Faculty of Economics and Management

Python as a tool of a modern student

Aleksej Gaj

Lecture 1
1.10.2025

Organisational info

- Aleksej Gaj, pythonforstudents24@gmail.com (or gajaleksej@pef.czu.cz)
- When & Where: **Wednesday 15.45 - 18.00, PEF-C26**
- lecture and tutorial are combined (přednáška+cvičení v jednom)
- the course is electible:
  - *5 credits* 😃
  - no *mandatory* homework 😃
  - *mandatory attendance and exam* 😕
- How to earn your 5 credits:
  - attendance (**max. of 2 absences are allowed**)
  - small project to test your progress in the middle of the semester
  - exam = prepare your own project in Python (details on the next lecture)

Course webpage

https://aleksejgaj.cz/pef_python/

What can you find there?

- up-to-date course schedule
- lecture slides
- code (jupyter notebooks)
- optional literature
- other useful info

What can you expect?

- learn Python at the user level (basics)
- get a simple but powerful tool for solving your tasks (BP/DP projects, homework, etc.)

Warning:

⚠️ This course is **not** “Programming in Python” - more like “introducing Python as a useful tool” *

⚠️ Do **not** expect that after this course you will be experts in Python

⚠️ This course will cover less than 10% of Python potential, but make you able to deepen your knowledge/skills independently.

* Those who are interested in “fundamental” knowledge see literature on webpage.

ChatGPT: do we need to learn coding?



- ChatGPT (OpenAI), Gemini (Google), GitHub Copilot, …
- **Good servant, but bad master.**
- Be able to construct solution -> leave technical steps to the servants.
- With deep understanding, implementation is a matter of time.
- Starting with implementation, you will be lost.

„Učit se bez přemýšlení je zbytečné. Přemýšlet bez učení je nebezpečné.“ - Konfucius

Syllabus (very preliminary)

1. Úvod do kurzu, úvod do programování (filozofie programování), programování jako nástroj, ilustrační příklady.
2. Úvod do programovacího jazyka Python, jeho výhody a nedostatky, přehled knihoven, seznámení s používanými nástroji (vývojové prostředí, lokální a vzdálené prostředí).
3. Základní syntaxe Pythonu (typy proměnných, podmínky, logické operátory, cykly, funkce).
4. Úvod do objektově orientovaného programování, příklady.
5. Numerické výpočty v Pythonu (knihovny NumPy, SciPy), symbolické výpočty v Pythonu (knihovna SymPy).
6. Práce s daty, čtení dat ze souboru (.txt, .csv, .xlsx), základní statistická analýza (knihovna Pandas).
7. Grafické zobrazení dat (knihovny seaborn, matplotlib).
8. Analýza dat (základní regresní model, grafy).
9. Demo strojového učení v Pythonu (představení knihoven, perceptron a interpretace, klasifikátor).
10. Zpracování obrazové informace (manipulace s obrazem v Pythonu, histogramy, šum).
11. Jednoduché GUI (knihovna Qt6/PySide6).
12. Práce na vlastních projektech.

What will we do? (syllabus explained)

Part 1:

- intro, Python - basic syntax, types, conditions, cycles
- numerical calculations, symbolic calculations

for beginners, necessary minimum to begin

Part 2:

- figures (different plots)
- data: loading dataset, preprocessing, statistics

Part 3:

- Python for basic ML tasks (regression, classification, …)

What will we do? (syllabus explained)

Part 1:

- intro, Python - basic syntax, types, conditions, cycles
- numerical calculations, symbolic calculations

Part 2:

- figures (different plots)
- data: loading dataset, preprocessing, statistics



useful tools for working on projects, BP/DP, etc.

Part 3:

- Python for basic ML tasks (regression, classification, …)

What will we do? (syllabus explained)

Part 1:

- intro, Python - basic syntax, types, conditions, cycles
- numerical calculations, symbolic calculations

Part 2:

- figures (different plots)
- data: loading dataset, preprocessing, statistics

Part 3:

- Python for basic ML tasks (regression, classification, …)

advanced + more time demanding

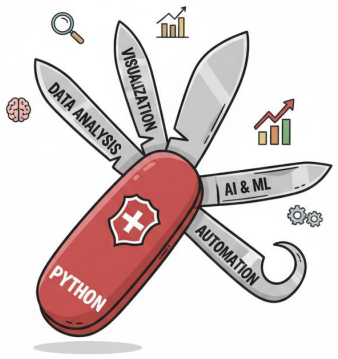
How will we work?

“I hear and I forget. I see and I remember. I do and I understand.” - Confucius

- classical lectures - apprx. 10% of time
- working with code - 60% of time
- exercises - 30% of time
- optional homework - up to you

End of organisational part. Questions?

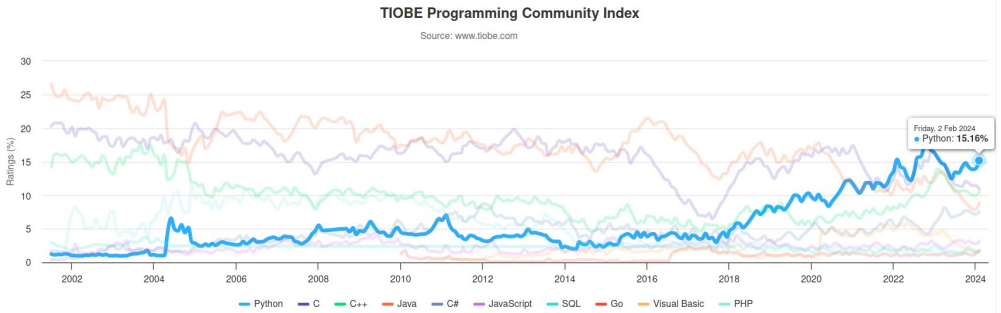
Why Python is so popular?



- open source => anyone can contribute
- free (as in ‘free beer’, as in ‘free speech’)
- easy to learn (meant to be understood with only knowledge of English)
- huge variety of libraries available
- active & growing community

Weakness:

- slower than compiled languages



Popularity:

- PYPL: https://pypl.github.io/PYPL.html
- Google Trends: https://shorturl.at/psOT0
- TIOBE: https://www.tiobe.com/tiobe-index/

Python - a bit of history…



- 1989: Guido van Rossum had been looking for a “hobby programming project that would keep him occupied during the week around Christmas” as his office was closed when he decided to write an interpreter for a “new scripting language (...): a descendant of ABC that would appeal to Unix/C hackers”
- Name origin: "slightly irreverent mood” and *Monty Python's Flying Circus*
- v.0.9.0 - 1991
- v.1.0.0 - 1994
- v.3.12 - 2023 (modern version)

Zen of Python



Core philosophy (coding conventions) the Zen of Python (‘`import this`’):

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Goals (“Computer Programming for Everybody”, 1999):

- An easy and intuitive language **just as powerful as major competitors**
- **Open source**, so anyone can contribute to its development
- Code that is as **understandable as plain English**
- **Suitability for everyday tasks,** allowing for short development times

Live demo - how to run&work with Python

in this course we will work with: Google Colaboratory (colab.research.google.com)

- online - run in browser without installation
- needs to sign in via Google account
- convenient (same installation, free GPU), intuitive

Local installation (Python + IDE):

- installation: https://www.python.org/
- IDE = intelligent development environment: an editor with code completion, ability to run it, test it, etc. …
- ex. **PyCharm**, VS Code/VS Codium, Spyder, Thonny…

Script vs. ‘notebook’

script - `.py` file

- ➕ classic code
- ➕ fast to load
- ➕ small size of file
- ➕ no additional SW needed to read it

- ➖ no markdown, only comments
- ➖ to try something you need to run it whole

jupyter notebook - `.ipynb` file

- ➕ Julia+Python+R
- ➕ ability to run small pieces of code and see the input immediately
- ➕ supports pictures, Markdown, basic formulas, html, …

- ➖ sometimes big file
- ➖ only interactive use
- ➖ only GUI use

Google Colab works only with `.ipynb` files.

Python - how the code is compiled?

Stage	**Python** (Interpretation)	**C++** (Compilation)
Preprocessing	No preprocessing step (comments remain)	Expands header files, removes comments, defines macros
Compilation	No compilation step; code remains human-readable	Translates C++ code to machine code specific to the CPU
Bytecode	Bytecode generated on the fly during interpretation	No bytecode; code directly compiled to machine code
Output File	Script file (main.py) remains unchanged	Executable file (e.g., main.exe on Windows)
Execution	interpreter reads and executes bytecode line by line	OS directly executes the machine code
Portability	Script can run on any system with a Python interpreter	Compiled executable might not run on other systems
Speed	Can be slower due to interpretation	Generally faster due to native machine code execution

Questions?

Optional homework:

- play with Colab
- install Python and PyCharm on your laptop
- try to create a program which writes your name and wether year of your birthday is odd or even.

Sources of images



- https://en.wikipedia.org/wiki/Guido_van_Rossum#/media/File:Guido-portrait-2014-drc.jpg
- https://en.wikipedia.org/wiki/History_of_Python#/media/File:Python_logo_1990s.svg
- https://en.wikipedia.org/wiki/History_of_Python#/media/File:Python_logo_and_wordmark.svg
-
-