Python for Data Science & AI

Dasun Athukoralage

web: www.dasuna.me
email: dasun@nirvanaclouds.com

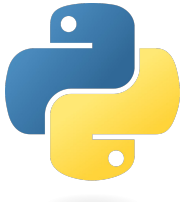
Computer Programming

Computer programming involves **writing instructions**, also known as code, that tell a computer how to perform specific tasks.

There are numerous **programming languages** available, each with its own syntax and purpose.

Programming Languages

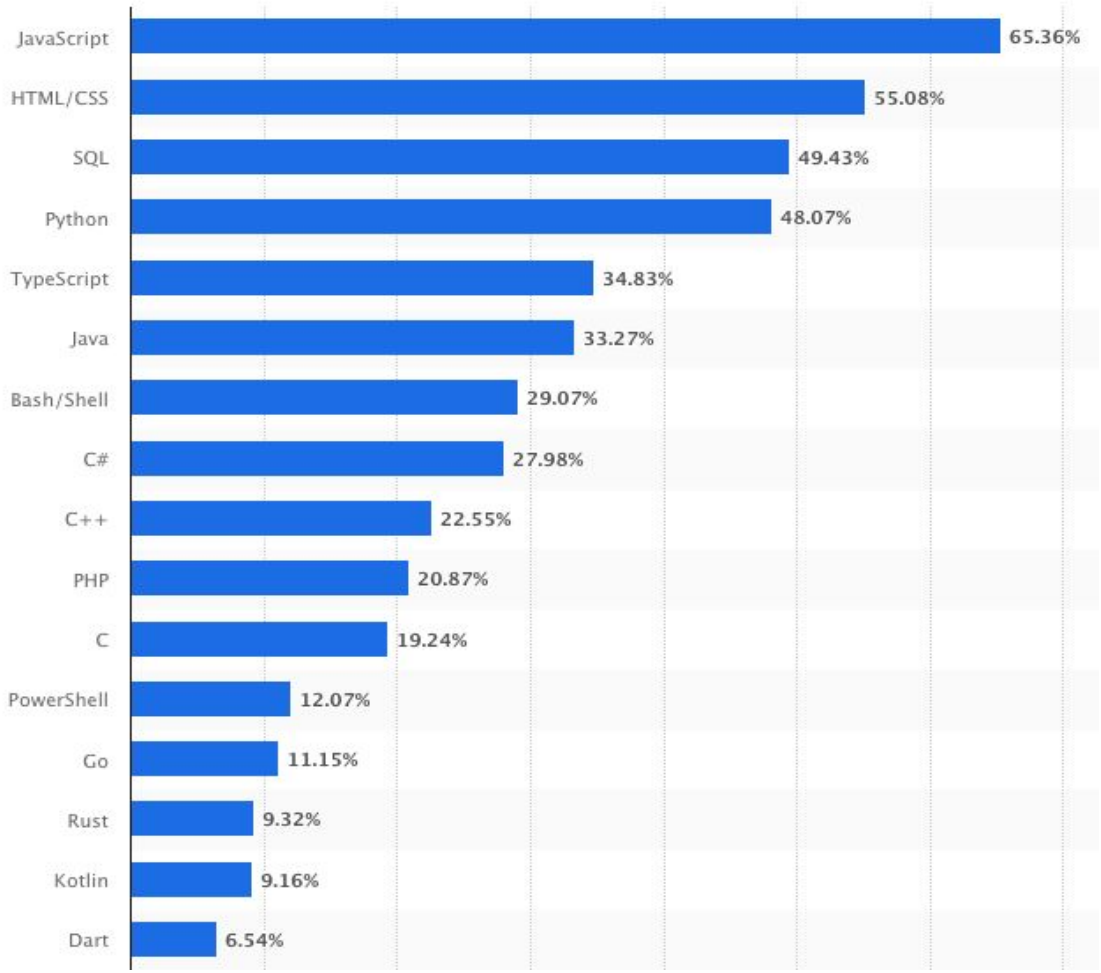
- Python
- Java
- C++
- JavaScript
- Ruby
- Go



Why Python?

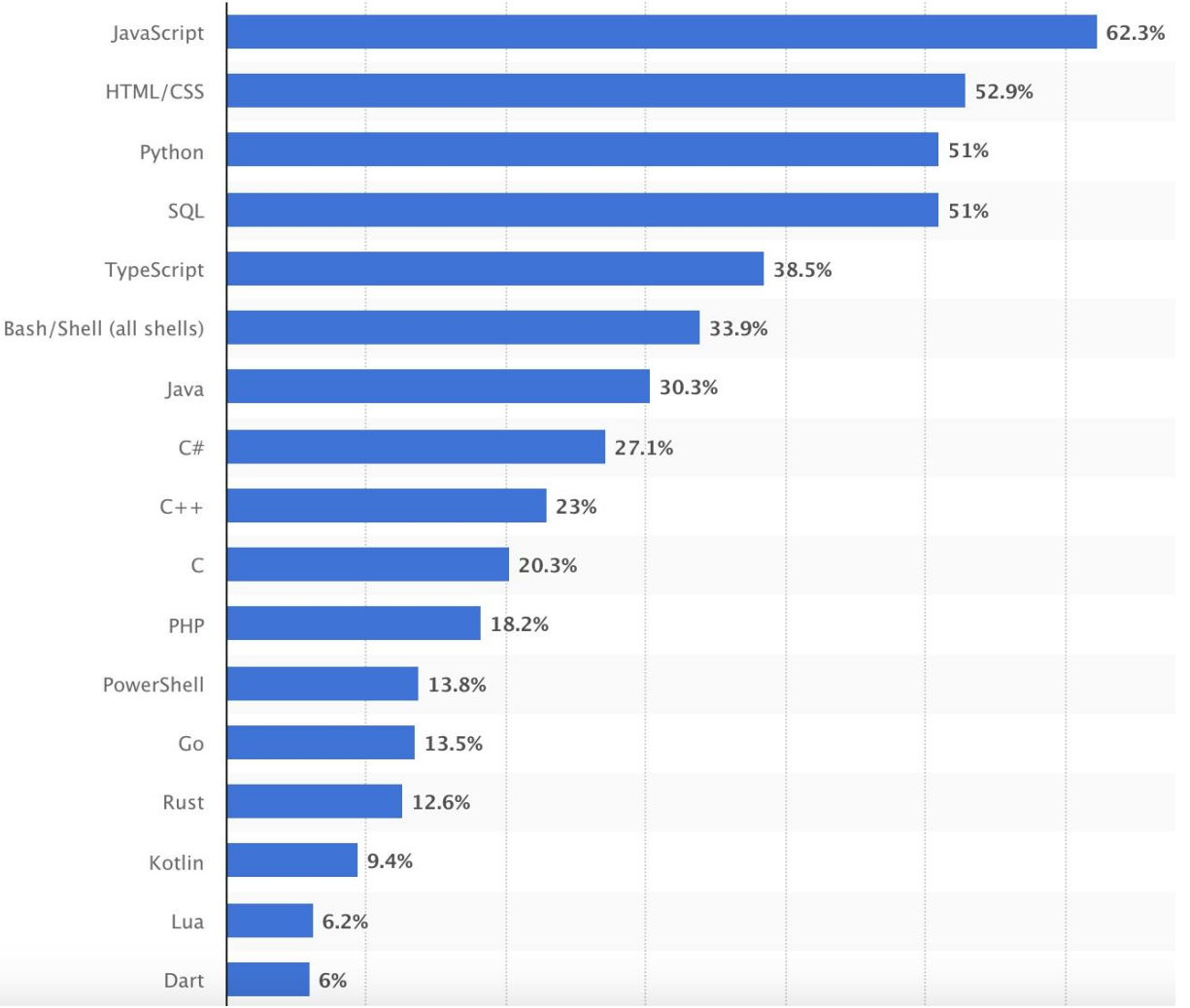
Most used programming languages among developers worldwide as of **2022**.

(Source: Statista)



Why Python?

As of 2024, most top programming languages have declined in popularity, while Python has gained even more traction—driven by the growing interest in AI.



Why Python?

Python is a high-level, interpreted programming language with a **simple syntax**, which makes it easily readable and extremely **user- and beginner-friendly**.

Why Python?

Python

```python
print("Hello world.")
```

vs.

Java

```java
public class HelloWorld {
    public static void main (String[]args) {
      System.out.println.("Hello world");
    }
}
```

Why Python?

- Python can be used for just about anything, from **web and software development** to **machine learning and artificial intelligence (AI)**.

- **Large & Active community**: making it easier to find solutions to problems, get help, and share knowledge.

Why Python?

- **Integration Capabilities**: Python can seamlessly integrate with other languages and technologies. It has extensive support for integration with C, C++, Java, and other languages
- **Job Opportunities**: Python is in high demand across various industries, including web development, data science, machine learning, and automation. Learning Python can open up numerous job opportunities and career paths.

Tools & Software

What is an Integrated Development Environment (IDE)?

It is a software application that provides developers with a comprehensive set of tools for writing, testing, and debugging code in various programming languages.

IDEs are designed to streamline the software development process and make it more efficient for programmers.

Popular IDEs

- Visual Studio (free)
- Eclipse (free & Open-source)
- IntelliJ IDEA (there is a free version and paid version)
- Android Studio (free)
- PyCharm (there is a free version and paid version)
- NetBeans (free & Open-source)

What are Code Editors?

A code editor is a lightweight software application designed specifically for editing and writing code. It offers features tailored to developers' needs, such as syntax highlighting, code autocompletion, line numbering, and code folding.

IDEs vs Code Editors

- Code editors are suitable for quick editing tasks, scripting, and small to medium-sized projects. They are often preferred for their speed, simplicity, and the ability to run on various operating systems.

Popular Code Editors

- Visual Studio Code (VS Code) (free)
- Sublime Text (free)
- Atom (free & Open-source)(has been sunset)
- Notepad++
- Vim

Popular Code Editors or IDEs for Python

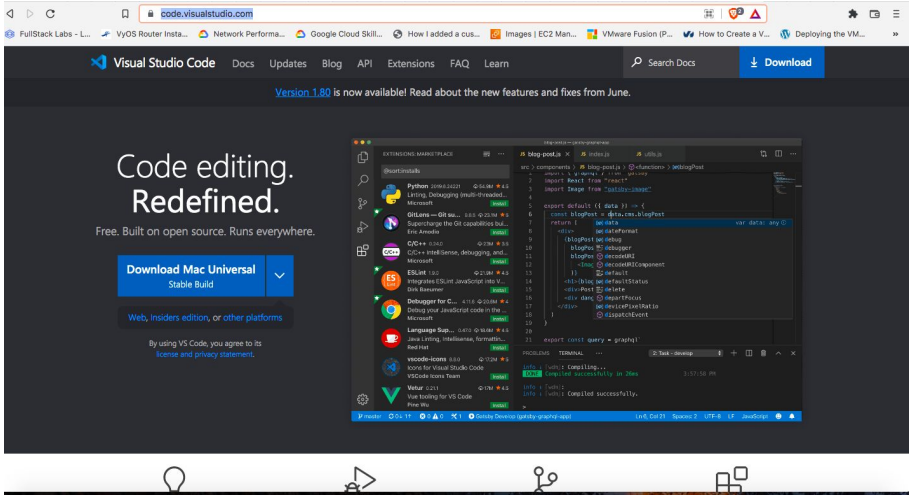
- Visual Studio Code (VS Code) (free & Open-source)
- PyCharm (there is a free version and paid version)
- Spyder (free & Open-source)

- Jupyter Notebook (free & Open-source)

VS Code

It is a free and open-source code editor developed by Microsoft. VS Code is available for Windows, macOS, and Linux operating systems.

You can download it from official website : https://code.visualstudio.com/



Python in VS Code

Working with Python in Visual Studio Code, using the Microsoft Python extension, is simple, fun, and productive. The extension makes VS Code an excellent Python editor, and works on any operating system with a variety of Python interpreters.

Download URL : https://marketplace.visualstudio.com/items?itemName=ms-python.python

Python in VS Code

Visual Studio | Marketplace

Sign in

Visual Studio Code > Programming Languages > Python

New to Visual Studio Code? Get it now.

Python

Microsoft microsoft.com | 91,510,357 installs | ★★★★★ (548) | Free

IntelliSense (Pylance), Linting, Debugging (multi-threaded, remote), Jupyter Notebooks, code formatting, refactoring, unit tests, and more.

Install Trouble Installing?

Overview Version History Q & A Rating & Review

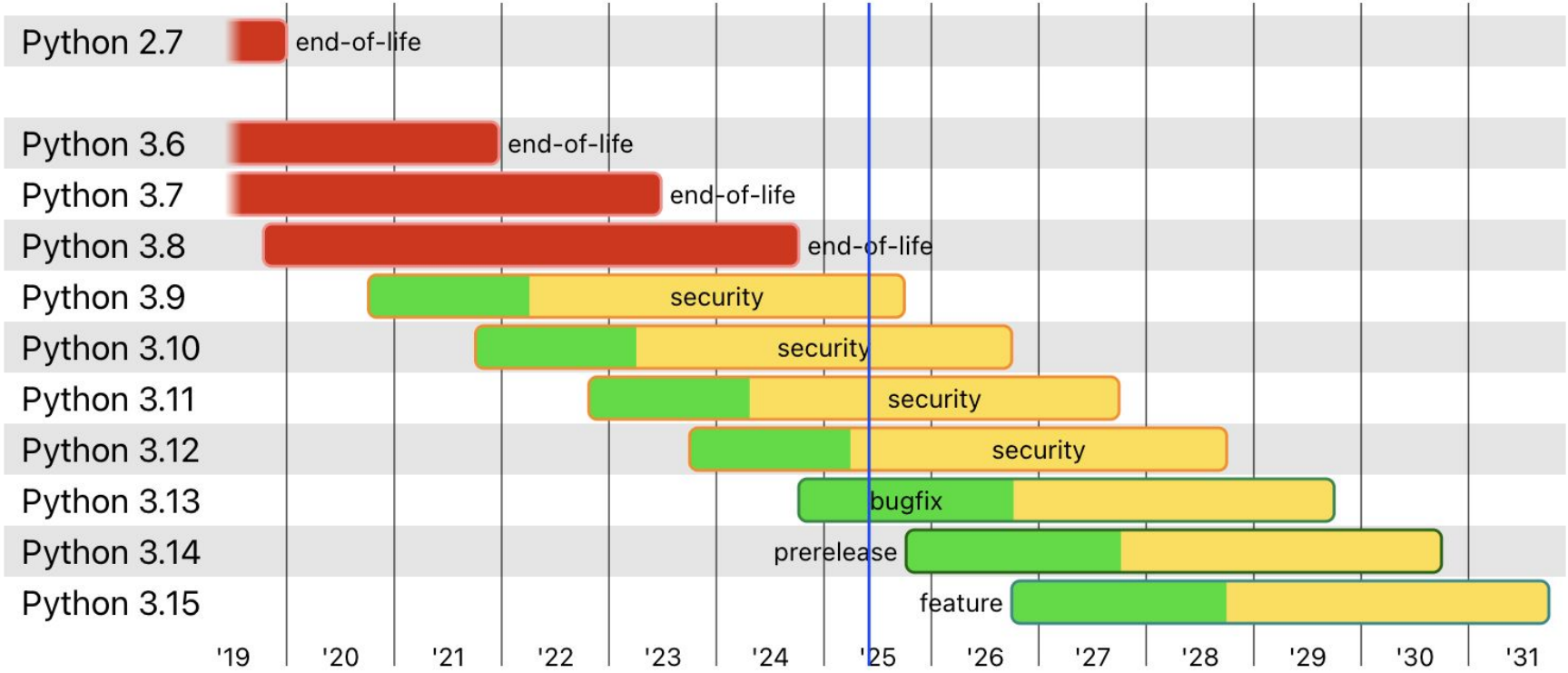
Python extension for Visual Studio Code

A Visual Studio Code extension with rich support for the Python language (for all actively supported versions of the language: >=3.7), including features such as IntelliSense (Pylance), linting, debugging, code navigation, code

Categories

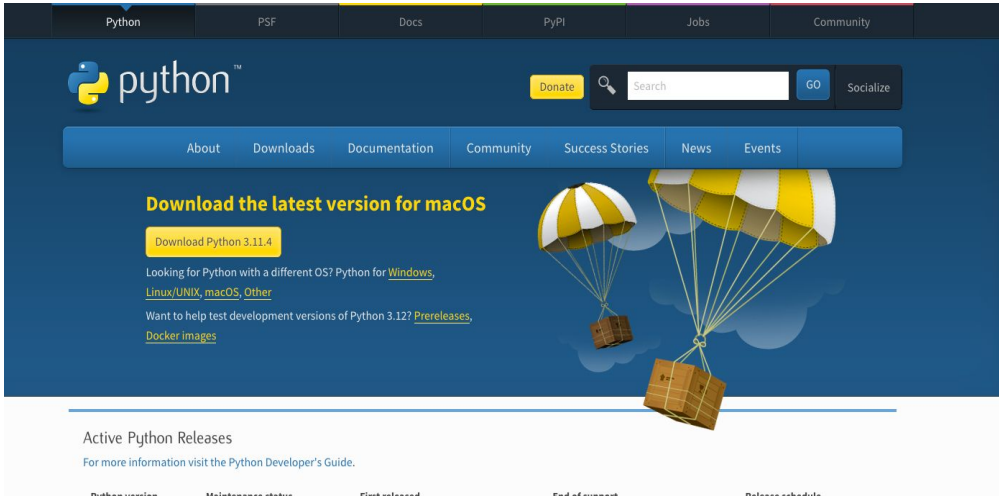
Programming Languages | Linters | Debuggers | Formatters | Data Science | Machine Learning

Status of Python Versions



Installing Python

Visit https://www.python.org/downloads/ and download the required Python version. Downloads are available for different OSs (Windows, Linux, macOS).



Installing Python

Currently latest (stable) version is: **3.13.7**

As of September 2025, Python 3.13.7 is the stable release, and **3.13** is the only version with active (as opposed to just security) support.

How to check Python has been installed?

Open a Command Prompt (Windows) or Terminal (macOS and Linux). Type the following command.

```
python --version
```

It will return something like this Python 3.13.0

But in some OS (macOS) default python version 2.X. So if you type the above command even though you have installed python 3.X version it will return 2.x version

How to check Python has been installed?

Instead type the following command in the terminal

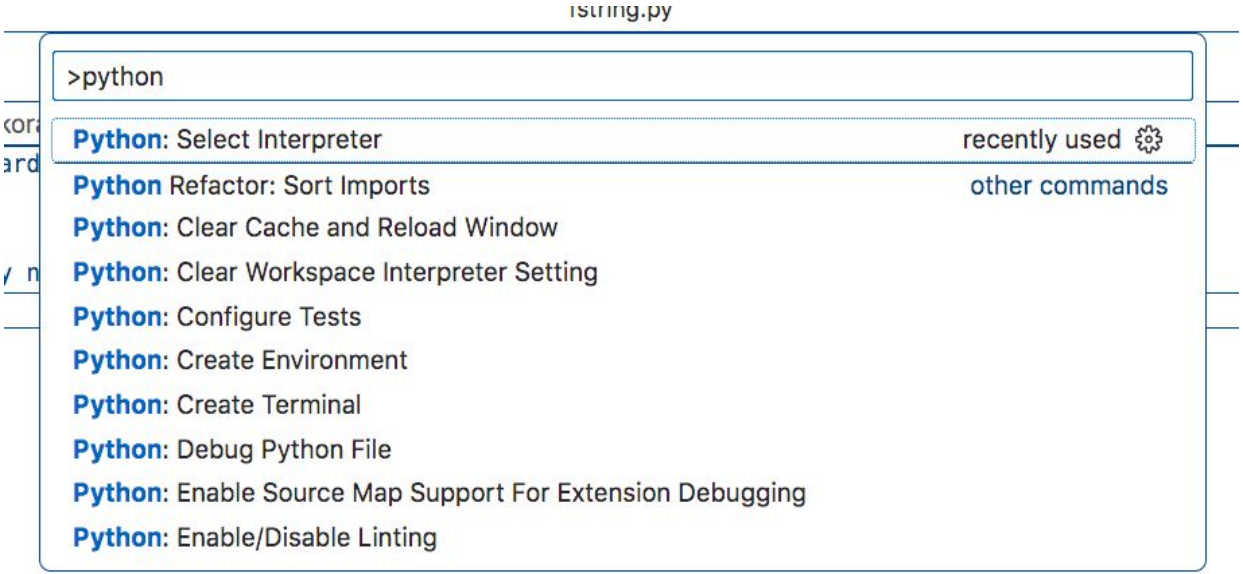
python3 --version

It will return something like this Python 3.13.0

Then you can confirm that Python3 has been installed successfully.

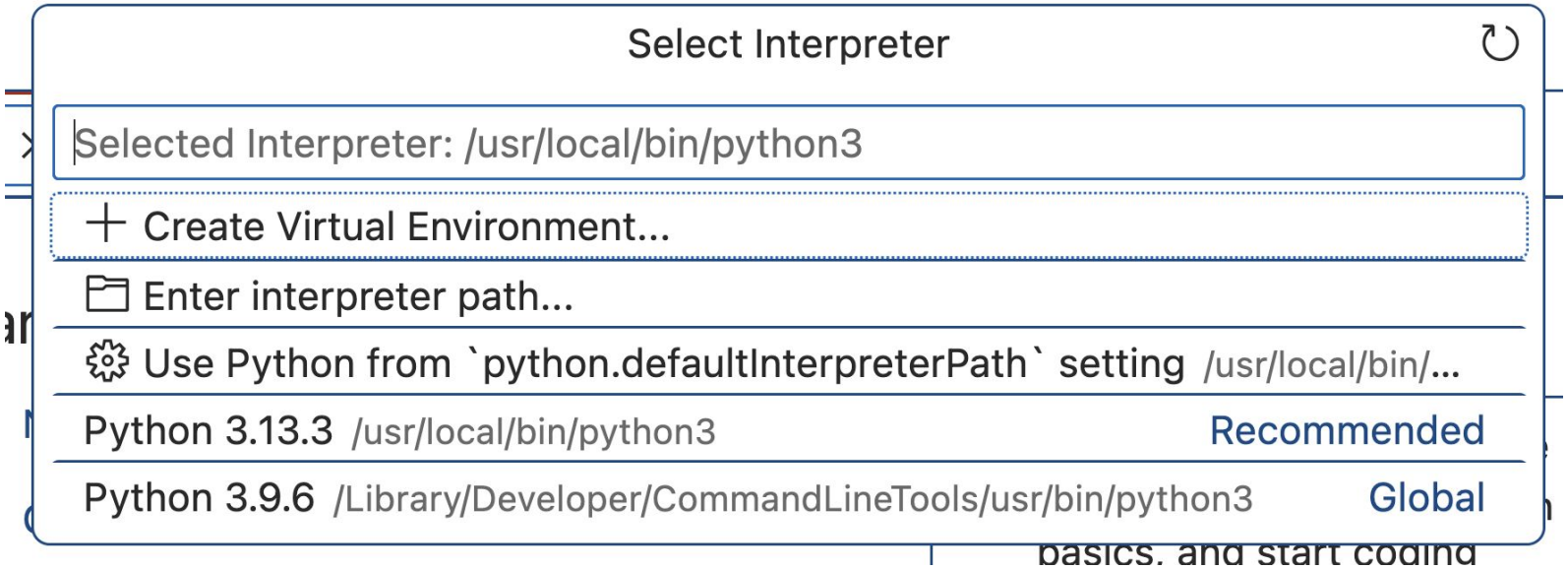
How to make sure it will use correct Interpreter?

Open Command Pallete and Type "Python: Select Interpreter" and press Enter.

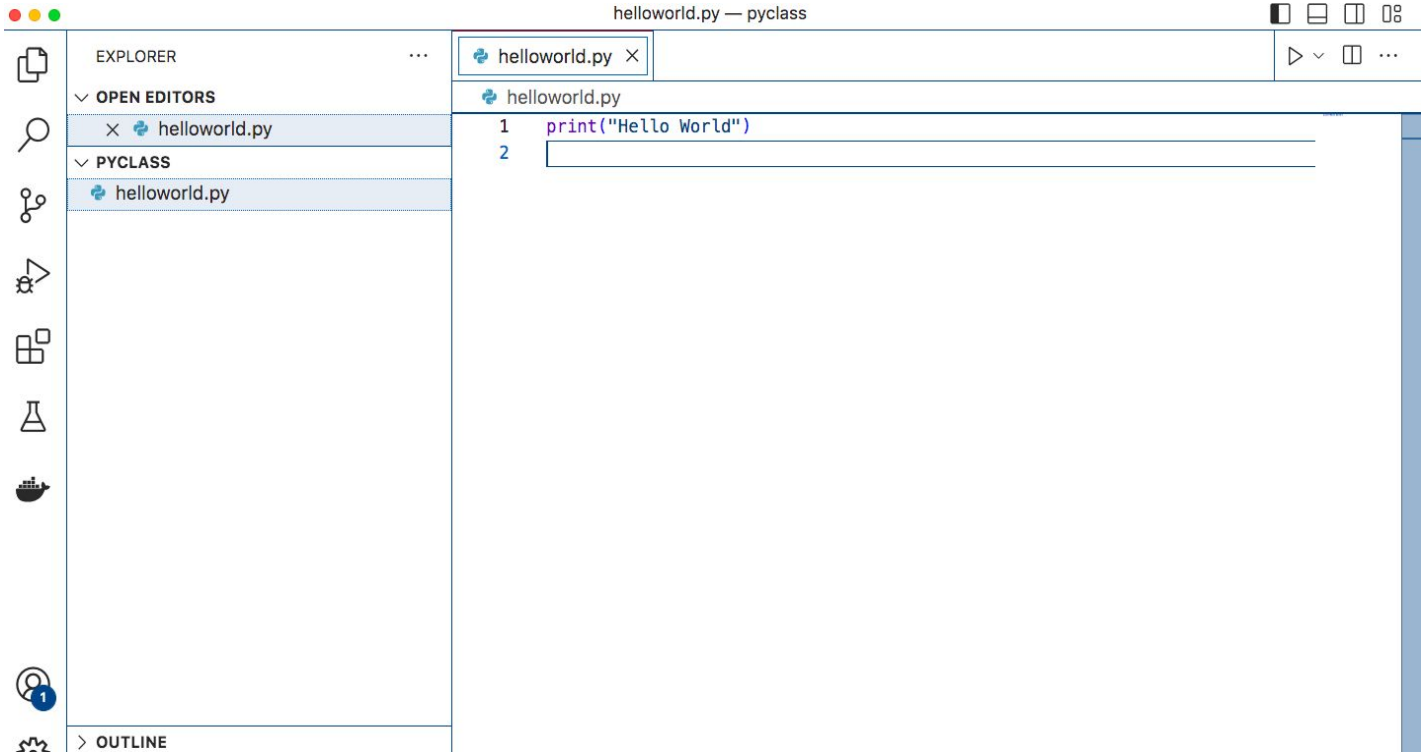


How to make sure it will use correct Interpreter?

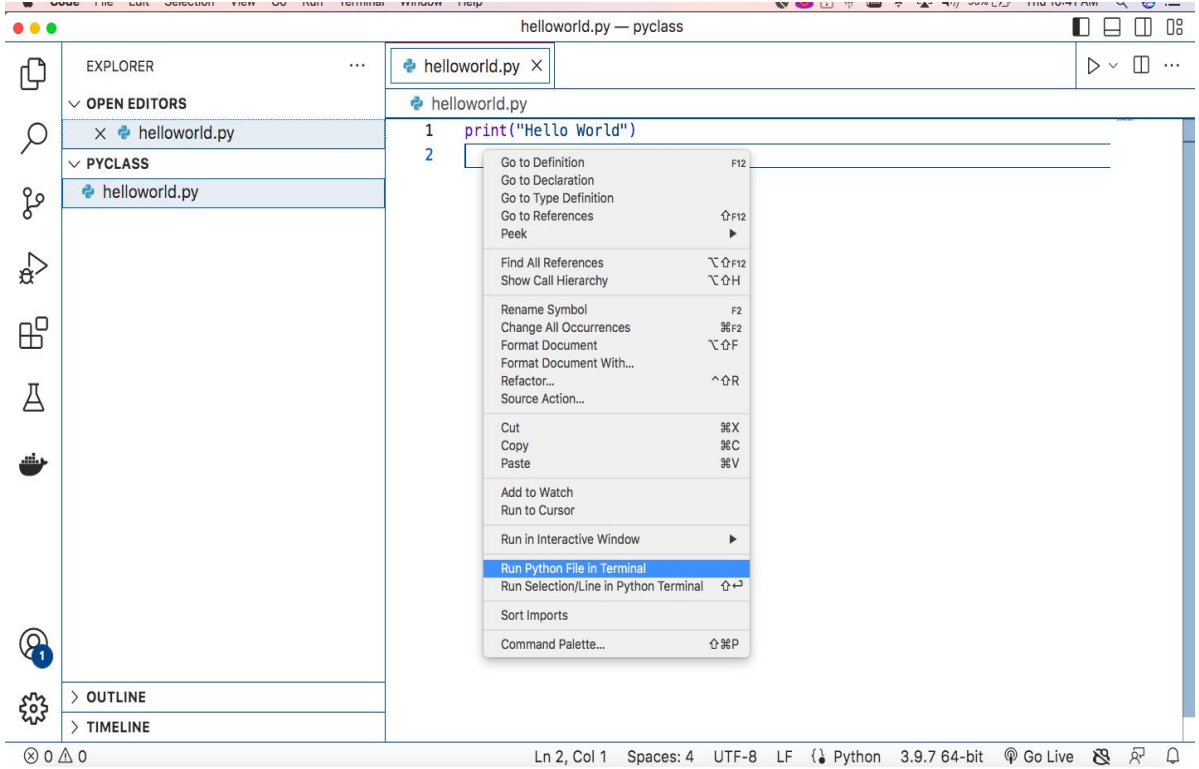
A list of Python interpreters will be displayed. Select the Python interpreter that you want to use.



Write your first Python programme



Run your first Python programme



Run your first Python programme : Output

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python

```
Dasuns-MacBook-Air:pyclass dasunathukorala$ /usr/local/bin/python3 /Users/dasunathukorala/Downloads/pyclass/helloworld.py
Hello World
Dasuns-MacBook-Air:pyclass dasunathukorala$
```

Jupyter Notebook

Jupyter is a free, open-source, interactive web tool known as a ***computational notebook****, which researchers can use to combine software code, computational output, explanatory text and multimedia resources in a single document.*

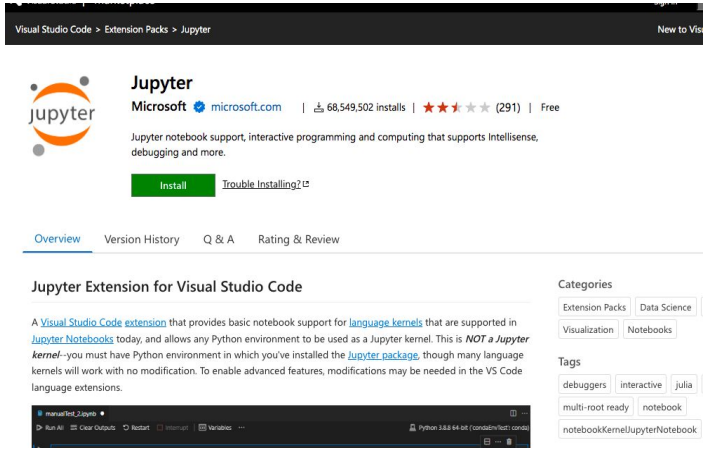
https://jupyter.org/

Jupyter Notebook on VS Code

Since we have already installed Python extension we need to install only Jupyter Notebook extension.

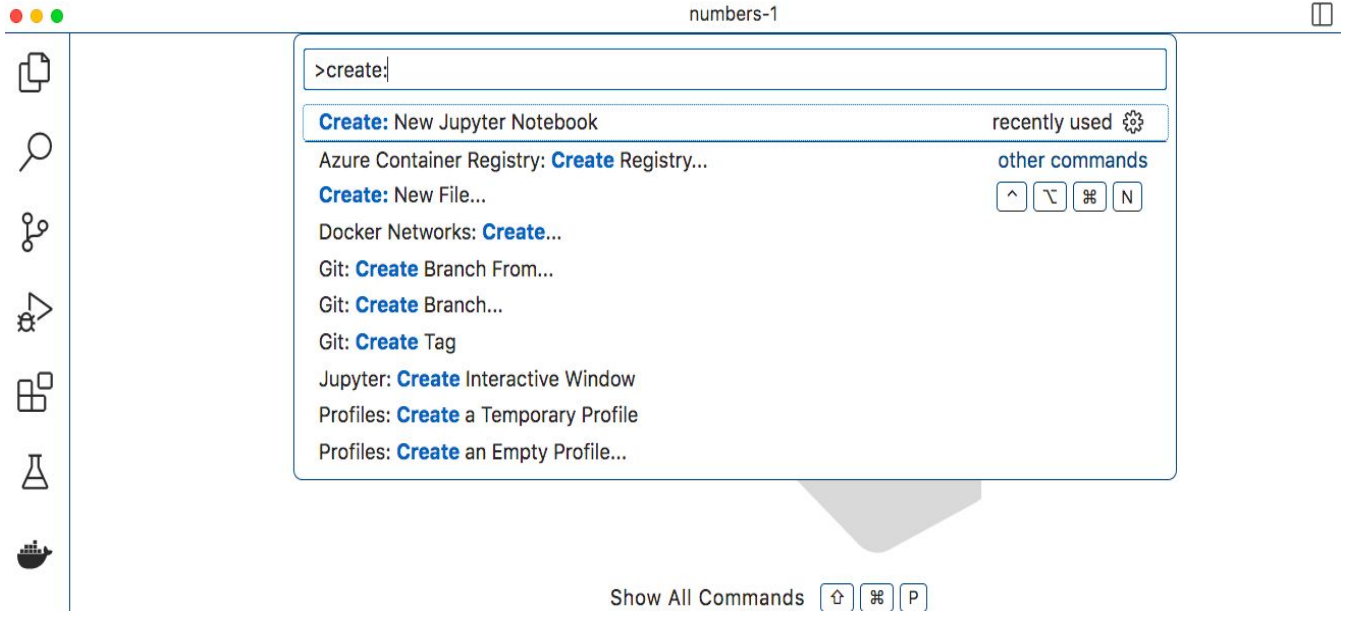
Download URL :

https://marketplace.visualstudio.com/items?itemName=ms-toolsai.jupyter



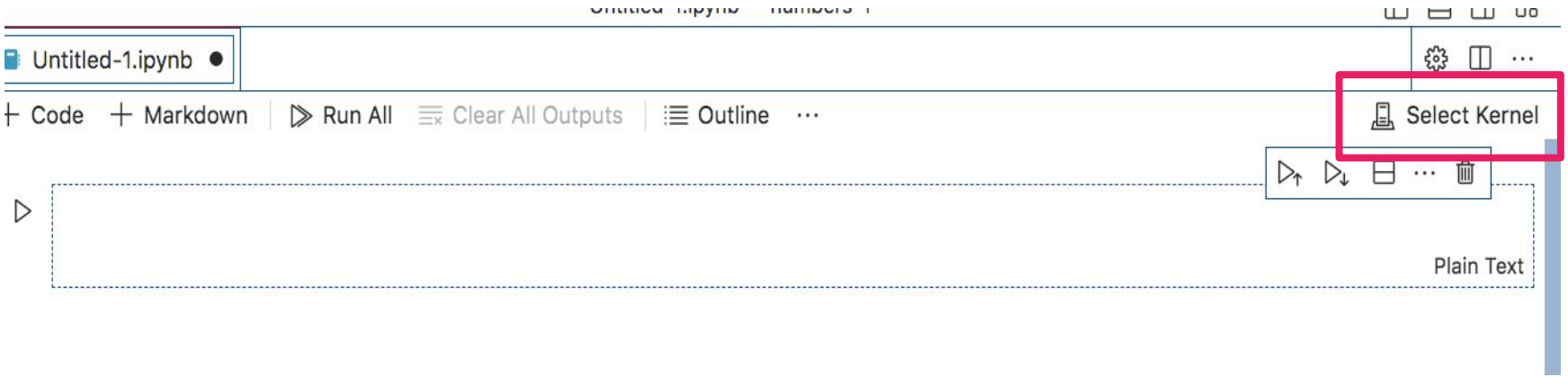
Jupyter Notebook on VS Code

You can create a Jupyter Notebook by running the **Create: New Jupyter Notebook** command from the **Command Palette**.



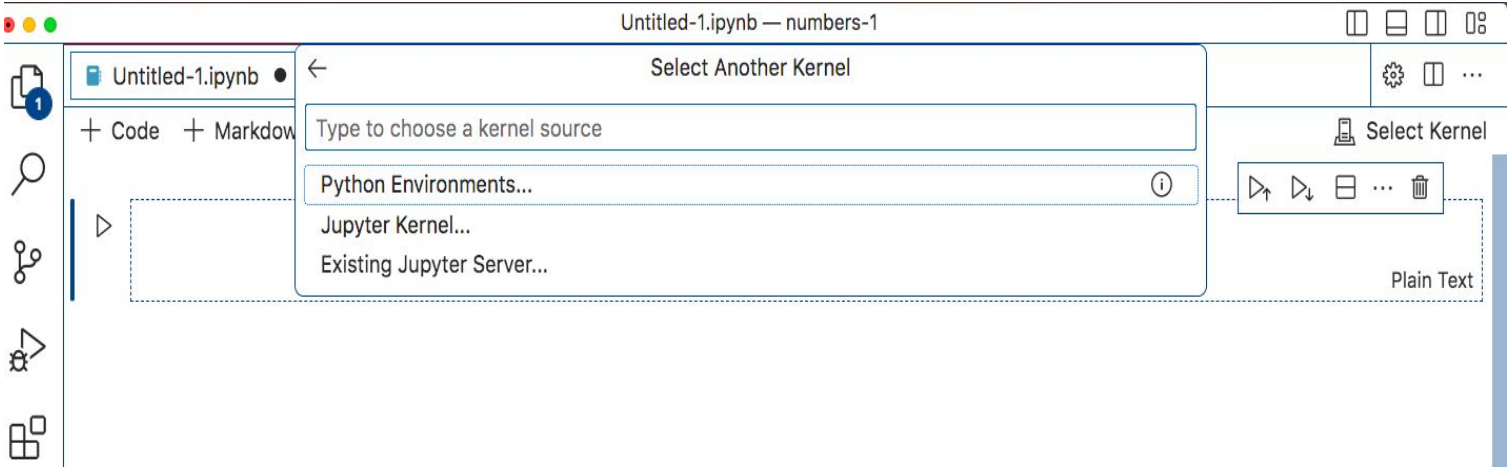
Jupyter Notebook on VS Code

Next, select a **kernel** using the kernel picker in the top right.



Jupyter Notebook on VS Code

Select the **Python environment** you want to use for running your Jupyter Notebook code.



Jupyter Notebook on VS Code

After selecting a kernel, the language picker located in the bottom right of each code cell will automatically update to the language supported by the kernel.

Data Types & Operators

Syntax for creating a variable (in Python)

```
variable_name = value
```

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

Python is dynamically typed

Python is dynamically typed, so you don't need to specify the data type explicitly; Python will infer the data type based on the value assigned to the variable.

```
age = 30          # Integer variable

name = "John"     # String variable

is_student = True   # Boolean variable

pi_value = 3.14   # Floating-point variable
```

Rules for naming Python Variables

- A variable name **must start with a letter or the underscore character**
- A variable name **cannot start with a number**
- A variable name can only contain **alpha-numeric characters** and **underscores** (A-z, 0-9, and _)
- Avoid using Python keywords & reserved words:

  Ex: **if, else, while, for**

Rules for naming Python Variables

- Python variable names should be **lowercase**, with words separated by underscores as necessary to improve readability.(Ref: PEP 8 style guide : https://peps.python.org/pep-0008/)

  Ex: **my_variable_name**
  total_volume
  number_of_items

Rules for naming Python Variables

- The PEP 8 style guide also recommends that Python variable names should be descriptive.

This means that the name of the variable should give you a good idea of what the variable is storing.

For example, the variable name `total_volume` is **more descriptive** than the variable name `volume`.

Rules for naming Python Variables

- If you have **constants** (variables that should never change their value), use all **uppercase letters** with underscores separating words.

  Ex: **PI_VALUE**, **MAXIMUM_ATTEMPTS**

Rules for naming Python Variables

```
legal variable names:
newvariable = "John" (not illegal, but not good)
new_variable = "John"
_new_variable = "John"
newVariable = "John" (not illegal, but not good)
NEW_VARIABLE = "John" (not illegal, but not good)
newvariable2 = "John" (not illegal, but not good)

Illegal variable names:
2newvariable = "John"
new-variable = "John"
new variable = "John"
```

Python Data Types

In computer programming, data types specify the type of data that can be stored inside a variable.

```
num = 25
```

Here, **25 (an integer)** is assigned to the **num** variable. So the data type of **num** is of the **int class**.

Built-in data types in Python

- **Numeric Types**: int, float, complex
- **String data types**: str
- **Sequence types**: list, tuple, range
- **Binary types**: bytes, bytearray, memoryview
- **Mapping data type**: dict
- **Boolean type**: bool
- **Set data types**: set, frozenset

Python Numeric Data Type

In Python, numeric data type is used to hold numeric values.

- **int** - holds **signed** integers of **non-limited length**.
- **float** - holds floating decimal points and it's accurate up to **15 decimal places**.
- **complex** - holds complex numbers.

Python Numeric Data Type

```
#create a variable with integer value

a = 200

print("The type of variable having value", a," is ", type(a))

#create a variable with float value

b = 12.57467

print("The type of variable having value", b," is ", type(b))
```

Python Numeric Data Type

```
#create a variable with complex value

c = 100+5j

print("The type of variable having value", c," is ", type(c))
```

Python Numeric Data Type

Float can also be scientific numbers with an "e" to indicate the power of 10.

```
x = 25e4   # 25x10^4
y = -67.6e100


print("variable x is type of ", type(x))
print("variable y is type of ", type(y))
```

Python Numeric Data Type Conversion

You can convert from one type to another with the **int()**, **float()**, and **complex()** methods:

```
x = 7     # int

y = 4.8   # float

z = 3j    # complex
```

Python Numeric Data Type Conversion

```
#convert from int to float

a = float(x)

#convert from float to int

b = int(y)

#convert from int to complex

c = complex(x)
```

Python Collections (Arrays)

There are four collection data types in the Python programming language:

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered, unchangeable, and unindexed. No duplicate members.
- **Dictionary** is a collection which is ordered and changeable. No duplicate members.

Python Lists

Lists are used to store multiple items in a single variable.

Lists are created using square brackets:

```
my_list = ["dog", "cat", "lion", "elephant"]
print(my_list )
```

List items are indexed, the first item has index [0], the second item has index [1] etc.

Python Lists are **ordered**.

If you add new items to a list, the new items will be placed at the end of the list.

Python lists are **changeable**. That means we can **modify, add, and remove** items in a list after it has been created.

Python Lists

Python list are also **mutable**.

Mutable means you can change the data or value it holds without changing its memory address.

```
my_list = [1, 2, 3, 4, 5]
print(id(my_list))  # Print the id of the list


my_list[0] = 10
print(id(my_list))  # The id will remain the same after modification
```

```
Output:
4398605696
4398605696
```

Python Lists

Lists **allow duplicates**. Since lists are indexed, lists can have items with the same value:

```
my_list = ["dog", "cat", "lion", "elephant", "cat"]
print(my_list )
```

To determine how many items a list has, use the **len()** function:

```
print(len(my_list))
```

List items can be of any data type:

```
list1 = ["dog", "lion", "elephant"]
list2 = [1, 6, 13, 11, 3]
list3 = [True, False, False]
```

A list can contain different data types:

```
my_list = ["xyz", 22, True, 50, "cat"]
```

Python Lists

It is also possible to use the list() constructor when creating a new list.

```
my_list = list(("cat", "lion", "dog")) # note the double round-brackets
print(my_list)
```

Access Items

List items are indexed and you can access them by referring to the index number:

Print the second item of the list:

```
my_list = ["dog", "cat", "lion", "elephant", "cat"]
print(my_list[1])
```

Negative indexing means start from the end. -1 refers to the last item, -2 refers to the second last item etc.

Print the last item of the list: `print(my_list [-1])`

Python Lists

Range of Indexes

You can specify a range of indexes by specifying where to start and where to end the range.

When specifying a range, the return value will be a new list with the specified items.

```
my_list = ["dog", "cat", "lion", "elephant", "rat",”fish”]

print(my_list[2:5])
```

Note: The search will start at index 2 (included) and end at index 5 (not included).

By leaving out the start value, the range will start at the first item:

```
print(my_list[:3])
```

By leaving out the end value, the range will go on to the end of the list:

```
print(my_list[2:])
```

Python Lists

Change List Items

To change the value of a specific item, refer to the index number:

```
my_list = ["dog", "cat", "lion", "elephant", "rat",”fish”]
my_list[1] = “tiger”
print(my_list)
```

Change the values "cat" and "lion" with the values "zebra" and "tiger":

```
my_list = ["dog", "cat", "lion", "elephant", "rat",”fish”]
my_list[1:3] = ["zebra", "tiger"]
print(my_list)
```

Python Lists

Insert Items

To insert a new list item, without replacing any of the existing values, we can use the insert() method.

The insert() method inserts an item at the specified index:

```
my_list = ["dog", "cat", "lion", "elephant", "rat",”fish”]

my_list.insert(2, "zebra")

print(my_list)
```

Append Items

```
my_list = ["dog", "cat", "lion", "elephant", "rat",”fish”]

my_list.append("zebra")

print(my_list)
```

Python Lists

Extend List

To append elements from *another list* to the current list, use the extend() method.

Add the elements of your_list to my_list:

```
my_list = [“dog", "cat", "lion"]
your_list = [“elephant", "rat",”fish”]
my_list.extend(your_list)
print(my_list)
```

Remove Specified Item

The remove() method removes the specified item.

```
my_list = [“dog", "cat", "lion"]
my_list.remove("cat")
print(my_list)
```

Python Lists

Remove Specified Index

The pop() method removes the specified index.

Add the elements of your_list to my_list:

```
my_list = [“dog", "cat", "lion""rat",”fish”]
my_list.pop(1)
print(my_list)
```

The del keyword also removes the specified index:

```
del my_list[0]
```

The del keyword can also delete the list completely.

```
del my_list
```

Python Lists

Clear the List

The clear() method empties the list. The list still remains, but it has no content.

```
my_list = [“dog", "cat", "lion""rat",”fish”]
my_list.clear()
print(my_list)
```

Python Lists

Sort Lists

List objects have a sort() method that will sort the list alphanumerically, ascending, by default:

Sort the list alphabetically:

```
my_list = [“dog", "cat", "lion""rat",”fish”]
my_list.sort()
print(my_list)
```

Python Lists

Sort the list numerically:

```
my_list = [100, 50, 65, 82, 23]
my_list.sort()
print(my_list)
```

In **alphanumeric sorting,**

- **0–9 come before A–Z.**
- **A–Z come before a–z (uppercase is smaller than lowercase in ASCII/Unicode order).**

Python Lists

Sort Descending

To sort descending, use the keyword argument reverse = True:

```
my_list = [“dog", "cat", "lion""rat",”fish”]
my_list.sort(reverse = True)
print(my_list)
```

Python Lists

```
num_list = [109, 60, 45, 72, 14]
num_list.sort(reverse = True)
print(num_list)
```

Case Insensitive Sort

By default the sort() method is case sensitive, resulting in all capital letters being sorted before lower case letters:

So if you want a case-insensitive sort function, use str.lower as a key function:

```
my_list = [“dog", "cat", "Lion""Rat",”fish”]
my_list.sort(key = str.lower)
print(my_list)
```

Python Lists

The **reverse()** method reverses the current sorting order of the elements.

```
my_list = [“dog", "cat", "lion""rat",”fish”]
my_list.reverse()
print(my_list)
```

Copy a List

You cannot copy a list simply by typing list2 = list1, because: list2 will only be a *reference* to list1, and changes made in list1 will automatically also be made in list2.

There are ways to make a copy, one way is to use the built-in List method copy().

```
my_list = [“dog", "cat", "lion""rat",”fish”]
new_list = my_list.copy()
print(new_list)
```

Python Lists

Another way to make a copy is to use the built-in method list().

```
my_list = [“dog", "cat", "lion""rat",”fish”]
new_list = list(my_list)
print(new_list)
```

Join Two Lists

There are several ways to join, or concatenate, two or more lists in Python.

One of the easiest ways are by using the + operator.

```
list_1 = ["a", "b", "c"]        list_2 = [1, 2, 3]

list_3 = list_1 + list_2

print(list3)
```

Python Lists

Or you can use the extend() method, which purpose is to add elements from one list to another list:

In the following example extend() method is used to add list2 at the end of list1:

```
list_1 = ["a", "b" , "c"]
list_2 = [1, 2, 3]

list_1.extend(list_2)
print(list_1)
```

Thank You…!


“FAILURE IS AN OPTION HERE. IF THINGS ARE NOT FAILING, YOU ARE NOT INNOVATING ENOUGH.”
Elon Musk
co-founder of PayPal and Tesla Motors