



Python I

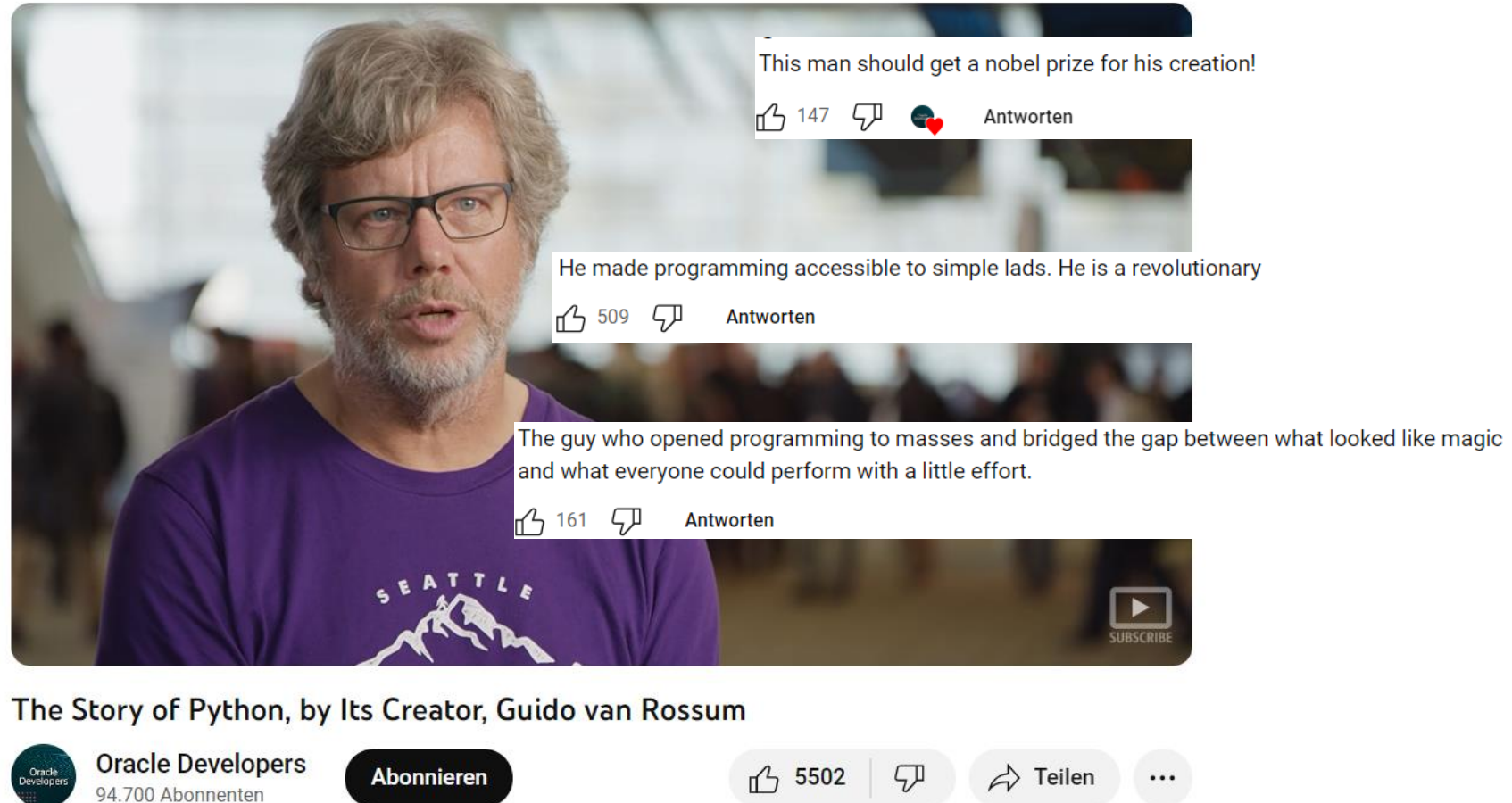
Programmierkurs 2 Data Science

WS23/24

Leonard Traeger
M. Sc. Information Systems
leonard.traeger@fh-dortmund.de



Who is Guido van Rossum?



Voraussetzungen

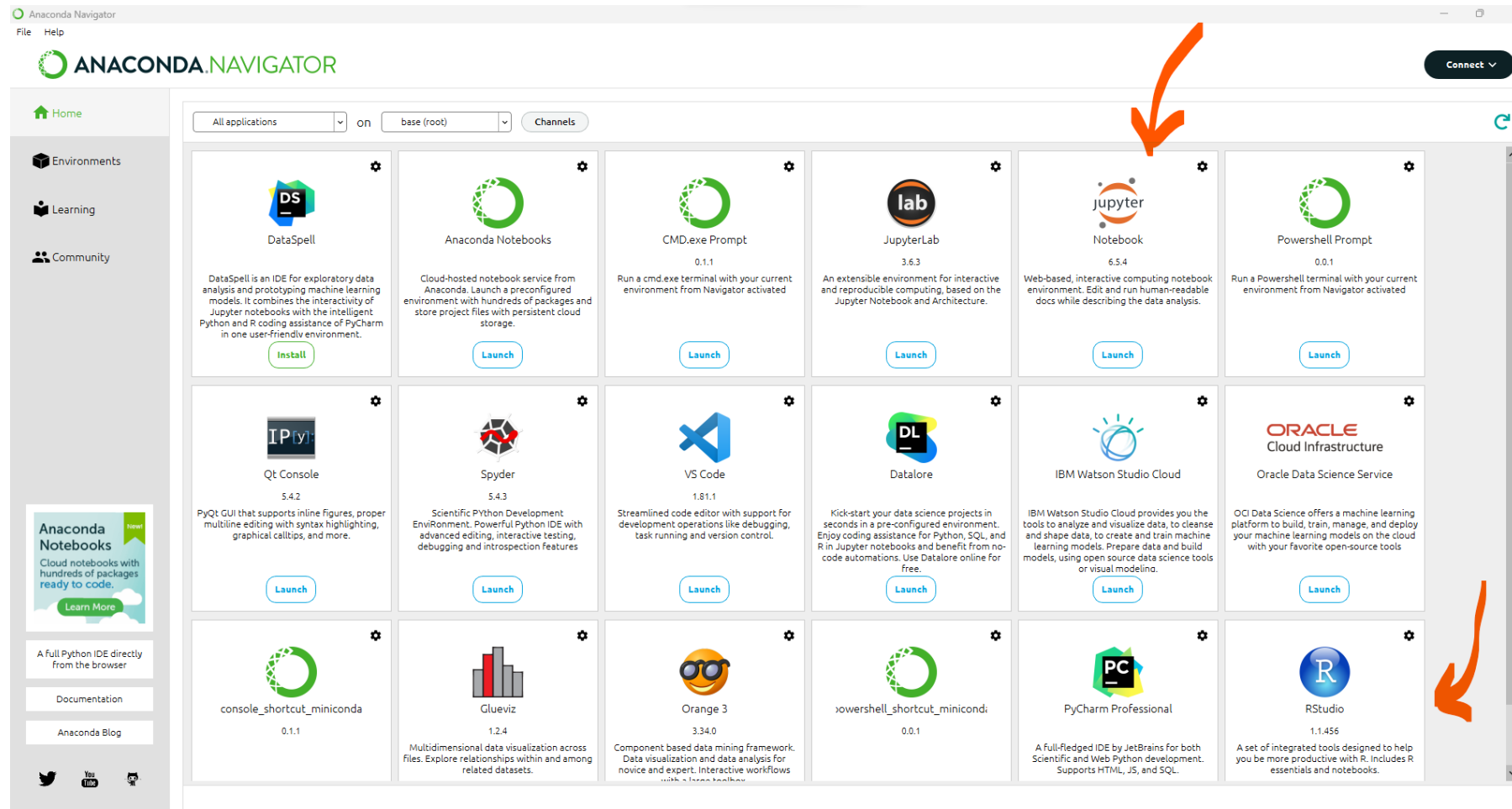
Hardware

- 4GB RAM, 2 CPU-Kerne und 20GB Speicher oder mehr.
- Windows 10, macOS 10.14, oder Ubuntu 14+/Centos7 (in 64-bit) oder neuer.
- Ein Computer mit ausreichender Internetgeschwindigkeit. Sollte das Vorlesungsgeschehen zu einem synchronen Onlineformat wechseln, stellen Sie sicher, dass Ihr Computer über einen Video- und Mikrofonanschluss verfügt.

Software

- Webbrowser, der technische Notebooks ausführen kann.
- Jupyter Notebook in Google Collab als Web IDE <https://colab.research.google.com/> (Vorteil: in Teams arbeiten, kommentieren, dokumentieren und programmieren).
- Jupyter Notebook als Stand-Alone Installation <https://www.anaconda.com/> (Vorteil: Offline arbeiten, RStudio und weitere Data Science Plattformen inkludiert): Python 3.10 oder neuer und Anaconda.

Anaconda I <https://www.anaconda.com/>

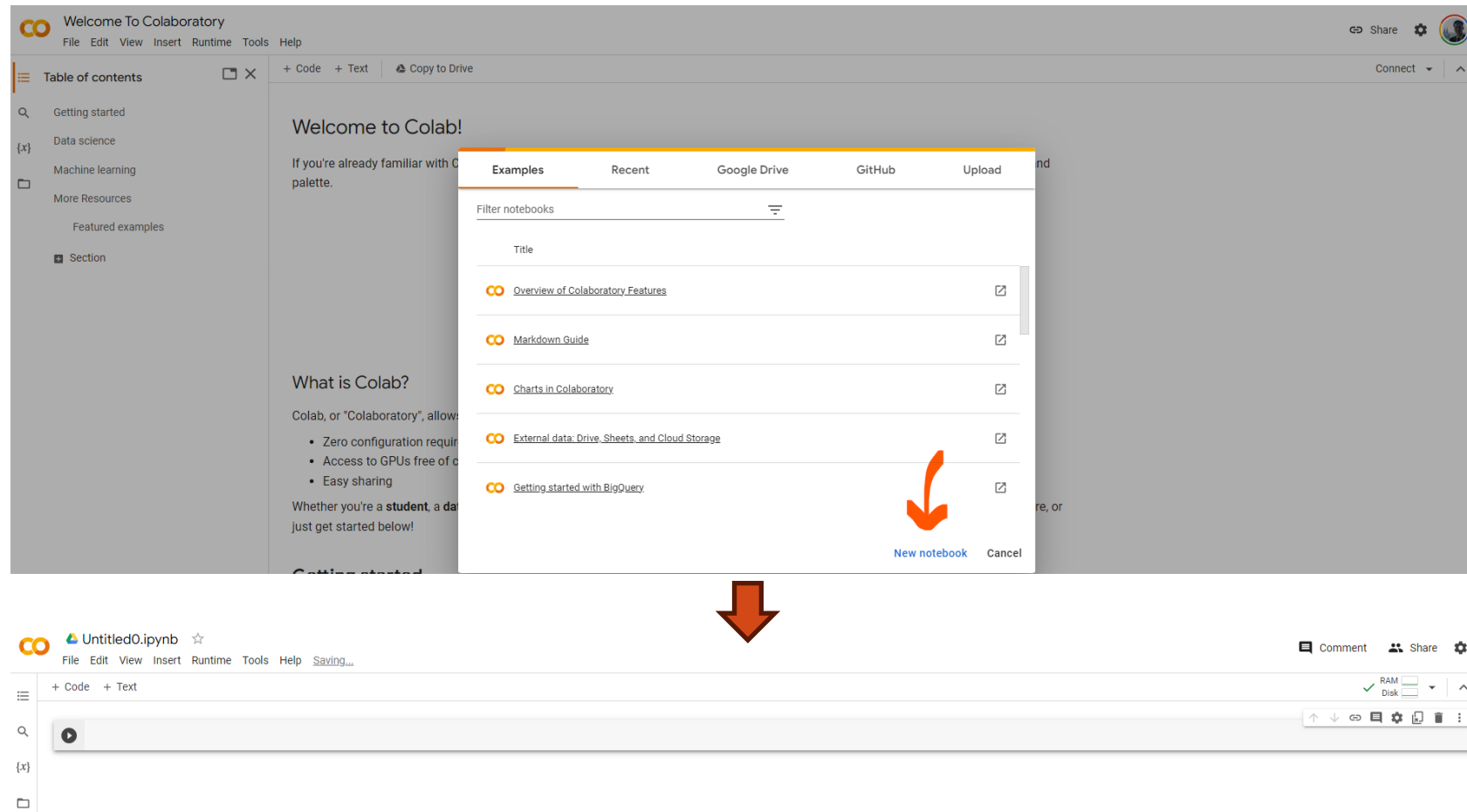


R wird in
Woche 10-12
behandelt

Anaconda II

The screenshot displays the JupyterLab web interface. At the top, the 'jupyter' logo is on the left, and 'Quit' and 'Logout' buttons are on the right. Below the logo are tabs for 'Files', 'Running', and 'Clusters'. A message states 'Select items to perform actions on them.' The file browser shows a directory structure: '/ Meine Ablage / WS23_24 PhD / FHDTM-P2DS-WS2324 / Data Science Projekt'. It contains a folder '..' and two files, 'Dateien' and 'Skripte'. A red arrow points from the 'New' button in the top right to a dropdown menu. This menu has a 'Notebook:' section with 'Python 3 (ipykernel)' selected, and an 'Other:' section with options 'Text File', 'Folder', and 'Terminal'. Another red arrow points from the 'Python 3 (ipykernel)' option to the main interface. Below the file browser, a large red arrow points down to the JupyterLab workspace. The workspace header shows 'jupyter Untitled' and 'Last Checkpoint: a few seconds ago (unsaved changes)'. On the right of the header are the Python logo and a 'Logout' button. Below the header is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. To the right of the menu bar are 'Trusted' and 'Python 3 (ipykernel)' buttons. Below the menu bar is a toolbar with icons for saving, creating new files, opening recent files, navigating, running cells, and other actions. The main area of the workspace shows a code editor with 'In []:' at the start of a new line.

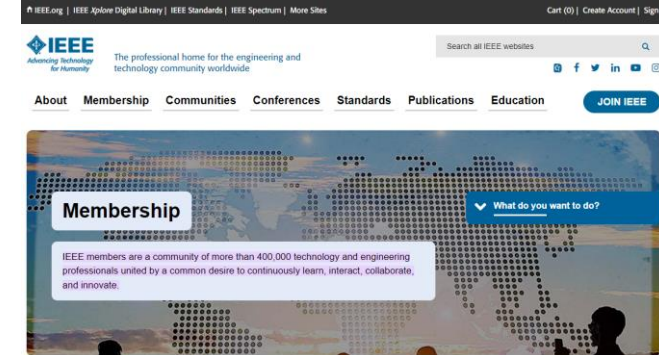
Colab Research <https://colab.research.google.com/> (only with Google Account)



Learning Goals Python I

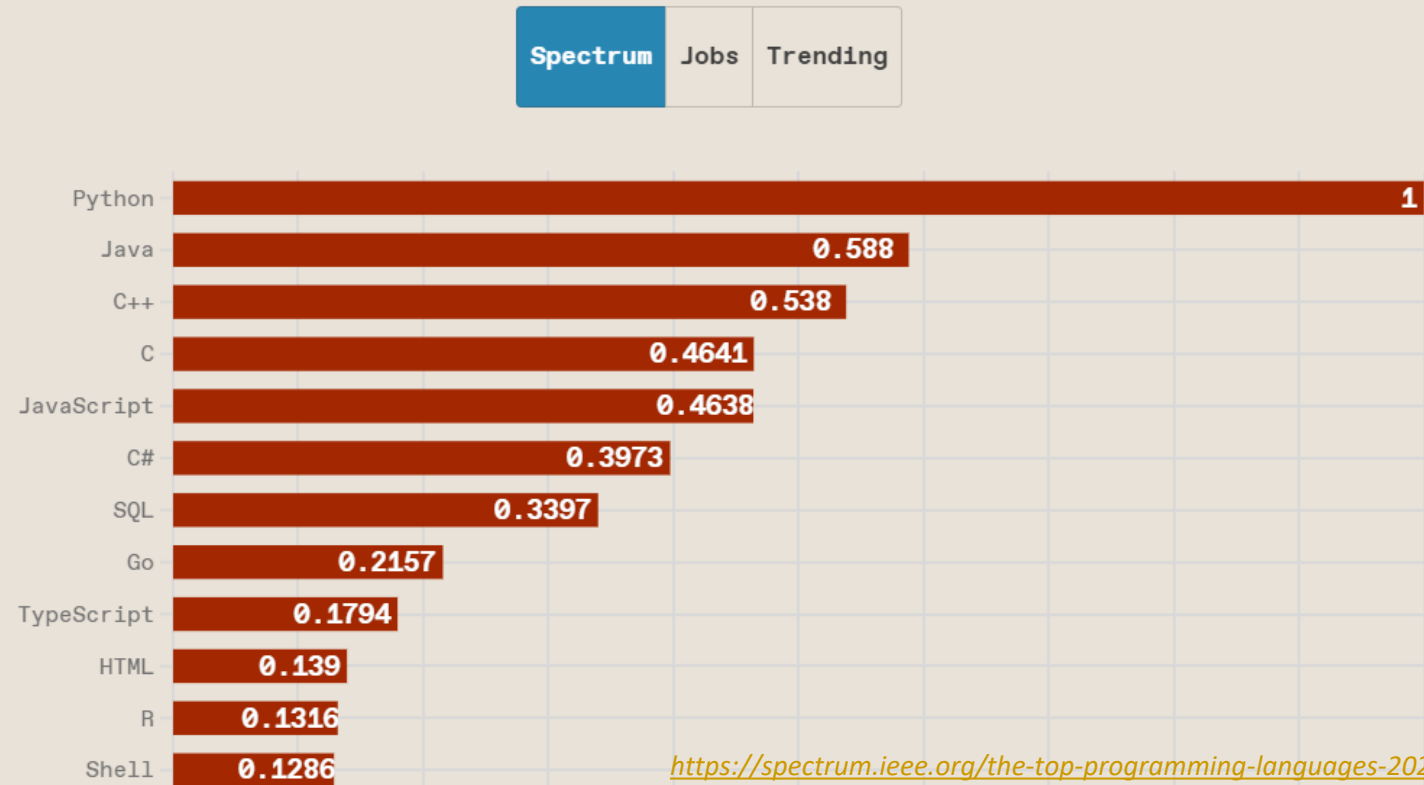
- **Describe** the reasons for the popularity of Python as a programming language and Jupyter as a development environment.
- **Explain** the advantages and disadvantages between dynamic and static typed programming languages.
- **List** potential limitations with Python and Jupyter.
- **Demonstrate** different documentation alternatives in Jupyter notebook and justify when to choose which alternative.
- **Create, change, and apply methods on** simple number, boolean, and string data typed variables.

Top Programming Languages



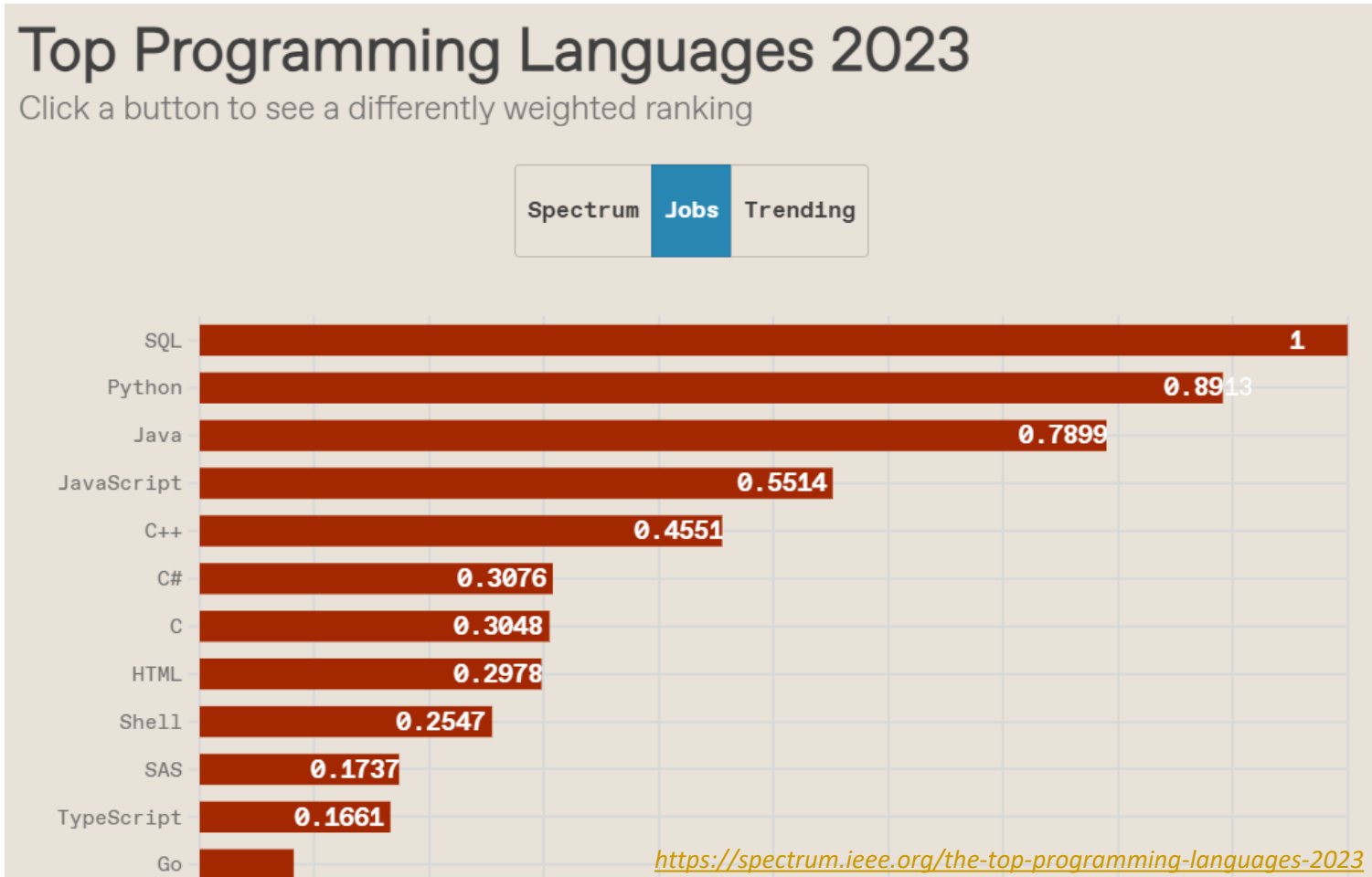
Top Programming Languages 2023

Click a button to see a differently weighted ranking



<https://spectrum.ieee.org/the-top-programming-languages-2023>

Top Programming Languages



Top Programming Languages

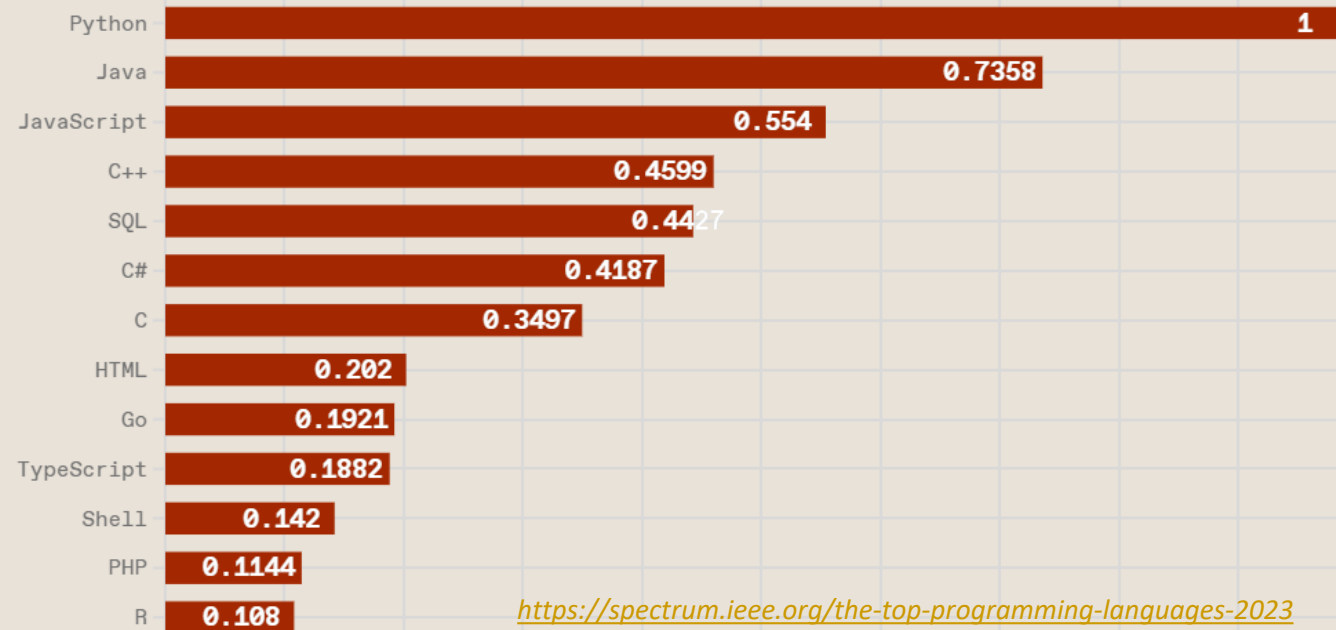
Top Programming Languages 2023

Click a button to see a differently weighted ranking

Spectrum

Jobs

Trending



<https://spectrum.ieee.org/the-top-programming-languages-2023>

Why Python?

- Open-source: widely used in science and more and more in industry.
- Supports structured, object-oriented and functional programming.
- Syntax with focus on readability “pythonic”.
- Dynamically typed and garbage-collected.
- Interface to other programming languages (C, Fortran,...).
- **Large, active, and growing ecosystem** of third-party packages.
 - NumPy: manipulation of numeric array-based data
 - Pandas: manipulation of tabular data
 - SciPy: scientific computing tasks
 - Matplotlib: data visualizations
 - Scikit-Learn: machine learning algorithms

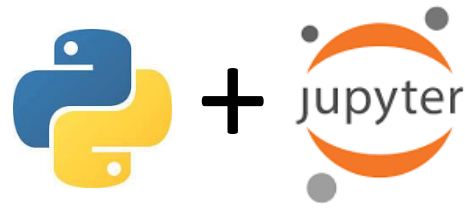
What is Jupyter Notebook?



- Web-based interactive computing platform.
- Document activities (enable reproducibility).
- Collaborative functionalities.
- Big data integration (Apache ecosystem, **Python**, **R**, Matlab,...).

https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html

Limitations



With Jupyter and Python, many obstacles such as **infrastructure management**, **documentation**, **collaboration**, and **automated version control** are eased.

For sufficiently large data sets and complex analytics, Jupyter and Python may not be ideal.

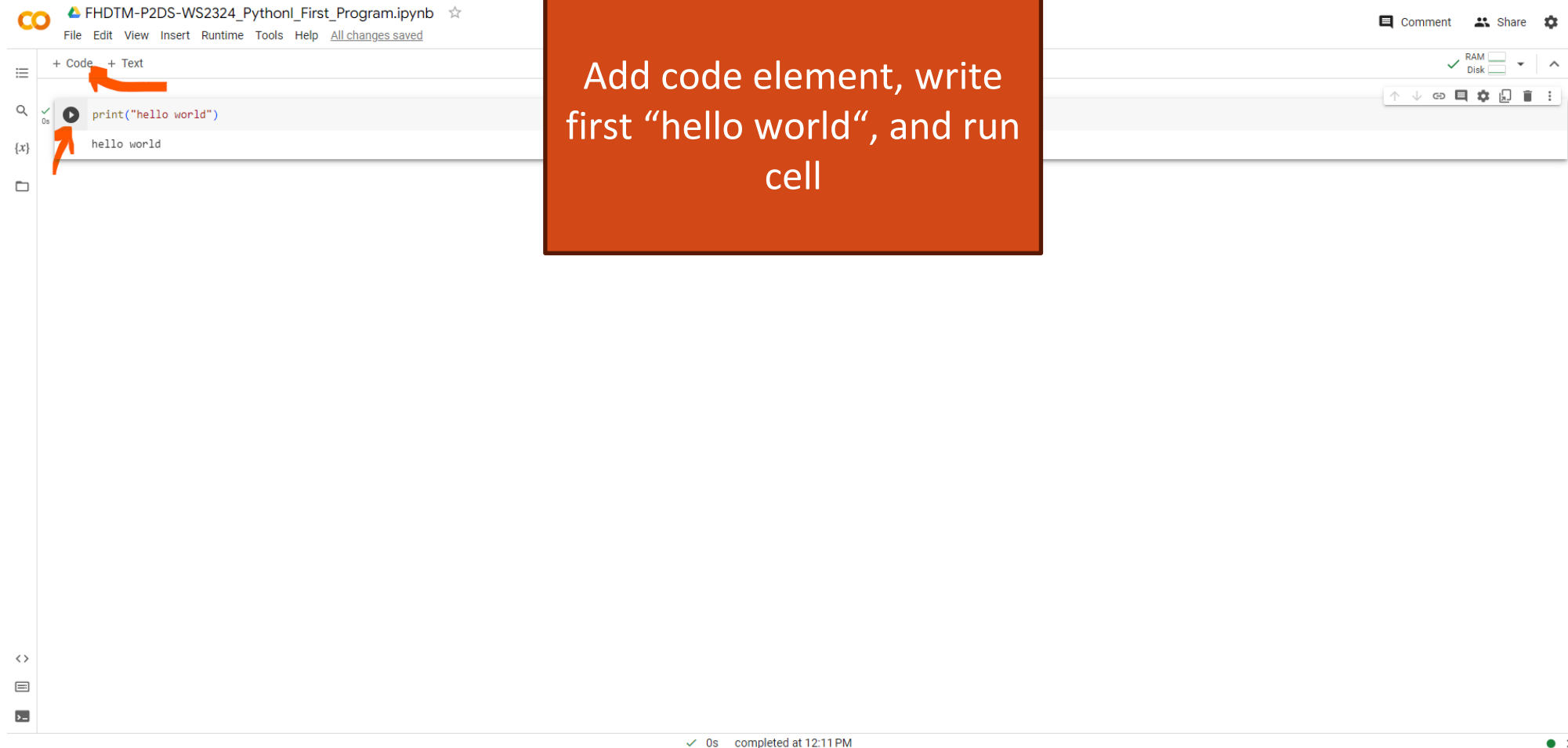
- Knowing *when to invest in switching tools* is a skill.
- Evaluate *trade-offs of flexibility, security, and speed for a given scale*.

In this class we use Jupyter and Python because

- Speed and
- Security

are typically not your priority during exploration.

Colab Jupyter Notebook Demo I



Python Interpreter

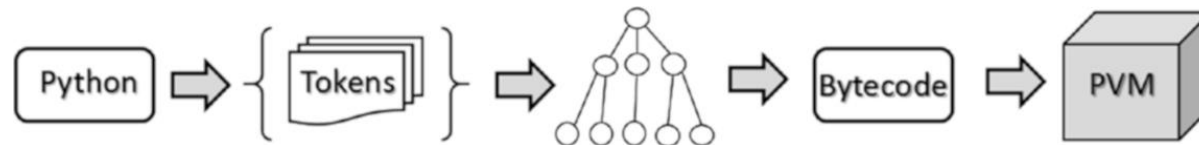
- Check version by importing the system-specific parameters and functions.

```
import sys
print(sys.version)
```

3.10.12 (main, Jun 11 2023, 05:26:28) [GCC 11.4.0]

This class requires
Python version 3.6 or
higher

- The Python interpreter reads and interprets the commands passed to the prompt.



- The interpreter accepts single commands at a time or entire files of Python code.
 1. Code scan and tokenization
 2. Tree arrangement (logical structure of the program)
 3. Bytecode conversion (.pyc or .pyo file)
 4. Python Virtual Machine execution (PVM)

Colab Jupyter Notebook Demo II

The screenshot shows the Google Colab Jupyter Notebook interface. The top menu bar includes File, Edit, View, Insert, Runtime, Tools, and Help. The main editor area contains a code cell with the text `print("hello world")` and its output, `hello world`. Five callout boxes are overlaid on the interface, each with an orange border and arrows pointing to specific menu items:

- File menu:** Contains options like 'Locate in Drive', 'New notebook', 'Open notebook' (Ctrl+O), 'Upload notebook', 'Rename', 'Move', 'Move to trash', 'Save a copy in Drive', 'Save a copy as a GitHub Gist', 'Save a copy in GitHub', 'Save' (Ctrl+S), 'Save and pin revision' (Ctrl+M S), 'Revision history', 'Download', and 'Print' (Ctrl+P). An arrow points from the 'Print' option to the text *print to submit*.
- Insert menu:** Contains options like 'Undo insert cell' (Ctrl+M Z), 'Redo' (Ctrl+Shift+Y), 'Select all cells' (Ctrl+Shift+A), 'Cut cell or selection', 'Copy cell or selection', 'Paste', 'Delete selected cells' (Ctrl+M D), 'Find and replace' (Ctrl+H), 'Find next' (Ctrl+G), 'Find previous' (Ctrl+Shift+G), 'Notebook settings', and 'Clear all outputs'.
- Runtime menu:** Contains options like 'Run all' (Ctrl+F9), 'Run before' (Ctrl+F8), 'Run the focused cell' (Ctrl+Enter), 'Run selection' (Ctrl+Shift+Enter), 'Run after' (Ctrl+F10), 'Interrupt execution' (Ctrl+M I), 'Restart runtime' (Ctrl+M .), 'Restart and run all', 'Disconnect and delete runtime', 'Change runtime type', 'Manage sessions', 'View resources', and 'View runtime logs'.
- Tools menu:** Contains options like 'Table of contents', 'Notebook info', 'Executed code history', 'Comments sidebar', 'Collapse sections' (Ctrl+)], 'Expand sections' (Ctrl+[), 'Show/hide code', 'Show/hide output' (Ctrl+M O), 'Focus next tab' (Ctrl+Shift+]), 'Focus previous tab' (Ctrl+Shift+[), 'Move tab to next pane', and 'Move tab to previous pane'.
- Cell menu:** Contains options like 'Code cell' (Ctrl+M B), 'Text cell', 'Section header cell', 'Scratch code cell' (Ctrl+Alt+N), 'Code snippets' (Ctrl+Alt+P), and 'Add a form field'.

Colab Jupyter Notebook Demo III

The screenshot shows a Google Colab Jupyter Notebook interface. At the top, the file name is "FHDTM-P2DS-WS2324_PythonI_First_Program.ipynb". The menu bar includes File, Edit, View, Insert, Runtime, Tools, Help, and a link to "All changes saved". On the right, there are buttons for Comment, Share, and a settings icon, along with RAM and Disk usage indicators.

On the left, a "Table of contents" sidebar is visible, listing "First Section", "Sub Section", "Sub Sub Section", and "Section". An orange arrow points to the "Table of contents" header.

The main editor area has tabs for "+ Code" and "+ Text". An orange arrow points to the "+ Text" tab. Below the tabs, the notebook content is displayed. It starts with a section header "Sub Sub Section" (indicated by an orange arrow) followed by a code cell containing `print("hello world")` which has been executed, showing the output "hello world".

A red box with white text is overlaid on the notebook content, stating: "Structure .ipynb file with text elements and # section annotations".

At the bottom of the notebook interface, a status bar shows "0s completed at 12:11 PM".

Colab Jupyter Notebook Demo IV

The screenshot shows a Google Colab Jupyter Notebook titled "FHDTM-P2DS-WS2324_PythonI_First_Program.ipynb". The interface includes a top menu bar (File, Edit, View, Insert, Runtime, Tools, Help), a left sidebar with a "Variables" panel, and a main code editor area. The "Variables" panel displays a table of variables:

| Name | Type | Shape |
|------|------|---------|
| x | int | |
| y | str | 3 chars |
| z | str | 8 chars |

The code editor shows three cells: a code cell with `x = 9`, a code cell with `y, z = "BVB", "Dortmund"`, and a code cell with `print(y + " " + z + " " + str(x))` which outputs "BVB Dortmund 09". A large orange box with white text is overlaid on the notebook, stating: "Inspect run time (dynamically typed) variables; comment code elements with peers". An orange arrow points from the "Comment" button in the top right to a comment thread on the right side of the notebook. The comment thread shows a conversation about a football club. The bottom status bar indicates the notebook is "completed at 12:39 PM".

Inspect run time (dynamically typed) variables; comment code elements with peers

Comment

Share

RAM
Disk

Leonard Traeger
12:43 PM Today
Is that a football club?

Leonard Traeger
12:43 PM Today
@OtherMe: Yes, it is a german one!

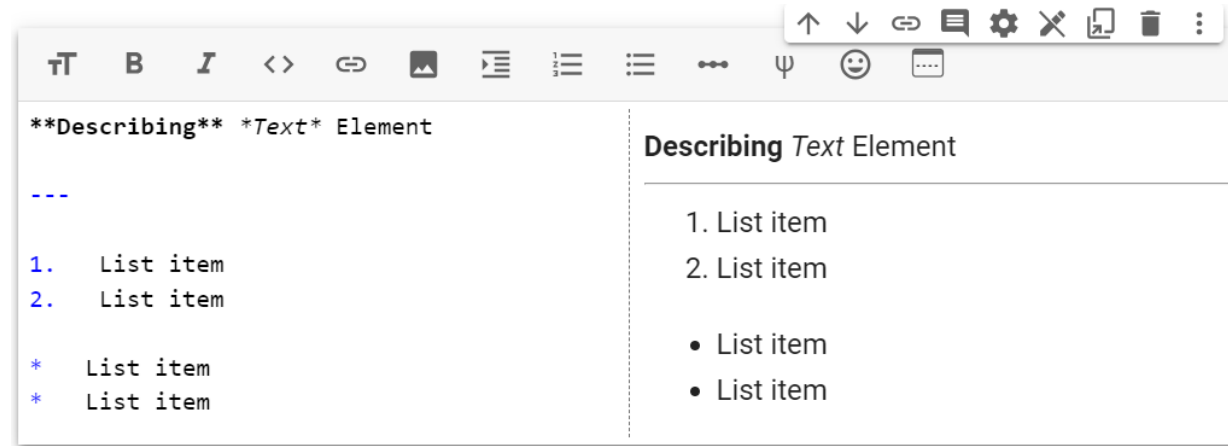
Reply or add others with @

Reply Cancel

completed at 12:39 PM

Documentation

- Text element
 - Few formatting styles
 - Emojis
 - Lists
 - Links
 - Images



- Code: inline comment with preceding #.

```
#inline comment
```

- Code: multiple lines comment block with preceding ''' and closing ''.

```
'''multiple  
line  
comment'''
```

Variables and Datatypes

Combination of an identifying label and a value.

```
x = 9
```

```
y,z = "BVB", "Dortmund"
```

→ *Variable **value** assignment.*

```
print(y + " " + z + " 0" + str(x))
```

BVB Dortmund 09

→ *Use **variable label** to get value.*

→ *Convert it for subsequent processing step e.g., concatenation: `str(9) = "9"`.*

Variable and Datatypes (cont.)

Assignment uses **dynamic referencing**.

- The **type/class** is determined from the **value**, not declared.
- **Type/class information** belongs to the **data**, not the name bound to that data.

```
x = 10000
```

- x is not just a “raw” integer.
- x is a pointer to a compound C structure, which contains several values.
- **Dynamic referencing** in Python is **more flexible** but also **more time** and **space** consuming than compared to raw C.

Variables and Datatypes (cont.)

- Anything can be a variable in Python: number, string, function, module, object, ...
- Names can be **any continuous** string but must **start** with a **letter** or **underscores**.
- The **more meaningful** your **names**, the **easier** it will be to **understand** the **code**.

```
snake_case_style_answer_to_everything = 42
```

```
camelCaseStyleAnswerToEverything = "42"
```

```
print( type(snake_case_style_answer_to_everything), type(camelCaseStyleAnswerToEverything))
```

```
<class 'int'> <class 'str'>
```

Indentation

- Python uses indentation to structure code unlike many other languages, e.g., R, C++, which use braces { }.
- In Python, **indentation** is **functional**, not just good style.
- Functional whitespace makes Python **more readable** to **humans** but also potentially harder to debug (in the beginning).
- A **colon :** is used to denote the **start** of an **indented block**.

```
k = 0
for i in range(4):
    k += i
    print(k)
```

```
k = 0
for i in range(4):
    k += i
print(k)
```

Modules / Libraries

- Open-source code-reuse: one of Python's essential reason for its power and popularity.
- Modules must be imported before they can be used.
- **Avoid** at all costs `from module import *`

```
#import libraries preferably in a separate cell  
#of running code to avoid conflicts and save time  
from random import randint  
from random import randint as ri
```

- Give imported object **an alias** to avoid writing the whole name every time.

```
print(randint(0,101))  
print(ri(0,101))  
#Prints (any random number between zero and including 100)"
```

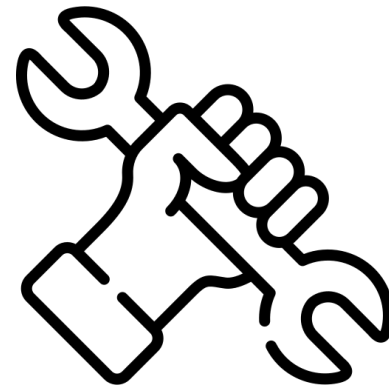
Numbers

- Integers and floats work as you would expect from other languages.

```
x = 3
print(type(x)) # Prints "<class 'int'>"
print(x)       # Prints "3"
print(x + 1)   # Addition; prints "4"
print(x - 1)   # Subtraction; prints "2"
print(x * 2)   # Multiplication; prints "6"
print(x ** 2)  # Exponentiation; prints "9"
x += 1
print(x)       # Prints "4"
x *= 2
print(x)       # Prints "8"
y = 2.5
print(type(y)) # Prints "<class 'float'>"
print(y, y + 1, y * 2, y ** 2) # Prints "2.5 3.5 5.0 6.25"
```

- Note: Python does not have unary increment (x++) or decrement (x--) operators!!!

Training #1



- 1) Count the number of people in the lecture room and the number of local computers.
- 2) Compute the average number of computers per person and assign this value to a variable.
- 3) Print the value and the data type of the computed variable.

Booleans

- Python implements all Boolean logic, but uses English instead symbols (&&, ||, etc.):

```
t = True
f = False
print(type(t)) # Prints "<class 'bool'>"
print(t and f) # Logical AND; prints "False"
print(t or f)  # Logical OR; prints "True"
print(not t)   # Logical NOT; prints "False"
print(t != f)  # Logical XOR; prints "True"
```

Only exception:
Pandas Masking!
Week 7+

String

```
hello = 'hello'    # String literals can use single quotes
world = "world"    # or double quotes; it does not matter.
print(hello)       # Prints "hello"
print(len(hello))  # String length; prints "5"
hw = hello + ' ' + world  # String concatenation
print(hw)  # prints "hello world"
hw12 = '%s %s %d' % (hello, world, 12)  # sprintf style string formatting
print(hw12)  # prints "hello world 12"
```

String (cont.)

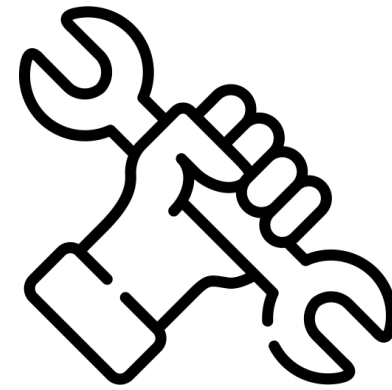
- Some very useful string methods:

```
string = "BVB Dortmund 09 "  
print(string[14])           # prints 14th element of the string "9"  
print(string.capitalize())  # Capitalize a string; prints "Bvb dortmund 09 "  
print(string.strip())       # Removes any whitespace from the beginning or the end;  
                           # prints "BVB Dortmund 09"  
print(string.replace('B', 'Borussia')) # Replace all instances of one substring with another;  
                           # prints "BorussiaVBorussia Dortmund 09 "  
print(string.lower())       # Convert a string to lower case; prints "bvb dortmund 09 "  
print(string.upper())       # Convert a string to uppercase; prints "BVB DORTMUND 09 "  
  
string_split = string.split(" ") # Splits string into substrings based on separator  
print(len(string_split))       # prints length of array list "4"  
print(string_split[0])        # prints first value of array list "BVB"
```

More to arrays and
lists next class!

<https://docs.python.org/3.5/library/stdtypes.html#string-methods>

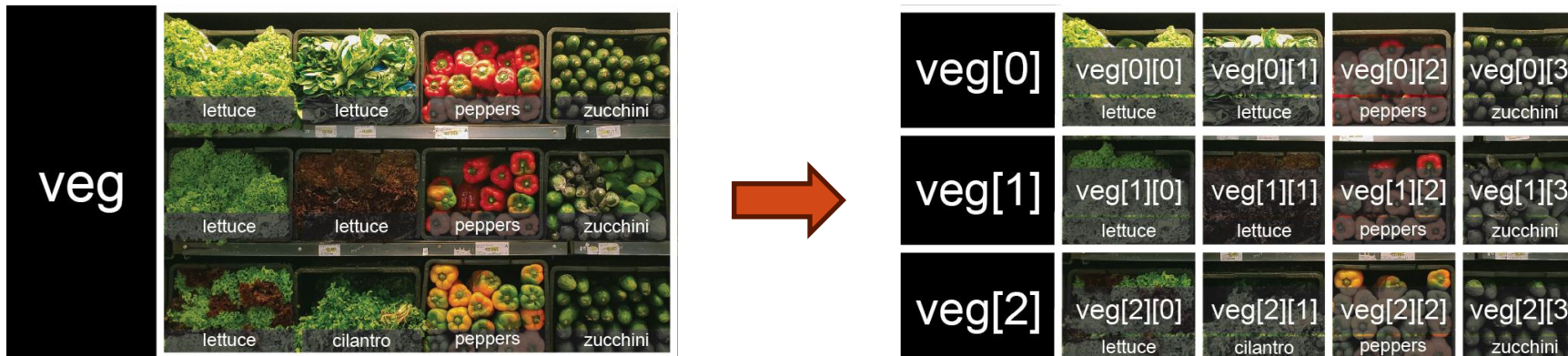
Training #2



1. Assign "ojyVTTyamyknYMakLmktDYNtyklnyhmyrn" to a variable s1.
2. Assign "kNZwykwoCHynsynDYtkhrMKrjYpr" to another variable s2.
3. Concat both strings in s2+s1 order.
4. Cast the lower-case method on the concatenated string.
5. Replace "k" with "i".
6. Replace "v" with "k".
7. Replace "y" with "e".
8. Print the output.

Outlook

- In week 3 we will start with Data Science Life Cycles, Data Literacy, and Ethics as you will need these for structuring your DS project.
- Afterwards, we will dive deep into containers and functions in Python.



- Refine expectations for project milestone #1.

See you again in **two weeks in person!**

Questions?