

Introduction to (some) computer tools for Python in TSFS12

Erik Frisk <erik.frisk@liu.se>

Department of Electrical Engineering
Linköping University

Introduction to (some) computer tools

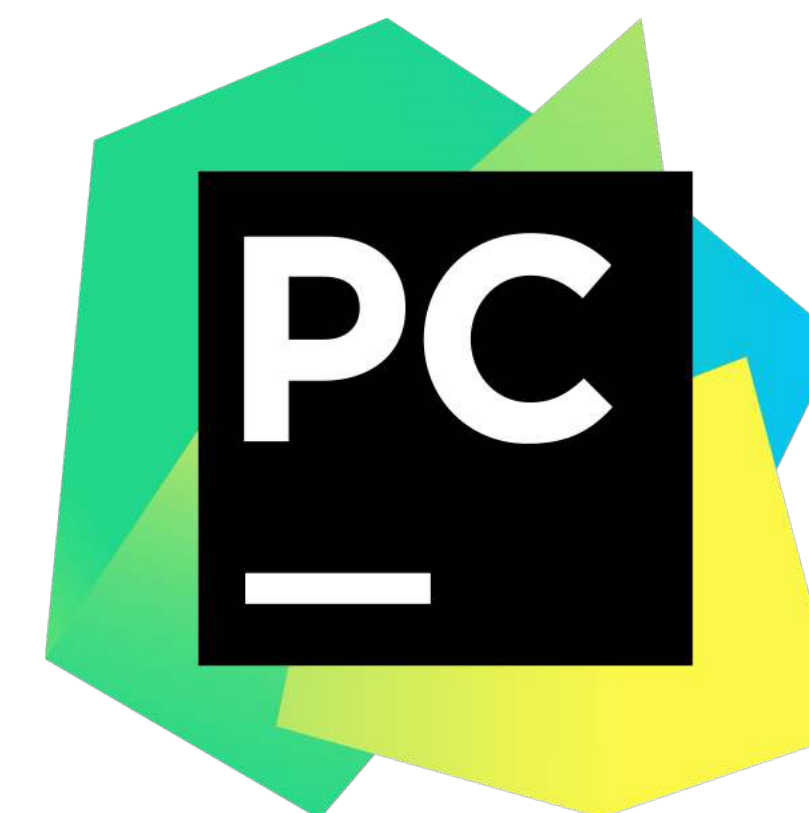
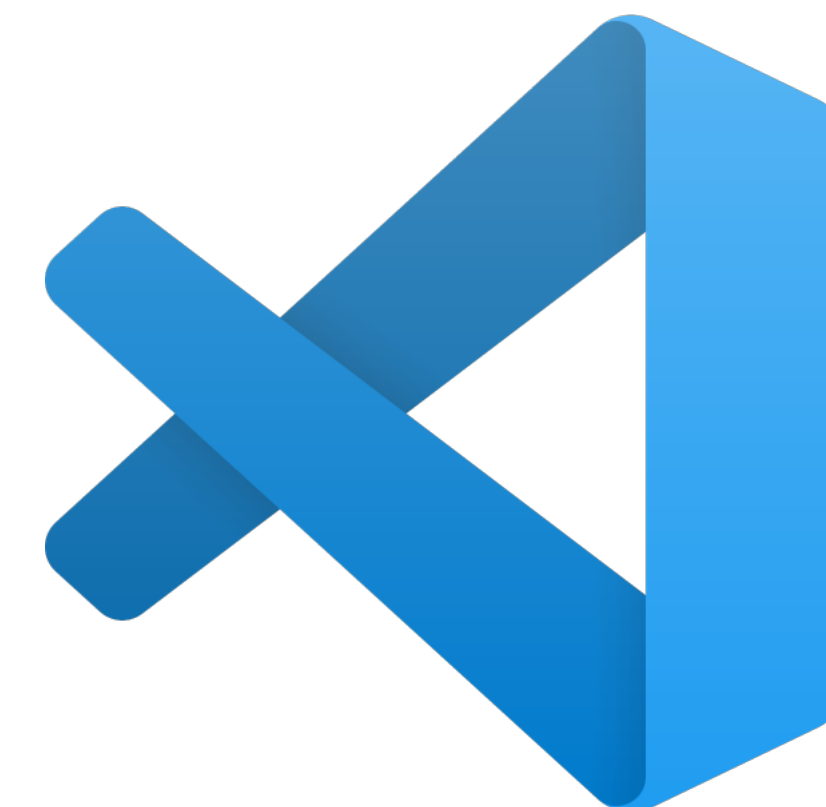
- If you have none, or little, experience in Python before; we support anyone taking the opportunity to learn.
- Bear in mind though that it will probably mean extra work for you. We can help, but this is not a course in Python.
- Experience from previous years:
some are hindered by not being efficient in the computer tools used
- To help, we include this non-mandatory lecture to get you started. If you are efficient and used to working in Python IDE:s; there will be nothing new for you here.

Introduction to (some) computer tools

- **What it is:** Introduction to Jupyter notebooks and Visual Studio Code
 - a brief discussion on when they are suitable
 - illustrate on code for the first hand-in — discrete planning
 - a quick guide to how to start configuring Visual Studio Code for Python
- **What it isn't:** Introduction to Python

Integrated Development Environments

- Three widespread tools (I use) are:
 - Jupyter notebooks (<https://jupyter.org>)
 - Visual Studio Code (<https://code.visualstudio.com>)
 - PyCharm (<https://www.jetbrains.com/pycharm/>)
- Free and available for Windows, Linux, and Mac
- **Jupyter notebooks**
 - Excellent for experimenting, exploring, and light coding
 - **Big +**: Documentation and code together
 - **Big -**: Poor debugging alternatives and a very simple editor
- **Visual Studio Code (VSC)**
 - More programming like environment, full editor capabilities
 - Good debug functionality
- **PyCharm**
 - Same as VSC — matter of preference, more tailored to Python
- There are also others: Sublime text, vim, atom, emacs, ...



Getting started with the hand-in files

Getting started

Get all files from the repository

```
% mkdir tsfs12_work
% cd tsfs12_work
% git clone https://gitlab.liu.se/vehsys/tsfs12.git
% cd tsfs12
% git pull # When you want to get all the latest updates
```

If you don't use git, you can also download a zip-file with the repo from <https://gitlab.liu.se/vehsys/tsfs12>.

Note to M1 Mac users: you might run into a few speed-bumps but you can always talk to me

Create virtual environment and install all packages needed

```
% cd tsfs12_work
% python -m venv env
% source env/bin/activate # On Windows: env\Scripts\activate
% (env) pip install -U pip # Update package manager is always a good idea
% (env) pip install -r requirements.txt # Install all required packages
```

Jupyter Notebooks

— demo on hand-in 1: Discrete Planning in a Structured Road Network

Some useful information

- Project homepage <https://jupyter.org>; for usage information go to Documentation->JupyterLab->"The JupyterLab interface"
<https://jupyterlab.readthedocs.io/en/latest/user/interface.html>

Useful shortcuts (there are more)

- Shift-enter — execute cell
- ESC — leave editing mode
- Return — edit cell
- a — add cell above
- b — add cell below
- x — cut cell
- v — paste cell
- Right-click + new console for notebook — get console
- Tab — tab completion, works for almost everything
- Shift-tab — get method help
- m — make cell Markdown, a documentation cell. See, e.g., (<https://www.markdownguide.org/basic-syntax/>)

Working with imports

- Python caches imports
- If you import your code from a separate file, e.g., `planners.py` like in the demo, run “ipython magic”
`%load_ext autoreload`
`%autoreload 2`
- Then your changes will take immediate effect
- See <https://ipython.org/ipython-doc/3/config/extensions/autoreload.html> for more information.

Visual Studio Code

— demo on hand-in 1: Discrete Planning in a Structured Road Network

Visual Studio Code

- A general purpose code editor that is highly configurable and very popular as a code editor and development environment for Python
- There are *many* extensions available that you can explore
- For Python development, I use the recommended, standard, Microsoft “*Python extension for Visual Studio Code*” (which is a collection of a number of useful extensions)
- Not as important, I also find the following packages useful
 - flake8 for linting the code
 - black for automated formatting

Configuration

- A lot of configuration possibilities
- For basic operation it is simple though:
 1. Install the python extension collection from Microsoft
 2. Enter the preferences and check the setting
“Jupyter: Send Selection To Interactive Window”

Take home messages

Short summary - Jupyter notebooks

- Pros
 - available for Windows, Linux, Mac
 - easy, runs in your browser (you can also run notebooks in Visual Studio Code but I won't cover that here)
 - Supports interactive experimentation & exploration (don't just run the file)
 - Supports documentation and code in same document.
 - Reports for hand-ins can conveniently be written directly in the notebook.
- Cons
 - very basic editing
 - does not really support debugging or more advanced coding

Short summary - Visual Studio Code

- Pros
 - available for Windows, Linux, Mac
 - Capable editor
 - Full debugging capabilities
 - Supports both interactive (Shift-enter) and running scripts
 - Highly extendable, with capable Python extensions easy to install
- Cons
 - Requires more “low-level” tinkering with configurations
 - Does not support documentation in the session
(actually, you can run jupyter notebooks also in VSCode but ...)

Take-home message

- Both Jupyter and Visual Studio Code are excellent tools
- Different objectives; with different pros and cons.
- These exercises, are they programming or experiments?
 - Personally, I spend more time in VSCode and PyCharm than in Jupyter notebooks
- Here: Entirely up to you.
- Submitting reports as notebooks is allowed.
- I will put these slides and link to the recorded video in the course git-repo.