Homework 1:
PyCharm, Python Basics, Testing

Robert Litschko*
Symbolische Programmiersprache

Due: Thursday, October 27, 2022, 16:00

In this exercise you will:

- Install and setup Python[1] and PyCharm[2].
- Review and implement some basic Python functionalities.
- Get some hands-on experience with:
  - The python `doctest` and `unittest` frameworks for testing.
  - Using basic Git features in PyCharm.
  - Run and debug applications in PyCharm.

If you have any questions, please ask us on Slack[3] by posting your question in the weekly channels, e.g. use `#week1` to post questions related to today's exercise.

Exercise 1: Setting up the Git project

In order to actively participate in the exercise you need to be registered to the moodle course[4] where we will release weekly homeworks. Doing the homeworks will allow you to collect bonus points for the exam, refer to the moodle course for details. Please note that today's exercise is only to get you setup and won't be graded. In order to be able to submit your homework starting from next week, make sure to do the following steps (ask the tutors if any of the steps is unclear to you):

1. Make sure you have a Gitlab account for gitlab.cip.ifi.lmu.de

*Credit: Exercises are based on previous iterations from Katerina Kalouli
[1]We will use Miniconda to manage Python environments: `https://docs.conda.io/en/latest/miniconda.html`
[2]You are free to use any other IDE, if you want to follow along during the exercise, install PyCharm following `https://www.jetbrains.com/pycharm/download/`
[3]`https://join.slack.com/t/symprog2223/shared_invite/zt-1hg1upofr-XwjdMsaMEHlYs2zdjoB89w`
[4]`https://moodle.lmu.de/course/view.php?id=24420`

2. By now you should have formed teams of 2 or 3 students (groups of 4 students are not allowed, we will reallocate randomly participants from those teams into a different group) by entering one of the predefined groups in moodle. If you would like to participate and collect bonus points and are not yet part of any group, please reach out to us as soon as possible.

3. Until next week, we will create a project in Gitlab for you. Download your homework on moodle and submit your solution by pushing to this project. For following along today, you can simply create a test project.

4. **Please do not create separate files or folders to submit your solution. Instead, change the files we provided.**

5. **Important: do NOT change the tests themselves, implement the missing functionality instead. Changing the tests will result in your exercise sheet scored with 0 points.**

Exercise 2: Python Basics

Exercise 2.1: Doctests

Use the doctest module to test your implementation of the functions in the module `hw01_basics.basics`.

Run your tests with (this assumes that you are in the src/ directory of your repository):

```
python3 -m doctest -v hw01_basics/basics.py
```

Exercise 3: Python Basics

For each function, replace the pass statement, so that the function works properly as described below and indicated by the doctests. Make sure the doctests pass the test afterward.

1. `hello_semester()`

   Print the string '*Welcome to "Symbolische Programmierung" WS 22/23*'

2. `modulo(x,y):`

   Given two variables x and y. Calculate x mod y.

3. `odd(x):`

   Determine whether x is odd or not. Use the function `modulo(x,y)` in your calculation.

4. `happy_birthday(name, age):`

   Given a name and an age. Print the sentence "Happy >age<th birthday, >name<!".

5. `word_multiplier(word,n):`

   Given a word and a natural number. Return the word repeated itself n times in a row.

6. `reverse(w):`

   Return the reverse of a word, as you would read it backwards.

7. `every_nth(word, n):`

   Return only every *nth* letter of a word.

8. `second_element(list):`

   Return the second element from a list.

9. `concatenate_lists(list_a, list_b):`

   Return the concatenation of both lists.

10. `swap_half(list):`

    Swaps the first half of a list with the second half of the list. If the length of the list is odd, then the first half has one less element than the second half. You can make use of the *concatenate_ lists(list_ a, list_ b)* function.

11. `replace_elements(list_a, replacement_indices, new_value):`

    Replace all elements in `list_a` at the positions in `replacement_indices` with the new value.

12. `long_strings(string_list, max_length):`

    Return a list of booleans: True for each string in `string_list` if the string is greater than `max_length` and False otherwise.

13. `print_squares(list):`

    Prints the square values of each element in the list. You can use a for-loop for this problem.

14. `count_to_k(k):`

    Print out the numbers counting from 0 to k, excluding k. If k is negative, count 'down' from 0, excluding 0. You can use a while-loop for this problem or the range(...) function.

15. `no_numbers(w):`

    Return True or False whether a string w contains no number symbols (0–9).

16. `contains_substring(string,substring):`

    Return True or False whether a string contains a substring.

Exercise 4: Self-study: Familiarize yourself with PyCharm

PyCharm is a powerful integrated development environment (IDE) that we will use in this course. Please familiarize yourself with the IDE by completing (1) the `Onboarding tour`, (2) `Editor basics`, (3) `Run and debug` and (4) `Git` lessons. You can access the lessons by navigating **View** > **Tools Windows** > **Learn** as shown in Figure 1. Let us know if you run into any issues.

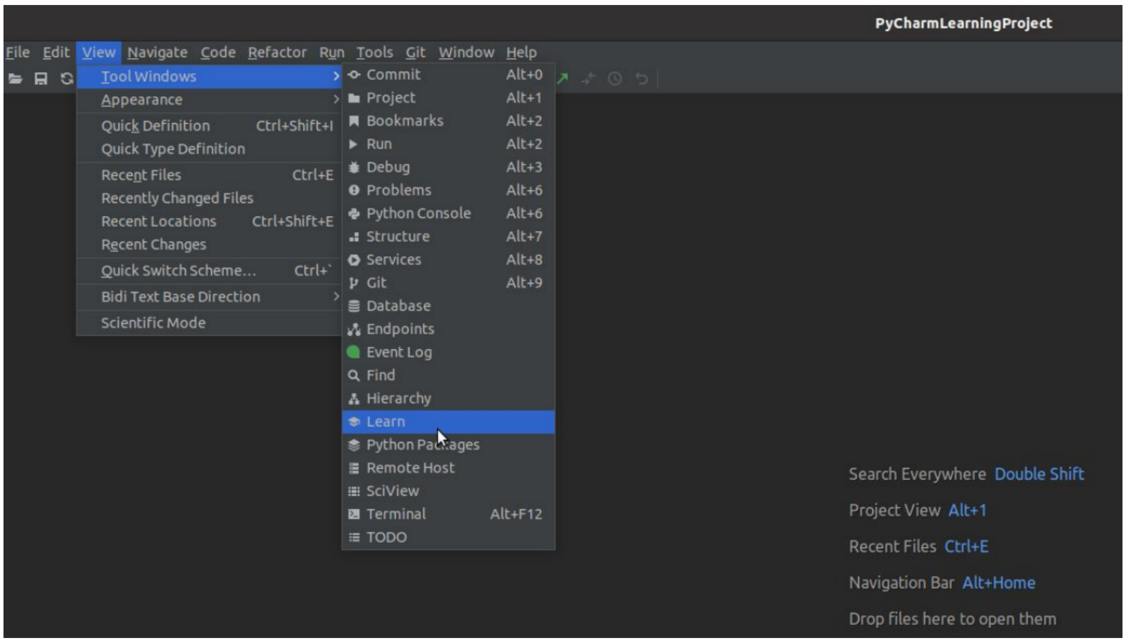

Figure 1: Access PyCharm's built-in tutorial: **View** > **Tools Windows** > **Learn**