

# Setting Up A Software Development Environment

---

Ryan M. Richard  
Ames National  
Laboratory and Iowa  
State University

Hi. I'm a Mac.  
And I'm a PC.



# Assumptions

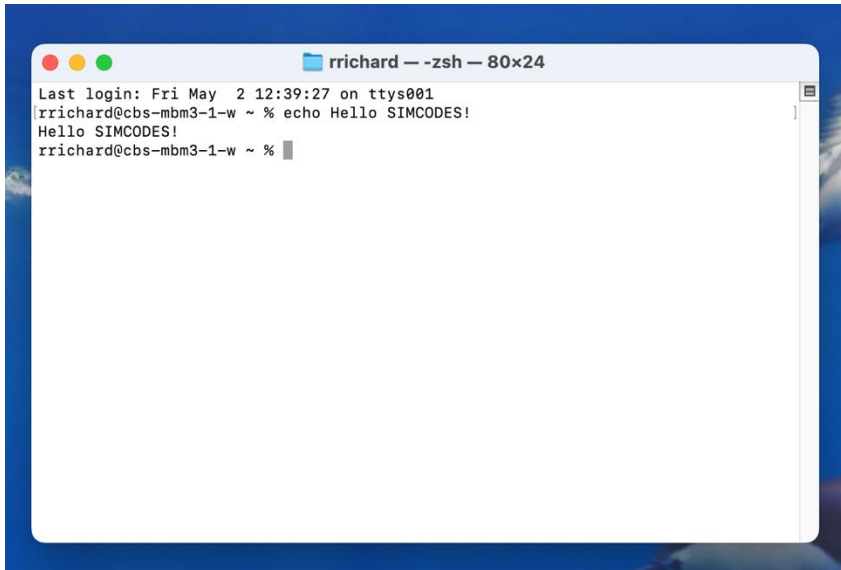
---

- Have access to a computer.
- The computer is running MacOS (preferred) or Windows (I'll begrudgingly accept).
- Can install software on that computer.
- Current programming level is: "Terminal, that's where I board a plane, right?"
  - If you're more experienced than this, the assumption is you'll let me oversimplify many many things.

# Objectives

---

- Know what a Linux terminal is.
- Know how to access a Linux terminal.
- Know the most important Linux terminal commands.
- Install Python.
- Write a “Hello World” Python module.
- Run the “Hello World” Python module.



# What is a Terminal?

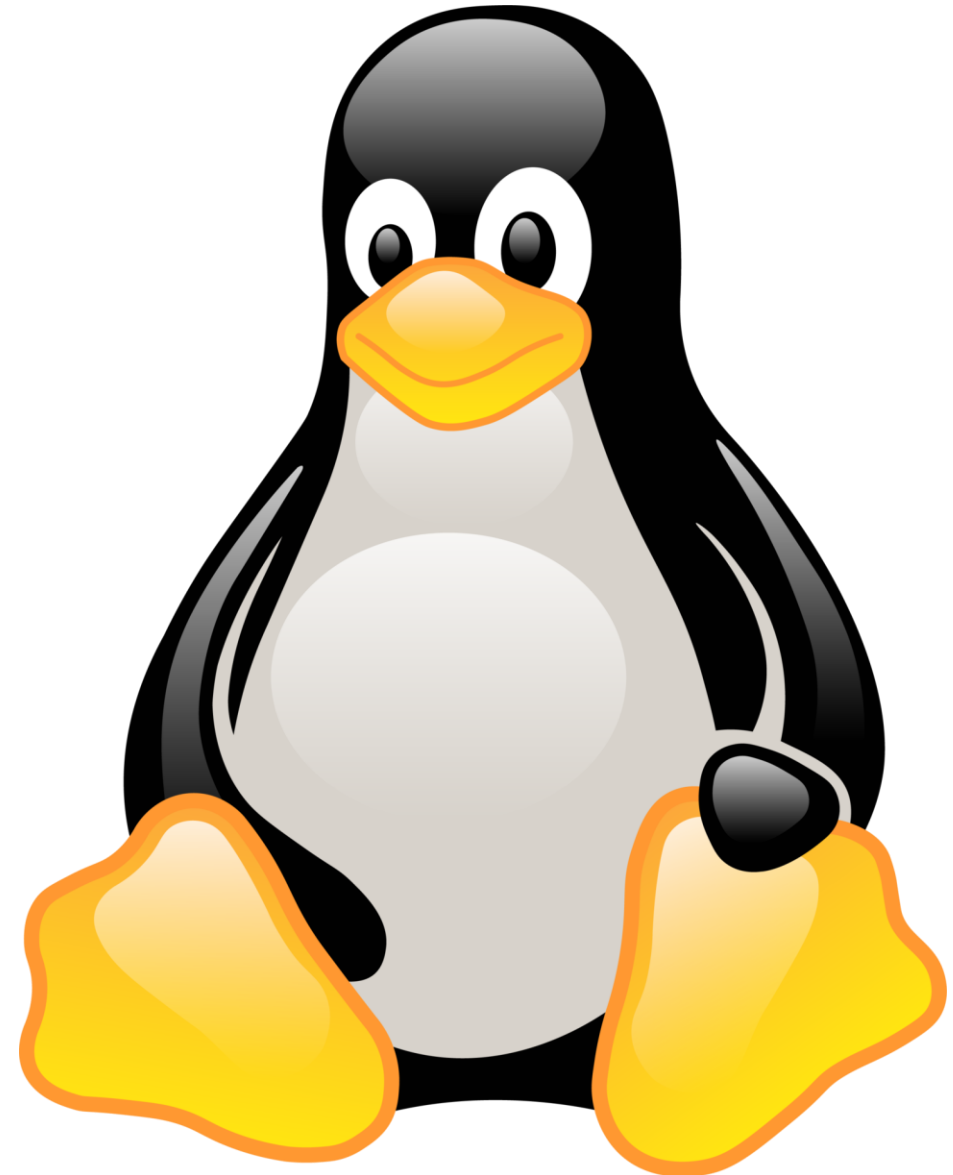
---

- Most people interact with their computer graphically.
  - Called a “Graphical User Interface” or GUI for short (pronounced gooey).
  - Graphics in the GUI represent pieces of the program.
  - Use mouse to select pieces of the graphics.
  - Typing is reserved for entering text/numerical data.
- Can also interact with a computer through text.
  - Use a program called a “terminal” to do this.

# What is Linux?

---

- Linux is an operating system (OS) alternative to MacOS and Windows.
  - (Pause for diehard Linux people to correct me...)
- Unlike MacOS/Windows, Linux is open source.
  - “Open source” means that the source code is publicly available.
  - Easier for people to add features.
- Programmer’s preference.
  - Virtually every programming language is supported.
  - Uses little resources.
  - Runs on most hardware.
  - Community supported.



# What is a Linux Terminal?

Strictly speaking, it's a terminal you can use to interact with the Linux OS.

More loosely, it's a terminal that supports Linux commands.

Contrast this with Window's PowerShell (if you know about it).

TL;DR.

"Linux" = best OS (according to programmers).

"Terminal" = way to interact with your computer via text.

"Linux Terminal" = is the best way to interact with your computer via text (according to programmers).

Disclaimer: From here forward we assume Linux terminals, i.e., if you try the commands in PowerShell or another terminal they may not work.

# Why Do I Want To Use a Terminal?

Because I said so... (just kidding)

Automation!!!

Great for tutorials.

- Easy: `git clone https://github.com/SIMCODES-ISU/.github`
- Hard: “Open a web browser. Navigate to GitHub. Up by the cat-thing, type in the search bar...”.

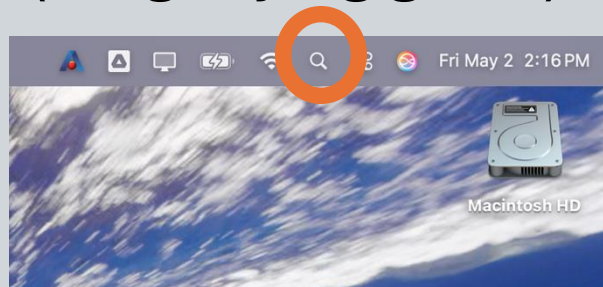
Usually how you interact with remote computers.

- Sending/receiving graphics requires sending a lot of extra data.

# Accessing a Linux Terminal (MacOS): Part 1

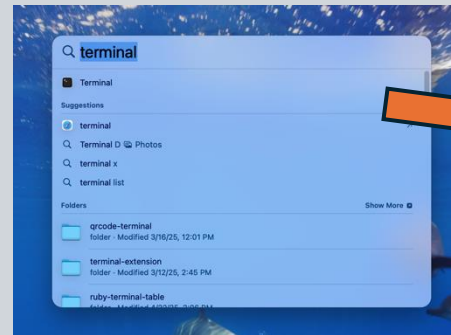
1

Open spotlight  
(magnifying glass)



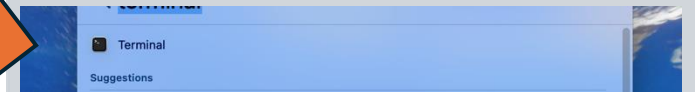
2

Type “terminal”



3

Click on terminal

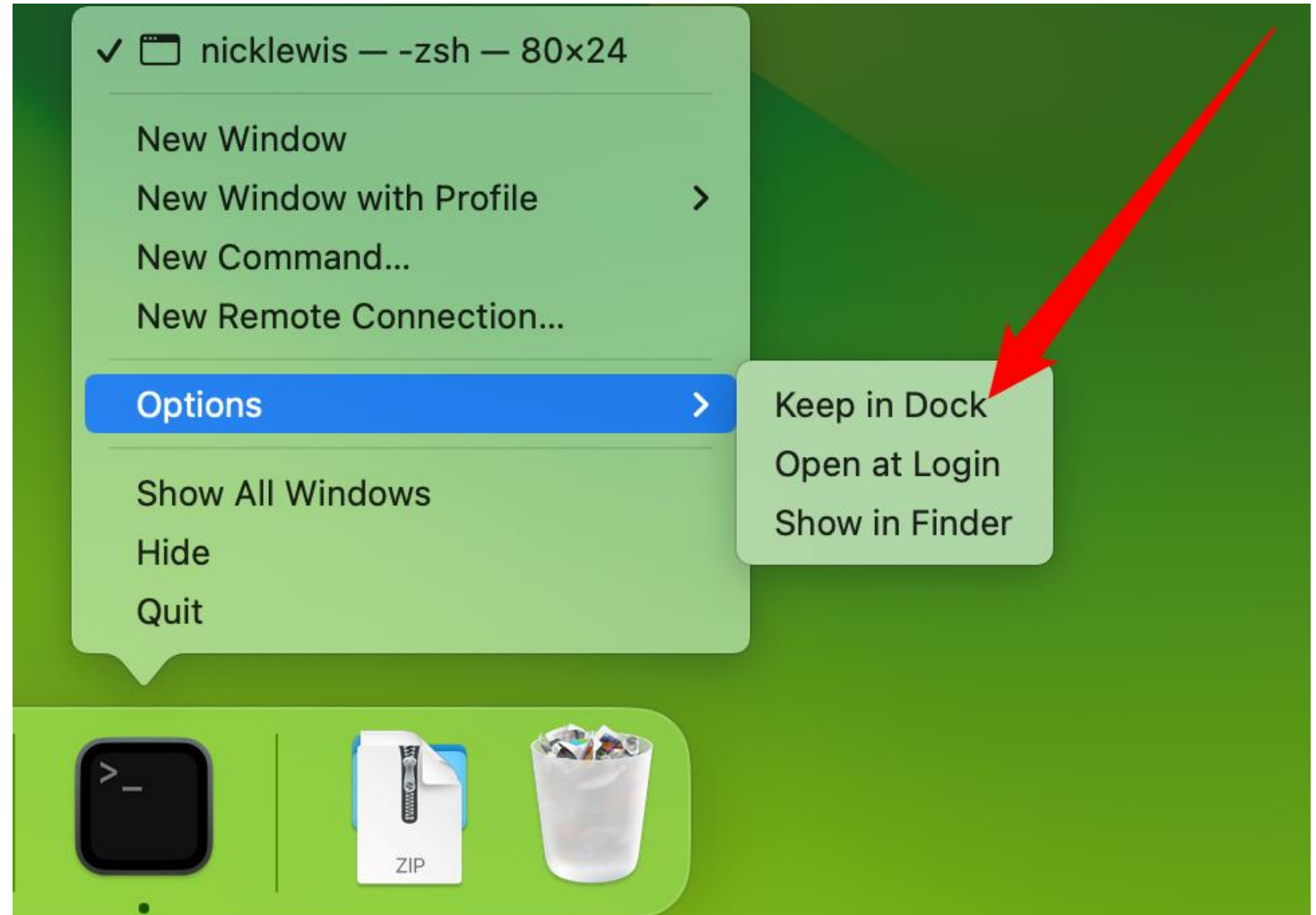




# Accessing a Linux Terminal (MacOS): Part 2

---

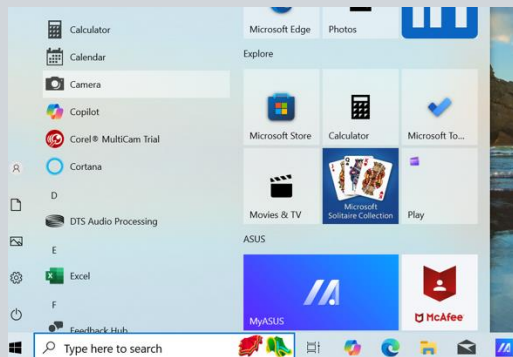
- Please wait for your Windows colleagues (it's going to be a while).
- In the meantime, consider pinning terminal to your dock.



# Accessing a Linux Terminal (Windows <11): Part 1

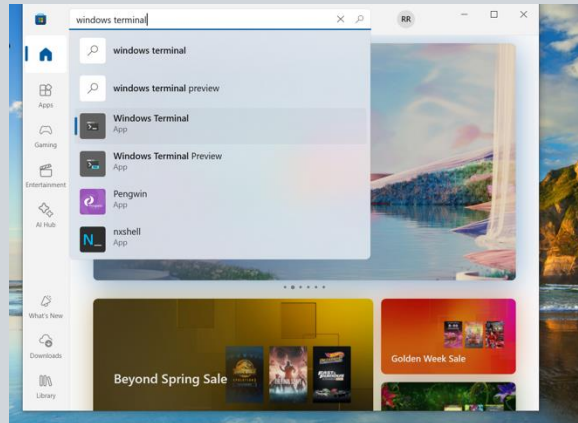
1

Go to Microsoft Store.



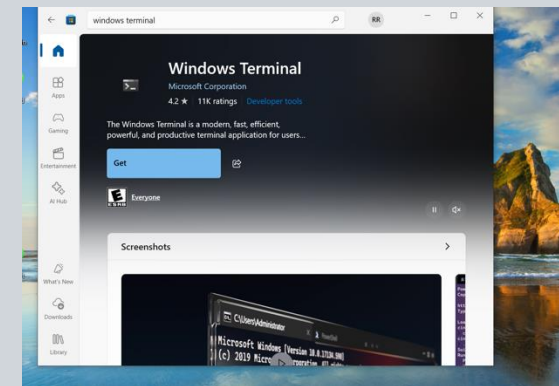
2

Search “Windows Terminal”.



3

Click “Get”.



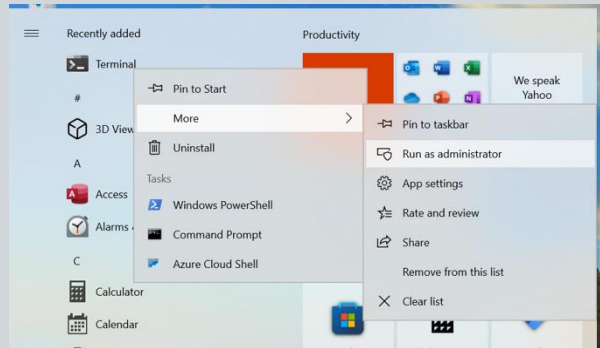
# Accessing a Linux Terminal (Windows): Part 2

4

Open start. Search for “Terminal.” Right click “Terminal”.

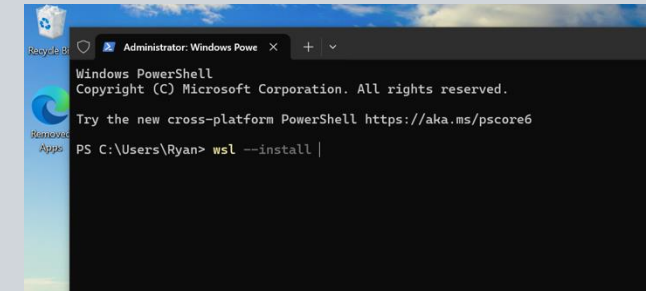
5

Choose “More” -> “Run as administrator”.



6

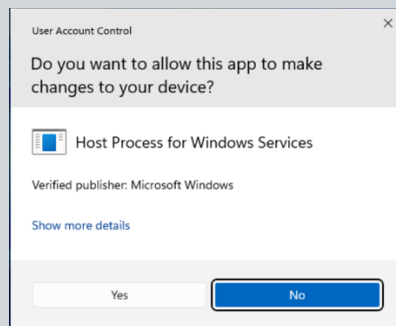
Type `wsl --install`



# Accessing a Linux Terminal (Windows): Part 3

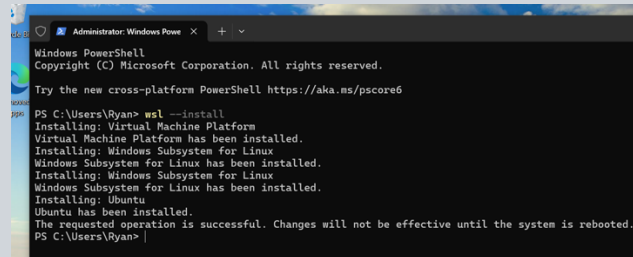
7

If it asks for permission, grant it.



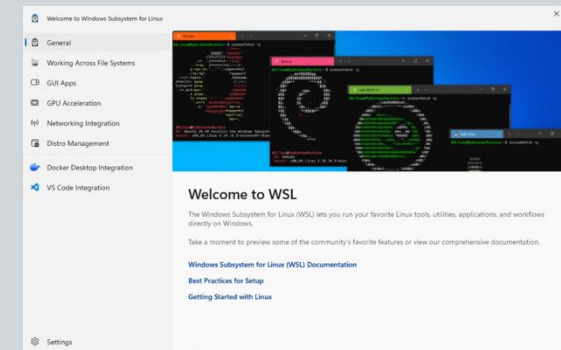
8

After install, reboot.



9

If you get a “Welcome to WSL” window, close it.



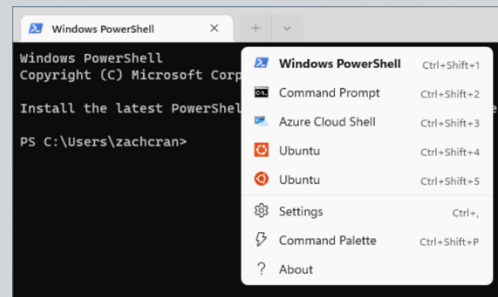
# Accessing a Linux Terminal (Windows): Part 4

10

Open terminal.

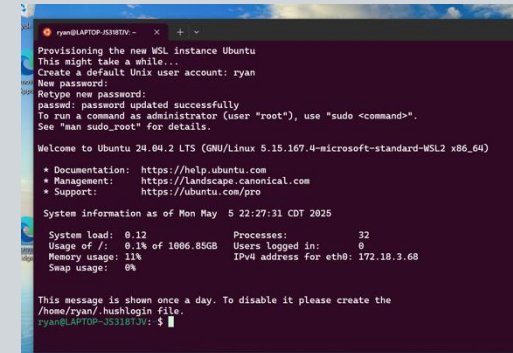
11

Choose an “Ubuntu” tab (if you have two pick either).



12

Create a Linux profile (username and password).



# Windows Terminal Notes

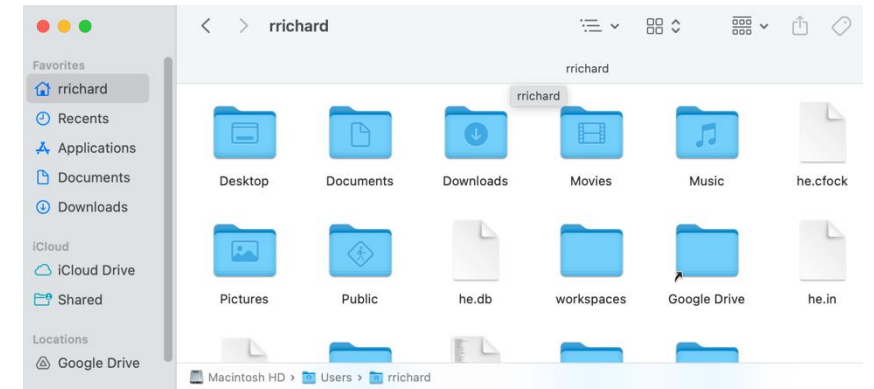
---

- From now on “terminal” should be thought of as being synonymous with “open an Ubuntu tab in Windows Terminal”.
- Linux profile need not be same as Windows profile.
  - Security wise makes little difference. Someone logged in with your Windows profile can get into your Linux profile (and vice versa).
  - Best practice: make sure password(s) is/are strong.
- Don't access Windows files (C:\Users\...) from Linux, nor Linux files (/home/...) from Windows.
  - Think of them as separate file systems (e.g., like one lives on a USB).
  - Windows and Linux use different file conventions, unless you know what you're doing you can easily corrupt files.

# I Have a (Linux) Terminal. Now What?

---

- Start by thinking of your terminal like a file system.
- When you start terminal, you are in a folder/directory, but which one?
  - Type `pwd` (and hit enter)
  - The result is your current directory.
  - `pwd` stands for:
    - Print (write to the terminal)
    - Working Directory (the directory you are currently working in).

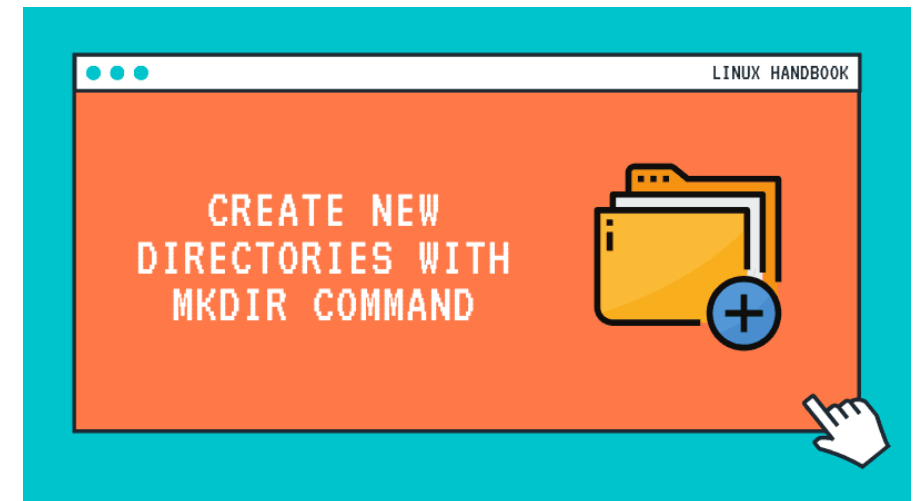


A terminal window with a black background and white text. The prompt is a white greater-than sign '&gt;'. The command 'pwd' has been entered, and the output is '/home/user'.

# Other Filesystem Commands

```
[rrichard@cbs-mbm3-1-w ~ % ls  
Desktop      Downloads    Library      Music        Public       spack  
Documents    Google Drive Movies        Pictures     simcodes.txt workspaces
```

- `ls` is used to see the files and directories in your current working directory.
  - `ls` is short for list (not sure why we needed to abbreviate it...)
  - You may not get color (if you do, the colors denote file types)
- `mkdir <dirname>` is used to create directories
  - `mkdir` is short for make directory.
  - `<dirname>` should be replaced with what you want to call the directory.





```
[rrichard@cbs-mbm3-1-w ~ % mkdir my_simcodes_tutorial_directory  
rrichard@cbs-mbm3-1-w ~ % █
```

## Pop Quiz: Where's My Directory?

- When I run `mkdir` nothing happens. How can I check if `mkdir` did its job?

```
[rrichard@cbs-mbm3-1-w ~ % mkdir my_simcodes_tutorial_directory  
[rrichard@cbs-mbm3-1-w ~ % ls  
Desktop                my_simcodes_tutorial_directory  
Documents              Pictures  
Downloads              Public  
Google Drive           simcodes.txt  
Library                spack  
Movies                 workspaces  
Music
```

# Pop Quiz: Answer

---

ls

## Even More Filesystem Commands

- To change directory use `cd`.
- Pro tip: “tab” key is auto-completes file/directory names. If you start typing the name of a long directory, like “my\_simcodes\_tutorial\_directory”, try hitting tab after a few letters to let the terminal fill in the rest.
  - Auto-complete will only fill in as much as it can unambiguously. If it only gives you part of the name hit tab a second time to see the choices.

```
[rrichard@cbs-mbm3-1-w ~ % cd my_simcodes_tutorial_directory
[rrichard@cbs-mbm3-1-w my_simcodes_tutorial_directory % pwd
/Users/rrichard/my_simcodes_tutorial_directory
rrichard@cbs-mbm3-1-w my_simcodes_tutorial_directory %
```

# Notes on File Names



Whoever created the terminal made a terrible decision, spaces are used to separate arguments to commands.

E.g., `cd my_simcodes_tutorial_directory` the command (`cd`) is separated from the argument (`my_simcodes_tutorial_directory`) by a space.



If your file/directory names contain spaces, we now run into a problem



TL;DR get in the habit **NOW** of avoiding spaces or special characters (&\*\$#!) in file/directory names. For our purposes “-” and “\_” are NOT special characters.

# Command Summary

- `pwd` prints the working directory.
- `ls` shows the files and directories in a directory.
- `mkdir` creates a directory.
- `cd` changes the directory.
- There are a ton more  
<https://www.geeksforgeeks.org/linux-commands-cheat-sheet/>

# Objectives



- ~~Know what a Linux terminal is.~~
- ~~Know how to access a Linux terminal.~~
- ~~Know the most important Linux terminal commands.~~
- Install Python.
- Write a “Hello World” Python module.
- Run the “Hello World” Python module.

# What is Python?

- Named for Monty Python
- [GeeksForGeeks](#): Most popular programming language.
- Design philosophy prioritizes code readability.
- “High-level” language.
  - Python programs are relatively performant without exposing “low-level” details like memory management.
  - “Natural language”-like syntax
- Becoming the *de facto* language of scientific computing.
  - Notable exception is high-performance computing where C++ still dominates, but Python is gaining.



# Installing Python (MacOS)

You're done.



# Installing Python (Windows)

---

You're also  
done.

# What is a Python Module?

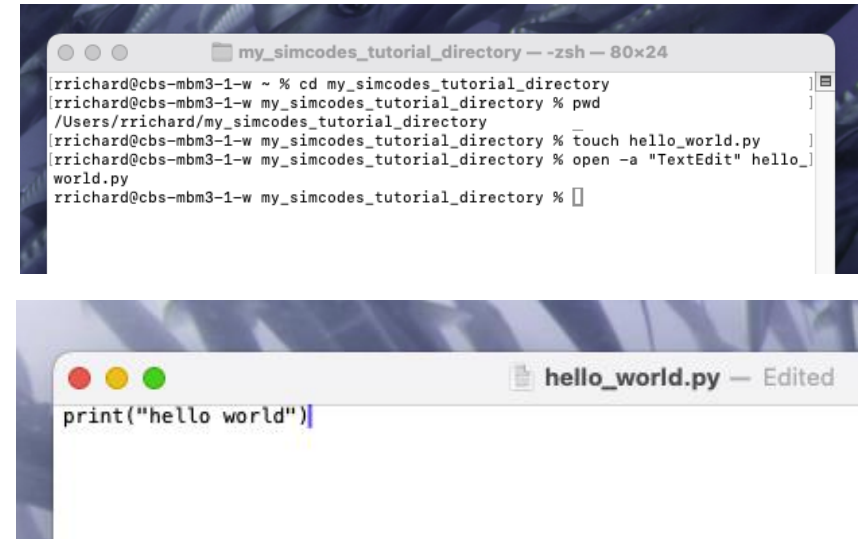
---

- It is a text file that contains Python source code.
  - Caveat: Need to be plain text files, e.g., **NOT** Word documents.
- Named like `name_of_file.py` (“`.py`” is standard extension).
- The Python “interpreter” is responsible for running the module.
  - `python` (or sometimes `python3`).
- To run a Python module you simply tell the interpreter the name of the module
  - E.g., `python name_of_file.py`

# Creating a Python Module (MacOS)

---

1. Open terminal.
2. `touch hello_world.py`
  - `touch` makes an empty file (if it doesn't exist) and updates its timestamp to now.
3. `open -a "TextEdit" hello_world.py`
  - If the file doesn't exist, this command will complain.
4. Type `print('hello world')`, then `⌘ + s` (saves).
5. Wait for your Windows colleagues (again).



```
my_simcodes_tutorial_directory — zsh — 80x24
rrichard@cbs-mbm3-1-w ~ % cd my_simcodes_tutorial_directory
rrichard@cbs-mbm3-1-w my_simcodes_tutorial_directory % pwd
/Users/rrichard/my_simcodes_tutorial_directory
rrichard@cbs-mbm3-1-w my_simcodes_tutorial_directory % touch hello_world.py
rrichard@cbs-mbm3-1-w my_simcodes_tutorial_directory % open -a "TextEdit" hello_world.py
rrichard@cbs-mbm3-1-w my_simcodes_tutorial_directory %
```

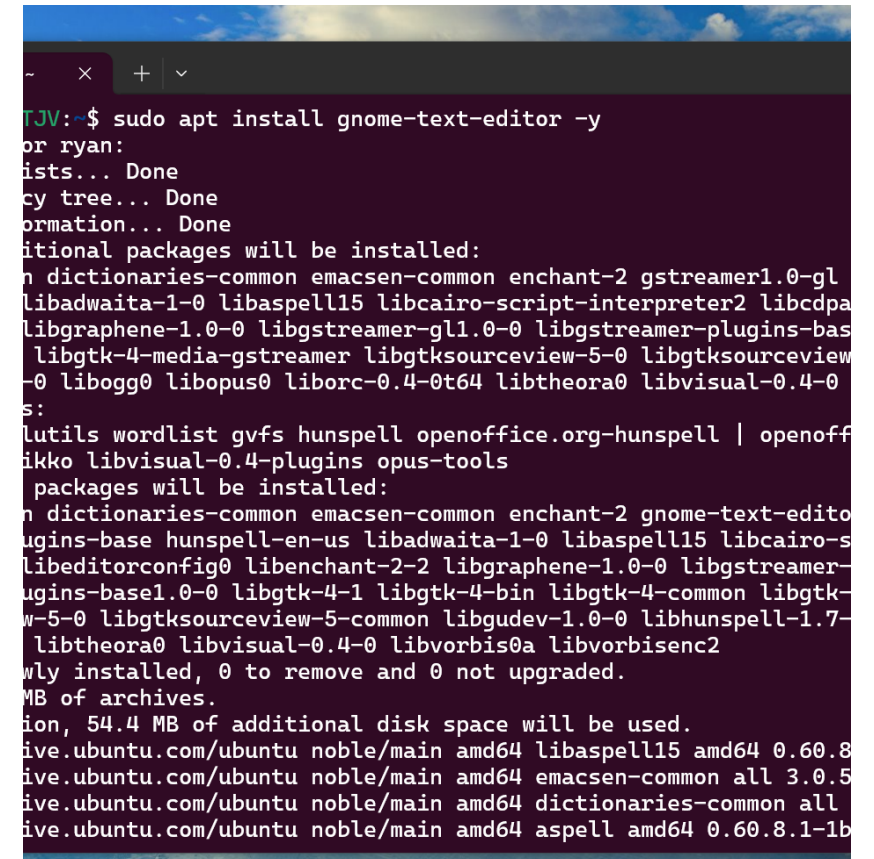
  

```
hello_world.py — Edited
print("hello world")
```

# Creating a Python Module (Windows): Part 1

---

1. Open a terminal.
2. `sudo apt install gnome-text-editor -y`
  - `sudo` stands for super user do (Linux equivalent of administrator).
  - `apt` stands for **A**dvanced **P**ackage **T**ool (Linux equivalent of Microsoft Store).
  - `install gnome-text-editor` (installs Gnome's text editor)
  - `-y` Answers “yes” to all questions (like “are you sure”).

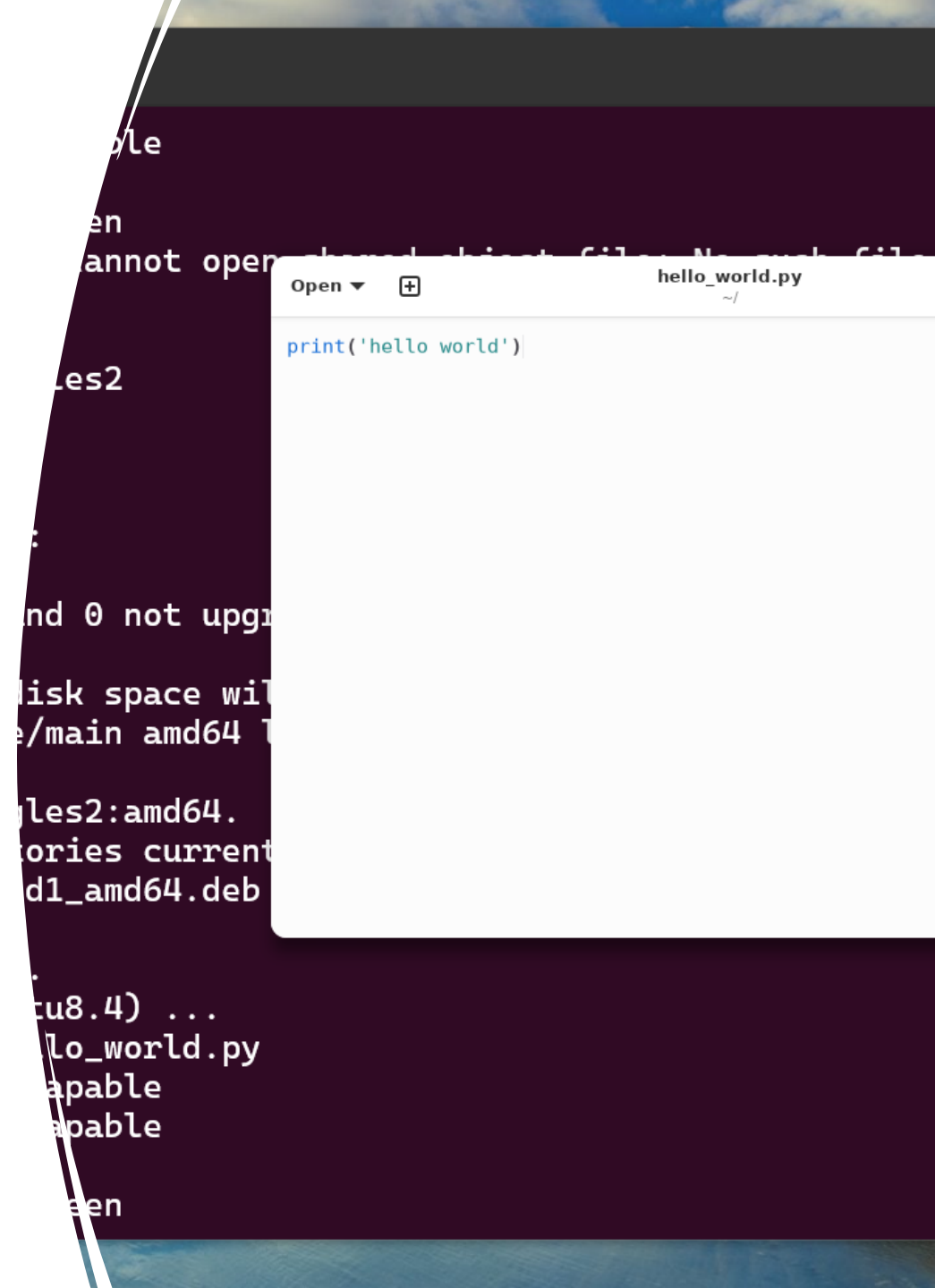


```
TJV:~$ sudo apt install gnome-text-editor -y
or ryan:
ists... Done
cy tree... Done
ormation... Done
ditional packages will be installed:
n dictionaries-common emacs-common enchant-2 gstreamer1.0-gl
libadwaita-1-0 libaspell15 libcairo-script-interpreter2 libcdpa
libgraphene-1.0-0 libgstreamer-glib1.0-0 libgstreamer-plugins-bas
libgtk-4-media-gstreamer libgtksourceview-5-0 libgtksourceview
-0 libogg0 libopus0 liborc-0.4-0t64 libtheora0 libvisual-0.4-0
s:
lutils wordlist gvfs hunspell openoffice.org-hunspell | openoff
ikko libvisual-0.4-plugins opus-tools
packages will be installed:
n dictionaries-common emacs-common enchant-2 gnome-text-edito
ugins-base hunspell-en-us libadwaita-1-0 libaspell15 libcairo-s
libeditorconfig0 libenchant-2-2 libgraphene-1.0-0 libgstreamer-
ugins-base1.0-0 libgtk-4-1 libgtk-4-bin libgtk-4-common libgtk-
w-5-0 libgtksourceview-5-common libgudev-1.0-0 libhunspell-1.7-
libtheora0 libvisual-0.4-0 libvorbis0a libvorbisenc2
wly installed, 0 to remove and 0 not upgraded.
MB of archives.
ion, 54.4 MB of additional disk space will be used.
ive.ubuntu.com/ubuntu noble/main amd64 libaspell15 amd64 0.60.8
ive.ubuntu.com/ubuntu noble/main amd64 emacs-common all 3.0.5
ive.ubuntu.com/ubuntu noble/main amd64 dictionaries-common all
ive.ubuntu.com/ubuntu noble/main amd64 aspell amd64 0.60.8.1-1b
```

# Creating a Python Module (Windows): Part 2

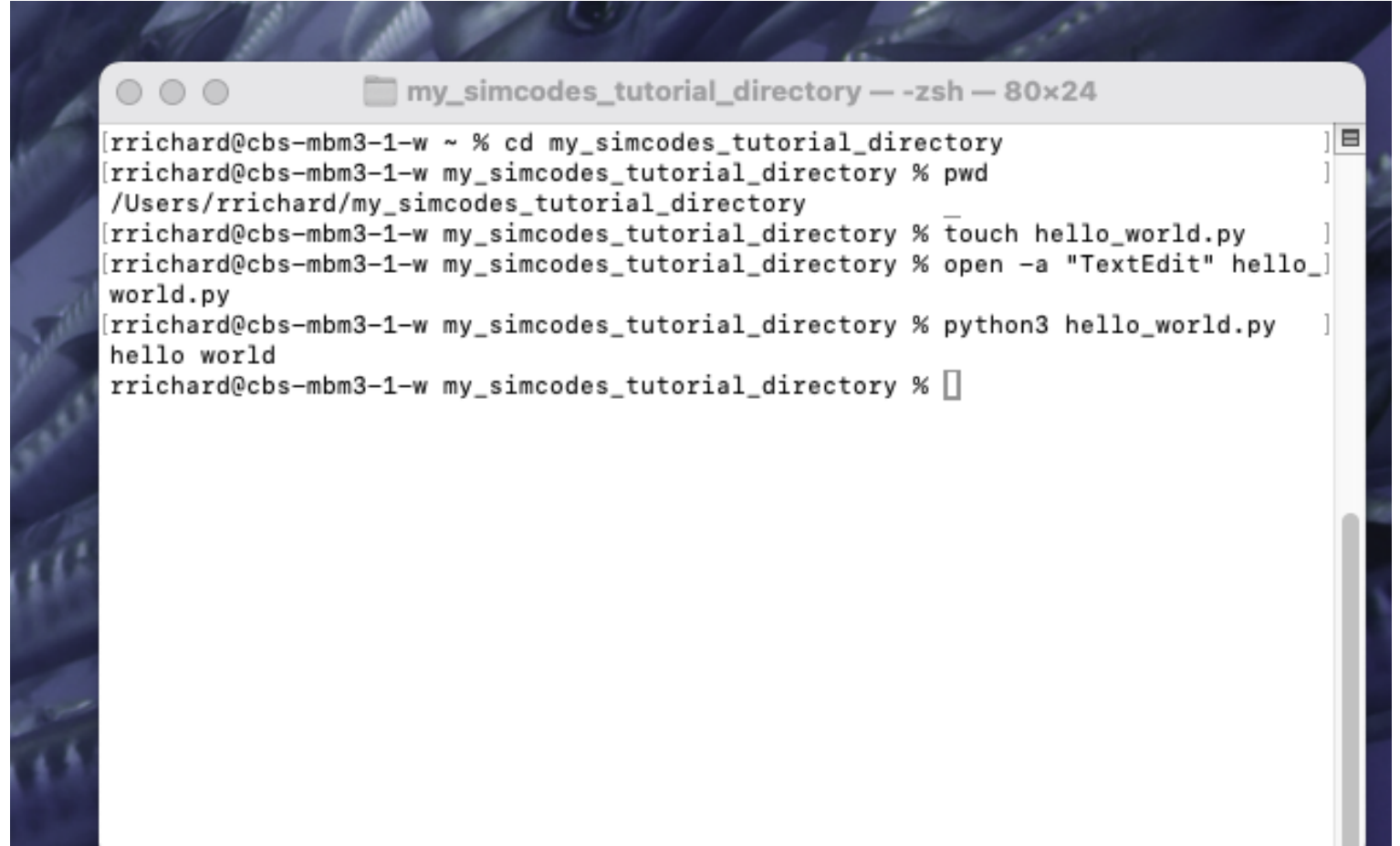
---

3. `gnome-text-editor hello_world.py`
  - (Do not have to create file ahead of time).
  - If this fails, Gnome Text Editor may be missing a file.
  - Run `sudo apt install gedit -y`
  - Try the command again.
4. Type `print('hello world')`, then ctrl + s (saves).



# Running Our Python Module

---

A terminal window titled "my\_simcodes\_tutorial\_directory — -zsh — 80x24" is shown. It displays a series of commands and their outputs. The user "rrichard" is at a machine "cbs-mbm3-1-w". The commands executed are: "cd my\_simcodes\_tutorial\_directory", "pwd" (outputting "/Users/rrichard/my\_simcodes\_tutorial\_directory"), "touch hello\_world.py", "open -a \"TextEdit\" hello\_world.py", and "python3 hello\_world.py" (outputting "hello world"). The prompt returns to the shell.

```
my_simcodes_tutorial_directory — -zsh — 80x24
[rrichard@cbs-mbm3-1-w ~ % cd my_simcodes_tutorial_directory
[rrichard@cbs-mbm3-1-w my_simcodes_tutorial_directory % pwd
/Users/rrichard/my_simcodes_tutorial_directory
[rrichard@cbs-mbm3-1-w my_simcodes_tutorial_directory % touch hello_world.py
[rrichard@cbs-mbm3-1-w my_simcodes_tutorial_directory % open -a "TextEdit" hello_
world.py
[rrichard@cbs-mbm3-1-w my_simcodes_tutorial_directory % python3 hello_world.py
hello world
[rrichard@cbs-mbm3-1-w my_simcodes_tutorial_directory % ]
```

# Objectives

---

- Know what a Linux terminal is.
- Know how to access a Linux terminal.
- Know the most important Linux terminal commands.
- Install Python.
- Write a “Hello World” Python module.
- Run the “Hello World” Python module.



# Software Development Environment (SDE)?

---

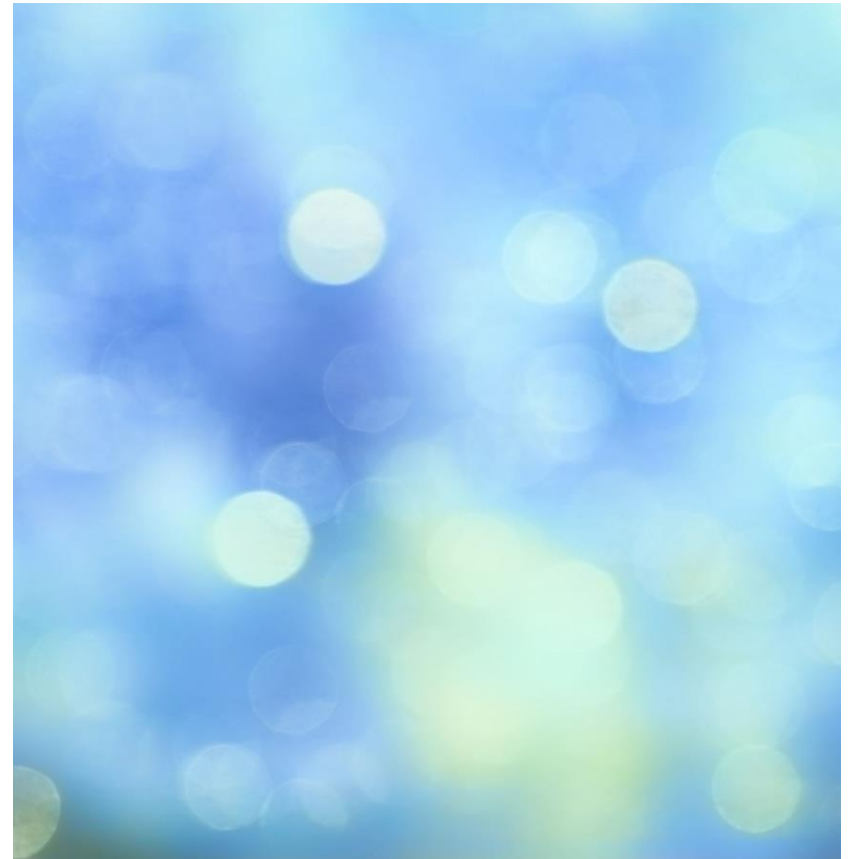
The title was “Setting Up A Software Development Environment” and we haven’t mentioned SDE yet...

An SDE is just a collection of tools to develop, test, and debug an application.

Terminal is arguably the most important tool for development.

Python comes with tools for testing and debugging. We’ll go over those tools in future lectures.

TL;DR you now have a bare-bones SDE!!!





# Acknowledgements

---

- MolSSI. I “borrowed” (i.e., stole) the content from them.
  - Installing terminal:  
<https://education.molssi.org/python-package-best-practices/setup.html>
  - Terminal: <https://github.com/msse-chem-280-2024/msse-chem-280-2024.github.io/>
- NSF for funding SIMCODES

