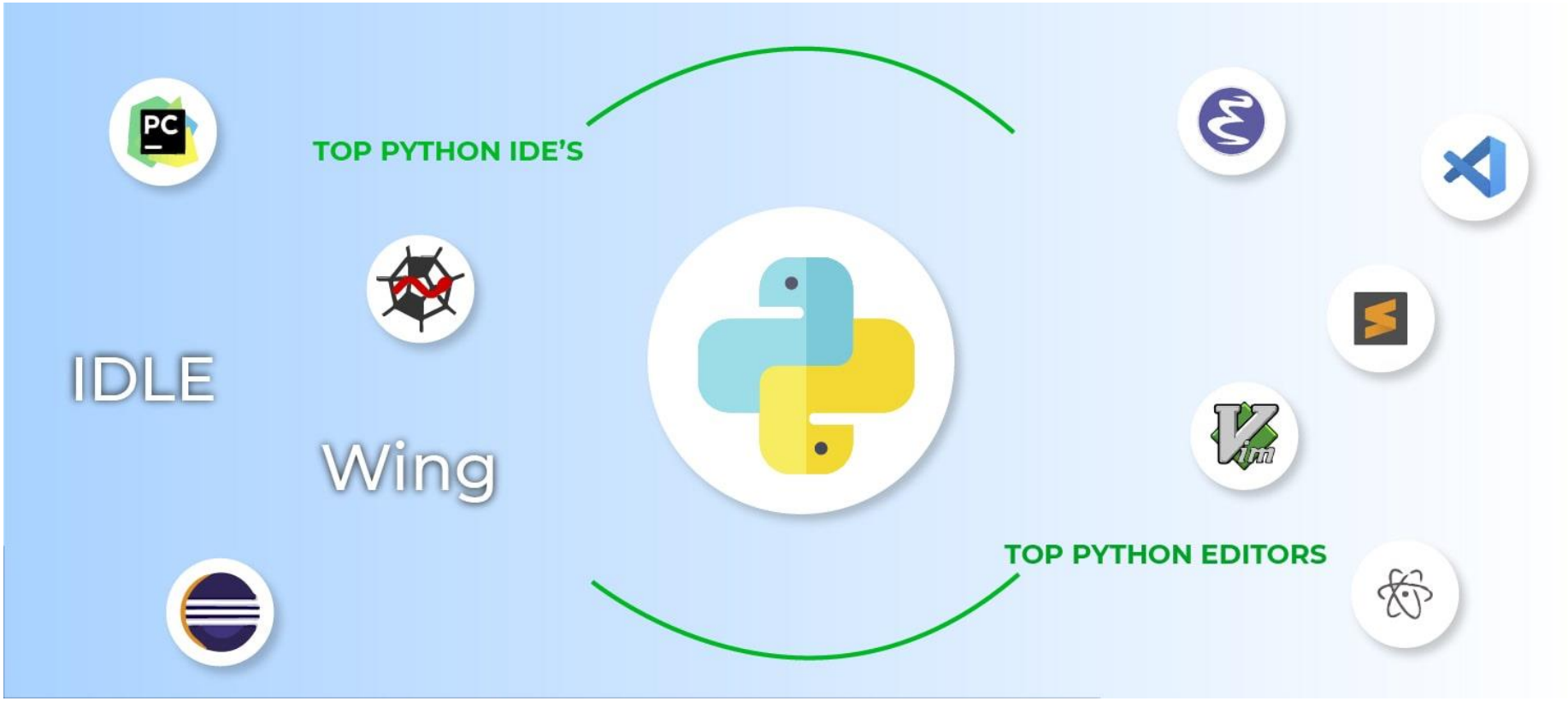
Python IDE

- IDEs versus Text Editors



Python IDE

- Python IDEs for python



Python IDE

- Python IDEs for Data Science
  - Spyder
  - **PyCharm**
  - Rodeo
  - Atom
  - **Jupyter Notebook**
  - **Jupyter Lab**
  - **Visual Studio Code**
  - **PyDev**

Python IDE

- Python IDEs for Data Science

  – http://www.jupyter.org/

  and

  – try it with python in http://jupyter.org/try
    - Code Cell
    - Markdaown Cell

Python IDE

- Python IDEs for Data Science

  Now, let's launch Jupyter Notebook in your PC

  - Open a terminal window or **anaconda prompt**
  - Enter the startup folder with cd
  - Create and/or activate your virtual environment
  - Type **jupyter notebook** to launch the Jupyter Notebook App
    - If note installed, use **conda install -c anaconda jupyter** (or using **pip3) using**
    - The notebook interface will appear in a new browser window or tab.
  - Ask for help if needed

Python

- Overview
- A hobby project by Guido Van Rossum and was first released in 1991.
- A general purpose language
- Guido compares Python to languages like Java or Swift and says that while the latter two are a great choice for software developers — people whose day job is programming, but

Python was made for people whose day job has nothing to do with software development but they code mainly to handle data.

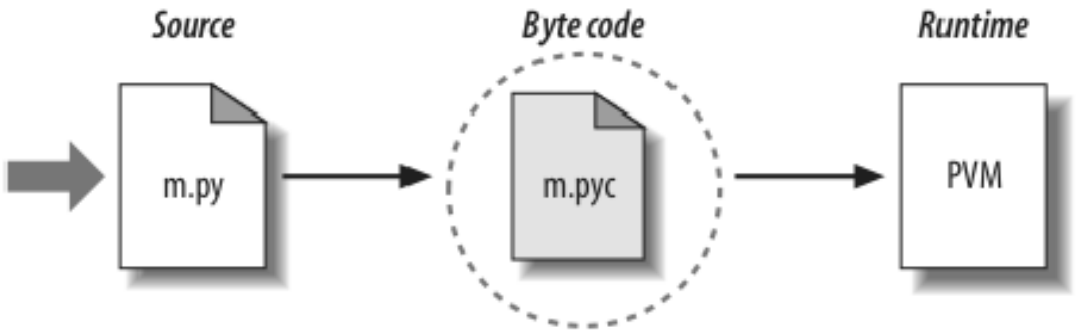
- Python is an interpreted, high-level, general-purpose programming language. It is dynamically typed and garbage-collected.

Python

- Python characteristics
- Simple
  - Python is a simple and minimalistic language in nature
  - Reading a good python program should be like reading English
  - Its Pseudo-code nature allows one to concentrate on the problem rather than the language
- Easy to Learn
- Free & Open source
  - Freely distributed and Open source
  - Maintained by the Python community
- High Level Language –memory management
- Portable – *runs on anything c code will (c compiler)

Python

- Compilation vs Interpretation
- Compilation = translating your human understandable code to machine understandable code. Machine code is the base level form of instructions that can be directly executed by the CPU.
- Interpretation = executing a program by reading the source code a line at a time, and doing what it says (shell like).
- Python is an interpreted language and not a compiled one, although compilation is a step. Python code, written in **.py** file is first compiled to what is called *bytecode* which is stored with a **.pyc** or **.pyo** format.

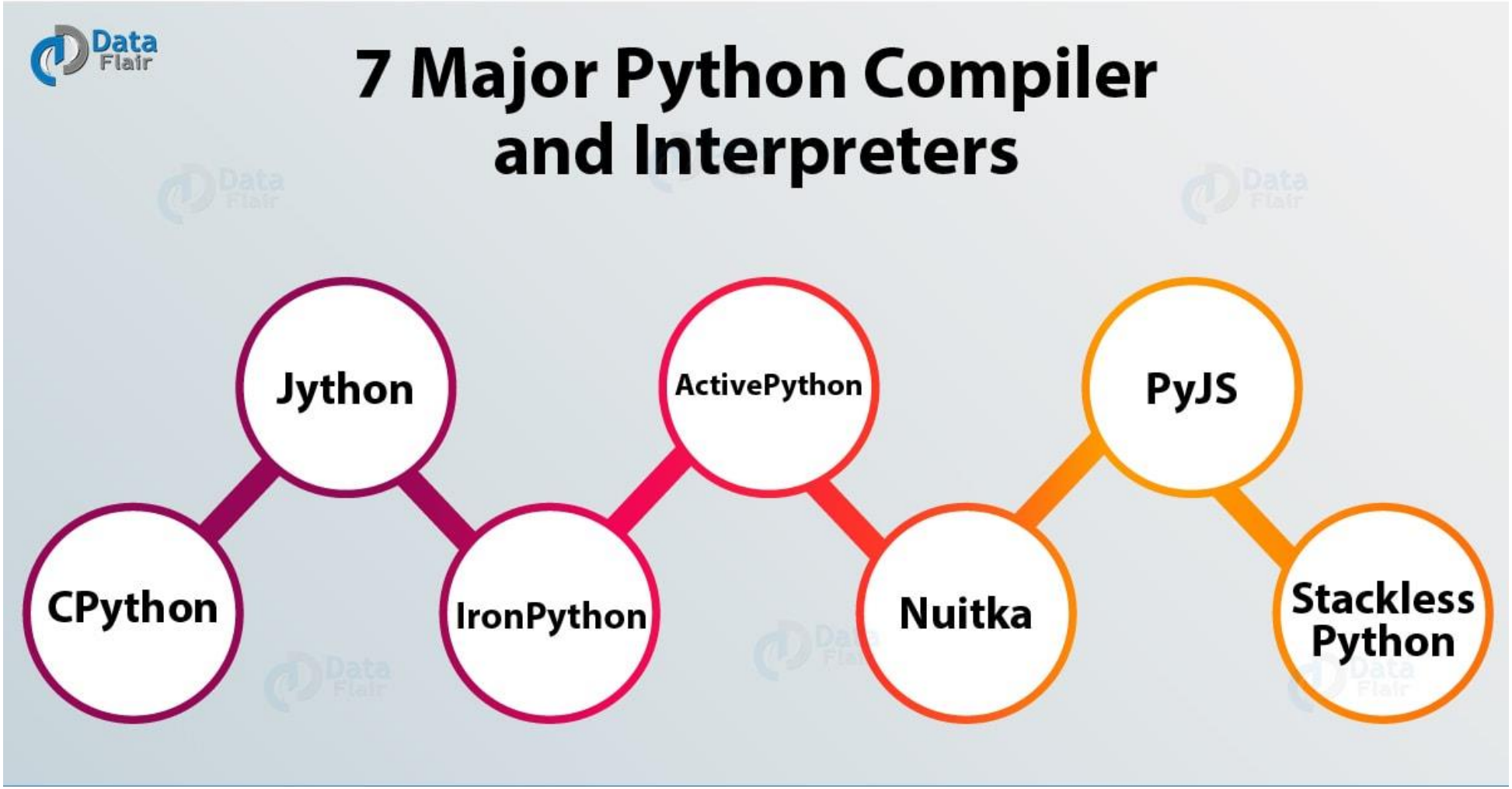


Python IDE

- Compilation vs Interpretation
- Platform-independent: you can just copy over your code to another system and it will auto-magically work (with python platform):.
- Dynamic typing:
  - In static-typed languages like C++, you have to declare the variable type and any discrepancy like adding a string and an integer is checked during compile time.
  - In strongly typed languages like Python, it is the job of the interpreter to check the validity of the variable types and operations performed.
  - Dynamic typing provides a lot of freedom, but simultaneously it makes your code risky and sometimes difficult to debug
- Garbage Collector: automatically frees up space without you doing anything.

Python IDE

- Python characteristics
- Interpreted (mixted)

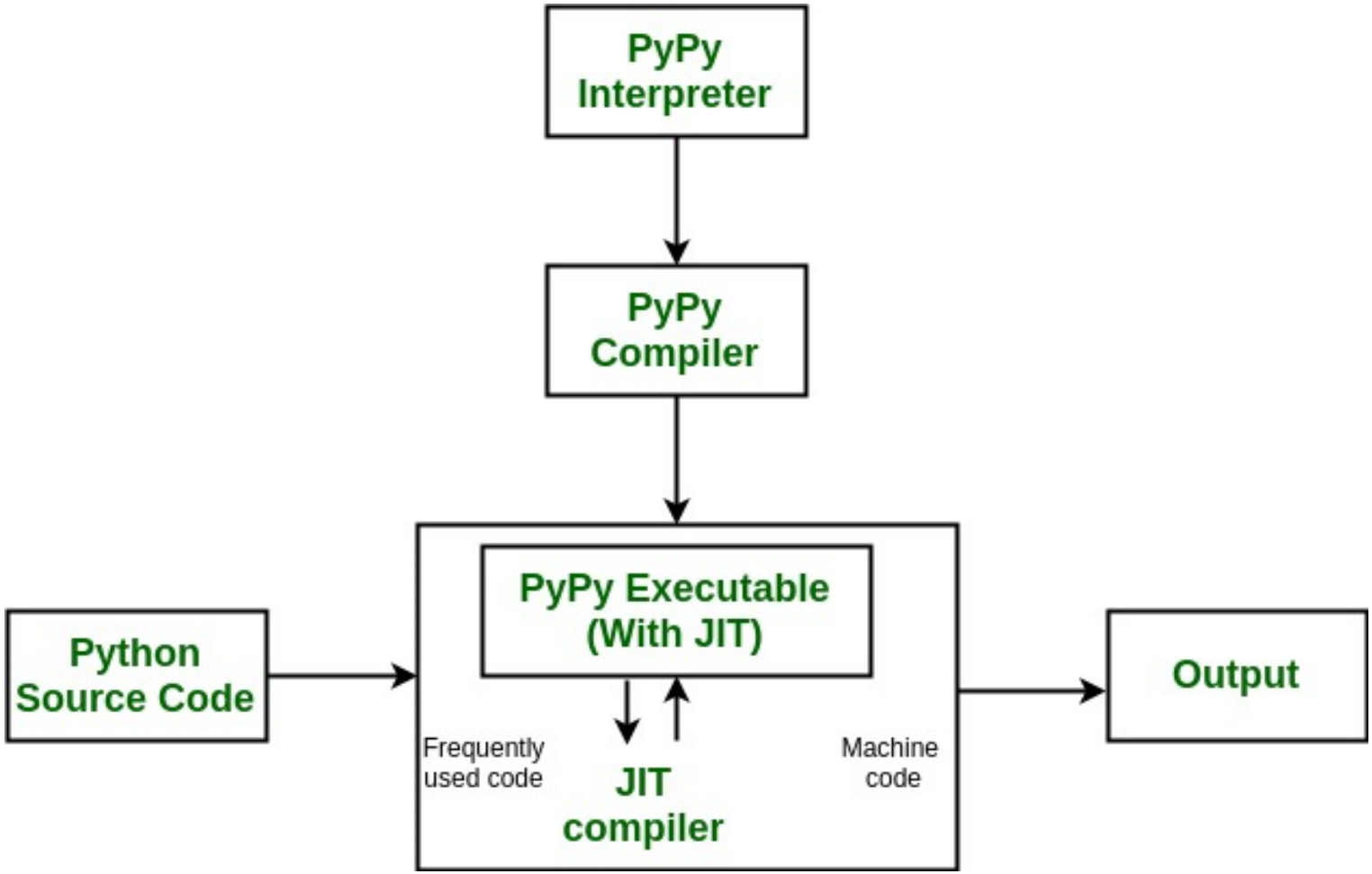


Python IDE

- Python characteristics
  - PyPy = an alternative implementation of Python which often runs faster than the standard implementation of Python, CPython.
  - The reason PyPy is a just-in-time compiler, while CPython is an interpreter. Most Python code runs well on PyPy, except for code that relies on CPython extensions, which either does not work or incurs some overhead when run in PyPy.

Python IDE

- Python characteristics



Python IDE

- Python characteristics

Numba: Numba is an open source JIT compiler that translates a subset of Python and NumPy code into fast machine code.

http://numba.pydata.org/

Python IDE

- Python characteristics
- Object-Oriented
  - Simple and additionally supports procedural programming
- Extensible – easily import other code
- Embeddable –easily place your code in non-python programs
- Extensive libraries
  - (i.e. reg. expressions, doc generation, CGI, ftp, web browsers, ZIP, WAV, cryptography, etc…) (wxPython, Twisted, Python Imaging library)