Basic Programming - Seminar 1

Alexandra Ciobotaru

22 October 2021

The scope of this lab is to get familiar with the basic elements of Python programming language, like language description and its particularities, syntax, as well as learning what are variables and how to use them. You will learn how to install an IDE (Integrated Development Environment), called PyCharm, create your first project and understand what are virtual environments.

1 Prerequisites

For this seminar you will need:

- Computer with Windows/Linux/Mac;
- Memory: 4 GB of free RAM / 8 GB of total system RAM (recommended);
- Disk space: 2.5 GB and another 1 GB for caches;
- Enthusiasm.

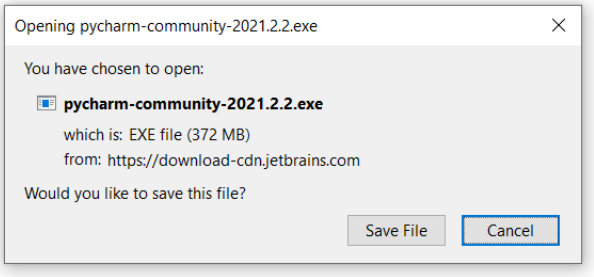
2 Install Python

If you haven't already, download version 3.9 of Python from: `https://www.python.org/downloads/`.

Install Python but make sure you check "Add Python 3.9 to PATH".

3 Install PyCharm Community IDE

Download the **Community** archive from `https://www.jetbrains.com/pycharm/download` corresponding to your operating system.



Next, follow the instructions on the screen to install PyCharm.

For Windows 10 users: `https://www.youtube.com/watch?v=XsL8JDkH-ec`
For mac users: `https://www.youtube.com/watch?v=K5cAu-Wro3M`

4 Test your Python Installation (optional)

After installation is done, open CMD and write `python` to check if this command opens Idle integrated development environment.

Type `help()` and see what happens. Type `math` in help console and see what happens. Press Ctrl+c to exit help or type `exit()`.

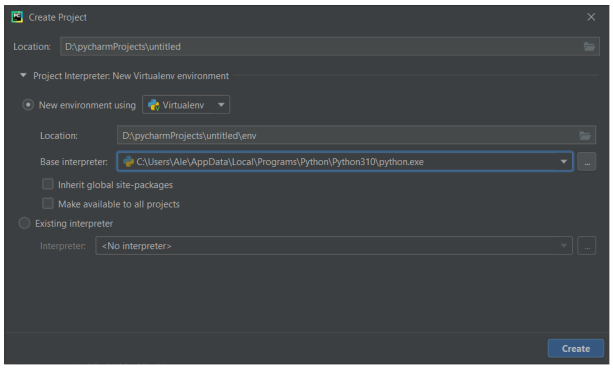
If we want to use math library we first need to import it: write `import math` in console. Next, try computing 2^3 using the pow function.

5 Create your first project

On this seminar you will create a mental health potion for a video game. Imagine you have a character who has some mental health, let's say he has MHP:45 mental health points, because he didn't sleep well and neglected the main meals of the day (he had a lot of work to do lately). Imagine this character walks into a health potion that magically increases his mental health by a random amount. But the game also has three degrees of difficulties: easy, medium and hard. Depending on the difficulty level selected, the amount of mental health that the person gets will change, and this will be less on higher difficulties.

In this project you will be using variables and numbers to create the functionality required for this mental health potion to work. You will also use Python's random module to build a random number generator.

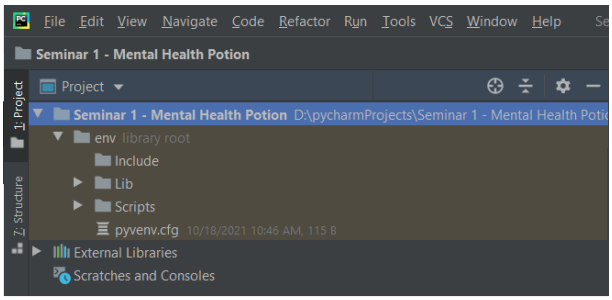
Open PyCharm Community Edition and create a new project. Instead of "untitled", type the name "Seminar 1 - Mental Health Potion". Select "New environment using Virtualenv" and select Python 3.9 as base interpreter. Press Create.



Press Settings $->$ Project: Seminar 1 - Mental Health Potion $->$ Project Interpreter and make sure the interpreter is Python 3.9.

Next, in the upper left of PyCharm window you will find your project structure. There, click *env* folder under *Seminar 1 - Health Potion* folder. Right click on *env* folder and select New $->$ Python file, and give a name to your python script, something like "health_potion". The extension *.py* will be assigned automatically. Remember all your names should be intuitive, in order to remember what you meant when you will read it later on.

Comments are written with # at the beginning of the line. It is a good practice to name your scripts suggestively, and also to leave comments along your code. This way, after reading your code years later, you will quickly remember what you meant, and also, if you give your code to someone else to read it, if your code is commented it is easier to understand.



So now, let's create the script:

1. Write a suggestive comment at the beginning of the script, explaining that you will be creating a mental health potion for a character;

2. Import the random library you will use later (imports are usually written at the top of the script):

   ```
   import random
   ```

3. Create an integer variable called "mental_health" and give it a value between 1 and 100.

   ```
   mental_health = 50
   ```

   So now you just created a representation of the player's mental health, using the variable "mental_health";

4. The potion is going to give the player a random amount of health, between 25 and 50, and for that you will use the random python module that you imported at the beginning of the script. First read a bit about what this module is all about by pressing Ctrl and clicking on "random". You can see that `random.py` script opened (don't modify it). Navigate to `randint` function at line 218 - you can see that it returns a random integer in range [a, b], including both end points. Using randint, create a variable called `health_potion` that stores a random number between 1 and 50:

   ```
   health_potion = random.randint(1,50)
   ```

   Now print your variables and see what you've got.

   ```
   print("Mental health points: ", mental_health)
   print("Health potion the player found: ",
   health_potion)
   ```

5. Compute final mental health our player has after drinking the magic potion, by creating a `final_health` variable to store the sum of its original mental health plus the mental health given by potion, and print the result:

```
final_health = mental_health + health_potion
print("Player's final mental health: ", final_health)
```

6. Let's generalize a bit our player's health potion by computing it as a random number between 1 and the maximum amount of health potion the player can get (up to 100). Just comment your health_potion declaration and write this one for a turn:

   ```
   health_potion = random.randint(1,(100-mental_health))
   ```

   Run your script and see what you got.

7. Define a variable called `difficulty` and set it to 1, meaning it's in easy mode by default. Medium will be 2 and Hard will be 3. Write a comment suggesting what yiu did, before the assignment of the variable:

   ```
   # easy = 1, medium = 2, hard = 3
   # difficulty is 1 by default
   difficulty = 1
   ```

8. Notice you have floating points in `health_potion` and `final_potion`. The reason for that is that python automatically converts an integer to float when needed. And because we applied some operations to there two variables, they became floats. Type `type()` over your variables in the print function to see what type your variables are:

   ```
   print("mental_health ", mental_health,
   type(mental_health))
   print("health_potion ", health_potion,
   type(health_potion))
   print("final_health: ", final_health,
   type(final_health))
   ```

   After your run your code and check your variables, you can delete these lines from here.

9. **Casting** is when we force a variable to become another type. So we will force our variable `health_potion` to become an integer by typing `int` around it. Modify your `health_potion` variable like this:

   ```
   health_potion = int(random.randint(1, 100 - mental_
   health)/ difficulty)
   ```

   When checking again the types of your variables you will see that `final_health` is also integer, because when we add an integer with another integer will always result an integer.

10. Let's complicate things a little more by taking the input from keyboard for variables `mental_health` and `difficulty`. Change these variables assignments with:

```
print("Insert your player's current mental health points: ")
mental_health = int(input())
print("Insert difficulty mode between 1 and 3: ")
difficulty = int(input())
```

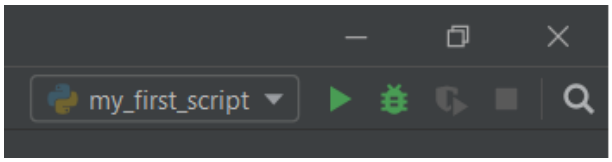
By default an input from the user keyboard is string, so that is why we had to cast the variables into int.

11. Run your code and enjoy your game!

6 PyCharm functionalities

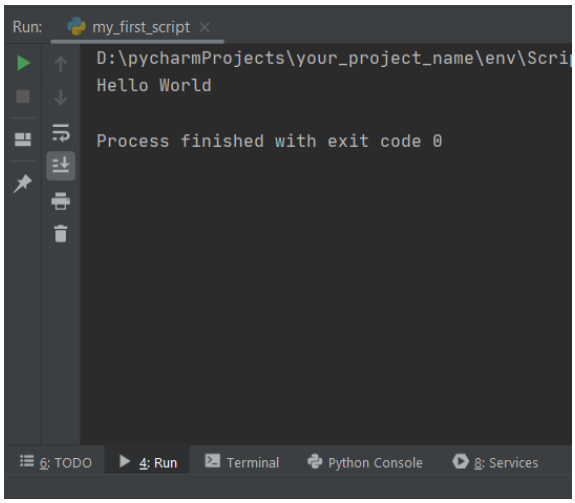
To run your script, you have three options. You can either run your script by pressing the green triangle in the upper right corner of PyCharm, or by right clicking in the code window and selecting *Run 'my_first_script'*. Alternatively, you can press Ctrl+Shift+F10 to run your script. The result of your script will be shown in the Run console.

It is a good practice to run your script often, to make sure everything is fine and you have no errors, than to write a lot of code and run it all.



In the lower left corner of PyCharm you will see three important tabs: Run, Terminal and Python console:

- Run: here you will see the results of your ran scripts;
- Terminal: here you will find your terminal, where you can install/uninstall programs, and navigate through the structure of your PC (just like in command prompt);
- Python Console: in the console you have your python idle running. The usefulness of this console is that you can try here small parts of your code, without creating and saving new scripts.



7 Explaining the virtual environment (venv)

Navigate in Terminal to env folder (by typing `cd env`). There you can see the name of your environment in brackets '(env)'. Seeing this at the beginning of the line means that the virtual environment is activated. To deactivate it, type `deactivate`. All commands you give now will have an effect on all your PC. For instance, if you install a new library here, it will be available regardless any virtual environment.

But Python programming language is known for its fast paced development, thus, there are many versions of libraries which could create conflicts between each other. This is why it is always recommended to use a virtual environment when programming in python, preferably one virtual environment per project.

To activate the virtual environment again, from env folder navigate to Scripts folder (by typing `cd Scripts`) and type `activate`.

For Mac users: to activate the virtual environment type in Terminal: `source /venv/bin/activate` and to deactivate it type `daectivate`.

8 Grading

After each seminar, you will have to fill out a form with a quiz (10 very easy questions) that will assess and also wrap what you have had learned during the seminar and the two lessons related to the seminar.

There will be 6 quizzes in total, counting up to 60% of your total seminar grade. You can receive the rest of 40% by annotating data to build a dataset for emotion detection from text. The aim of the annotation is to create a dataset of emotions for Romanian language, in order to create a machine learning model

that detects the emotion conveyed by a text. Emotions searched for are such as: happiness, sadness, fear, anger, and so on. For annotating, you will receive a csv with texts, and you will have to make sure those texts indeed represent the emotion they have been labelled with.

Presence

Presence is mandatory, but there always can exist some kind of problems. So, the minimum percentage of attendance required to be able to enter the exam is 4 presences out of 6. Quizzes missed can be filled out before the next seminar.