CITY
UNIVERSITY OF LONDON
EST 1894

IN3067/INM713 Semantic Web Technologies and Knowledge Graphs

Laboratory 1: Setting-up the Infrastructure

Ernesto Jiménez-Ruiz

Academic course: 2023-2024
Updated: February 1, 2024

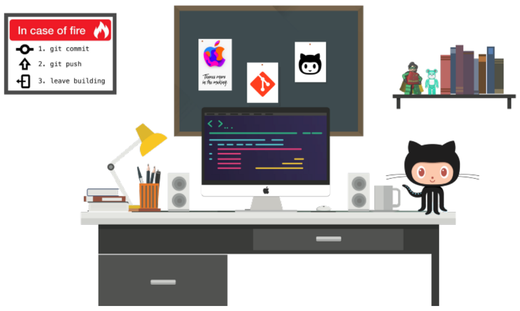
Contents

In this module we will use Java and Python as programming languages for the lab sessions. You can chose any of these languages for your lab solutions and coursework. The goal of this session is to start setting up the coding environment and the ontology editor Protégé.

1 Git Repositories

I have created for the module two *GitHub* repositories, one in Python (python-2024) and another in Java (java-2024), for the reference codes of the laboratory sessions:

`https://github.com/city-knowledge-graphs`



The use of a remote repository like GitHub for your codes is **optional**, but highly recommended in case you have any issues with your computer or if you work from different computers. This type of infrastructure is also extensively used when developing code collaboratively.

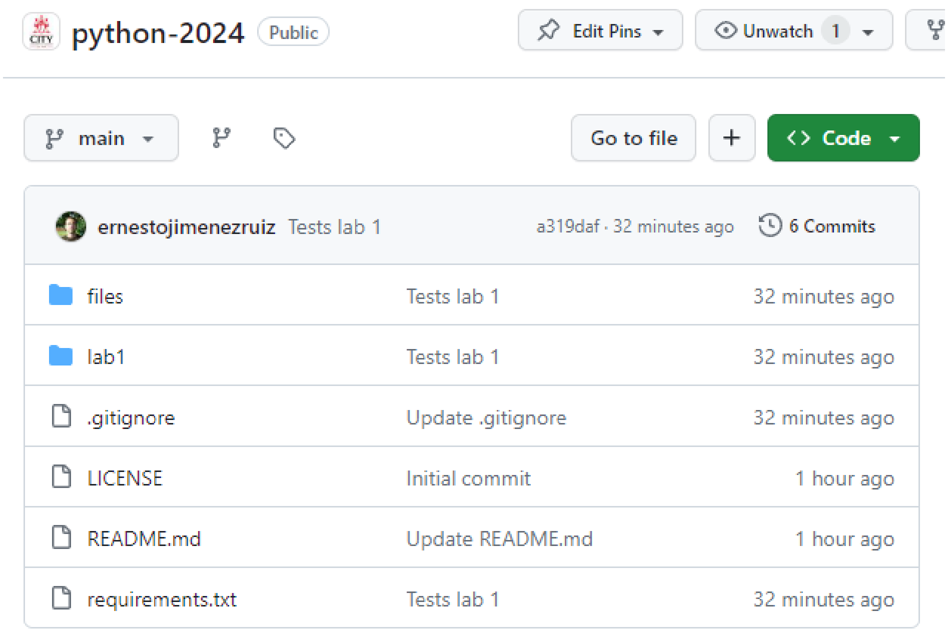
1.1 Accessing the example codes

In order to download the codes in the GitHub repositories, you can (1) use Git and clone the repository (see instructions in Section 1.2), or (2) directly download the full repository as a ZIP file. Option (1) will allow you to be up-to-date with my solutions and additional material without downloading the whole repository; option (2) is simpler, but you will need to download the whole repository every time there is a change. To download or get access to the relevant Git URLs click the Green "Code" button that appears in the GitHub repository (see Figure 1).

1.2 Getting around with Git

If you decide to go with Option (2), you can ignore this section.

Create a GitHub account. If you don not have already a GitHub account, you can set up a free account by visiting `https://github.com/`. GitHub is a public commercial widely service owned by Microsoft. Gitlab is an alternative solution (`https://about.gitlab.com/`). You can create both public (like the ones in `https://github.com/city-knowledge-graphs`) and private repositories.



ernestojimenezruiz Tests lab 1 a319daf · 32 minutes ago 6 Commits

files	Tests lab 1	32 minutes ago
lab1	Tests lab 1	32 minutes ago
.gitignore	Update .gitignore	32 minutes ago
LICENSE	Initial commit	1 hour ago
README.md	Update README.md	1 hour ago
requirements.txt	Tests lab 1	32 minutes ago

Figure 1: GitHub repository for Python

Git tutorials. The underlying technology for distributed version control/collaboration is called *git*. You can find several tutorial on the Web, for example:

- `https://www.w3schools.com/git/`
- `https://www.youtube.com/watch?v=SWYqp7iY_Tc`
- `https://www.freecodecamp.org/news/learn-the-basics-of-git-in-under-10-minutes-da548267cc91/`

Desktop application. Most tutorials refer to the command line for git. Using the command-line or terminal window is typically natural for macOS and Linux users, but may not be that common for Windows user. GitHub Desktop (`https://desktop.github.com/`) may be an easy solution to manage your git repositories. It is only available for macOS and Windows users. It comes with documentation (`https://docs.github.com/en/desktop`)

Command-line. Check if git is installed in your computer (*e.g.*, type in the command line `git --help`). You can download/install git from `https://git-scm.com/downloads`).

`clone` is the command to clone and download a git repository. First select the folder where you would like to download the codes and then run the command:

- Java: `git clone https://github.com/city-knowledge-graphs/java-2024.git`
- Python: `git clone https://github.com/city-knowledge-graphs/python-2024.git`

To update the local copy with new remote changes (*e.g.*, codes of future laboratory sessions) use the command pull: `git pull` from the respective local repository folders.

2 Python libraries

We will use Python 3.8 (or newer). Check if you have python installed in your computer with (`python --version` or `python3 --version`. Otherwise install from `https://www.python.org/downloads/`

I use Eclipse as Python editor, but Spyder IDE (`https://www.spyder-ide.org/`) and Visual Studio (`https://visualstudio.microsoft.com/`) is typically the choice for many Python developers.

In order to deal with the required libraries and dependencies for each lab and the final coursework it is recommended to use a virtual environment. In the labs I will use `pip` (`https://pip.pypa.io/en/stable/installation/`) and `venv` (`https://docs.python.org/3/library/venv.html`), but there are alternatives like `conda` (`https://docs.conda.io/`).

To set up a new environment:

```
python3 -m venv /home/ernesto/path_to_lab1_venv
```

To activate the environment, depends on the platform (`https://python.land/virtual-environments/virtualenv`). For example:

```
source /home/ernesto/path_to_lab1_venv/bin/activate
```

To deactivate a virtual environment: `deactivate`

The required libraries for each lab session are listed in the requirements.txt file (in GitHub). To install the libraries in the virtual environment:

```
pip3 install -r requirements.txt
```

Windows commands: using the command line in Windows is a bit different with respect to Linux and macOS. These are the commands suggested by one of the students:

- Creation of a virtual environment:

  ```
  py -m venv C:\Users\ernesto\path_to_env
  ```

- Activation of virtual environment (execute "activate" script):

  ```
  C:\Users\ernesto\path_to_env\Scripts\activate
  ```

- Installing pip and making sure pip, setuptools and wheels are up to date using:

  ```
  py -m pip install --upgrade pip setuptools wheel
  ```

- Installing requirements (requirements file in GitHub). Run one of the following:

  ```
  py -m install -r requirements.txt
  py -m pip install -r requirements.txt
  ```

2.1 Key libraries

- Jupyter notebook: Web-based application to execute (python) code together with comments.
  - Docs: `https://jupyter-notebook.readthedocs.io/en/latest/`
  - Example in lab1: `jupyter notebook lab1-notebook.ipynb`
- RDFlib: a library to manage RDF-based knowledge graphs.
  - Docs: `https://rdflib.readthedocs.io/en/stable/`
  - Examples: `https://github.com/RDFLib/rdflib`
- SPARQLWrapper: a python wrapper around a SPARQL service.
  - Docs and examples: `https://github.com/RDFLib/sparqlwrapper`
- Owlready: a package for ontology-oriented programming in Python.
  - Docs: `https://owlready2.readthedocs.io/en/latest/intro.html`

3 Java libraries

The created github project relies on maven (`https://maven.apache.org/install.html`) to deal with the library dependencies (see `pom.xml`). We will use Java 8 or higher `https://www.oracle.com/uk/java/technologies/javase-downloads.html`. I recommend Eclipse IDE as editor (`https://www.eclipse.org/eclipseide/`).

3.1 Key library

- Apache Jena: This API will lead with the Ontology and RDF management, and the SPARQL processing.
  - Documentation: `https://jena.apache.org/getting_started/index.html`
  - Installation. Maven is recommended. Alternative options: `https://jena.apache.org/download/index.cgi`

4 Code overview

The codes provided in both Python and Java (folder/package *lab1*) give an overview about some of the functionalities we will use in the lab sessions. They include methods for:

- Loading and ontology and printing its classes (*loadOntology* script/class).
- Loading and querying a local RDF knowledge graph (*loadRDFGraph* script/-class).
- Querying a remote RDF knowledge graph via a SPARQL endpoint (*queryEndpoint* script/class).

Exercise 1.1: Run the python scripts (or Jupyter notebook), or the java classes without compilation errors (this is the main target of this session). At this stage you do not need to understand all the details, but you can start giving a look to the codes and get familiar with the libraries.

5 Protégé ontology editor

Protégé is an editor of OWL ontologies (`https://protege.stanford.edu/`). We will use the desktop version.

The objective in this session is to just have the first contact with the editor by exploring the pizza ontology. We will explore more about its capabilities next week, once we know a bit more about ontologies and OWL.

The pizza ontology and its tutorial are well-known resources in the Semantic Web community. They were developed for educational purposes by the University of Manchester. The tutorial have been recently updated by Michael DeBellis.

The pizza ontology and the tutorial are found at:

- `https://tinyurl.com/NewPizzaTutorialV2`
- `http://protege.stanford.edu/ontologies/pizza/pizza.owl`

Exercise 1.2: Install Protégé (Desktop version) and load the Pizza ontology (File $\rightarrow$ Open from URL: `https://protege.stanford.edu/ontologies/pizza/pizza.owl`).

Exercise 1.3: Take some time to explore the interface. You may want to check the first steps of the tutorial. Browse the class hierarchy, the property hierarchies and the individuals and note how the ontology describes the domain of pizzas.

Exercise 1.4: Find `Margherita` and see how it is defined as a pizza with only cheese and tomato topping. Protégé uses OWL, and thus formal semantics to define the concepts of the domain. Look at the definition of `VegetarianPizza`. Is a `MargheritaPizza` a vegetarian pizza? Why / why not?