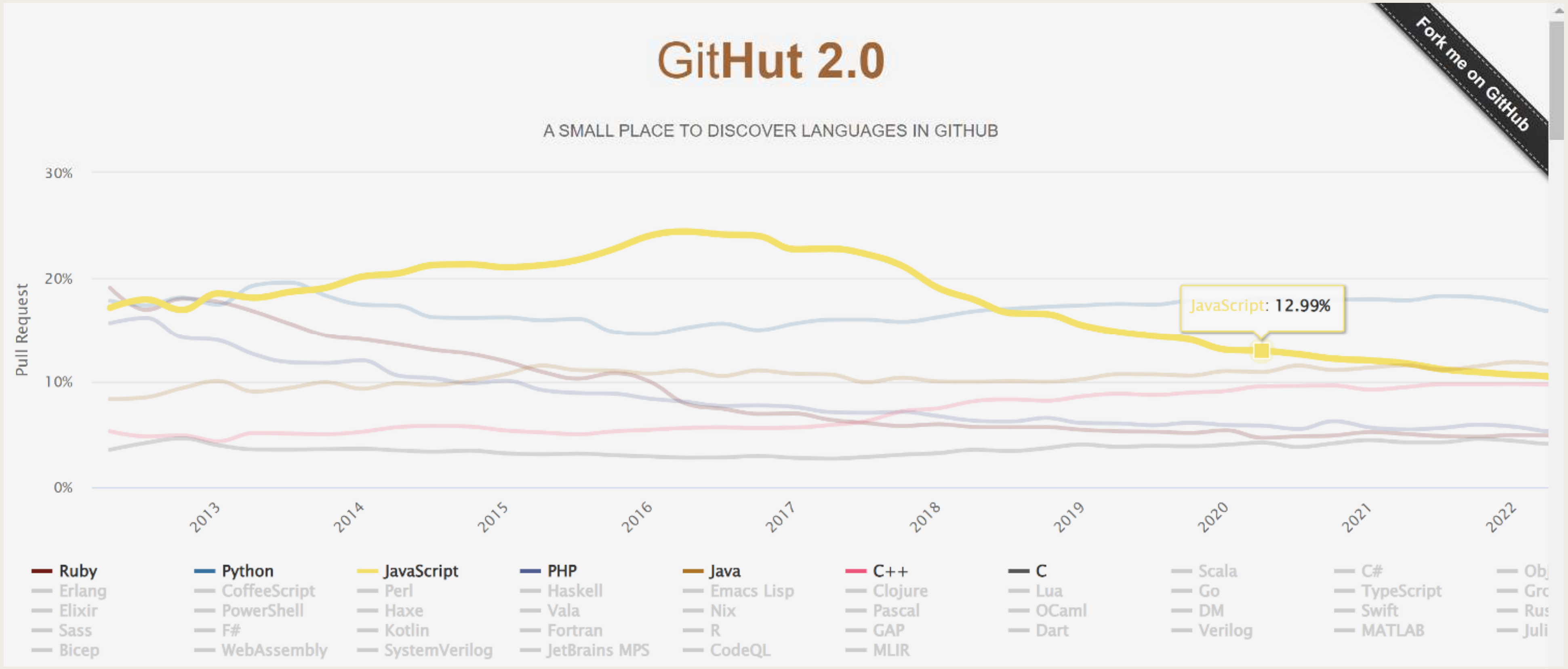
A BRIEF INTRODUCTION TO PYTHON

(With content based on the book “A Byte of Python”, by Swaroop [3])

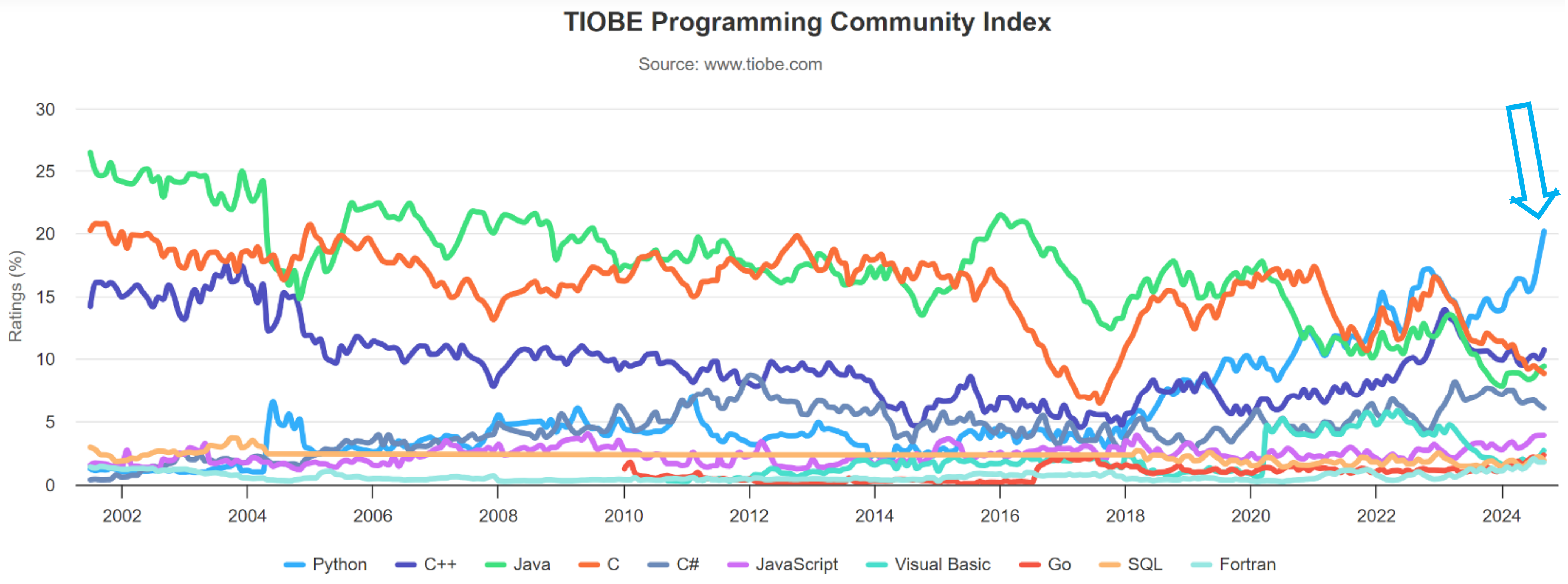
Python

- Guido van Rossum, the creator of the programming language, was inspired by the British comedy group Monty Python when naming it.
    - *has nothing to do with pythons, mainly because he is not a great connoisseur of snakes…*
- Python is currently one of the most popular and fastest-growing languages.
    - *in 2024 it was the 1nd most used language within the GitHub platform*
    - *Also, in Sep/2024 is already the 1st most popular language according to TIOBE index*
- Python is a programming language that can bring together the best of both worlds: simple (to learn and use) and powerful
- Python is used in all domains and application areas, including Data Science
    - *has increasingly become the language of choice in the Artificial Intelligence community and Machine Learning.*

Python on GitHub



Python popularity according tiobe index



Python Features

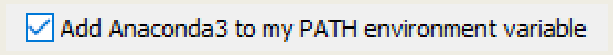
Python is an exciting and powerful language as it combines performance with simple and fun features.

- It is open access and open source
- It has a simple code to interpret (similar to pseudo-code).
- Easy to learn.
- High-level.
- Highly portable (if you avoid system resources, the same program runs on multiple platforms).
- It is an interpreted language (and convertible into intermediate code, the bytecode).
- Object-oriented (supports the paradigm in a simpler way than C++ and Java).- Allows easy extension of programs with code from other languages (you can encode C or C++ code snippets and use them in Python).
- Easily incorporated into C/C++ programs.
- Language extended through its powerful standard library and countless others (searchable in Python Package Index).

Anaconda installation

- Intending to install Python without integrating other specific modules, simply access https://www.python.org/downloads/ to download the latest version, Python 3.12.6, at the time of writing this text.
- However, instead of installing Python in isolation, it is suggested to use Anaconda, a Python distribution that integrates a set of libraries very useful for Machine Learning (ML),
  - *Otherwise, we would have to install these libraries later and separately.*
  - *Anaconda can be downloaded from https://www.anaconda.com/download/.*

    (At the installation, select the checkbox "Add Anaconda to my PATH environment variable" and accept the remaining default settings.)



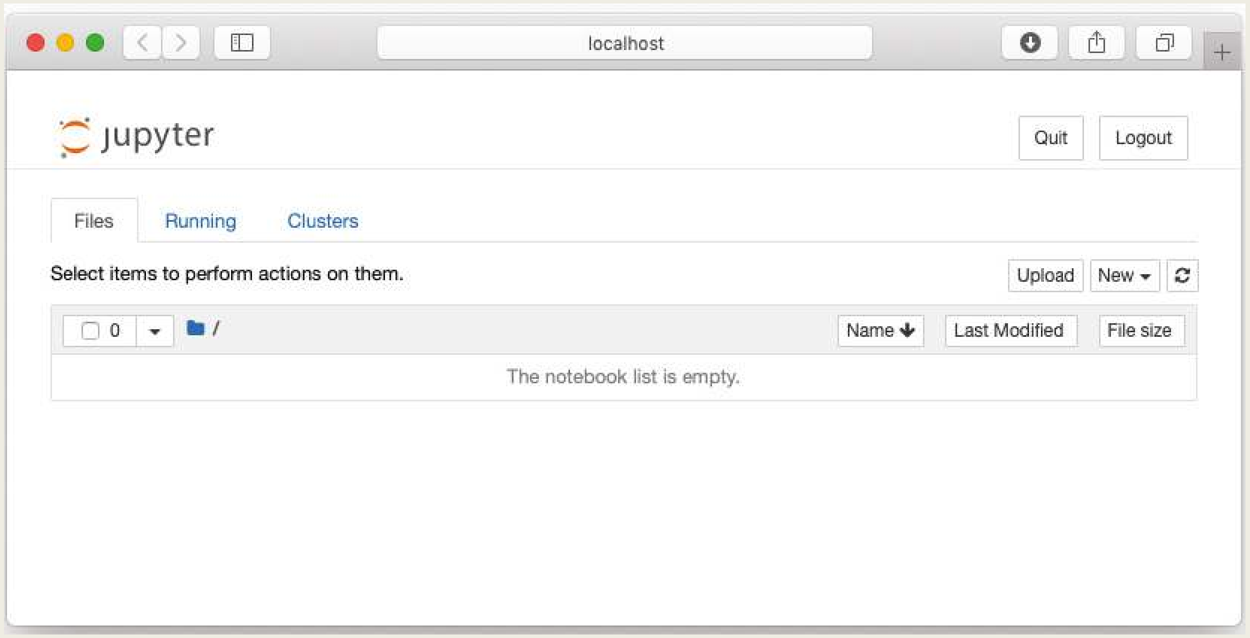
[✓] Add Anaconda3 to my PATH environment variable

- All ML models will be developed using the Scikit-learn library, which implements various types of ML algorithms: regression, classification, and clustering.
  - *In addition to Scikit-learn, other Python add-on libraries such as NumPy, Pandas, and Matplotlib will also be useful for ML.*
  - *Fortunately, the Anaconda distribution already includes all of these 4 libraries.*

Running Jupyter Notebook

- After installing Anaconda, we can use Jupyter Notebook, a web application that provides a graphical and interactive environment for creating and editing documents called notebooks. These notebooks can integrate code, documentation, graphics, and other resources.
    - *It is this application that is used in [1] to create and edit, in an interactive and didactic way, all ML solutions presented.*
- *To launch Jupyter Notebook in Windows,*
    - *run 'Anaconda Prompt'*
    - *inside its console window, type 'jupyter notebook'*

    (In Windows we may also directly call the 'jupyter notebook')



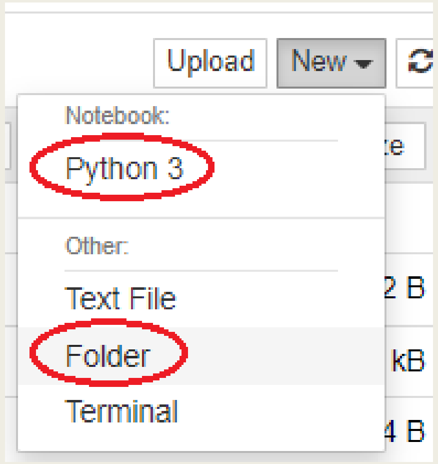
Select items to perform actions on them.

The notebook list is empty.

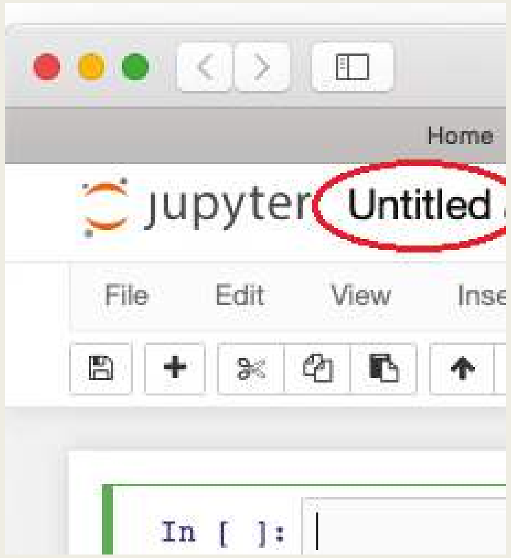
Getting started with the Jupyter Notebook

- Create a work folder by selecting 'Folder' on the 'New' button on the right side
    - *then change the name from 'Untitled Folder' to 'IA', for example, and click it to become the work folder.*

    (the 'AI' subfolder is created in the folder that contains the Anaconda installation)



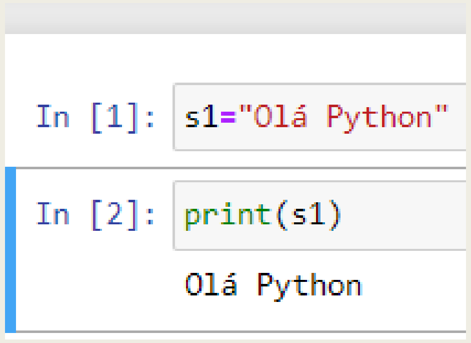
- To create a new notebook ("document"), select 'Python 3' on the same 'New' button on the right side.



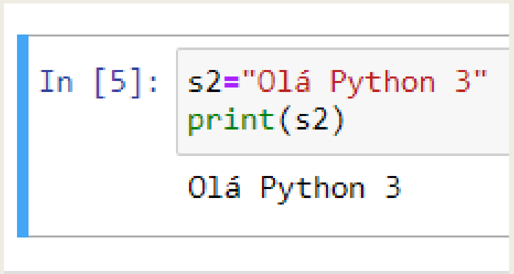
- Click 'Untitled' to give a suitable name to the notebook that has just been created
    - *the same is saved with the .ipynb extension, in the new 'AI' folder*

Getting started with the Jupyter Notebook

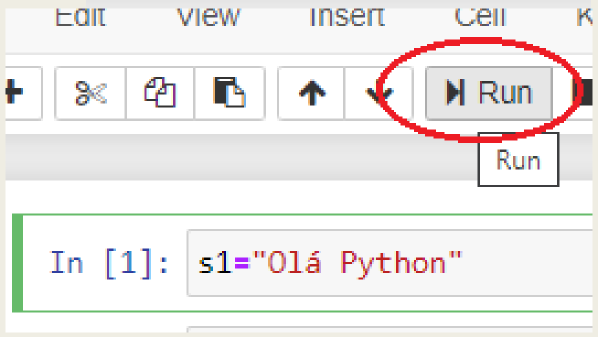
- A notebook contains one or more execution cells
  - *we can type a Python command in each of these cells, and run them separately*
  - *but we can also perform multiple commands at once, inserting in a single cell a section of several instructions*
  - *to run a cell, click the 'run' button, with that cell selected*
  - *to run all cells or cells before or after the selected one, go to the 'Cell' menu*



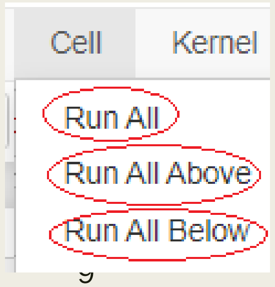
```
In [1]: s1="Olá Python"

In [2]: print(s1)
        Olá Python
```



```
In [5]: s2="Olá Python 3"
        print(s2)
        Olá Python 3
```

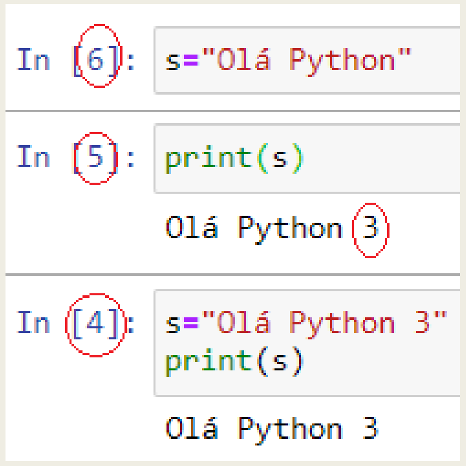


```
In [1]: s1="Olá Python"
```



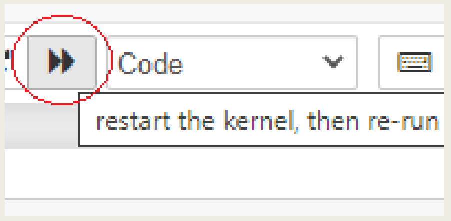
Getting started with the Jupyter Notebook

- Cells can be executed by any order
  - *to the left of each cell is the order number with which it was executed*

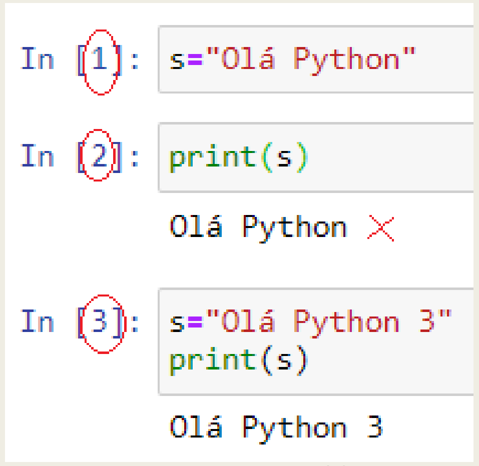


```
In [6]: s="Olá Python"

In [5]: print(s)
        Olá Python 3

In [4]: s="Olá Python 3"
        print(s)
        Olá Python 3
```

- When the order of execution is different from the order in which the cells are presented, unexpected results are often obtained or some difficulty in interpreting the results obtained
  - *in these situations it's possible to restart the Kernel, so that all variables are restarted, and we may re-run all cells in the order presented*



- For help with a function, position the cursor in the name of that function and press Shift+Tab



```
In [1]: s="Olá Python"

In [2]: print(s)
        Olá Python

In [3]: s="Olá Python 3"
        print(s)
        Olá Python 3
```

- After properly testing the solution created, the notebook code can be exported to a Python file
  - *for this, select File>Download as>Python (.py)*

The Visual Studio Code editor (VS Code)

- *Although only Jupyter Notebook can be used to fully build ML models in Python (option adopted in [1]),*
  - *we will also use the VS Code IDE,*
  - *It's more powerful and versatile editor that will give us another flexibility and additional functionality to create Python projects.*
- To use VS Code with Python you need to start by installing VS Code with the python extension

  Follow, for example, the instructions provided in

  [Original in pt]https://medium.com/@joaolggross/como-configurar-o-vs-code-com-anaconda-e-jupyter-notebooks-b05258bf65c1

  [Translated to en]https://github.com/worm69/AI/blob/main/Jupyter-anaconda-vscode/Jupyter-anaconda-vscode%20.md

  to properly install and configure VS Code, skipping step 1 if you already have Anaconda installed.

  (Changeback link, with installation instructions and a short usage tutorial: https://code.visualstudio.com/docs/python/python-tutorial)
- You can also use the Jupyter Notebook in the VS Code itself:
  - *https://code.visualstudio.com/docs/python/jupyter-support*
  - *If necessary, install in VS Code the extension for Jupyter Notebook*

Python Basics

- Comments with cardinal feature#
  - `print('hello world') # Note that print is a function`
- Constants
  - `5, 1.23, 9.25e-3, ‘This is a string’, “this is another one!"`
- Numbers
  - *integers (of any size), floating-point, and complex*
- Variables do not need to be declared ☺
  - *They are created by simply assigning them a value*
  - *No declaration or definition of data type is required*
- Python is case-sensitive
- Identifiers essentially follow the same rules as C (they use alphanumeric characters and underscore)
  - *but in Python 3 it is already possible to use accented characters and non-Anglo-Saxon alphabets*

Other particularities of Python

- Python is strongly object-oriented in the sense that everything is an object, including numbers, strings, and functions.
- Physical Line vs Logic Line
  - *A physical line is one you see when you write*
  - *A logical line is what Python sees as a single statement*
  - *Implicitly, Python encourages the use of a single statement per line, so that a physical line matches a logical line*
    - But if we want a physical line to include multiple statements (multiple logical lines), we only have to separate them by semicolon(;)
      - *example:* i=5; print(i);
    - If we want to distribute a statement across multiple physical lines, we should use the backslash (\) so that the physical line change is ignored
      - *example:*
        ```
        s = 'This is a \
        single string without line change'
        ```

Strings

- are immutable
- use UTF-8 Unicode encoding
- constants can be delimited by double quotes (`"…"`) or by single quotes (`'…'`)
- or by (`"""…"""`) e (`'''…'''`), to delimit strings with multiple lines or that include double quotes marks and single quotes inside

  ```
  ''' a string with
  "two lines"! '''
  ```

- to concatenate them just separate them by a space: day= 'It's' ' ' 'Friday' '-’ day'
- the usual exhaust sequences are used: `\  \n  \t  \\  \'  \” ...`
- raw strings, preceded by r or R: `r"i'm a string without any formatting"`
  - *take special meaning to the feature* `\`
  - *especially indicated for the handling of regular expressions*
- there’s NO char type

Strings - the format() method

- To build more elaborate strings, from a 'formatting string'':

```
age = 20
name = 'Swaroop'
print('{0} was {1} years old when he wrote this book'.format(name, age))
print('Why is {0} playing with that python?'.format(name))
```

```
Swaroop was 20 years old when he wrote this book
Why is Swaroop playing with that python?
```

- How it works
  - *The formatting string uses certain format specifiers (similar to the C printf), and later the format() method replaces these specifiers with the values of their parameters in the format indicated*
    - in the {0} the 1st parameter is placed, in the {1} position the 2nd, in the {2} the 3rd, and so on.
- More detailed specifications can be used:

```
print('{0:.3f}'.format(1.0/3)) # decimal precision of 3 for float '0.33333..'
# fill with underscores (_) with the text 'hello' centered (^) to 11 width
print('{0:_^11}'.format('hello'))
# keyword-based 'Swaroop wrote A Byte of Python'
print('{name} wrote {book}'.format(name='Swaroop', book='A Byte of Python'))
```

```
0.333
___hello___
Swaroop wrote A Byte of Python
```

Indentation

- Spaces, at the beginning of the line, are important in Python
- It is used to create instruction blocks (instead of {...})
    - *all instructions in a block must have the same indentation.*
- With wrong indentation we get errors, either logical or syntactic
    - *Example:*

```
i = 5
# Error below! Notice a single space at the start of the line
 print('Value is', i)
```

Result:

```
    print('Value is', i)
    ^
IndentationError: unexpected indent
```

- Therefore, in Python the use of indentation is not merely cosmetic.

Operators

- Most Python operators match C/C++ operators. Some of the different are the following:
    - *not* - *Logical operator NOT*
    - *and* - *Logical operator AND*
    - *or* - *Logical operator OR*
    - **** - *exponential (2**3=2^3=8)*
    - */* - *float division(5/3=1.6666..., 9/1.81=4.9723...)*
    - *//* - *entire division(5//3=1, 9//1.81=4.0)*

solve exercise #1

from the book of exercises