Programming Lab I
Handout 0

Dr. Martin Vogt
martin.vogt@bit.uni-bonn.de

April 6, 2020

Semester start: April 20, 2020

Online folder: `https://uni-bonn.sciebo.de/s/wukB3cTMMeCDYG3`

General remarks

- For the lab course you will be given assignments once per week or once per fortnight. For the assignments deadlines will be given by which you have to hand in your solutions individually by e-mail.
- The solutions will be discussed with you individually or in small groups (via Zoom/Skype).
- For help and discussions we will be available online during specific time slots.
- Additionally, given enough interest, online classes might be offered from time to time where sample solutions are discussed.
- As this is the first time, the course will be held in this form, there might still be some changes. In any case, more details on the course structure will follow.
- During the course we welcome feedback to see which things work and which don't work so well.

The lab course uses Python 3. Probably, the most convenient way to get it is via the Anaconda distribution `https://www.anaconda.com/distribution/`. Most programming tasks can be solved using Jupyter notebooks,a text editor (preferably with syntax highlighting like `notepad++` for Windows) to edit standalone scripts, and a command line.

Optionally, you can also use an integrated development environment (IDE) like PyCharm available from `https://www.jetbrains.com/pycharm/`; however, it might be more inconvenient than useful for smaller programming tasks unless you are already familiar with it.

Python prerequisites

The course assumes some familiarity with Python. In order to prepare for the course the introductory Python book *Think Python* can be found in the online materials. The book covers the basic programming elements and data structures of Python, which you should be familiar with.

> **As course preparation, use the following two weeks until the official semester start to study the book and solve the suggested exercises.**

A guideline highlighting important chapters and exercises is given below. It is intended as a refresher and as a guide to deepen your understanding of programming in general and Python in particular.

Especially, if you have been struggling with programming and understanding its concepts, *take your time* and don't rush through the initial chapters. Most exercises are very short and designed to understand basic concepts. I have denoted some exercises with a (*), which you might find harder initially.

Programming elements: Variables, expressions, statements, conditionals, recursion, iteration

- *Chapter 1* is a very light introduction to the theoretical principles of programming. It suggests running the Python interpreter from the command line. Given the widespread use of Jupyter notebooks and graphical user interfaces, you might never have used Python this way. It is strongly suggest you try it at least once. For Anaconda, you can, for instance, use the *Anaconda prompt* (available from the start menu in Windows) and enter `python` to start the Python interpreter.
- *Chapter 2,3.* Variables, expressions, statements and functions are elementary building blocks of programming languages. Some students have a hard time understanding the basic logic of programming languages. A thorough understanding of these chapters is vital.

  Suggested exercises:

  - **Exercise 3.1**
  - **Exercise 3.2**
  - **Exercise 3.3**
- *Chapter 4* A very useful chapter for learning about program flow, designing functions, and solving simple problems. It uses so-called turtle graphics where you can see the effect of your code reflected in the movement and drawings of the turtle.

  Suggested exercises:

  - **Exercise 4.2** (*)
  - **Exercise 4.3** (*)
- *Chapter 5.* Conditionals are an essential part of programming languages and are required for writing more complex programs. Recursion is often not required in programming (and can be avoided most of the time); however, it provides an elegant concept for a lot of programming tasks.

  Suggested exercises:

  - **Exercise 5.3**
  - **Exercise 5.4**
  - **Exercise 5.5**
  - **Exercise 5.6** (*)
- *Chapter 6* should deepen your understanding of functions and recursion.

  Suggested exercises:

  - **Exercise 6.3**
  - **Exercise 6.4**
  - **Exercise 6.5**
- *Chapter 7.* Iteration using `while`-loops is the last basic and essential structural element of Python that influences program flow.

  Suggested exercises:

  - **Exercise 7.1**
  - **Exercise 7.3**

Data structures: Strings, lists, dictionaries, tuples

- *Chapter 8.* Strings are an essential data structure. Manipulation of string data is very common in life science informatics problems dealing with sequence data and experimental data given in text form.

  Suggested exercises:

  - **Exercise 8.4**
  - **Exercise 8.5**

- *Chapter 9.* This chapter contains problems revolving around a dictionary of English words. Exercises 9.1-9.6 are small problems dealing with how to "translate" common problems into programming logic. The exercises at the end (9.7–9.9) are somewhat more complex. Try those, if you like puzzle solving.

  Suggested exercises:

  - **Exercise 9.1**
  - **Exercise 9.2**
  - **Exercise 9.3**
  - **Exercise 9.4**
  - **Exercise 9.5**
  - **Exercise 9.6**

- *Chapter 10.* Lists are the basic collection data structure of Python and understanding of these is essential.

  Suggested exercises:

  - **Exercise 10.1**
  - **Exercise 10.2**
  - What is the difference between **Exercise 10.3** and **Exercise 10.4**
  - **Exercise 10.7**
  - **Exercise 10.11** (*)

- *Chapter 11.* Dictionaries are a mapping data structure mapping *keys* to *values*. It is one of the most versatile and ubiquitously useful data structures. A related data structure is the **set**, which only holds keys but no values. It has not been part of the early versions of the Python programming language and is discussed later in this book. See *Section 19.5*.

  Suggested exercises:

  - **Exercise 11.1** (Here you could also use the set data structure instead.)
  - **Exercise 11.2**
  - **Exercise 11.4**
  - **Exercise 11.5** (*)

- *Chapter 12.* Tuples are the final basic data structure of Python. They are (usually short) fixed length collections of elements. Tuples are immutable in contrast to lists, which are mutable. Take some time to understand the concept of mutablility. If you like puzzles, you can try exercises 12.2, 12.3.

  Suggested exercises:

  - **Exercise 12.1** (For instance, use the dictionary file `words.txt` and determine the letter frequencies.)

- Chapter 14. Dealing with (text) files is one of the most common tasks in bioinformatics. Thus knowing how to read and write (text) files is very important.

  Suggested exercises:

  - **Exercise 14.1**

- *Chapter 19* contains syntactical elements of Python, which are commonly used in Python programming, and are designed to make programming more easy and expressive: Conditional expressions, list comprehensions, generator expressions, and sets.

  In addition, generally useful functions and classes like `any`, `all`, `Counter`s, `defaultdict`, and `namedtuple`s are discussed. The section is about gathering args and keyword args in function definitions. These allows one to write functions that have arbitrary parameters and keyword parameters as you might already have encountered in libraries like `pandas`.

  You will want to be familiar with these concepts once you gain proficiency in Python.

  Suggested exercises:

  - **Exercise 19.1**

Final notes

You can skip *Chapter 13*. However, if you are already proficient in Python, it is quite interesting to perform the Markov analysis of Exercise 18.8 and the application of Zipf's law in Exercise 13.9.

Chapters 15–18 deal with object-oriented programming. Python is not a purely object-oriented language and you can be a proficient Python programmer without programming your own classes. (However, classes are pervasive and you use them all the time as part of the Python standard library or popular libraries like pandas, NumPy, or scikit-learn) These chapters are an introduction to object-oriented programming in Python. Study them if you are interested.