Exercise class 0

(week 7)

Introduction to Programming and Numerical Analysis

Class 8 and 4

Rosa Haslund Meyer

Spring 2024

With inspiration from Annasofie Marckstrøm Oelsen

KØBENHAVNS UNIVERSITET

Welcome!

Tips for installation and DataCamp

Exercises

Atomic types

Containers / data structures

Conditionals and loops

More exercises

Welcome!

Short about me

- Rosa Haslund Meyer
- 6th semester bachelor student in Computer science and economics at the Department of Computer Science at the Faculty of Science
- Haven't participated in this specific course but I've taken plenty of programming courses through my bachelor degree
- My first time as a TA!
- Very open to any kind of feedback about what works and what doesn't!
- Contac information: jdc334@econ.ku.dk or through Absalon

The exercise classes

As you know the lectures will mainly consist of video-lectures, so the exercise classes are the main platform and opportunity for you to ask questions in person.

You need to put in a lot of work when it comes to learning programming! It's important that you spend a lot of time on this and get your hands dirty – it's the best way to learn how to program and to locate and fix possible future errors!

General plan for exercise classes (changes may occur):

- I give a recap of important concepts from the given week's lecture and tips for the problem set: 15.15-15.30
- Individual/group work on problem set: 15.30-16.45 (including a break)
- Recap and questions 16.45-17.00

Course site – Absalon and GitHub?

The pages you need to know:

- Course website
- Lectures GitHub repository
- Lecture videos
- Exercises GitHub repository

I will put material relevant to the exercise classes in class 4 and 8's shared GitHub repository here.

Please contact me if you can't make it work!

For your information, the use of Absalon during this course is very limited.

Tips for installation

You'll need to install:

- **Anaconda**: a Python distribution, which helps you download Python and packages
- **Git**: a version control system that keeps track of changes to files and allows you to cooperate on projects online
- **Visual Studio Code/VSCode**: an IDE / text editor where you can write and run code files and Jupyter Notebooks

Tips for installation

Follow the installation guide on the course website carefully and in the right order.

Make sure you're installing Anaconda on the **right path** with no spaces or special characters.

If something fails, the best approach is often to **restart** the program – or uninstall and start from scratch.

It may take some time to get through the installations – you can work on DataCamp or go through lecture slides/problem set 0 while waiting.

Tips for DataCamp

You must have completed the following 4 courses from DataCamp by **February 25th**:

- Introduction to Data Science in Python
- Intermediate Python
- Python Data Science Toolbox (Part I)
- Python Data Science Toolbox (Part II)

The first two exercises classes are dedicated to working on DataCamp – but expect to do a lot of work on it at home as I want to make sure you understand the basic concepts and get all the installation done.

Don't rush through the exercises – focus on understanding the code to make a good foundation. Understanding the basics is paramount for solving the later problem sets and assignments.

If you have trouble understanding any concepts, please let me know and I'll be happy to help.

If you get an error message or forgot some syntax, try to Google it before asking. When it comes to programming, Google (and StackOverflow) can solve a lot of minor and syntax errors!

Time for exercises

Your tasks:

- Follow the installation guide
- DataCamp
- Problem set 0

If this is your very first time programming or working Python and your feeling overwhelmed, I can recommend to look at some of the exercises in DataCamp introducing Python such as “Introduction to Python”, which gives a gentle introduction to data types and variables.

At **16:15**, I’ll do a recap on **atomic types**, **containers/data structures** and **loops**.

What is a variable?

Two definitions of a variable:

- Variables are “containers” for storing data values.
- Variables in Python are reserved memory locations to store different values.

In other words: a variable **points** to some **data** stored some specific place in the **memory**.

Atomic types in Python

Four **atomic types**:

- Booleans
  - `True, False`
- Integers
  - `1, 0, 10, -5, 3892, ...`
- Floating points
  - `0.5, -0.3782, 3.14159, ...`
- Strings
  - `‘abcd’, ‘Hello World’, ‘2024’, ...`

Atomic types doesn’t keep their reference (to memory) when changed. Think of it as if they are put in a new “folder” instead.

(“Introduction to Data Science in Python” touches the concept of atomic types)

Containers / data structures

Containers can be made of several atomic type variables:

- Lists, `[]`
- Dictionaries, `{}`
- Tuples, `()`
- Pandas DataFrame, `pd.DataFrame()` DataFrame
- NumPy arrays, `np.array()`
- …

Containers can be changed and keep their reference, which can often lead to bugs!

(Tuples, as an exception, are immutable in python, meaning their elements cannot be modified, added or removed after the creation of the tuple)

("Intermediate Python" introduces dictionaries and the Pandas DataFrame)

Conditions

Python supports the usual logical conditions (Boolean operators) know from mathematics:

- Equals: `a==b`
- Not equals: `a!=b`
- Less than: `a<b`
- Less than or equal to: `a<=b`
- Greater than: `a>b`
- Greater than or equal to: `a>=b`

These conditions can be used in several ways some of which is `if-statements` and loops.

Conditions/`if-statements` consists of the keywords `if`, `elif` or `else` followed by an expression which evaluates to a **boolean type**.

If-statements

Conditions/`if-statements` consists of the keywords `if, elif` or `else` followed by an expression which evaluates to a **boolean type**:

```
if x>0:
   print('x is positive')
if x<0:
   print('x is negative')
if x==0:
   print('x is exactly 0')
```

Boolean expressions like this can also include boolean operators as the ones mentioned on the previous slide.

Loops

Python has two loop commands:

- `for-loop`: for when you want a **specific number of iterations**. Eg. iterate over elements of a list, the number of variables in a data set, or number of time periods in a dynamic model, etc.

  For when you know the number of iterations ahead of time.
- `while-loop`: for when you want to repeat a process u**ntil some condition is met**. Eg. Evaluating a function until it converges, while the input of some user is not between 1-5, etc.

  For when you don't know the number of iterations ahead of time.

Computers are generally good at performing the same tasks over and over, which is where loops come in handy! Loops also makes code make more manageable, organized and often more readable.

(The DataCamp course "Intermediate Python" explains conditionals and loops. The concept of list comprehension is covered in "Python Data Science Toolbox (Part II)".)

Next time...

Video lectures:

- Functions
- Floating point numbers
- Classes
- Numpy basics

Exercises:

- DataCamp
- (Exercise 0)

Questions and/or comments?