

efl Python Course

Introduction

Version 1.0.0
2024-12-06
By Dr. Benjamin M. Henrich

About us: Who We Are?



Mission (Since 2019):

- Use expertise in Data Science to deliver cutting edge research in the fields of
 - Financial Services
 - Retail & Marketing
 - Health
 - Law
 - General, cross-sectional research

Scientific Advisory Board



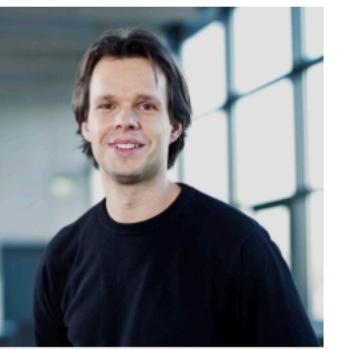
Prof. Dr. Oliver Hinz

Chairman of the Board
Oliver Hinz is Professor of Information Systems and Information Management. His main topics at efl - the Data Science...



Prof. Dr. Peter Gomber

Vice Chairman of the Board
Prof. Dr. Peter Gomber holds the Chair of e-Finance at the University of Frankfurt. He is Co-Chairman and Member of the...



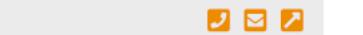
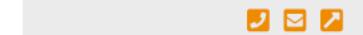
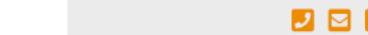
Prof. Dr. Carsten Binnig

Member of the Board
Prof. Dr. Carsten Binnig is part of the Department of Computer Science at the TU Darmstadt in Darmstadt, Germany. He...



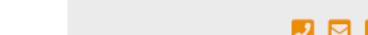
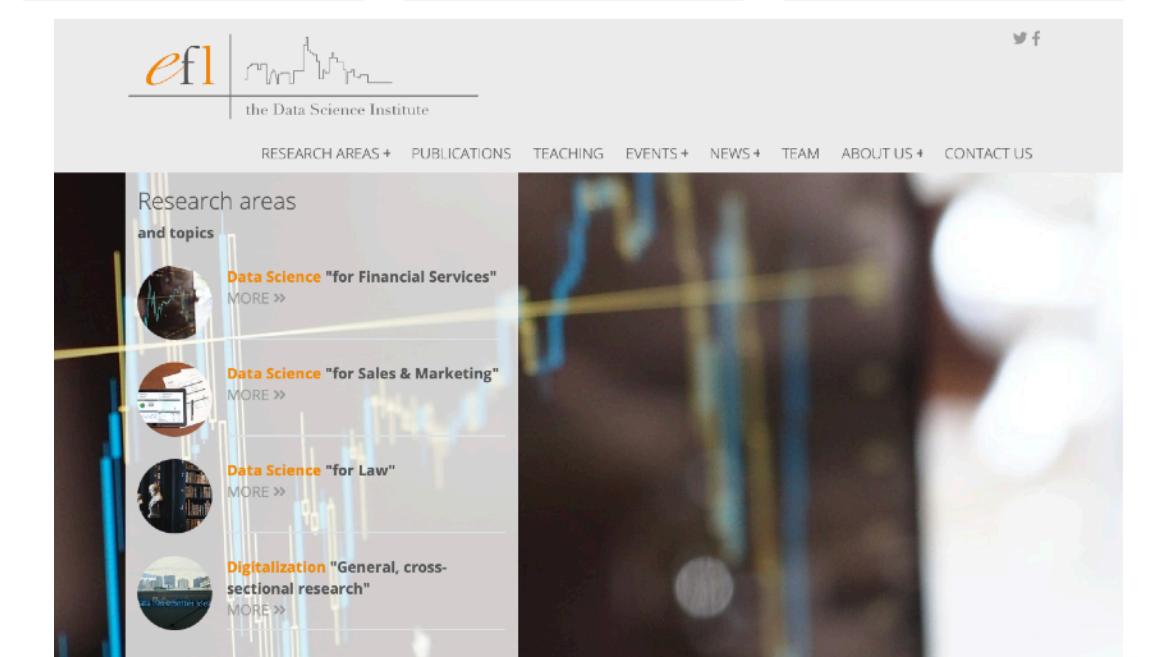
Prof. Dr. Andreas Hackethal

Member of the Board
Prof. Dr. Hackethal's empirical research is on individual investment behavior, the role of financial advice and on...



Prof. Dr. Bernd Skiera

Member of the Board
Prof. Dr. Bernd Skiera has held the Chair of Electronic Commerce at the Faculty of Economics and Business at...

The screenshot shows the efl website homepage. At the top, there is a navigation bar with links for RESEARCH AREAS, PUBLICATIONS, TEACHING, EVENTS, NEWS, TEAM, ABOUT US, and CONTACT US. Below the navigation, there is a section titled "Research areas and topics" featuring four circular icons: "Data Science "for Financial Services"" (with a graph icon), "Data Science "for Sales & Marketing"" (with a chart icon), "Data Science "for Law"" (with a person icon), and "Digitalization "General, cross-sectional research"" (with a globe icon). To the right of these icons is a large, blurred background image of a person working on a computer.

Our **mission** is to contribute to the evolution of traditional industries towards "**digitized and high-performance industries**" by solving practical and theoretical problems with the tools of **data science** and methodology of scientific research.

A COOPERATION OF
 GOETHE
UNIVERSITÄT
FRANKFURT AM MAIN
 TECHNISCHE
UNIVERSITÄT
DARMSTADT

AND A NETWORK OF INDUSTRY PARTNERS.
efl - THE DATA SCIENCE INSTITUTE FRANKFURT AM MAIN WAS FOUNDED IN OCTOBER 2002 AND STARTED OPERATIONS ON JANUARY 30, 2003.

Upcoming Events



The screenshot shows the efl website's events page. It features a grid of event cards. One card is visible, showing the date "dec 02" and the title "Do investors Use Sustainable Assets as Carbon Offsets?". There are also links for "events" and "archive".

Latest News



The screenshot shows the efl website's news page. It features a news item with the headline "efl Annual Conference 2024 successfully took place" and the date "Friday, 15. November 2024". Below the headline, it says "efl Annual Conference 2024 on "Privatkundengeschäft und KI" successfully took place". There is a "Details" button and a thumbnail image of people at the conference.

About us: The efl



Industry - Academic Partnership

Universities



Sponsors



About US

Lecturers



M.Sc. Anjana Cordes

Research Assistant

Anjana Cordes joined the efl - the Data Science Institute as a research assistant in October 2022.



M.Sc. Tino Cestonaro

Research Assistant

Tino Cestonaro joined the team in April 2020. His research focuses on Market Microstructure, Financial Machine Learning,...



M.Sc. Florian Ewald

Research Assistant

Florian Ewald joined the efl research team in May 2023. His research focuses on the application of machine learning and...



Dr. Benjamin M. Henrich

Alumnus

Since March 2018, Benjamin M. Henrich is a research assistant of Prof. Dr. Oliver Hinz at the Chair of Information...



Agenda

Teaching

Day 1 (Python Course) (06.12.2024)

9:00 - 10:30 Uhr

Python Basics
Introduction and Primitive Data Types

10:40 - 12:10 Uhr

Data Structures
Tuples, Lists, Sets, Dictionaries

13:30 - 15:00 Uhr

Control Structures
Loops, Conditional Statements

15:15 - 16:45 Uhr

Structure of Functions and Application
Functions in Python

Day 2 (Python Course) (13.12.2024)

9:00 - 10:30 Uhr

Libraries in Python
Os, numpy, re, csv

10:40 - 12:10 Uhr

Data Wrangling Libraries in Python
Pandas, dataframes

13:30 - 15:00 Uhr

Data Visualization
Matplotlib, seaborn

15:15 - 16:45 Uhr

Data Analysis with Python
Merging data, explanatory analysis

Day 3 (Data Science) (20.12.2024)

9:00 - 10:30 Uhr

Introduction to Data Science
Terminology and basic concepts

10:40 - 12:10 Uhr

Preprocessing
Normalization, train-test split

13:30 - 15:00 Uhr

Simple Model Application
Decision trees, random forests

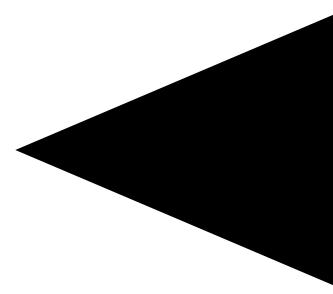
15:15 - 16:45 Uhr

Deep Learning
Neural Networks, keras



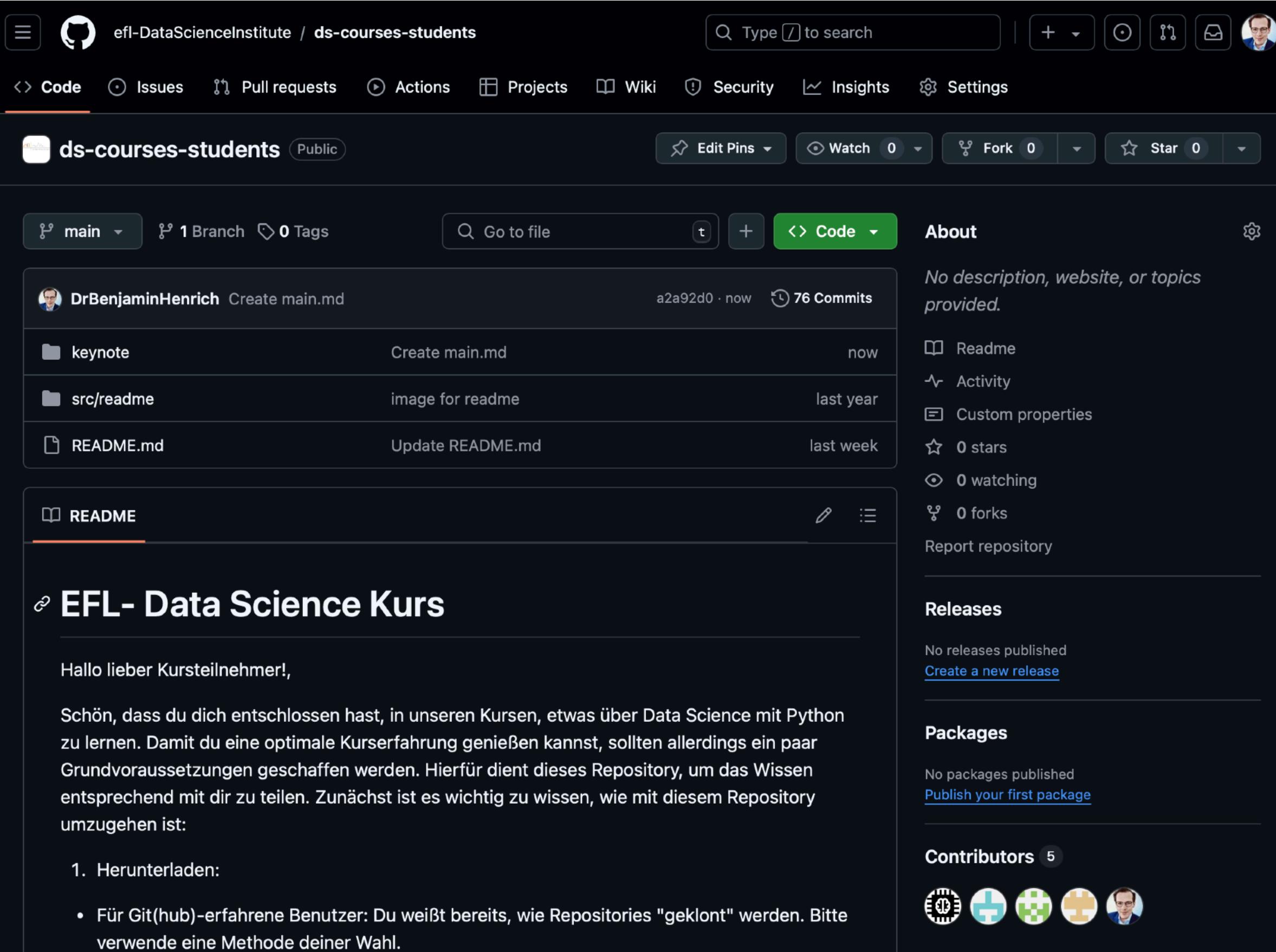
<https://www.eflab.de/teaching>

Certificates



- Completion of tasks from the last class
- Submission: Description of how the tasks were solved
- Formalities:
 - Min. 2 pages (Arial 11, 1.5 Line spacing, 3 CM Correction margin)
 - Code must be delivered separately (code folder)

Course Material?



The screenshot shows a GitHub repository page for 'ds-courses-students'. The repository is public and has 1 branch and 0 tags. The main file listed is 'main.md' created by DrBenjaminHenrich. The repository has 76 commits. The README file contains the following content:

```

EFL- Data Science Kurs

Hallo lieber Kursteilnehmer!,

Schön, dass du dich entschlossen hast, in unseren Kursen, etwas über Data Science mit Python zu lernen. Damit du eine optimale Kurserfahrung genießen kannst, sollten allerdings ein paar Grundvoraussetzungen geschaffen werden. Hierfür dient dieses Repository, um das Wissen entsprechend mit dir zu teilen. Zunächst ist es wichtig zu wissen, wie mit diesem Repository umzugehen ist:

1. Herunterladen:
  • Für Git(hub)-erfahrene Benutzer: Du weißt bereits, wie Repositories "geklont" werden. Bitte verwende eine Methode deiner Wahl.

```

About

No description, website, or topics provided.

- Readme
- Activity
- Custom properties
- 0 stars
- 0 watching
- 0 forks

Report repository

Releases

No releases published

[Create a new release](#)

Packages

No packages published

[Publish your first package](#)

Contributors 5

Icons for contributors: DrBenjaminHenrich, efl, Data Science Institute, GitHub, and a person icon.



<https://github.com/efl-DataScienceInstitute/ds-courses-students>

**Why Are You Here?
What is your Motivation?
Why Coding?**

Why Coding?



NACH ABSCHLUSS IN ...
**INFORMATIK, MATHEMATIK,
 WIRTSCHAFTSINFORMATIK**



TOP BERUFSTITEL*

IT Manager
92.275 €

IT-Projektmanager/in
86.695 €

Senior Developer
78.635 €

Analyst
74.469 €

Software-Architekt/in
74.152 €

TOP BRANCHE

Banken
85.067 €

Chemie- und Erdöl-
 verarbeitende Industrie
84.033 €

Konsum- und
 Gebrauchsgüter
82.586 €

Unternehmensberatung,
 Wirtschaftsprüfung und Recht
82.336 €

Finanzdienstleister
81.331 €

Why Coding? (Personal View)

Coding Rewires Your Brain!

Concept	Explanation
Abstraction	Hiding complex details and exposing only the essential functionality. Abstraction helps in simplifying system complexity, making the code more understandable and easier to manage.
Coupling	Refers to the degree of dependency between different modules or components of a system. Tight coupling means high interdependence, while loose coupling is preferred for better maintainability and flexibility.
Continuous Integration (CI)	A development practice where code changes are automatically built and tested to ensure they integrate well into the main codebase. It helps in identifying integration issues early and speeds up development cycles.
Divide and Conquer	A problem-solving strategy where a problem is broken down into smaller, manageable sub-problems. Each sub-problem is solved independently, and results are combined. Common in algorithms like Merge Sort and Quick Sort.
Modularity	The design principle that divides software into smaller, independent, and interchangeable modules or components. Each module is responsible for a specific functionality and can be developed, tested, and maintained independently.
Refactoring	The process of restructuring existing code without changing its external behavior. It improves code readability, reduces complexity, and makes it easier to maintain without introducing new bugs.
Never Repeat Yourself (DRY)	A principle in software development aiming to reduce repetition of software patterns. Repeating code leads to redundancy, maintenance issues, and bugs. The goal is to use reusable functions or modules.
Test-Driven Development (TDD)	A software development process where tests are written before the code itself. This ensures that the code meets its specifications and helps catch bugs early in the development cycle.

Fail Fast!

What is Coding?



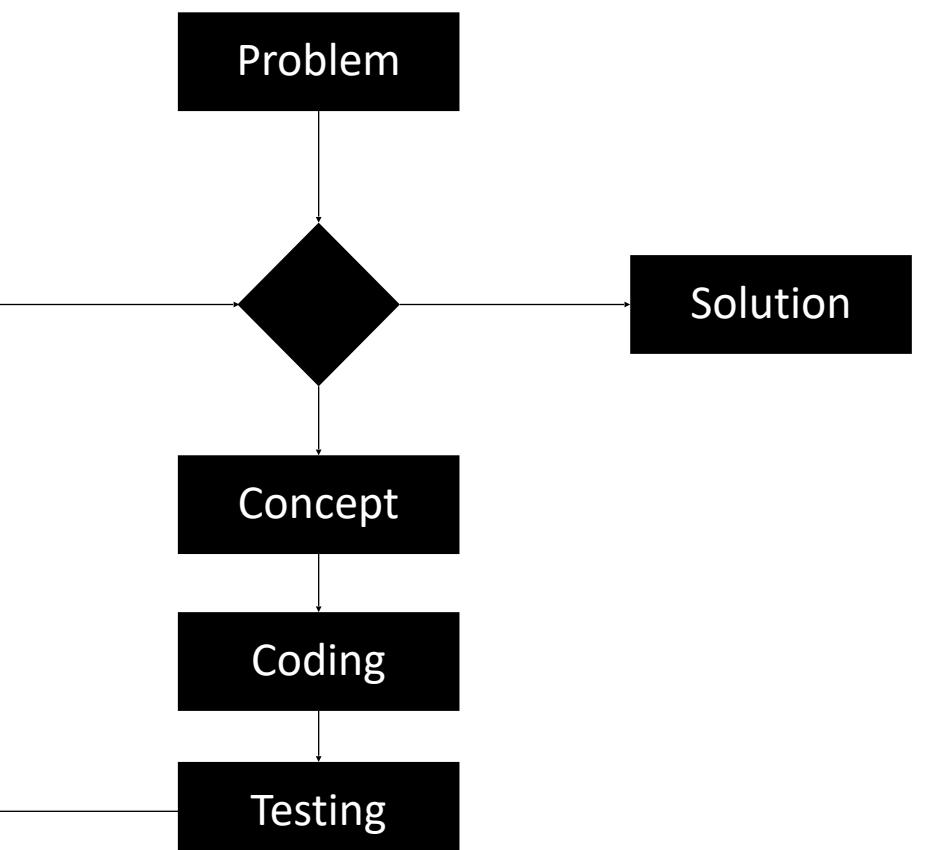
Computer programming

[Article](#) [Talk](#)

From Wikipedia, the free encyclopedia

Computer programming or **coding** is the composition of sequences of instructions, called **programs**, that computers can follow to perform tasks.^{[1][2]} It involves designing and implementing **algorithms**, step-by-step specifications of procedures, by writing **code** in one or more **programming languages**. Programmers typically use **high-level programming languages** that are more easily intelligible to humans than **machine code**, which is directly executed by the **central processing unit**. Proficient programming usually requires expertise in several different subjects, including knowledge of the **application domain**, details of programming languages and generic code **libraries**, specialized algorithms, and formal **logic**.

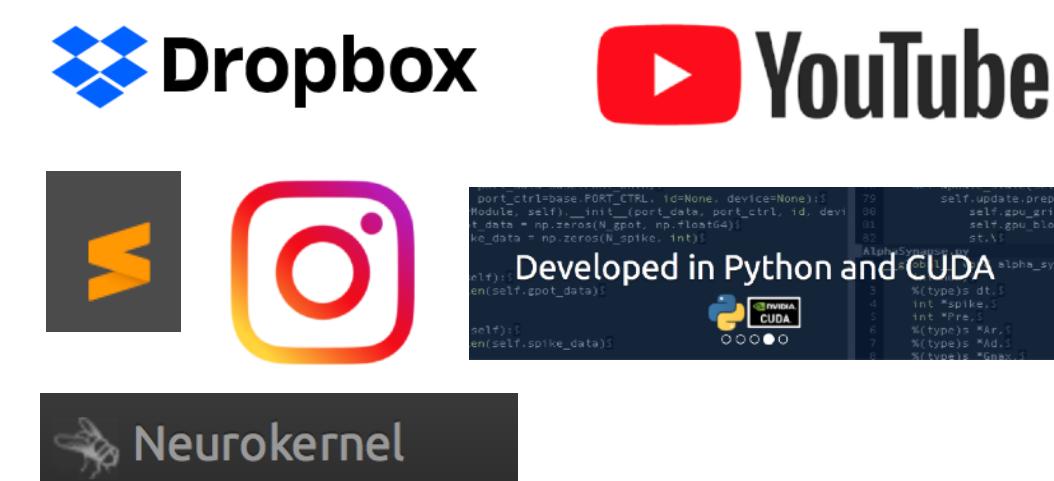
https://en.wikipedia.org/wiki/Computer_programming



Motivation: Why Python?- Part I



- Released in 1991 by Guido van Rossum
- With the explosive growth of ‘big data’ in disciplines such as bioinformatics, neuroscience and astronomy, programming know-how is becoming ever more crucial (Perkel 2015, p. 125).



Programming, scripting, and markup languages

JavaScript has been a mainstay in the developer survey and on Stack Overflow since our first survey. The most popular programming language has been JavaScript every year we have done the survey except for 2013 and 2014, when SQL was the most popular language.

Which programming, scripting, and markup languages have you done extensive development work in over the past year, and which do you want to work in over the next year? (If you both worked with the language and want to continue to do so, please check both boxes in that row.)



Motivation: Why Python - Part II

```
public class Test {  
    public static void main(String args[]) {  
        String array[] = {"Hello, World", "Hi there, Everyone", "6"};  
        for (String i : array) {  
            System.out.println(i);  
        }  
    }  
}
```

Java

```
stuff = ["Hello, World!", "Hi there, Everyone!", 6]  
for i in stuff:  
    print(i)
```

Python

Motivation: The Elementary Basics in Python

Pros Python:

- General purpose language
- Higher programming language
- It's simple
- Fast to read
- Structuring by indenting
- No {} or ; => Faster to Code
- Data types are managed dynamically. There is no static type check like in java
- Widespread in science
- Extensive Support Libraries (important data science, math and many more)
- Integration Feature
- Productivity (Many Frameworks such as unit testing)



Ozgur, C., Colliau, T., Rogers, G., & Hughes, Z. (2017). MatLab vs. Python vs. R. Journal of data Science, 15(3), 355-371.
Abdel-Karim, B. M. (2022). Data Science. In Data Science: Best Practices mit Python (pp. 57-62). Wiesbaden: Springer Fachmedien Wiesbaden.

Cons of Python

- Python is an interpreted language, this is slow as compared to C/C++
- See Interpreted vs. Compiled programming languages
- Python can have runtime errors (dynamical typing feature)
- Consumes a lot of memory space
- Not easy to test



Small Satellite*
*) <https://www.nbcnews.com/id/wbna43313086>

May not in Python ...

Prechelt, L. (2000). An empirical comparison of c, c++, java, perl, python, rex and tcl. *IEEE Computer*, 33(10), 23-29.

Newhall, T. K. (1999). Performance Measurement of Interpreted, Just-in-Time compiled, and Dynamically Compiled Executions. The University of Wisconsin-Madison.

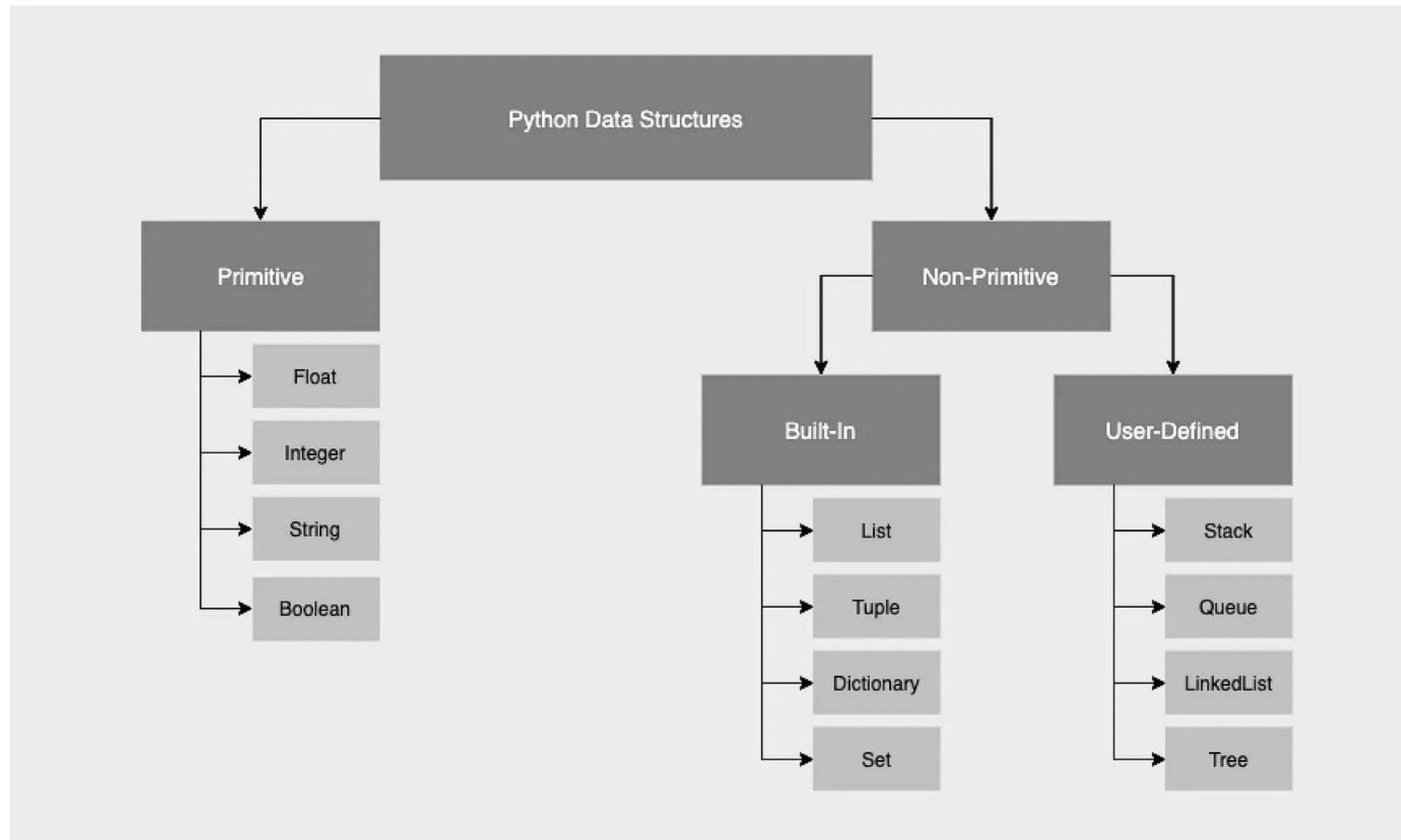
Agreement



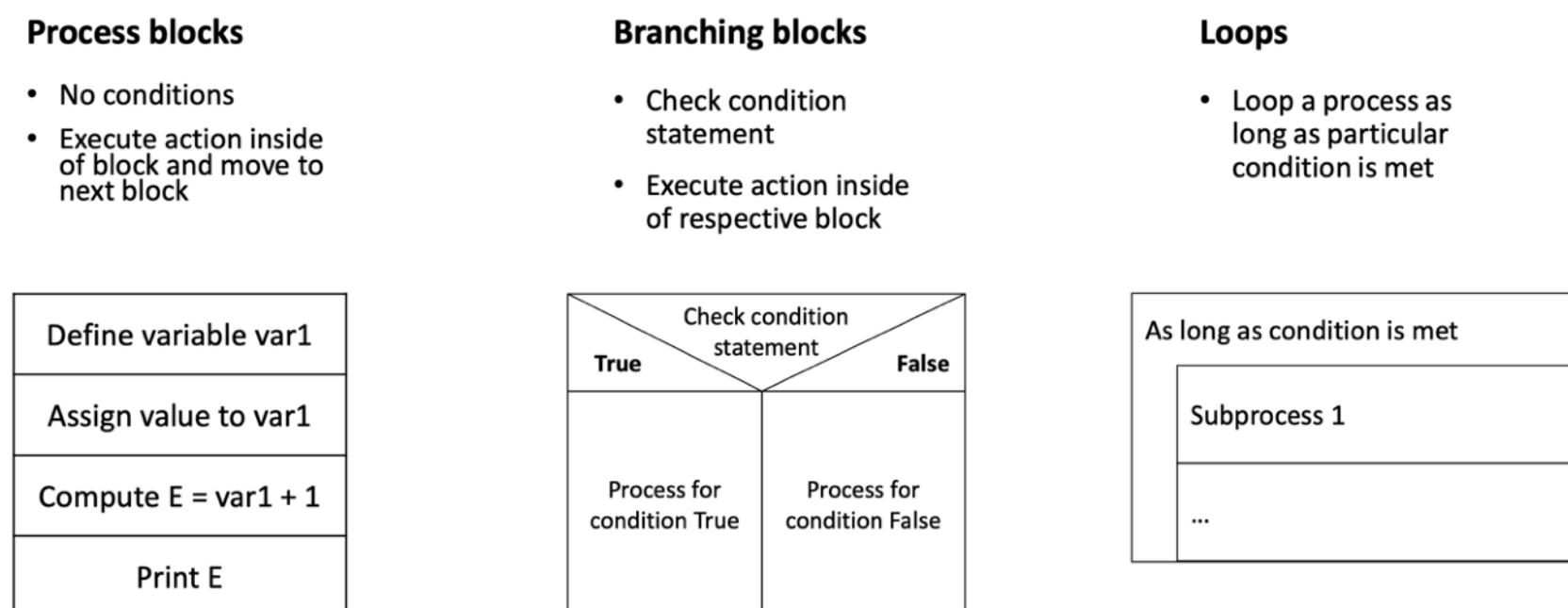
- Introduction to programming => All Questions are allowed!!!
- For beginners
- The module is interactive => Use your computer
- **We develop the solutions together!**
- Please be on time
- There are no dumb questions
- Nobody knows everything
- Copying solutions is plagiarism

A Glimpse Into Content

Primitive Data Structures

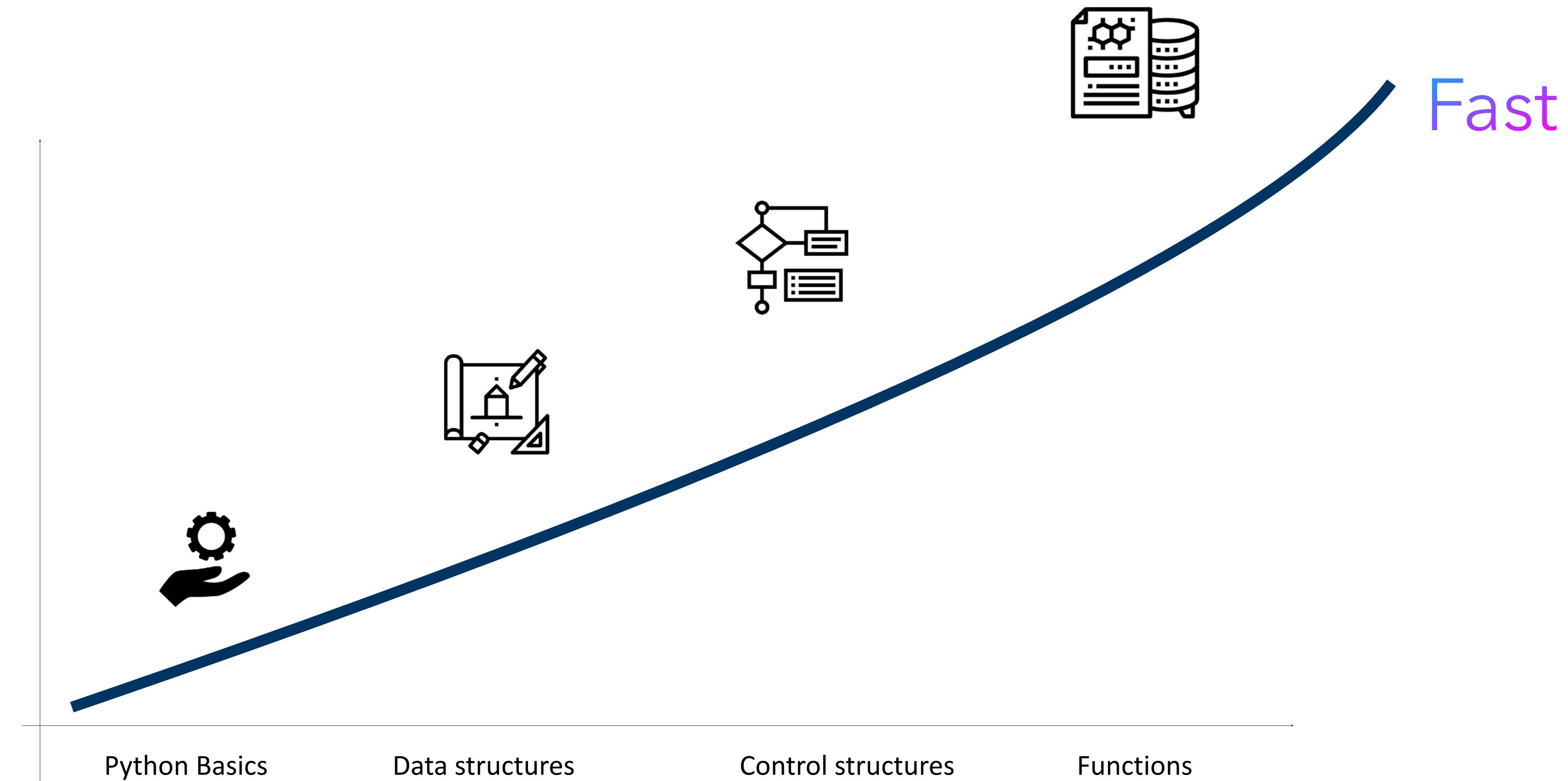


Control Structures in Python

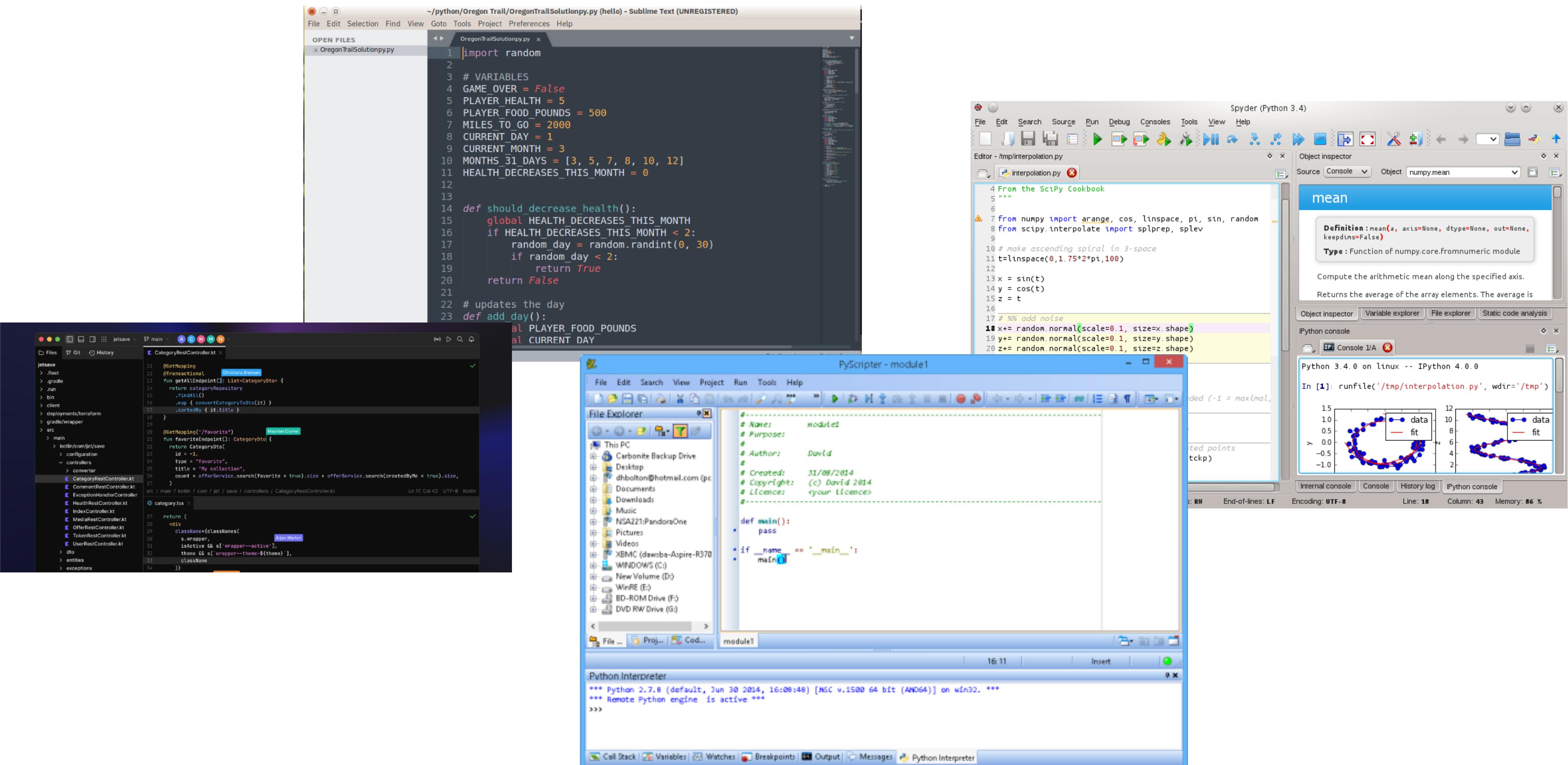


	Mutable	Ordered	Indexing	Duplicate Elements
Tuple		X	X	X
List	X	X	X	X
Set	X			
Dictionaries	X		X	

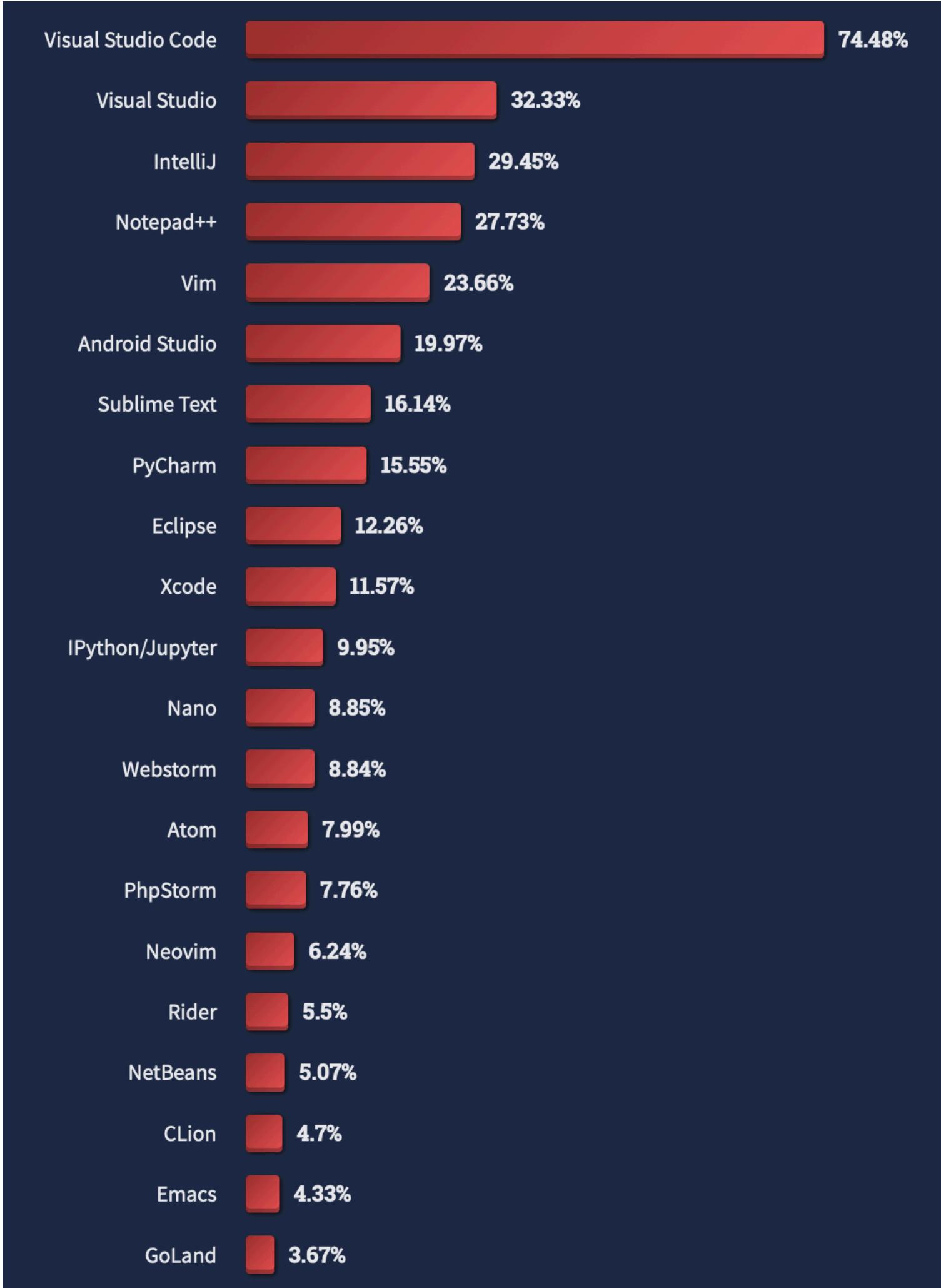
Note: Learning Curve for Today



Integrated Development Environment IDE a Deeper look? The Problem of Choice

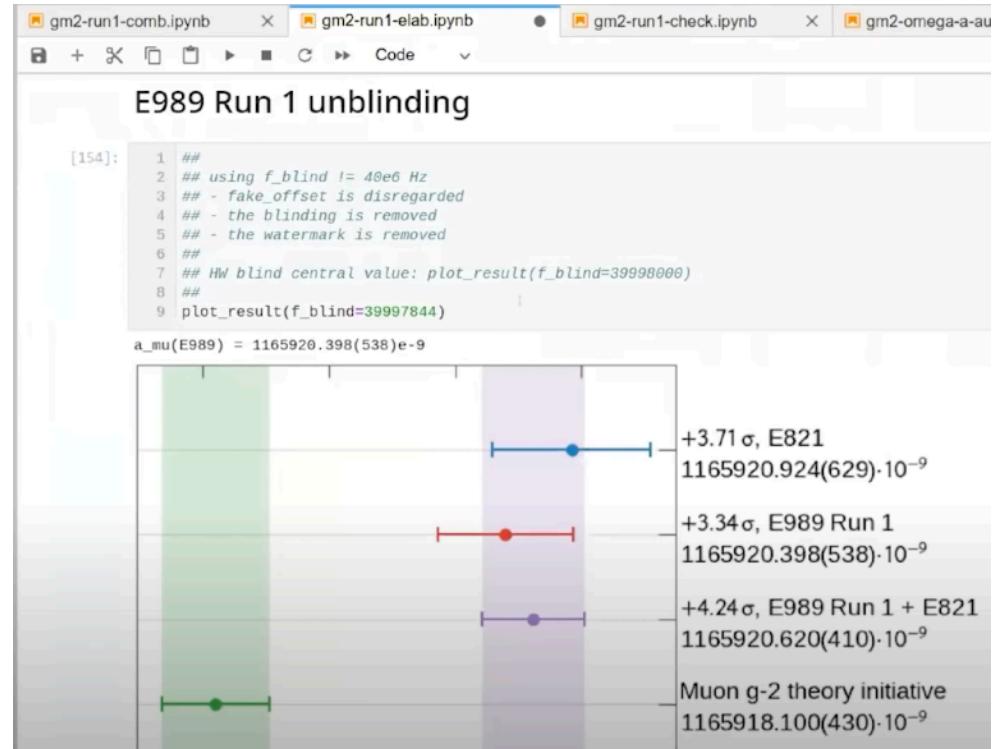


The Problem of Choice



<https://survey.stackoverflow.co/2022/#most-popular-technologies-new-collab-tools-prof>

A Question of Style

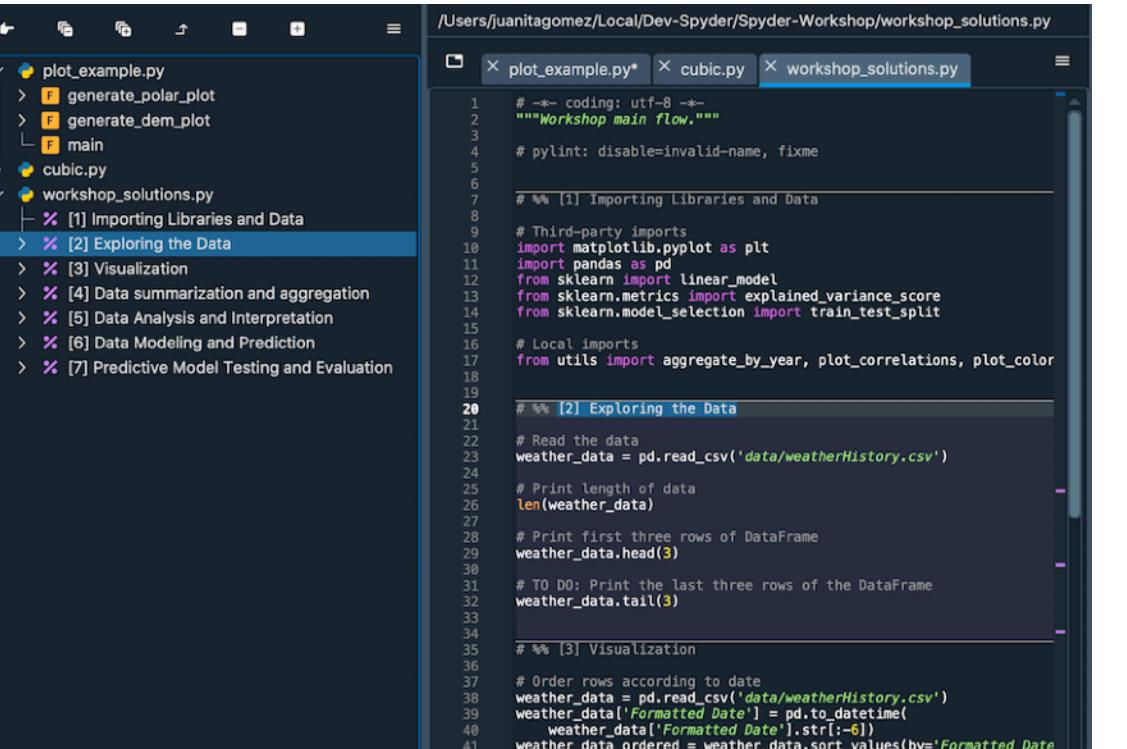


```
[154]: 1 ## using f_blind != 4@e6
2 ## - fake_offset is disregarded
3 ## - the blinding is removed
4 ## - the watermark is removed
5 ##
6 ## HM blind central value: plot_result(f_blind=39998000)
7 ## HM blind central value: plot_result(f_blind=39997844)

a_mu(E989) = 1165920.398(538)e-9
```

E989 Run 1 unblinding

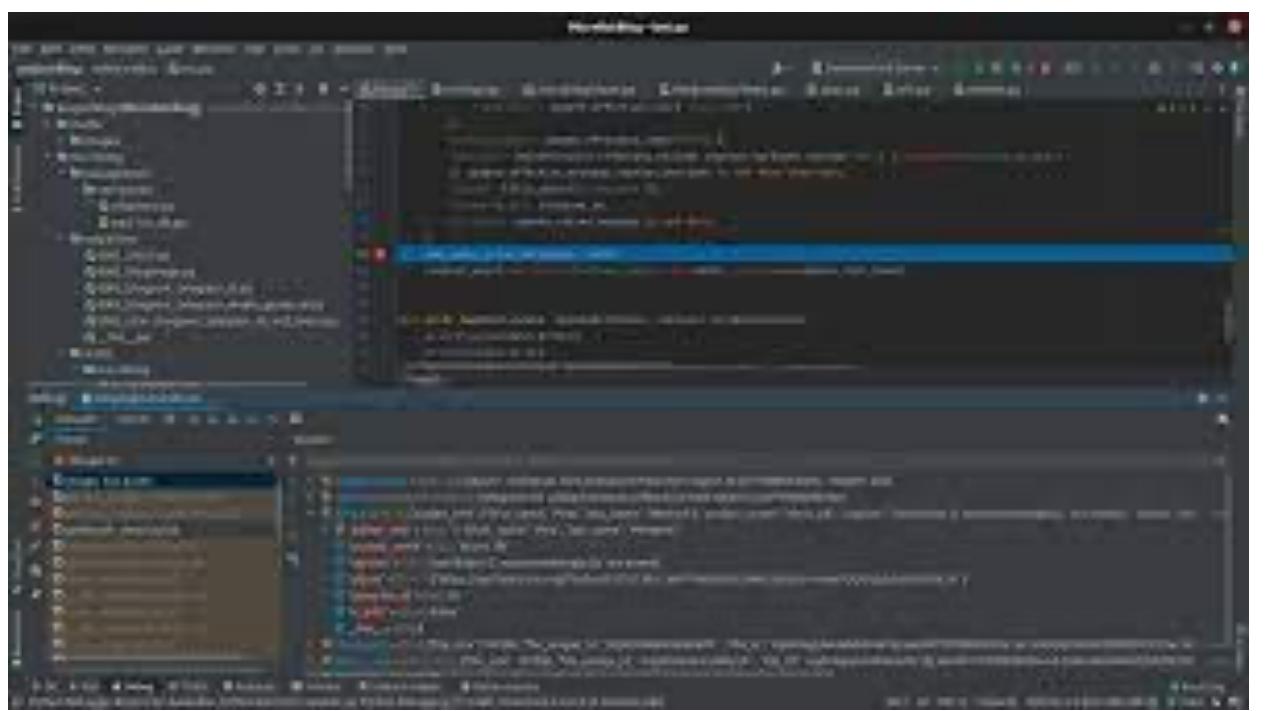
Notebooks (Research)*



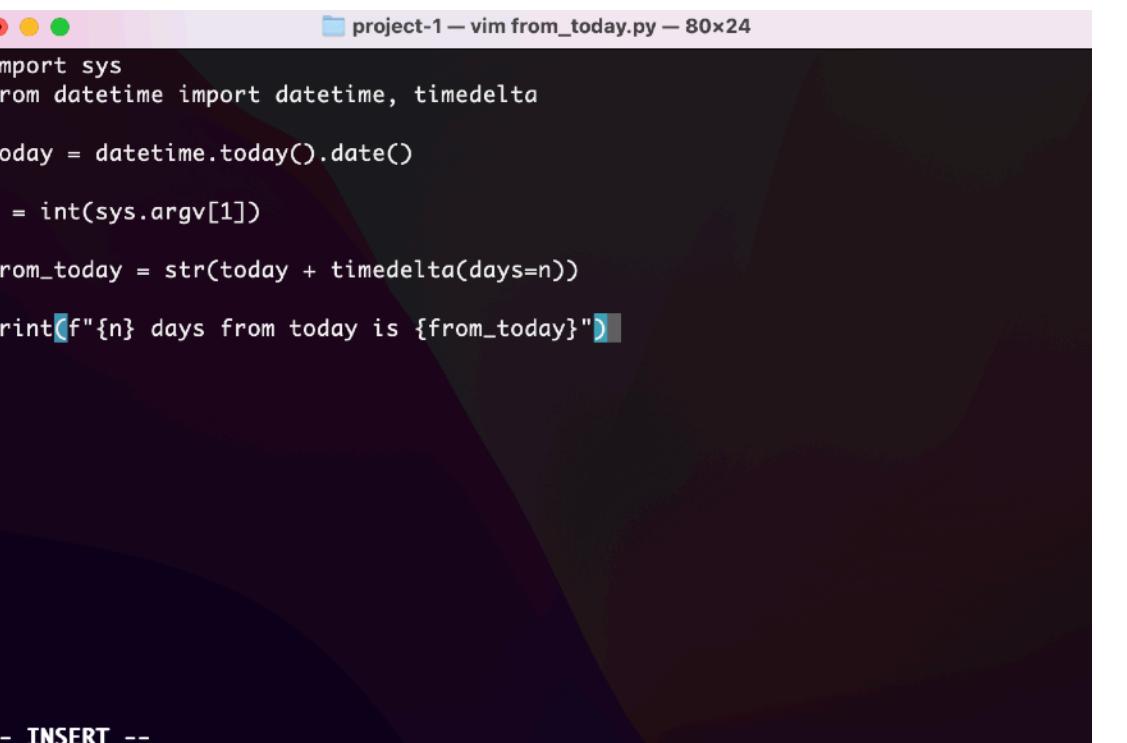
```
1 #-- coding: utf-8 --
2 """Workshop main flow."""
3
4 # pylint: disable=invalid-name, fixme
5
6 # %% [1] Importing Libraries and Data
7 import matplotlib.pyplot as plt
8 import pandas as pd
9 from sklearn import linear_model
10 from sklearn import metrics
11 from sklearn.model_selection import train_test_split
12
13 # Local imports
14 from utils import aggregate_by_year, plot_correlations, plot_color
```

plot_example.py

Hybrid #%% (standard cell separator)



Classical IDE Coding



```
import sys
from datetime import datetime, timedelta

today = datetime.today().date()

n = int(sys.argv[1])

from_today = str(today + timedelta(days=n))

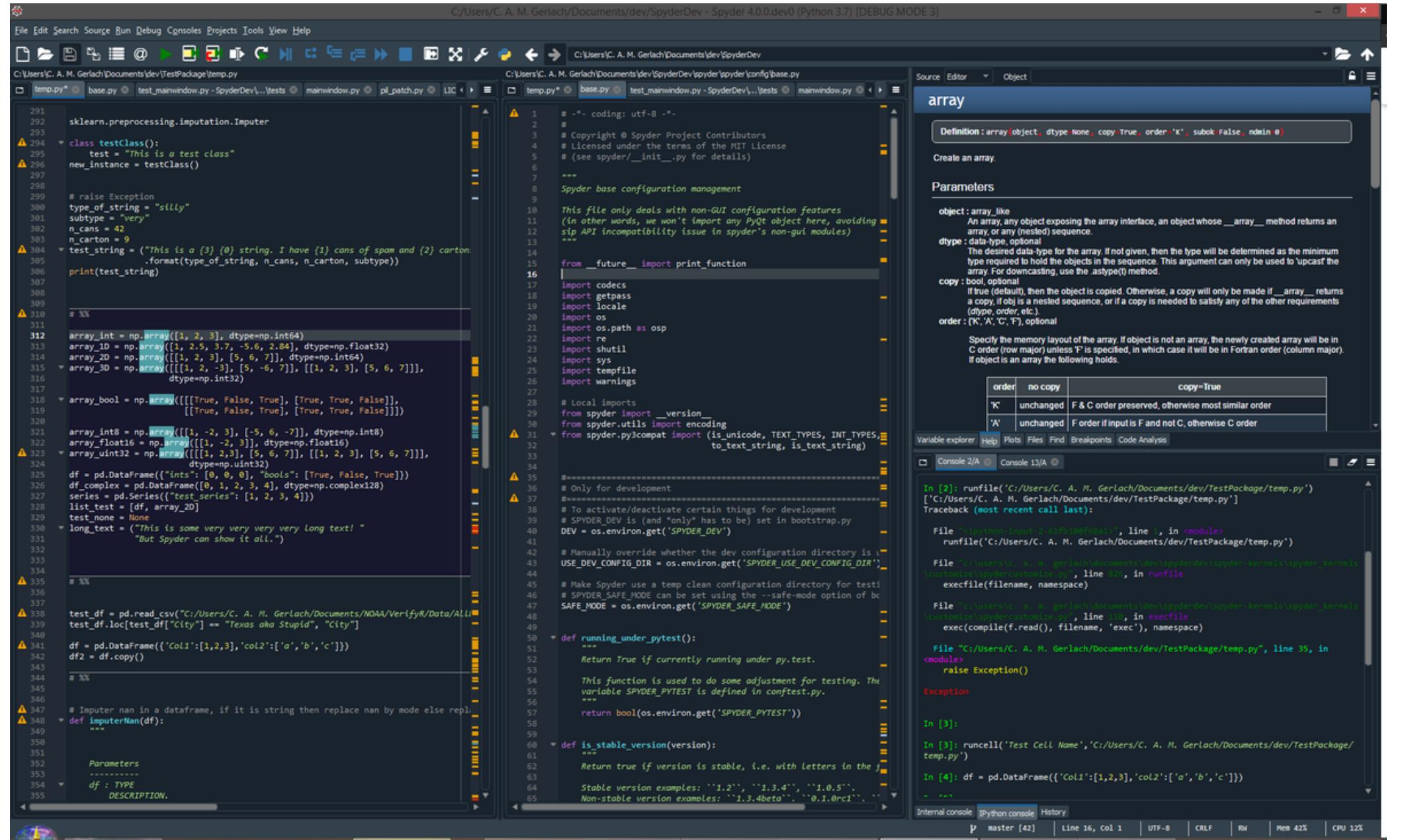
print(f"{n} days from today is {from_today}")
```

-- INSERT --

Coding via Terminal (Ultra Mode)

*) Famous experiment: https://www.youtube.com/watch?v=hL2w_FkTae8

What We Use for This Course

The screenshot shows the Spyder 4.0.0.dev0 Python 3.7 [DEBUG MODE 3] interface. The code editor displays a script named `temp.py` containing various Python code snippets, including imports from `sklearn.preprocessing`, `numpy`, and `pandas`. The documentation sidebar on the right provides details about the `array` object, including its definition, parameters (like `object`, `dtype`, `copy`, `order`, and `ndmin`), and examples. The console at the bottom shows the execution of several commands, including running a file and displaying the output of a `pd.DataFrame` print statement.

```

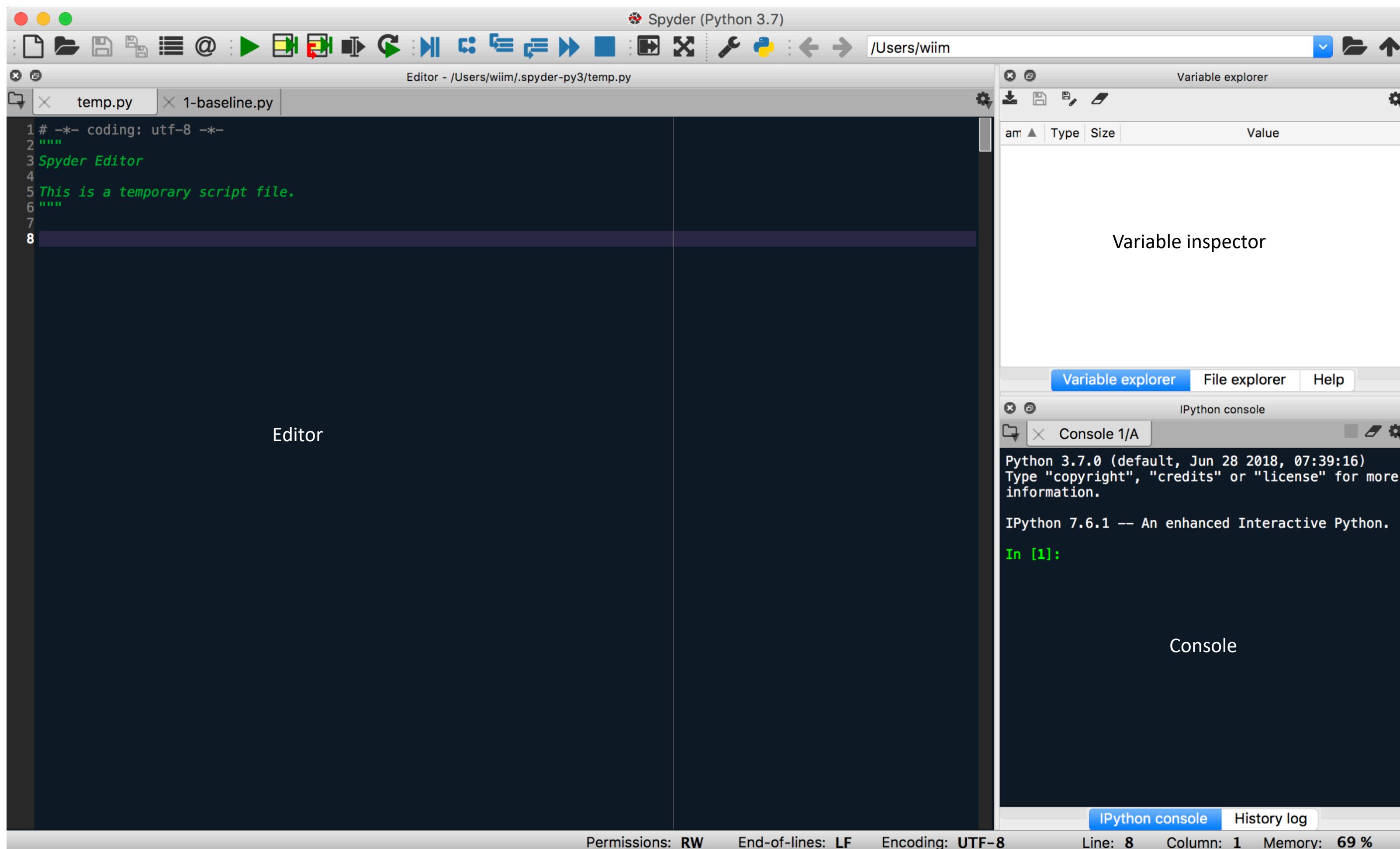
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:/Users/C. A. M. Gerlach/Documents/dev/spyderDev - Spyder 4.0.0.dev0 (Python 3.7) [DEBUG MODE 3]
C:/Users/C. A. M. Gerlach/Documents/dev/spyderDev/spyder/config/base.py
C:/Users/C. A. M. Gerlach/Documents/dev/spyderDev/spyder/mainwindow.py - SpyderDev - Tests - mainwindow.py
temp.py base.py test_mainwindow.py - SpyderDev - Tests - mainwindow.py
Source Editor Object
array
Definition: array(object, dtype=None, copy=True, order='K', subok=False, ndmin=0)
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.
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. This argument can only be used to 'upcast' the array. For downcasting, use the astype() method.
copy : bool, optional
If True (default), then the object is copied. Otherwise, a copy will only be made if __array__ returns a copy, if it is a nested sequence, or if a copy is needed to satisfy any of the other requirements (dtype, order, etc.).
order : {'C', 'F', 'A', 'K'}, optional
Specify the memory layout of the array. If object is not an array, the newly created array will be in C order (row major) unless 'F' is specified, in which case it will be in Fortran order (column major). If object is an array the following holds:
order | no copy | copy=True
-----|-----|-----
'K' | unchanged | F & C order preserved, otherwise most similar order
'A' | unchanged | Order if input is F and not C, otherwise C order
Variable explorer Help Plots Files Find Breakpoints Code Analysis
Console 2/A Console 13/A
In [2]: runfile('C:/Users/C. A. M. Gerlach/Documents/dev/TestPackage/temp.py')
[C:/Users/C. A. M. Gerlach/Documents/dev/TestPackage/temp.py]
Traceback (most recent call last):
File "C:/Users/C. A. M. Gerlach/Documents/dev/TestPackage/temp.py", line 1, in <module>
runfile('C:/Users/C. A. M. Gerlach/Documents/dev/TestPackage/temp.py')
File "C:/Program Files/Anaconda3/lib/python3.7/site-packages/spyder/config/utils.py", line 10, in runfile
execfile(filename, namespace)
File "C:/Program Files/Anaconda3/lib/python3.7/site-packages/spyder/config/utils.py", line 10, in execfile
exec(compile(f.read(), filename, 'exec'), namespace)
File "C:/Users/C. A. M. Gerlach/Documents/dev/TestPackage/temp.py", line 35, in <module>
raise Exception()
Exception
In [3]:
In [4]: runcell("Test Cell Name",'C:/Users/C. A. M. Gerlach/Documents/dev/TestPackage/temp.py')
In [4]: df = pd.DataFrame(['Col1':[1,2,3], 'Col2':['a','b','c']])
...
Internal console Python console History
P master [4] Line 16, Col 1 UTF-8 CR LF RD Mem 42% CPU 12%

```

Practice and Questions

- Take your computer and let's get started!
- Create a Folder 'Day 1' on our Desktop
- Launch Spyder and select the folder

In a Nutshell: Integrated Development Environment

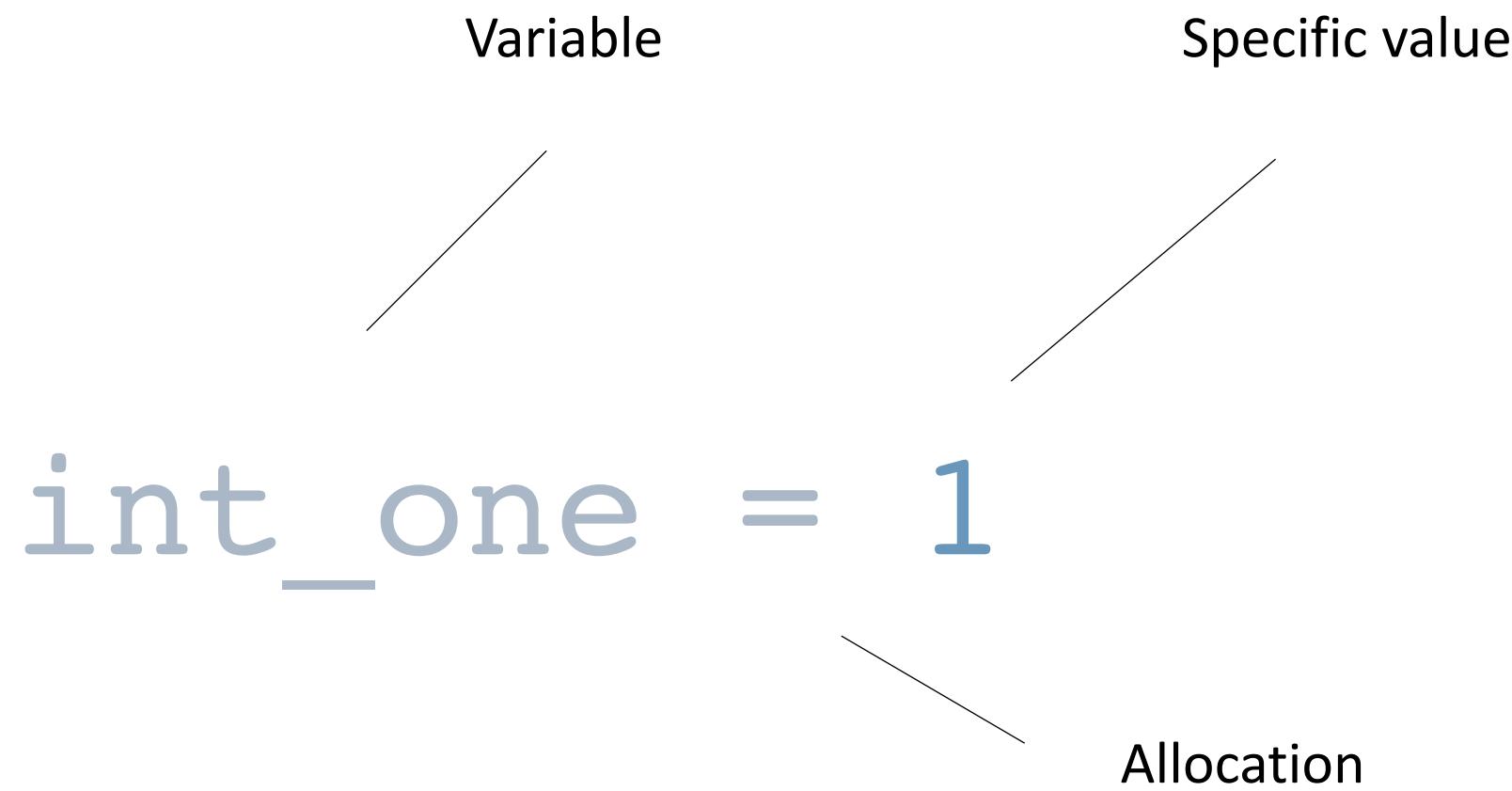


In a Nutshell: Hello World

```
print('Hello World')
```

Abstraction (Computer Science)

The first step - The concept of variable



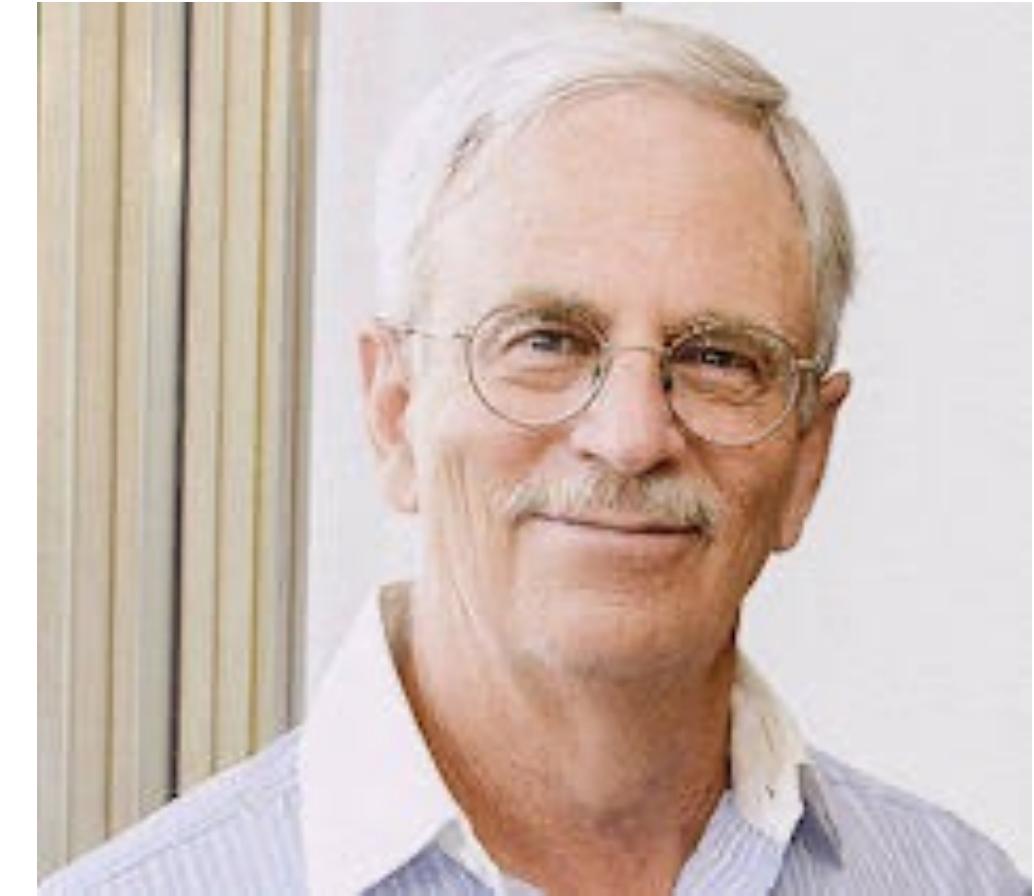
We assign the value 1 to the variable 'one'.

Now we can continue working with the variable 'one'.

Advantage: We are independent of concrete value

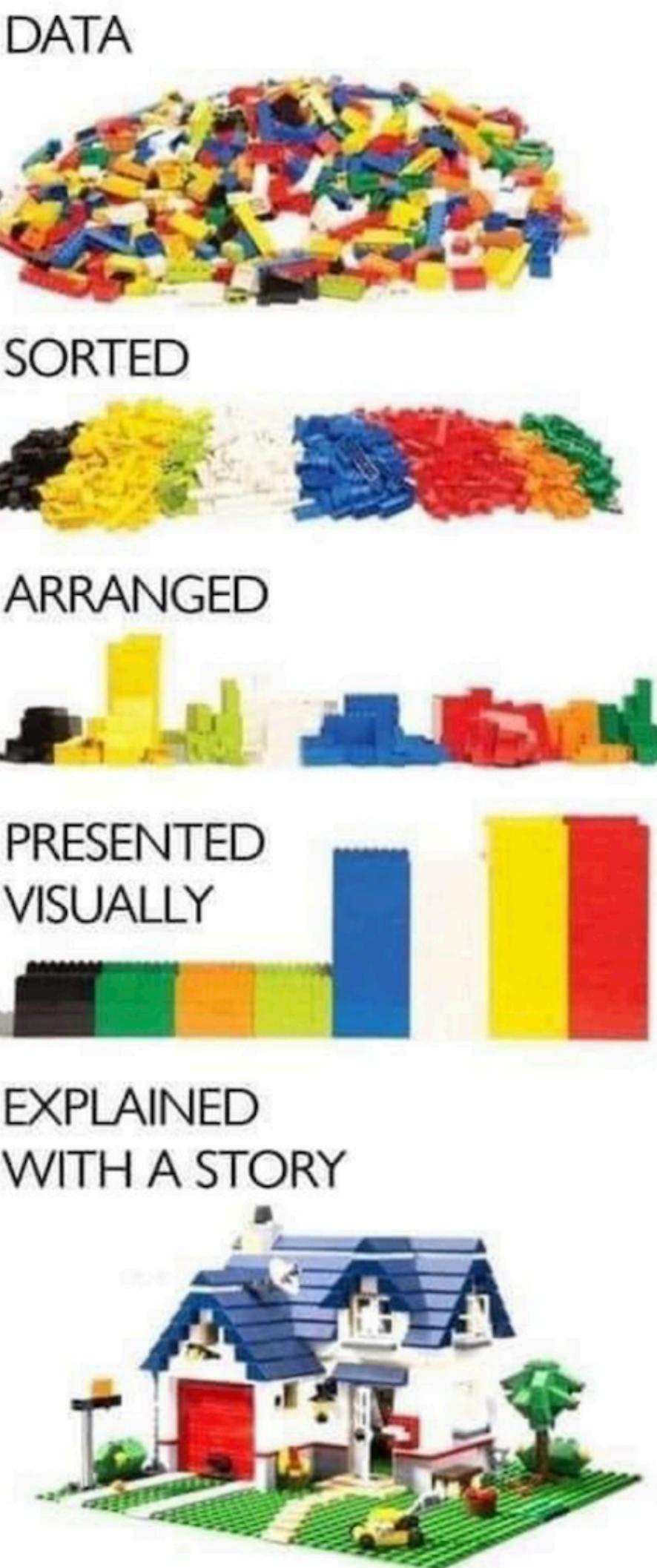
The essence of abstractions is preserving information that is relevant in a given context, and forgetting information that is irrelevant in that context.

– John V. Guttag



Different Building Blocks

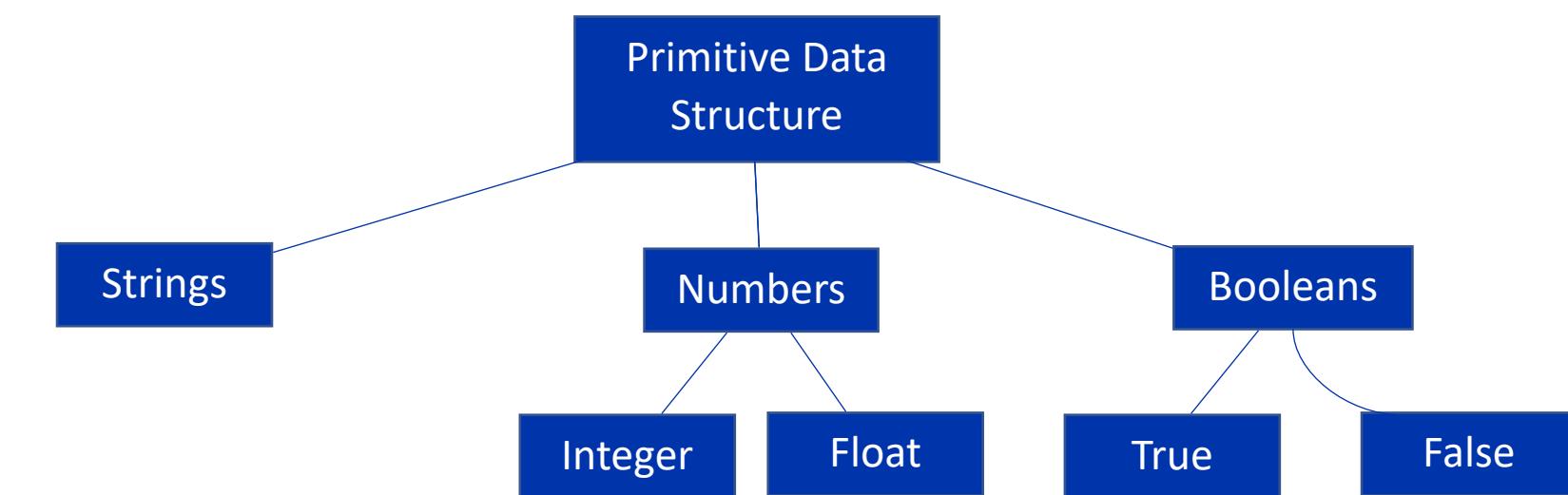
- Integer: int (signed integers) – They are often just called integers or ints, are positive or negative whole numbers with no decimal point.
- Float: Float (floating point real values, double) – Also called floats, they represent real numbers and are written with a decimal point dividing the integer and fractional parts. Floats may also be in scientific notation, with E or e indicating the power of 10.
- Strings are a sequence of chars. We can create them simply by enclosing characters in quotes. “Hello World” is a String! Therefore strings in Python are bytes representing Unicode characters. In Detail: Python does not have a character data type, a single character is simply a string with a length of 1.
- Boolean: We can perform logical operations with True and False in combination with AND and OR.



Primitive Data Structures

- Integer: int (signed integers) – They are often just called integers or ints, are positive or negative whole numbers with no decimal point.
- Float: Float (floating point real values, double) – Also called floats, they represent real numbers and are written with a decimal point dividing the integer and fractional parts. Floats may also be in scientific notation, with E or e indicating the power of 10.
- Strings are a sequence of chars. We can create them simply by enclosing characters in quotes. “Hello World” is a String! Therefore strings in Python are bytes representing Unicode characters. In Detail: Python does not have a character data type, a single character is simply a string with a length of 1.
- Boolean: We can perform logical operations with True and False in combination with AND and OR.

Primitive Data Structure: A First Overview



* Simplified Illustration

Python Documentation: <https://docs.python.org/3/library/stdtypes.html>

Other classification: Chun, W. (2001). Core python programming (Vol. 1). Prentice Hall Professional.

Documentation

Source Code Documentation

@author: name

@since: first implementation date

@version: date of last update

@source: if you using links etc.

@code: special code note

@param: if special parameter is used or you have to describe.

```
# I am a comment
@author: My Name
@since: 2022-10-03
@update: 2023-10-26
@version: v.0.0.2
print('Hello World')
```

Naming Convention

Select what you like! But be consistent!

- Names of attributes, variables, functions/methods* start with a small letter
 - may use letters without ß or similar
 - which points to the data type like i, s or l
- This is standard in professional software development.
- Camel Case: Compound words are written in programming language. Every new word is capitalized.

(More PeP8 Style)

- name = Is the name of...
- bscript_main = Simple code file that does something
- CName = Class (later more)
- float_name = Variable that saves a floating point (double)
- int_name = Variable that saves an integer value
- str_name = Variable that saves a string value
- b_name = Boolean for true or false values
- list_name = Object from type list
- method_name = Self-written function

(Camel Case Style)

- Name = Is the name of...
- bScriptName = Simple code file that does something
- CName = Class (later more)
- dName = Variable that saves a floating point (double)
- iName = Variable that saves an integer value
- sName = Variable that saves a string value
- bName = Boolean for true or false values
- LName = Object from type list
- fName = Self-written function

*) A **function** is a part of code that is called by the name definition. It can do an operations by the input parameters and can optionally return values).

A **method** is a part of an Object, is called by name that is associated with an object

Code is some kind of Art. Therefore...



Jackson Pollock Art

These are all approaches. Find your own style!

Let's Do Coding

Thank You

