PHYS4038/MLiS and ASI/MPAGS

Scientific Programming in

mpags-python.github.io

Steven Bamford

University of Nottingham
An introduction to scientific programming with Python

Session 3: Staying organised
Session 3

In this session:

- Solutions to exercises 1 & 2
- Organising your python installation
- Version control
- GitHub tools and workflow
- How to submit coursework
- Help with conda, git, etc.
- Advice on coursework
Questions

- Talk to me:
  - During teaching sessions *(preferred)*
    - Specific questions, clarifications – just ask
    - Bigger issues – wait until end of lecture / start of examples class
    - Remote students connect via skype group:
      - [https://join.skype.com/KpW5oCLNNiJt](https://join.skype.com/KpW5oCLNNiJt)
      - text during lecture, video during examples class
  - Via email: [steven.bamford@nottingham.ac.uk](mailto:steven.bamford@nottingham.ac.uk)
  - Arrange a meeting
    - email me
    - office: CAPT A112b
    - or in skype group
Exercise solutions

https://github.com/mpags-python/exercises

https://nbviewer.jupyter.org/github/mpags-python/exercises/blob/master/Exercises2.ipynb
Managing your environment

• **Some good things about Python**
  • lots of modules from many sources
  • ongoing development of Python and modules

• **Some bad things about Python**
  • lots of modules from many sources
  • ongoing development of Python and modules

• **A solution**
  • Maintain (or have option to create) separate environments (or manifests) for different projects
Managing your environment

• **Desirable**
  • long term stability of your programs
  • help others easily install same dependencies
  • benefit from latest features and bugfixes

• **Solution**
  • maintain separate environments for different projects
    • Anaconda: conda
    • native Python: pip and virtualenv
Managing your environment

- **conda** – [http://conda.pydata.org](http://conda.pydata.org)
  - specific to the Anaconda Python distribution
  - install modules
    - automatically manage dependencies and compatibility
    - similar to 'pip', but can install binaries and not just for python
    - can use pip within a conda environment (but try conda first)
  - create and switch between environments
    - specific collections of compatible modules and executables

- Windows: use Anaconda Prompt
- Linux/Mac: use any terminal
Managing your environment

• **Some extra details for using on UoN computers**

  • Anaconda is installed on hard drive (C:) of each machine
  
  • User can write to C:\Anaconda3 – but will not be available from other machines, and probably wiped periodically
  
  • User folder is Z: – available from all machines (but 4Gb limit)

• Either recreate environment in C: each time, or create in Z:

  • specify environment by directory (-p option):
    
    $ conda create -p Z:\envs\python_course
  
  • set as default (every time):
    
    $ conda config --prepend envs_dirs Z:\envs
Managing your environment

- conda basic usage

```bash
$ conda create -n python_course # -n <name> or -p <path>
$ conda activate python_course # <name> or <path>
$ conda install scipy matplotlib
$ ipython # use the environment
$ conda deactivate
```
Managing your environment

- Saving your environment (to use on another machine or distribute)

```bash
$ conda env export -n python_course > environment.yml
$ conda create -n new_env -f environment.yml
```

- environment.yml contains all dependencies and versions
- maybe neater to manually maintain your own environment.yml

```yaml
name: myenv
dependencies:
- python
- numpy
- matplotlib
```

- to make your environment match an environment.yml file:

```bash
$ conda env update -n myenv -f myenv.yml --prune
```
Managing your environment

- **virtualenv**
  - modules are installed with pip – [https://pip.pypa.io](https://pip.pypa.io)

```
$ pip install virtualenv  # install virtualenv
$ virtualenv ENV1        # create a new environment ENV1
$ source ENV/bin/activate # set PATH to our environment
(ENV1)$ pip install emcee # install modules into ENV1
(ENV1)$ pip install numpy==1.8.2 # install specific version
(ENV1)$ python           # use our custom environment
(ENV1)$ deactivate       # return our PATH to normal
```
Managing your environment

- **virtualenv**
  - can record current state of modules to a 'requirements' file

(ENV1)$ pip freeze > requirements.txt
$ cat requirements.txt
emcee==2.1.0
numpy==1.8.2
$ deactivate
$ virtualenv ENV2
$ sourceENV2/bin/activate
(ENV2)$ pip install -r requirements.txt
Managing your environment

• **Updating packages**

$ conda update --all
$ conda update scipy emcee

OR

$ pip install --upgrade
$ pip install --upgrade scipy emcee
Jupyter kernel discovery

- Can install and run Jupyter notebook in an environment, but better to run from base environment and then select kernel within notebook
- Jupyter can autodiscover conda environments
- Just need to install nb_conda_kernels in notebook environment

$ conda install -n base nb_conda_kernels

- and ipykernel in any environments you want to use in notebook

$ conda install -n myenv ipykernel
Version control

• Keep a secure backup of your work
• Maintain a record of significant changes
• Undo mistakes
• Undo undone mistakes that turned out to not be mistakes
• Log the reasons why you made particular changes
• Separate your work on different features
• Collaborate more easily

\[\text{Git}\]

• Distributed version control
  • everyone has a full copy of history
GitHub

- Where many projects keep and share code
  - particularly open-source projects
- Unlimited private repos for education and research:
  - https://education.github.com

Similar alternative:
Getting started with version control

- Create a GitHub account
- Join assignment to create a new repository
  https://classroom.github.com/a/K4tUSkL0
- Create README in the browser
- Brief intro to Markdown
  https://guides.github.com/features/mastering-markdown/
- Installing git (with conda)

$ conda install git
Getting started with version control

- Clone your repo locally
  
  ```
  $ git clone <link_to_your_repo>
  ```

- Edit README.md locally, then check status and diff
  
  ```
  $ git status
  $ git diff # show changes
  ```

- Add files to commit, perform commit and push commit to GitHub
  
  ```
  $ git add README.md
  $ git commit -m "Edited the readme"
  $ git push
  ```

- If files changed on GitHub, fetch and merge the changes
  
  ```
  $ git pull
  ```

https://guides.github.com/introduction/git-handbook/
Good practice and GitHub extras

- Using branches and tags
- Issues
- Pull requests

For more information:
- https://guides.github.com
- https://www.atlassian.com/git/tutorials
- https://lab.github.com
Assessment

For those taking this module for MPAGS credits

• Assessed by development of a Python program relevant to your interests
  • put course material into practice
  • opportunity to become familiar with Python
  • get feedback on your coding

• Your code should...
  • be written as an executable module (.py file) or Jupyter notebook (.ipynb)
  • do something meaningful: analyse real data or perform a simulation
  • define at least two user functions (but typically more)
  • make use of appropriate specialist modules
  • produce at least one informative plot
  • comprise >~ 50 lines of actual code
    • excluding comments, imports and other ‘boilerplate’
  • contain no more than 1000 lines in total
    • if you have written more, please isolate an individual element
Code development

• Three stages (first two optional for MPAGS students)

1. hand-in by 1st November (optional for feedback)
   • README describing what you intend your code to do
   • Rough outline of the code (classes, functions, snippets, comments, pseudocode)

2. hand-in by 15th November (optional for feedback)
   • Rough version of your code, may be incomplete, have bugs, although try to make it reasonable and easy to understand!

3. hand-in by 13th December (required for MPAGS credits)
   • Complete working version of your code

Deadlines are 3pm on Fridays.
Now...

Work on your coursework README
Practise with conda and git

Any questions?

- shout and wave
- skype (spbamford)
  - https://join.skype.com/KpW5oCLNNiJt
- email steven.bamford@nottingham.ac.uk