
Links

Etherpad: https://pad.carpentries.org/2023-03-20CodingSydney
Workshop page: https://adacs-australia.github.io/2023-03-20-Coding-Best-Practices-Workshop/
Github for workshop: https://github.com/ADACS-Australia/2023-03-20-Coding-Best-Practices-Workshop
Zoom info: Link https://unsw.zoom.us/j/86138935425?pwd=dXBpVC9nWnh5RnBSWmpGUlpRSUc0QT09

- Password: 721203

Example repo: https://github.com/PaulHancock/upgraded-system

Voting link for final day's content: https://www.menti.com/aluf3c4xrzn7 or www.menti.com with code **4426 7714**

Zhuochen (contact at NCI): zhuochen.wu@anu.edu.au

Need to Learn/Refresh yourself on Bash or Python?

The Software Carpentries have some excellent courses on these, they are linked below:
* Unix Software Carpentries: https://swcarpentry.github.io/shell-novice/
* Programming with Python Software Carpentries: https://swcarpentry.github.io/python-novice-inflammation/
* Plotting and Programming with Python Software Carpentries: http://swcarpentry.github.io/python-novice-gapminder/

Extra Comments

- What's your take on global variables? Avoid??
    - One Word: pain, except for constants. But define them properly and maybe in one place

    like astropy (speed of light is the best example of a constant that should be global).
    - Thanks. So anything that won't need updating.

- I have found that well documented and easy to use code has really taken off in astronomy. Think of e.g. ppxf, s-extractor, GalFit, profound, etc (some have thousands of cites). So it is worth making our code accessible, well documented, portable, etc.
    - Some people have used their code repositories to make Academic papers as well, take FHD (hopefully, PyFHD....shameless self promotion it's a WIP) and Bifrost as examples (they are on arXiv).

Introductions

T - Computer simulations of galaxy formation (Sevendust / my dog)

M - Hi, I'm a PhD student working with simulations of binary systems. I love dogs and my dream is to have a turtle that outlives me. I like bands like Tool, Dream Theather, Pearl Jam but also fan of electro swing, video games OST, anime openings.

P - Listening to Caravan Palace. I like good dogs (which is all dogs).

Currently I am an honours student studying gravitational lensing. I am currently obsessed with "We go down together"! I love all animals, but I don't actually have any pets. Maybe a guinea pig, haha

AK - I am a PhD student at UNSW, studying galaxy evolution. I've been listening to Joji, Bring Me The Horizon. Pet: Buffalo.

CF- I am an astronomer at UNSW working on galaxy evolution and formation. I mostly code in R, but love the functionality of reticulate to use python together with R :D. I I like the idea of no tech shaming and using the best tool for the job! :D I like listening to musicals and just saw "Joseph and the Amazing Technicolour Dreamcoat" this weekend in Capitol theater (highly recommend it)! An old fave classic! Seriously the happiest of musicals! In my spare time I like to do rock climbing and play clarinet! Oh, I have a cat named Peaches.

JD - Mostly Metal but if Spotify is to be believed Bo Burnham (a lot!) - Cat, I have one called Amirah, she is tiny and has a frog face, have you ever looked at something and kinda know maybe they're not thinkling anything at all, yea she looks like that constantly, silly thing - I'm a Software Developer at the Curtin Institute for Computation (CIC), I know many programming languages! (Python is one of my favourites)

VG - I am a post-doc at CSIRO (Marsfield). I listen a lot to Coke Studio, and I am cat person! And I'm attending online :)

N: I am a post doc at UNSW. Attending the workshop in person.

M - I like listening to Lo-Fi anime and video game soundtracks and I do not apologise for this. I am an

4th/3rd year Undergraduate student at UNSW. My favourite type of pet are Rats. They are intelligent, cute, and easy to care for.

M - Hi I am a postdoc at UNSW astro, attending in person, sitting at a table alone. I have listened to the same music most of my life so nothing new. I like maltese poodles, I watch dog videos all day. I come from an exoplanet atmosphere modeling background. Here I am on the Veloce team.

RB - Radiohead - Cats

M: I am a PhD student at UNSW working on generating computational spectroscopy data to help in the scientific search for life on other planets. I am currently listening to Carly Rae Jepsen and video game soundstracks while I am working. I am a cat person.

F: honours student - currently listening to Maisie Peters - Dogs

M - USyd PhD student. Listening to Muse, other people's pets (no responsibilty).

C: postdoc at UNSW, d

Y: Hi I'm a PhD student at USYD. I join over zoom. Cats & Dogs, 50-50

A - Hi, I am a PhD student at Macquarie University, just started it. I work on simulations of the CE phase in binary systems. I love dogs and have one back home, she's my best friend. I like pop, Bruno Mars, Maroon 5... (pretty basic, I know)

C - Hi, I am an incoming PhD student coming to Macquarie University to study Common Envelope (But I am still waiting for visa so I haven't arrived in Sydney yet!) I favourite artist is LindseY Stirling, and my favourite pet is dog (Although I never owned any pets, so I am not really sure.)

T - I'm a PhD student at USYD, researching links between galaxy kinematics and environment. Dogs are clearly the best type of pet, and it isn't close.

L: I'm a PhD student at UNSW, currently listening to a lot of Hands Like Houses. I have a pet axolotl, so I guess that's my favourite pet :)

A - currently into techno/psytrance artists, favourite pets are definitely rats

S: Hello everyone I'm a PhD student at USYD. I'm studying on Milky Way-like galaxies. I enjoy diving, walking, people watching, museums and travelling. I like cats as they are not interested in us and staying silence without barking! :P

H -- I'm a PhD student from USYD. My research is in Galactic Archaeology. My favourite type of pet is a dog
Hi there, everybody. I'm a postdoc at Macquarie. I work with computer simulations of binary stars. My favorite bands are Extreme and Aerosmith. And I have a black cat whose name is Sombra, which in Spanish means "Shadow""

K - Hello, I am a Masters student at Macquarie Uni, studying modelling the re-accretion of gas from a circumbinary disk around post-AGB binary systems. I mainly code in Python and use bash. I love cats (they are the best, obviously) and I love Metallica and the Foo Fighters!

sky_sim.py

wc -l catalog.csv
  999936 catalog.csv -- I only got 999936 stars - me too!

  The last line is:
  0999934,   14.765162,   41.444815 -- I just copy the code

  a = 1_000_000
a
Out[9]: 1000000. --- but when I remove the _ and rerun the code. It has 1000000 stars now. weird.
If you are getting this problem, remove the underscores in the nsrc number. I honestly can't tell you why this is hapenning right now. I'm on Python 3.10.6, can't tell you why it's not consistent.

P.S. Python 2 no likey those _ in 1_000_000 fyi (thats ok, I don't like pytho2 much either)

This underscore feature became available since Python 3.6

Version control question

One can actually use git in Overleaf, so we can use the same system for the paper writing.
Look at the older saved drafts in Overleaf. I tend to save a draft everytime I've made a major revision to a paper.
  Uncomment the excellent conclusion
If using overleaf go into the version history, copy paste it back
I use Overleaf and usually put written but unused paragraphs in a seperate tex file within the same project so that I can return to it later if I need it. Overleaf also tracks history, so if I forget to save the paragraph I could return to previous versions of the document.
Do not let ppl edit the overleaf! Send the pdf and ask them to annotate! (especially as not everyone is latex proficient)
Work in overleaf, comment out instead of deleting text, thought that does get unruly... or save snapshots of variosu versions at key points. Can then easily find old text in pervious versions.
My method for Word: Save copies of every previous versions (& commented versions from supervisors) and copy the excellent conclusions from previous docs to the current one
I used Overleaf to track the comments and changes of my co-authors. Usually, every one of them use different colors in their comments.
I save previous paragraphs as comments, if I feel the paragraphs are worth it.
When I am unhappy about a paragraph, I rewrite it below and save the original version. This means I will have multiple paragraphs serving the same purpose. I later select the sections I like in a separate document.
Comment out paragraphs if I'm making big changes, don't remove comments until final submission
I have only used Overleaf for this kind of thing, which I usually check check the history on and review the changes. Also, I/my collaborators have instead made comments instead of changing the document directly, and then the main editor would implement those changes (if required).
Re paper writing by different authors: You can have different conributors highlighting their changes in different colours, that has worked for me in the past

Coding Style

Seems it is unhappy about function "pow" being redefined by the import? Am I interpreting this right?
"redefined-builtin (W0622): *Redefining built-in %r*
  Used when a variable or function override a built-in."
Python has a pow function built into the language and is automatically imported any time you run a python program (there are many languages that do this). This pylint message is letting you know that you are overriding the built in python pow function with the math pow function. Ideally you are clear with your imports so you know where you're getting the function.

```
------------------------------------------------------------------
Your code has been rated at 9.47/10 (previous run: 8.95/10, +0.53)
```
 :D

Is there a best practice format for the module doc string? What should one include? **Paul's answer:** Intent, author, last change. Also list sub-modules with 1-line descriptions. List functions, classes and constants.

Your code has been rated at 10.00/10 (previous run: 10.00/10, +0.00) Woah!
My code rating went from -40 to 7.5 Yay!!! Update 9/10 | 9.5/10 | 10/10 :-p Woohoo!

Day 2

VG -- Good morning :)
Good Morning :)
Did I make a mistake testing.assert_all_close(answer, result, atol=1./3600)
E   AttributeError: module 'numpy.testing' has no attribute 'assert_all_close'
You should use assert_allclose, numpy is being annoying with their use of _ here.
thanks that works! it passed.... :cheer:

test_sky_sim.py:15: AttributeError

ON BREAK, return 13:30 sydney time.

VG:
```
(adacs) gup037@ASHMORE:~/Codes/adacs_workshop/mypackage$
(adacs) gup037@ASHMORE:~/Codes/adacs_workshop/mypackage$
(adacs) gup037@ASHMORE:~/Codes/adacs_workshop/mypackage$ pytest --cov=sky_sim --cov-report=term ./test_sky_sim.py
ERROR: usage: pytest [options] [file_or_dir] [file_or_dir] [...]
pytest: error: unrecognized arguments: --cov=sky_sim --cov-report=term
  inifile: None
  rootdir: /Users/gup037/Codes/adacs_workshop/mypackage
```

```
(adacs) gup037@ASHMORE:~/Codes/adacs_workshop/mypackage$ pytest --cov=sky_sim --cov-report=term ./test_sky_sim.py
==================================================================================== test session starts ====================================================================================
platform darwin -- Python 3.11.0, pytest-7.2.2, pluggy-1.0.0
rootdir: /Users/gup037/Codes/adacs_workshop/mypackage
plugins: cov-4.0.0, anyio-3.6.2
collected 2
items


test_sky_sim.py .
.
[100%]/Users/gup037/miniconda3/envs/adacs/lib/python3.11/site-packages/coverage/inorout.py:507: CoverageWarning: Module sky_sim was never imported. (module-not-imported)
  self.warn(f"Module {pkg} was never imported.", slug="module-not-imported")
/Users/gup037/miniconda3/envs/adacs/lib/python3.11/site-packages/coverage/control.py:858: CoverageWarning: No data was collected. (no-data-collected)
  self._warn("No data was collected.", slug="no-data-collected")
WARNING: Failed to generate report: No data to report.

/Users/gup037/miniconda3/envs/adacs/lib/python3.11/site-packages/pytest_cov/plugin.py:311: CovReportWarning: Failed to generate report: No data to report.

  warnings.warn(CovReportWarning(message))


---------- coverage: platform darwin, python 3.11.0-final-0 ----------


==================================================================================== 2 passed in 0.51s ====================================================================================
```

```
pytest --cov .
```

Do these ssh keys also work for scp?
yes, scp is also using ssh, if it doesn't work for some reason try using the -i identity file argument of scp

Favourite Python Modules/Packages

- I LOVE TQDM.NOTEBOOK

- astropy, numpy, scipy, pandas, matplotlib
- astropy, mpdaf, numpy, matplotlib, pandas, ppxf
- jax, astropy.io - fits
- import numpy as np
- from numpy import *
- matplotlib, numpy, pandas
- matplotlib, numpy, pandas, dill, glob, rdkit
- numpy, astropy, photutils, scipy, os, glob, sys, warnings, matplotlib, pandas, unagi, multiprocessing
- matplotlib
- emcee, dynesty, jax, corner, tqdm
- numpy, matplotlib.pyplot, pandas, sklearn, seaborn, sys, scipy, astropy, mesa_reader
- scarlet, eazy, astropy, numpy, mathplotlib, scipy
- f2py, seaborn, multiprocessing, sarracen
- astropy, numpy,
- psrchive, sigpyproc, selenium, sklearn
- astroquery, pymultinest
- tensorflow
- numba, plotly, pytorch, dask, taichi (havent tried it yet, but it looks awesome)
- cython
- plotnine - R's ggplot2 but in python

Package Wish List

- I wish there was the equivalent of reticulate in R but for python to use R within python!
- I wish there was a module that would install my modules (conda and pip can do this too!)
- I wish there was a module that makes my code faster (checkout numba, taichi, dask and jax!)
- A package to write my code for me like AI, based on an input goal for the code
- ^ - chatGPT? Yeah but inside python notebooks (no copy/paste) (ChatGpt API, maybe GitHub Copilot Extension?)
- I wish there was a package to search stackoverflow and replace my stupid functions with better ones from stackoverflow (I concur would be awesome) ...also ChatGPT....Copilot, they are trained from stackoverflow and GitHub. They are *sometimes* scary good at it.
- I wish there was a package that would invent variable names (Yes Please!)

The explainer for Entry Points inside Setup.py

The course currently has a way to run python scripts as a command in the terminal using a file that we made executable using a shebang and the chmod commands. This works, however it's harder than using entry points, both things will achieve what we want here. Entry_points however is considered more Pythonic (I dislike this word but it has its uses) as we can specify any function inside our entire package to be something that is executable from the terminal.

In our example, if you sky_sim.py file has a **if __name__ == 'main'** that contains the code to run sky_sim, put all that code inside a function call **main()**. So your sky_sim looks like so:

def main():

- **# main function code here**

if __name__ == '__main__':

- **main()**

Inside the setup.py file remove the scripts line:

#REMOVE ME
scripts = ['..']

And instead add the following into setup.py inside the setup function call:

setup (

- **...,**
- **entry_points = {**
    - **"console_scripts" : ["sky_sim="mymodule.sky_sim:main"]**
- **}**

)

What the above entry_points does is tell Python we want some commands available in the terminal, in this case we wanted one called 'sky_sim', after this we tell Python when using the command script, what function do we want to run. In this case we tell it its in mymodule.sky_sim:main, this is in the general format of <module>.<submodule>.<..submodules>:<function>.

In this case we called the function main, but the function name can be called whatever you want it to be. You can specify as many entry_points as you want, you can even set different console scripts to go to the same function (probably don't but you could).

Since we have changed setup.py we'll need to rerun pip install -e . (make sure you remember the -e, editable, which allows us to make changes to the package and not have to reinstall with pip).

From there, try out your new command

sky_sim

Once you have your python environments setup with your module installed, you can call this command in your terminal from any path on your machine because your Python path now has your command script. Pretty cool!

General Steps for Setting up SSH Keys on any machine to another Remote Machine (Supercomputer or Otherwise)

Before beginning these steps, make sure your SSH connection to the remote machine you are wanting to connect works and your password is valid and works.

1. Every machine (Windows, Linux and Mac) will have a .ssh directory in the home folder, in Linux/Mac this will be in **/home/<user_name>/.ssh (~/.ssh)** and in Windows it will be in **C:/Users/<user_name>/.ssh**. Depending on your machine connect to the .ssh directory inside your home directory.

```
cd ~/.ssh
```

Thankfully the contents of this directory will be consistent across all machines, typically by default you'll see only a known_hosts file which keeps track of the remote machines that have connected with it, successful or otherwise (this is why it asks for a SSH fingerprint everytime you log in for the first time). By the time we're done here we'll have 3 extra files, a private key file, a public key file and a config file.

2. Now let's create a key pair using the **ssh-keygen** command. The command will look like this:

```
ssh-keygen -t ed25519
```

The **-t** option, stands for **type**, there are different types of algorithms which can produce keys and certificates, in this case we are using ed25519. You generally don't need to worry about what type you use, by default RSA is chosen. Your organization or remote machine that you're connecting to might have certain rules or defaults about this, abide by these carefully when they are suggested (it's probably for a good reason).

When the above command gives you prompts, in this case you can almost ignore them and keep hitting enter until the command does no more prompts, but in case you want to do something custom the first prompt will ask for the filename you want the keypair to have, the second will be a passphrase for the key, and the third prompt will be a confirmation of the passphrase. Typically, you leave the prompts empty which means no more passwords or passphrases when logging into a remote machine with the keys. If everything has gone right the output will look something like this:

```
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/skywatcher/.ssh/id_ed25519):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/skywatcher/.ssh/id_ed25519
Your public key has been saved in /home/skywatcher/.ssh/id_ed25519.pub
The key fingerprint is:
....
The key's randomart image is:
+--[ED25519 256]--+
...
+----[SHA256]-----+
```

If you run the ls (list) command inside the .ssh directory you should see the keys generated as id_ed25519 (private key) and id_ed25519.pub (public key)

3. In the next step we are going to copy the public key to the remote server, there are many ways to do this, but the easiest way is to use the **ssh-copy-id** command.

```
ssh-copy-id -i ~/.ssh/id_ed25519.pub <user>@<host>
```

The **-i** argument stands for **identity**, we want to use this option to point the command to the public key (identity file) we just created. During this step it will ask for your password and then copy the public key to the authorized_keys file inside your home directory of the remote machine.

4. Now test your new SSH keys, by trying the same SSH command you used to test the connect.

```
ssh <user>@<host>
```

If all went well, you'll connect without having to put in a password.

5. From here you can stop, but I'd advise taking it a step futher, because deep down inside we all want to do less keystrokes to get the same result. We are going to create and edit a config file inside the SSH file which will tell your machine when using SSH with this host, follow the configuration inside the config file, as a bonus, it also makes using ssh easier!

Inside the .ssh directory create a config using the touch command:

```
touch config
```

Next you're going to open with your favourite editor and type in the following:

```
# The Host can be called whatever you want, I'd advise using something more meaningful than mine
Host pleasedontbeonfire
   # The Hostname should contain only the host address you're connecting to
   Hostname <host>
   # The User is you, the user
   User <user>
   # The IdentityFile is the private key we created during the ssh-keygen
   IdentityFile ~/.ssh/id_ed25519
```

A more practical example is using Gadi:

```
# The Host can be called whatever you want, I'd advise using something more meaningful than mine
Host gadi
   # The Hostname should contain only the host address you're connecting to
   Hostname gadi.nci.org.au
   # The User is you, the user
   User jd9617
   # The IdentityFile is the private key we created during the ssh-keygen
   IdentityFile ~/.ssh/id_ed25519
```

From here you can now connect to your ssh remote using with:

ssh pleasedontbeonfire

(or if you did the more practical example)

ssh gadi

If you're *particularly* adverse to keystrokes, you can take it even futher and make an alias inside your .bashrc inside your home directory.

alias gadi="ssh gadi"

Which means you can ssh to a remote machine like NCI using 4 characters!

gadi

Where To Put Your Constants in a Python Package

There are three good places to put constants in a Python package, and the answer to where is the best place to put them, is it depends.

- If you only use the constant once, in one module inside a Python package, then the best place is likely to be inside that module, or inside the __init__.py of that module.
- If you use the constant in more than one part of the package, there are two main places to put constants
    - You can create a constants module, where you have a python file called constants.py containing all variables that are only uppercase. This means you'll have a file that might look like this:
            - **constants.py**
    - **FOO = 1**
    - **BAR = 2**
    - To use the constants inside another module, you'll import it and use the constants like so:
    - **import constants**
    - **constants.FOO**
    - The other way is to put all the constants inside the**__init__.py** of the modules that require or inside the **__init__.py** for your entire package
    - Both of the ways above are good practices, it entirely depends on your use case as to what is more appropriate. If the constants need to be used by other people, then perhaps a module is best.
- If the constants aren't *strictly* constant, consider putting the constant as an argument to your program using **argparse** with the constant you set as the default. Using **configargparse** in combination with **argparse** also allows you to create a config file for your argparse, which means you can put your constants in a YAML or JSON file. This can be powerful because it can also give you an easy way to save your configs for your programs, making troubleshooting for others using your code a little easier, while also making it easier to use your package.

I am proudly a sticky tape programmer!!! Sometimes post-it glue kinda programmer even!

I feel sometimes learning a new tool can take some time and seem like low reward (in terms of optimisation) when it is first done. But once it is learned, it can be easily applied to other projects and thus may still be worth taking the time to learn in the end.