

HELLO PYTHON! INTRODUCTION TO PYTHON AND PYTHON DEVELOPMENT SETUP

Introduction to Data Science

Session 1

Marysia Tańska

Peckham Digital Accelerator Zone

14th January 2025

WELCOME TO INTRO TO DATA SCIENCE!

- Hi students! After the lecture, you'll meet your personal tutors for the course.
- If you need help, please remember your tutors have weekly office hours
- you can email them for a 15-minute online tutorial in-between classes, at an agreed time.
- Next week, we'll discuss the project brief for this module.

THE TEACHING TEAM



Elena Petrova
she/her



Yadira Sanchez Benitez
she/elles



Dan Hearn
he/him



Rosie Walker
she/her



Marysia Tańska
she/her

Introduction to Data Science, Session 1.
Peckham DAZ Programme, 14th January 2025

***WHAT WE WILL LEARN
IN MODULE 2: INTRODUCTION TO DATA SCIENCE
- AND WHERE TO FIND THE MATERIALS***

<https://github.com/peckham-daz/24-intro-to-data-science>

WHAT WE WILL LEARN TODAY

- what is **Python**
- when you may want to use Python and when maybe not
- how to set up a Python development environment on your computer
- what a **virtual environment** is and how to use it
- what a **Version Control System** is and how to manage code versions with **git** and **GitHub**
- how to use **Visual Studio Code** and how to connect it to GitHub
- **basic operations in Python** using the **console** and a **Jupyter Notebook**

BEFORE WE START

Today's lecture may be a lot to take in.

Please make sure you can access these **slides and the coding demo for reference.**

Ask your tutors as many questions as needed - that's why they're here!

You and your colleagues may have **different levels of experience** with coding, so don't be stressed if you need more time than some other students - it takes practice to learn :)

This course is designed so that at the end of this module you should be able to make your own Python project, with a bit of practice in class and at home.

PYTHON

WHAT IS PYTHON?

- a **programming language**
- **high-level** - it mostly interacts with C “under the hood”
- **not typed** (unless you install a library for typing)
- **garbage-collected** - you mostly don't need to worry about memory management
- it often uses **libraries** - bundles of code published by other people, allowing you to easily reuse the code for specific purposes
- Python is maintained by the **Python Software Foundation**
<https://www.python.org/psf-landing/>

WHAT CAN YOU DO WITH PYTHON?

- work with large amounts of **data: editing, analysis, visualisation**
- training and using **Machine Learning models**
- server-side operations - **Back-End development**, for example interacting with databases, automating data operations
- **interacting with APIs** (Application Programming Interfaces) - we will learn more about this later
- **almost anything else**, keeping in mind it may be better for some uses than others

HOW CAN YOU USE PYTHON?

- directly from the **console**
- you can run Python's **scripts** (.py files)
- through an **API** - we will learn more about this later during this module
- through an **interface** using a library called IPython - from the console or a **Jupyter notebook**

All of these methods require Python installation on the machine running Python code.

You can also run it online via Google Collab. This method does not require Python installation on your machine but provides limited functionality.

***LET'S RUN SOME PYTHON CODE
IN THE CONSOLE***

WHEN YOU MAY WANT NOT TO USE PYTHON?

- **things that need to run smoothly in the browser** on someone else's machine - because Python is not built into browsers (unlike JavaScript), and if you run things on the server it may be slow.
- **real-time audio, video, image processing** - it will likely be too slow. There are some projects that do that but you may want to consider other options such as JavaScript, Processing with Java, OpenFrameworks with C++, tidalcycles, etc.

MOST POPULAR DATA TYPES IN PYTHON

- **str** - string `'one', 'two', 'November', '1'`
- **int** - integer `1, 2, 524`
- **float** - floating point number `1., 2.98, 5213.83434`
- **bool** - Boolean value `True, False`
- **list** `[1, 'cat', 3.14, False, {'rating': 10}]`
- **dict** - dictionary `{'item': 'book', 'title': 'Ways of Being', 'colour': 'red', 'price': 17.99}`

MORE ON BUILT-IN DATA TYPES

https://www.w3schools.com/python/python_datatypes.asp

WHAT IS A VARIABLE?

WHAT IS A VARIABLE?

You can think of it as a container for data.

a = 3

my_name = 'Marysia'

PYTHON OPERATORS

https://www.w3schools.com/python/python_operators.asp

CODING DEMO

The code will be here after the class:

https://github.com/peckham-daz/24-intro-to-data-science/blob/main/session06/session06_coding_demo.ipynb

***PYTHON - WHERE TO LOOK
FOR HELP AND RESOURCES***

LINKS TO PYTHON RESOURCES ON GITHUB

https://github.com/peckham-daz/24-intro-to-data-science/blob/main/resources/python_resources.md

USING THE HELP FUNCTION IN PYTHON

```
import numpy as np
help(np.array)

[3] ✓ 0.0s

... Help on built-in function array in module numpy:

array(...)
    array(object, dtype=None, *, copy=True, order='K', subok=False, ndmin=0,
          like=None)

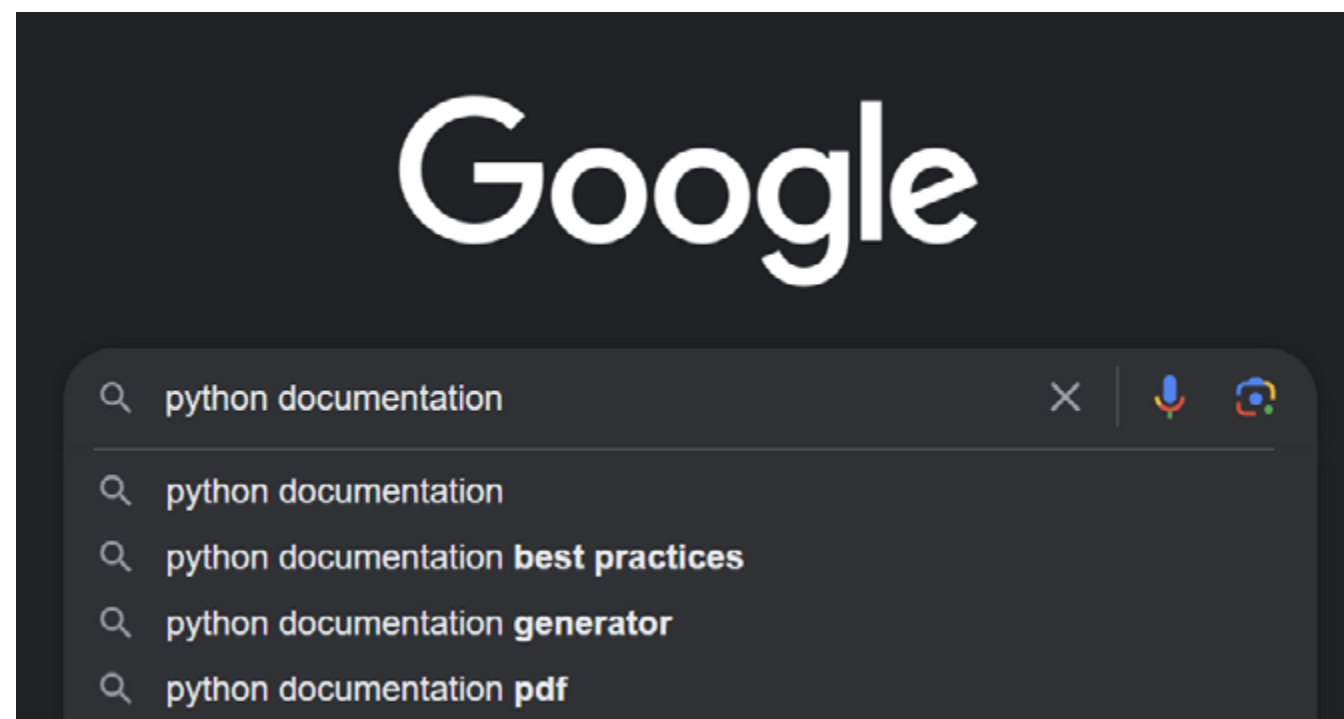
    Create an array.

    Parameters
    -----
    object : array_like
        An array, any object exposing the array interface, an object whose
        __array__ method returns an array, or any (nested) sequence.
        If object is a scalar, a 0-dimensional array containing object is
        returned.
    dtype : data-type, optional
        The desired data-type for the array. If not given, then the type will
        be determined as the minimum type required to hold the objects in the
        sequence.
    copy : bool, optional
        If true (default), then the object is copied. Otherwise, a copy will
        only be made if __array__ returns a copy, if obj is a nested sequence,
        or if a copy is needed to satisfy any of the other requirements
        (`dtype`, `order`, etc.).
    order : {'K', 'A', 'C', 'F'}, optional
    ...

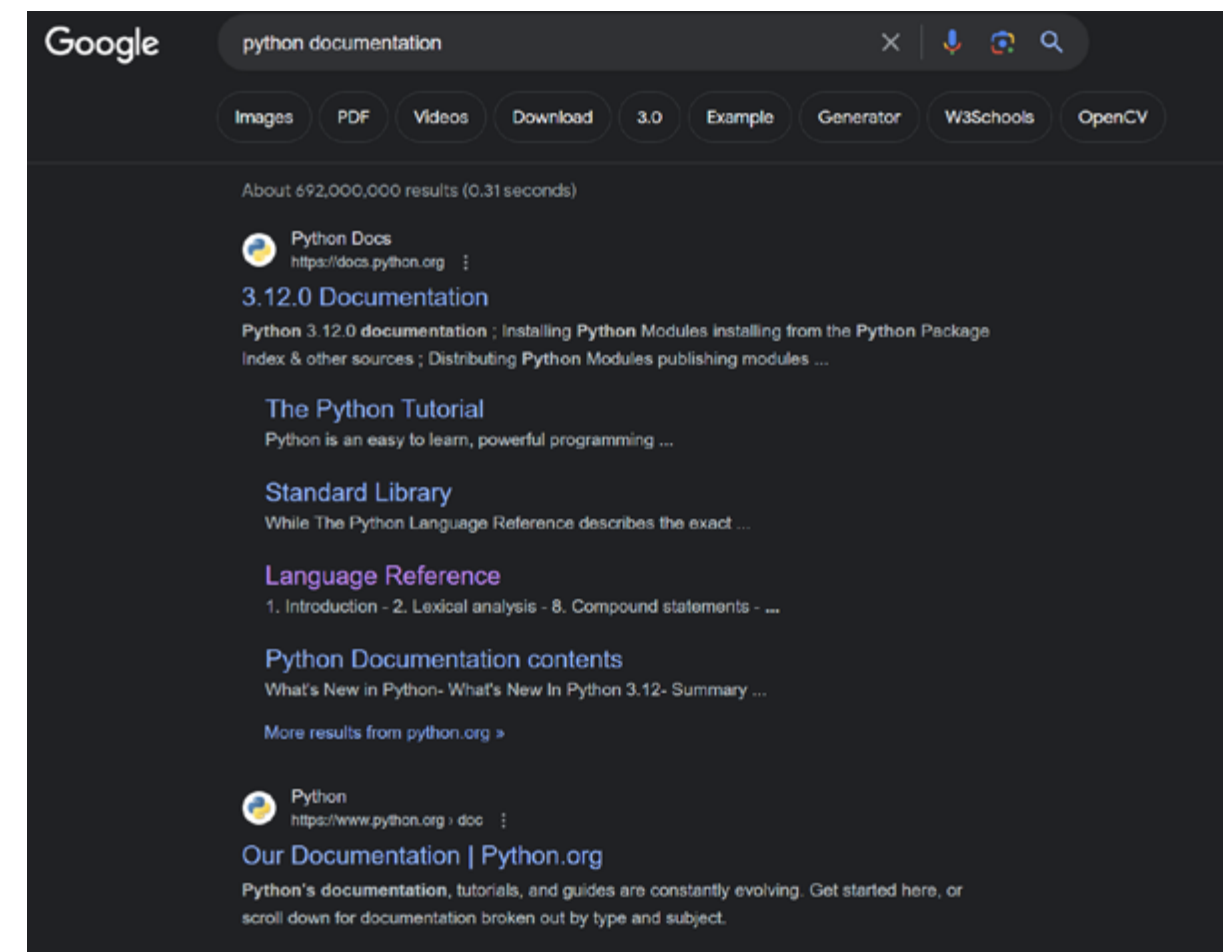
    >>> np.array(np.mat('1 2; 3 4'), subok=True)
    matrix([[1, 2],
            [3, 4]])

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

OFFICIAL DOCUMENTATION FOR PYTHON OR ANY SELECTED LIBRARY

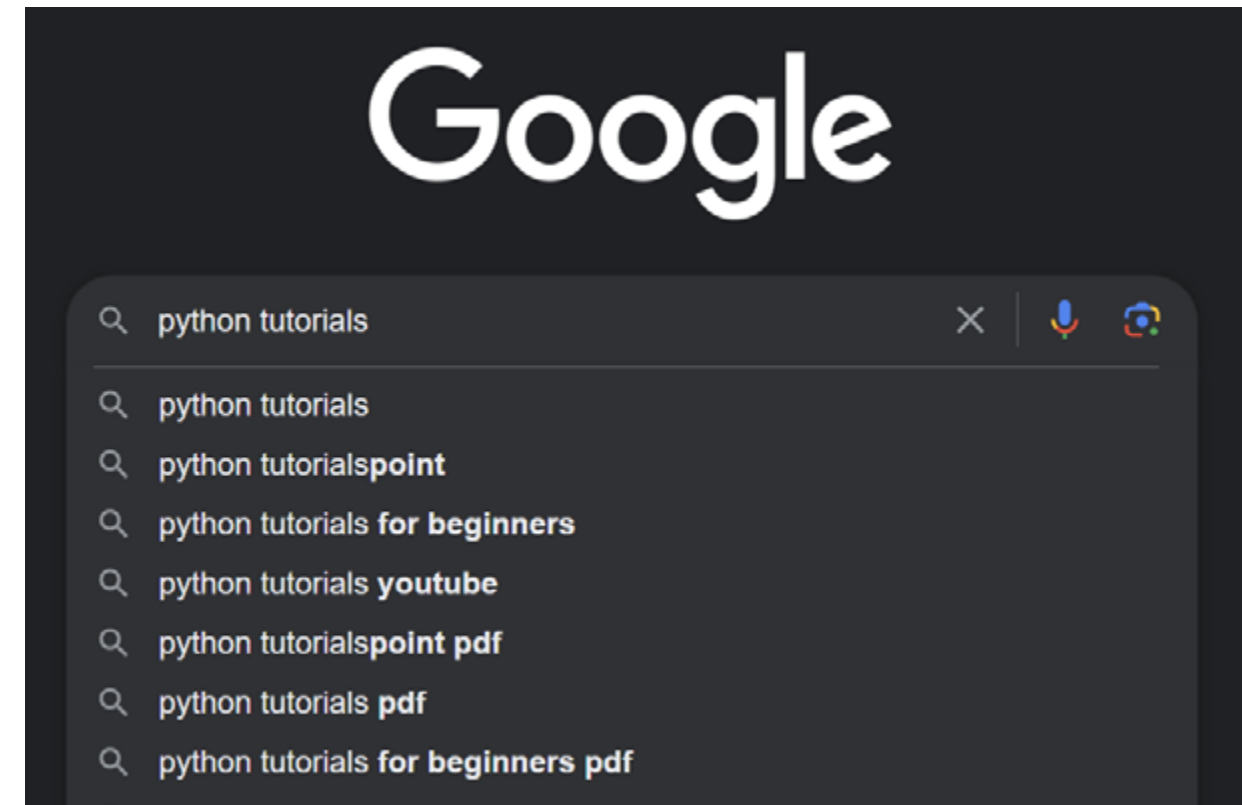
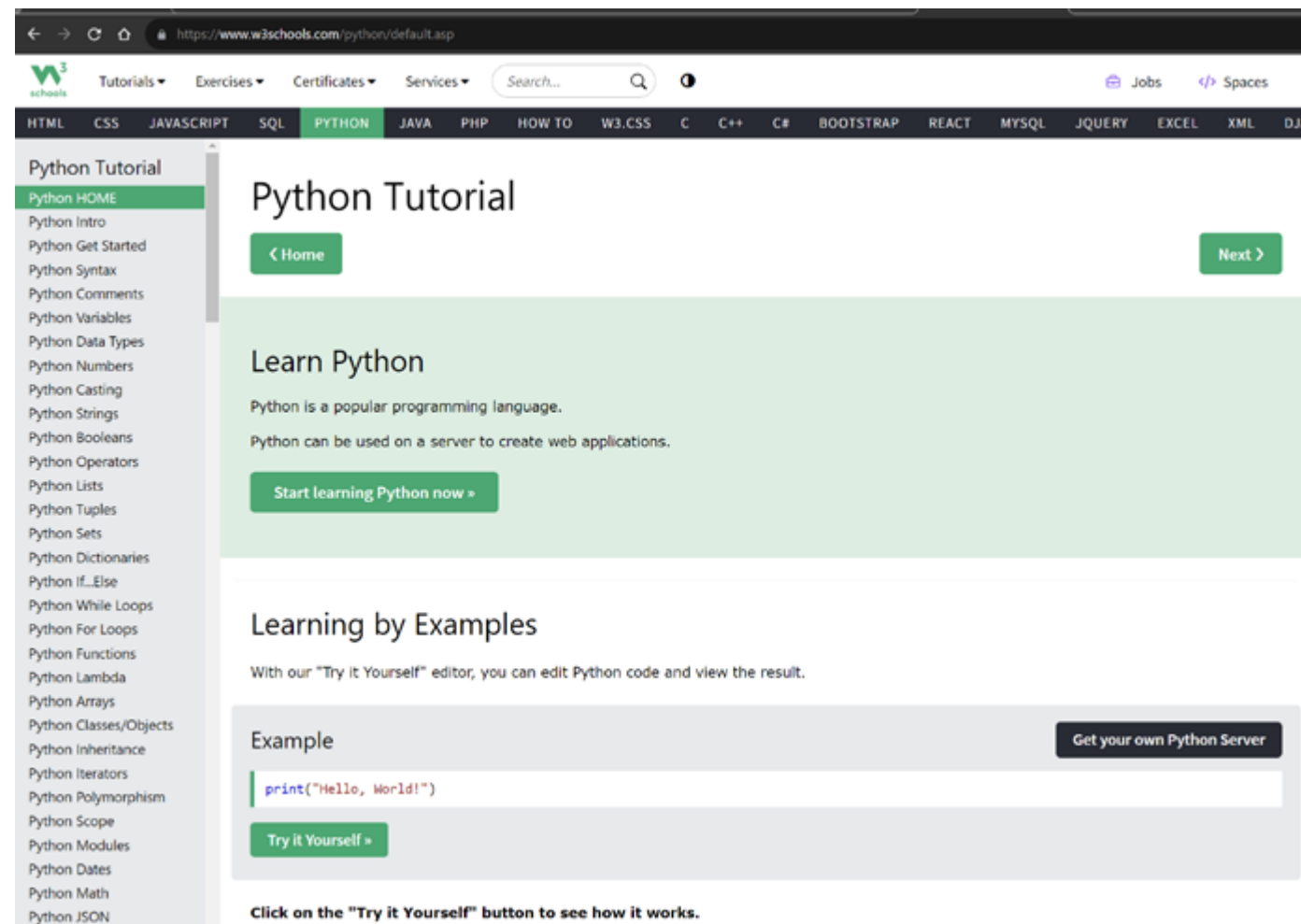


1. Go to the search engine of your choice.



2. It's most likely here :)

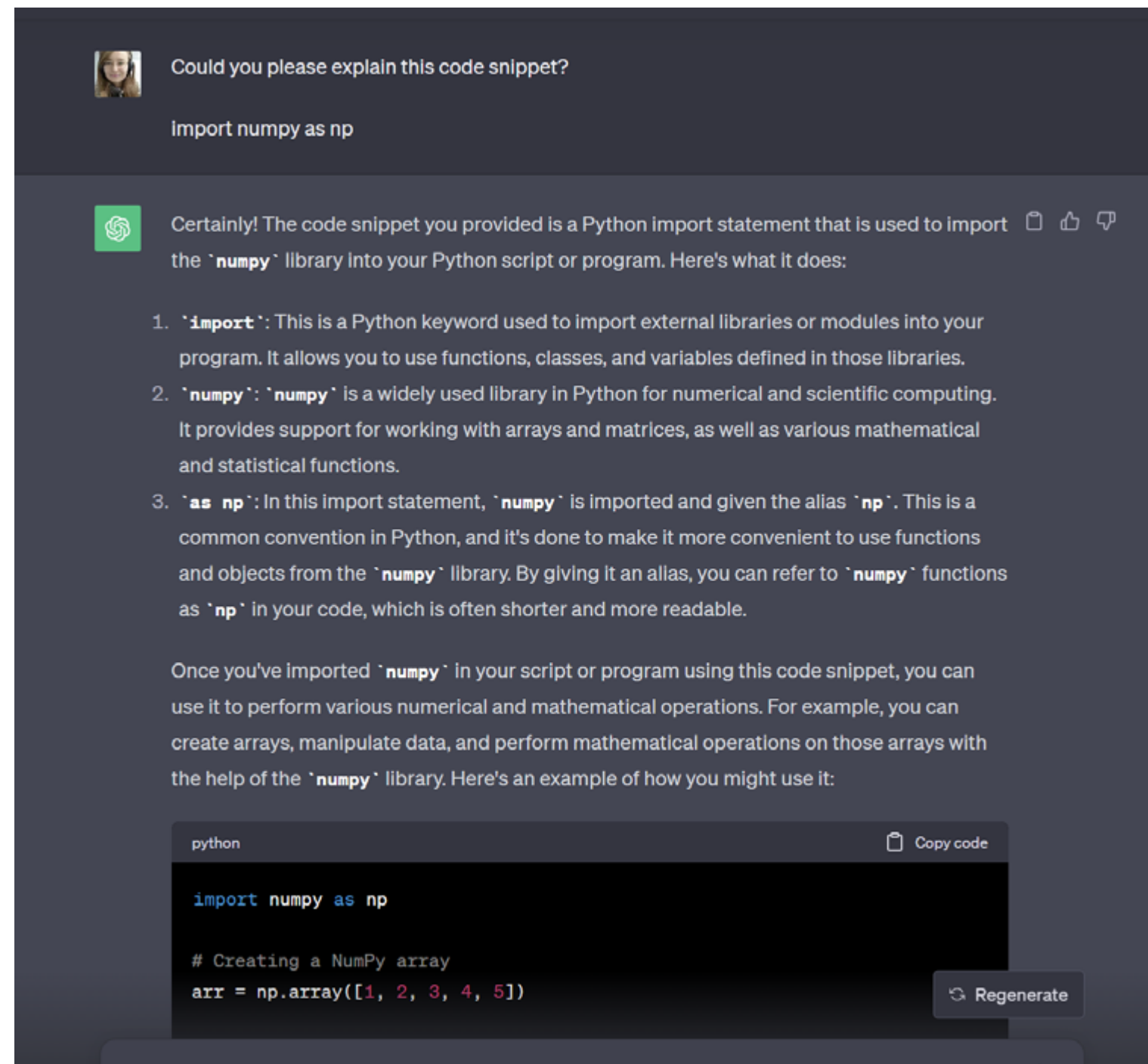
SELF-STUDY WEBSITES, E.G. W3SCHOOLS




1. <https://www.w3schools.com/python/>

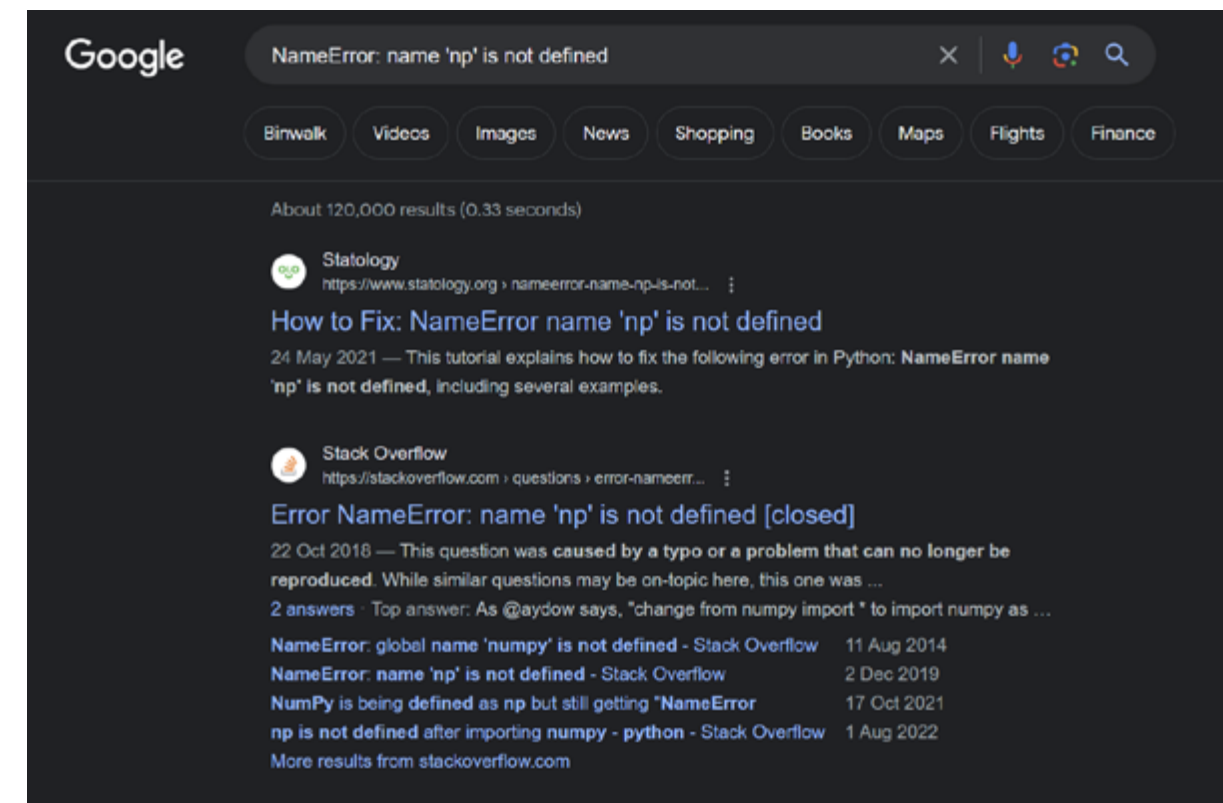
2. You can find more here :)

***FEEL FREE TO ASK A LARGE LANGUAGE MODEL,
SUCH AS CHATGPT, FOR HELP
- BUT PLEASE USE PROMPTS SUCH AS “PLEASE EXPLAIN THIS
CODE...” RATHER THAN “GENERATE CODE” :)***



DE-BUGGING (FIXING ERRORS - BUGS)

```
[1]  0.2s  
...  
-----  
NameError                                Traceback (most recent call last)  
c:\coding_projects\cci ual\STEM-4-Creatives-23-24\STEM-Week-1.ipynb Cell 46 line 1  
----> 1 np.array(1, 0)  
  
NameError: name 'np' is not defined
```



1. Search for your errors (make sure not to paste custom code such as variable names).

2. It's most likely there. Stack Overflow is usually helpful!

DE-BUGGING (FIXING ERRORS - BUGS)

```
[3] a + b
    ⓧ 0.0s
...
-----
TypeError                                Traceback (most recent call last)
c:\coding_projects\cci ual\STEM-4-Creatives-23-24\STEM-Week-1.ipynb Cell 48 line 1
----> 1 a + b

TypeError: can only concatenate str (not "int") to str
```

Read the error message - they can be quite informative! Here, it says one of the values is a string data type and the other one an integer, which is why you cannot perform the operation.

DE-BUGGING (FIXING ERRORS - BUGS)

```
[3] a + b
    ⓧ 0.0s

... -----
TypeError                                Traceback (most recent call last)
c:\coding_projects\cci ual\STEM-4-Creatives-23-24\STEM-Week-1.ipynb Cell 48 line 1
----> 1 a + b

TypeError: can only concatenate str (not "int") to str

print(a)
print(type(a))
print(b)
print(type(b))

[7] ✓ 0.0s

... hello world
    <class 'str'>
    100
    <class 'int'>
```

If not sure - print the values and data types!

***SEARCHING FOR INFORMATION
IS A VERY IMPORTANT CODING SKILL
- PROFESSIONAL PROGRAMMERS DON'T
REMEMBER IT ALL!***

PYTHON CONVENTIONS

FORMATTING AND NAMING

- IS IT IMPORTANT?

NAMING CONVENTION

- snake_case and camelCase

```
CONSTANT_VALUE = 'a' # constant values on top of the file / cell  
  
my_variable = 1.  
  
def python_function():  
    return 0.  
  
class PythonClass():  
    def __init__(self):  
        pass
```

USE MEANINGFUL NAMES

```
my_list = ['Daisy', 'Rose', 'Violet']  
alpaca_names = ['Daisy', 'Rose', 'Violet']  
birthday_party_flower_choices = ['Daisy', 'Rose', 'Violet']
```


WHEN POSSIBLE, USE LOGICAL ORDER AND SPLIT LINES

```
# is 'Poppy' on the list?
ALPACA_NAMES = ['Rose', 'Violet', 'Sunflower', 'Daisy', 'Margaret', 'Daffodil', 'Lily', 'Jasmine', 'Iris']

ALPACA_NAMES = ['Daffodil', 'Daisy', 'Iris', 'Jasmine', 'Lily', 'Margaret',
                'Rose', 'Sunflower', 'Violet']
```

```
# is 'Poppy' on the list?
ALPACA_NAMES = [
    'Daffodil',
    'Daisy',
    'Iris',
    'Jasmine',
    'Lily',
    'Margaret',
    'Rose',
    'Sunflower',
    'Violet',
]
```

**Constant values - often CAPITALISED and on top of the file -
easy to find & change**

SAME RULES APPLY TO IMPORTS :)

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

from math import (
    atan,
    cos,
    sin,
    tan,
)
from seaborn import heatmap
```

A LOT OF LIBRARIES HAVE TYPICAL IMPORT NAMES

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import tensorflow as tf
```

Use functions - it's easier to select a piece of code to paste into another file

**Use git and save multiple versions
(if you think it's useful)**

Questions? :)

GIT ***VERSION CONTROL SYSTEM***

WHAT IS A VERSION CONTROL SYSTEM?

- a piece of software which lets you **keep track of changes in your code** without saving multiple files
- when things go wrong, **you can get back to a previous version** - for selected files or the whole project
- it also allows for having **multiple versions of the same coding project** - these are referred to as **branches**
- you can also **collaborate with multiple people** on a project and then **coordinate the changes**
- it helps you **share your code online and update it**
- do not expect to be fluent in it right away - **it takes a lot of practice** to learn VCS and experienced developers still make mistakes :)

WHAT IS GIT?

- git is a **type of Version Control System**, it's a piece of software.
- there are many **platforms** which allow for **git integration**. Some of the most popular ones are GitHub, Git Lab and Bitbucket.
- we will be working with **GitHub**.
- sometimes companies and organisations have their own servers connected to these services, for example GitHub Enterprise. This increases code safety as it can only be shared internally.

HOW DO YOU USE GIT IN A CODING PROJECT?

- You need to have **git** installed.
- You need an **account** on an online platform which supports git, such as **GitHub**.
- Then, you need to log into the account on your computer. You can do it either through the **console**, an **IDE** (Integrated Development Environment), or through an additional piece of software such as **GitHub Desktop**.
- We will briefly go through the **git CLI** (Command Line Interface), and then go through the **GitHub integration in the Visual Studio Code IDE**.

MORE ABOUT GIT

Git CLI (command line interface) cheat sheet

<https://education.github.com/git-cheat-sheet-education.pdf>

What is Version Control?

<https://git-scm.com/video/what-is-version-control>

What is Git?

<https://git-scm.com/video/what-is-git>

LET'S DO A QUICK GIT DEMO!

PYTHON ***VIRTUAL ENVIRONMENTS***

WHAT IS A VIRTUAL ENVIRONMENT?

- **Python and Python libraries have many different versions.**
- A virtual environment is a “**box**” for the **Python version and library versions** that work well together.
- You can run Python projects without a virtual environment. Then, all the libraries will be “attached” to your default version of Python. But...

WHAT IS A VIRTUAL ENVIRONMENT?

- It is **good practice**, and it makes life easier in the long run, if you use **virtual environments**.
- Some libraries need other libraries to run. These are called **dependencies**.
- When you work on **different projects**, sometimes you **need different versions of Python and its libraries**.
- You want to **keep them separate** so that things don't break. Trust me, fixing mixed up library versions is not pleasant!
- You can **run similar projects with the same virtual environment**.
- I usually create **a new environment for every bigger project**. It also ensures that old projects still run even if new library versions are released.

HOW DO WE KEEP TRACK OF LIBRARY VERSIONS IN THE PROJECTS?

- You can install all libraries at once using a requirements file.
- It's a file with a list of all the libraries needed for a project.
- It may have library versions specified. If not, it will install the latest versions.
- You can also create a requirements file with a list of all library versions currently installed. You can then use it to re-create the virtual environment.
- You can also copy environments in-between projects.
- More on requirements files:
<https://learnpython.com/blog/python-requirements-file/>

LET'S SET UP PYTHON! (AFTER THE BREAK)

https://github.com/peckham-daz/24-intro-to-data-science/blob/main/environment_setup.md

