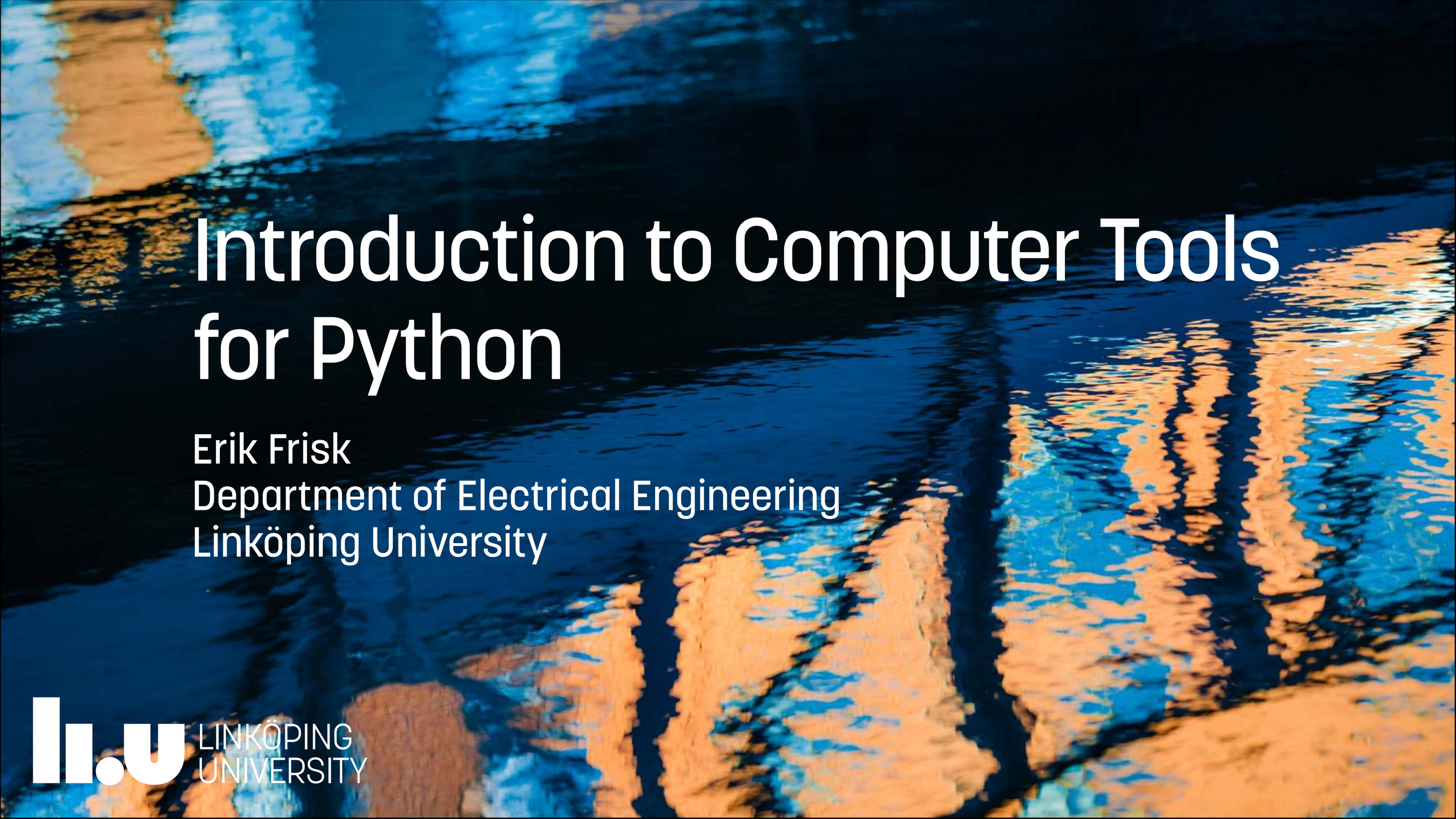


Introduction to Computer Tools for Python

Erik Frisk
Department of Electrical Engineering
Linköping University

LINKÖPING UNIVERSITY

Introduction to (some) computer tools

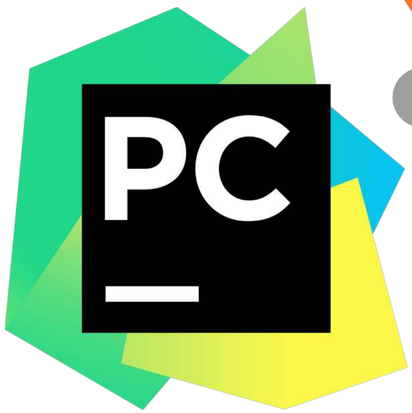
- If you have little or none experience in Python before;
  we support anyone taking the opportunity to learn.
  - Bear in mind though that it will probably mean
    extra work for you.
  - We can help, but this is not a course in Python.
- Experience from previous years:
  some are hindered by not being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

Introduction to (some) computer tools

- What it is: Introduction to working with Python in 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
  - If no previous experience with NumPy — I recommend https://numpy.org/devdocs/user/quickstart.html

Integrated Development Environments

- Three free (Windows, Linux, and Mac) widespread tools (I use) are:
  - Visual Studio Code (https://code.visualstudio.com)
  - Jupyter notebooks (https://jupyter.org)
  - PyCharm (https://www.jetbrains.com/pycharm/)
- **Jupyter notebooks in a web browser**
  - For experimenting, exploring, and light coding
  - Big +: Documentation and code together
  - Big -: Poor debugging alternatives and a very simple editor
- **Visual Studio Code**
  - More programming like environment, full editor capabilities
  - Good debug functionality
  - Good support for Jupyter Notebooks
- **PyCharm**
  - Same as VSCode — matter of preference, directly tailored to Python
- There are also other: Spyder, Sublime text, (neo)vim, atom, emacs, …



Outline - five small modules

1. Working in Visual Studio Code - the basics
   - Scripts and interactive sessions
   - Working with notebooks, in Visual Studio Code and a web browser
   - Basic plotting
2. Working in Visual Studio Code - some next steps
   - Multiple files, import caching, and interactive development
   - Debugging
   - Matplotlib backends
3. Setting up Python on your computer
   - Python version, package management, and virtual environments
4. Setting up Visual Studio Code
   - Extensions and settings
5. Install all packages needed for this course TSFS12

Working in VSCode - the basics

– interactive sessions, scripts, and notebooks

Erik Frisk
Department of Electrical Engineering
Linköping University

LINKÖPING UNIVERSITY

Module content

- How to run Python programs/scripts from VSCode
- Interactive sessions
  - Python files (`.py`)
  - Notebooks in a web browser and in VSCode (`.ipynb`)
- Some simple plotting using matplotlib
- Exemplify using the first handin — Discrete planning

Demo

Take-home messages

- You open a folder, not a file.
  - Gives you a project view of the files
- You can run and debug python scripts (`.py`) directly from the IDE
- Interactive session
  - Add `#%%` to indicate a start of a executable cell (Shift/Control-Enter)
  - Run `%matplotlib` if you want your plots as separate interactive window where you can zoom etc.
- Notebooks
  - Run `% jupyter lab` from a terminal to start a browser session for notebooks (`.ipynb`)
  - VSCode has extensive notebook support, and you get strong debugging capabilities

Working in VSCode - some next steps

– debugging, plotting backends, multiple files

Erik Frisk
Department of Electrical Engineering
Linköping University

LINKÖPING UNIVERSITY

Module content

- Multiple files, import caching, and interactive development
- Debugging python scripts and notebooks
- Matplotlib - a good plotting and visualization library
  - If you want interactive plots (zoom, rotate, …), you need to know (a little) about backends
  - How to save plot to a high-quality file

Demo

Take-home messages

- Utilize debugging functionality — it will save you hours
- If you work interactively with multiple files:
  - your imports are cached and changes in imported files will not be directly visible
  - Use the IPython commands (https://ipython.org/ipython-doc/3/config/extensions/autoreload.html)
    ```
    %load_ext autoreload
    %autoreload 2
    ```
- If you want interactive plots:
  - Use the IPython command
    ```
    %matplotlib
    ```
  - Set a default Matplotlib backend (https://matplotlib.org/stable/users/explain/backends.html), e.g., TKAgg, in your matplotlibrc file (https://matplotlib.org/stable/tutorials/introductory/customizing.html#the-matplotlibrc-file)
- Use `fig.savefig("file.pdf/png")` to save to suitable format

Setting up Python on your computer

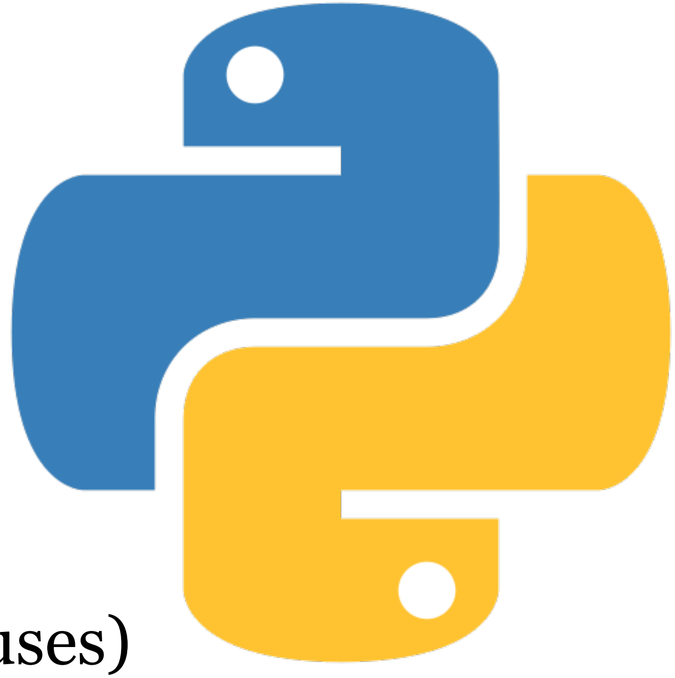
– versions, packages, and virtual environments

Erik Frisk
Department of Electrical Engineering
Linköping University

LINKÖPING UNIVERSITY

Which Python version should you use?

- There is a new version coming out once a year
- For most uses, exact version is not that important
  - Not too old
  - Could be good to stay away from the bleeding edge since package managers might not have updated yet
- There are two main distributions
  - https://www.python.org/ (my preference for most uses)
  - https://www.anaconda.com/download
- This presentation will use python.org, but similar can be done in Anaconda.



Python - Virtual Environments

- Python is a small language supported by *many* freely available packages
  - On the main package site, https://pypi.org/, there are currently 470,125 projects
- **Situation**: Running the system python installation and installing all packages globally
  - Different projects might need different versions of a package
  - You will need administrator rights, system pollution, and complex to go back to original system state
  - Conflicting packages
- **A solution**: Virtual environments
  Python virtual environments give you the ability to *isolate* your Python development projects *from your system installed Python*.
  A Python virtual environment is a *folder* with everything you need to run a *lightweight* and isolated Python environment .

Virtual environments and the pip package manager

- I always run using a virtual environment
- The main package manager is called pip
- I will demo
    1. How to create one (and delete all traces of the installation)
    2. How to install packages and work with the package manager
- One suggestion: Create a virtual environment for this course
- See https://docs.python.org/3/library/venv.html for further documentation

Common packages that will get you started

- Ipython - *Interactive python environments* (https://ipython.org/)
- NumPy - *Fundamental package for numerical computing* (https://numpy.org/)
- SciPy - *Fundamental package for scientific computing* (https://scipy.org/)
- Matplotlib - *Visualization and plotting* (https://matplotlib.org/)
- Jupyter - *Notebooks* (https://jupyter.org/)
- Scikit-learn - *Machine learning* (https://scikit-learn.org/)
- PyTorch - *Deep learning* (https://pytorch.org/)
- Pandas - *Fast and powerful data analysis tool* (https://pandas.pydata.org/)
- ...

Demo

Take-home messages

- Use virtual environments for everything!
- Create a virtual environment in folder `<name>` [You only do it once!]
  ```
  % python-3.11 -m venv <name>
  ```
- Activate a virtual environment [Everytime! — or let your IDE do this for you]
  ```
  % source <name>/bin/activate # Linux/Mac
  % <name>\Scripts\activate  # Windows
  ```
- Install a package
  ```
  % pip install matplotlib
  ```
- Update a package
  ```
  % pip install -U matplotlib
  ```
- Remove a package
  ```
  % pip uninstall matplotlib
  ```
- List installed packages
  ```
  % pip list
  ```

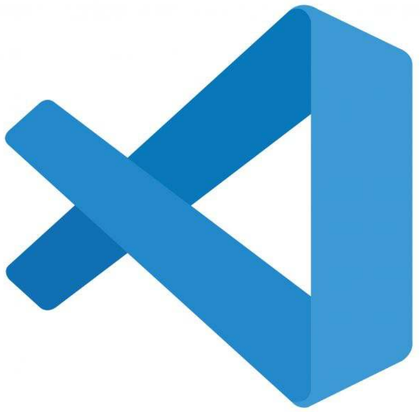
Setting up Visual Studio Code for Python

– extensions and settings

Erik Frisk
Department of Electrical Engineering
Linköping University

LINKÖPING UNIVERSITY

Setting up Visual Studio Code for Python Development



1. Install application (Windows/Mac/Linux) from https://code.visualstudio.com/
2. Install the extensions you like, a good start are the 4 extensions recommended from https://code.visualstudio.com/docs/languages/python
   - Python - main extension
   - Pylance - extended language support in the editor (installed automatically with python extension
   - Jupyter - Jupyter notebook support
   - IntelliCode - some AI-assisted development features
3. A couple of settings — I will demo those

Demo

Take-home messages

- If you want to use VSCode as a code editor for Python (scripts or Notebooks), to get started follow a few simple steps
- Install from https://code.visualstudio.com/
- Install basic extensions (4 suggested, there are more if you want to explore)
- My (subjective) suggested settings:
  - Activate **Jupyter › Interactive Window › Text Editor: Execute Selection**
  - If you have your Python virtual environments in one location, e.g., in a folder your home directory (~/pyenv), set **Python: Venv Path** and then VSCode will find them directly.
  - Disable **Files: Hot Exit** - has tricked me many times

Install all packages for TSFS12

Erik Frisk
Department of Electrical Engineering
Linköping University

LINKÖPING UNIVERSITY

Demo

Take-home messages

- There is a file ‘requirements.txt’ in the course git-repo (folder Handin_Exercises) that lists all packages needed to solve the hand-in exercises
- To install all packages in your virtual environment, run

```
% pip install -r requirements.txt
```