Preparations for Python course

Please follow the instructions below to set up your computer and the python environment.

1. Required software for the Python course and setup instructions

We require the following software to be installed on your computer:

- VS Code (with Python and Jupyter extensions)
- Git
- GitHub account
- `uv` Python environment manager
- Further instructions for setting up the python environment are provided during the course.

Please follow the instructions below to install the required software on your computer. The specific steps may differ between operating systems.

1.1. Setup VS Code

Download and Install VS Code

- Go to the VS Code download page and download the installer for your operating system.
- Follow the installation instructions specific to your operating system.

Install Python and Jupyter Extensions for VS Code

- Open VS Code.
- Go to the Extensions view by clicking on the Extensions icon in the Activity Bar on the side of the window or by pressing `Ctrl+Shift+X` (Windows) or `Cmd+Shift+X` (Mac).
- Search for "Python" and install the official Python extensions by Microsoft.
- Similarly, install the "Jupyter" extension by Microsoft.

Install additional, useful extensions

- "Data Wrangler" (Microsoft) - search for: `ms-toolsai.datawrangler`
- "Github Copilot" (GitHub) - search for: `github.copilot`
- "Github Copilot Chat" (GitHub) - search for: `github.copilot-chat`

- "Git Graph" (mhutchie) - search for: `mhutchie.git-graph`
- "Ruff" (Astral Software) - search for: `charliermarsh.ruff`

1.2. Git Setup

Install Git on your computer

- **Windows**: Download and install Git for Windows from git-scm.com.
- **Mac**: Follow the instructions on the git-scm.com website to install Git for Mac.
- **Linux / WSL (Windows Subsystem Linux):** Use your package manager to install Git. For example, on Ubuntu, you can run:

```
sudo apt-get install git
```

Configure Git (only required once per machine and user)

Configure Git (in system terminal / PowerShell / git bash, or VS Code Terminal):

```
git config --global user.name "Your Name"
git config --global user.email "you@example.com"
```

> 💡 **Tip:** If you want your initial branch to be named `main` instead of the default (`master` in older Git versions), you can add:
>
> `bash`
> `git config --global init.defaultBranch main`
>
> This ensures consistency with platforms like GitHub, which default to `main`.

Check your setup:

```
git config --list
```

1.3 GitHub Account Setup

1. Go to https://github.com/
2. Click **Sign up**.
3. Enter a username, email, and password.
4. Verify your email address.
5. Choose the free plan (sufficient for this course).

6. Apply for an education account (if you have a .edu email address): navigate to the GitHub Education page and follow the instructions. This will allow you to access additional benefits, such as a free GitHub Copilot subscription.

You now have access to create and manage repositories on GitHub.

> **Tip** - You may want to setup a a **personal access** token in GitHub (if you're pushing to GitHub and prompted for credentials) - See https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token for more details.

1.4. Install `uv`

This section gives the minimal, standalone steps to install the `uv` Python environment manager on your machine.

Windows

The instructions for installing Windows depend on whether you have **administrative rights** on your machine.

Select the appropriate instruction manual from below:

- Option A: with **administrative rights** or the pdf (setup-uv-windows-admin.pdf)
- Option B: without **administrative rights** or the pdf (setup-uv-windows-no-admin.pdf)

macOS / Linux / WSL

- Install using the shell installer (works with `curl` or `wget`):

```
curl -LsSf https://astral.sh/uv/install.sh | sh
# or, if you prefer wget:
# wget -qO- https://astral.sh/uv/install.sh | sh
```

- After installation, ensure `~/.local/bin` is on your `PATH` (the installer can update your shell):

```
export PATH="$HOME/.local/bin:$PATH"
# Add the line above to ~/.bashrc, ~/.profile, or ~/.zshrc to persist.
```

- Open a new terminal session. Then:

```
uv --version
```

If the command prints a version, the installation succeeded.

2. Clone the repository

2.1 Choose where to store the repository on your machine

> **IMPORTANT Guidelines for choosing a project directory**: 1. you should store python projects on a local drive (**not on a network drive**) and avoid using special characters or spaces in the directory name.
>
> 1. Furthermore, things will run more smoothly, if you pick a folder that is **not synchronized with OneDrive** (e.g. usually `C:\Users\<you>\projects`. In contrast, `C:\Users\<you>\Documents\projects` might cause issues as the `Documents`, `Desktop` and similar folders are usually synchronized with OneDrive).

Open a terminal and navigate to the directory (you will create a projects folder first, if necessary):

Windows:

```
mkdir $HOME\projects -f # creates the directory if not existing
cd $HOME\projects
```

MacOS/Linux:

```
mkdir -p ~/projects # creates the directory if not existing
cd ~/projects
```

2.2 Initial clone

Open a terminal (or PowerShell on Windows) and run the following command to clone the repository:

```
git clone https://github.com/keller-tobias/{{REPO-NAME}}
```

2.3 Updating the repository

Because this repository is updated in a specific way, you will receive the following error if you try to use the `git pull --no-rebase` command to update it:

```
fatal: refusing to merge unrelated histories
```

Therefore, I recommend you clone the repository again into a new directory. This is preferred because it will keep your local changes (and possibly notes) intact.

Alternatively, you could use the following commands to reset your local branch to match the remote branch, using a `git reset` but this will discard your local changes:

```
# 1) Fetch everything from remote and prune stale refs/tags
git fetch origin --prune --prune-tags --tags --force
# 2) Hard-reset your local branch to the remote default branch
git switch -C main origin/main      # creates/resets local main to track origin/main
git reset --hard origin/main        # guarantees pointers match exactly
# 3) Optional: Remove untracked/ignored files so the tree is identical to a fresh clone
# git clean -xdf
```

3. Download Raw Data

Download and extract the raw data files into the `data/raw` directory. We use the terminal for this step. Open a terminal (or PowerShell on Windows) in the project directory (where you cloned the repository). If you opened the project folder in VS Code, you can create a new terminal in VS Code and it will automatically open in the project directory.

For Windows (PowerShell):

```
# cd path/to/{{REPO-NAME}} # if not already in the project directory
mkdir data\raw -Force
Invoke-WebRequest -Uri "http://www.tobiaskeller.net/data/data_raw.zip" -OutFile "data_ra
Expand-Archive -Path "data_raw.zip" -DestinationPath "data\raw" -Force
Remove-Item "data_raw.zip"
```

For MacOS or Linux:

You may need to install `curl` and `unzip` first (if not already installed):

```
# install curl and unzip
sudo apt-get update && sudo apt-get install curl unzip
```

Then run the following commands:

```
# cd path/to/{{REPO-NAME}} # if not already in the project directory
mkdir -p data/raw
curl -L http://www.tobiaskeller.net/data/data_raw.zip -o data_raw.zip
```

```
unzip data_raw.zip -d data/raw
rm data_raw.zip
```

4. Setting up the python environment

It is assumed that you have already installed `uv` as described in the installation instructions specifically for your operating system.

4.1. Open a terminal and navigate to the project directory.

> Note: If you use VS Code and if you have already opened the project folder, creating a new terminal in VS Code will automatically open it in the project directory.

Otherwise: use `cd` to change into the project directory:

```
# Example (replace with the real path on your machine):
cd 'path/to/{{REPO-NAME}}'
```

4.2. Create a new virtual environment with python 3.13 (will be downloaded automatically if not installed).

> Note: this step will create a new virtual environment inside the project folder (named ".venv"):

```
uv python pin 3.13
uv venv --python=3.13
```

4.3. Activate the virtual environment.

> **Note:** If you are using conda, you should first deactivate any active conda environment (use `conda deactivate`).

Activating the virtual environment (venv) differs between operating systems:

On Windows (PowerShell):

```
.\.venv\Scripts\activate
```

If activation fails with a scripts-disabled error, set the execution policy for **this session only** and retry activation:

```
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
.\.venv\Scripts\activate
```

(For a persistent setting per user, use `Set-ExecutionPolicy -Scope CurrentUser -ExecutionPolicy RemoteSigned`.)

On MacOS / Linux:

```
source .venv/bin/activate
```

4.4 Install the project dependencies.

```
uv sync
```

5. Optimizing your workspace for remote course attendance

If you are attending the course remotely, consider the following tips to optimize your workspace:

🖥️ Use a second monitor

It may be helpful to work with a second monitor so that you can watch the instructions while working on your main screen.

Meet up with other participants

If you have a meeting room with projector capabilities, consider meeting up with other participants to follow along with the instructions together.

📷 Turn on your camera:

The course is intended to be as interactive as possible. Please turn on your camera during the sessions to enhance engagement and collaboration.