Python from scratch

Technical Introduction

www.sdacademy.pl

Agenda

1. Few words about the course
2. Why Python
3. Course modules
4. Common use cases of Python
5. Am I ready for the course?
   - Python
   - PyCharm
6. Q&A

Few words about the course

1. Each presentation contains much more information than required for a Junior position - you don’t have to solve each exercise and understand very deeply each topic.
2. Each module is **practical** - there are no theoretical-only modules. Typically you will spend about 70-80% of your time on practice and 20-30% on learning new theory.
3. From the very beginning - try to **work as a team**. Learn, gain experience and review your knowledge together for best results.

Why Python?

Why Python?

1. It is an interpreted language.
2. Made to be understood.
3. It is dynamically typed.
4. Made for everyone.
5. Batteries included.
6. Everything is an object.
7. Python has a huge and active community.

Course modules

Python fundamentals

1. Introduction to language
2. Basic data structures & language elements
3. Bite-sized examples
4. Exercises with provided solutions
5. Many elements common with other languages

- **Must: -**
- **Should:** -

Git system - video

1. Wide-spread version control software
2. Git enables teams to work together
3. Command line gives basic control of your operating system

- **Must: -**
- **Should:** -

Python technology

1. Working with multiple Python versions
2. Package management via Pip
3. PyInstaller

- **Must:** -
- **Should:** Git, Python Fundamentals

software **development** academy

Software testing and TDD

1. Built-in testing library: unittest
2. Quick overview of testing concepts and different test types
3. Test structure explained
4. Test-driven Development concept in theory and in practice

- **Must:** Python Fundamentals
- **Should:** Git, Python Technology

Python Advanced Features

1. Advanced Python constructs
2. Python-specific abstractions
3. Multithreading/multiprocessing
4. Inheritance
5. A mention of Python internals, like MRO, memory management, GIL
6. User leaves with an understanding of key concepts.

- **Must:** Python Fundamentals, Python Technology
- **Should: -**

software **development** academy

Software Testing Advanced

1. Introduction to popular testing libraries
2. Introduction to common concepts like mock/mocking
3. Performance testing

- **Must:** Python Fundamentals, Python Technology, Software Testing Fundamentals
- **Should:** Python Advanced Features

software **development** academy

Algorithms and Data Structures

1. Classical computer science topic
2. Shows how to solve common algorythmic problems
3. Introduction to complex data structures like queues, trees and so on

- **Must:** Python Fundamentals
- **Should:** Python Advanced Features

Design patterns & good practices

1. Shows good coding style, used throughout the industry
2. Gives course atendees tools to be used during their whole carreer:
   - Clean code concept
   - KISS
   - SOLID
   - Common design patterns

- **Must:** Python Advanced Features
- **Should: -**

SQL databases

1. New language – SQL
2. Enables users to communicate with databases
3. Introduction to relational databases
4. CRUD
5. Transactions

- **Must:** -
- **Should: -**

Databases - programming

1. Teaches atendee how to communicate with the databases from their own scripts/applications.

- **Must:** Python Advanced
- **Should:** Relational databases, NoSQL databases

HTTP basics - video

1. Internet’s communication protocols
2. Teaches how services communicate with each other
3. Command-line tools
4. REST-ful APIs

- **Must:** -
- **Should:** -

HTML, CSS, JavaScripts

1. Three common languages used in frontend development
2. HTML
3. CSS
4. JavaScript

- **Must:** -
- **Should:** -

Backend technologies

1. Most common libraries used in backend development
2. MVC pattern
3. REST-ful APIs

- **Must:** Python Advanced Features, Relational Databases
- **Should:** Frontend Technologies, HTTP

Agile, Scrum - video

1. Shows a modern approach to IT project
2. Used in many companies

- **Must:** -
- **Should:** -

Practical project

1. A chance to put fresh skills to use
2. Valuable practical, teamwork-centered experience

- **Must:** -
- **Should:** everything

Am I ready for the course?

Am I ready for the course?

1. Python 3.7 or later installed
2. Git installed
3. Dedicated room on disk to store everything course-related
4. Set up GitLab account

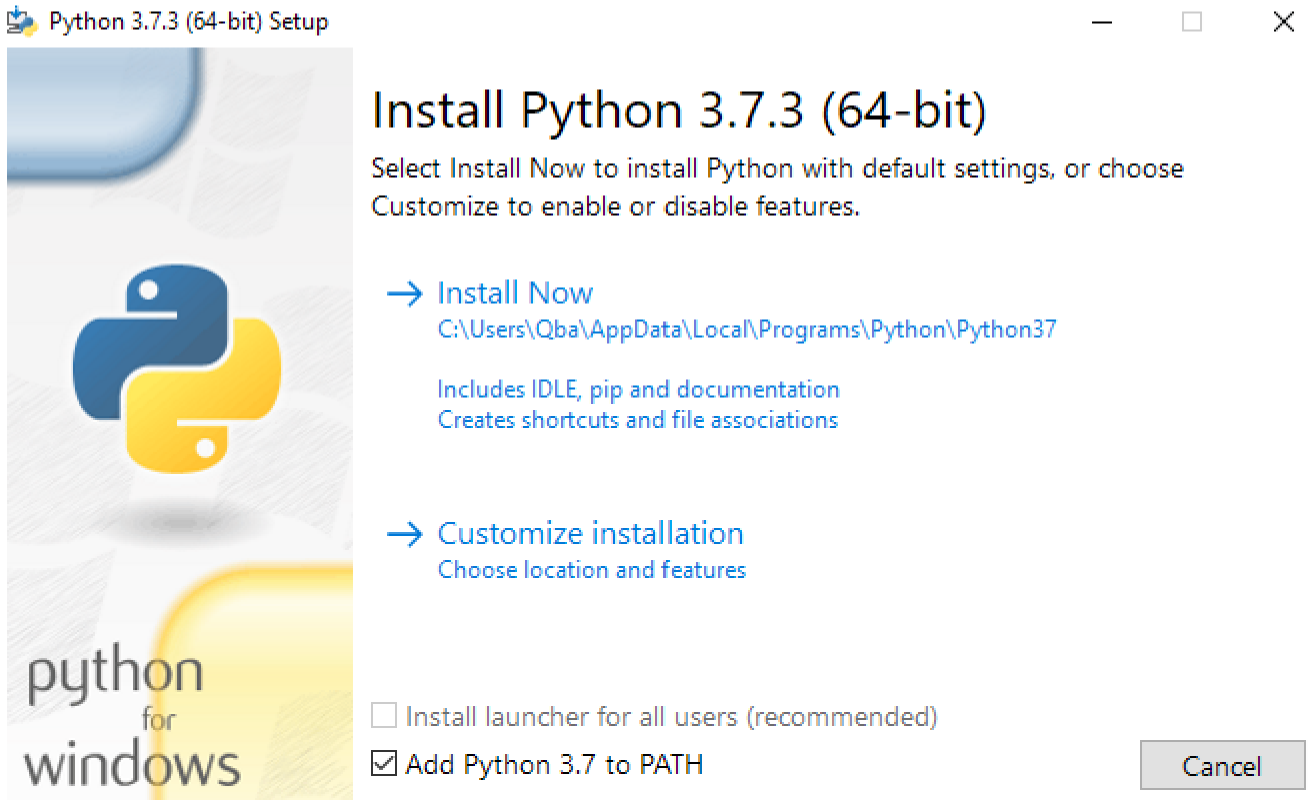
Installing Python

Installing Python - Linux

1. Your best option is to install latest package provided by your distribution. Most popular distributions provide python3.7 packages in their repositories.
2. For example for Ubuntu: `apt update && apt install python3.7`
3. If your distribution does not package Python 3.7 yet, you'll have to download from https://www.python.org/downloads/source/ and compile it yourself.
4. Package dependencies vary from distribution to distribution, check out `ubuntu_dependencies.sh` for required packages on Ubuntu
5. General compilation steps are:
   1. Install dependencies
   2. Download `Python-3.7.x.tar.xz`
   3. Unpack it `tar -xf Python-3.7.x`
   4. Enter the newly created directory and run `./configure`
   5. Run `make`
   6. Run `make test`
   7. If you already have python3, installing by make install will change this alias to point to python3.7. You can force it to keep current alias by running `make altinstall` (**not** recommended)
   8. Run `make install`
   9. Ensure `pip` has been installed by running `python3.7 -m ensurepip`
6. Ensure installation has gone well by checking version `python3.7 --version`

Installing Python - Windows

1. Download executable installer from https://www.python.org/downloads/windows/
2. **Important**: check *Add to PATH* checkbox
3. Run the installer by clicking *Install Now*
4. Verify installation by running `python3.7 --version` in PowerShell



Install Python 3.7.3 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

→ Install Now
C:\Users\Qba\AppData\Local\Programs\Python\Python37

Includes IDLE, pip and documentation
Creates shortcuts and file associations

→ Customize installation
Choose location and features

☐ Install launcher for all users (recommended)
☑ Add Python 3.7 to PATH

PyCharm

What is an IDE?

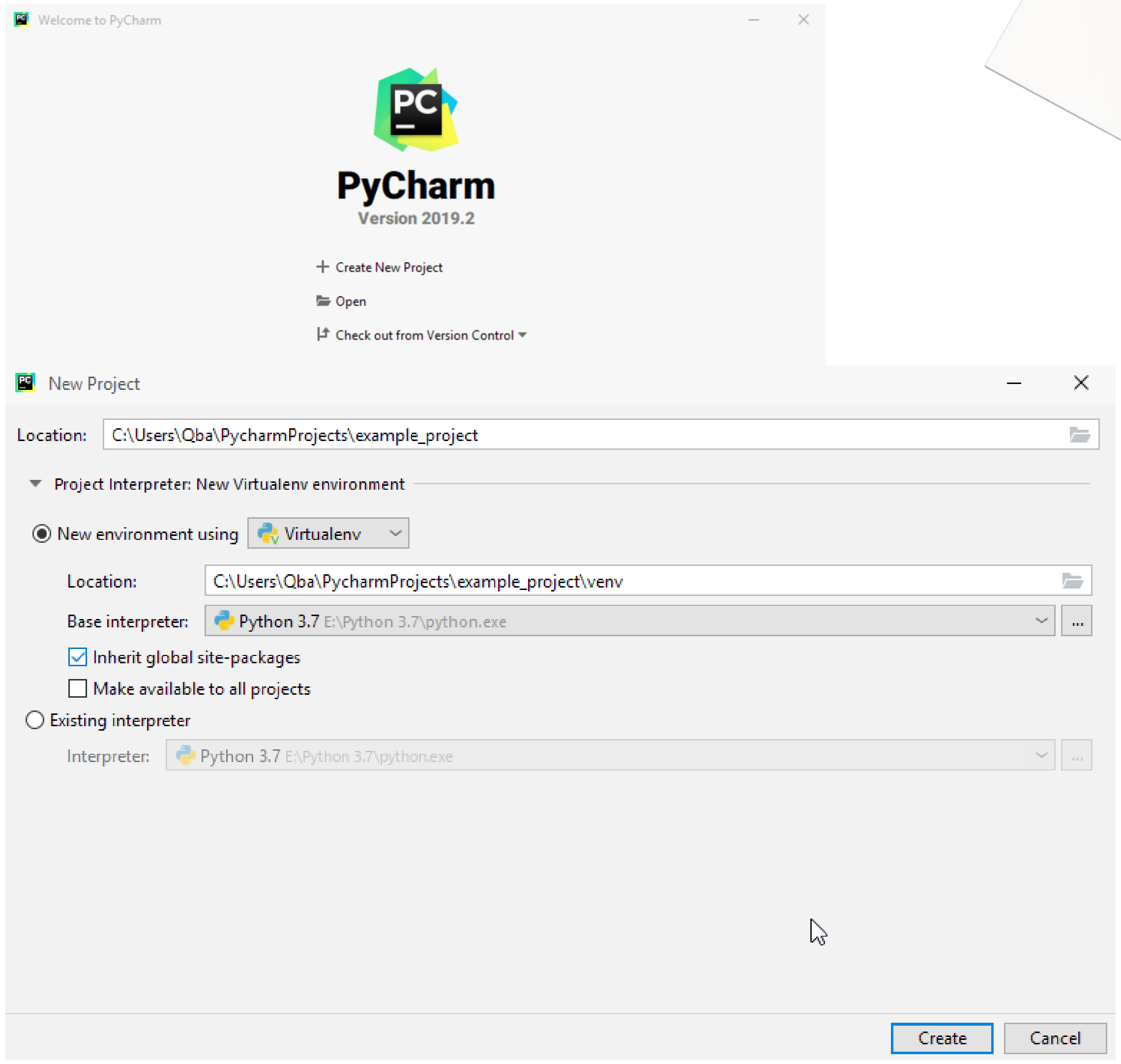
- PyCharm is an IDE – an Integrated Development Environment
- It provides:
  - an editor
  - interpreter integration
  - autocompletion
  - code introspection
  - linting
  - debugger integration
  - and much more

PyCharm – Installing PyCharm

1. Download binaries from https://www.jetbrains.com/pycharm/download/
2. **Windows:** follow installer steps
3. **Ubuntu:** sudo snap install pycharm-community –classic
4. **Other Linux distributions:**
    1. Download tar archive
    2. Unpack it where you wish to install it (/opt/PyCharm is a good choice)
    3. Run bin/pycharm.sh

PyCharm – Create new project

1. Create New Project
2. It is preferable to create an environment for it
3. You can create environment inside project directory
4. Make sure you pick the right interpreter, if you have more than one installed.



Location: C:\Users\Qba\PycharmProjects\example_project

Project Interpreter: New Virtualenv environment

[● New environment using] Virtualenv

Location: C:\Users\Qba\PycharmProjects\example_project\venv

Base interpreter: Python 3.7 E:\Python 3.7\python.exe

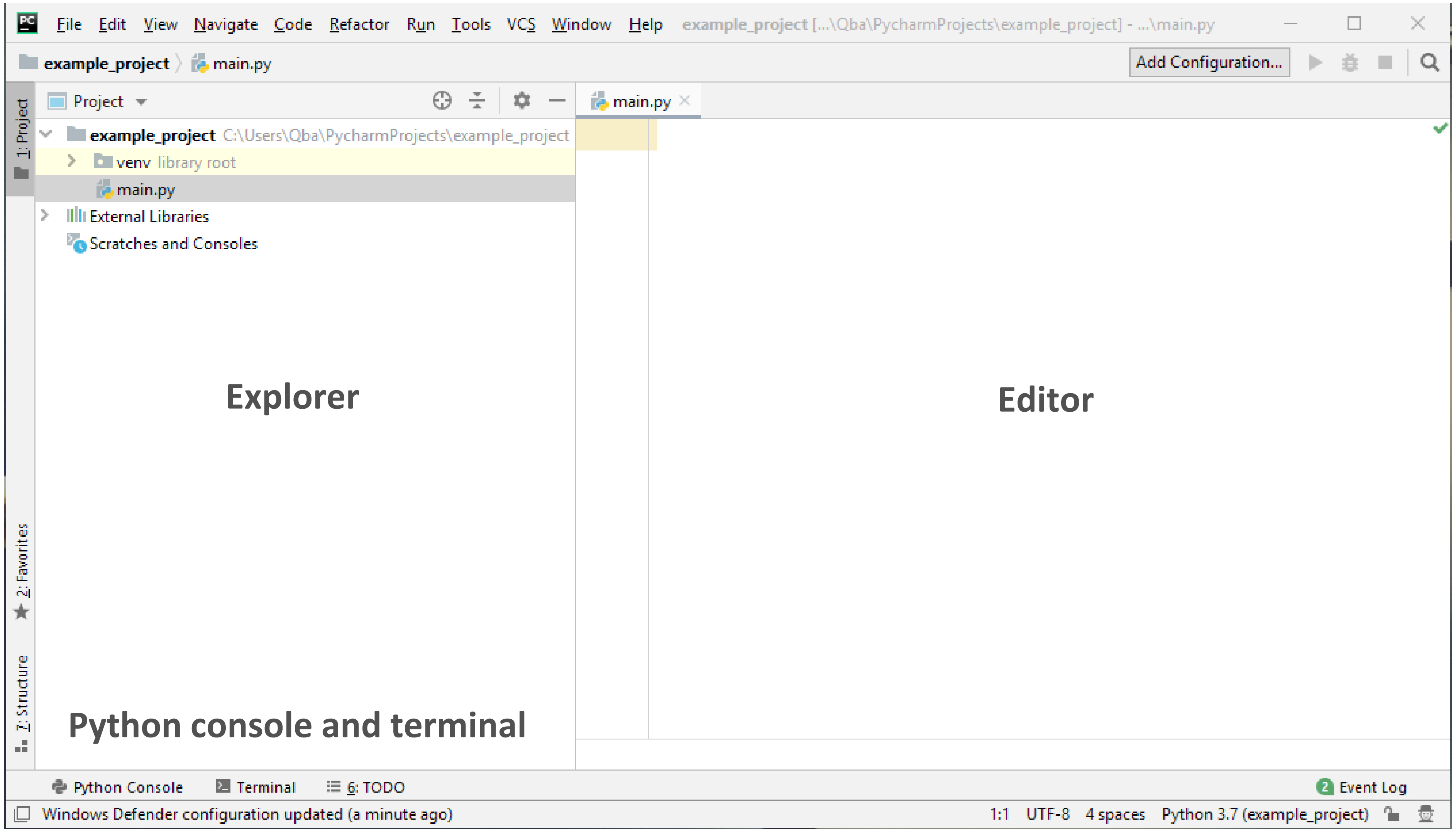
[✓] Inherit global site-packages

[ ] Make available to all projects

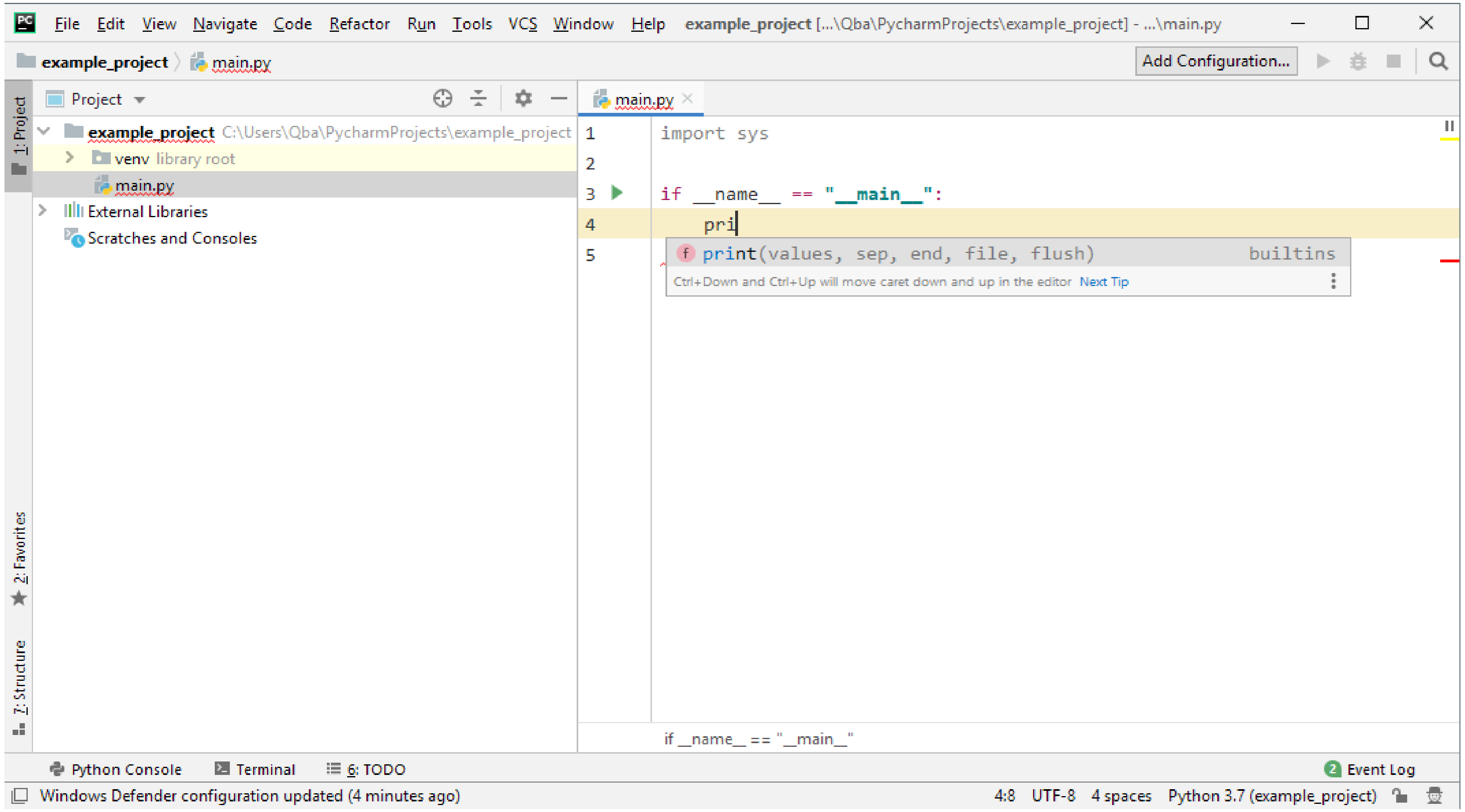
○ Existing interpreter

Interpreter: Python 3.7 E:\Python 3.7\python.exe

PyCharm – UI elements


File Edit View Navigate Code Refactor Run Tools VCS Window Help
example_project [...\Qba\PycharmProjects\example_project] - ...\main.py
example_project 〉 main.py
Add Configuration...
Project
main.py
1: Project
2: Favorites
7: Structure
example_project C:\Users\Qba\PycharmProjects\example_project
venv library root
main.py
External Libraries
Scratches and Consoles
Explorer
Editor
Python console and terminal
Python Console
Terminal
6: TODO
Event Log
Windows Defender configuration updated (a minute ago)
1:1 UTF-8 4 spaces Python 3.7 (example_project)

PyCharm – Autosuggestions


File Edit View Navigate Code Refactor Run Tools VCS Window Help example_project [...\Qba\PycharmProjects\example_project] - ...\main.py
example_project
main.py
Add Configuration...
Project
example_project C:\Users\Qba\PycharmProjects\example_project
venv library root
main.py
External Libraries
Scratches and Consoles
1: Project
2: Favorites
7: Structure
Python Console
Terminal
6: TODO
Event Log
4:8
UTF-8
4 spaces
Python 3.7 (example_project)

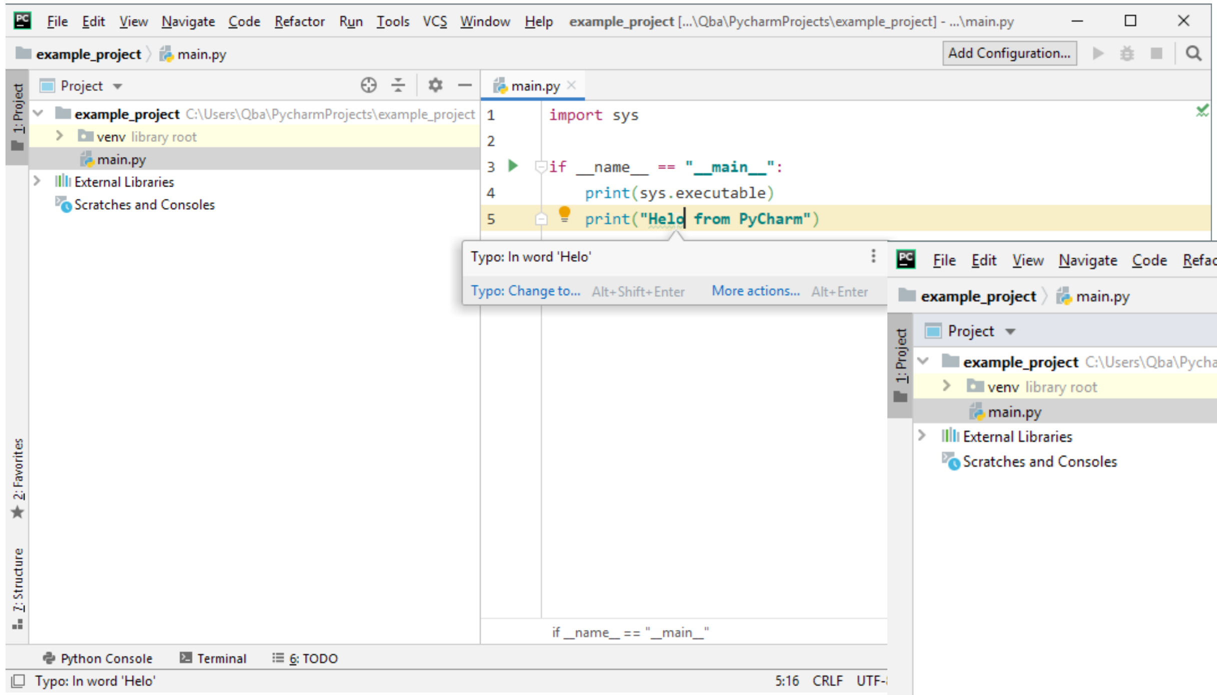
```
import sys

if __name__ == "__main__":
    pri
```

```
print(values, sep, end, file, flush)    builtins
Ctrl+Down and Ctrl+Up will move caret down and up in the editor  Next Tip
```

if __name__ == "__main__"

Windows Defender configuration updated (4 minutes ago)

PyCharm – Helpers

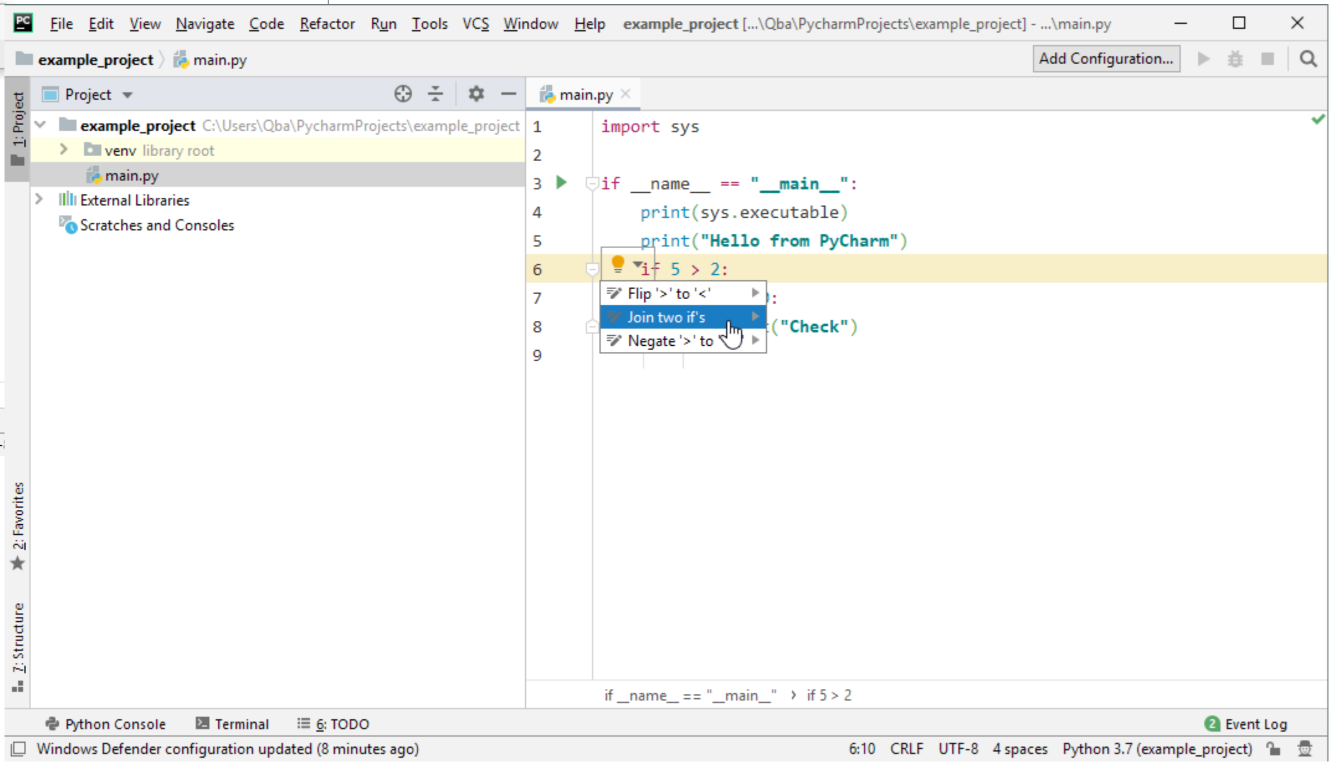


```
import sys

if __name__ == "__main__":
    print(sys.executable)
    print("Helo from PyCharm")
```

Typo: In word 'Helo'

Typo: Change to... Alt+Shift+Enter More actions... Alt+Enter

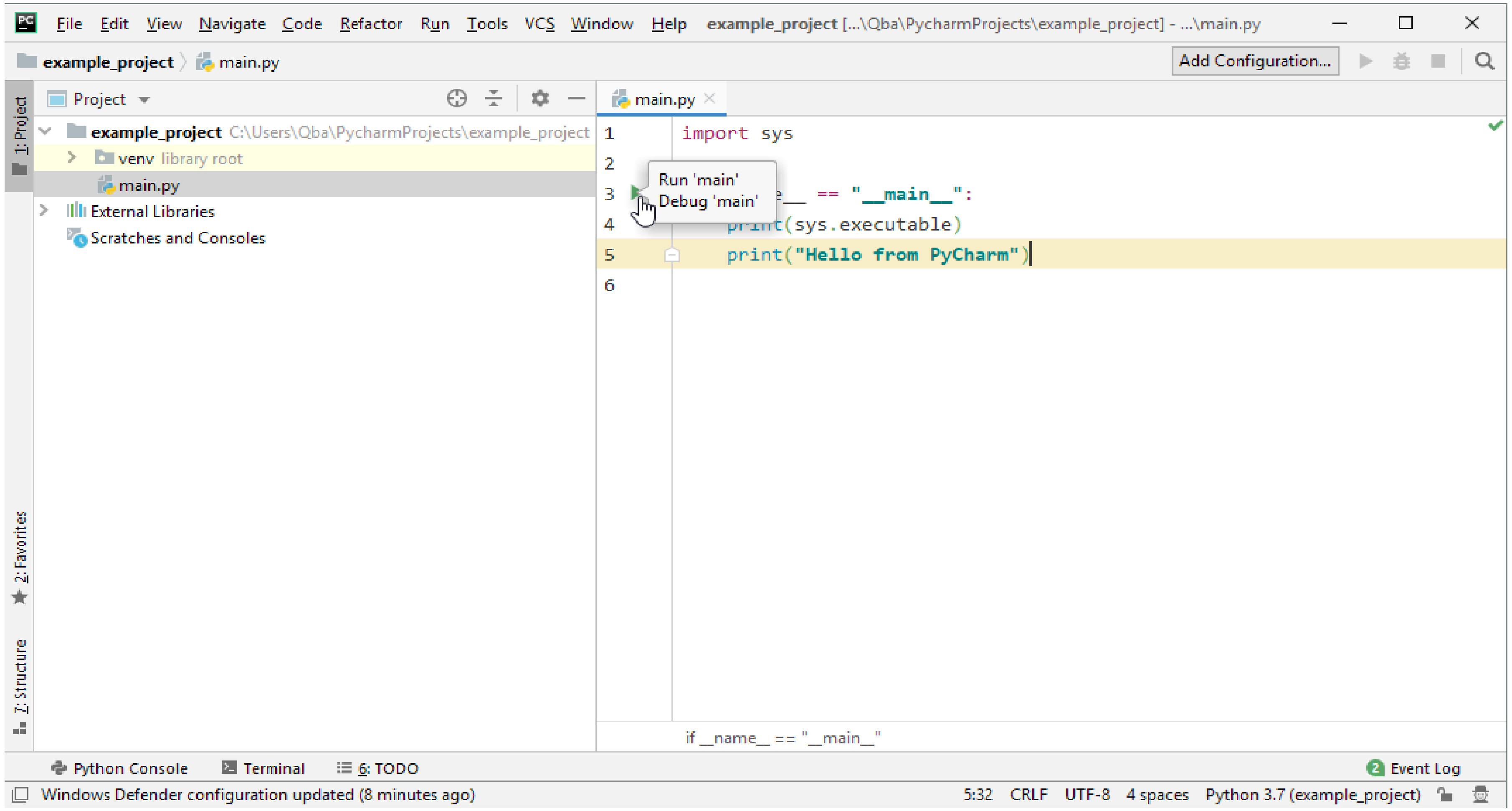
File Edit View Navigate Code Refactor Run Tools VCS Window Help example_project [...\Qba\PycharmProjects\example_project] - ...\main.py
example_project main.py
Add Configuration...
Project
example_project C:\Users\Qba\PycharmProjects\example_project
venv library root
main.py
External Libraries
Scratches and Consoles
if __name__ == "__main__"
Python Console
Terminal
6: TODO
Typo: In word 'Helo'
5:16 CRLF UTF-



```
import sys

if __name__ == "__main__":
    print(sys.executable)
    print("Hello from PyCharm")
    if 5 > 2:
        :
            ("Check")
```

File Edit View Navigate Code Refactor Run Tools VCS Window Help example_project [...\Qba\PycharmProjects\example_project] - ...\main.py
example_project main.py
Add Configuration...
Project
example_project C:\Users\Qba\PycharmProjects\example_project
venv library root
main.py
External Libraries
Scratches and Consoles
Flip '>' to '<'
Join two if's
Negate '>' to
if __name__ == "__main__" › if 5 > 2
Python Console
Terminal
6: TODO
Event Log
Windows Defender configuration updated (8 minutes ago)
6:10 CRLF UTF-8 4 spaces Python 3.7 (example_project)

PyCharm – Running your scripts

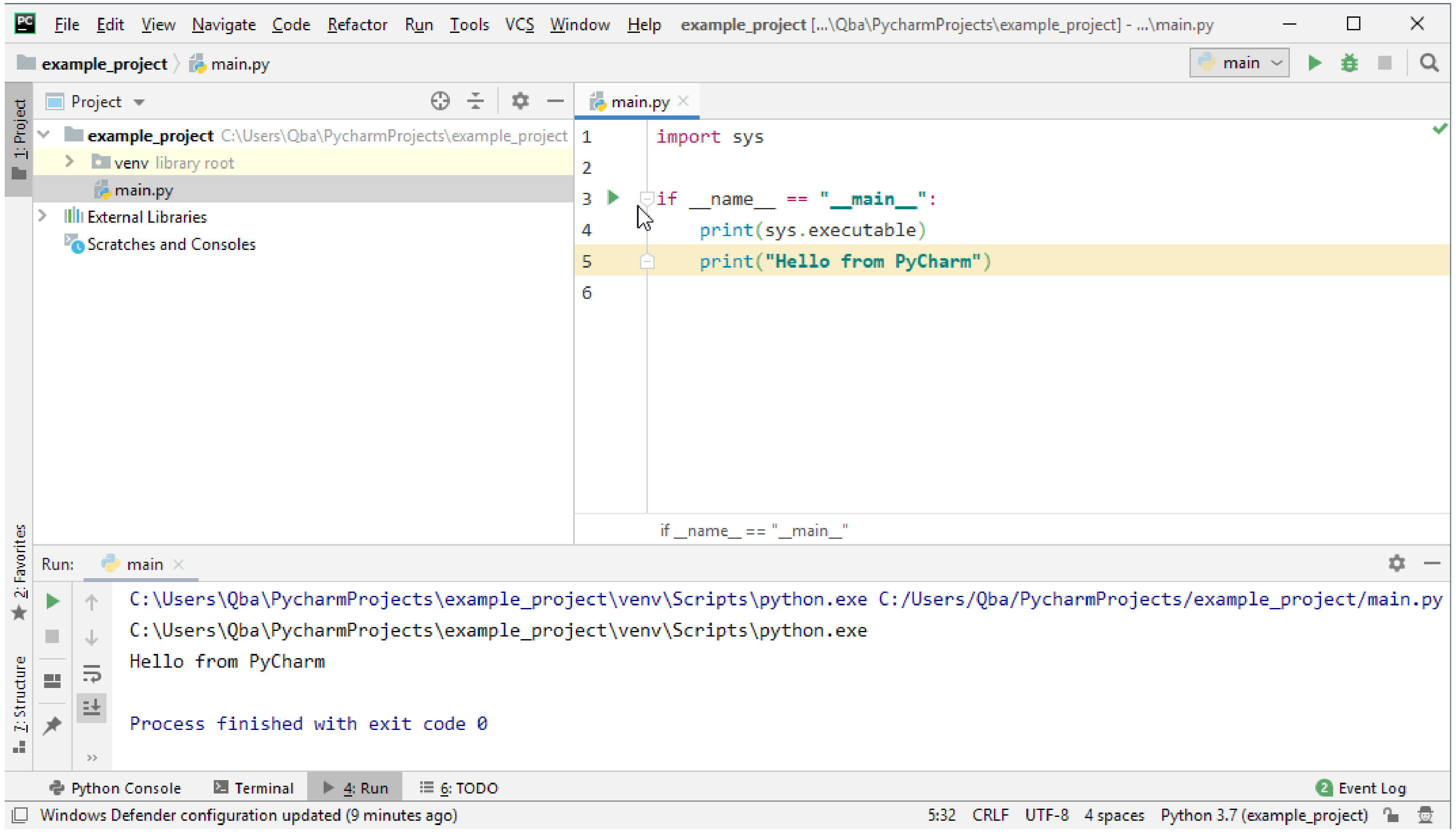

File Edit View Navigate Code Refactor Run Tools VCS Window Help
example_project [...\Qba\PycharmProjects\example_project] - ...\main.py
example_project › main.py
Add Configuration...
Project
main.py
example_project C:\Users\Qba\PycharmProjects\example_project
venv library root
main.py
External Libraries
Scratches and Consoles
Run 'main'
Debug 'main'
1: Project
2: Favorites
7: Structure
Python Console
Terminal
6: TODO
Event Log

```
import sys

if __name__ == "__main__":
    print(sys.executable)
    print("Hello from PyCharm")
```

if __name__ == "__main__"

Windows Defender configuration updated (8 minutes ago)   5:32 CRLF UTF-8 4 spaces Python 3.7 (example_project)

PyCharm – Running your scripts


File Edit View Navigate Code Refactor Run Tools VCS Window Help example_project [...\Qba\PycharmProjects\example_project] - ...\main.py
example_project main.py
main
Project
example_project C:\Users\Qba\PycharmProjects\example_project
venv library root
main.py
External Libraries
Scratches and Consoles
main.py
if __name__ == "__main__"
Run: main
Python Console Terminal 4: Run 6: TODO Event Log
Windows Defender configuration updated (9 minutes ago)
5:32 CRLF UTF-8 4 spaces Python 3.7 (example_project)

```python
import sys

if __name__ == "__main__":
    print(sys.executable)
    print("Hello from PyCharm")
```

```
C:\Users\Qba\PycharmProjects\example_project\venv\Scripts\python.exe C:/Users/Qba/PycharmProjects/example_project/main.py
C:\Users\Qba\PycharmProjects\example_project\venv\Scripts\python.exe
Hello from PyCharm

Process finished with exit code 0
```

Q&A