Good coding practices (#GCP) Tips to make your code better for you and others



Vanessa Moss ASTRON

IMAGE CREDIT: A. CHERNEY



Introduction

- My title: observatory astronomer, based at ASTRON in the Netherlands
- Previously **affiliated** with University of Sydney, CSIRO, Sydney Observatory in Australia
- **NCSS Challenge** during 2nd year uni (Python), previously Matlab or more basic computing
- Astronomers writing code vs. programmers developing for astronomy (**different skills**)
- **Code-savvy astronomers** help bridge the gap!

ASTERICS-OBELICS 2019 (V. Moss, @cosmicpudding)



Selfie with a confirmed peryton-generating microwave

This talk

- **Goal:** to introduce you to some guiding principles that will help make your code better
- First half from a talk by **Zheng Meyer-Zhao**, given at last year's school, second half is new
- The last 60 min of this talk will be spent on an exercise getting you to put some good coding practices into use
- There is no ***one*** way to code things*, but there are definitely good ways and bad ways to approach programming!



Overall guidelines With credit to Zheng's 2018 talk







1. Write programs for people

- Write programs for people, not computers
- A program shouldn't need people to hold many facts in their head at once to understand it - good clear comments help with this!
- Variable names should be consistent, distinct and meaningful to aid with reading the code
- Keep a **consistent** code style/formatting (there are various styles, but consistency is key!)

```
import os
import sys
from astropy.io import ascii
import matplotlib.pyplot as plt
# Read RA/dec values from table
table data = ascii.read('table.txt')
ra list = table data['ra']
dec list = table data['dec']
# Plot the results as a scatter plot
fig, ax = plt.subplots(figsize=(10,6))
ax.scatter(ra list, dec list)
# Save the results in a PNG
plt.savefig("radec plot.png", dpi=200,
bbox inches='tight')
```



2. Develop pragmatically

- Approach development of a new script or package in a pragmatic way
- The amount of effort should be **proportional** to the scope of the code, and its legacy value
- Fast code, good code, or cheap code: pick two
- Whatever time you think the code will take to develop, **double it** (at least)
- Best learnt over time, but invest the right **amount of effort** for different parts of the code



3. Let the computer do the work

- Make use of the tools that exist to be as efficient as you can
- Make the **computer** repeat tasks (e.g. cron job)
- Use **IDE/build tool** to make workflow efficient \bullet
- "If you've only got a hammer, everything looks like a nail" = use the **right tool** for the job
- **Pycharm/Sublime**, etc are good for quick runs of code, **iPython** for interactive commands

	atdb_parsets.py	UNREGI
< >	atdb_parsets.py × functions.py ×	
1 2	<pre># APERTIF PARSET GENERATOR ATDB VERSION 1.2 (atdb_parsets.py) - now with ARTS # Input: source text file</pre>	 A stand of the sta
3	# V.A. Moss 19/12/2018 (vmoss.astro@gmail.com)	The subscription based in the subscription of
4	author = "V.A. Moss"	 A start of the sta
5	date = "\$19-dec-2018 17:00:00\$"	
6	version = "1.2"	Barrianti den et ante
7		Allers in constants and and constants with and the constant of the and the constant of the constant of the constant of the and the constant of the constant of the constant of the and the constant of the constant of the constant of the constant of the and the constant of the constant of the constant of the constant of the and the constant of the constant of the constant of the constant of the and the constant of the and the constant of
8	import os	Statut for all factor and and the all factor and and the all factor and
9 10	<pre>import sys from modules.beamcalc import *</pre>	B. Same and Price and Same And Sam And Same And Same A
10	from datetime import datetime,timedelta	and design
12	from astropy.io import ascii	There is any different starting of least there are also as a second start are also as a second start are also as a second start
13	import numpy as np	A loss of the second se
14	from argparse import ArgumentParser, RawTextHelpFormatter	The section of the se
15	<pre>from modules.functions import *</pre>	If any set of the set
16	import time	s for it courses
17		
18	<pre>def main():</pre>	1 de la comercia presente de la esta esta esta esta esta esta esta esta
19		and a feature of the second seco
20	The main program to be run.	
21 22	:return:	And a standard of the standard
22		
23	# Time the total process length	
25	<pre>start = time.time()</pre>	An alterna in control on a second second second sec
26		Party and a second
27	# Parse the relevant arguments	and - Ans and - Ans and - Ans
28	<pre>parser = ArgumentParser(formatter_class=RawTextHelpFormatter)</pre>	
29	<pre>parser.add_argument('-f', 'filename',</pre>	The second secon
30	<pre>default='input/start.csv',</pre>	Linder soldt solder soldtast - austalistik soldtast - fastistik Ling for holder contactor
31	<pre>help='Specify the input file location (default: %(default)s)')</pre>	2.7.15.0%
32	<pre>parser.add_argument('-m', 'mode',</pre>	
33	<pre>default='imaging', halp=!Specify.ubsther_mode_is_imaging(SC1/SC4 (default.sk(default)s)!)</pre>	and the second s
34 35	<pre>help='Specify whether mode is imaging/SC1/SC4 (default: %(default)s)') parser.add_argument('-t', 'telescopes',</pre>	
36	default='23456789ABCD',	A second se
37	help='Specify which telescopes to include (default: %(default)s)')	The second secon
38	<pre>parser.add_argument('-c', 'cluster_mode',</pre>	And a second sec
39	default='ATDB',	
40	<pre>help='Specify which ARTS cluster mode, either standard/ATDB (default: %(default)s)')</pre>	
['sour	ce', 'ra', 'dec', 'date1', 'time1', 'date2', 'time2', 'freq', 'weight', 'sbeam', 'ebeam', 'pul	sar', 'be



4. Make incremental changes

- Small constant changes are the best way to prevent destabilising your code
- Work in **small steps**, and if your code is in version control, commit these changes often
- Which also means: **use version control**!
- Version control is **critical** for code shared with others, but is also incredibly useful for your own code for history, backups, deploys, etc...

Wrong variable name from dict Vanessa Moss authored and Vanessa Moss committed 9 days ago	武 5f38af6
Use new function to identify calibrators from a previous day	武 3cec775
Vanessa Moss authored and Vanessa Moss committed 9 days ago	
Add previous day functionality	武 cdd7276
Vanessa Moss authored and Vanessa Moss committed 9 days ago	
Add archived status check to calibrators	武 df89202
Vanessa Moss authored and Vanessa Moss committed 9 days ago	
ommits on Mar 26, 2019	
Add print of return to verify contents	武 38b8a1c
💭 Vanessa Moss committed 11 days ago	
ommits on Mar 25, 2019	
Cannot know success for errored	🛃 d4aefc8
💽 Vanessa Moss committed 11 days ago	
bracket typo	武 01e83a4
💭 Vanessa Moss committed 11 days ago	
Also for the one case	武 db36daf
💭 Vanessa Moss committed 11 days ago	
Accept new pipeline response, add slack id functionality	武 ca0e415
Vanessa Moss committed 11 days ago	
Test tagging people in Slack messages +1	🔁 19a095a
Vanessa Moss committed 11 days ago	
Test tagging people in Slack messages	武 8002ed8
Vanessa Moss committed 11 days ago	
ommits on Mar 22, 2019	
Deal with case when one calibrator is a None type	武 0b57878
Vanessa Moss committed 14 days ago	
ommits on Mar 20, 2019	
Set any other cal to none if only length one, when others are several	会 244946f
Vanessa Moss authored and Vanessa Moss committed 16 days ago	
Missing a break for calibrators before target	武 c2533d3
Vanessa Moss authored and Vanessa Moss committed 16 days ago	
	🔂 8d81d5d



5. Don't repeat yourself

- Repetitive code is usually redundant, and is really difficult to both read and debug
- If you find yourself **copying and pasting code**, then this usually means it would make a good function or module
- **Re-use** code instead of rewriting it: identify code that you need often, and make that into easily-accessible functions (when modular)
- That doesn't mean you shouldn't look for ways to **improve** the re-used code though!



6. Plan for mistakes

- If it works the first time, be suspicious
- No matter how carefully you program, there will always be **bugs**, often in unusual places
- For debugging, print statements* are extremely useful to check if the code does what you think it should be doing (*better: logging)
- Make use of **existing libraries** for testing (e.g. off the shelf libraries)
- Use bugs as good tests (e.g. **unit tests**)

ASTERICS-OBELICS 2019 (V. Moss, *@cosmicpudding*)



See: <u>https://nedbatchelder.com/text/test0.html</u>

7. Make it work, then optimise

- You should optimise software after it works correctly, based on the use case required
- The first goal should always be to get the code **working** - it is rarely the case that the design from the beginning limits optimisation
- Use profilers to identify bottlenecks e.g. **SnakeViz** (<u>https://jiffyclub.github.io/snakeviz</u>)
- Write code in a **high-level** language, and avoid re-inventing the wheel where possible
- But also optimise **as you go**, where you can!

ASTERICS-OBELICS 2019 (V. Moss, @cosmicpudding)



a colle	à tattima		å	A		aren.
ncalls	tottime	 percall 	cumtime	· • •	filename:lineno(function)	
5094	24.56	0.004822	24.56	0.004822	~:0(<method '_socket.socket'="" 'recv_into'="" objects="" of="">)</method>	
4	0.437	0.1093	0.437	0.1093	~:0(<built-in builtins.print="" method="">)</built-in>	
254	0.1008	0.0003969	0.1008	0.0003969	~:0(<method '_io.fileio'="" 'read'="" objects="" of="">)</method>	
11	0.08377	0.007615	0.08377	0.007615	~:0(<method '_socket.socket'="" 'connect'="" objects="" of="">)</method>	
81	0.06313	0.0007794	0.06313	0.0007794	~:0(<method '_sre.sre_pattern'="" 'search'="" objects="" of="">)</method>	
11	0.04632	0.004211	0.04632	0.004211	decoder.py:345(raw_decode)	
254	0.03378	0.000133	0.03378	0.000133	~:0(<built-in marshal.loads="" method="">)</built-in>	
1219	0.03175	2.605e-05	0.03175	2.605e-05	~:0(<built-in method="" posix.stat="">)</built-in>	
25/24	0.03019	0.001258	0.03313	0.00138	~:0(<built-in _imp.create_dynamic="" method="">)</built-in>	
11	0.02747	0.002497	0.02757	0.002507	~:0(<built-in _socket.getaddrinfo="" method="">)</built-in>	
937/935	0.02116	2.263e-05	0.04039	4.32e-05	~:0(<built-in builtinsbuild_class="" method="">)</built-in>	
254	0.01735	6.832e-05	0.1182	0.0004652	<frozen importlibbootstrap_external="">:830(get_data)</frozen>	
651	0.01573	2.416e-05	1.408	0.002162	~:0(<method '_io.bufferedreader'="" 'readinto'="" objects="" of="">)</method>	
5094	0.01497	2.939e-06	24.6	0.004829	socket.py:572(readinto)	
283/1	0.0115	0.0115	25.8	25.8	~:0(<built-in builtins.exec="" method="">)</built-in>	
5094	0.01043	2.048e-06	0.01043	2.048e-06	~:0(<method '_checkclosed'="" '_ioiobase'="" objects="" of="">)</method>	
651	0.00743	1.141e-05	1.444	0.002218	response.py:375(read)	
651	0.007301	1.122e-05	1.42	0.002182	client.py:438(read)	
11	0.006949	0.0006317	0.006949	0.0006317	~:0(<built-in _scproxyget_proxy_settings="" method="">)</built-in>	
521	0.005784	1.11e-05	0.0515	9.885e-05	<frozen importlibbootstrap_external="">:1233(find_spec)</frozen>	
7376	0.005203	7.054e-07	0.005266	7.139e-07	~:0(<built-in builtins.getattr="" method="">)</built-in>	
312/85	0.005096	5.995e-05	0.01512	0.0001778	sre_parse.py:469(_parse)	
5105	0.004901	9.6e-07	0.004901	9.6e-07	socket.py:611(readable)	
						_

e.g. "First do it, then do it right, then do it fast"





8. Document what you've done

- Don't trust your future self to remember what you coded and why
- Documentation should favour design + purpose over mechanics, but all are important
- Block comments above code can be helpful for keeping track of what sections do (*docstrings*)
- **Revisit** code even after it works (plan this!), and **refactor** to make the flow clearer/better
- Keep documentation **close** to the software

ASTERICS-OBELICS 2019 (V. Moss, @cosmicpudding)

Revamped visualisation of APERTIF MS

Manage topics						
🗇 175 commits	ំ 1 branch	\bigcirc 0 releases	초 1	contributor		ა⊉ু MIT
Branch: master - New pull	request		Create new file	Upload files	Find File	Clone or downlo
Vanessa Moss and Vanes	sa Moss Comment out the block	about bad elements		Late	st commit 5	f8157a on 13 Dec 2
modules	Comment out the block	about bad elements				4 months
	First commit to new rep	ository				7 months
README.md	Update readme with exa	imples				5 months
vam_plots.py	Okay, all good, beam pa	ttern reads now				4 months

E README.md

vam_plots: Visualisation of APERTIF MS

Suite of software to produce various visualisations of APERTIF measurement sets.

Basic usage

The package can be run with minimal commands using the defaults

python vam_plots.py -f 181010001

This contains built-in assumptions about the location of the data (e.g. /data/apertif/) and beam-pattern (e.g

An example of a more complicated command:

-f 181010001 -b element -p /data/moss/ -L 0.0 -H 600.0 -m 2.0 -x XX,YY

parameters, to only use half of the system-detected CPUs and to generate plots for both XX and

Output

The package currently generates 18 inspection plot images and 4 pickle files.

Contact

For help: vmoss.astro@gmail.con



9. Collaborate

- Your code will always find improvements when others try to understand or use it
- Use pre-merge code reviews (e.g. pull requests) for a **double-check** on what you have changed
- **Pair-programming** can help a lot when trying to implement new or complicated code
- Use **issue-tracking tools** to keep track of progress (e.g. JIRA, Redmine, Github issues)
- Ask people to try **using your code**, in order to find out what you should change or improve

ເປ	1 Open	✓ 175 Closed	Author -	Labels -	Projects -	Milestones -	Reviews •	Assignee -	
۶		shadow flaggi adebahr was merge							
۶			eline should reco merged 8 days ago • /	-	< .				
۶		ault tolerant se tammojan was merg	elfcal × ged 8 days ago • Appro	oved					
۶	-		see cosmicpudd ged 9 days ago • Appro		#7e7637a <mark>×</mark>				
۶		imaging empty tammojan was merg	chunks × jed 8 days ago • Appro	oved					
۶	-	e to robust=0 i tammojan was merg							
۶	-	es of paramete tammojan was merg	ers for 300MHz × ed 9 days ago						
۶		oved phase unw adebahr was merge	rapping for selfc	al QA plots	×				
۶		e ccalqa in star tammojan was merg							
۶		• •	os to start_pipelir ed 11 days ago • Appro		lue ×				
۶		eams × tammojan was merg	ed 11 days ago						
۶		-by-one error i tammojan was merg	n scal line flaggin ed 11 days ago	ıg ×					
۶		e except × tammojan was merg	ed 11 days ago						
۶		tammojan was merg	ed 11 days ago						
۶		nvert io × tammojan was merg	ed 12 days ago						
۶		gging of ghosts tammojan was merg	s (is by default di ed 12 days ago	sabled) ×					
۶	-		e.py to deactivate merged 12 days ago •		e ×				
۶		el ccal × tammojan was merg	ed 12 days ago • Appro	oved					
0	Config	file pipeline ×							



Specific coding tips These will be important in the exercise!





Argparse for argument handling

- Python allows you to pass variables as arguments after the code, e.g. **sys.argv[1]**
- Argparse is a **better way** to handle input variables into your scripts, giving you more control over their types and format: https:// docs.python.org/2/howto/argparse.html
- It also enables users of the script to get an **overview** of what each parameter means: e.g. python vam_plots.py --help
- As part of the **exercise**, you will add a (long) list of arguments to an argparse formatter :)

ASTERICS-OBELICS 2019 (V. Moss, *@cosmicpudding*)

```
parser = ArgumentParser(formatter_class=RawTextHelpFormatter)
parser.add_argument('-p', '--path',
       default='/data/apertif/',
       type=str,
       help='Specify path location of target folder (default: %(default)s)')
parser.add_argument('-f', '--folder',
       default='181130004',
       type=str,
       help='Specify path location of target folder (default: %(default)s)')
parser.add_argument('-L', '--low',
       default=200.,
       type=float,
       help='Specify amp colorscale low-value (default: %(default)s)')
parser.add_argument('-H', '--high',
       default=1000.,
       type=float,
       help='Specify amp colorscale high-value (default: %(default)s)')
parser.add_argument('-PL', '--pmin',
       default=-90.,
       type=float,
       help='Specify phase colorscale low-value (default: %(default)s)')
parser.add_argument('-PH', '--pmax',
       default=+90.
       type=float,
       help='Specify phase colorscale high-value (default: %(default)s)')
parser.add_argument('-r', '--rfiflag',
       default='"< 1000MHz"',</pre>
       type=str,
       help='Specify properties of RFI-flagging (default: %(default)s)')
parser.add_argument('-b', '--beam_pattern',
       default=None,
       type=str,
       help='Specify whether element or compound beams (default: %(default)s)')
parser.add_argument('-m', '--multi_factor',
       default=1.0,
       type=float,
       help='Specify what factor to divide available CPUs by (default: %(default)s)')
parser.add_argument('-x', '--pol',
        type=str,
       default='XX',
       help='Specify which polarisation/s to image (default: %(default)s)')
parser.add_argument('-t', '--file_type',
        type=str,
       default='png',
       help='Specify what file-type save images as (default: %(default)s)')
parser.add_argument('-s', '--slices',
       default=False,
       action = 'store_true',
       help='Specify whether to use the time slices method (default: %(default)s)')
# Parse the arguments above
```

args = parser.parse_args()



Make the code importable

- Scripts are useful, and by default what we tend to write for uses in astronomy (e.g. a linear flow to do a specific task, such as make a plot)
- If you make the bulk of your code **a function** (**skyviewbot()** for example), you can use it as a function as well as running it with default settings as normal Python code
- This means the functionality can then be called by another script, as well as being used on its own - so you've made your code **importable**
- This is done **by default** in the exercise later

ASTERICS-OBELICS 2019 (V. Moss, @cosmicpudding)





def skyviewbot():

Modularisation and functions

- Whenever you use some calculation or more complicated code **more than once**, you should make it into a function
- Functions enable **much more flexibility** because you break down the "function" of that command into its input parameters
- Often you can put the functions at the top of the script, but when there are many, it is neater to store them in a dedicated **functions.py** or so
- You'll turn code into a function in the **exercise**

ASTERICS-OBELICS 2019 (V. Moss, @cosmicpudding)



https://phiresky.github.io/emojidome/



Docstrings

- Inline comments and block comments are nice for someone reading the code, but they don't help you get an **overview** of the whole code from the outside
- **Docstrings** are a good way to include an overview of functionality, arguments, and return variables in a structured way
- For the **exercise**, we recommend using the **Google docstring conventions**: <u>https://</u> sphinxcontrib-napoleon.readthedocs.io/en/ <u>latest/example_google.html</u>



```
def call_skyview(field, survey, pos, fov, coord, proj='Car', pix=500):
    """Call Skyview to download data from a survey based on input parameters
    Args:
        field (str): name of the field, used in naming the output file
        survey (str): name of survey, from https://skyview.gsfc.nasa.gov/current/cgi/survey.pl
        pos (float, float): position coordinates as a tuple
        fov (float): FOV in degrees
        coord (str): coordinate system (e.g. Galactic, J2000, B1950)
       proj (str): projection of image. (e.g. Car, Sin)
        pix (int): pixel dimensions of image (e.g. 500)
    Returns:
        str: name of resulting fits file
   Examples:
             call_skvview('pks1657-298', 'dss',
        'skyview_pks1657–298_dss.fits'
        >>> call_skyview('B0329+54', 'nvss', (144.99497,-01.22029), 0.5, 'Gal')
        'skyview_B0329+54_nvss.fits'
```



Exercise: SkyviewBot Let's make some code better using #GCP





Exercise: SkyviewBot

- **Goal:** to put together a Python script (using good coding practices!) that will fetch a survey region of your choice from NASA **Skyview**, image the FITS file using **APLpy** and then send your masterpiece to Slack using Webhooks
- **Dependencies:** APLpy, PyDrive, Java for Skyview (1.8.0_151, or use the included FITS)
- There is a base repository on Github: <u>https://</u> github.com/cosmicpudding/skyviewbot
- **Tomorrow:** package the resulting scripts as a proper pip-installable (T. Dijkema)

ASTERICS-OBELICS 2019 (V. Moss, @cosmicpudding)



Have people run git pull origin master in School2019, then conda env update -f environment.yml to get the latest requirements (in particular apply and pydrive)





NASA Skyview

- **Skyview** is a VO tool that provides easy access to 100+ multi-wavelength surveys
- Easily accessible from Python using a thin wrapper around the **skyview.jar** file (provided you can get Java working!)
- The **astroquery** package also provides alternative access to Skyview (try this if you get stuck later with the .jar wrapper)
- See their documentation for more info: <u>https://</u> <u>skyview.gsfc.nasa.gov/current/docs/jar.html</u>

ASTERICS-OBELICS 2019 (V. Moss, *@cosmicpudding*)





Skyview homepage: <u>https://skyview.gsfc.nasa.gov/current/cgi/titlepage.pl</u>

Sidequest: Java

```
>> which java
/usr/bin/java
java version "1.8.0_151"
Java(TM) SE Runtime Environment (build 1.8.0_151-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)
```

• You *may* have to add to your .bash_profile if the above isn't working:

```
# Java
```

ASTERICS-OBELICS 2019 (V. Moss, @cosmicpudding)



• Downloading the skyview.jar file might just "work"... here are my settings:

>> /Library/Internet\ Plug-Ins/JavaAppletPlugin.plugin/Contents/Home/bin/java -version

export PATH="/Library/Internet Plug-Ins/JavaAppletPlugin.plugin/Contents/Home/bin:\$PATH"

Alternative method: <u>https://astroquery.readthedocs.io/en/latest/skyview/skyview.html</u>





APLpy

- **Partner package** of astropy, developed by T. Robitaille and E. Bressert (v2 = Python 3.5+)
- Allows quick **visualisation** of astronomical FITS images, with coordinates and wrappers around standard Matplotlib functions
- APLpy website: <u>https://aplpy.readthedocs.io</u> or http://aplpy.github.io
- APLpy examples: http://aplpy.sourceforge.net/ static gallery mirror (cloned? \@)
- Code for making a figure included in **exercise ASTERICS-OBELICS 2019 (V. Moss, @cosmicpudding)**



API: Google upload

- Various APIs exist to assist with automating otherwise manual actions such as uploading an image to Google Drive, sending batch emails or tweeting from Python (e.g. @zootopialicense)
- Once you've made your image using APLpy, you can use **PyDrive** to automatically upload it via the Google Drive API to make it **web-accessible**
- Google Dev: <u>https://console.cloud.google.com</u>
- PyDrive: <u>https://pypi.org/project/PyDrive</u>
- We may have fun times **authorising**... TBD! **ASTERICS-OBELICS 2019 (V. Moss, @cosmicpudding)**

🗏 Goog	gle Cloud Platform 🛛 💲 Skyvi	iewBot 👻	٩
Client	ID for Web application	DOWNLOAD JSON	C RESET S
Client ID	1072399314963-jjc9nm6oqu	ru4d534lvikl68i08gr9no.apps.goo	gleusercontent.
Client secret	uoqL3fZz9Kb1VxmS0x9jdU0)p	
Creation date	Apr 6, 2019, 9:44:16 AM		
Name 🕜			
SkyviewBotCli	ient		

Restrictions

Enter JavaScript origins, redirect URIs, or both Learn More

Origins and redirect domains must be added to the list of Authorized Domains in the OAuth consent settings.

Authorized JavaScript origins

For use with requests from a browser. This is the origin URI of the client application. It can't contain a wildcard (https://*.example.com) or a path (https://example.com/subdir). If you're using a nonstandard port, you must include it in the origin URI

https://www.example.com

Type in the domain and press Enter to add it

Authorized redirect URIs

For use with requests from a web server. This is the path in your application that users are redirected to after they have authenticated with Google. The path will be appended with the authorization code for access. Must have a protocol. Cannot contain URL fragments or relative paths. Cannot be a public IP address

http://localhost:8080/oauth2callback

http://localhost:8090/oauth2callback

http://localhost:8080/

http://localhost:8090/

https://www.example.com

Type in the domain and press Enter to add it

Cancel

<u>autoskyview@gmail.com</u> astericsannecy



API: Slack web hooks

- **Slack webhooks** allow you to easily send automated messages to a channel on Slack
- Slack webhooks are a **custom integration** that you can add to any Slack, resulting in a URL that hooks into the particular channel
- In this case, we'll send your awesome images to the **ASTERICS Slack** channel for this talk: **#gcp**
- You can also do **outgoing** Slack webhooks
- See documentation here: <u>https://api.slack.com/</u> <u>docs/message-attachments</u>

ASTERICS-OBELICS 2019 (V. Moss, @cosmicpudding)



SICICK

See: <u>https://obelics-school.slack.com/apps/A0F7XDUAZ-incoming-webhooks?next_id=0</u>

Extra: your Slack ID

- You should tag **your Slack ID** with any posts that you make to the **#gcp** channel
- Go to ASTERICS OBELICS Slack (better: app): http://obelics-school.slack.com
- Click your name under "**Direct Messages**", then click the username in the DM window
- Click "...", and then "Copy member ID" this is your unique Slack ID for the workspace, which can then be used to tag yourself in posts

ASTERICS-OBELICS 2019 (V. Moss, *@cosmicpudding*)

This is your space. Draft messages, list your to-dos, or keep links and files handy. You can also talk to yourself here, but please bear in mind you'll have to supply both sides of the conversation.

Vanessa Moss •					
Set a status					
Edit Profile	••••				
Mor	re actions				
Display name	vamoss				
Timezone	1:59 PM local time (change)				
Email	moss@astron.nl				





Let's get started!

- **Github base repo** (see **README** file): https://github.com/cosmicpudding/skyviewbot
- I've left **specific sections** for you to fill in from the skeleton code (e.g. add argparse) - read the comments to find out what to do!
- Feel free to take the code in **whichever** direction you want, and improve it in any way you see fit! e.g. more plot options, more elaborate Slack messages, error handling, etc
- You have **60 minutes** to complete this exercise, and the best image post* will win a **prize!**

ASTERICS-OBELICS 2019 (V. Moss, @cosmicpudding)



* verified based on modified skyviewbot repo, so please fork+push your changes!

