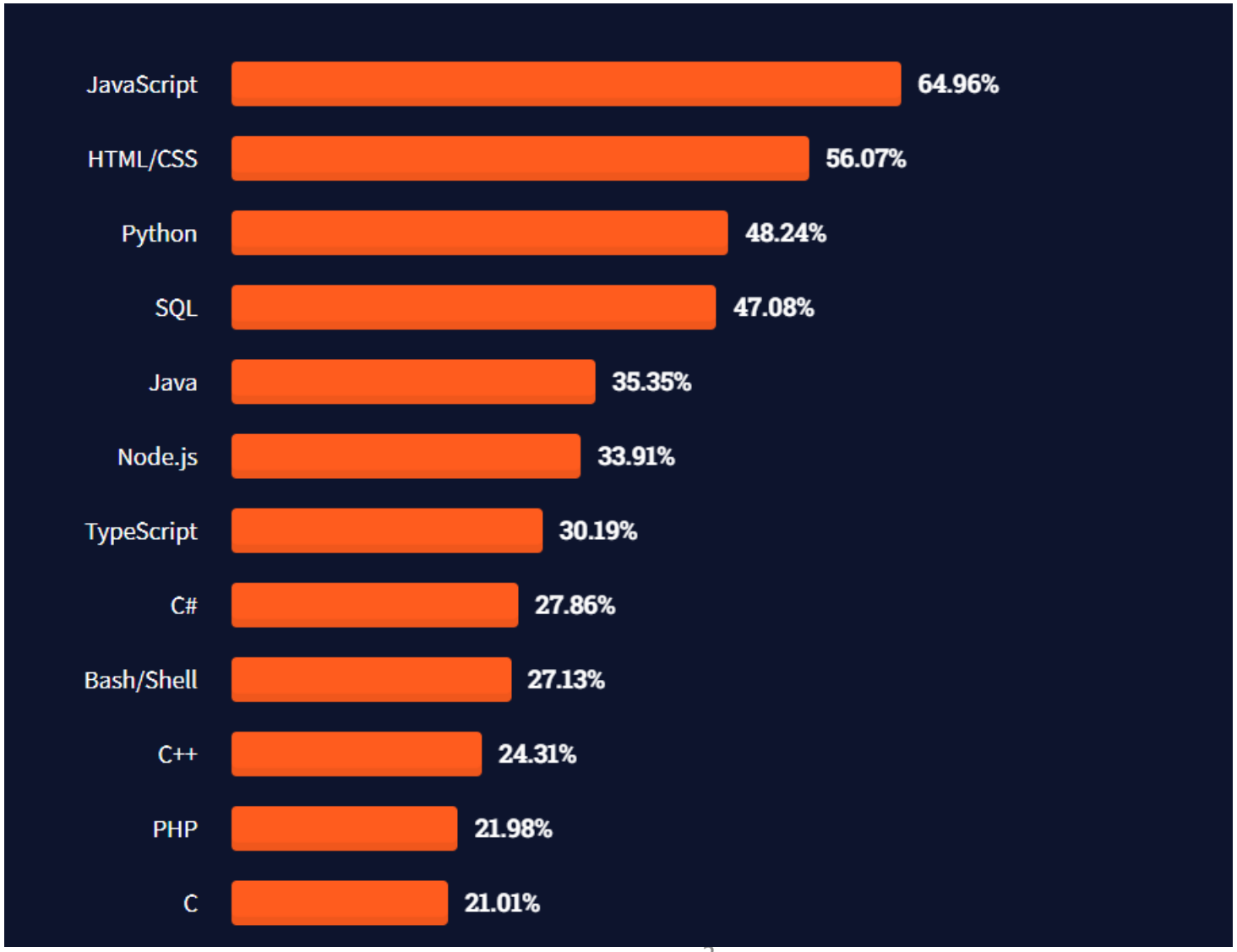
Python overview

Intro to Python

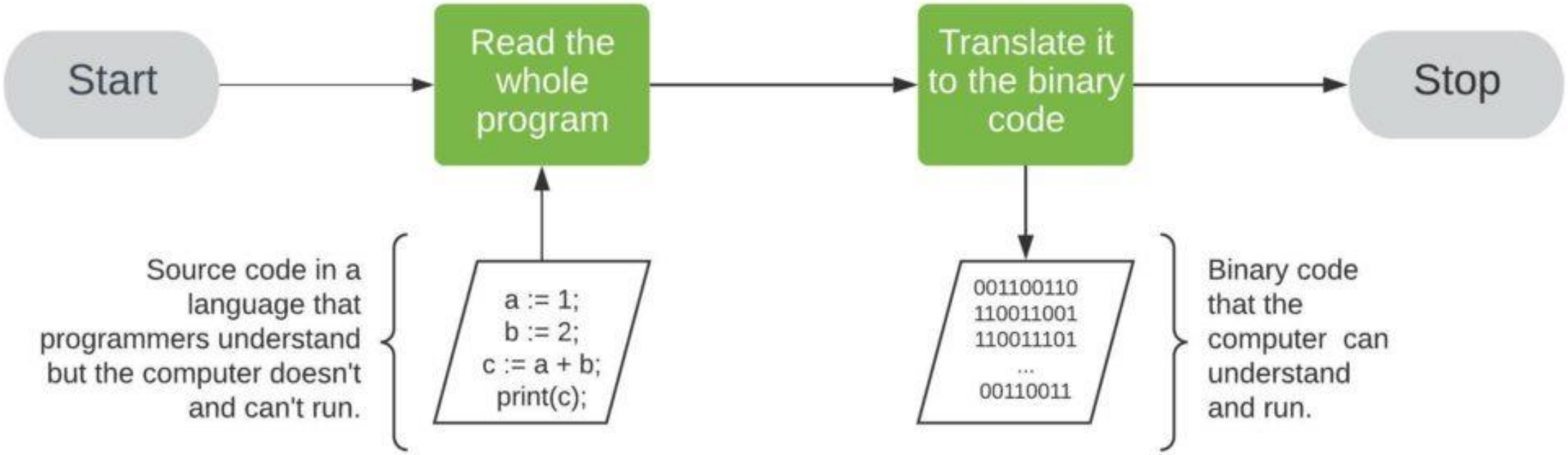
Stack Overflow developer survey 2022



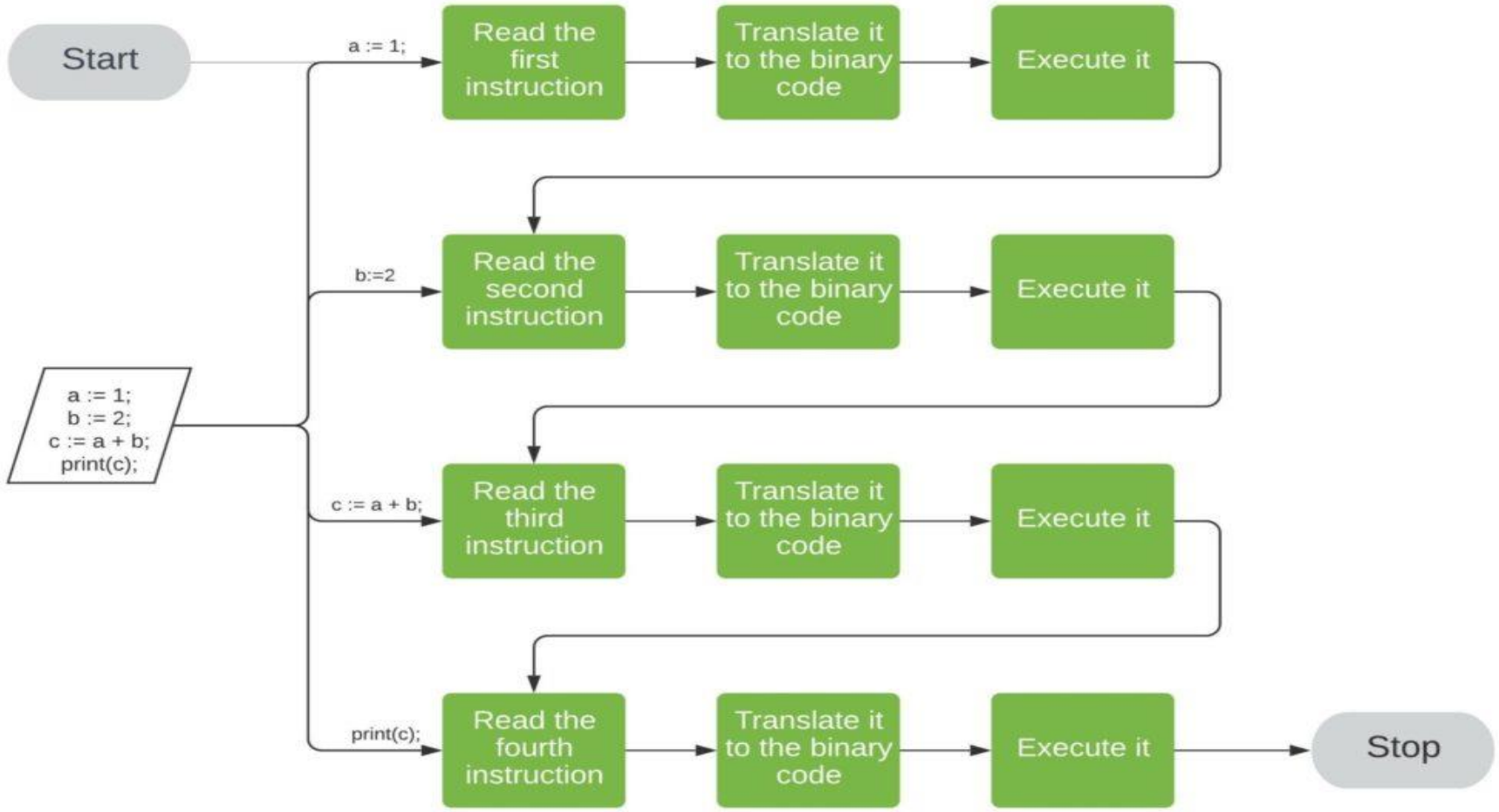
Key features of Python

- Open source and multi-platform
  - Runs on Windows, Mac, Linux, Raspberry Pi
- General purpose
  - Used in various domains: Data science, machine learning, AI, Automation, Web applications, etc
- Interpreted
  - Python interpreter converts the source code, line by line, into native machine code of the platform
  - Interpretation is slower to execute compared to compilation, but helps portability

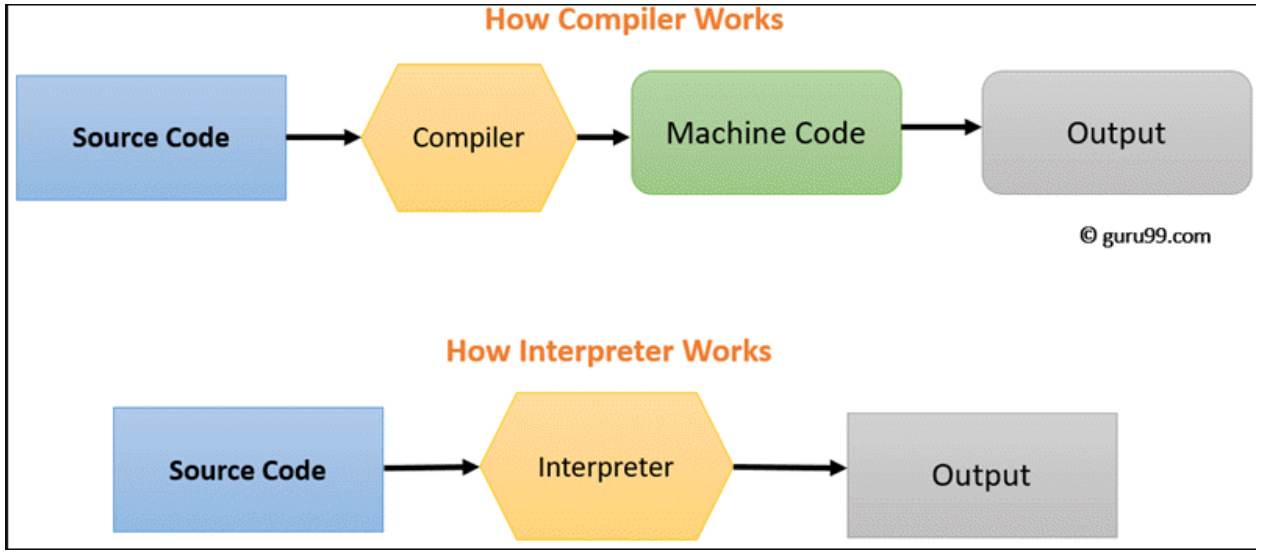
Compiled program

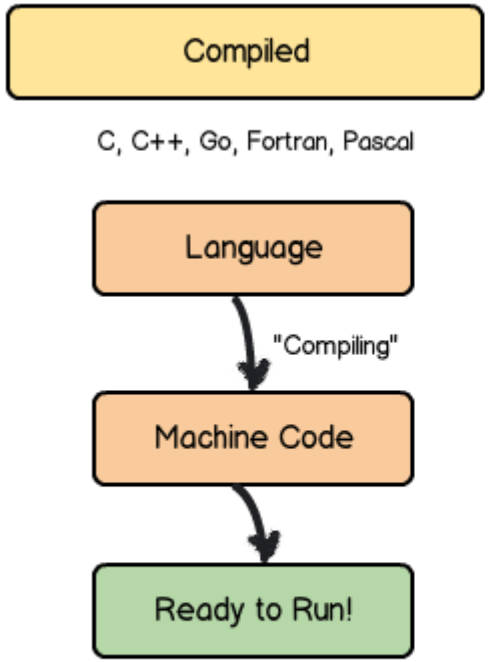


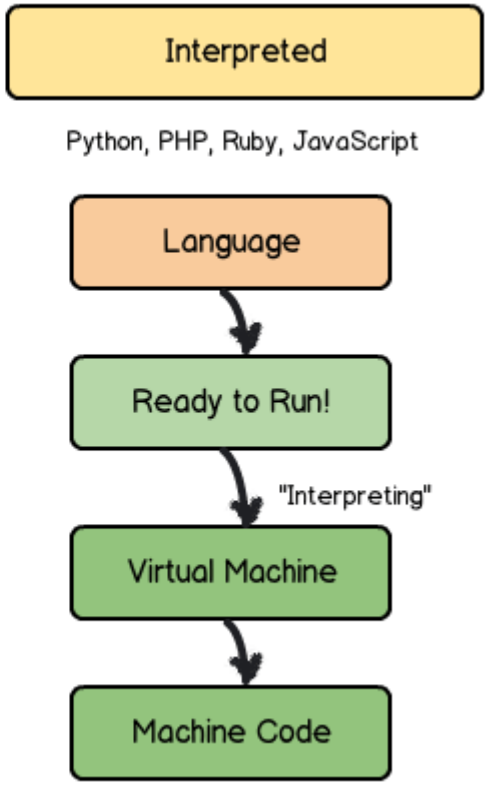
Interpreted program



Compiled vs interpreted







Key features of Python

- Dynamic typing
  - Variables can have values of different types assigned to them throughout their lifetime
- Designed for simplicity and conciseness
  - Accomplishes same functionality in less lines of code (for e.g. compared to languages like Java)
- Large collection of open source libraries
  - Provides a wide range of functionalities that simplify program construction

Python history

- created by Guido van Rossum
  - scripting language that is more user friendly than C programs or shell scripts
  - name derived from BBC comedy sketch Monty Python's Flying Circus
- first released 1991
  - initial development completely guided by Rossum
  - became Python's Benevolent Dictator for Life (BDFL) until 2018

Guido van Rossum (BDFL)

Python versions

2021 3.10.0

2008 3.0

2000 2.0

1994 1.0

1991 0.9.0

python SOFTWARE FOUNDATION

Donate | Search | GO | Socialize

About | Sponsorship | Membership | Donate | Volunteer | Grants | PyCon US | News & Community

The Python Software Foundation is an organization devoted to advancing open source technology related to the Python programming language.

We Support The Python Community through...

Grants

Infrastructure

PyCon US

Python Enhancement Proposal (PEP)

- design document targeting the Python community
  - describing a new feature for Python or its processes or environment
- the primary mechanisms for proposing major new features
  - for collecting community input on an issue
  - documenting the design decisions that have gone into Python.
- Typical primary audience for PEPs
  - core developers of the CPython reference interpreter
  - developers of other implementations of the Python language specification
  - Steering Council - make final decision to accept or reject

THE ZEN OF PYTHON

The Zen of python was written by Tim peters
Infographics by Nagaraj Bhat
Twitter : @nagarajbhat92

1. BEAUTIFUL IS BETTER THAN UGLY.
2. EXPLICIT IS BETTER THAN IMPLICIT.
3. SIMPLE IS BETTER THAN COMPLEX.
4. COMPLEX IS BETTER THAN COMMPLICATED.
5. FLAT IS BETTER THAN NESTED
6. SPARSE IS BETTER THAN DENSE
7. READIBILITY COUNTS
8. SPECIAL CASES AREN'T SPECIAL ENOUGH TO BREAK THE RULES
9. ALTHOUGH PRACTICALITY BEATS PURITY

10. ERRORS SHOULD NEVER PASS SILENTLY.
11. UNLESS EXPLICITLY SILENCED.
12. IN THE FACE OF AMBIGUITY REFUSE THE TEMPTATION TO GUESS.
13. THERE SHOULD BE ONE-- AND PREFERABLE ONE --OBVIOUS WAT TO DO IT.
14. ALTHOUGH THAT MAY NEVER BE OBVIOUS AT FIRST UNLESS YOU ARE DUTCH.
15. NOW IS BETTER THAN NEVER.
16. ALTHOUGH NEVER IS OFTEN BETTER THAN *RIGHT* NOW.
17. IF THE IMPLEMENTATION IS HARD TO EXPLAIN, IT'S A BAD IDEA.
18. IF THE IMPLEMENTATION IS EASY TO EXPLAIN, IT MAY BE A GOOD IDEA.
19. NAMESPACES ARE HONKING GREAT IDEA -- LETS DO MORE OF THOSE!

Python for data science

- The most popular language for data science
- Massive collection of libraries that can be used for all aspects of the data science workflow
  - Data cleansing, analytics, visualization over large datasets (structured, semi-structured, unstructured)
  - Machine learning / Deep learning for complex predictive / prescriptive analytics
- Can be easily integrated with other libraries to incorporate or extend data science functionality

Key Python data science libraries

- Pandas
  - Most basic data science library that provides all tools to work with raw data of different formats
  - Data cleansing, loading, visualization, statistical analysis
- Numpy
  - High performance library for implementing complex array and matrix operations
  - Fundamental package for scientific computing
  - Provides basic structure for Pandas

Key Python data science libraries

- Matplotlib / Seaborn
  - Used specifically for visualizing data (graphs, charts, etc)
  - Can create simple to complex data visualization (animated and interactive)
- Scikit learn
  - Provides implementation for all the core machine learning algorithms and models (supervised and unsupervised learning)
  - Builds on NumPy, Pandas and SciPy

Key Python data science libraries

- Tensorflow
  - Developed by Google for deep learning models
  - well-documented framework and particularly well suited for deploying trained models easily into production
- Pytorch
  - Developed by Facebook for deep learning models
  - Popular for computer vision and natural language processing

Key Python data science tools

- Jupyter Notebook / Lab
  - tool for interactively developing and presenting data science projects
  - combines visualizations, narrative text, maachine learning library related code
  - essential part of the Python / R related data science workflow

Other areas of application

- Web application development
  - Django, Flask
- GUI applications for desktop
  - tkinter, pyqt
- Web Scraping
  - process of collecting and parsing raw data from the Web with an automated tool (crawler)
  - BeautifulSoup, Scrapy, Requests

Other areas of application

- Speed up and automate workflow
  - Automate general file / directory operations
  - Used in DevOps life cycle in CI/CD pipeline
  - Ansible, DockerCompose
- Isolating Development Environment
  - Complex software development requires isolation of packages, libraries, and Python versions in per-project virtual environments.
  - conda, pip, venv, pipenv

Python source code

- source code files have extension .py
  - sometimes .pyw on Windows
- Script
  - source code file intended to be directly executed by user (i.e. top level program)
- Module
  - source code file that is intended to be imported and used from another script (or even module)

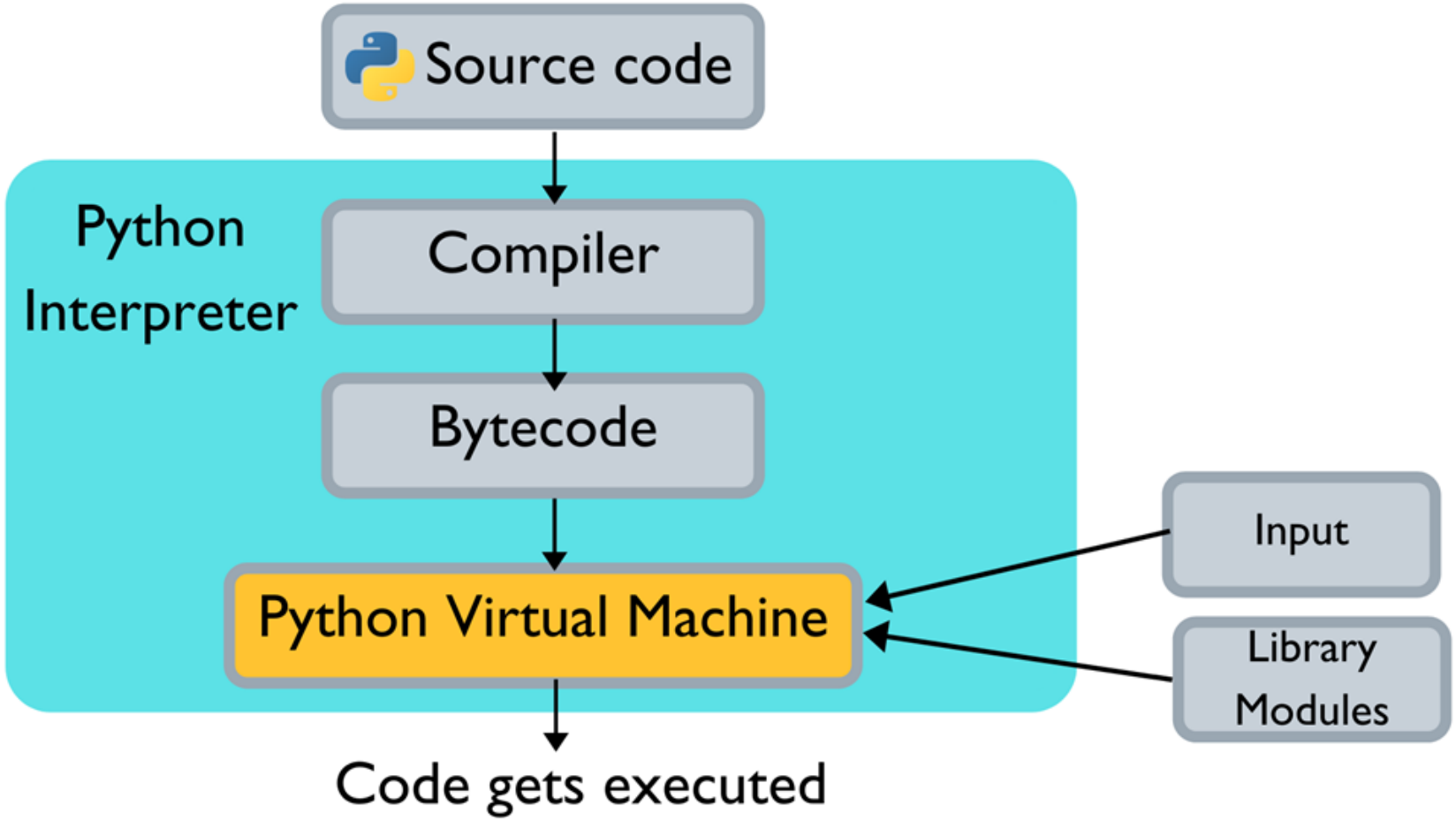
Python interpreter

- The interpreter can run Python code either
  - as a script or module
  - or as statements in an interactive session in the standard shell (REPL shell)
- Depending on the Python implementation, interpreter can be written in:
  - C (like CPython) which is the core / reference implementation of the language
  - Java (like Jython)
  - .NET / C# (like IronPython)

Python interpreter and PVM

- Compiles source code of modules to an intermediate format known as bytecode
  - stored as *.pyc or *.pyo files
  - this is a set of instructions that are platform-independent
  - generating bytecode optimizes code execution by allowing the interpreter to skip it the next time
- executed by Python Virtual Machine (PVM)
  - this is part of the interpreter that translates bytecode to native machine code

Python interpreter and PVM



Python standard / interactive shell

- Also called REPL (Read-Eval-Print Loop) shell
  - provides a CLI that allows users to type in commands directly for the interpreter to execute in an interactive manner
- Performs the following steps:
  - Reading user input, which consists of standard expressions and statements
  - Evaluating these expressions and statements, which generates a result
  - Printing any output so that you can check the results and get immediate feedback
  - Looping back to initial step to continue interaction

Uses of Python interactive shell

- Quickly evaluate code snippets to test out ideas and implementation
- Edit and refactor code for later use in script mode
- Perform code and type introspection
- Get interactive help on how to use the language
- Run basic code debugging
- Explore standard library modules

Common Python IDEs

- IDLE (Integrated Development and Learning Environment)
  - default editor that accompanies standalone Python installations (python.org)
  - open source, suitable for beginner level developers
- PyCharm
  - created by JetBrains on freemium model
  - suitable for professional developers and facilitates the development of large Python projects

Common Python IDEs

- Jupyter
  - open source and free, widely used in data science
  - supports interactive and live code sharing and visualization
  - Integration of data science libraries like NumPy, Pandas, and Matplotlib
- Spyder
  - open-source, most commonly used for scientific development and comes bundled with Anaconda distribution
  - Integration of data science libraries like NumPy, Pandas, and Matplotlib

Common Python IDEs

- Visual Studio Code
  - open-source and free IDE created by Microsoft
  - very popular for other languages such as HTML, JavaScript.
  - lightweight but with powerful features provided by some of the paid IDEs

Which IDE to use ?

- Level of Knowledge
  - Beginner - IDLE, Thonny
  - Intermediate - PyCharm, VS Code,
- Application area
  - Data science, machine learning - Spyder, Jupyter, Pycharm Professional
  - Web full stack development - VS Code, Pycharm Professional
  - Automation scripting - VS Code, Pycharm Community

Anaconda distribution

- distribution of the Python and R programming languages
  - includes specifically all the major libraries / packages related to scientific computing, data science, machine learning applications, large-scale data processing, etc.
- Incorporates the conda tool
  - this simplifies package management, environment management and deployment for many packages and multiple versions of Python
  - There is a public package service (anaconda.org) that hosts packages that can be managed by conda

Anaconda Distribution

Everything you need to get started in data science on your workstation.

- Free distribution install
- Thousands of the most fundamental DS, AI, and ML packages
- Manage packages and environments from desktop application
- Deploy across hardware and software platforms

Anaconda Repository

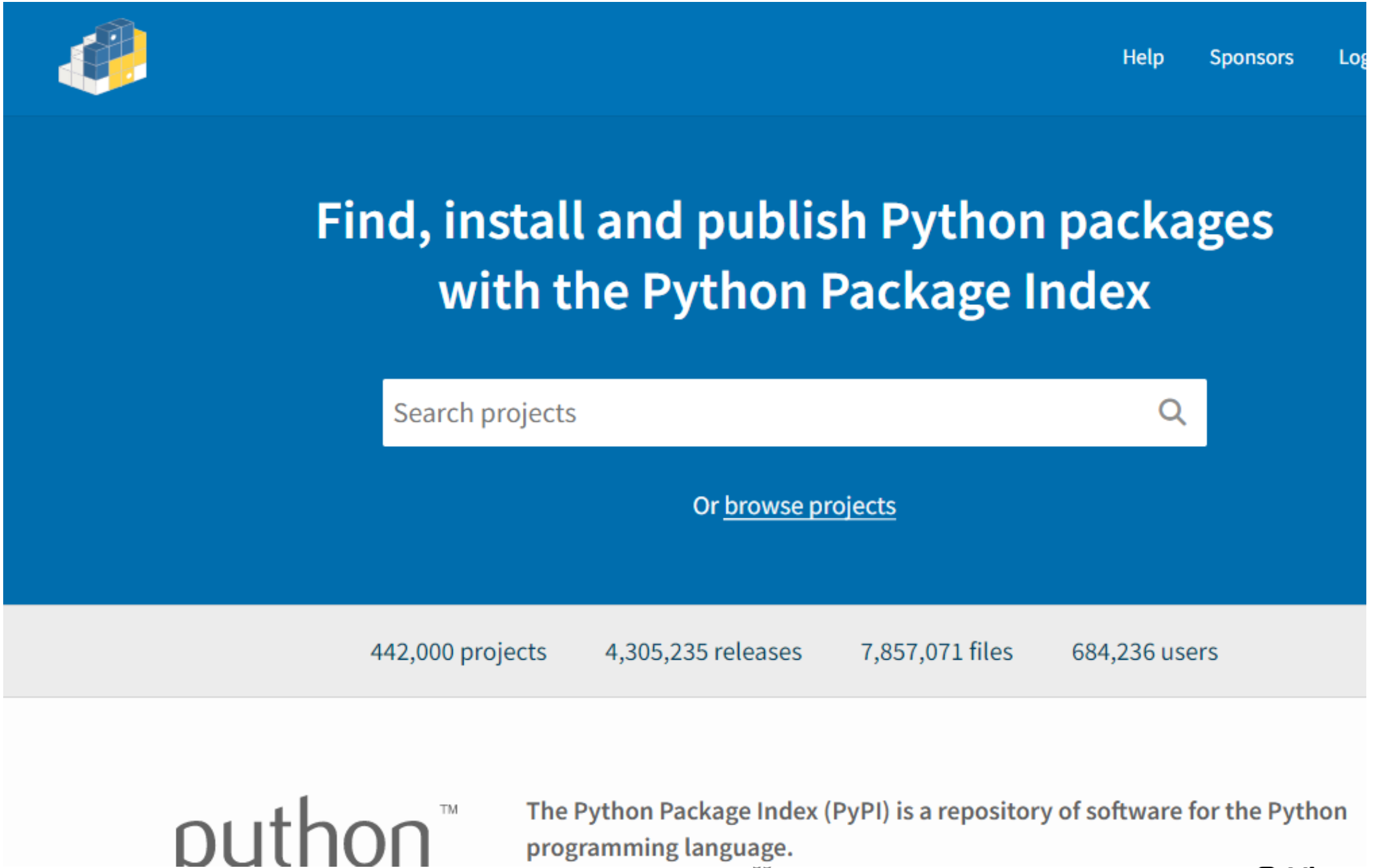
Our repository features over 8,000 open-source data science and machine learning packages, Anaconda-built and compiled for all major operating systems and architectures.

Anaconda vs standalone Python

- standalone Python must have any additional data science packages / libraries installed explicitly using the pip package manager
  - Anaconda distribution comes with all these packages preinstalled
- Anaconda Navigator provides a GUI centric option of selecting packages / tools to use
  - compared to CLI in standalone Python

Anaconda vs standalone Python

- Standalone python uses pip for package management and venv for setting up isolated environments
  - Packages are installed from the Python Package Index (largest open source public repo of Python code)
  - Packages only contain python code
- Anaconda uses conda to perform both package and environment management
  - Packages are installed from various channels: some which are private and others which are open source (conda-forge)
  - Packages can contain modules from other languages



ANACONDA. ORG

Where packages, notebooks, projects and environments are shared.

SEARCH PACKAGES

Search Anaconda.org