Week 1: Introduction

Basic Programming in Python

Julia Wippermann, Robin Horn,
Kamran Vatankhah-Barazandeh, Leonard Frommelt

April 9, 2021

UNIVERSITÄT OSNABRÜCK

Who we are

1. Julia Wippermann (jwippermann@uos.de):
   2nd Semester Info Master / Bachelor Lehramt
2. Robin Horn (rhorn@uos.de):
   6th Semester CogSci
3. Kamran Vatankhah-Barazandeh (kvatankhahba@uos.de):
   6th Semester CogSci
4. Leonard Frommelt (lfrommelt@uos.de):
   10th Semester CogSci Bachelor

Who this course is for

You are in the right course if...

a. You are a master student coming from a non-technical discipline
b. You have little to no experience with programming
c. You felt a little overwhelmed by Informatik A (Algorithmen & Datenstrukturen) and would like to repeat the core principles of programming with another language

Who this course is for

You are NOT in the right course if...

a. Informatik A / AuD was a piece of cake for you and you would just like to learn another language
   $\rightarrow$ Scientific Programming in Python
b. You already know to program in Python or another language
   $\rightarrow$ You will not learn anything in this class
c. You have a specific application area that you want to learn about in detail
   $\rightarrow$ Specialized courses (CV, CL, ML etc.)

Tentative Schedule

1. Hello World
2. Variables & Assignments
3. Control Structures
4. Data Structures
5. Strings & Formatting
6. Input & Output
7. Debugging & Good Practices
8. Built-In Packages
9. Object-Oriented Programming

→ More lectures on external packages → Working on projects

Structure

1. Lecture:
   - Uploaded in Courseware on Monday 8am
2. Coding Support
   - Live-Sessions for questions and help with the homework
   - Each Thu 16.00-18.00 and Mo 12.00-14.00
   - BBB (StudIP $\rightarrow$ Meetings)
3. Homework
   - Uploaded under Files&Vips on Monday 8am
   - Hand in until next Tuesday 23:59:59 via StudIP $\rightarrow$ Vips
   - $\rightarrow$ You have 1.5 weeks to work on it

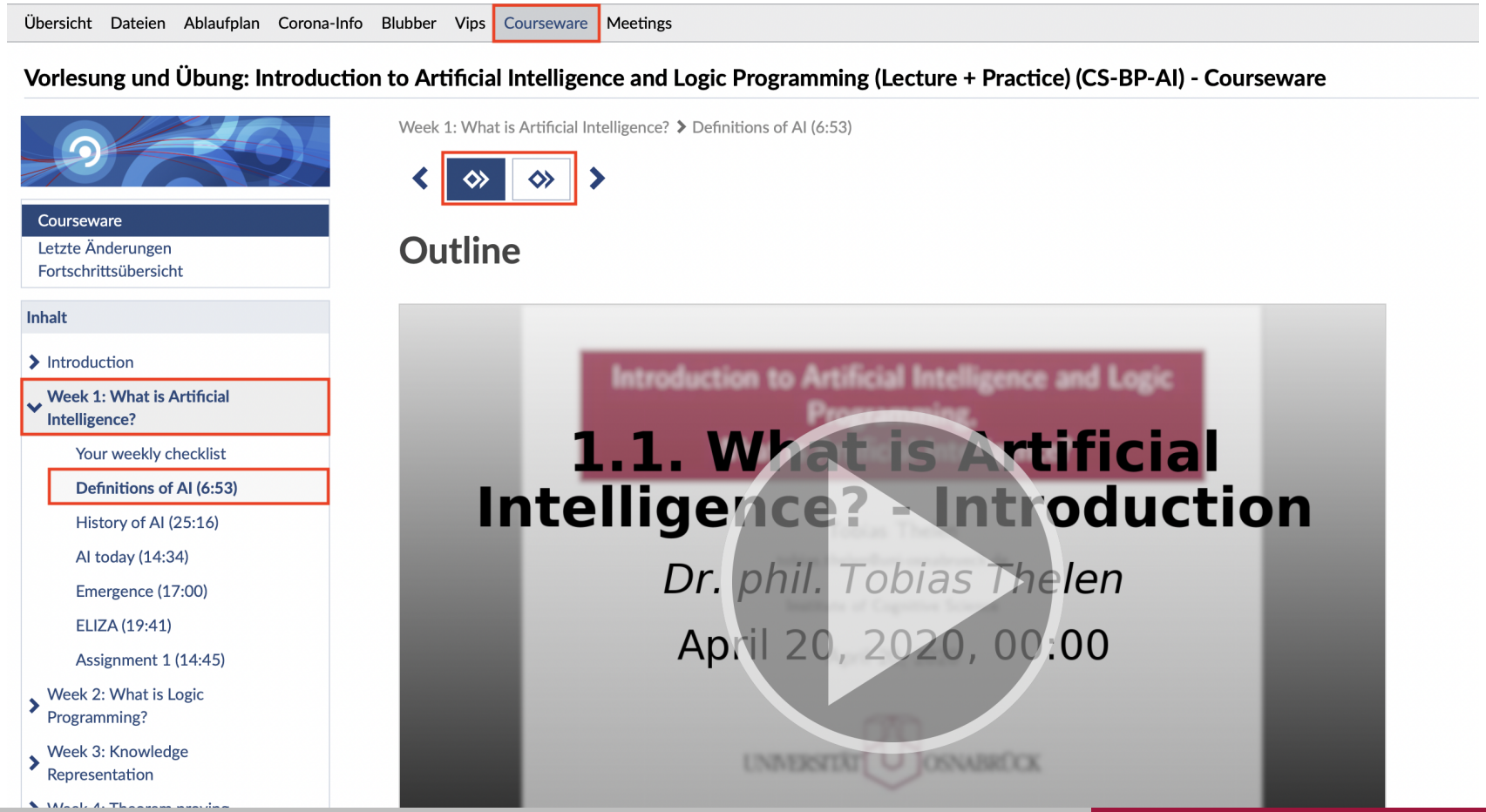
Homework and Grading

Homework Regulations

1. One homework sheet per week (12 in total)
2. Sheet submission in groups of 2-3 via Vips on StudIP
3. You need 50% of the points to pass a sheet
4. You have to pass **10 out of 12** homeworks to pass the course
5. Grading will be optional, likely as some kind of final project / assignment. Information on this will follow.

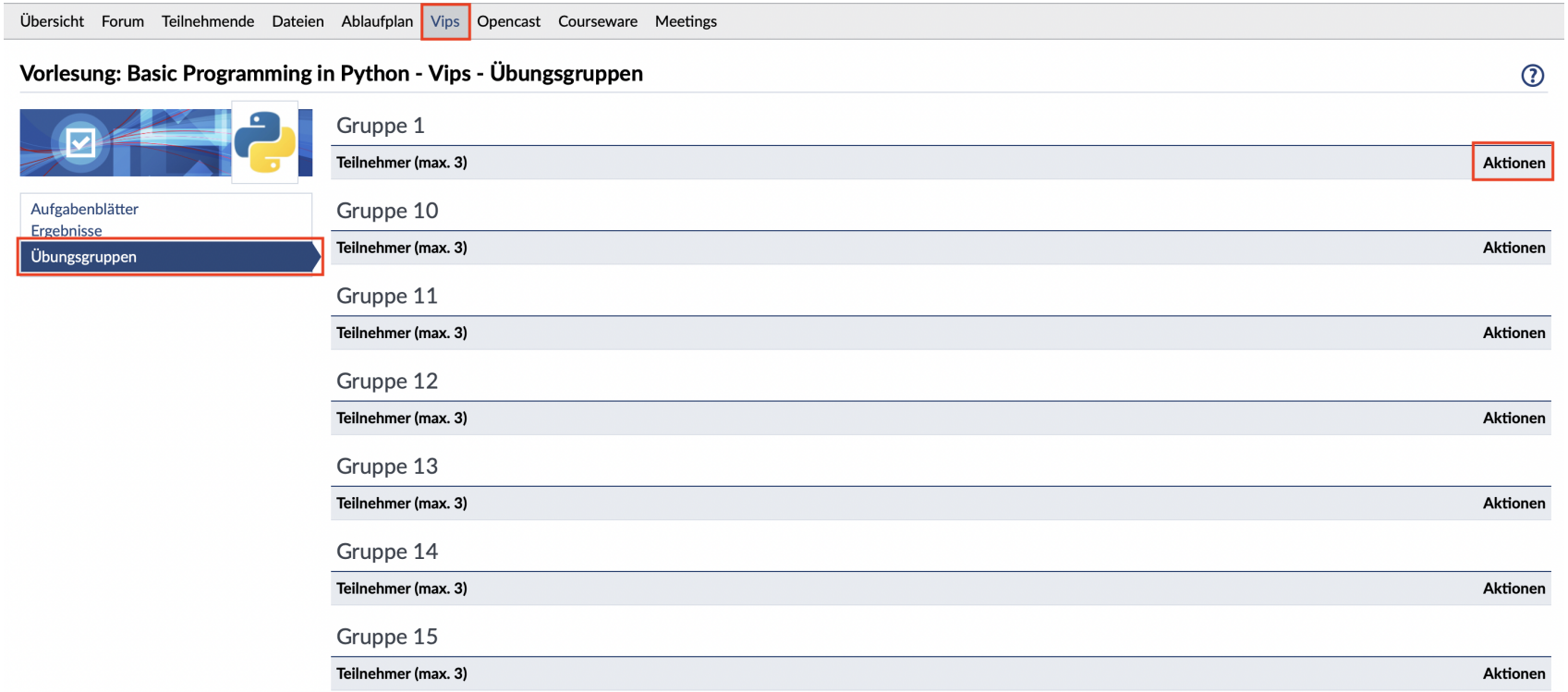
Courseware

This is where you find Lectures



Homework Groups

1. There are 42 homework groups available
2. In each group there should be 2-3 students
3. You can enter a group between 12.4. 18:00 and Sunday



Some Examples

Some Examples

- **Visualizing Data**
  here could be a nice pyplot example

Some Examples

- **Visualizing Data**
  here could be a nice pyplot example
- **Machine Learning**
  Artificial Neural Networks, Reinforcement learning, Computer Vision, etc...

Some Examples

- **Visualizing Data**
  here could be a nice pyplot example
- **Machine Learning**
  Artificial Neural Networks, Reinforcement learning, Computer Vision, etc...
- **And a few more selected Examples**
  Statistics, automatisation, surveys, ...

Some Examples

- **Visualizing Data**
  here could be a nice pyplot example
- **Machine Learning**
  Artificial Neural Networks, Reinforcement learning, Computer Vision, etc...
- **And a few more selected Examples**
  Statistics, automatisation, surveys, ...

Some sort of conclusion or whatever

What is an Algorithm?

Definition

[...] an *Algorithm* is an unambiguous specification of how to solve a class of problems.[1]

[1] **wiki:algorithm**.

What is an Algorithm?

Definition

[...] an *Algorithm* is an unambiguous **specification** of how to solve a class of problems.[1]

- **specification**
  meaning a description / instructions

[1] **wiki:algorithm**.

What is an Algorithm?

Definition

[...] an *Algorithm* is an **unambiguous** specification of how to solve a class of problems.[1]

- **specification**
  meaning a description / instructions
- **unambiguous**
  at each point you know exactly what the next step is

[1] **wiki:algorithm**.

What is an Algorithm?

Definition

[...] an *Algorithm* is an unambiguous specification of how to solve a **class of problems**.[1]

- **specification**
  meaning a description / instructions
- **unambiguous**
  at each point you know exactly what the next step is
- **problem-specific**
  an algorithm for sheering sheep won't help milking cows

[1] **wiki:algorithm**.

What is an Algorithm?

Definition

[...] an *Algorithm* is an unambiguous specification of how to solve a class of problems.[1]

- **specification**
  meaning a description / instructions
- **unambiguous**
  at each point you know exactly what the next step is
- **problem-specific**
  an algorithm for sheering sheep won't help milking cows

Example: A cooking recipe

[1] **wiki:algorithm**.

Pseudocode

We need a way of writing down algorithms!

Pseudocode

We need a way of writing down algorithms!

Example: Baking a Cake

```
start: gather all ingredients
REPEAT
        add the next ingredient to the bowl
UNTIL all ingredients are used
stir dough thoroughly
put dough into oven at 200°C
wait 50 minutes
REPEAT
        bake for another minute
UNTIL cake looks good
IF cake tastes bad GOTO start
```

Pseudocode

We need a way of writing down algorithms!

Example: Baking a Cake

```
start: gather all ingredients
REPEAT
        add the next ingredient to the bowl
UNTIL all ingredients are used
stir dough thoroughly
put dough into oven at 200°C
wait 50 minutes
REPEAT
        bake for another minute
UNTIL cake looks good
IF cake tastes bad GOTO start
```

Good:

- individual steps
- structure
- fairly readable

Bad:

- not specific enough
- dough, oven, etc. not defined

Programming Languages...

...are an even more formal way of writing algorithms.

- easier to understand for computers
- strict rules regarding syntax etc.
- there are tons and Python is one of them!
- even this presentation is written in a programming language[2]

[2] **hicks**.

From High-Level to Low-Level

Actually, computers really only understand binary

Some binary code

0100110111100101101101101001...

- only a few, very basic instructions
- higher-level programming languages build on top of that
- all programs must be translated into binary code (compilation, interpretation)
- we don't need to worry about that

Soo.. what is programming?

Two aspects for solving a problem with programming:

Soo.. what is programming?

Two aspects for solving a problem with programming:

- **Designing an algorithm**

Soo.. what is programming?

Two aspects for solving a problem with programming:

- **Designing an algorithm**
- **Implementing said algorithm**

Both are equally important for a good program

Soo.. what is programming?

Two aspects for solving a problem with programming:

- **Designing an algorithm**
- **Implementing said algorithm**

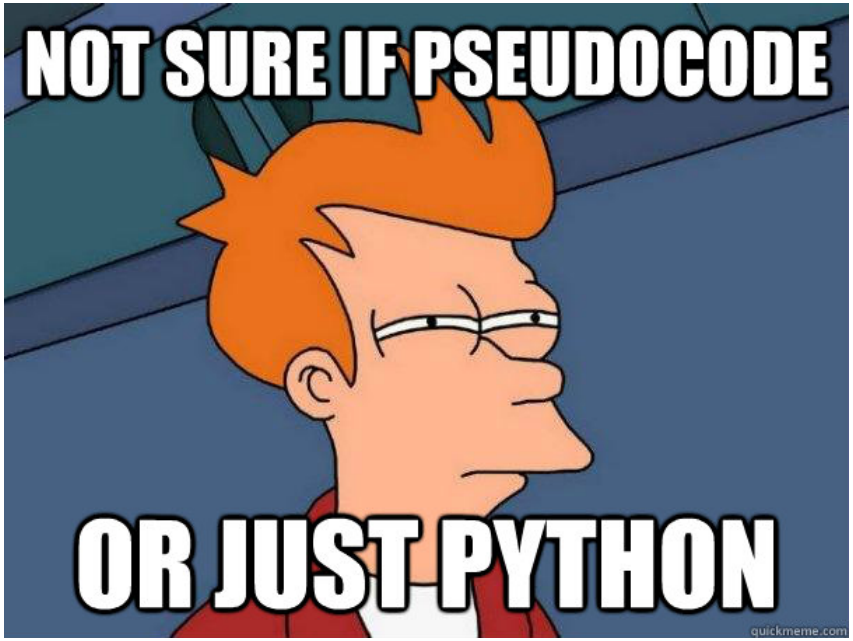
Both are equally important for a good program

We will focus more on implementation

Python

Python

A high-level language that is easy to learn, read and write.



Why Python?

Advantages

1. Widespread usage (especially in academia)
2. Open source environment
3. Steep learning curve
4. Multiplatform support (Windows, Linux, Mac)
5. Large ecosystem of libraries and packages

Why Python?

Disadvantages

1. Slow execution
2. High memory usage
3. Requires Python Interpreter

```
print("hello world!")
```

```
# live demo
```

print("hello world!")

```
>>> print("hello world!")
hello world!

>>> print(hello world!)
  File "<stdin>", line 1
    print(hello world!)
                      ^
SyntaxError: invalid syntax
```

Python Shell as a Calculator

```
>>> print(42)
42

>>> print(20 + 22)
42

>>> print("4" + "2")
42

>>> print("42" * 5)
4242424242
```

Using the Terminal

```
# live demo
```

Python Scripts

With algorithms in mind, we often want to execute many lines of code in immediate succession

```
print("I am a script!")
print("All I do is print stuff.")
print("But I can do this: " + "blub" * 10)
```

If we save this in a file `my_script.py`, we can run everything with `python my_script.py`

This Week's Homework

1. Install Python
2. First experiments with terminal
3. Use the Python turtle environment

→ For details see file 01_Introduction_Ex.pdf
→ For help come to the Walk-In Practice Session on Thursday from 12:15

References