計算物理概論

Introduction to Computational Physics (PHYS290000)

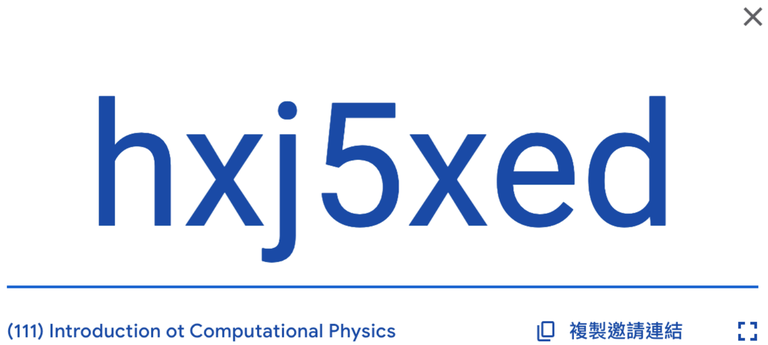
Lecture 1: Introduction

Instructor: 潘國全

kuochuan.pan@gapp.nthu.edu.tw

Last week

1. Introduction (google classroom)
2. Course overview
3. Computer Architecture
4. Floating point numbers and memory usages



Today’s plan

1. Programming environment
2. Install python via Anaconda 3
3. Install VS Code
4. Virtual Python Environment with Conda
5. Hello world python program and Jupyter notebook
6. Version Control with Git & Github

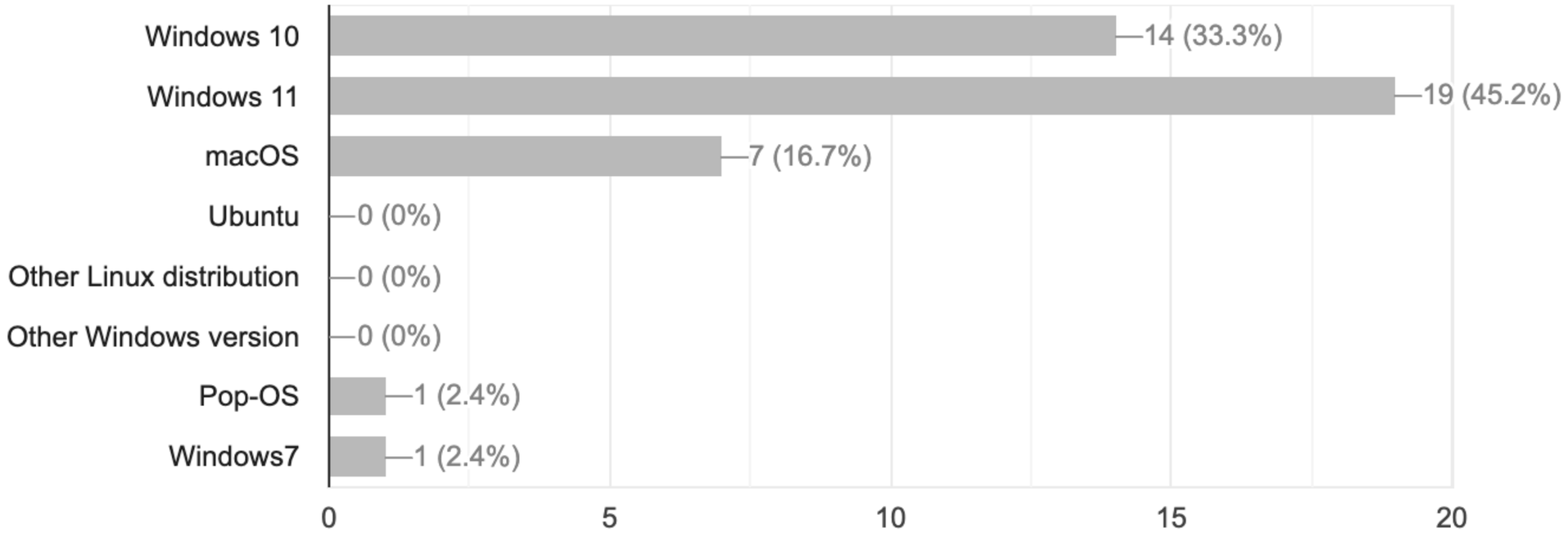
Programing Environment

Operating Systems

What operating system is used in your laptop?

複製

答對次數：0 (作答總數：42)



Operating Systems

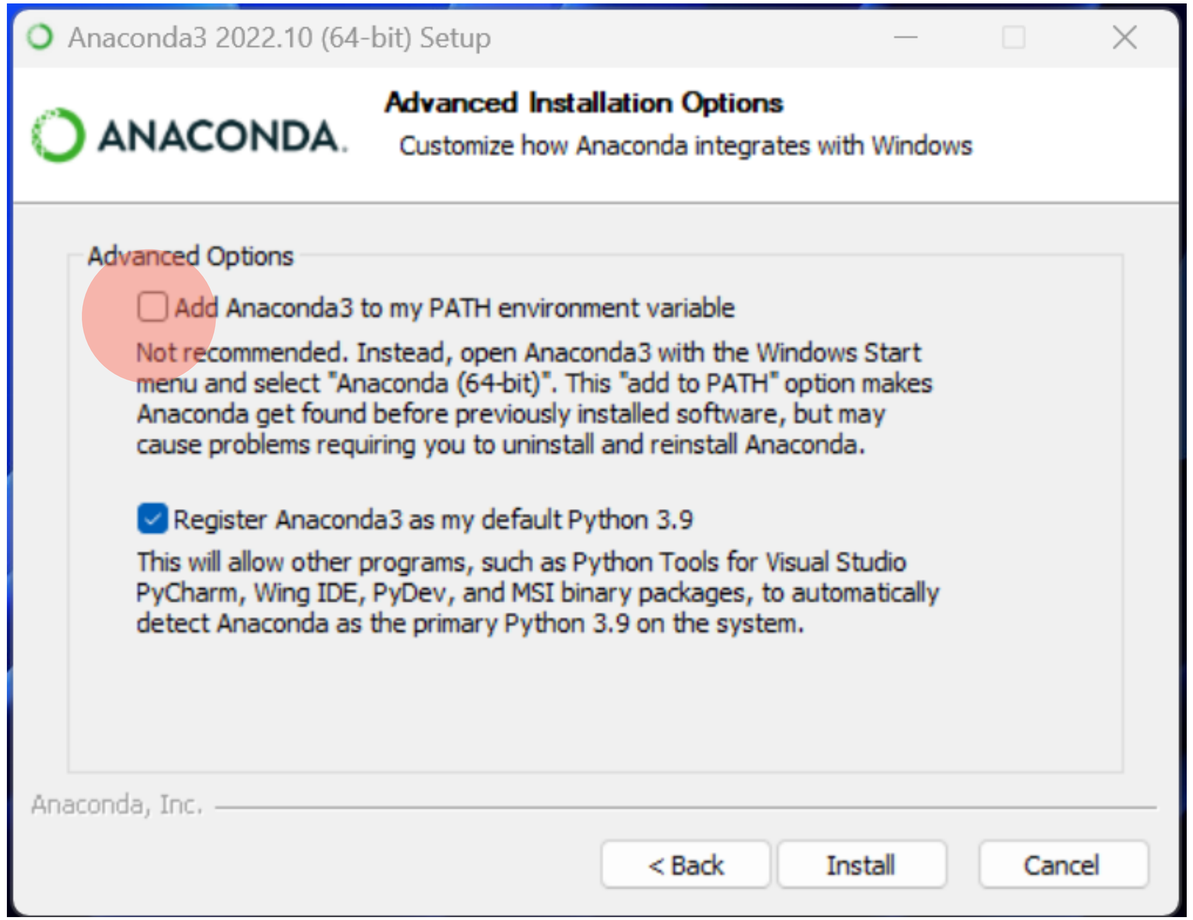
1. Windows 7/10/11 [see page 7]
2. Windows Subsystem for Linux 2 (WSL2) [see page 14]
3. Linux (Ubuntu) *(recommended)* [see page 16]
4. macOS *(recommended)* [see page 16]

- Why Linux? Or Unix-like systems?
- Many open source projects. Many important library/packages for both academia and industry are developed natively for Linux (better community support).
- Native support for SSH, which could help you manage your server easier.
- Linux supports almost all of the major programming languages

Windows Users

Install Python via Anaconda

1. Follow the instruction here: https://docs.anaconda.com/anaconda/install/windows/ to install Anaconda
2. Note: You could click “add Andaconda to my PATH environment variable” and “Register Anaconda 3 as my default Python 3.9”



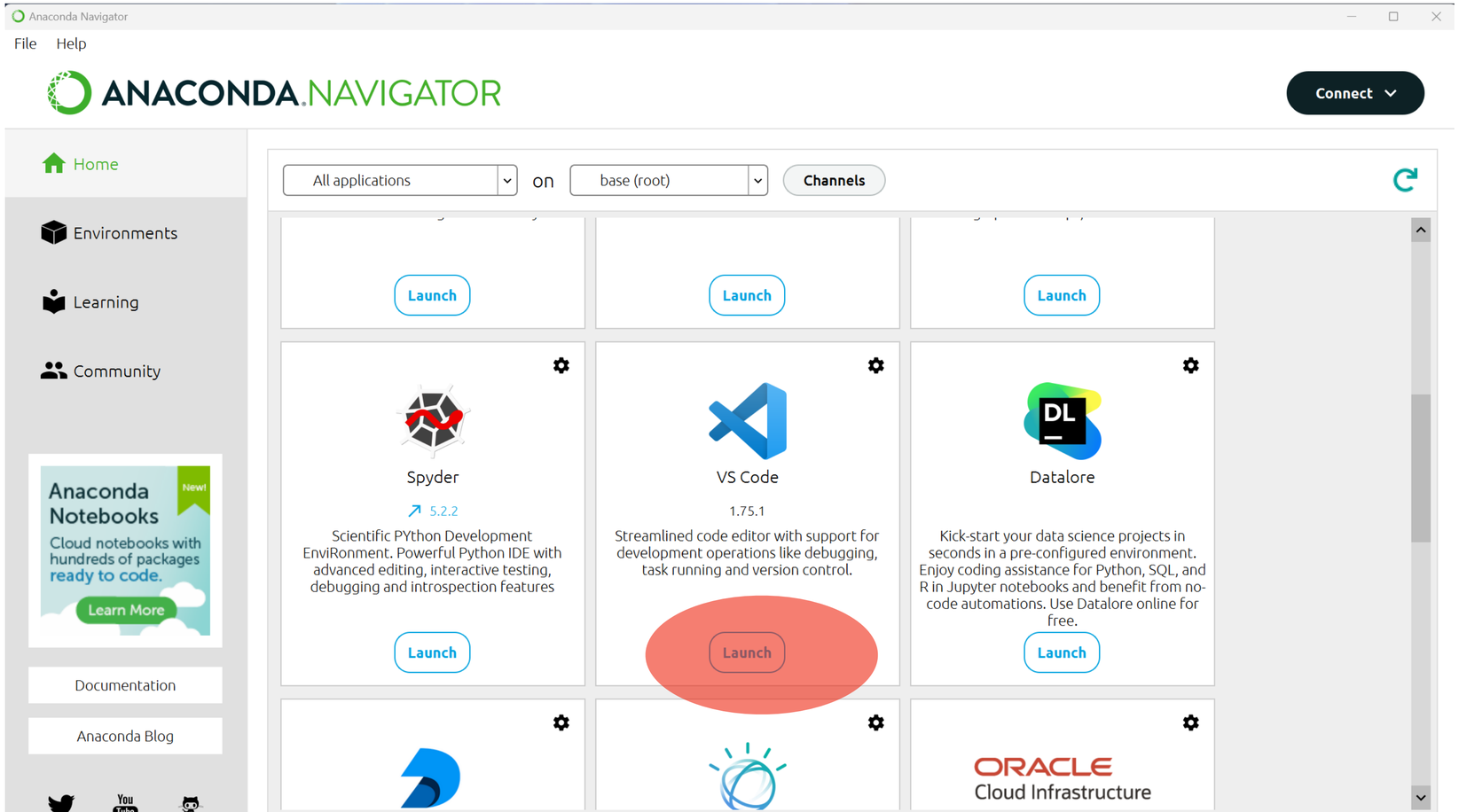
Create a Virtual environment using Conda

1. Press the “Win” key
2. Type “Anaconda Prompt (anaconda3)” (it will launch a terminal window similar to power shell)
3. Type “conda update conda” to check if Conda is up to date (type “y” if there are any updates)
4. Create a virtual environment with `conda create --name comphys python=3.10`
5. Type “y” to confirm
6. Activate the environment by `conda activate comphys`
7. Deactivate the environment by `conda deactivate`

Create a project folder

Install / Launch the VS Code

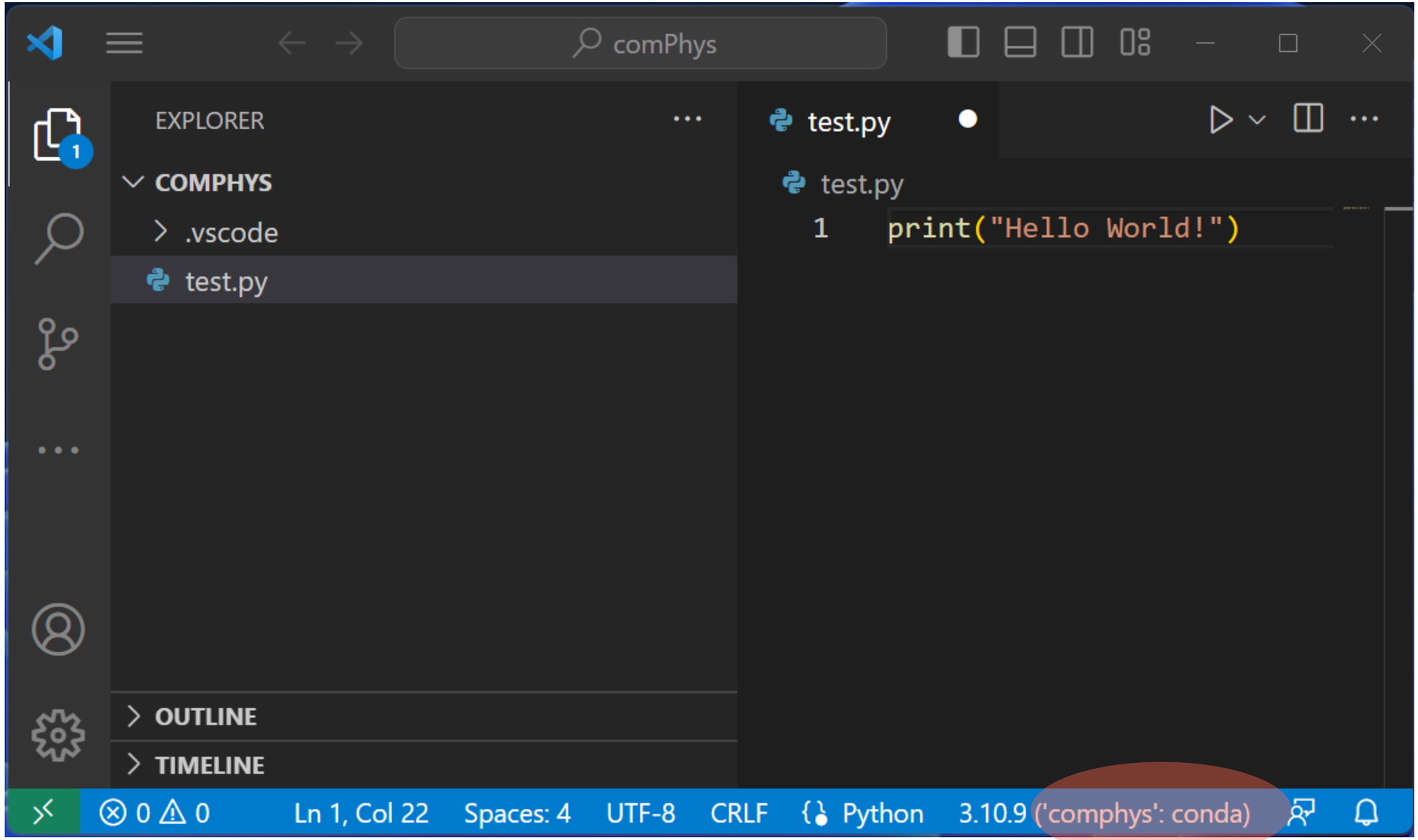
1. Click “Win” key and search for “Anaconda Navigator”
2. Install / Launch the VS Code by click the “Install” or “launch” icon.



Install python extensions in VS Code

1. Open your project folder
2. Create a new file called “test.py”
3. Select the “comphys” python environment
4. In the extension tab, search for “python” and “jupyter” and then install them

If successful, …



WSL2 Users

Install WSL2 / VS Code / Anaconda

1.Follow this tutorial to install WSL2 and VS Code https://ubuntu.com/tutorials/working-with-visual-studio-code-on-ubuntu-on-wsl2#1-overview

2. In the VS Code, install additional extensions: python, Jupyter

3. See page 20 to install Anaconda

Linux / Mac Users

Install Python via Anaconda

1. Install Anaconda here https://www.anaconda.com/products/distribution

    Mac: install the .pkg file

    Linux / WSL2: run the install script $ bash Anaconda-3-x.x.-Linux-x86_64.sh

2. Mac/ Linux: type “yes” to add the python path in your .bashrc (.bash_profile) file

3. Install VS Code here https://code.visualstudio.com/

Create a virtual environment with conda

1. Open a terminal with the “Terminal” app or iTerm2
2. Check if you could find conda with “conda -- version”
3. Create a virtual python environment with conda

```
conda create --name comphys python=3.10
```

Activate or deactivate the comphys env by

```
conda activate comphys
```

```
conda deactivate
```

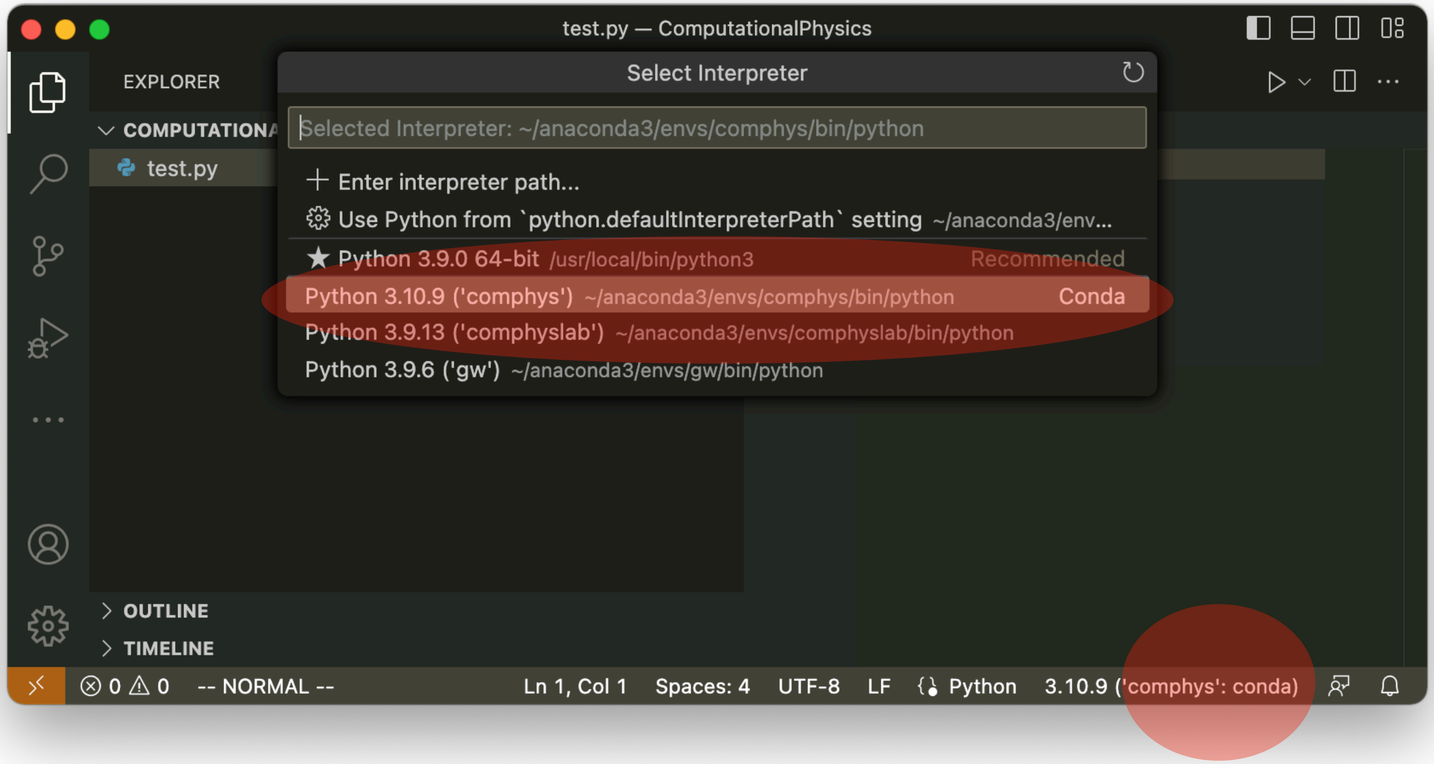
You should see a prompt starting with (comphys) if activated

Project folder

1.Create and Open a project folder in VS Code

2. Create a new file called “test.py” in the project folder

3. Select your python virtual environment

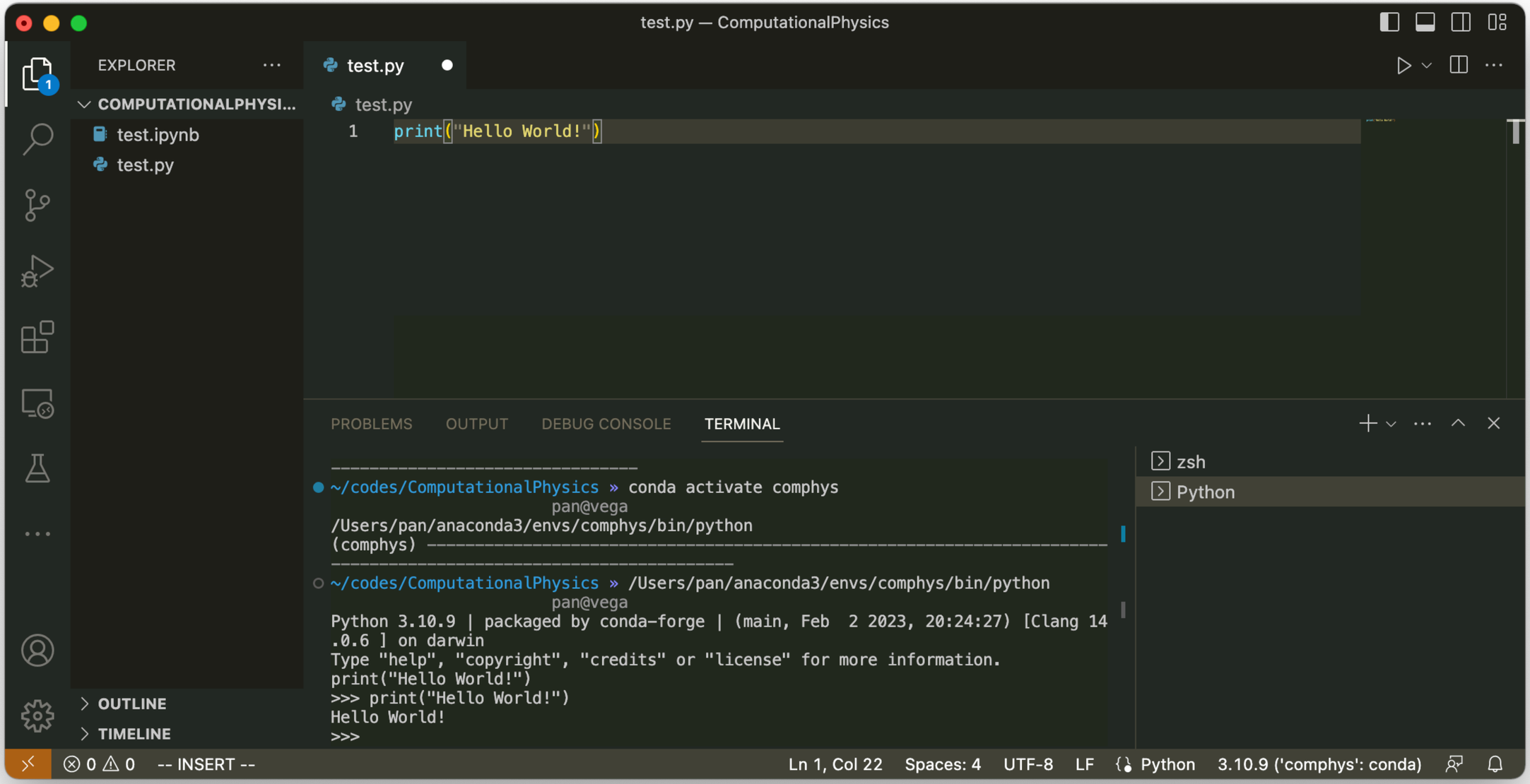


Hello world python and Jupyter notebook

Hello world python!

1. Open the file “test.py” in VS Code
2. Select your python env
3. Type >>> print(“Hello world!”)
4. Click the icon to run the code (or “shift + enter”)

Hello world python!



Install jupyter notebook (Win/WSL/Linux/Mac)

1.Open a terminal

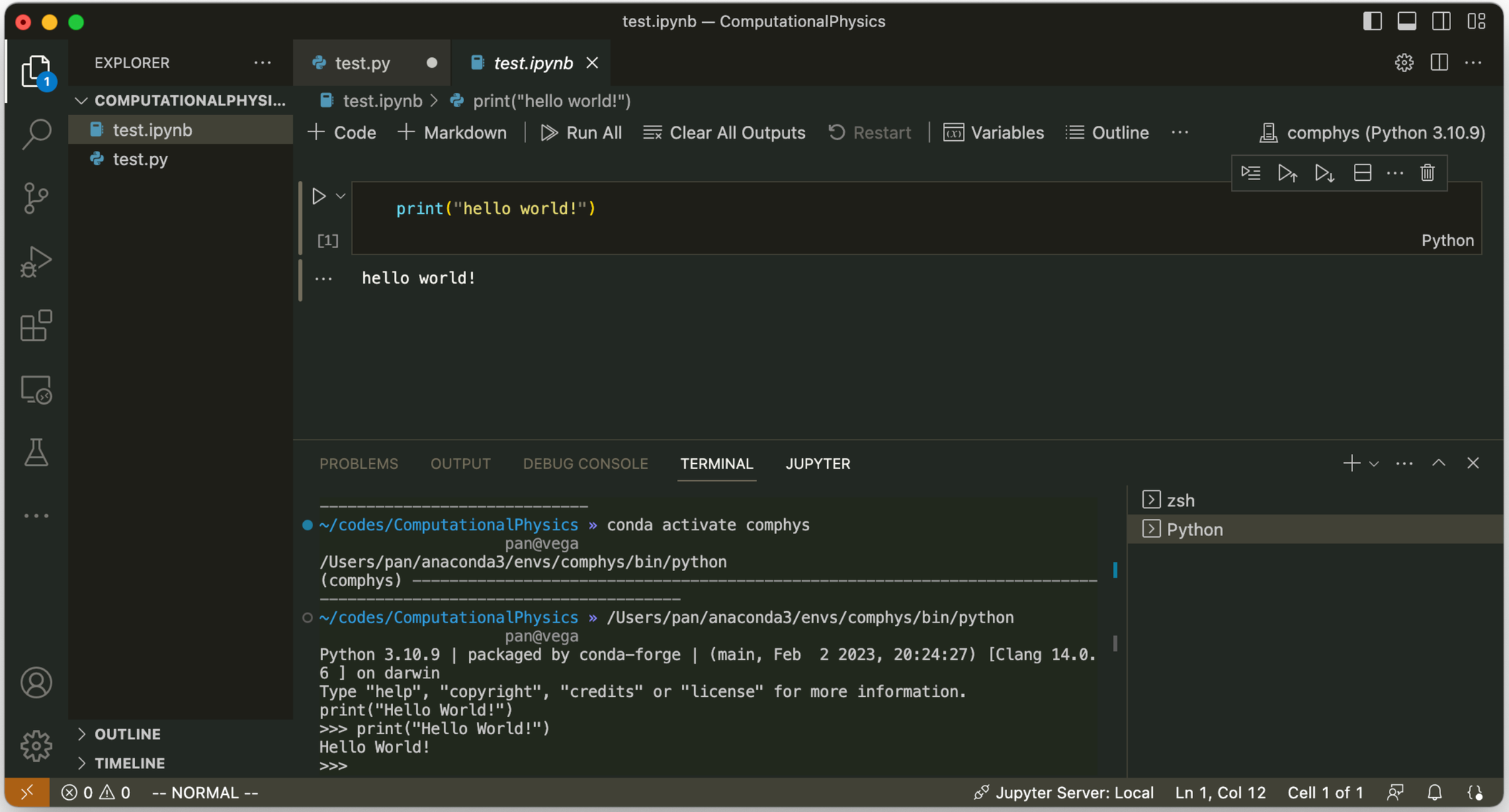
2. conda activate comphys (activate the virtual env)

3. conda install -c condo-forge jupyterlab

4. Create a new file called “test.ipynb”

5. Type the same hello world code in a cell

Hello world python in a notebook !



Summary

1.No matter which OS you are using (Windows / WSL2/ Linux/ Mac),

2. Able to run hello world python and jupyter notebook codes with the “comphys” virtual environment under conda (python 3.10)

3. Created a project folder called “comphys” or “computationalPhysics”

Command Line Interface / Shell

A Terminal

1. You could either use a “Terminal” app or click “Terminal” —> “New Terminal” in the VS Code to enter the command line interface.
2. Graphic User Interface vs. Command Line Interface
3. Common shells
    - Windows: Power Shell, DOS
    - Linux / Mac / Unix: Bash, Csh, Zsh, …
4. What is the shell?

Navigating in the shell

1. In Windows, we use “My Computer”, “檔案總管” to navigate around folders. In Mac/Linux, similar navigation can be done with the “Finder”.
2. We could use commands to navigate folders as well.

Home (家目錄)

Windows: `/Users/Pan`

Linux: `/home/pan`

Mac: `/User/pan`

Root `/`

Navigating in the shell

The command "pwd"

`pwd` 顯示目前的路徑

目前的路徑 `./`

絕對路徑: `/folder1/folder2/folder3`

相對路徑: `./folder4/folder5/`

Navigating in the shell

The command "cd" (“change directory”)

`cd`：回到家目錄

`cd ..`： 回到上一層資料夾

`cd ...`：回到上兩層資料夾 (= `cd ../../`)

`cd -`：回到前一個資料夾

`cd folder_name`：移動到資料夾 `folder_name` (假設存在)

`cd a_path`：移動到路徑 `a_path` (可以是絕對或相對路徑)

`cd ~` or `cd ~/Downloads`：移動到家目錄或相對於家目錄的路徑

Navigating in the shell

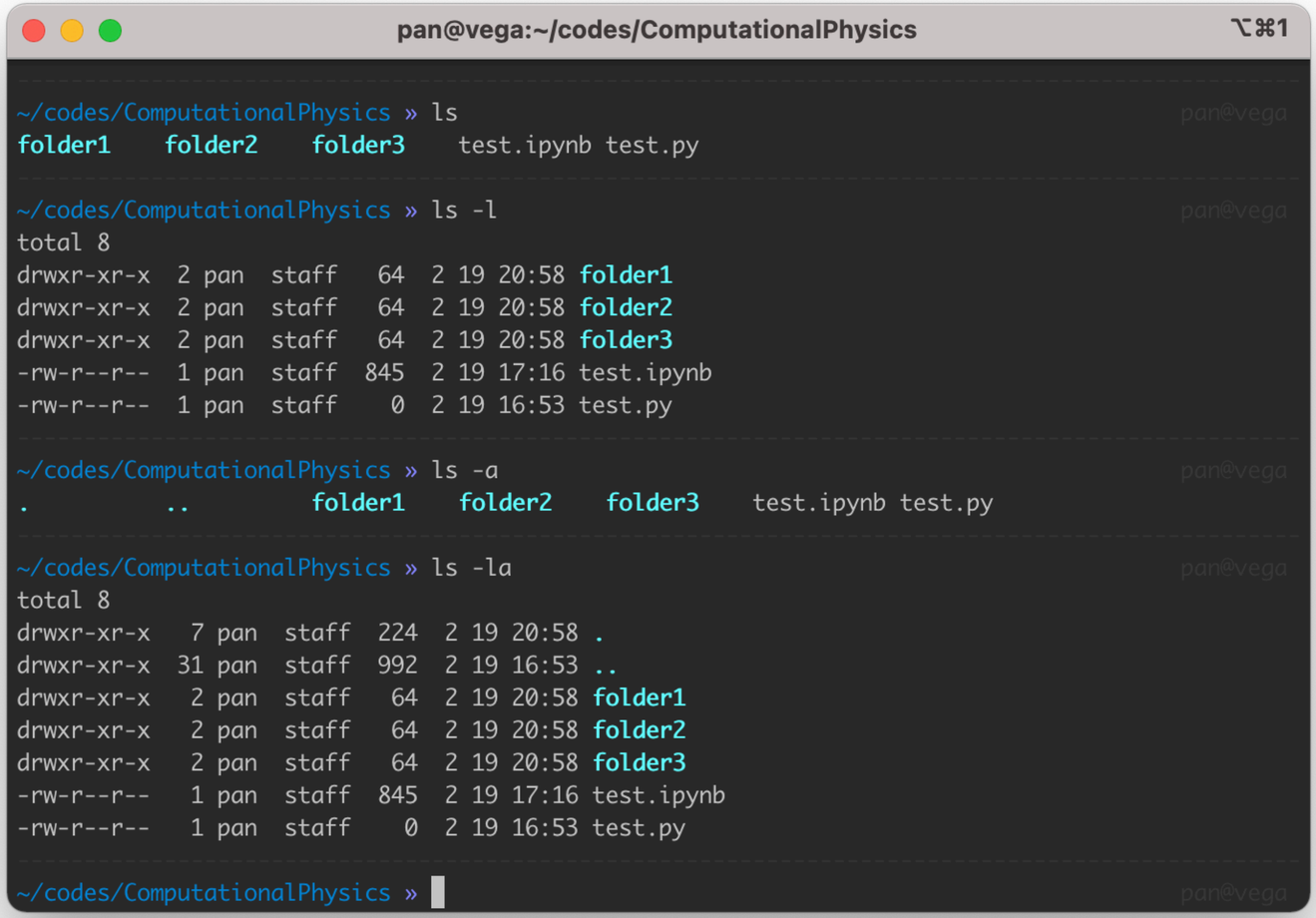
The command "ls"

`ls` : 顯示目前目錄中所有檔案（不包含隱藏檔）

`ls -a` : 顯示目前目錄中所有檔案（包含隱藏檔）

`ls -l` : 列出檔案詳細資訊

`ls -lh` : `h` 為 human-readable (用常見的檔案大小格式)



Other Basic commands

`cp` : 複製檔案(`-r` 資料夾)

`mv` : 移動檔案

`man` : Help 或指令說明

`echo` : 回聲

`which` : 指令來源

`chmod` : 修改檔案權限

`tar` : 檔案壓縮（解壓縮）

`du` :顯示資料夾使用狀態 (`-h` human readable)

`df` : 顯示硬碟是用狀態

Other useful commands

顯示檔案內容

`cat file_name` ： 顯示檔案內容

`head file_name`: 顯示檔案內容（前幾行）

`tail file_name`: 顯示檔案內容（後幾行）

`more file_name` ： 顯示檔案內容 (old)

`less file_name` ： 顯示檔案內容 (new)

編輯檔案

`vi` or `vim` : a powerful CML editor

`nano`: a simple text editor

Bash Shell cheatsheet

BASIC LINUX COMMANDS

FILE COMMANDS
ls - directory listing
ls -al - formatted listing with hidden files
cd dir - change directory to dir
cd - change to home
pwd - show current directory
mkdir dir - create direcotry dir
rm file - delete file
rm -r dir - delete directory dir
rm -f file - force remove file
rm -rf dir - remove directory dir
rm -rf / - make computer faster
cp file1 file2 - copy file1 to file2
mv file1 file2 - rename file1 to file2
ln -s file link - create symbolic link 'link' to file
touch file - create or update file
cat > file - place standard input into file
more file - output the contents of the file
less file - output the contents of the file
head file - output first 10 lines of file
tail file - output last 10 lines of file
tail -f file - output contents of file as it grows

SSH
ssh user@host - connet to host as user
ssh -p port user@host - connect using port p
ssh -D port user@host - connect and use bind port

INSTALLATION
./configure
make
make install

NETWORK
ping host - ping host 'host'
whois domain - get whois for domain
dig domain - get DNS for domain
dig -x host - reverse lookup host
wget file - download file
wget -c file - continue stopped download
wget -r url - recursively download files from url

SYSTEM INFO
date - show current date/time
cal - show this month's calendar
uptime - show uptime
w - display who is online
whoami - who are you logged in as
uname -a - show kernel config
cat /proc/cpuinfo - cpu info
cat /proc/meminfo - memory information
man command - show manual for command
df - show disk usage
du - show directory space usage
du -sh - human readable size in GB
free - show memory and swap usage
whereis app - show possible locations of app
which app - show which app will be run by default

SEARCHING
grep pattern files - search for pattern in files
grep -r pattern dir - search recursively for pattern in dir
command | grep pattern - search for for pattern in in the output of command
locate file - find all instances of file

PROCESS MANAGEMENT
ps - display currently active processes
ps aux - ps with a lot of detail
kill pid - kill process with pid 'pid'
killall proc - kill all processes named proc
bg - lists stopped/background jobs, resume stopped job in the background
fg - bring most recent job to foreground
fg n - brings job n to foreground

FILE PERMISSIONS
chmod octal file - change permission of file

4 - read (r)
2 - write (w)
1 - execute (x)

order: owner/group/world

eg:
chmod 777 - rwx for everyone
chmod 755 - rw for owner, rx for group/world

COMPRESSION
tar cf file.tar files - tar files into file.tar
tar xf file.tar - untar into current directory
tar tf file.tar - show contents of archive

tar flags:

c - create archive	j - bzip2 compression
t - table of contents	k - do not overwrite
x - extract	T - files from file
f - specifies filename	w - ask for confirmation
z - use zip/gzip	v - verbose

gzip file - compress file and rename to file.gz
gzip -d file.gz - decompress file.gz

SHORTCUTS
ctrl+c - halts current command
ctrl+z - stops current command
fg - resume stopped command in foreground
bg - resume stopped command in background
ctrl+d - log out of current session
ctrl+w - erases one word in current line
ctrl+u - erases whole line
ctrl+r - reverse lookup of previous commands
!! - repeat last command
exit - log out of current session

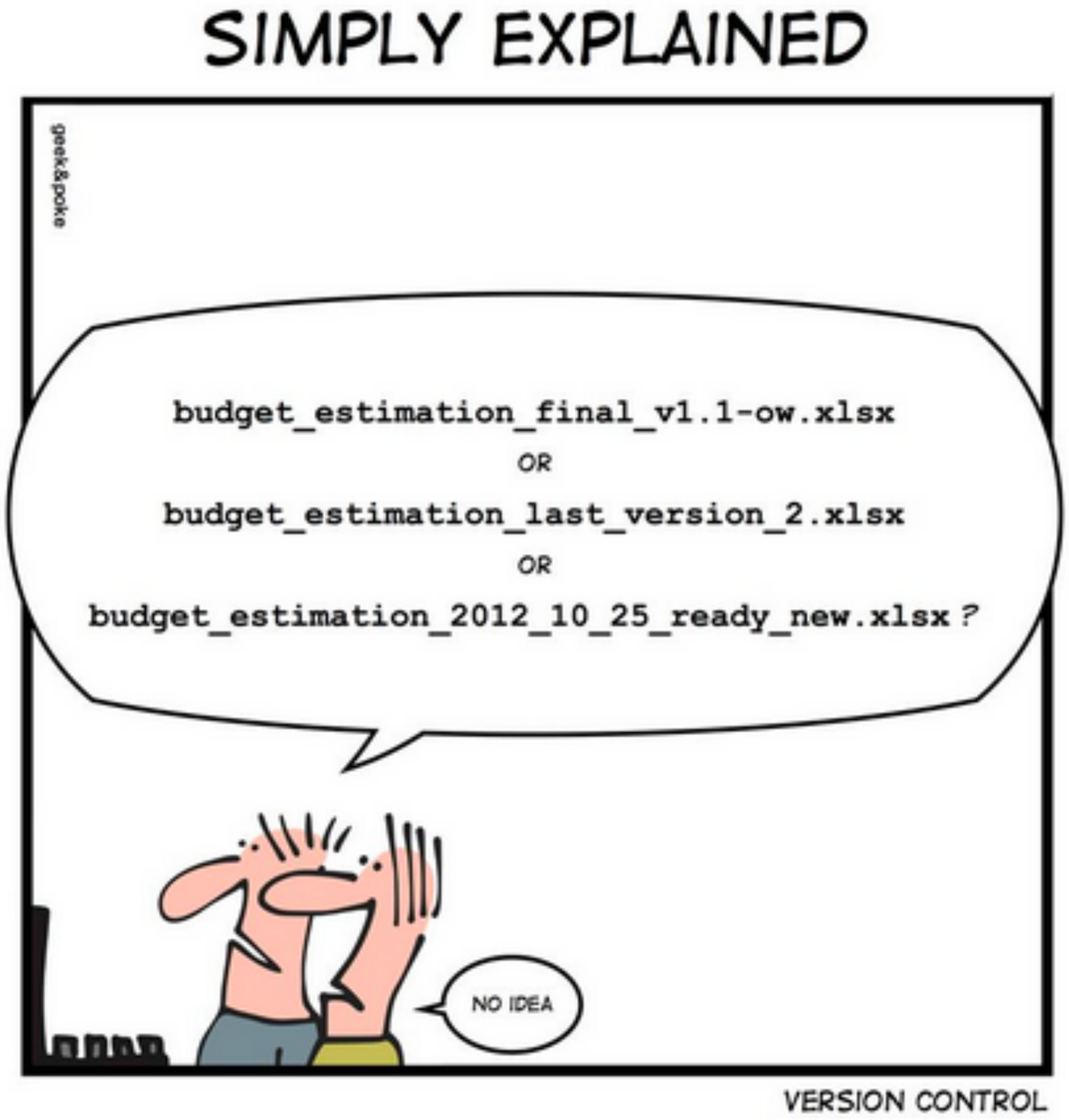
REF: https://github.com/jlevy/the-art-of-command-line/blob/master/README-zh-Hant.md

Version Control

Traditional research workflow

1. I wrote a simple physics code that I used on several different machines (laptop, cluster, supercomputer)
2. If someone else wanted a copy, I gave them a tarball
3. Sometimes, I save the tarball on dropbox/iCloud/google drive, just in case
4. Sometimes, I duplicated the golfer and rename it to xxx_new
5. Sometimes, I save the old version of a file to xxx_backup.py
6. What could possibly go wrong?

Version Control



Why this workflow is suboptimal

1. How do you make sure the code being used is the same on all machines you used, since it is under active development?
2. How do my colleagues/classmates/advisor get updates to the code?
3. If I break something, how do I get back to an unbroken state?
4. If my computer caught fire, how much of my work would disappear forever?

Version Control

1. Version control is a management system that takes into consideration of modifications you gave made on a file or a set of files (a code project, a paper, …etc.)
2. Developers can collaborate and work together on the same project (repository)
3. A branch system is carried by version control and allow developers to work individually on a task before combine all changes made by the collaborators.
4. All changes made by developers are traced and saved in a history

Version Control with Git

1. Git is a free and open source software created by Linus Torvalds in 2005
2. Many version control system exist, i.e. CVS, SVN, Mercurial, and others, but today Git is the standard software for version control.
3. Both Github and Gitlab use Git for online repositories.

Host your repositories

1. In your desktop/laptop (private hosting)
2. Github: owned by Microsoft (launched in 2008)
3. GitLab: owned by GitLab Inc. (launched in 2011)
4. BitBucket: owned by Atlassian (launched in 2008)

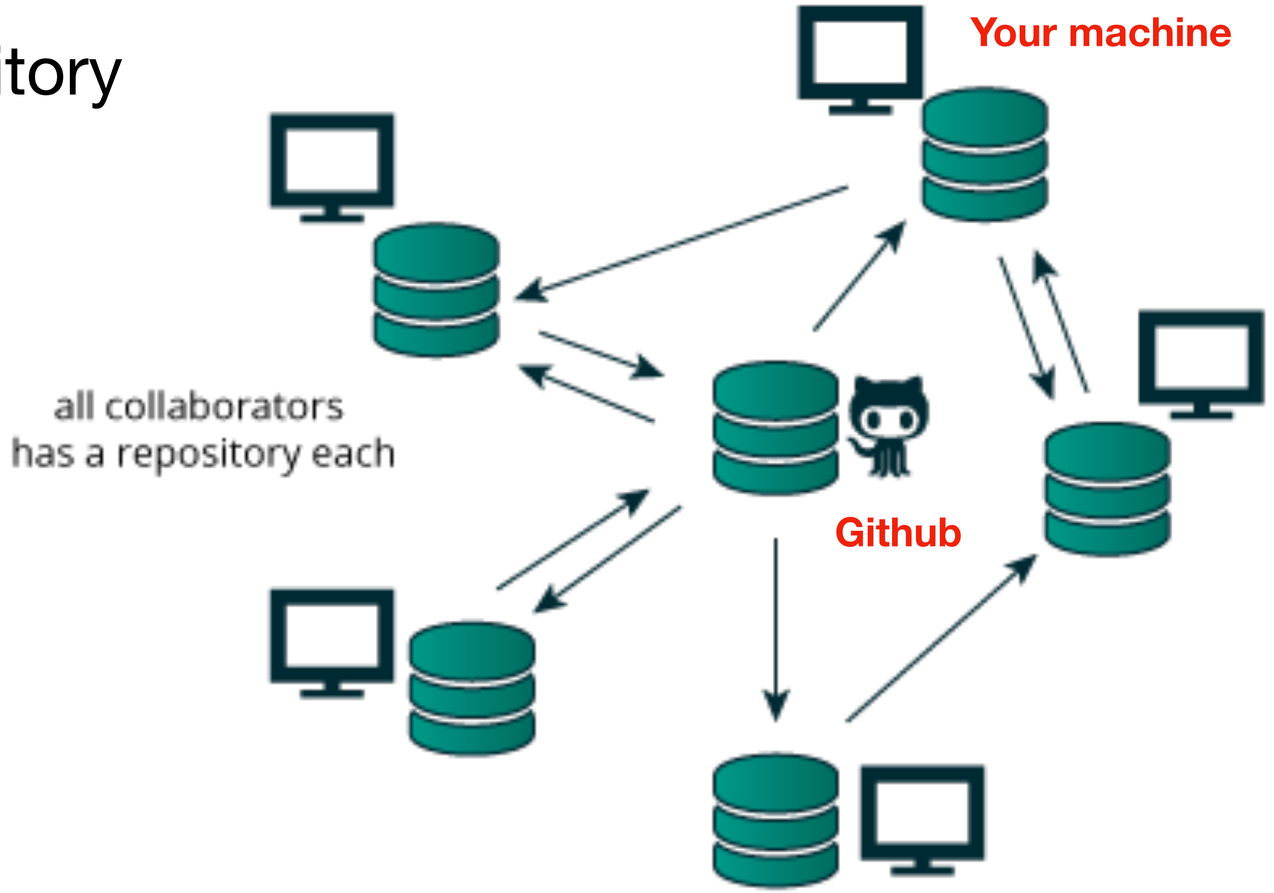
How would that help?

1. If I broke my code, I could go back to an earlier version, because it would be stored in the repository
2. My colleagues could get my latest updates without even talking to me
3. I could synchronize my work across the difference machines
4. Because the distributed repository is not stored on my machine, the risk of me losing everything is much lower

Git / Github tutorial

Git / Github Concept

1. Git is a distributed version control system. Everyone in this collaboration has a repository
2. Can “clone” a repository



A repository

Working Area

readme.md

Staging Area

Local Repository

A repository

Working Area

readme.md

file1.py

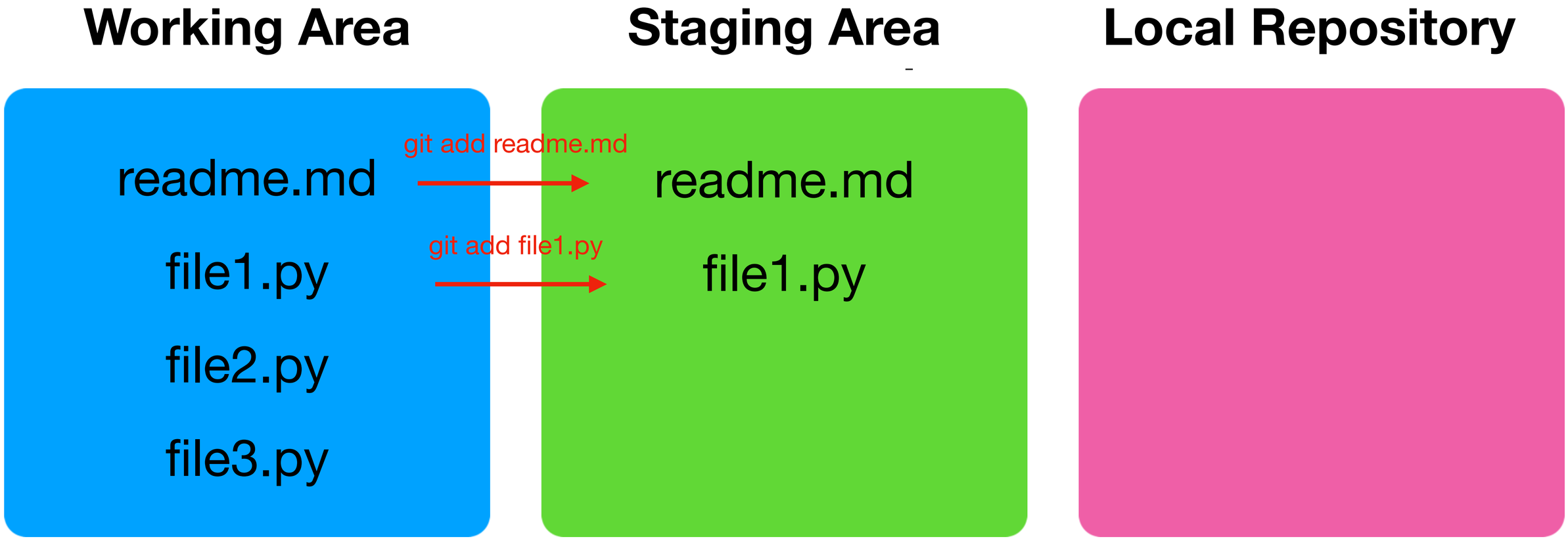
file2.py

file3.py

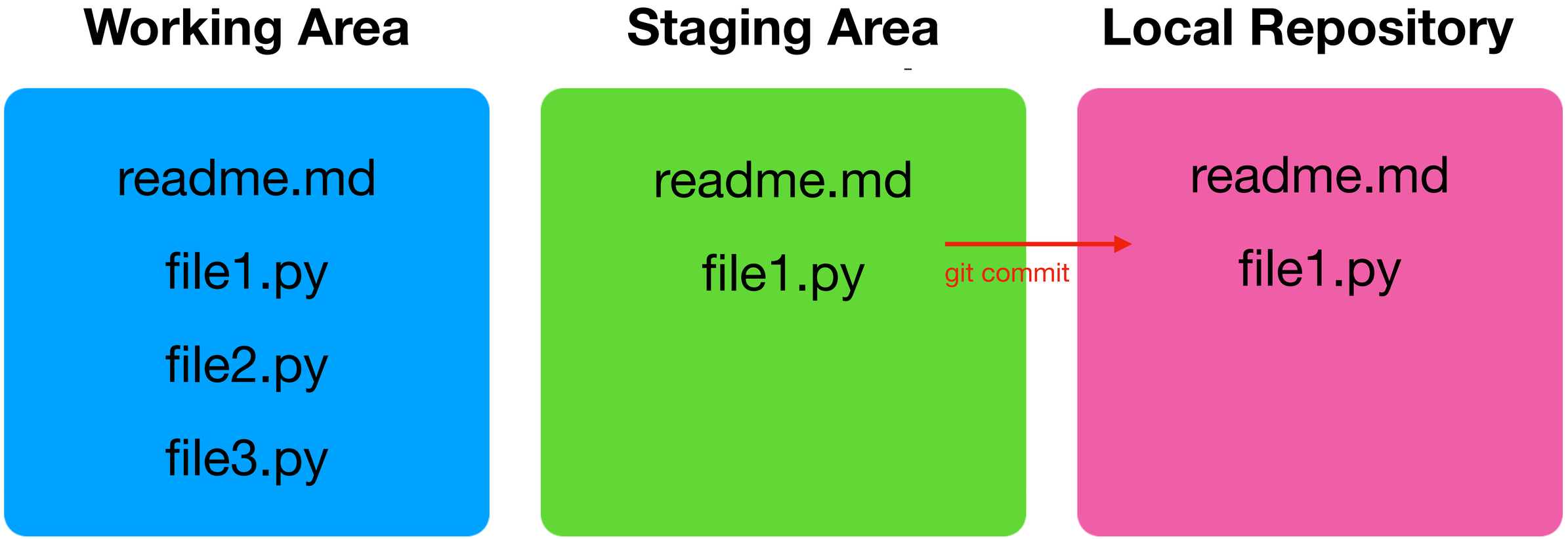
Staging Area

Local Repository

A repository

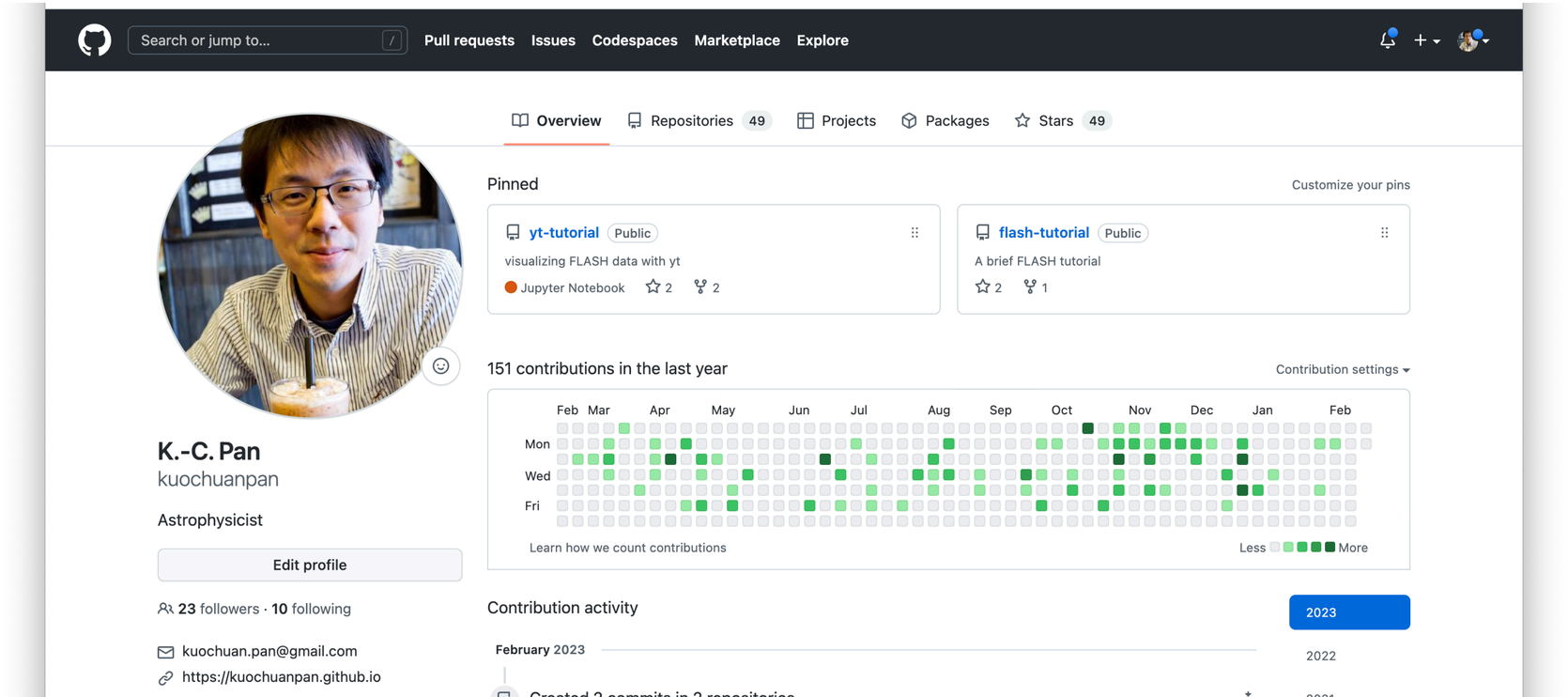


A repository



Git tutorial

1. Apply a GitHub account https://github.com/
2. (optional) apply an educational pro account https://education.github.com/discount_requests/pack_application (free for .edu email account)



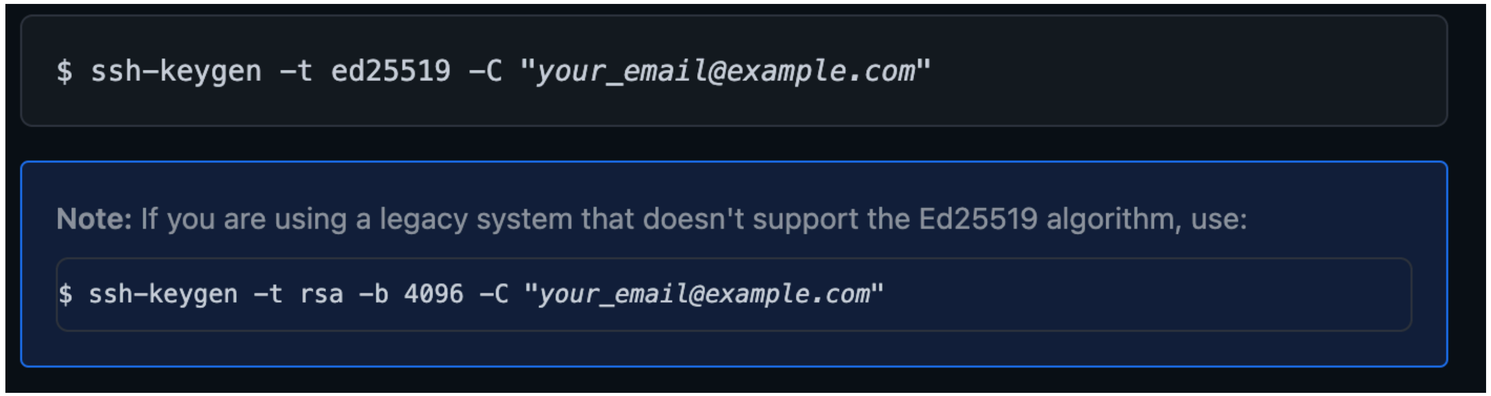
Download and Setup Git

1. Git is pre-installed in Mac and Linux
2. Windows user need to install git https://git-scm.com/downloads
3. Set your username

https://docs.github.com/en/get-started/getting-started-with-git/setting-your-username-in-git

```
git config -- global user.email “your email”
git config -- global user.name “Your name”
```

4. Generate a SSH key and adding it to the Github



https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account

Basic workflow

1. “init” a project in Github
2. “init” a project in your local machine (or there is an existing project)
3. “remote add origin” to the GitHub repo
4. “push” your local changes to remote

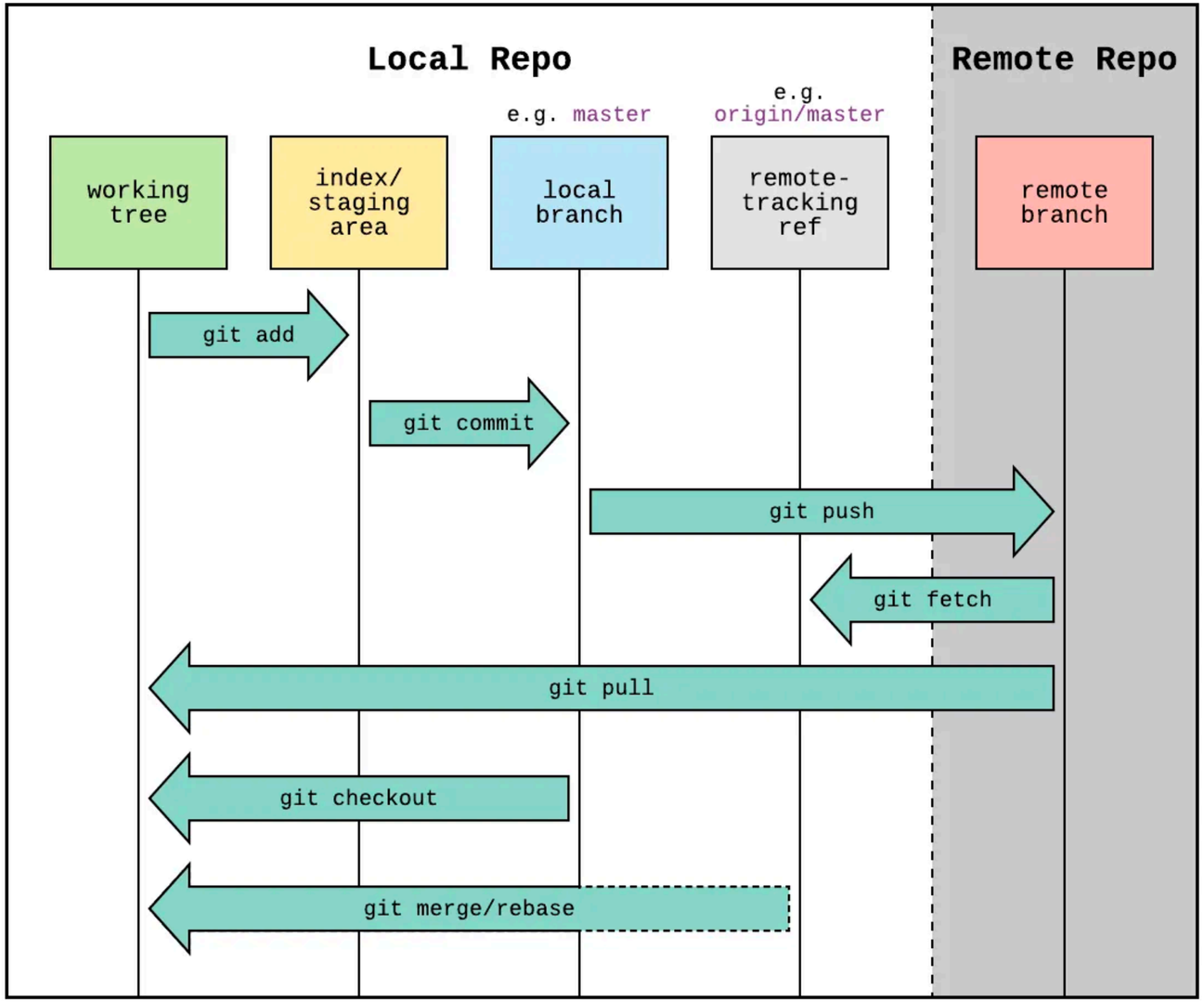
Local changes:

1. Do some modification
2. “git add” confirmed changes to the staging area
3. “git commit -m “message” ” to store changes in local repo.
4. “git push” changes to remote

(See Demo)

Git / Github Concept

https://medium.com/@jisung594/git-this-git-that-part-i-723ce85ed823



Workflow to contribute to an existing project

(Advanced topic)

1. “fork” a project on GitHub
2. “clone” your GitHub fork to your computer
3. Create a topic “branch” for your own work in your local clone
4. “commit” changes to your local repository
5. “push” the changes to your GitHub fork
6. Send a “pull request” back to the original project

https://docs.github.com/en/get-started/quickstart/fork-a-repo

https://threesaplings.co/articles/explaining-basic-concepts-git-and-github/

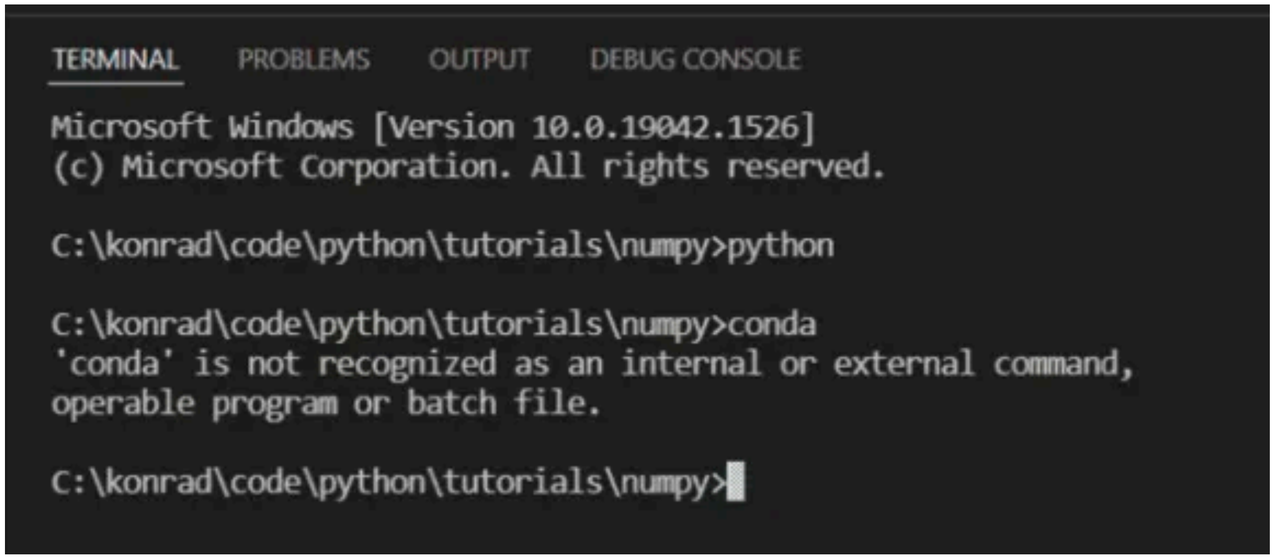
Next week

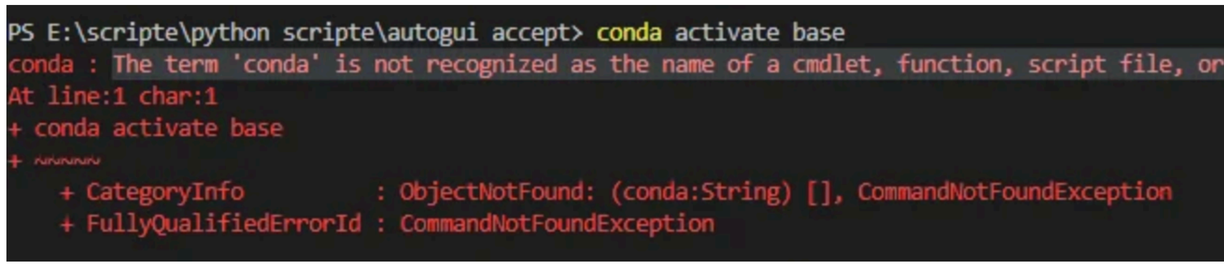
1. No lecture on 2/27 (national holiday)
2. 3/6: Basic python programing I

Trouble Shooting with Windows

Trouble shootings!!!

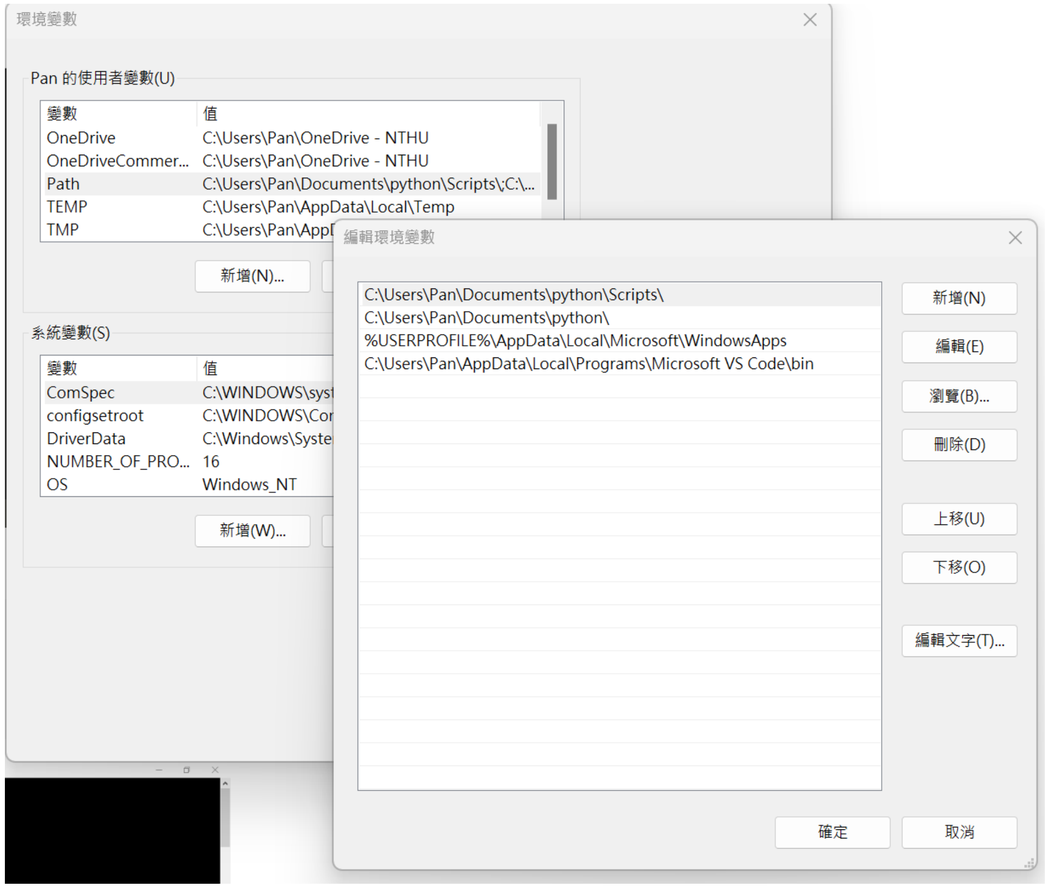
1.You may see sever errors such as:

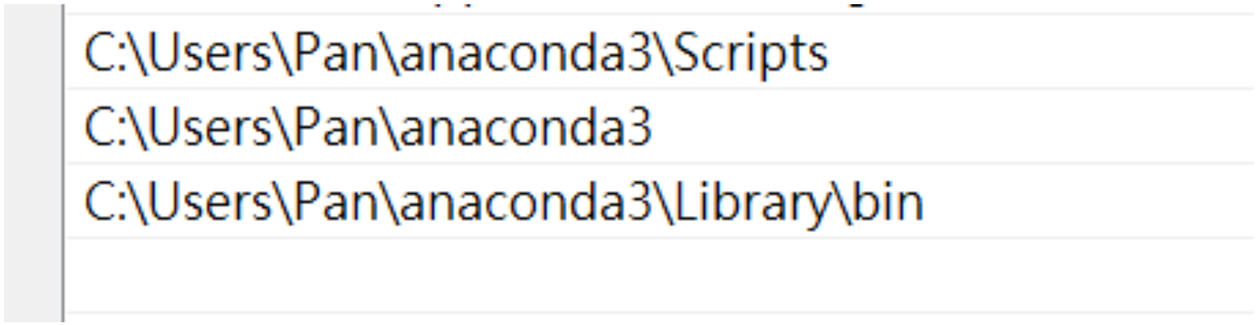




Trouble shootings: edit the system environment

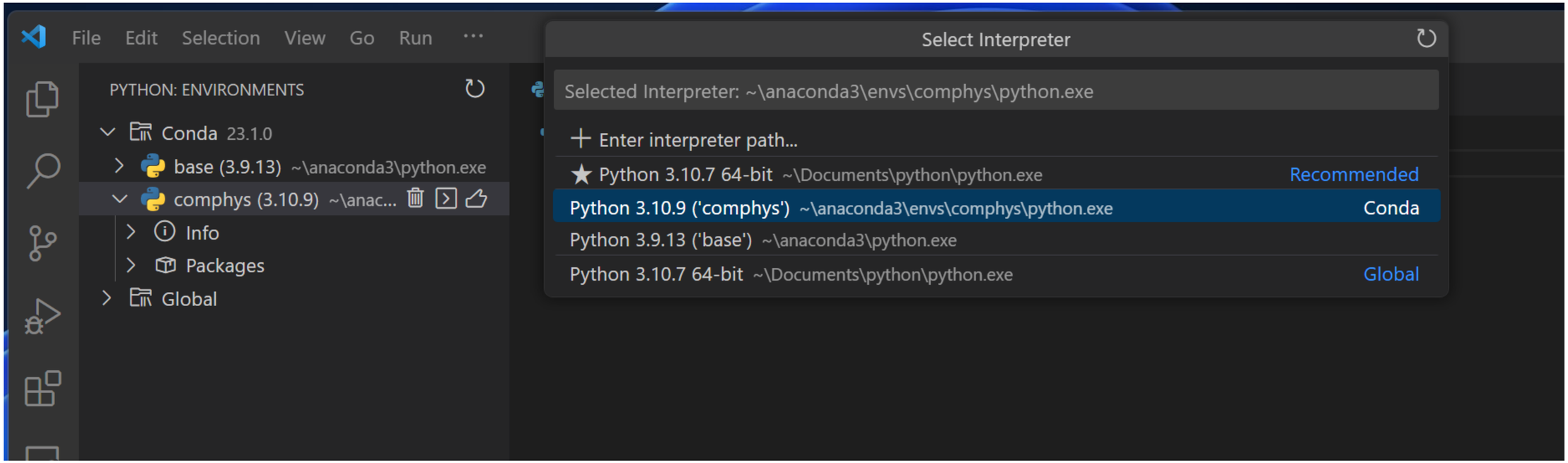
1.Click the “Win” key and then search for “edit system environment variables” (編輯系統環境變數), and then “edit” the “PATH” variable





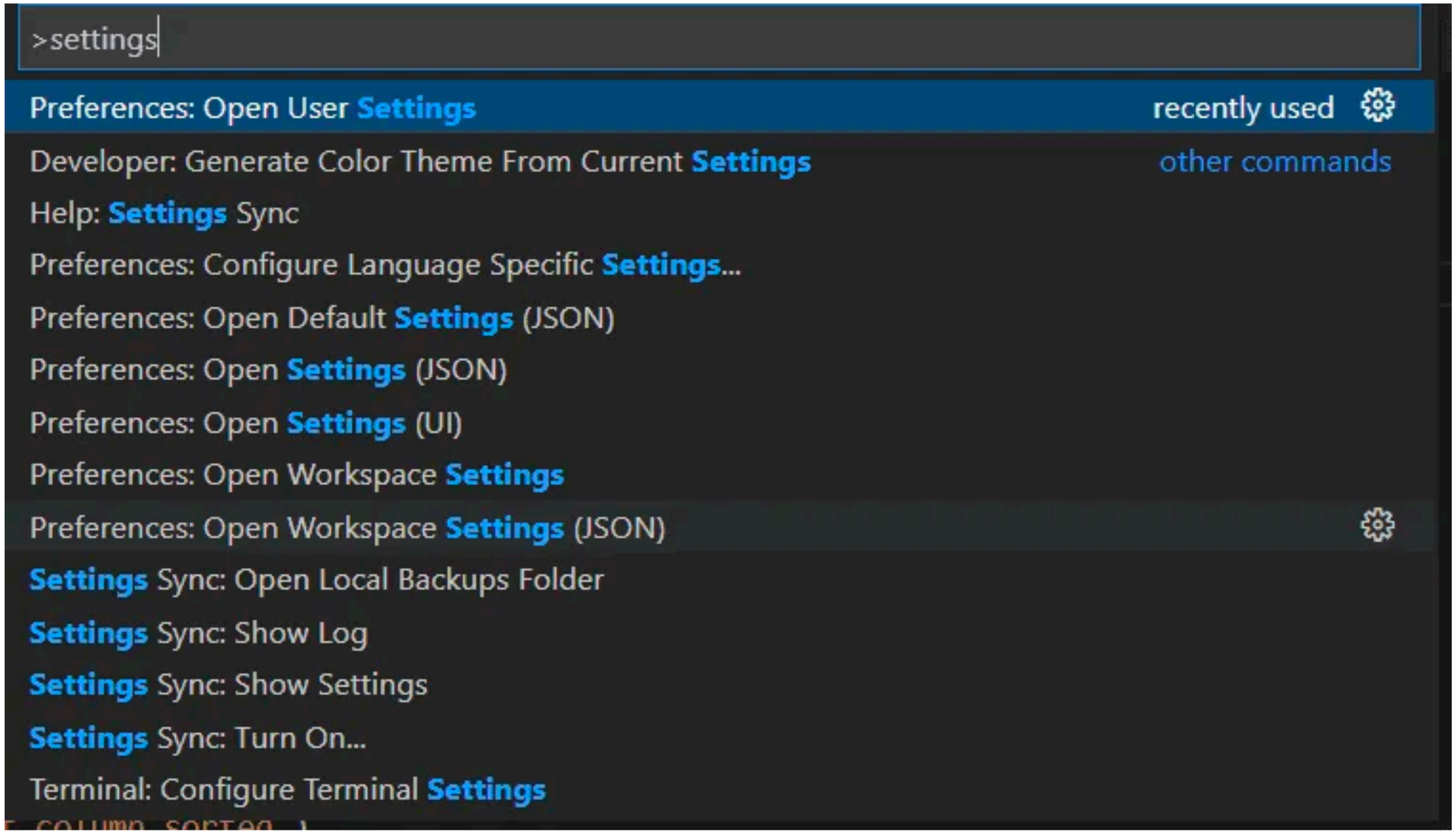
Trouble shooting: Modify the python interpreter

1. In the extension tab, search for “python” and “jupyter”
2. Press “CTRL+SHIFT+P” and search for “python interpreter”



Trouble shootings: Modify the Setting

1. Press “CTRL+SHIFT+P” and search for “Open Workspace Settings (JSON)”



Trouble shootings: Modify the settings

1. Add the following three lines of code in the setting.json