

Introduction to Python (hour 1)

- Introduction to the course and Python programming.
- Setting up the development environment (Python installation, text editor/IDE).
- Writing and executing the first Python program (Hello, World!).

Professor's presentation

- **First name:** Alexander
- **Last name:** KOROVIN
- **From:** Orsay, France
- **Education:**
 - 1995 – Engineer of electronics @ Lviv polytechnical Institute, Lviv, Ukraine
 - 2002 – Doctor @ Institute for physics of semiconductors
 - 2014 – Habilitated doctor @ Institute for physics of semiconductors
- **Experience:**
 - High performance numerical simulations
 - Web programming (main developer of the education site prd-mecaqu.centralesupelec.fr)
 - Artificial intuition
 - Programming languages: Python, C/C++/C#/Java, matlab, javascript, SQL, fortran
- **Interests:**
 - Artificial intuition
 - Shader programming
- **Why do I teach?** I am passionate about this subject. I enjoy sharing my knowledge and experience with others, and teaching allows me to delve deeper into my subject area, helping others to understand and appreciate it.

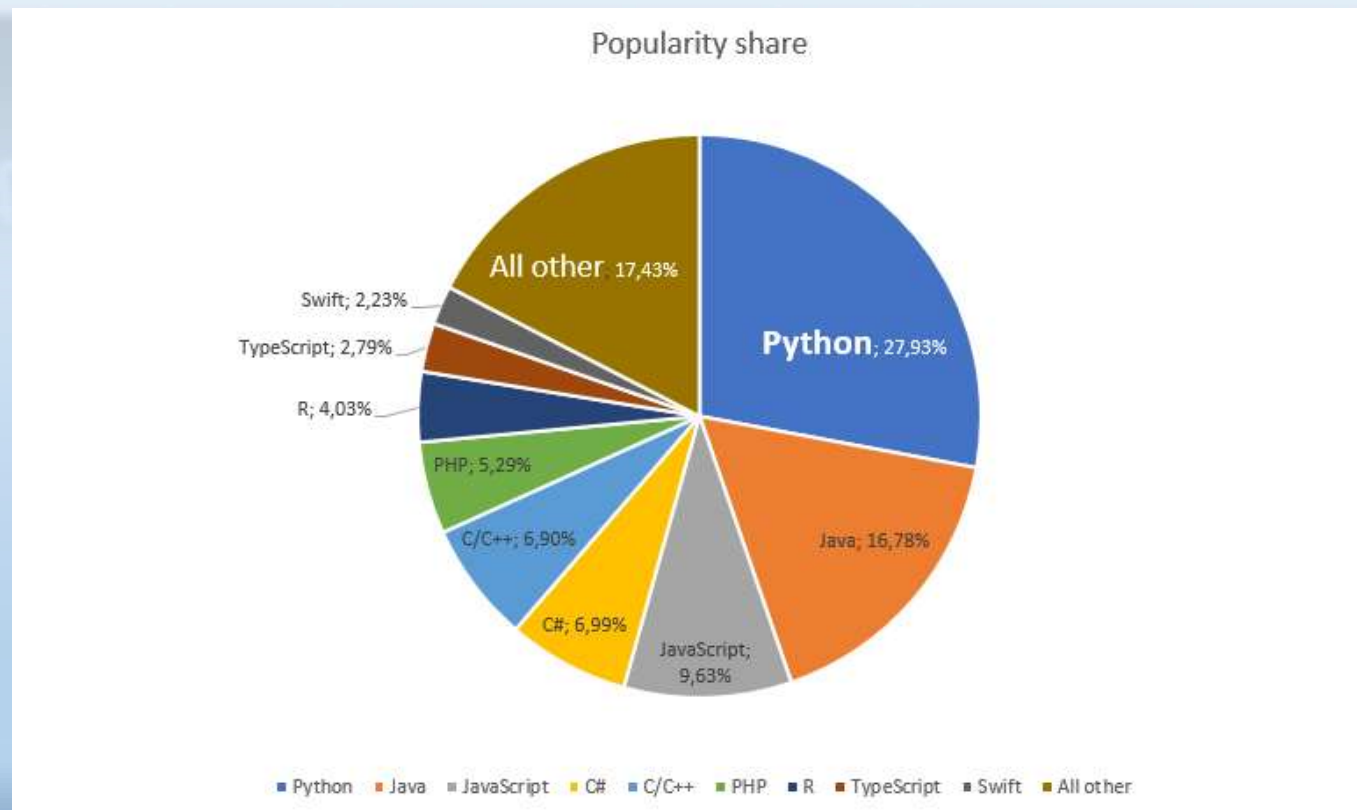
Introduction of students

- First name, last name:
- City/village/country:
- From: country, city
- Status: married, single, ...
- Children:
- Education:
- Work:
- Interests:
- Why do I learn? Expectations:
- Previous classes - where?
- How did you find us?

Why learn python?

- **Versatility:** Python is a versatile programming language that can be used for a wide range of applications. It is commonly used in web development, data analysis, machine learning, artificial intelligence, scientific computing, game development, and more. Learning Python opens up numerous opportunities to work on different projects and domains.
- **Readability and ease of use:** Python is known for its clean and readable syntax, which makes it easier for beginners to understand and write code. It emphasizes simplicity and readability, which reduces the learning curve and allows programmers to focus on solving problems rather than dealing with complex syntax.
- **Large community and extensive libraries:** Python has a vast and active community of developers. This means that there are abundant resources available, including online tutorials, documentation, and forums, where you can seek help and guidance. Python also has a rich ecosystem of libraries and frameworks that provide pre-built functionality, making development faster and more efficient.
- **Data analysis and scientific computing:** Python has become one of the leading languages for data analysis, data visualization, and scientific computing. Libraries such as NumPy, Pandas, and Matplotlib provide powerful tools for handling and analysing data. Python's popularity in these domains is attributed to its ease of use, extensive libraries, and integration with other data-related tools.
- **Machine learning and artificial intelligence:** Python has gained significant traction in the fields of machine learning and artificial intelligence. Libraries like TensorFlow, Keras, and PyTorch have made it easier to build and train machine learning models. Python's simplicity and the availability of these libraries have contributed to its dominance in the AI and ML community.
- **Career opportunities:** Learning Python can open up a wide range of career opportunities. Python developers are in high demand due to its versatility and applications across various industries. Whether you're interested in web development, data science, machine learning, or automation, Python skills can enhance your career prospects and increase your marketability.
- **Rapid prototyping and development:** Python's simplicity and extensive libraries enable developers to build prototypes and iterate quickly. It promotes a faster development cycle, making it an ideal choice for startups, small projects, and scenarios where time-to-market is crucial.
- **Community support and open-source contributions:** Python is an open-source language, which means that it benefits from continuous community contributions and improvements. The active Python community ensures that the language remains up-to-date, secure, and well-supported.

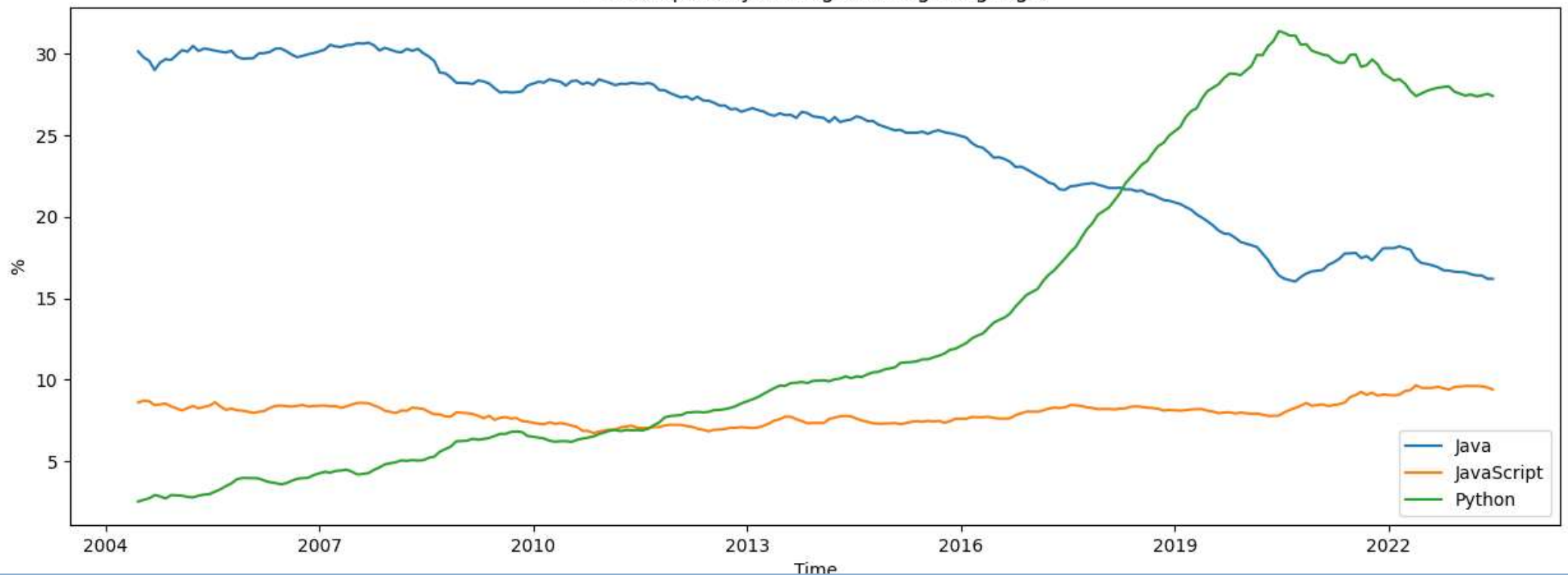
Most popular programming languages (the beginning of 2023)



<https://hitech-us.com/articles/entry/000/what-are-the-most-popular-programming-languages-in-2023>

Popularity of programming languages (worldwide)

PYPL (Popularity of Programming Language)



<https://pypl.github.io/PYPL.html>

A brief history of Python

- **Origin and Guido van Rossum:** Guido van Rossum, a Dutch programmer, started working on Python in the late 1980s. He aimed to create a programming language that was easy to read, had a clean syntax, and emphasized code readability and simplicity.
- **Early development and release:** The first version of Python, known as Python 0.9.0, was released in February 1991. It included features like exception handling, functions, and classes. Python's design philosophy, influenced by principles such as "Readability counts" and "There should be one-- and preferably only one --obvious way to do it," shaped its development from the beginning.
- **Python 2.x:** The next major release, Python 2.0, came in October 2000. Python 2.x series introduced several new features and improvements, including list comprehensions, garbage collection, and a unified object model. The Python 2.x series continued with various updates and reached its final release as Python 2.7 in July 2010.
- **Python 3.x:** Python 3.0, a significant milestone, was released in December 2008. Python 3.x introduced backward-incompatible changes to the language to improve its design and address some of the limitations and inconsistencies of Python 2.x. While Python 2.x continued to be used for many years, the Python community actively encouraged the adoption of Python 3.x.
- **Growth in popularity:** Over the years, Python has gained widespread popularity and a strong community of developers. Its simplicity, readability, and versatility have contributed to its success. Python is widely used in various domains, including web development, data analysis, scientific computing, machine learning, artificial intelligence, and automation.
- **The Python Software Foundation (PSF):** In 2001, the Python Software Foundation was established as a nonprofit organization to promote, develop, and maintain Python. The PSF supports the development and improvement of Python, organizes conferences, maintains the official Python documentation, and oversees the licensing and intellectual property aspects of Python.
- **Recent developments:** Python continues to evolve, with regular releases that introduce new features, improvements, and bug fixes. Python 3.10, the latest major release as of my knowledge cutoff in September 2021, brought enhancements like pattern matching and structural pattern matching.

Python features

no compiling or linking	rapid development cycle
no type declarations	simpler, shorter, more flexible
automatic memory management	garbage collection
high-level data types and operations	fast development
object-oriented programming	fewer restrictions and rules
embedding and extending in C	interactive, dynamic nature
classes, modules, exceptions	programs can be modified without stopping
dynamic loading/reloading of C modules	cross-platform programming
access to interpreter information	system tools, GUIs, persistence, databases, etc.
built-in interfaces to external services	

Advantages and disadvantages

Advantages:

- **Versatility:** Python is a versatile language that can be used for various purposes, including web development, data analysis, scientific computing, machine learning, artificial intelligence, automation, and more. It offers a wide range of libraries and frameworks that support different domains.
- **Readability and simplicity:** Python emphasizes code readability and has a clean, easy-to-understand syntax. This makes it a great language for beginners and enhances collaboration among developers.
- **Large community and extensive libraries:** Python has a large and active community of developers. This means there are abundant resources, libraries, and frameworks available for various purposes. The Python Package Index (PyPI) hosts a vast collection of third-party libraries that can be easily installed and used in Python projects.
- **Rapid prototyping and development:** Python's simplicity and extensive libraries enable developers to quickly prototype and develop applications. It has a shorter learning curve compared to some other programming languages, allowing developers to focus on problem-solving rather than dealing with complex syntax.
- **Integration capabilities:** Python can easily integrate with other languages like C, C++, and Java. This allows developers to leverage existing codebases and libraries from different languages, enhancing the overall capabilities of their projects.
- **Cross-platform compatibility:** Python is available on various operating systems, including macOS, Windows, Linux, and more. This cross-platform compatibility ensures that Python code can run seamlessly across different environments.

Disadvantages:

- **Performance:** Python is an interpreted language, which means it can be slower in terms of execution speed compared to compiled languages like C or Java. While Python has made performance improvements over the years, it may not be the best choice for performance-critical applications.
- **Global Interpreter Lock (GIL):** Python's Global Interpreter Lock (GIL) allows only one thread to execute Python bytecode at a time. This can limit the parallelism of multi-threaded Python programs, affecting performance in certain scenarios.
- **Mobile and browser support:** Python is not as widely used for mobile app development or in-browser scripting as languages like Java, Swift, or JavaScript. While there are frameworks like Kivy and BeeWare that enable mobile app development in Python, the ecosystem is not as extensive as in other languages.
- **Memory consumption:** Python's memory consumption can be relatively high compared to some other programming languages. This can become a concern when dealing with large-scale applications or scenarios where memory optimization is critical.
- **Limited in some domains:** While Python is widely used, there are certain domains where other languages may have more extensive libraries and frameworks. For example, in game development, languages like C++ or Unity/C# may offer more specialized tools and performance optimizations.

Literature

- www.theinsaneapp.com/2021/05/best-free-python-programming-books.html
- www.learnpython.org

Modes

1. IPython

Python can be run interactively

Used extensively in research

2. Python scripts

What if we want to run more than a few lines of code?

Then we must write text files in .py

Working with python

- **IDE** (Integrated development environment). Most popular:
 - PyCharm
 - Spyder
 - MS Code
- **Jupyter Notebook**
 - Local server
 - Google Colab (<https://colab.research.google.com>), Microsoft Azure Notebooks, CoCalc...
- **Online:**
 - www.online-python.com
 - <https://www.programiz.com/python-programming/online-compiler/>
 - ...
- **Any text editor**
 - Notepad (Notepad++), Vim, gedit, Atom, ...

Spyder

The screenshot displays the Spyder IDE interface with the following components:

- File Explorer (Left):** Shows a project structure with folders like 'Plots' and 'IPythonConsole', and files like 'plugin.py', 'chart_plot_example.py', and 'plugin.py - ipythonconsole'.
- Code Editor (Center):** Displays the source code for 'plugin.py'. The code includes comments, imports, and a class definition for 'Plots'.
- Variable Explorer (Top Right):** A table showing the current state of variables in the environment.
- Plots Panel (Bottom Right):** Displays two plots: a 3D surface plot and a polar plot.

Name	Type	Size	Value
bool	bool	1	True
data	Array of str128 (3, 3)		ndarray object of numpy module
datetime_object	datetime	1	2021-04-14 17:35:14.687085
df	DataFrame	(2, 2)	Column names: Col1, Col2
filename	str	53	/Users/Documents/spyder/spyder/tests/test_dont_use.py
li	list	5	['abcd', 745, 2.23, 'efgh', 70.2]
myset	set	3	{'2', '1', '3'}
r	float	1	6.46567886443
t	tuple	5	('abcd', 745, 2.23, 'efgh', 70.2)
tinylist	list	2	[123, 'efgh']
x	float64	1	1.1235123099439

```
1 # coding: utf-8
2
3 # Copyright © Spyder Project Contributors
4 # Licensed under the terms of the MIT License
5 # (see spyder/_init_.py for details)
6
7
8
9
10
11
12 # Third party imports
13 from qtpy.QtCore import Signal
14
15 # Local imports
16 from spyder.api.plugins import Plugins, SpyderDockablePlugin
17 from spyder.api.translations import get_translation
18 from spyder.plugins.plots.widgets.main_widget import PlotsWidget
19
20 # Localization
21 _ = get_translation('spyder')
22
23
24 class Plots(SpyderDockablePlugin):
25     """
26     Plots plugin.
27     """
28     NAME = 'Plots'
29     REQUIRES = [Plugins.IPythonConsole]
30     TABIFY = [Plugins.VariableExplorer, Plugins.Help]
31     WIDGET_CLASS = PlotsWidget
32     CONF_SECTION = NAME
33     CONF_FILE = False
34     DISABLE_ACTIONS_WHEN_HIDDEN = False
35
36     # --- SpyderDockablePlugin API
37     #
38     def get_name(self):
39         return _('Plots')
40
41     def get_description(self):
42         return _('Display, explore and save console generated plots.')
43
44     def get_icon(self):
45         return self.create_icon('hist')
46
47     def register(self):
48         ipyconsole = self.get_plugin(Plugins.IPythonConsole)
49
50         # Signals
51         ipyconsole.sig_shellwidget_changed.connect(self.set_shellwidget)
52         ipyconsole.sig_shellwidget_process_started.connect(
53             self.add_shellwidget)
54         ipyconsole.sig_shellwidget_process_finished.connect(
55             self.remove_shellwidget)
```

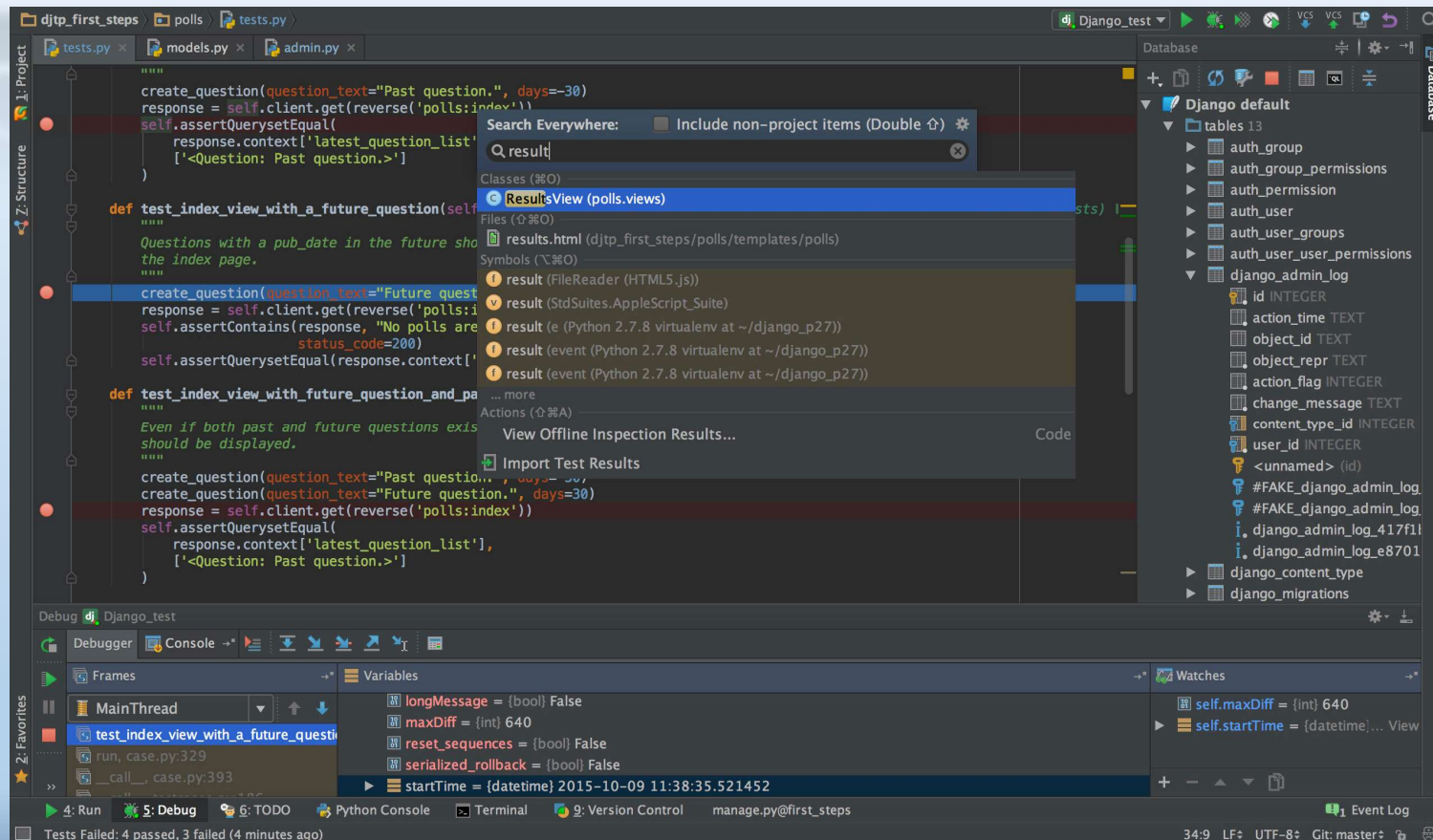
github.com/spyder-ide/spyder

03.07.2023 - 21.08.2023

PYTHON FOR BEGINNERS

14

PyCharm community/professional



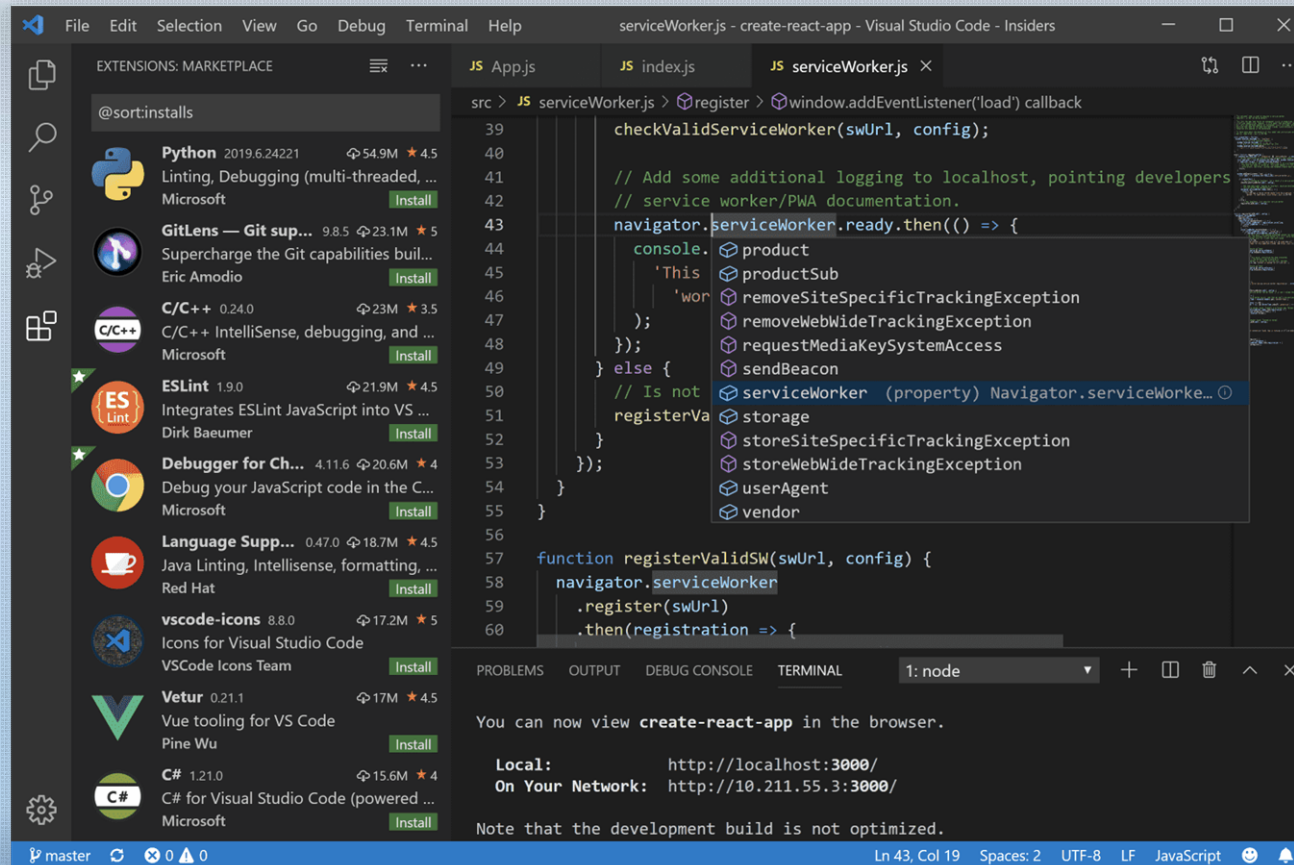
www.jetbrains.com/pycharm

03.07.2023 - 21.08.2023

PYTHON FOR BEGINNERS

15

Microsoft Code

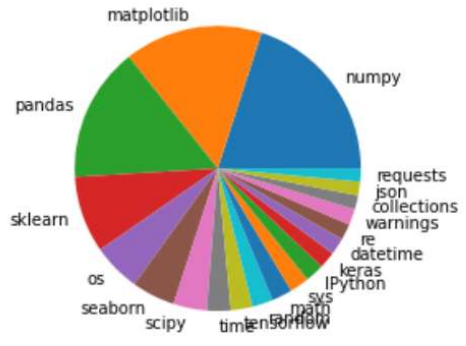


code.visualstudio.com/docs/python/python-tutorial

Jupyter notebook

example.ipynb

```
In 5: 1 import matplotlib.pyplot as plt
      2 plt.pie(kernel_stats['total_count'],
      3          labels=kernel_stats['library'])
      plt.show()
```



Jupyter Variables

- kernel_stats = {DataFrame: (20, 7)} [View as DataFrame](#)
 - at = {_AtIndexer} <pandas.core.indexing._AtIndexer>
 - attrs = {dict: 0} {}
 - axes = {list: 2} [RangeIndex(start=0, stop=... [View](#)
 - columns = {Index: (7,)} Index(['library', 'ke... [View](#)
 - dtypes = {Series: (7,)} ('library', c... [View as Series](#)
 - empty = {bool} False
 - flags = {Flags} <Flags(allows_duplicate_labels=Tr
 - iat = {_iAtIndexer} <pandas.core.indexing._iAtInde
 - iloc = {_iLocIndexer} <pandas.core.indexing._iLoc
 - index = {RangeIndex: (20,)} RangeIndex(start=0, :
 - library = {Series: (20,)} (0, 'num... [View as Series](#)
 - loc = {_LocIndexer} <pandas.core.indexing._LocI
 - ndim = {int} 2
 - plot = {PlotAccessor} <pandas.plotting._core.Plot
 - shape = {tuple: 2} (20, 7)
 - size = {int64: ()} 140
 - style = {Styler} <pandas.io.formats.style.Styler ok
 - T = {DataFrame: (7, 20)} 0 ... [View as DataFrame](#)
 - total_count = {Series: (20,)} (0, 4... [View as Series](#)

www.jetbrains.com/help/dataspell/jupyter-notebook.html

Installation of python

- Windows
- Linux
- MacOS

Windows installation of python

- 1. Download the Python installer:** Go to the Python website at <https://www.python.org/>. This website provides the official Python distribution for various operating systems. Click on the "Downloads" tab. You will see different versions of Python available for download. Choose the version that suits your needs. For most users, it is recommended to download the latest stable release.
 - Note: There are two major Python versions, Python 2 and Python 3. It is generally recommended to use Python 3 as Python 2 is no longer actively developed or supported.
- 2. Select the installer:** After choosing the Python version, scroll down to the section titled "Files." You will find different installers available based on your Windows version (32-bit or 64-bit). Select the installer appropriate for your Windows version.
- 3. Run the installer:** Once the installer is downloaded, double-click on it to run the installation process. You may need administrator privileges to install Python on your system.
- 4. Customize the installation (optional):** The installer will provide an option to customize the installation. You can choose to modify the installation location, add Python to the system PATH (recommended), and select optional features. By adding Python to the system PATH, you can use Python from any command prompt without specifying its full path.
- 5. Complete the installation:** Proceed with the installation process by clicking "Next" or "Install" and follow the instructions provided by the installer. The installer will extract and install Python on your Windows system.
- 6. Verify the installation:** After the installation is complete, you can verify if Python is installed correctly. Open the command prompt (press Windows key + R, type "`cmd`," and press Enter) and type "`python`" or "`python3`" followed by Enter. If the installation was successful, you should see the Python interactive shell with the Python version displayed.

Linux (UBUNTU) installation of python

By default, Ubuntu comes with Python 2.x pre-installed. However, it's recommended to install the latest version of Python 3.x.

- 1. Install Python.** Open a terminal (by pressing **Ctrl+Alt+T** or by searching for "Terminal" in the Ubuntu applications).
 - Update package lists. Before installing any software, it's a good practice to update the package lists to ensure you have the latest versions available. You will be prompted to enter your password. Note that when entering the password, no characters will appear on the screen, but it is being entered correctly:
sudo apt update
 - Install Python. During the installation, you will be asked to confirm the installation by typing 'Y' and pressing Enter. The installation process will then begin.
sudo apt install python3
- 2. Verify the installation** After the installation is complete, you can verify that Python is installed correctly by checking its version. Run the following command:
 - **python3 --version**
 - This will display the installed Python version, confirming that Python is successfully installed on your Ubuntu system.
- 3. Set Python 3 as the default version (optional)** If you prefer to use Python 3 as the default version when running the python command, you can create a symbolic link. Run the following command in the terminal:
 - **sudo ln -s /usr/bin/python3 /usr/bin/python**

This will create a symbolic link named python that points to the python3 executable. That's it! Python is now installed on your Ubuntu system. You can start using Python by running python3 or python in the terminal, depending on whether you set Python 3 as the default version or not.

MacOS installation of python

1. **Check if Python is already installed:** macOS comes with a pre-installed version of Python. Open the Terminal application (found in the "Utilities" folder within "Applications") and type `python --version` or `python3 --version` to see if Python is already installed and which version it is.
2. **Download the Python installer:** If Python is not already installed or you want to update to a newer version, you can download the installer from the official Python website. Visit the Python downloads page at <https://www.python.org/downloads/> and click on the macOS installer appropriate for your version of macOS.
3. **Run the installer:** Once the installer file (usually a .pkg file) is downloaded, double-click on it to start the installation process. You may be prompted to enter your administrator password.
4. **Customize the installation (optional):** During the installation process, you can customize the installation by clicking on the "Customize Installation" button. Here, you can choose the components you want to install and select the option to add Python to the system PATH.
5. **Complete the installation:** After customizing the installation (if desired), click on the "Install" button to begin the installation process. The installer will copy the necessary files to your system.
6. **Verify the installation:** Once the installation is complete, open a new Terminal window and type `python --version` or `python3 --version` to confirm that Python has been installed and to check the version number.

Python should now be successfully installed on your macOS system. You can start using Python by opening the Terminal and typing `python` or `python3` to enter the Python interactive shell. Alternatively, you can write Python code in a text editor and save it with a .py extension, then run it using the Terminal with the `python filename.py` or `python3 filename.py` command, replacing "filename" with the actual name of your Python file.

Virtual environment in Python

Using a virtual environment allows you to manage different sets of packages and their versions for different projects. It helps keep project dependencies isolated and avoids conflicts between packages used in different projects.

- **Create** a new virtual environment:

- Open your terminal or command prompt.
- Navigate to the directory where you want to create the virtual environment.
- Run the following command to create a new virtual environment (replace myenv with the desired name for your environment):

```
python3 -m venv myenv or python3 -m venv /path/to/myenv
```

- **Activate** the virtual environment:

- On Windows: `myenv\Scripts\activate.bat`
- On macOS and Linux: `source myenv/bin/activate`

After activation, your terminal prompt should change to indicate that you are now working within the virtual environment.

- **Deactivate** the virtual environment: To exit the virtual environment and return to your system's global Python environment, use the following command:

```
deactivate
```


Write your first python program

Open a text editor and create a new file.

Type or copy the following line into the file:

```
print("Hello world!")
```

Save the file with a .py extension. For example, you can save it as hello.py.

Open a terminal or command prompt. Once you are in the correct directory, you can run the Python file using the python command followed by the filename. In this case, you would run:

```
python hello.py
```

or

```
python3 hello.py
```

That's it! You have successfully run a Python file containing a "Hello world!" program in the terminal.

5-minute pause

5-minute pause

Introduction to Python (hour 2)

- Variables, data types, and basic operations.
- Working with numbers and strings.
- User input and output.

Comments in Python

Python supports two types of comments:

1. Single-line comments: Single-line comments start with a hash symbol (#) and continue until the end of the line. Everything after the # symbol on the same line is considered a comment.

Here's an example:

```
# This is a single-line comment  
print("Hello, world!") # This is another comment
```

2. Multi-line comments: Multi-line comments are used when you want to add comments that span multiple lines. In Python, there is no specific syntax for multi-line comments like some other programming languages. Instead, you can use triple quotes (''' or ''') to create a multi-line string, which is treated as a comment and ignored by the interpreter.

3. Here's an example:

```
'''  
This is a multi-line comment.  
It can span multiple lines.  
'''  
print("Hello, world!")
```

Variables in Python

Variables are used to store data in Python. They act as containers that hold values. You can think of variables as named references to memory locations where data is stored.

Assigning Values to Variables: use the assignment operator “=”:

```
variable_name = value
```

Here, `variable_name` is the name of the variable, and `value` is the data you want to store.

```
# Assigns the integer value 5 to the variable x
x = 5
# Assigns the string 'John' to the variable name
name = 'John'
# Assigns the floating-point value 3.14 to the variable pi
pi = 3.14
# Assigns the boolean value True to the variable is_true
is_true = True
```

Variable names

- **Valid Characters:**

- Variable names can contain letters (**a-z**, **A-Z**), digits (**0-9**), and underscores (**_**)
- The first character of a variable name cannot be a digit.
- Python is case-sensitive, so **myVariable** and **myvariable** are considered different variable names.

- **Cannot use reserved keywords as a variable name:**

- Variable names cannot be the same as reserved keywords in Python.
- Reserved keywords are predefined words in the Python language that have special meanings and are used to define the syntax of the language.

- **Length Limit:**

- Variable names can be of any length, but it's recommended to keep them concise and meaningful.
- Although there is no specific limit on the length of a variable name, excessively long names can make code harder to read and maintain.

- **Clarity and Readability (recommended):**

- It's important to choose variable names that are clear, descriptive, and reflect the purpose or meaning of the data they represent.
- Variable names should be meaningful and provide a clear understanding of the stored value.

List of Python keywords

and as assert break class continue def	del if is import in elif else	except False finally for from global lambda	None nonlocal not or pass raise	return True try while with yield
dict	list	set		