PROGRAMMING IN PYTHON I

Editor and Debugger

Michael Widrich
Institute for Machine Learning

Michael Widrich
Institute for Machine Learning

JYU

EDITOR AND DEBUGGER

Editor

- Comfort in programming has come a looong way
- You don't have to program in a plain text editor anymore
- Modern editors allow for:
  - Syntax highlighting
  - Auto-completion of variable names and small syntax
  - Automatic check for errors and warnings in your code
  - Automatic reformatting of your code to specific coding standards
- Many editors for Python also include a debugger

```
1    print("Hello World!")
2    a = 5
3    b = 4
4    c = a + b
5    print(c)
```

Syntax highlighting in PyCharm Editor

Debugger

- Unintended errors/behaviors in a program are referred to as bugs
- Searching for and removing these bugs is referred to as debugging
- Debuggers allow you to analyse your program while it is executed
- Modern debuggers allow for:
    - Exploring variables during runtime
    - Executing your code line by line and pausing the program at will
    - Interacting with/Modifying the code during a pause
    - Handling multiple parallel processes correctly

PYCHARM

PyCharm

■ We recommend using PyCharm
  □ Modern editor and debugger for Python (with support for LaTeX, shell scrips, . . .)
  □ Free to use even without student licence
  □ Integration of version control tools such as git (relevant for next semester)
  □ We will only touch upon a small subset of its functions

[Image: PyCharm Logo, JetBrains]

Task 0: Install the Pycharm editor

■ Install Pycharm Community Edition

(https://www.jetbrains.com/pycharm/download/)

    ☐ Installation is straight-forward

    ☐ Community edition is free and sufficient for this lecture

    ☐ Ubuntu:

       `sudo snap install pycharm-community --classic`

■ The next slides will show you how to use the Editor and
Debugger

    ☐ There might be small differences depending on OS and
version

PYCHARM – EDITOR

Task 1: Create a new Pycharm project (1)

■ We will start by creating a new PyCharm project
   □ A project is a folder managed by PyCharm with configurations for Python interpreter, git, etc.
■ Follow these steps for the creation of the project (see following slides for help):
   1. Select `File -> New Project...` in the menu or click `Create New Project` at the first start of PyCharm
   2. Select the project folder (does not need to be empty) to create the project in
   3. Select the Python interpreter
   4. Click `Create` to create the project

Task 1: Create a new Pycharm project (2)
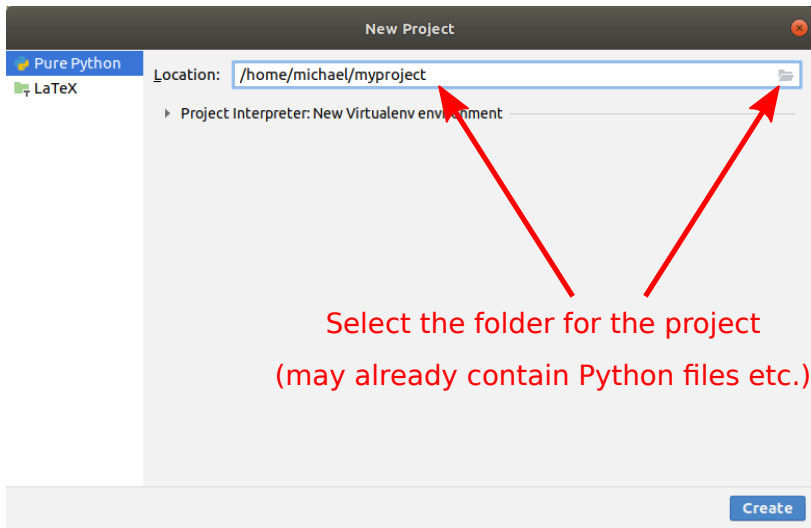
Task 1: Create a new Pycharm project (3)
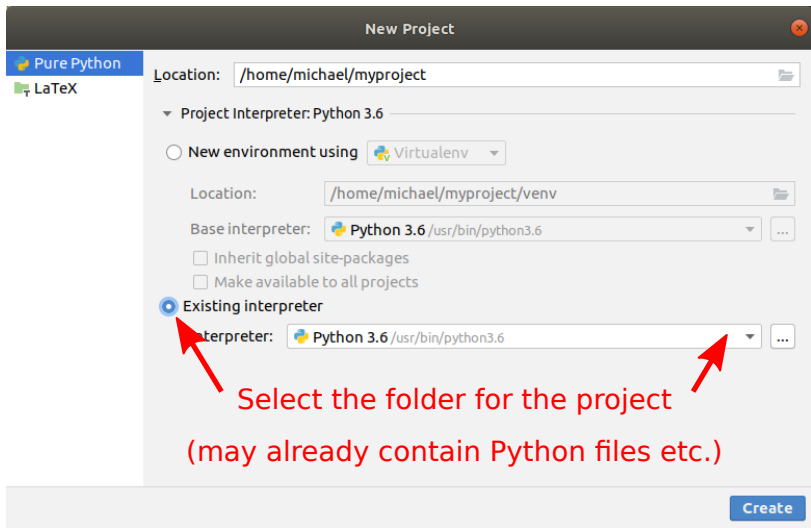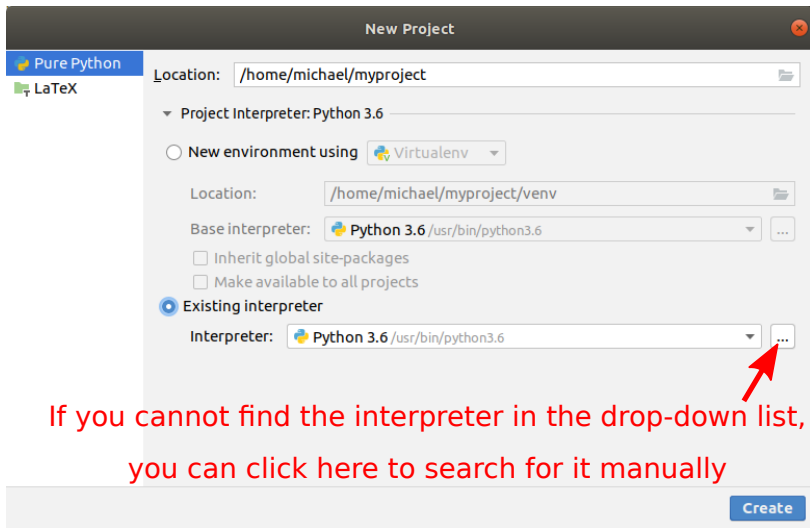
Select the folder for the project

(may already contain Python files etc.)

Task 1: Create a new Pycharm project (4)

If you cannot find the interpreter in the drop-down list,

you can click here to search for it manually

Task 1: Create a new Pycharm project (5)

Select "System Interpreter" and click here
to search for the installed Python version

Task 1: Create a new Pycharm project (6)

Select the Python version you installed (prefer versions with "m" at the end) and click "OK"
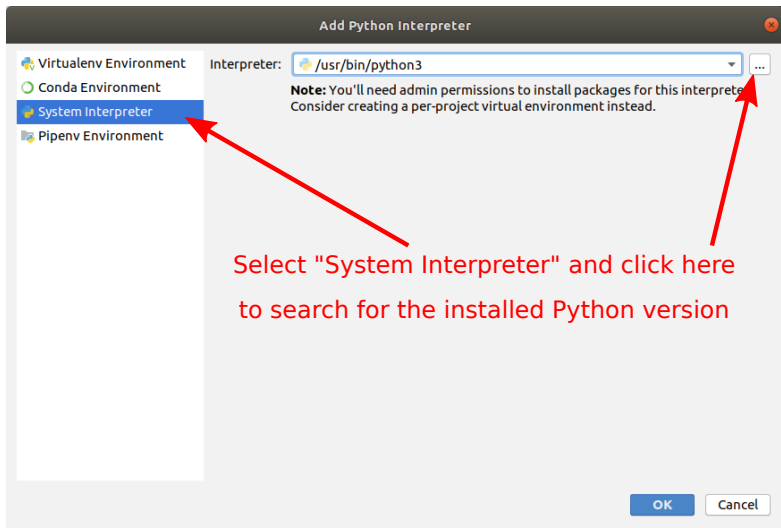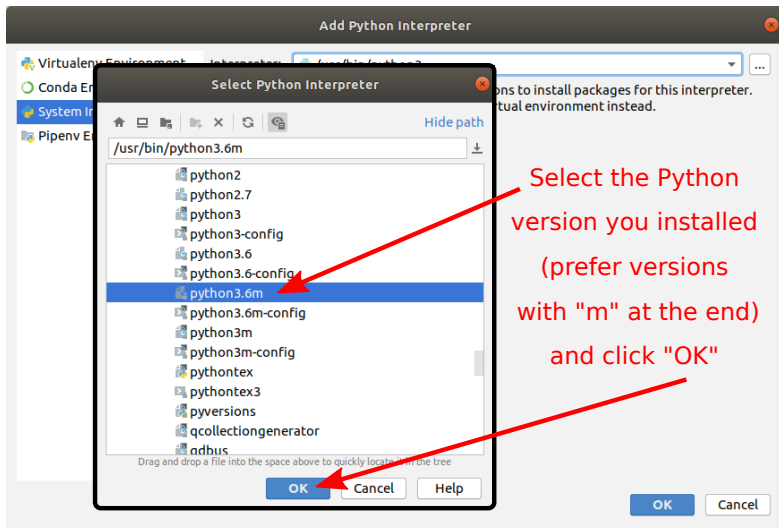
Task 1: Create a new Pycharm project (7)
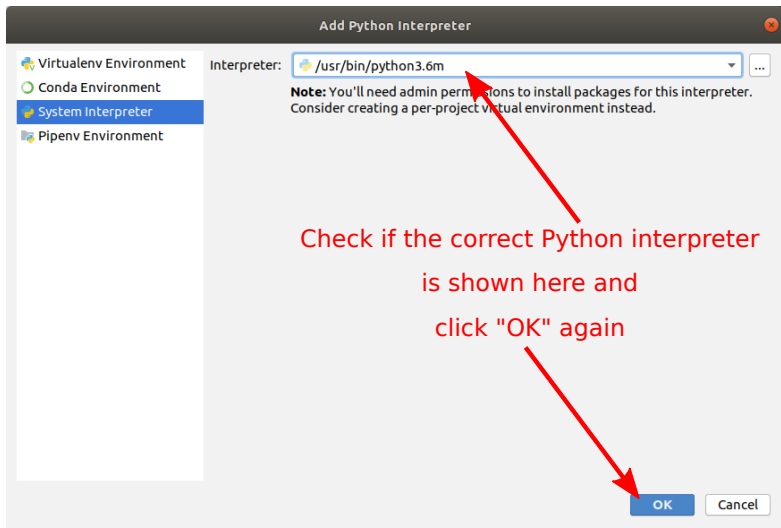
Task 1: Create a new Pycharm project (8)
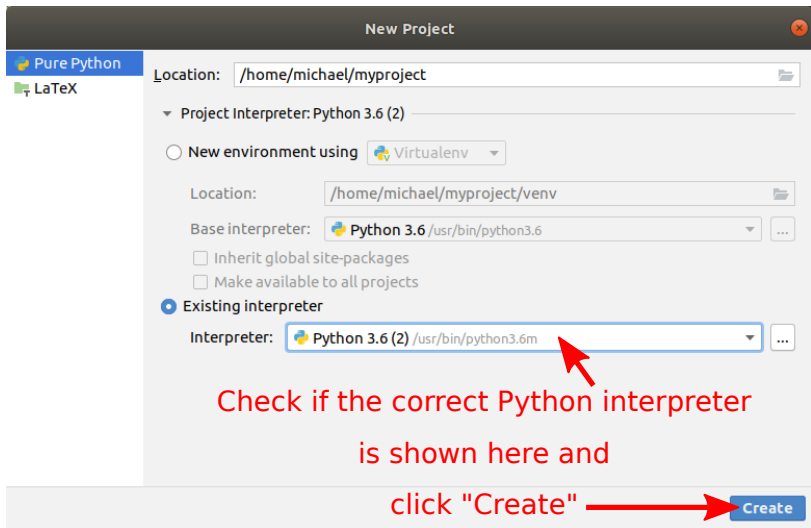
Task 1: Create a new Pycharm project (9)
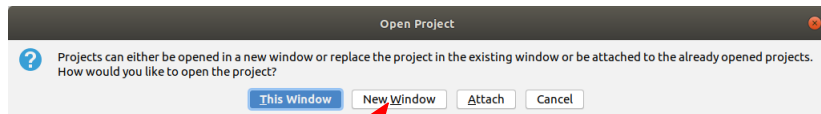
Click "New Window" and wait for the project to be created

(might take a few seconds)

PYCHARM – PYTHON CONSOLE

Task 2: Use the project Python console (1)

■ We will now use a Python console in the PyCharm project in these steps (see following slides for help):

1. Open a PyCharm project or create a new one
2. Click on `Python Console` at the lower left corner of the PyCharm window
3. Type `print("Hello world")` into the console, press Enter, and check the output
4. Type `print(a=5)` into the console, press Enter, and check the variable explorer on the right side
5. Close the console by closing the `Python Console` tab

JⴑU

JⴑU
Institute for
Machine Learning
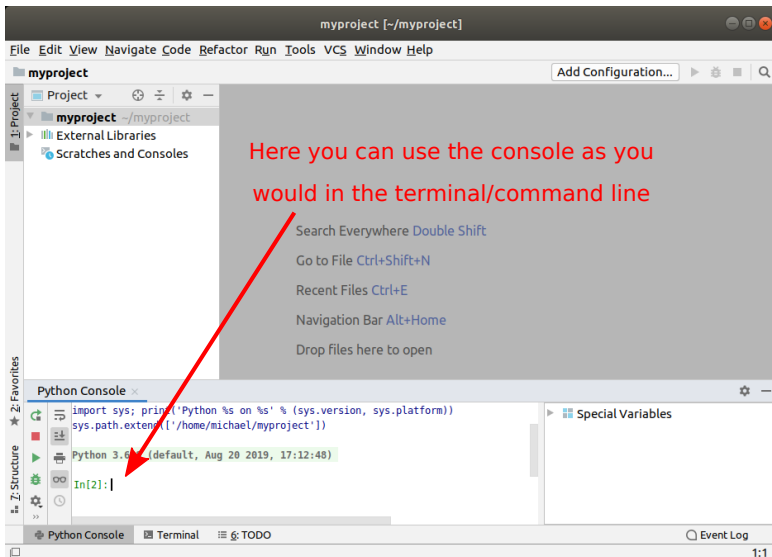
Task 2: Use the project Python console (2)
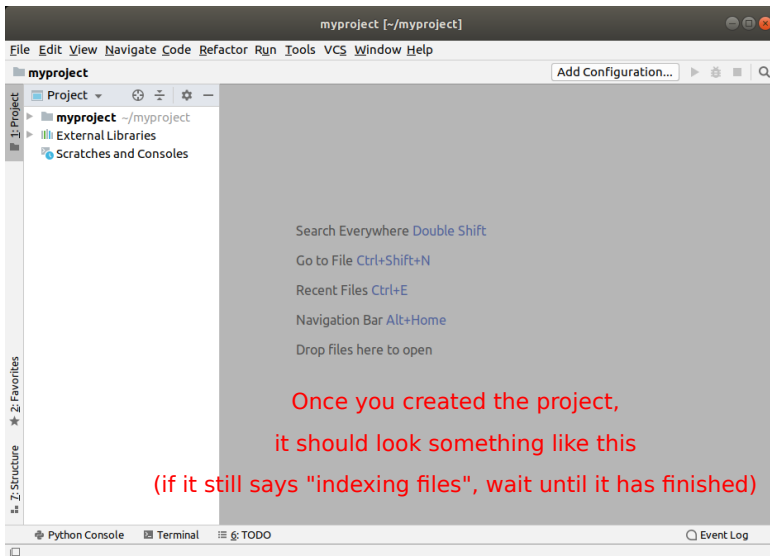
Task 2: Use the project Python console (3)
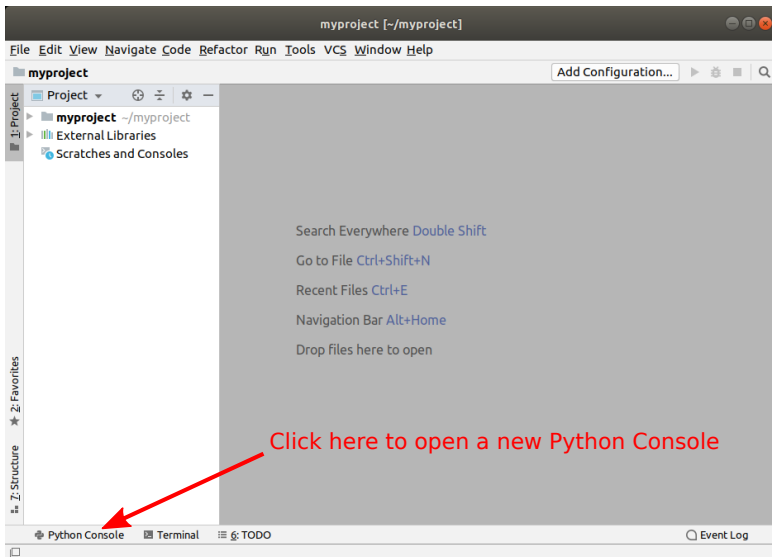
Task 2: Use the project Python console (4)

Here you can use the console as you
would in the terminal/command line

Task 2: Use the project Python console (5)

Let's start by typing some Python code
(type print("Hello world") as shown here)
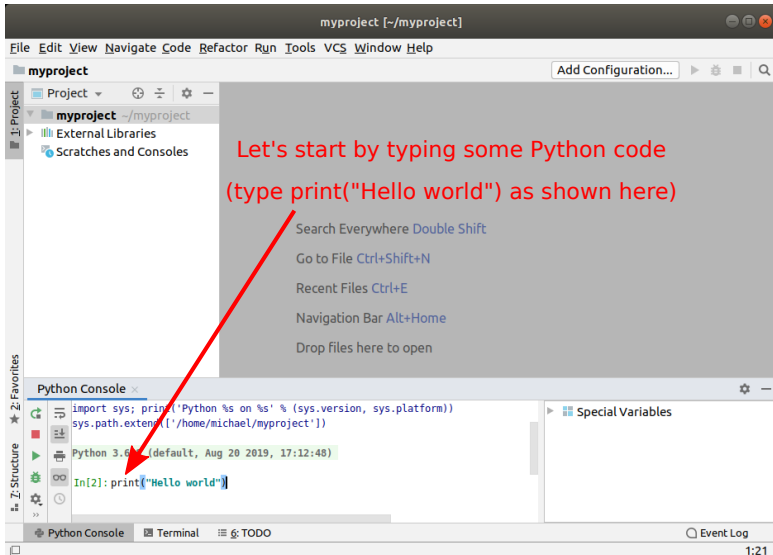
Task 2: Use the project Python console (6)

Task 2: Use the project Python console (7)

Create a variable "a" that is 5 by typing "a=5" (we'll see what that means later)

Task 2: Use the project Python console (8)

Task 2: Use the project Python console (9)
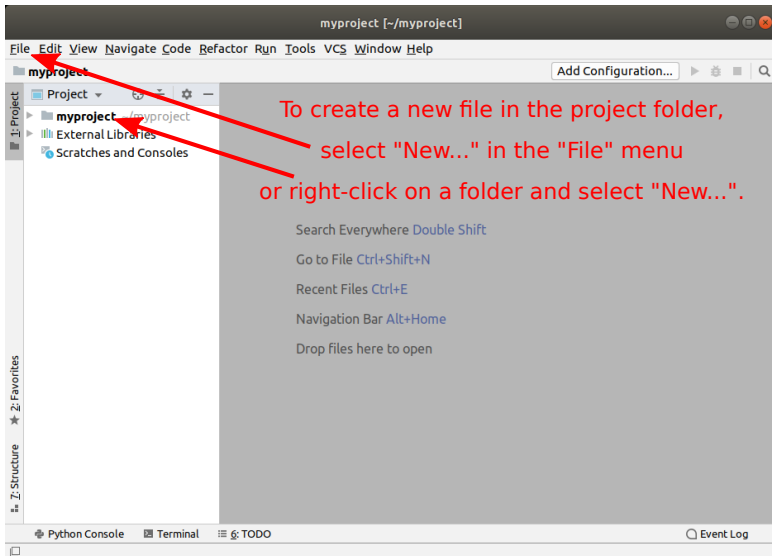
PYCHARM – RUNNING A PYTHON PROGRAM
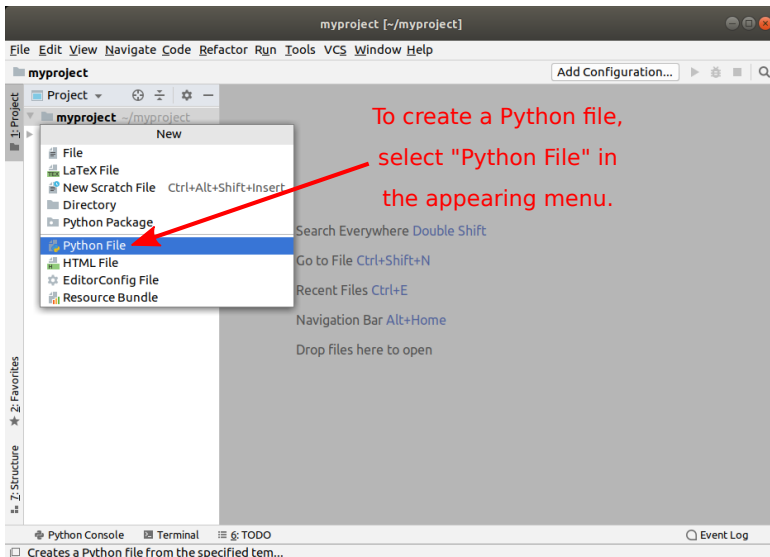
Task 3: Running a Python program (1)

■ We will now execute (=run) a Python program in PyCharm in these steps (see following slides for help):

1. Create a new Python file `test.py` with content
   `print("Hello World!")`
2. Create a run configuration for this file
3. Run the file by clicking on the "Run" button (green triangle)
4. Check the `Console` tab output (bottom of the screen); it should write "Hello World!"
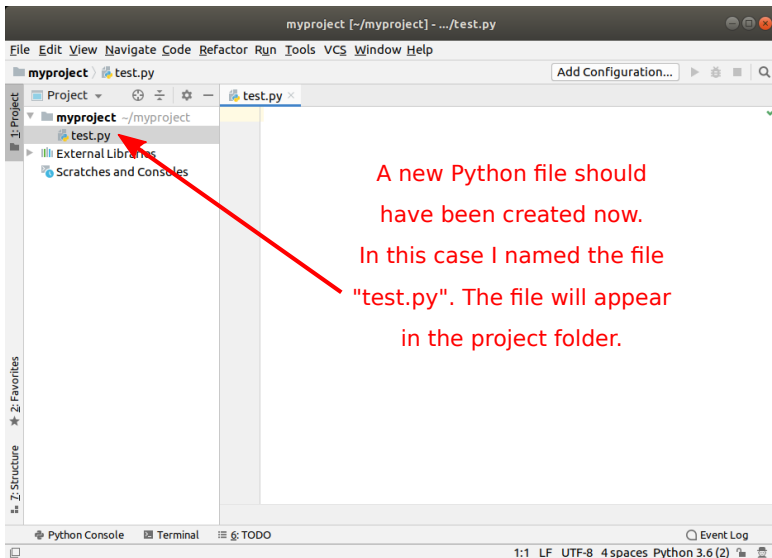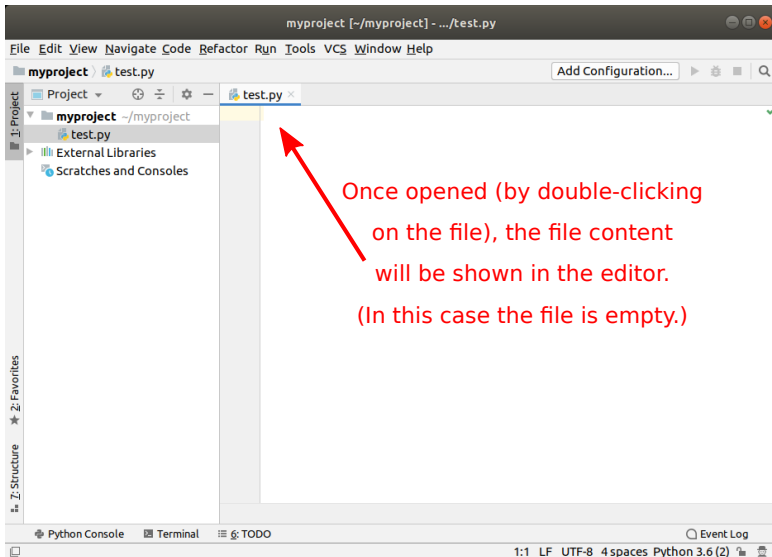
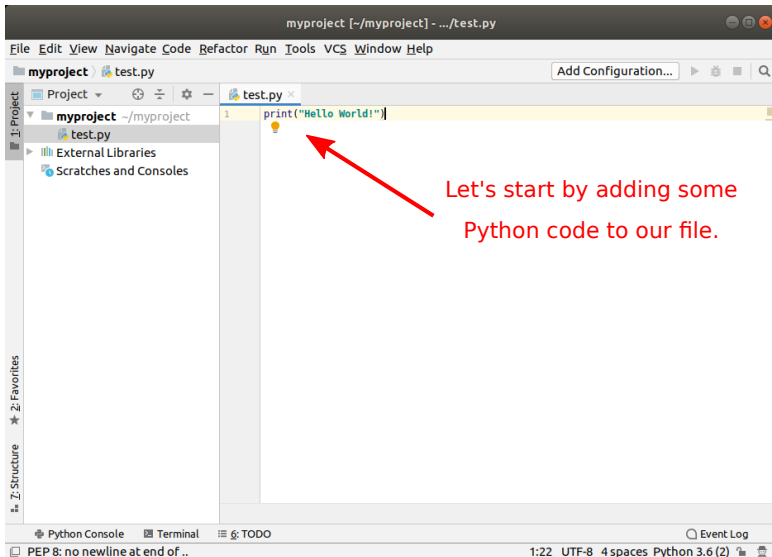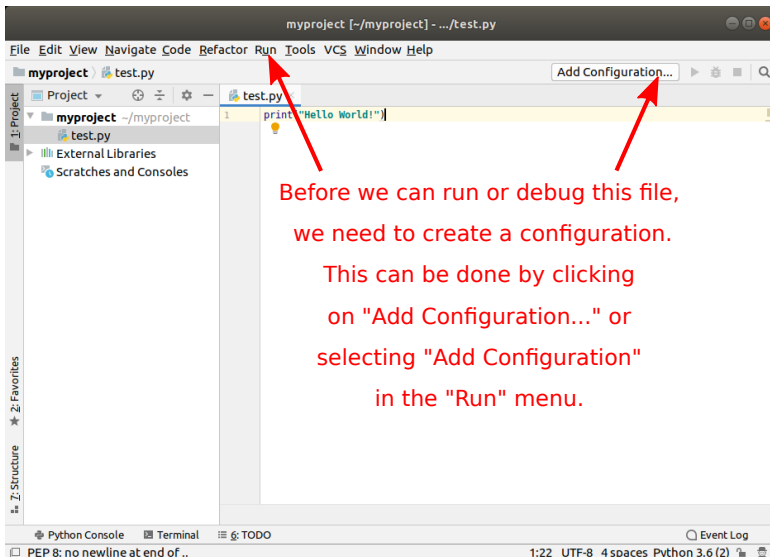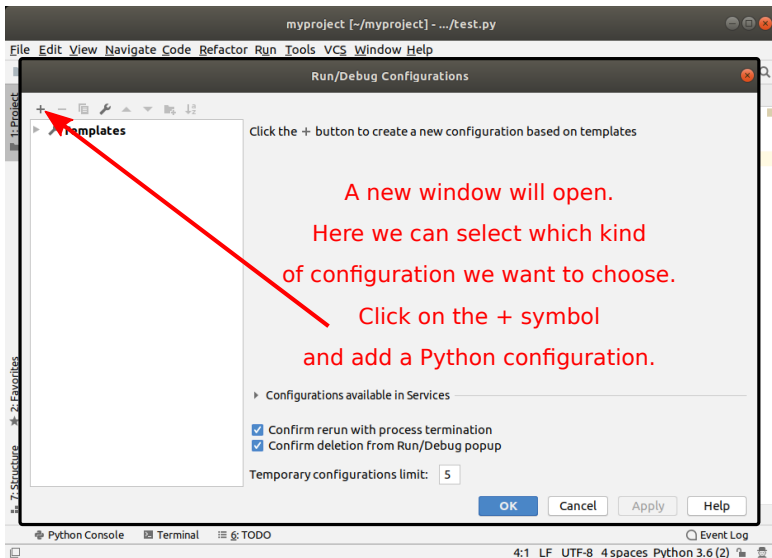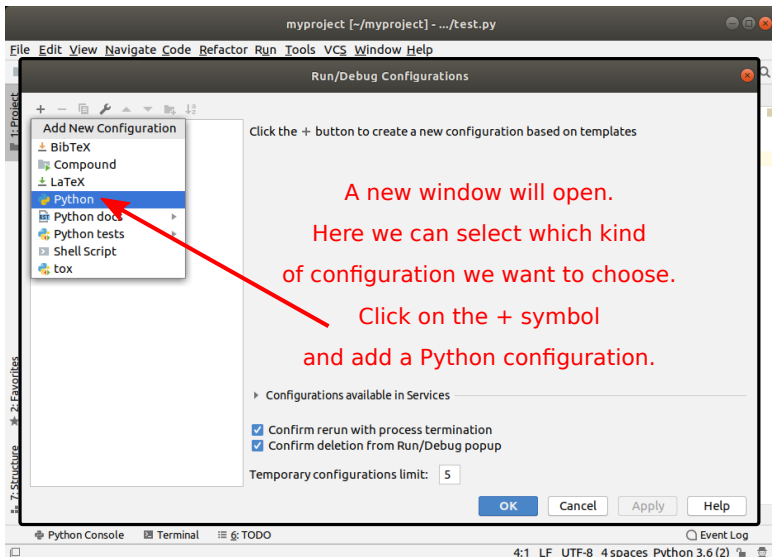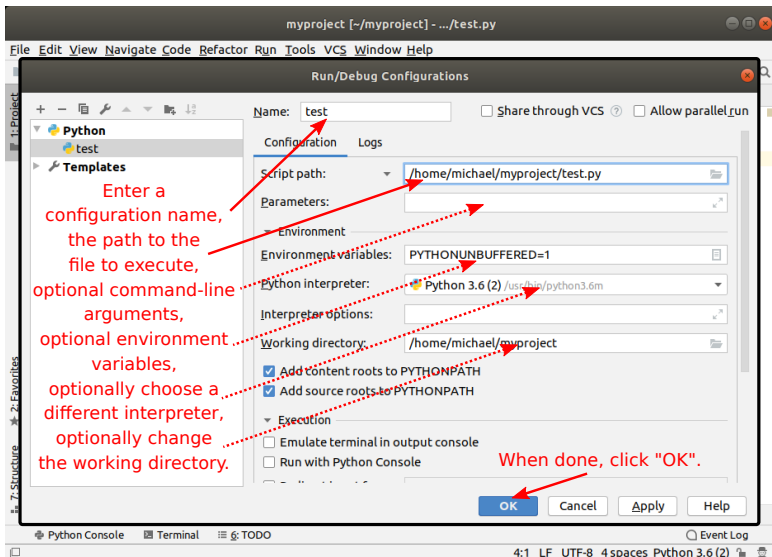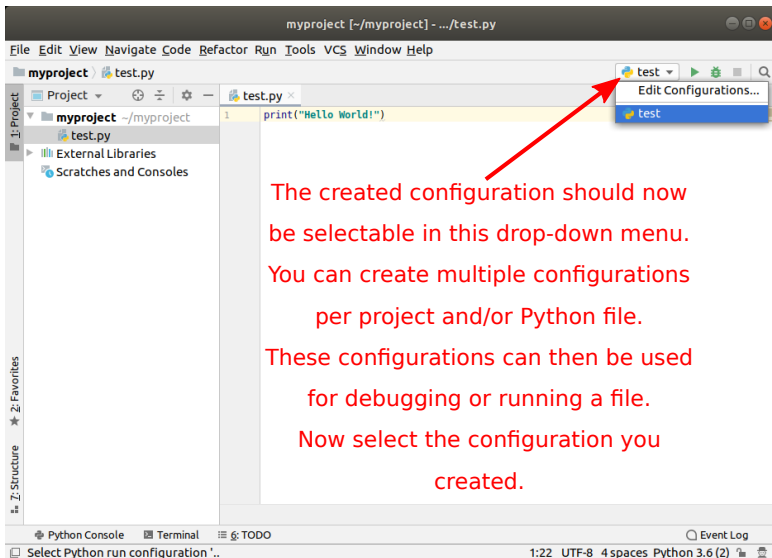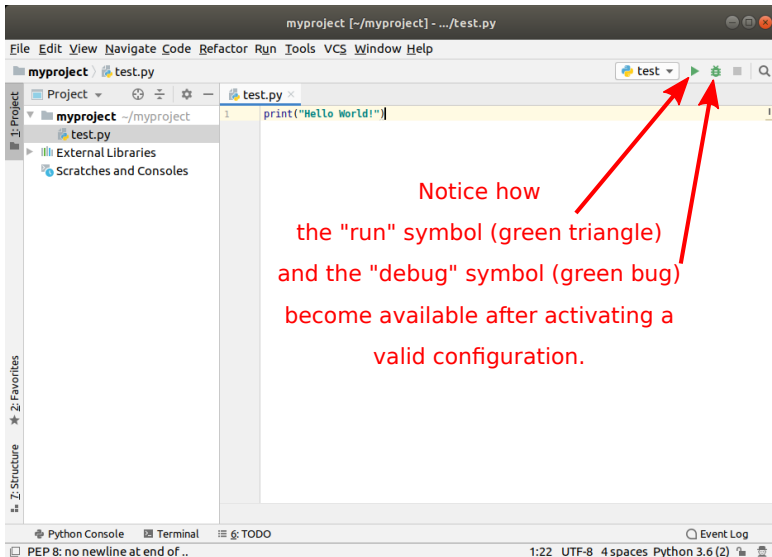# Task 3: Running a Python program (2)

Task 3: Running a Python program (3)

Task 3: Running a Python program (4)

A new Python file should have been created now. In this case I named the file "test.py". The file will appear in the project folder.

Task 3: Running a Python program (5)

Once opened (by double-clicking on the file), the file content will be shown in the editor.
(In this case the file is empty.)

Task 3: Running a Python program (6)

Let's start by adding some Python code to our file.

Task 3: Running a Python program (7)

Before we can run or debug this file, we need to create a configuration. This can be done by clicking on "Add Configuration..." or selecting "Add Configuration" in the "Run" menu.
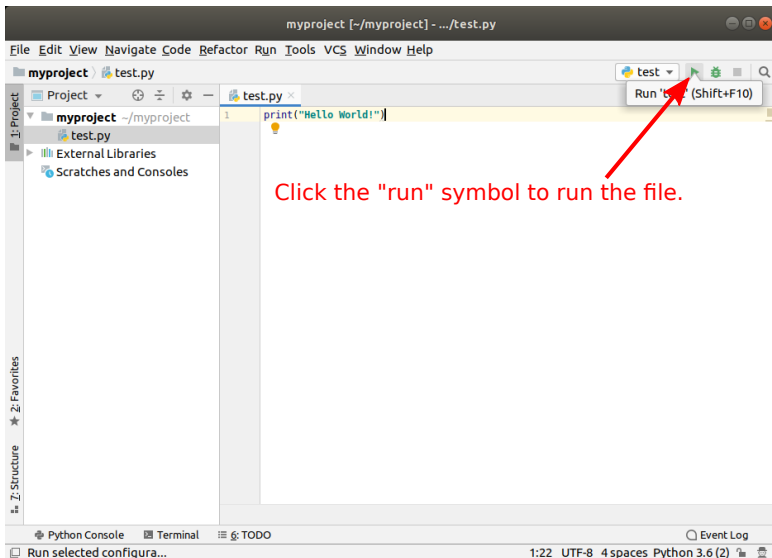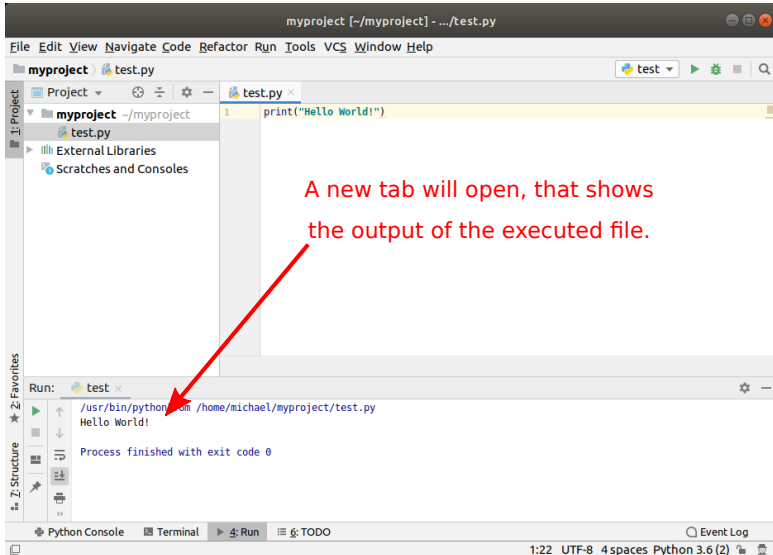
Task 3: Running a Python program (8)

Task 3: Running a Python program (9)

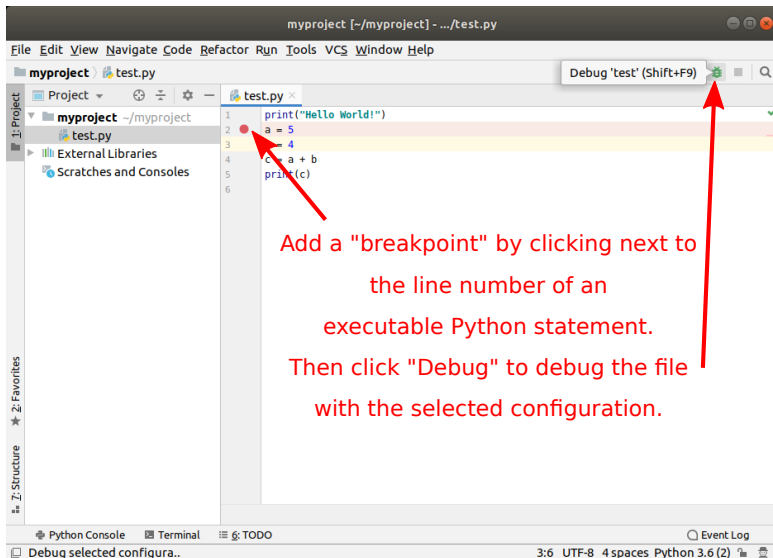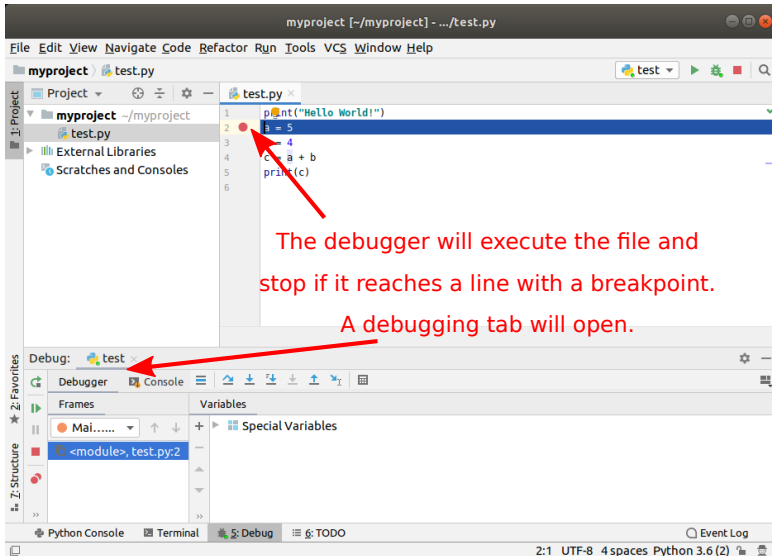# Task 3: Running a Python program (10)

Task 3: Running a Python program (11)

The created configuration should now be selectable in this drop-down menu. You can create multiple configurations per project and/or Python file. These configurations can then be used for debugging or running a file. Now select the configuration you created.

Task 3: Running a Python program (12)

Task 3: Running a Python program (13)

Click the "run" symbol to run the file.

Task 3: Running a Python program (14)

A new tab will open, that shows the output of the executed file.

PYCHARM – DEBUGGING A PYTHON PROGRAM

Task 4: Debugging a Python program (1)

- We will now debug a Python program in PyCharm in these steps (see following slides for help):

    1. Select a valid run configuration
    2. Left-click next to the line number in the editor to create a breakpoint
    3. Click the `Debug` symbol (small green bug)
    4. The program should be executed until the breakpoint or the end of the program is reached
    5. Use the `Debugger` tab to inspect or change variables
    6. Use the `Console` tab to inspect the program output
    7. Enter Python code by clicking the `Show Python Prompt` symbol
    8. Execute a single line of the program by clicking on `Step Over` or continue execution via `Resume Program`

Task 4: Debugging a Python program (3)

Add a "breakpoint" by clicking next to
the line number of an
executable Python statement.
Then click "Debug" to debug the file
with the selected configuration.

Task 4: Debugging a Python program (4)

The debugger will execute the file and stop if it reaches a line with a breakpoint. A debugging tab will open.

Task 4: Debugging a Python program (5)

The debugger tab contains 2 entries:
"Debugger" for viewing variables.
"Console" for interactions
with the Python console.
Click on "Console".

Task 4: Debugging a Python program (6)

You will see the program output so far.
In this case, the line print("Hello World!")
will have been executed and
printed "Hello World!" to the console.

Task 4: Debugging a Python program (7)

Notice how the debugger now stopped at the next line b=4.
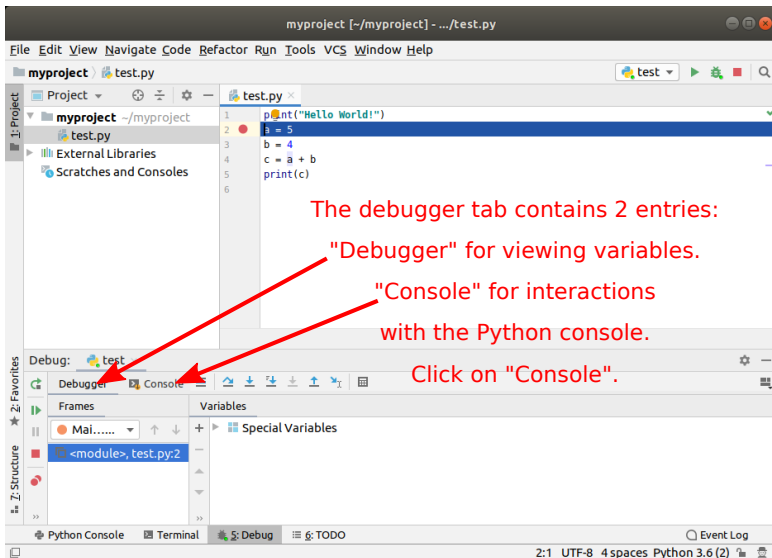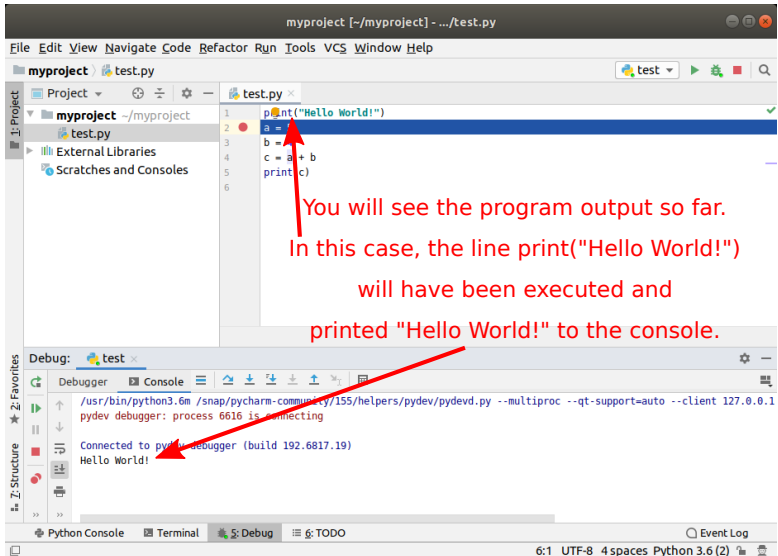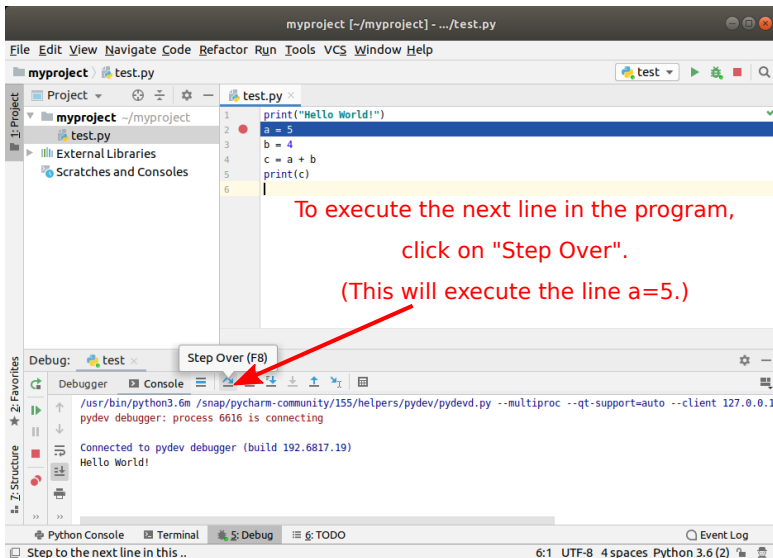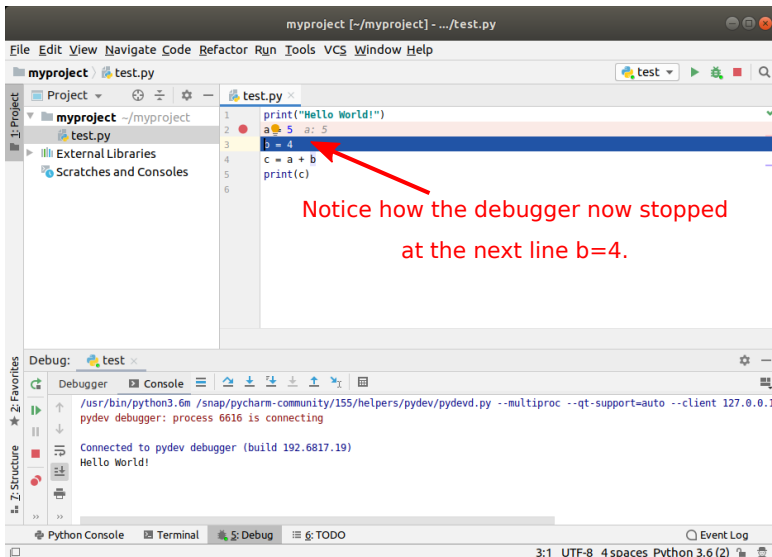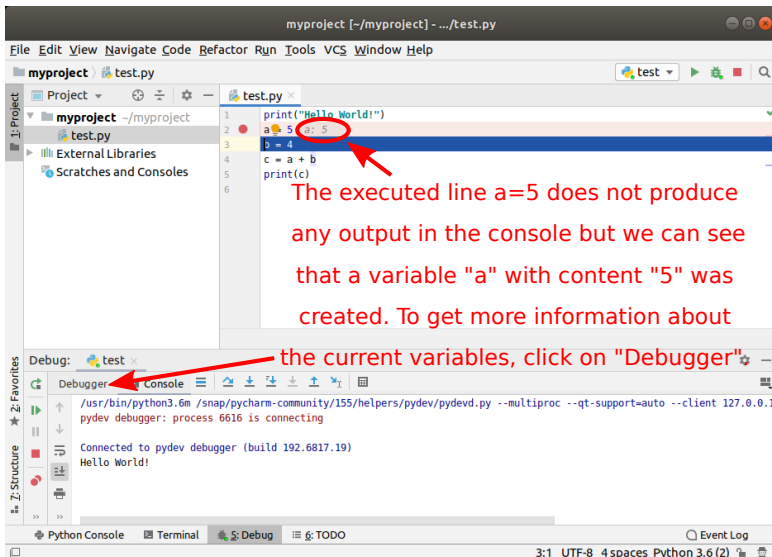
Task 4: Debugging a Python program (9)

The executed line a=5 does not produce
any output in the console but we can see
that a variable "a" with content "5" was
created. To get more information about
the current variables, click on "Debugger"

Task 4: Debugging a Python program (10)

We can see that a variable with name "a" exists now. It has type "int" and content "5".
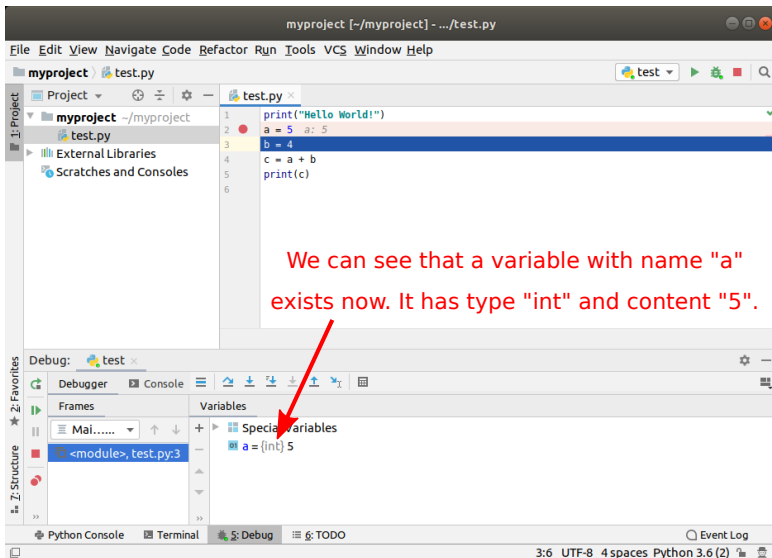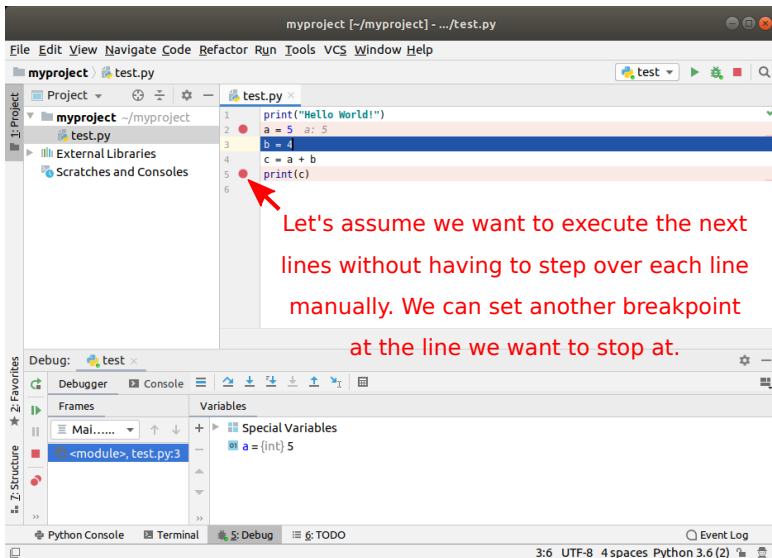
Task 4: Debugging a Python program (11)

Let's assume we want to execute the next lines without having to step over each line manually. We can set another breakpoint at the line we want to stop at.

Task 4: Debugging a Python program (12)

Click on "Resume Program" to resume the execution of the program until the next breakpoint or the end of the program.
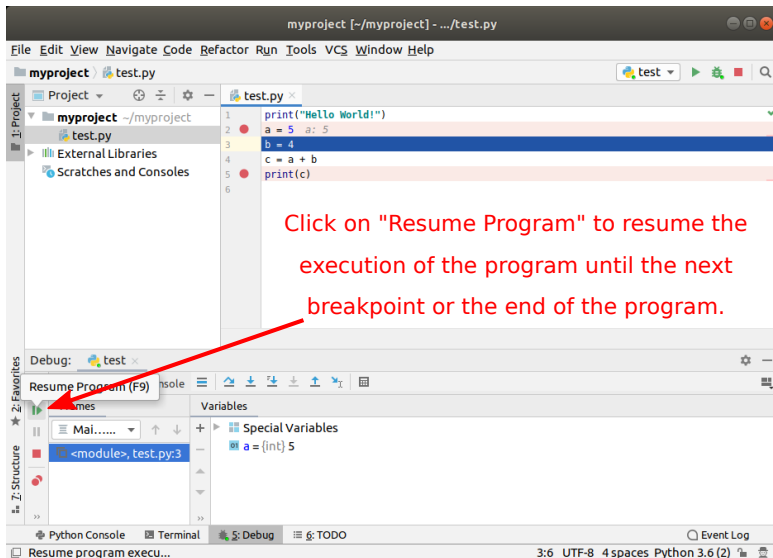
Task 4: Debugging a Python program (13)

The debugger will execute the program until it reaches the next breakpoint. We can see that now more variables have been created.

Task 4: Debugging a Python program (14)

If we continue the execution again, the debugger will continue until the end of the program, since no more breakpoints were set. After execution, no variables are shown.

Task 4: Debugging a Python program (15)

In "Console" we can see the output of the program after execution has been completed. We can see that "Hello World!" and "9" have been printed to the console.

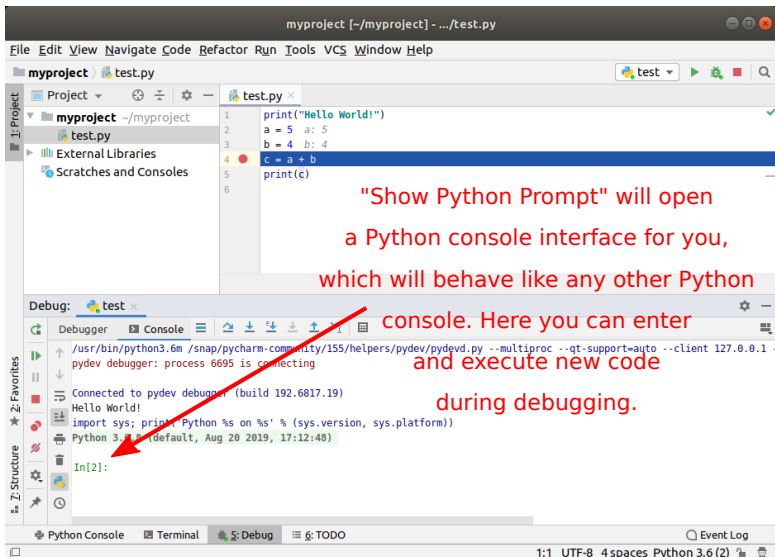This means the program was executed without errors.

Task 4: Debugging a Python program (16)

Task 4: Debugging a Python program (17)
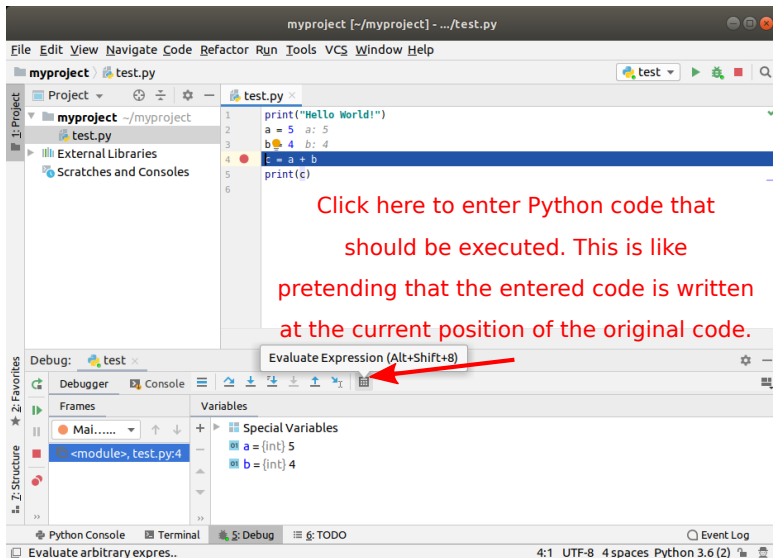
Task 4: Debugging a Python program (18)
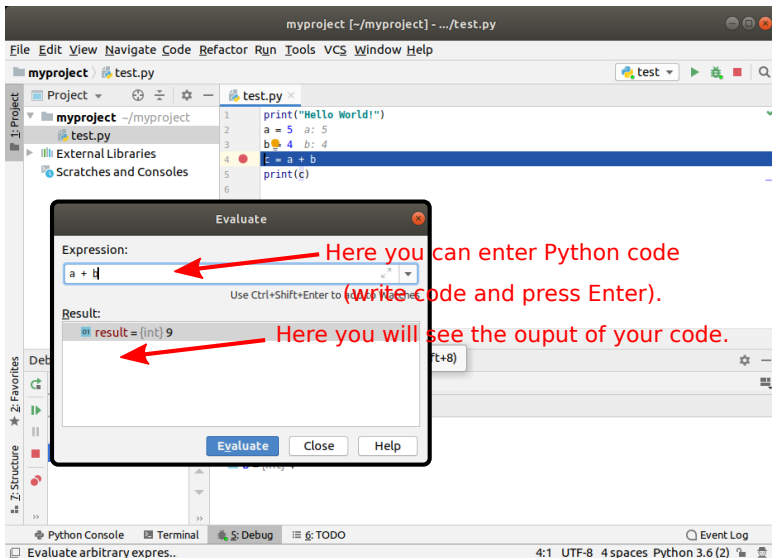
Task 4: Debugging a Python program (19)

Task 4: Debugging a Python program (20)

Task 4: Debugging a Python program (21)

For example, we can manipulate the content of variable "b" to be 6 instead of 4.

Type "b=6" and press Enter.

Task 4: Debugging a Python program (22)
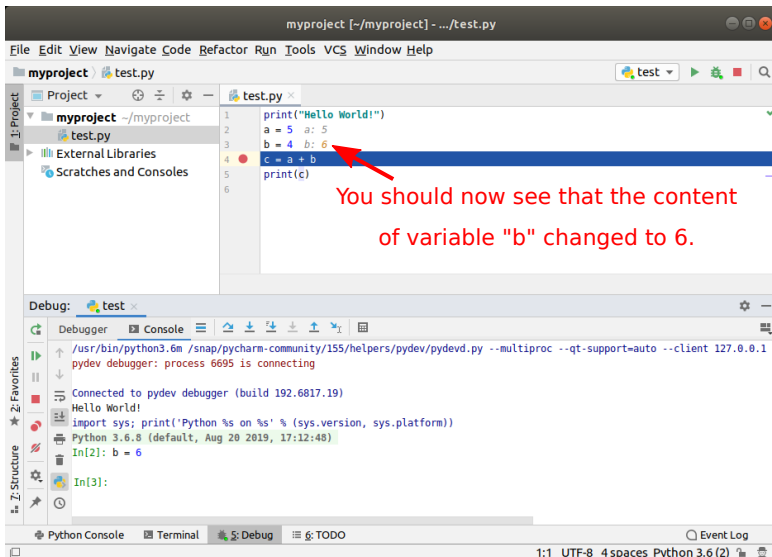
Task 4: Debugging a Python program (23)
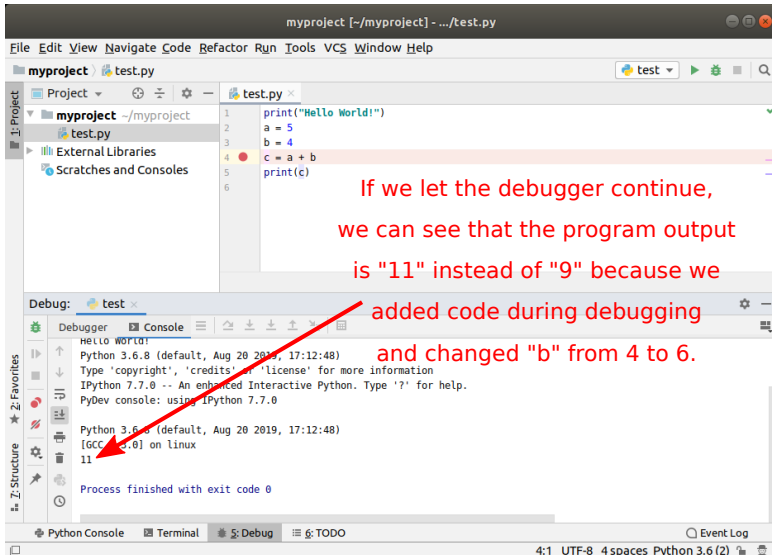
Now we are all set up and good to go! :)

OUTLOOK

Outlook

- Next time we will finally dive into Python code and learn about
  - Python syntax, file structure, and variables
  - Basic datatypes and conversions
  - Basic Python operators
- Next files:
  - `00_comments_variables.py`
  - `01_tuples_lists_indices_dictionaries_slices.py`