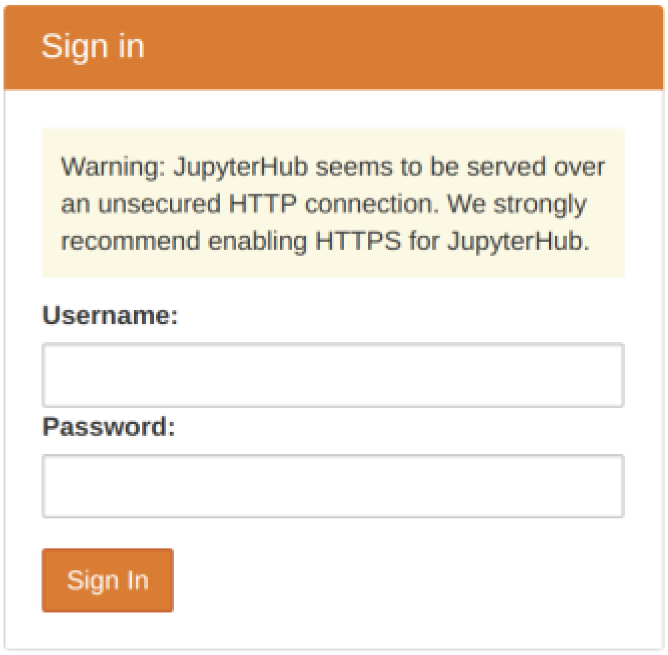
Introduction to the Jupyter Hub environment

Introduction

Jupyter Hub is an interactive multi-user environment to run Python Notebooks and Linux terminal sessions. It contains a file manager, a file editor, a notebook interface and a terminal interface.

Logging in Jupyter Hub

To enter Jupyter Hub, go to the following link: https://compbio-2023.ddns.net Upon opening the above link, you will be presented with the following login panel:


Sign in

Warning: JupyterHub seems to be served over an unsecured HTTP connection. We strongly recommend enabling HTTPS for JupyterHub.

Username:

Password:

where you should login with user a, where is the your UALG student ID. The initial password given by the instructor is the same for every student, so you should change it immediately upon login. To do so, visit the following link in the brower:

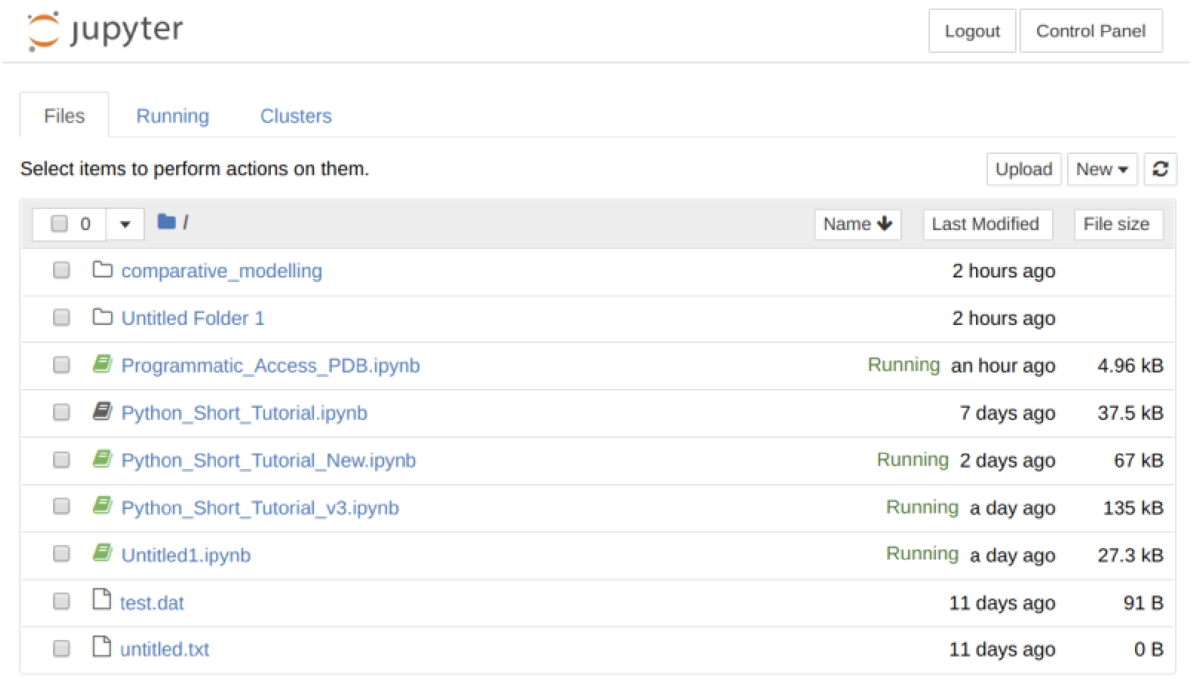
https://compbio-2023.ddns.net/hub/auth/change-password

and input the new password as prompted. (**NOTE:** you will be asked the password only once, so ensure you are really typing it as intended.)

Logout and log in again to test your new password.

The Jupyter Hub file manager interface

Upon logging in, you will be presented with the **file manager interface** of the Hub.



Select items to perform actions on them.

Name		Last Modified	File size
comparative_modelling		2 hours ago	
Untitled Folder 1		2 hours ago	
Programmatic_Access_PDB.ipynb	Running	an hour ago	4.96 kB
Python_Short_Tutorial.ipynb		7 days ago	37.5 kB
Python_Short_Tutorial_New.ipynb	Running	2 days ago	67 kB
Python_Short_Tutorial_v3.ipynb	Running	a day ago	135 kB
Untitled1.ipynb	Running	a day ago	27.3 kB
test.dat		11 days ago	91 B
untitled.txt		11 days ago	0 B

The file manager displays a list of a user's current files. Three types of file can be seen in the above image: folders (folder icon), notebook files (book icon) and a data files (sheet icon). Different things will happen when you click each type of file:

- **Folder:** clicking these will bring a new list presenting the files contained in the folder you just clicked.
- **Notebook:** clicking these will open the *Notebook Interface* for the file you just clicked (more on this below).
- **Datafile**: clicking these will open the *Editor Interface* for the file you just clicked (see below).

On the top right-hand side, you will find a drop-down name called "New", allowing you to create new Notebooks, Data Files, Folders or Terminal Sessions (we will discuss Terminal Sessions in greater detail below).

The Notebook Interface

If you clicked on a file with the notebook icon, it will open in the Notebook Interface, where you can edit, add, remove or execute *cells* in the notebook. Think of cells in the Notebook a little like snippets of programming code (here in the Python language) that can produce an output when executed, or data that can convey commentary and information. The best way to learn how notebooks work is to actually use one: try opening the notebook named "Introduction_to_Python.ipynb".

For now, notice that cells have a "In [number]" preceding them. You can "execute" (run the code) in a cell by clicking on it and pressing the `SHIFT` and `ENTER` keys simultaneously (`SHIFT+ENTER`). A new box will appear below the executed cell, with an "Out [number]" preceding it. As you may have guessed, this is the *output* produced by the cell upon running the code contained in it. You can also use the `ALT+ENTER` combination to run a code, but in this case a blank cell will be created right below the cell you just executed.

Not all cells contain Python code commands. Some have just (almost) plain text, and are used to write descriptions, explanations, headings, etc. These two types of cells are appropriately named *code cells* and *markdown cells*. To switch between the two types, click on a cell, press the `ESC` key, and then `Y` key (code cell) or `M` key(markdown cell) to switch the cell type.

On the menus you have various commands to insert, delete, copy and paste cells, and also to run sets of cells or execute the entire notebook.

It is important to note that code executed in a cell can affect the result of running cells *above* or *below* it. You must keep this in mind when trying to execute cells that are in the middle of a notebook (they may produce error messages complaining about missing variables, functions or libraries).

The Terminal Interface

If you choose "Terminal" on the "New" pull-down of the File Manager Interface, a new tab will open in the browser with a black screen with a prompt (something like `jupyter-xxxx@tljh-2023:~$` where `xxxx' is your login name) and a blinking cursor. This may appear daunting at first, but it is actually quite simple: the terminal is waiting for your commands. This type of setup is called a "command line interface".

Try typing `ls` and pressing the `ENTER` key. A list of the files in your Jupyter Hub home folder will appear, followed by a new line starting with the same prompt and blinking cursor - the system executed the `ls` command and is now waiting for a new command. That's all that there is to it, really: just typing commands, and watching the resulting output. Let's list a few very basic (and important) commands:

- `cd <folder>` - changes to the named folder below your current folder (don't actually type the "<" and ">", they are a notation meaning that you should actually replace "" with the name of your folder
- `ls <file/folder>` - without arguments (just `ls`) this command will list the contents of the current folder. If given a folder as argument, it will list the contents of that folder. If given a file as argument, it will simple list that file (or produce an error message if the file doesn't exist). By using the "wildcard character" `*`, you can list multiple files observing a defined pattern. Example: `ls *.ipynb` will list all files with extension `ipynb`, that is all Jupyter notebook files.
- `pwd` - "pwd" stands for "print working directory" and that is exactly what it does: it prints the name of your current directory (another name for folder). Good to avoid getting lost amid multiple folder changes.
- `cat <file>` - list the contents of a file. Be careful not to run this command on binary files (not containing text) or very large files, or you will most likely get your terminal stuck.
- `nano <file>` - brings up a simple editor to edit the contents of `<file>`, which is supposed to be a *text file*. Don't run this command on binary files.
- `cp <file> <file/folder>` - copy the first file to a new file in the same folder (it will have to have a different name) or to another folder, depending on whether the second argument is a file or a folder.
- `mv <file1> <file2>` - rename `<file1>` to the new name `<file2>`
- `rm <file>` - delete `<file>`. Be careful using this command, because *it cannot be undone*.
- `less <file>` - list contents of `<file>`, pausing at every page (use `SPACE` to move to the next page).

To navigate the folder tree, there are two important special folder names:

- `.` - means "the folder we are in"
- `..` - means "the previous (parent) folder"

Complete file paths are specified thus: folder1/folder2/folder3

For instance:

```
jupyter-pjmartel@tljh-2023:~$ cd folder1/folder2
jupyter-pjmartel@tljh-2023:~$ cd ..
```

will first move into `folder2` (a subfolder of `folder1`), and then move one folder up (or down, depending on your perspective) the folder tree, leaving us in `folder1`.

One very important fact about the command-line interface: Python code can be run directly, for instance:

```
jupyter-pjmartel@tljh-2023:~$ python3 my_script.py
```

will run the Python instructions contained in the "my_script.py" file (the ".py" extension is used to tell the computer these are Python files).

In this course we will use the command line interface to run various software applications and programs.

What do to next

Go to the browser tab where you have loaded the "Introduction_to_Python.ipynb" and study it through.