

# CSI2100-01 Lab 3

Bernd Burgstaller  
Yonsei University

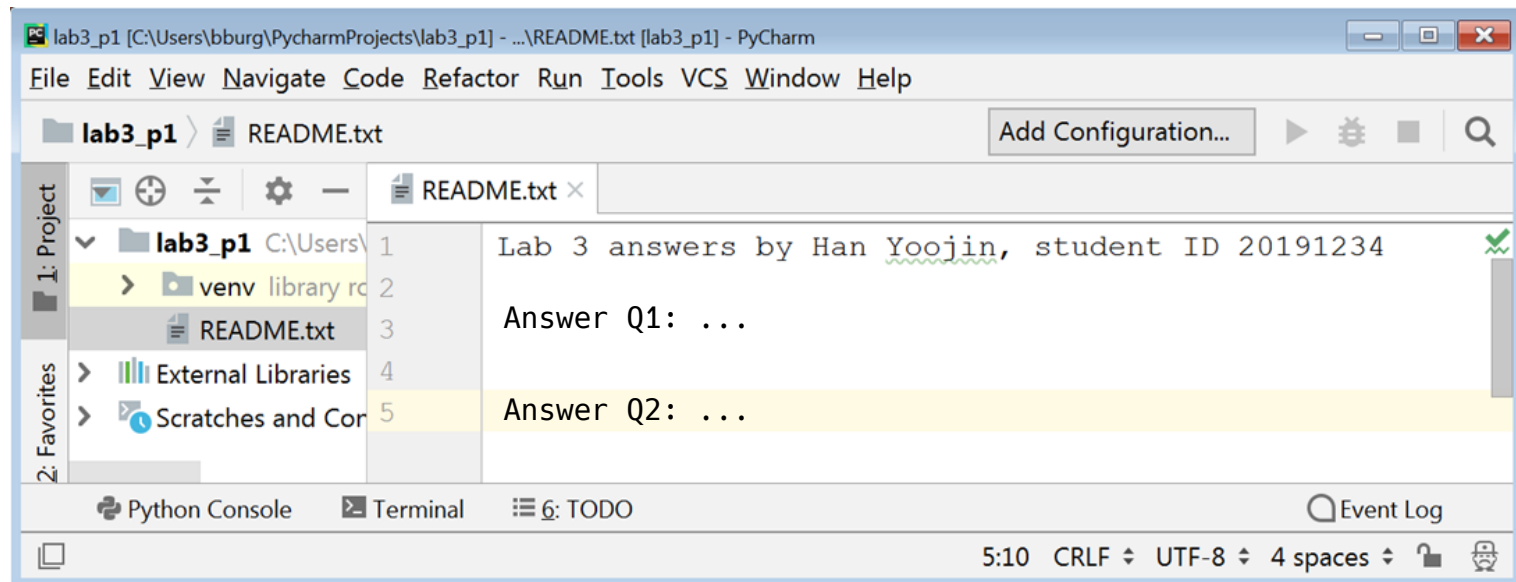


# Overview

- Questions
- Programming Problems
- Deliverables, due-date and submission
- **Notes:**
  - 1) You do not have to consider faulty user input in any of the programming problems
  - 2) Please consider to use the OnlinePythonTutor when debugging your code.
  - 3) Because of our coronavirus precautions, you are provided with an excerpt of the relevant sections of the textbook together with this lab assignment.

# Questions

- You are kindly asked to submit the answers to the questions on the following page in a **single file** named **README.txt**
- Use the editor of your choice to create and edit file README.txt (same as you would edit source code).
- For example, in PyCharm:
  - File → New → File → enter the filename ``README.txt"
  - After composing your answers, export the file README.txt in the same way as you export your source-code files.
    - This is explained in our PyCharm installation instructions.
  - Example:



## Questions (cont.)

- See the slide on "Deliverables" on how to submit your README.txt file.
- **Question 1:** Exercise 13 from p. 75 of the textbook

(Only one question for Lab 3. 😊)

# Programming Problems

**Problem 1:** Exercise D3 from p. 77 of the textbook

- **Note 1:** please do not report partial images. E.g., if the exact number of files that fit on a USB drive would be 5.5 images, you should report 5 images instead. (We assume that there's no point for the user to store 1/2 of an image on a USB drive.)

**Hint:** To achieve this, you must find a way to compute the floor function  $\lfloor x \rfloor$  of a given float value  $x$  (the Python 3 `math` module contains this function). You can assume that the input type is int. For the TIFF format, we assume a 1:1 compression ratio (no compression). Output a right-justified int of fieldwidth 5 in place of ``xxxxx''.

- **Example 1:**

```
Enter USB size (GB): 1
10416 images in GIF format can be stored
17361 images in JPEG format can be stored
 5555 images in PNG format can be stored
 347 images in TIFF format can be stored
```

- **Note 2:** for larger USB drives, a fieldwidth of 5 may be insufficient to accommodate the number of images. In such a case it is permissible to exceed the 5-digit fieldwidth (the `format()` function has the expected behavior).

```
Enter USB size (GB): 64
666666 images in GIF format can be stored
1111111 images in JPEG format can be stored
355555 images in PNG format can be stored
22222 images in TIFF format can be stored
```

# Programming Problems (cont.)

- **Note 3:** the textbook's table that describes the image formats gives more information than what is required to solve this problem.  
As we discussed in the first lecture (Introduction), we need to find the appropriate **representation** for a problem and **leave out un-necessary details (= abstraction)**.

# Programming Problems (cont.)

**Problem 2:** European countries use a 13-digit code, known as the European Article Number (EAN) to identify consumer products (see the example to the right). The EAN is used instead of the 12-digit Universal Product Code (UPC) found in North America. Each EAN ends with a check digit. The purpose of the check digit is to help identify an error in the previous digits. If the check digit does not match the other EAN digits, something is wrong (a scratched digit, a miss-print, ...).



The **procedure to compute the check digit** is as follows:

1. Add the second, fourth, sixth, eighth, tenth, and twelfth digits.
2. Add the first, third, fifth, seventh, ninth, and eleventh digits.
3. Multiply the first sum by 3 and add it to the second sum.
4. Subtract 1 from the total.
5. Compute the remainder when the adjusted total is divided by 10.
6. Subtract the remainder from 9.

*Continued on the next page...*

## Programming Problems (cont.)

For example, consider a product with an EAN of 8691484260008. The first sum is  $6 + 1 + 8 + 2 + 0 + 0 = 17$ , and the second sum is  $8 + 9 + 4 + 4 + 6 + 0 = 31$ . Multiplying the first sum by 3 and adding the second sum yields 82. Subtracting 1 gives 81. The remainder upon dividing by 10 is 1. When the remainder is subtracted from 9, the result is 8 (the last digit of the original EAN code).

Your job is to write a program that asks the user to enter the first 12 digits of an EAN and computes the check digit.

### Example:

```
Enter the first 12 digits of an EAN: 869148426000
Check digit: 8
```

### Hint:

- Convert the user input to an int, and extract the 12 EAN digits through a sequence of **truncated division** and **remainder** operations.
  - For example, if  $n$  is an integer, then  $n \% 10$  is the last digit in  $n$ , and  $n // 10$  is  $n$  with the **last digit removed**.
  - This is the same algorithm that we applied to convert a base-10 number to base-2 in Lecture 1 "Introduction" on page 62.
  - Store the individual digits in variables, for example:  $\text{digit\_1} = \dots$ ,  $\text{digit\_2} = \dots$ ,  $\text{digit\_3} = \dots$
- Once you have the 12 individual digits available, it is straight-forward to proceed with the 6-step procedure on the previous page.



## Programming Problems (cont.)

**Problem 3:** Write a program that outputs the **source-code** of the following Fahrenheit to Celsius conversion program. That means, when your program executes, it should **print** the following **output**:

```
Line 1: # Fahrenheit to Celsius conversion program
Line 2:
Line 3: fahrenheit = float(input('Enter degrees Fahrenheit: '))
Line 4: celsius = (fahrenheit - 32) * 5 / 9
Line 5: print(fahrenheit, 'degrees Fahrenheit equals',
Line 6:       format(celsius, '.1f'), 'degrees Celsius')
```

- This is a very simple idea: some programs print "Hello, world" as their output. You are asked to write a program that **prints the source-code of the above program** as its output. ☺
- As an example, Line 1 and Line 2 of the above program can be output by the following `print()` statement:

```
print('# Fahrenheit to Celsius conversion program\n');
```

- Please note that the program's print statement spans two lines (Line 5 and Line 6). If you prefer, you can combine those two lines in one long line.
  - Please do not print the red line-numbers, which have been added to facilitate understanding.
  - Your printed program must be a correct version of the Fahrenheit to Celsius conversion program. Please refer to the next page on how to test your program.


# Programming Problems (cont.)

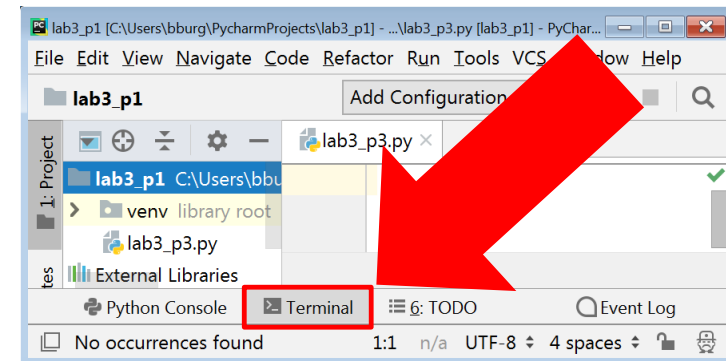
**Hint 1:** To test your program, you can proceed as follows:

- (1) Run your program and save the generated output to a file named ``generated.py``.
- (2) Run ``generated.py`` and examine its output. We assume that your initial Python program is in file ``lab3\_p3.py``. The following steps must be run in the **PyCharm terminal** (see the screenshot to the right).

- (1)

% python lab3\_p3.py > generated.py
- (2)

% python generated.py
- 



**Notice** the output redirection '>' in Step (1) above. It causes the output of program `lab3_p3.py` to be written to the file `generated.py`. The output of `lab3_p3.py` is the Python program (source-code) that converts Fahrenheit to Celsius temperatures. (You may open file ``generated.py`` with PyCharm and check its contents!)

# Programming Problems (cont.)

**Hint 2:** please do not confuse what you shall submit: we require you to submit `lab3_p3.py` and **not** the printed output (`generated.py`).

**Hint 3:** if your command interpreter does not recognize the `python` or `python3` command (*`python` is not recognized as an internal or external command*), then it's likely that Python was not correctly installed.

From our PyCharm installation instructions, re-run **Step 1** and **make sure that "Add Python 3 to PATH" is selected.**

# Programming Problems (cont.)

**Problem 4 (Lotto):** Write a program that generates 6 random natural numbers in the integer interval `[1, 45]`, and prints out the result in a single line as follows:

```
Lotto numbers of the week: 17 1 9 10 45 32
```

- Your program does not have to sort the numbers; any order of the numbers is ok. Duplicates are ok, too! 😊
- The program does not receive any input.
- Please put a blank ' ' character between any two numbers, and end the output with a newline `"\n"` character (if you use the `print` function, the newline is already provided).

Hint: the Python `rand` module provides several functions to generate random numbers. Please read the Python 3 online documentation, especially on `random.randint(a, b)`.

# Programming Problems (cont.)

## **Problem 5:** Chapter 2, p.77, Problem M6

- 1) Please carefully read the explanations in Section 2.5 (p. 67) of the textbook to understand the solution of the ``age in seconds" program.
- 2) Create a Python program for the ``age in seconds" program as described in the textbook.
- 3) *Modify your program as asked. Please do **not** print the program greeting in your solution. You do not have to consider faulty user input.*

*Example:*

```
Person 1: Enter month born (1-12): 10
Person 1: Enter day born (1-31): 10
Person 1: Enter year born (4-digit): 2000
Person 2: Enter month born (1-12): 11
Person 2: Enter day born (1-31): 11
Person 2: Enter year born (4-digit): 2000
Age difference in seconds: 2716200
```

*continued on the next page...*

# Programming Problems (cont.)

Please note: the age difference **must be a number  $\geq 0$** . Negative age differences are not allowed. We discussed a Python function in Lecture 2 which allows you to do this.

Please note that you must come up with your own testplan, because the numbers in the textbook are from 2013 (the publishing year), so they won't work in 2020 😊. You are not required to submit your testplan.)

# Marking Criteria and Plagiarism

- Marking Criteria
  - Score is only given to programs that compile and produce the correct output with Python 3.
  - Points are deducted for programs that are named wrongly. See the list of deliverables for the required file names.
  - Points are deducted for programs that produce warnings.
  - Points deductions on programming style: please provide comments in your code.
  - Please pay particular attention to the requested output format of your programs. Deviating from the requested output format results in points deductions.
- Plagiarism (Cheating)
  - This is an individual assignment. All submissions are checked for plagiarism.
  - Once detected, measures will be taken for **all** students involved in the plagiarism incident (including the ``source" of the plagiarized code).

## Deliverables

- Please prepare the files for the programming problems and questions. The names of the files, their due-dates and the archive file-name are given in the below table.
- Please refer to the Lab 1 specification on how to create zip archives.

Problem	File name	Due	Archive name
1	lab3_p1.py	Wednesday April 8 23:00	lab3_<student id>.zip
2	lab3_p2.py		
3	lab3_p3.py		
4	lab3_p4.py		
5	lab3_p5.py		
Questions	README.txt		

- For instructions on how to upload your zip archive on YSCEC, please see Lab 1.