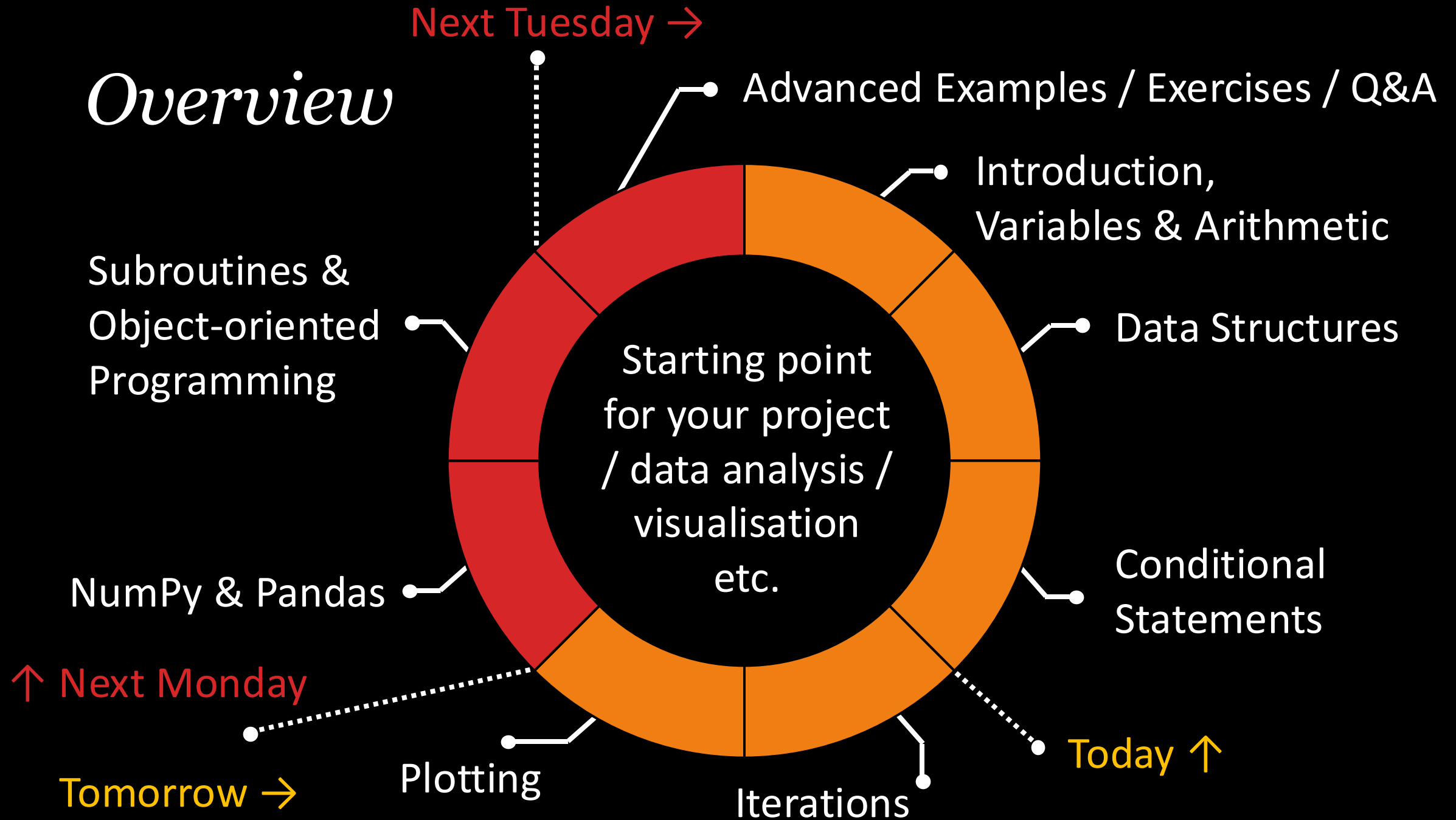


# Python Crash Course

Dr. Maxim Samarin  
Senior Data Scientist @ Swiss Data Science Center

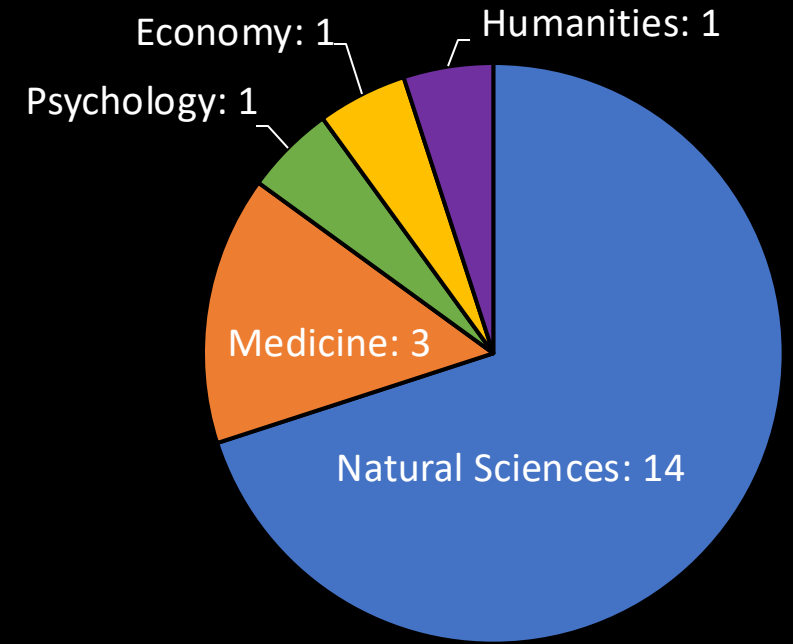
24<sup>th</sup> / 25<sup>th</sup> February and 3<sup>rd</sup> / 4<sup>th</sup> March 2025

# Overview



# *Your Experience and Goal*

- What is your **programming experience** so far and an **intermediate goal** for you?



Diell ---> Janine ---> Valentina ---> Joel ---> Anamaria --->  
---> Ghéreint ---> Lauren ---> Raphael ---> Manuel ---> Jonas --->  
---> Kakha ---> Mariia ---> Debdatto ---> Franziska ---> Ana --->  
---> Marietta ---> Jochen ---> Bernd ---> Eric ---> Cedric

# *My Programming Career*

**ETH** zürich



**ETH** zürich



School classes programming in **Java**

Studying Physics: **C++**, **MatLab**, **Python**

Project: Analysis & Graphical User Interface in **C**, **Python**

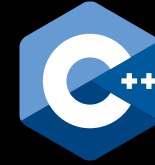
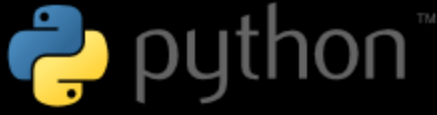
Physics modelling in **Fortran**, **Python**

M.Sc. thesis: Data Mining & simulations in **R**, **Python**

Ph.D. thesis: Machine / Deep Learning research in **Python** ...

Data Scientist and ML / AI Researcher : **Python** ...

# *Python vs. other Languages*



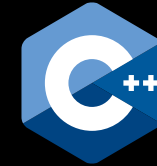
- Python is a dynamically typed programming (script) language
- Code is interpreted
- A lot of details *under the hood*
- **Zen** of Python: Simple, explicit, sparse, readable, practical

→ **Fast prototyping, quick to learn**

- Statically typed: fix type and object
- Code is compiled
- A lot of explicit control
- Efficient implementations, elaborate syntax

→ **Fast and powerful codes**

# *Python & C++ Examples*






```
1 name = input("Please enter your name: ")
2 print("Good morning,", name)
```

- Both have the same output:

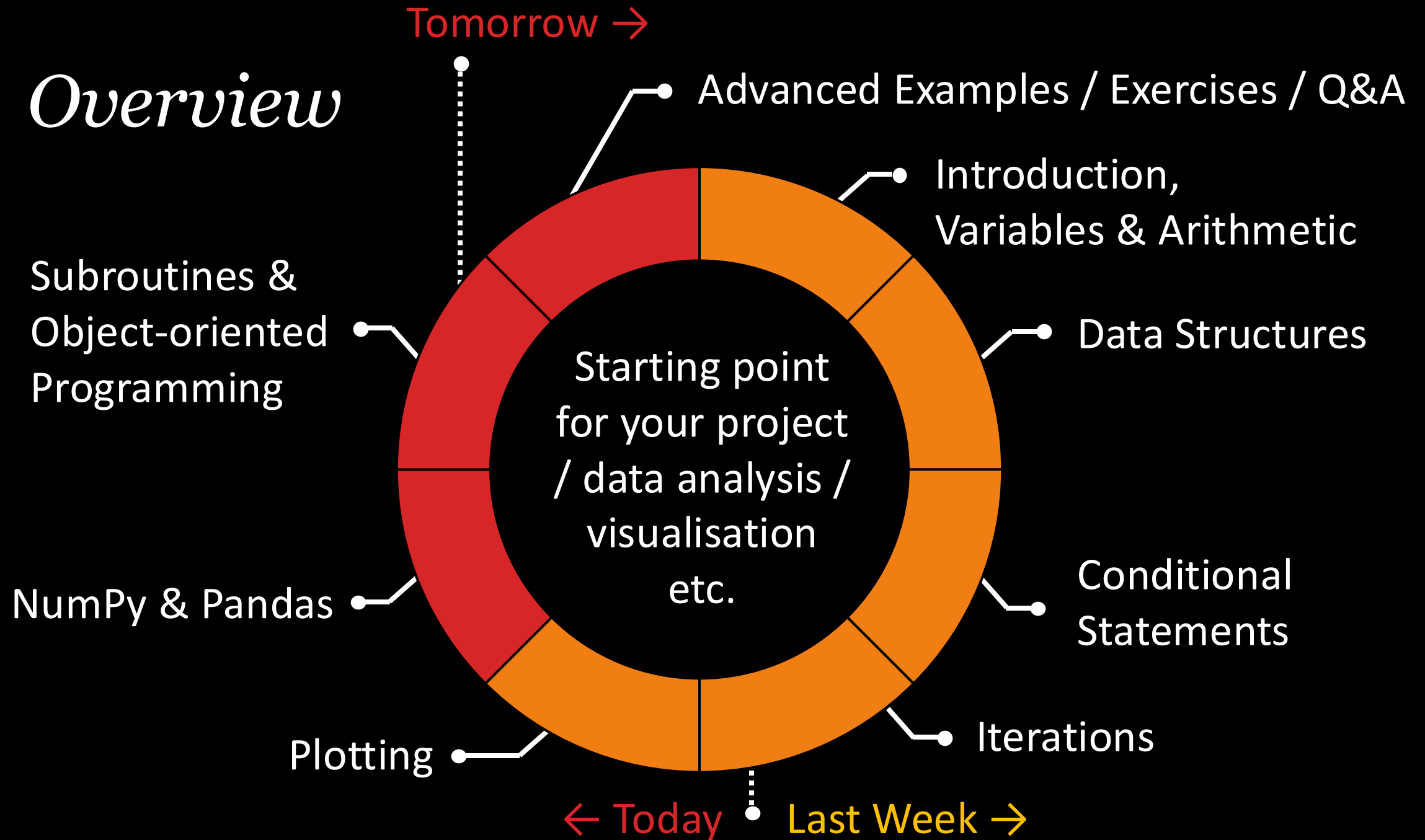
Please enter your name: Monty  
Good morning, Monty

```
1 #include <iostream>
2 #include <string>
3
4 using namespace std;
5
6 int main() {
7     string name;
8
9     cout << "Please enter your name: ";
10    cin >> name;
11    cout << "Good morning, " << name << endl;
12
13    return 0;
14 }
```

# *Start your Working Environment*

1. Access the course environment via **Renku link**, sign in with your SWITCH edu-ID and launch the session
  2. **Or:** Download new material, start Jupyter Lab and open notebooks
- Suggestions on how to work in this course:
    -  +  : Follow presentation, while executing scripts yourself, making adjustments and notes in your own notebook
    -  : Follow presentation, switch to programming environment for exercises

# Overview





# *Different Ways to Execute Scripts*

- **Jupyter Notebooks** (as in this class): Document-style, combining executable scripts with rich documentation and formatting  
→ Markdown and LaTeX can be used
- **Terminal**: Execute scripts via the terminal
  1. Write your script in an editor and save it to a *my\_script.py* file
  2. Execute the script in the terminal with *python my\_script.py*
- **Spyder** (beginner friendly): Integrated development environment (IDE) → editor with a terminal and other useful functionalities
- Or any other editors / IDE such as Atom, Sublime Text, PyCharm etc.

# *Advanced Setup: IDEs & Copilot*

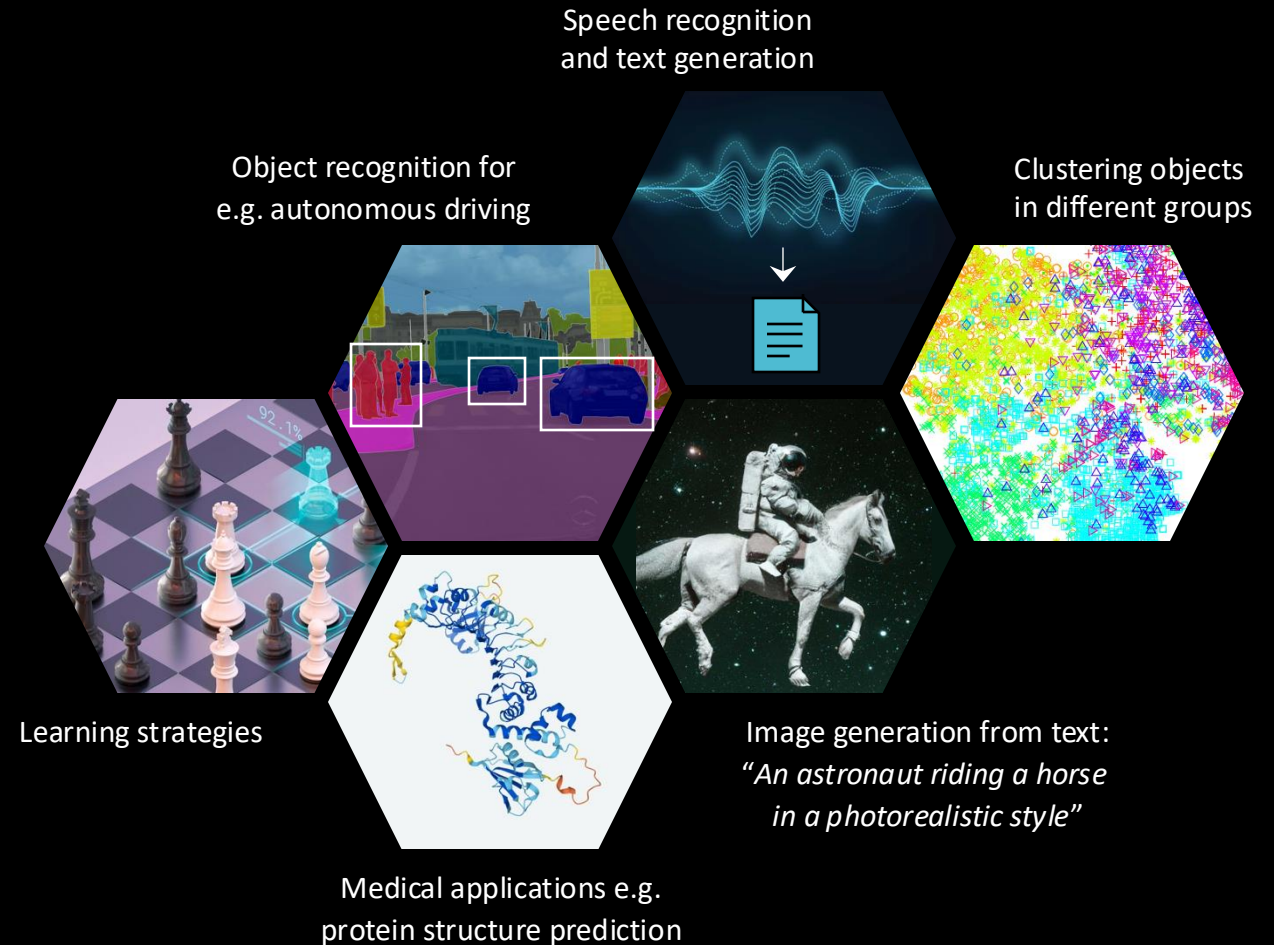
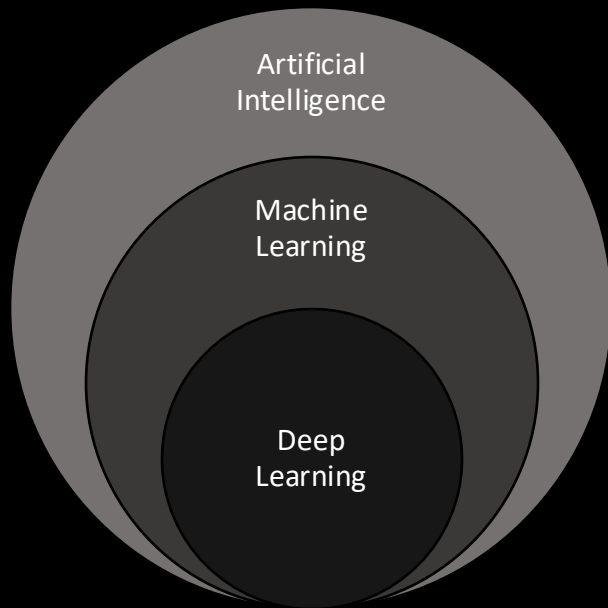
- **Visual Studio Code** (VS Code): Free for all
  - Installation: <https://code.visualstudio.com/>
- **PyCharm Professional**: Free for students (Ph.D.) & teachers (PostDoc)
  - Installation: <https://www.jetbrains.com/pycharm/>
  - License: <https://www.jetbrains.com/community/education/#students>
- **GitHub Copilot Pro**: Free for students (Ph.D.) & teachers (PostDoc)
  - Get a certificate of employment ("Arbeitsbestätigung") from HR (in EN)
  - Documentation for license: <https://docs.github.com/en/copilot/managing-copilot/managing-copilot-as-an-individual-subscriber/managing-your-github-copilot-pro-subscription/getting-free-access-to-copilot-pro-as-a-student-teacher-or-maintainer>

# *Selection of References*

- **Quick overview** with interactive tutorials on some basic topics (similar to the course) and more advanced concepts:  
<https://www.learnpython.org/>
- **More detailed overview** with interactive tutorials on a lot of topics:  
<https://www.w3schools.com/python/default.asp>
- One of my favourites with tutorials on specific (advanced) topics, easy-to-read books:  
<https://www.realpython.com/>
- Overview of resources for beginners:  
<https://wiki.python.org/moin/BeginnersGuide>

# Advanced Python & Machine Learning

- **Next course** (next spring):
  - More advanced Python concepts
  - Introduction to Machine Learning



# *Suggestions for the Feedback*



- Preparation information / YouTube videos adequate?
- What did / didn't you like about Renku?
- Content appropriate: Is anything missing for you?
- Too fast or slow, shallow or deep?
- Insightful exercises?

***Thank you and good luck as new Pythonistas! 😊***

# *Some Additional Links*

- Download the course material from  
<https://github.com/samarinm/pythonCC>
- On script languages:  
[https://en.wikipedia.org/wiki/Scripting\\_language](https://en.wikipedia.org/wiki/Scripting_language)
- On the difference between interpreter and compiler:  
<https://www.programiz.com/article/difference-compiler-interpreter>
- On dynamic and static type checking:  
[https://en.wikipedia.org/wiki/Type\\_system#Type\\_checking](https://en.wikipedia.org/wiki/Type_system#Type_checking)