Biophysics 210: Biological Light Microscopy
Section 10: Image analysis using Python
Tuesday 1:00 – 2.30 pm GH N114

Setting Up Your Environment for Python + Image Analysis

This guide is written for

- People who have never used the command line before
- People with some coding experience
- Advanced users who just want the shortest path to a working setup

The goal is not to explain every detail immediately. The goal is to get everyone to a working environment without breaking their computer.

Part A: Getting started with your Mac or PC

Part 1: What you are installing (Big Picture)

You will install/discover four things:

Tool	What it does	Why you need it
Terminal / Command Line	A text-based way to interact with your computer	Used to run Python and developer tools
Git	Version control system	Used to download and manage code projects
Python	Programming language	Used for image analysis and scientific computing
uv	Python environment manager	Keeps project dependencies isolated and reproducible

Optional:

Tool	Why you might want it
GitHub account	Useful for collaboration and backups
AI coding tools (Claude Code, Codex, ChatGPT)	They write code much faster than you can, and allow you to create image analysis code even with little or no coding experience

Part 2: Install the Terminal / Command Line

macOS: Your command line program is called Terminal. To open Terminal

> Press Command + Space --> Type Terminal --> Press Enter

Recommended: Drag Terminal into your Dock so you can find it easily later.

Windows: Windows has several command line tools (avoid using the old cmd.exe prompt unless instructed). Recommended options:

- PowerShell (installed already)
  - To open PowerShell:

    > Press the Windows key --> Type PowerShell --> Open "Windows PowerShell"

- Git Bash (installed automatically with Git, see below)

Part 3: Install Git

- Git is used to download (clone) and manage code projects.
- To install, go to https://git-scm.com/install/ . Download and install the version for your operating system.

Windows note: Installing Git also installs Git Bash, which many people prefer over PowerShell for scientific Python work.

Part 4: Install Python + uv

Why do virtual environments matter? Different Python projects often require different package versions. Without isolated environments:

- One project can accidentally break another

- Updating packages may cause old code to stop working
- Debugging becomes extremely difficult

A virtual environment keeps each project self-contained. So why are we using uv? There are multiple Python environment managers (e.g. `venv, conda, poetry, uv`). We will use uv because it is:

- Fast
- Simple
- Reliable
- Increasingly common in scientific Python workflows

Installing Python and uv

Follow the official instructions here https://docs.astral.sh/uv/getting-started/installation/

After installation, test that it worked. Open Terminal (or PowerShell / Git Bash) and type:

```
uv --version
```

If you see a version number, installation was successful!

Part 5: GitHub Account (optional but recommended)

GitHub is a website for hosting and collaborating code.

- Create an account if you do not already have one: https://github.com
- Students may also qualify for free developer benefits. https://education.github.com/
- At the very least this can serve as a backup for your code, but more importantly, this will allow you to collaborate with others on code.

Part 6: AI Coding Tools (optional, but extremely useful and highly recommended)

AI tools can, explain unfamiliar code, generate code, help debug errors, and search across your project files.

Tool	Notes
ChatGPT	Good general-purpose assistant
Claude Code	Strong terminal/project integration
Codex CLI	OpenAI coding agent

Important:

- AI tools are useful assistants, not replacements for understanding your workflow
- Avoid uploading protected or sensitive data unless approved by UCSF policy
- Enterprise restrictions may apply

You can use a web-based AI interface (such as ChatGPT), however, having a coding agent run in your terminal is much easier and more powerful as it can inspect the local files and use the information in them.

Claude code is amazing but currently costs $20 per month. Codex (from OpenAI) works extremely well too and currently is free.

Install codex from this link: https://developers.openai.com/codex/cli. Please note that UCSF Enterprise ChatGPT does not make Codex available (because of HIPA concerns), so you may need to use a private email to sign up with ChatGPT for free use of Codex.

Part B: Your first project setup

You should have all neccessary tools. Now we will:

1. Create a projects folder
2. Download (clone) a GitHub repository
3. Create a Python environment
4. Run the software

Step 1: Create a Projects Folder

Open Terminal and type

```
mkdir -p ~/projects
cd ~/projects
```

What these commands do?

Command	Meaning
`mkdir`	Make a new directory (folder)
`-p`	Do nothing if the folder already exists
`~/projects`	A folder named `projects` in your home directory
`cd`	Change directory

You are now inside your projects folder.

Step 2: Download (clone) the Repository

- Run the following code

```
git clone https://github.com/nicost/cellquant.git
cd cellquant
```

Command	Meaning
`git clone`	Download (clone) a copy of the repository
`cd cellquant`	Move into the downloaded folder

Step 3: Switch to the Correct Branch

- Run the following code

```
git checkout BP210
```

What is a branch? A branch is a version of the project. Think of it as a parallel workspace where development happens.

Step 4: Create the Python Environment

- Run the following code

```
uv sync
```

This command creates a virtual environment, downloads all required Python packages, and installs compatible dependency versions. This may take several minutes the first time.

Step 5: Run the Project

- Run the following code

```
uv run cellquant.py
```

This launches Python inside the project environment and runs the script. If everything worked, congratulations! You now have a functioning Python environment.

Part C: TroubleShooting

"Command not found"

Usually means:

- The tool was not installed
- Terminal needs to be restarted
- Installation path was not added correctly

Try:

- Closing and reopening Terminal
- Reinstalling the tool
- Checking the installation instructions again

Git clone fails

Possible causes:

- Internet issue
- Git not installed correctly
- Corporate firewall restrictions

Test Git:

```
git --version
```

uv sync fails

Usually caused by:

- Network problems
- Missing Python installation
- Package incompatibilities

Test uv:

```
uv --version
```

Part D: Useful Commands Cheat Sheet

Command	Purpose
`Pwd`	Show current directory
`Ls`	List files
`cd foldername`	Enter a folder
`cd ..`	Go up one folder
`mkdir name`	Create folder
`git status`	Show Git changes
`uv sync`	Install dependencies
`uv run script.py`	Run a Python script

Part E: What you actually need to understand right now

You do not need to understand everything before starting!

At this stage, the important concepts are:

1. The terminal is just another way to interact with your computer
2. How to use git to download (clone) a repository to your computer
3. Python packages can conflict with each other
4. Virtual environments isolate projects safely
5. uv automates most environment setup

Everything else becomes easier after you successfully run a real project.