UCF Physics PHZ 3150: Introduction to Numerical Computing

Windows 10 Software Installation

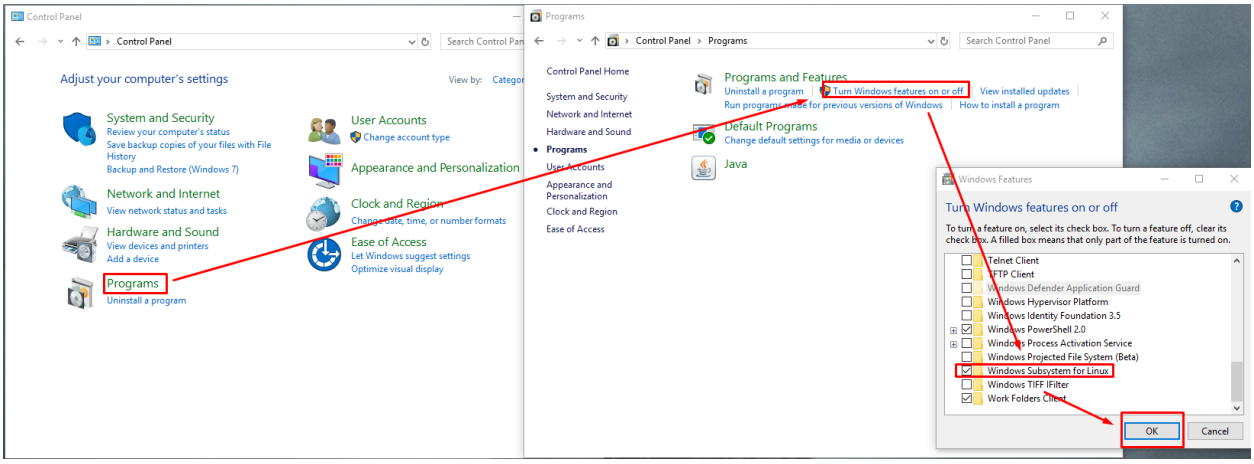
CAREFUL: If you cut-and-paste from a PDF document, likely all non-alphanumeric characters, including commas, apostrophes, quotation marks, underscores, dashes, hyphens, tabs, symbols like $\sim$, and so on will look right but work wrong. This is because typesetting programs use prettier versions of these characters, and the shell doesn't understand anything but simple ASCII characters. Either retype these symbols or just type the whole line by hand.
It is important to log what you do when you're doing system management! Especially log anything that deviates from these instructions.

1 BASH on Windows

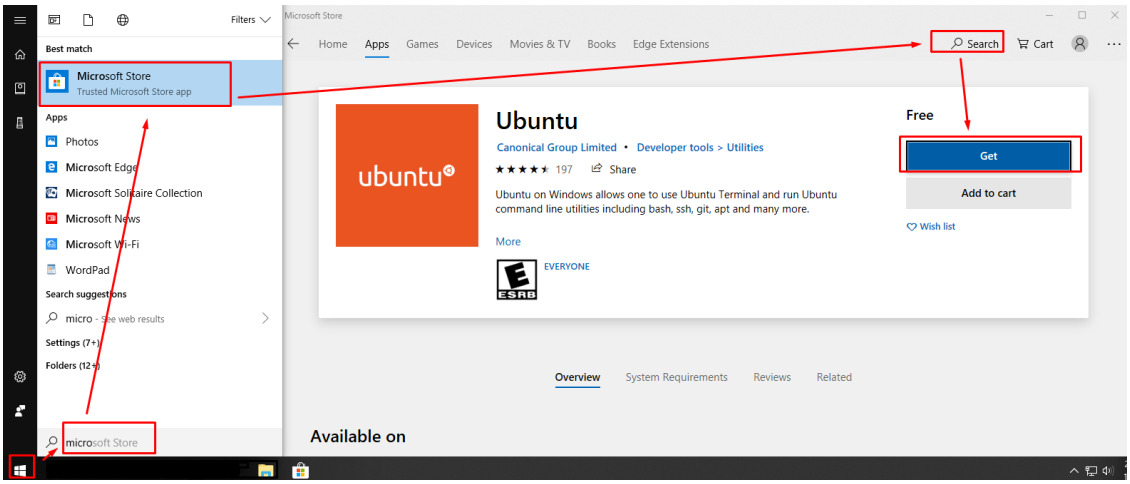
A goal of this class is for you to become familiar with command lines and shell scripting. The Windows Command Prompt does not have nearly the capabilities of the Unix shells. Luckily, Microsoft recognizes this and, in Windows 10, has added the ability to enable a Linux subsystem. The following is derived from a guide at:
https://www.howtogeek.com/249966/how-to-install-and-use-the-linux-bash-shell-on-windows-10/

1.1 Getting Ubuntu Linux and Its Shell

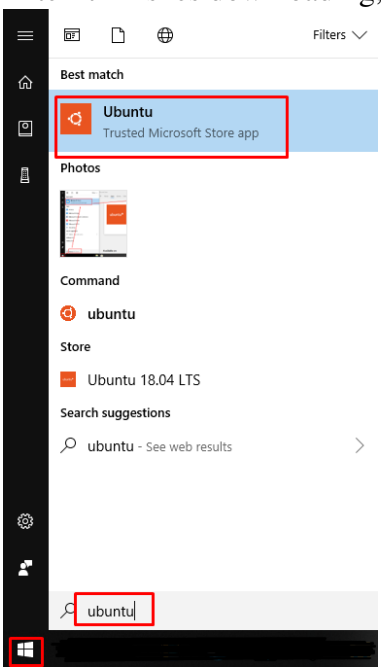
Note: This will only work on a 64-bit system. First, go to Control Panel → Programs → Turn Windows Features On Or Off and enable Windows Subsystem for Linux:



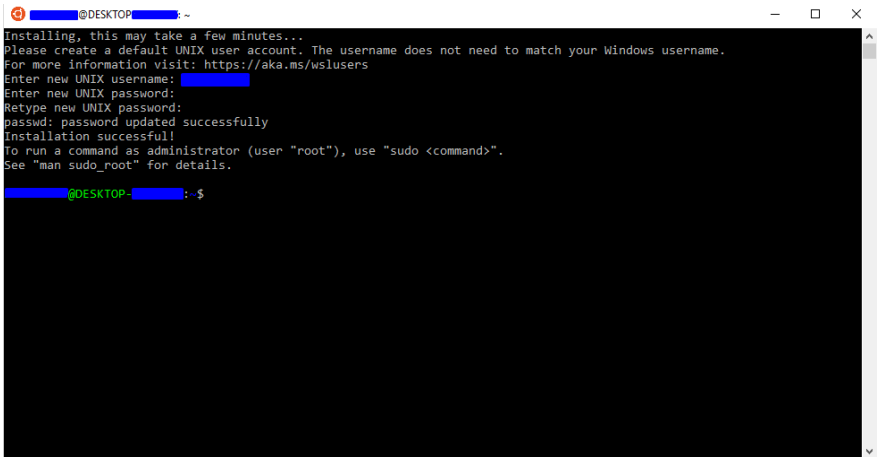
Restart your computer for this to take effect.
Get Ubuntu from the Microsoft Store:



After it finishes downloading, install it by launching Ubuntu:



It will take a few minutes to install. Afterwards, it will prompt you to set up a username and password. This is completely independent of your Windows account. You can make your username whatever you want, but please be professional and base it on your real name and/or initials. It should be **all lowercase** and no longer than 8 characters. After you enter a username, it will prompt you for a password TWICE, to make sure you typed it correctly.



You now have Ubuntu on your Windows machine!

1.2 Updating Ubuntu packages

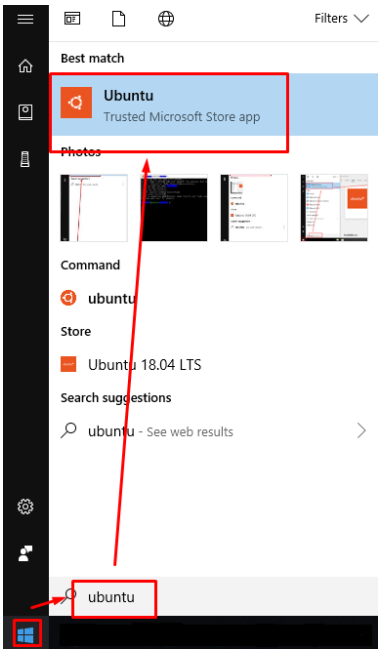
Any time you get a new operating system, the first thing you should do is update the software to get any security fixes or improvements since the version you installed. Run the following:

```
sudo apt update
sudo apt upgrade
```

Here, `sudo` (Substitute the administrator's Userid and DO something) will make the command run as an administrator ("root" in Unix-speak). It will ask for your password. The first command updates the database of packages available. The second actually updates the packages. You will have additional prompts to confirm doing the update. Press enter to these. It will take a few minutes to update.

1.3 Running BASH

Your BASH environment is separate from your Windows Command Prompt, and also from the "Git-BASH" prompt we'll install below. When you try to do something in a BASH terminal, make sure you are typing to the Ubuntu shell prompt, not Windows Command Prompt. You can launch BASH by pressing the Windows key and typing "ubuntu".



The Windows BASH is a simple environment. It does not run Linux's X Window System. Further, moving files between the environments can get tricky. So, we'll do our Python work in Windows and our shell work in Linux. We'll install Emacs and Git in both environments.

1.4 Getting from Linux to Windows and back

You will have a Windows home directory where you run Python and a Linux home directory where you do your shell work. You will want to move files (like your homework) between them, so we'll make routes between them. These are called "shortcuts" in Windows and "links" in Linux (more formally, "symbolic links").

1.4.1 Linux to Windows

For the link to reach Windows from Linux, we need to know your Windows home folder. From Linux, it looks like your entire Windows C: drive is under the `/mnt/c` folder. Open a BASH terminal (instructions above) and go (with the Unix `cd` command) to the folder containing user accounts:

```
cd /mnt/c/Users
ls
```

The `ls` command lists a directory (the Unix name for a folder). Take note of which listed item is your Windows home folder. We'll call this `<USERDIR>` from now on. For example, `<USERDIR>` might be `/mnt/c/Users/Sue Shih`. **When you see "`<USERDIR>`", replace it with the actual directory.**

Change directories to your Linux home directory, make a class directory, and make the link:

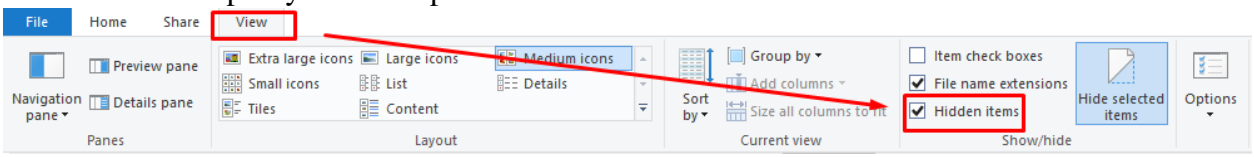
```
cd
mkdir phz3150
```

```
cd phz3150
ln -s "<USERDIR>" Windows
```

The quotation marks are necessary if there are any spaces or punctuation in the Windows folder name. Check that you have the link and that it points to the right place:
`ls -l`
This is the long form of directory listing, and it should show something like:
`lrw-rw-r-- 1 jh jh Jan 14 18:24 Windows -> /mnt/c/Users/Sue Shih`
To check whether it's right, do:
`ls Windows/`
It should list your Windows home directory (note the trailing `/`). The output might contain error messages and/or have strange colors. If you see any of your Windows files or folders, it worked.

1.4.2 Windows to Linux

Open a file browser and go to your Windows home folder (click on your name in the left column; do not go to your Desktop). If you do not have View Hidden Items enabled, enable it from the View tab at the top of your file explorer:



The Ubuntu stuff is deeply buried! From here, go to:

```
AppData\Local\Packages\
CanonicalGroupLimited.UbuntuonWindows_79rhkp1fndgsc\
LocalState\rootfs\home
```

Right click on your home folder and click Create Shortcut. Move that shortcut to your Windows-side class folder and rename it `Linux`. Visit it to make sure you see your Linux files, including your Linux-side class directory and the link inside that.

1.5 Customizing the Shell

The shell is an interactive programming language. You can control and enhance its interactive behavior! Whenever you start an interactive shell, it first runs several *shell scripts* containing configuration instructions, such as setting up your prompt. These files include `.profile`, `.bash_profile`, and `.bashrc` (the Bourne Again SHell ResourCe file). Which of these files run depends on multiple factors. They are all in your home directory. You may or may not have all of them, and that's ok.
First, we'll make a backup of your current shell configurations. Do this ONLY ONCE. If you repeat these instructions, do not repeat this step, or you will destroy your backup.

```
cd
mkdir .bash_prePHZ3150
cp -a .bashrc .bash_profile .profile .bash_aliases .bash_prePHZ3150
```

Don't worry about any "file not found" messages.
The Ubuntu default `.bashrc` file runs another file, if it exists, with your configurations, called `.bash_aliases`. Download the file:
WebCourses → Files → handouts → bash_aliases
It will be in your Windows-side Downloads folder. Copy it into the right place:

```
cp -a Windows/Downloads/bash_aliases ~/.bash_aliases
```

Adding a "." to the front of `bash_aliases` is important! If it asks if you want to overwrite an existing version, say no. This means you already have a `.bash_aliases` file. If it was there before you started the course, you'll need to merge the contents of the two files. Ask course staff if you don't know how.
The next time you start a shell, you'll have some useful configurations and new commands. We'll talk about these in class. If you want them right away, do:

```
source ~/.bash_aliases
```

in every terminal.

1.6 Moving Files

If you move a file from the Linux side to the Windows side through the links you made, it should work. Moving files from Windows to Linux is problematic: If it's done using the shell (i.e., you `cp` or `mv` through the link you made, it should work:

```
cp -a Windows/Downloads/bash_aliases ~/.bash_aliases
```

However, if you use Windows Explorer, it will not see the file on the Linux side, until you logout and login again. So, always use Linux commands to pull, rather than using Windows Explorer to push.

1.7 File Permission

We defined default permissions in `bash_aliases` (see line about "umask"), but you did some work before then and those files got "promiscuous" permissions (i.e., world-write enabled, `rwxrwxrwx` in long listings). To fix this, type:

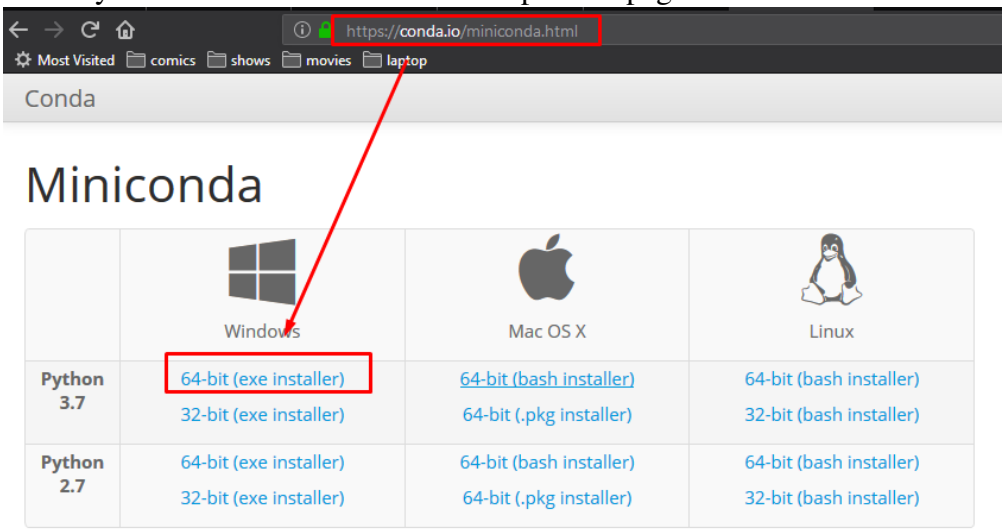
```
cd
chmod -R o-w .
```

This recursively changes the permissions of everything under your home directory to disallow writing by others (who are not in your group) on your computer.
Linux programs like the shell will automatically make new files without world-write permissions. However, Windows Explorer may still create files brought from the Windows side with promiscuous permissions. They will show up in bright green when you list them. You can change individual files with:

```
chmod o-w <filename or directory name>
```

2 Python

To install Miniconda, go to: `https://conda.io/miniconda.html` and download the Windows Python **3.7** 64-bit installer at the top of the page.



Click on the downloaded file to run it. Follow the prompts and use the default settings. You will be asked to read and agree to a license. You can scroll with down arrow or page down with the spacebar. At the end, type 'yes' to agree. Press Enter at the next prompt to install as a user (i.e., not administrator). Finally, type 'yes' and press Enter at the last prompt to integrate Miniconda into your environment.

2.1 Using Miniconda

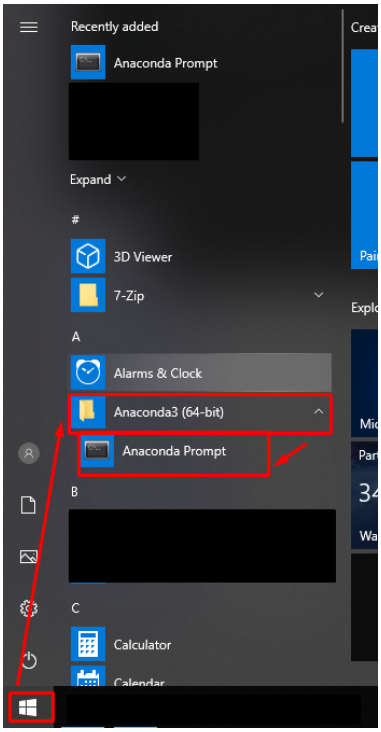
To run Python or to install or update more Conda packages, pull down the "Anaconda" menu in the Windows Start menu and click on "Anaconda Prompt."
From here, you can start a Python session with `python` (or `ipython`, once you install it, below) or use `conda` with:

```
conda <command> [package(s)]
```

To begin with, update all the Miniconda packages with:

```
conda update --all
```

To use the Spyder Integrated Development Environment to edit or run Python programs (after you install it, below), pull down the "Anaconda" menu in the Windows Start menu and click on "Spyder."

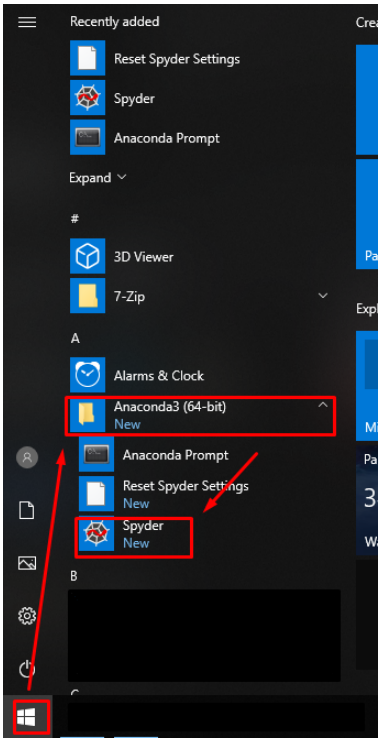


2.2 Add Packages to Conda

We will use conda to install some additional packages. You can install individual packages with:
`conda install [packagename]`
(This is called metasyntax; you replace `[packagename]` with the name of the package.) The packages you want are `numpy`, `scipy`, `ipython` (interactive python), and `matplotlib`:
`conda install numpy scipy ipython matplotlib`
Press Enter when prompted.

2.3 Spyder

Python differs from a language like C, in that code is not complied before being run. Rather, it is interpreted by the Python program as it runs. This makes it easy to start coding in a good text editor (like Emacs) and test lines of code in a Python interpreter as you write them. However, some prefer to use an Integrated Development Environment (IDE), which is a program that lets you edit the code, run it, and helps you debug any problems.
Spyder is a well known Python IDE. Install it with:
`conda install spyder`
You can run Spyder from the Miniconda folder in your Start menu:



Try starting it, to see if it works.

2.4 Jupyter

Jupyter is a python notebook that is very popular in industry. Like Spyder it lets you edit, run and debug a code, but in your favorite web browser. To install it use:

```
conda install jupyter
```

You should be able to run Jupyter from the Miniconda folder just like Spyder.

3 Emacs

Emacs is primarily a text editor, but to the experienced user is a full IDE (especially considering the various libraries that can be added), as well as just about anything (terminal, window manager, web browser, email reader, etc.).
We will need both Windows emacs (which opens its own window and uses the mouse) and Linux emacs (which could, if BASH on Ubuntu on Windows supported a window system, but it doesn't, yet).

3.1 Emacs for the Shell

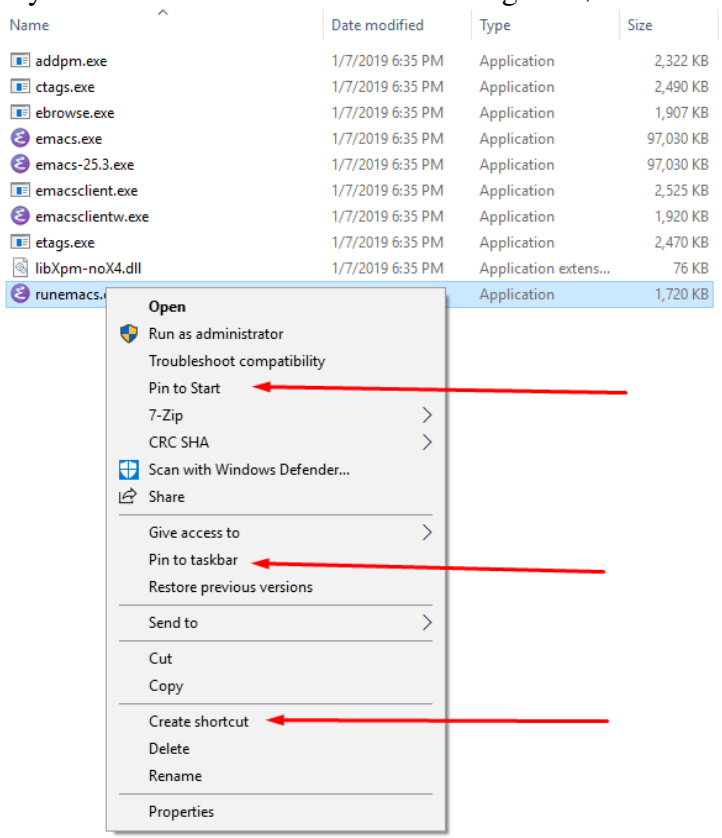
Open a BASH terminal (instructions above). Type:

```
sudo apt install emacs
```

and follow the prompts.

3.2 Emacs for Windows

To get emacs, go to `http://ftp.gnu.org/gnu/emacs/windows/emacs-25/` and download `emacs-25.3_1-x86_64.zip` (or whatever the latest version is now). Unzip the file you downloaded by right clicking and clicking `Extract All...`. This should generate a folder called `emacs-25.3_1-x86_64`. Move this folder wherever you want it (maybe in your home folder or Roaming). Once you move the folder, enter it and go into the subfolder named `bin`, right click on `runemacs.exe`, and click `Create shortcut`. Move that shortcut somewhere convenient. Alternatively, right click on `runemacs.exe` and pin to Start or Taskbar. This is just so you can launch Emacs without finding `bin/runemacs.exe` every time.



4 Git

We need to install two copies of Git, too, one to work in your Linux environment and one for Windows.

4.1 Git for the Shell

Open a BASH terminal (instructions above). Type:
`sudo apt install git`
and follow the prompts.

4.2 Git for Windows

Visit: `https://git-scm.com/download/win`
You likely want the **64-bit Git for Windows Setup**, unless you're running in 32 bits (why??). Download it and run it.
Choose the install path and other basic settings as you like.

Adjusting your PATH environment
As a command-line program, Git needs a command line. You have already installed BASH on Ubuntu on Windows, but this Git installation doesn't know about that. It can use the Windows Command Prompt, its own version of BASH, called Git BASH, or a version of Windows Command Prompt with some Unix behavior added. Choose the third option. This will still allow you to use either Git BASH or Windows Command Prompt. This option will replace some Windows Command Prompt functions' behavior with the Unix behavior, like the `find`, `kill`, and `sort` functions. Unless you are used to the Windows Command Prompt, this should not be a problem.

Configuring the line ending conversions
In DOS/Windows, text files have both carriage return and newline characters at the end of each line. In Unix and Max, they just have newline. Use **Checkout Windows-style, commit Unix-style line endings.** Then, you can checkout and edit on your machine without any problems, but your repository can push to repositories on other operating systems.

Configuring the terminal emulator to use with Git BASH
If you like the Windows console window, choose that one. Otherwise, go with the MinTTY terminal.

Configuring extra options
The default options should be fine.

Configuring experimental options
There possibly are useful tools here, but they need more testing. Use at your own risk!