FÖ1 Course introduction and introduction to Python

TNK128 HT2025

Nils Breyer

TNK128 Fundamental programming for data analytics

Why this course?

- Programming is everywhere
- You will need programming for coming courses
- Even if you don't become a full-time programmer basic knowledge is important

Agenda

- Course information
  - Teachers
  - Aims
  - Organization
  - Material
  - Feedback
- Introduction to Python
- Computations
- Input/output
- Debugging

Fire safety

- Know where the nearest emergency exit, assembly point, fire extinguisher equipment and alarm button are.
- In case of fire
  - Rescue
  - Warn others
  - Raise the alarm
  - Extinguish
  - Evacuate
  - Gather at the assembly point

See: https://liuonline.sharepoint.com/sites/student-campus-och-lokaler/SitePages/brand.aspx

Nils Breyer

- Assistant professor in traffic systems at KTS
- Background
  - BSc Computer Science, TU Braunschweig
  - MSc Intelligent Transport Systems and Logistics, LiU
  - PhD in Infrainformatics (2021), LiU
- Interests
  - Data analysis, railway and public transport
  - CoderDojo Norrköping
- Languages: English, Swedish, German

Teachers in the course

Nils

Lucie

Mohamud

Tatiana

Anastasia

Course aims

In this course, you will learn how to use programming for problem solving and analysis of data.

After completing the course, the student should be able to:

- Write **scripts for data analysis** using **Python**
- Use **basic data structures** for problem solving in P**ython**
- Apply tools available in some commonly used **Python packages**
- Generalize programming skills in Python to other script languages, specifically **Matlab**

Course contents

- Introduction to different types of programming paradigms and languages
- Python basics: programming environment and documentation, program flow, variables, comments, numerical operators, loops, conditional statements
- Python data structures and looping techniques: tuples, lists, dictionaries, sets, iterators, and generators
- Python standard libraries and essential third-party packages for data manipulation, numerical computing, and visualization
- Debugging of code
- Data retrieval from various sources, such as json files, csv files, html files, XML files, databases or APIs
- Introduction to Matlab programming and toolboxes

Organization

- All information is distributed through Lisam
- The teaching plan gives an overview
- Lectures
  - Weekly lectures introducing a new topic
  - Recommended literature
- Tutorial
- Labs
  - Weekly lab sessions
  - Groups of two students

Examination

- *LAB1* (Python, 1,5hp) and *LAB2* (Matlab, 1,5hp)
  - LAB1: Homework 2-4 (Hand-ins and oral examination)
  - LAB2: Homework 5-6 (Hand-ins and oral examination)
  - Graded Pass/fail
- *DAT1* (Computer exam in Python, 3hp)
  - Similar to a written exam, but in a computer room
  - Graded U, 3, 4, 5
- Course grade is equal to the grade for DAT1

Computer labs

- Register for a group of two students
- Two parallel lab sessions
- 6 labs, of which 5 will be examined with
  - written submissions and
  - two oral examinations

Material

- Downey, A. B. (2024). Think Python: How to Think Like a Computer Scientist
- McKinney, W. (2022). Python for Data Analysis: Data wrangling with pandas, NumPy and Ipython, 3rd Edition. O'Reilly Media.
- Python Documentation
- Additional material for Matlab

Feedback from last year

- Last year's course evaluation: grade 3.50 (35% response rate)
- Labs are a great for learning and practicing
- Oral examination good to check understanding
  - This year fewer but broader examinations
- Gap between lectures, labs and exam
  - More practical examples in the lectures
  - More quizzes after to reflect on lecture content
  - New Q&A session before the exam

Chat-GPT

Do's	Dont's
Ask for feedback	Use lab assignment as prompt
Use as inspiration	Copy the answer
Ask for help when debugging/lost	Generate documentation

Keep in mind:

- You must be able to explain your code
- You will have no access to the internet during the computer exam

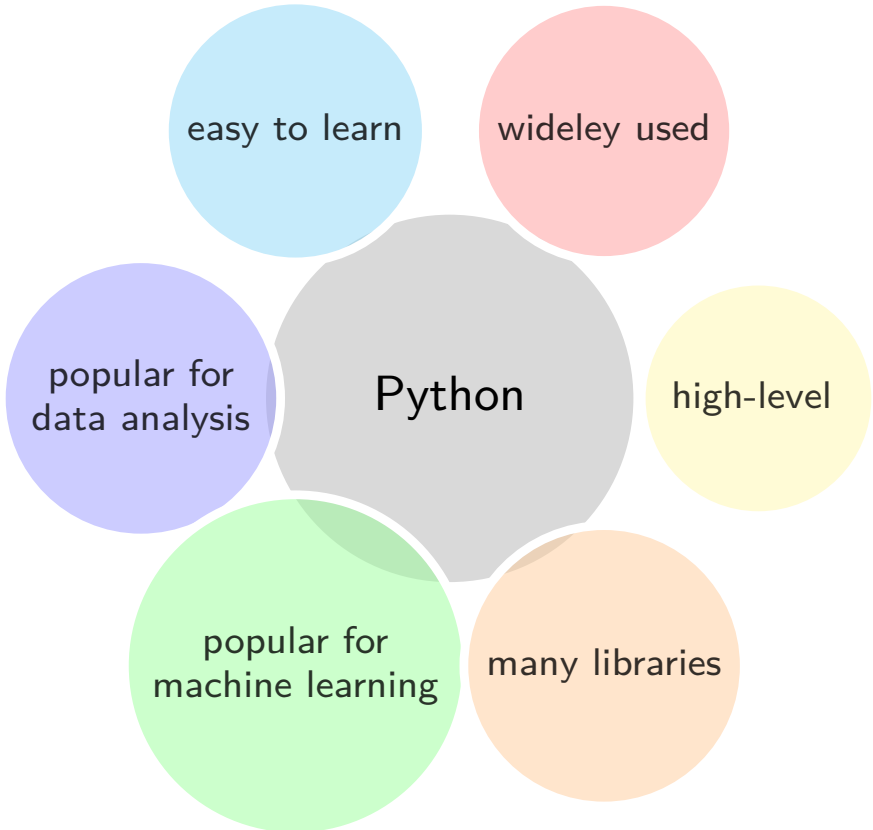
What is programming?

- Writing a sequence of instructions that specifies how to perform a computation
  - Input -> [Program] -> Output
- Different programming paradigms
  - imperative vs declarative
- Formal vs natural language

Self-judgement test

$=>$ *Mentimeter*

Why Python?


easy to learn
wideley used
popular for data analysis
Python
high-level
popular for machine learning
many libraries

The Python language

Figure 1: Interpreter

```
$ python
Python 3.11.4 (main, Jul 25 2023, 17:36:13) [Clang 14.0.3 (cl
Type "help", "copyright", "credits" or "license" for more inf
>>> print(1 + 1)
2
```

Two ways to use Python

Interactive mode

- Jupyter notebook or terminal
- Output is printed by default
- Useful for smaller projects, experimenting, data analysis

Script mode (use .py files)

- Text editor and terminal or IDE (VS Code,..)
- No output by default
- Debugger can be used through IDE
- Useful for bigger projects, application or web development

Programming environment

Tool	Examples
Python interpreter	Python 3.11.4
Package manager/Distribution	pip/Anaconda
Text editor	Notepad++, Sublime
Integrated Development Environment (IDE)	VS Code, PyCharm, ...
Interactive notebook	Jupyter

- For the labs we recommend to use Jupyter notebooks
  - Available through Anaconda in the computer rooms

Computations

```
40 + 2
```

```
42
```

```
84 / 2
```

```
42.0
```

```
1 + 82 / 2
```

```
42.0
```

Computations

```
40 + 2, type(40 + 2)
```

```
(42, int)
```

```
84 / 2, type(84 / 2)
```

```
(42.0, float)
```

```
1 + 82 / 2, type(1 + 82 / 2)
```

```
(42.0, float)
```

Variables

> A *variable* is a name that refers to a value

```
message = "Hello world"
print(message)
```

```
Hello world
```

Naming variables

- Names may not start with numbers or contain special characters (except underscore)
- Names may not be a Python keyword

```
import keyword
print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'aw
```

Naming variables

- Bad variable names: `num_s`, `s`, `total_NumberOfStudentsInClass`
- Good variable names: `numberStudents`, `students`

Two common naming styles: `camelCase` `snake_case`

Good programming practice

- Use short, but meaningful names
- Use consistent style (messageStr/message_str)

Input/Output

- Command line and files are common ways for input/output
- But also: Databases, Application Programming interfaces (API), Graphical user interfaces (GUI)

```
a = input()
print(a)
```

Regarding examples for input, see Think Python Chapter 5.11.

Debugging

Syntax error

A *syntax error* is a violation of Python's rules for correctly written code.

Example:

```
print(1+2
```

```
SyntaxError: incomplete input (664251341.py, line 1)
  Cell In[9], line 1
    print(1+2
              ^
SyntaxError: incomplete input
```

Debugging

Runtime error

A *runtime error* that occurs during execution and leads to unexpected behavior, typically raising an *Exception*.

Example:

```
99/0
```

```
ZeroDivisionError: division by zero
---------------------------------------------------------------
ZeroDivisionError                                 Traceback (most rec
Cell In[10], line 1
----> 1 99/0
```

Debugging

Semantic error

A *semantic error* that causes the script to run without exceptions, but doing something that was not indented.

Example: Calculate the average of 3 and 5

```
3 + 5 / 2
```

```
5.5
```

Further readings

- Downey, A. B. (2024). Think Python: How to Think Like a Computer Scientist, 3rd Edition. O'Reilly Media.
  - Chapter 1
  - Chapter 2