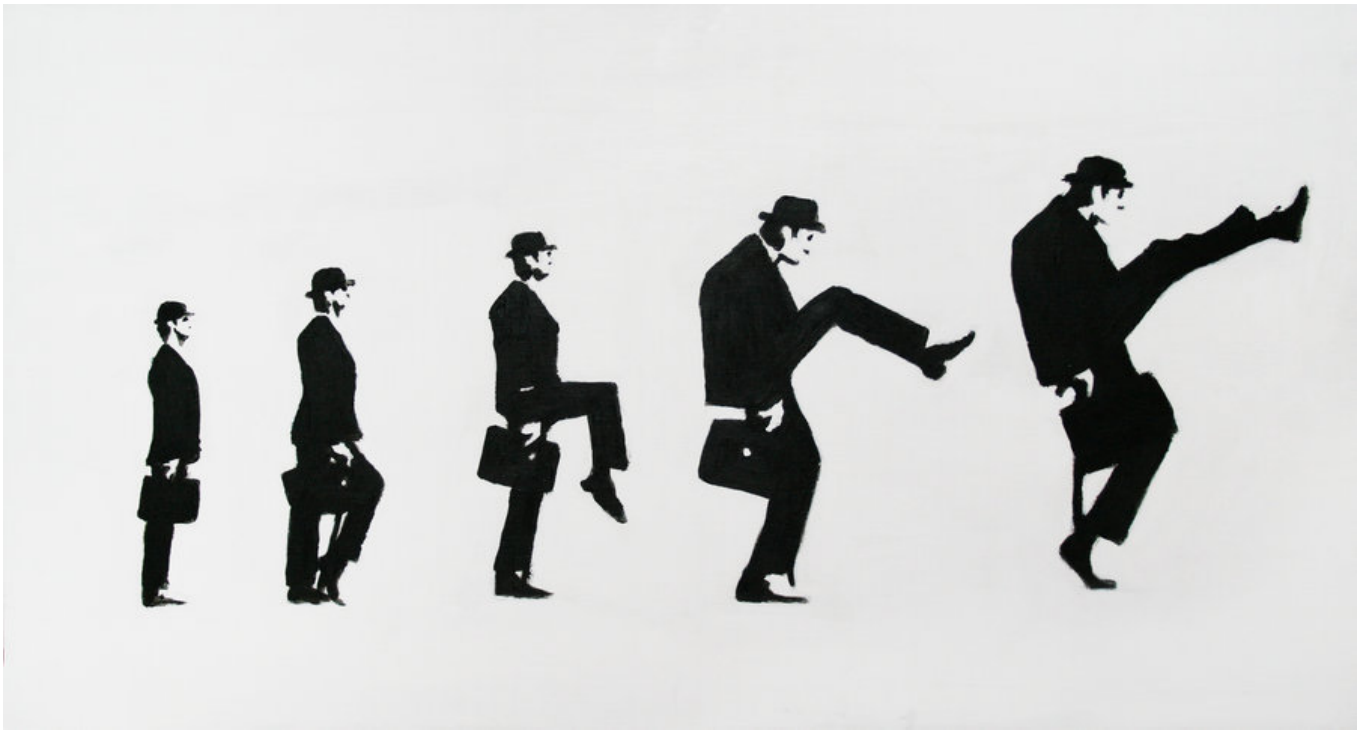
PH2150- Scientific Computing and Employabilty Skills

Introduction to Python (1 week)

Dr. Andrew Casey (a.casey@rhul.ac.uk, W054)



A walk through Python

Aims and Objectives:

Students will be introduced to the computing language *Python*.

Practice with the language will allow students to:

- Gain a basic understanding of the syntax of *Python*
- Plot and analyse experimental data
- Gain insight into some topics from 1st and 2nd year courses
- Preparation for final year project or possible summer placements

Employability skills

- Write a professional CV and use STAR analysis techniques to produce a report. Should increase chances of gaining summer internships.

Course Structure:

The course consists of:

- Post exam Python
- Two lab session per week in the Autumn term (Mon Am, Thurs PM)
- Weekly one hour employabilty skills lectures (Tues AM External speakers, CV training, Alumni talks)

The assesment is made up of:

- Eight scientific Computing Skills problem sheets (60%)
- Two employabilty assesment exercises (10%)
- Final three week extending programming exercise (30%)

Course Structure:

The first 4 weeks covering the basics of Python,

- Variables (types assignment)
- Operators
- Inputs/Outputs
- Control Structures (Loops, if, while, for)
- Functions (User Defined, Modules)
- Plotting
- Fitting Data
- Errors

Expanding the basic Python for scientific computing with the libraries:

- NumPy (Python's n-dimensional array variable type, and functions operating on arrays)
- SciPy (Fundamental library for scientific computing)
- Matplotlib (Comprehensive 2D Plotting)

Course Structure:

After the basic syntax exercises will be related to your other Physics courses:

These problems come from:

- Quantum Mechanics
- Optics
- Electromagnetism
- Mathematical Methods

Finally you will bring your new computing / communications skills together in producing a GUI to demonstrate a chosen aspect of undergraduate physics.

In the weeks in which the employabilty exercises are due there will not be a computing problem sheet due in.

Course Structure:

Assesment of PH2150 problem sheets

- Each problem sheet will be submitted to Turnitin via the PH2150 moodle page
- Post exam problem sheets (PS1, PS2, PS3 will not be due in until the Autumn term)
- PS1 will be due at the end of the day after the first session, PS2 after the 2nd and PS3 before the third.
- From week two you will be introduced to a new problem sheet each week, which will be due in at the end of that week, except for the two weeks in which the employability exercise is due.
- We will mark the problem sheets on an individual basis within the lab sessions, this gives instant feedback and tips on how to improve, and hopefully leaves you with a working programme for later reference.
- The final report and code will be submitted to the office and marked anonymously.

They should start gently and build up to more complex problems as you become more proficient in *Python*.

The Jupyter Notebook.

Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, machine learning and much more.

The lecture notes will be stored as Jupyter notebooks, this is a mix of formatted text, python code and code output. It requires the Jupyter notebook server to run it though, and therefore isn't a stand-alone but there is no difference between the Python code that goes into a programme file or a Jupyter notebook.

Why Python?

More detail about this question in the moodle notes, but the essence is that:

- Rapid development time
- Open source (free to have at home, on your own machines)
- Cross-platform (Windows, Macs, Linux, Android, Raspberry Pi) software that you write on one system will work in any other.
- Its easy to do easy things, and possible to do really complicated things.
- Growing user base in academia and industry.
- Check out the number of SEPnet summer placement students who were using Python (also recent Alumni talks where they had an edge over others because of their python skills)

Who's using python?

"Python has been an important part of Google since the beginning, and remains so as the system grows and evolves. Today dozens of Google engineers use Python, and we're looking for more people with skills in this language." said Peter Norvig, director of search quality at Google, Inc.

Who's using python?

"Python is fast enough for our site and allows us to produce maintainable features in record times, with a minimum of developers," said Cuong Do, Software Architect, YouTube.com.

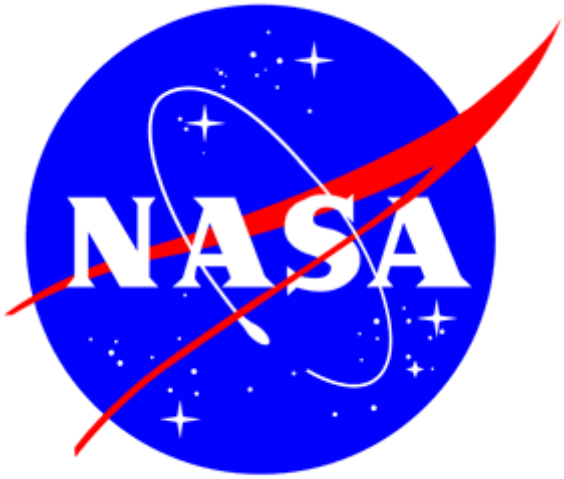
Who's using python?

"Python plays a key role in our production pipeline. Without it a project the size of Star Wars: Episode II would have been very difficult to pull off. From crowd rendering to batch processing to compositing, Python binds all things together," said Tommy Burnette, Senior Technical Director, Industrial Light & Magic.

"Python is everywhere at ILM. It's used to extend the capabilities of our applications, as well as providing the glue between them. Every CG image we create has involved Python somewhere in the process," said Philip Peterson, Principal Engineer, Research & Development, Industrial Light & Magic.

Who's using python?

"NASA is using Python to implement a CAD/CAE/PDM repository and model management, integration, and transformation system which will be the core infrastructure for its next generation collaborative engineering environment. We chose Python because it provides maximum productivity, code that's clear and easy to maintain, strong and extensive (and growing!) libraries, and excellent capabilities for integration with other applications on any platform. All of these characteristics are essential for building efficient, flexible, scalable, and well-integrated systems, which is exactly what we need. Python has met or exceeded every requirement we've had," said Steve Waterbury, Software Group Leader, NASA STEP Testbed.



Which version of Python?

- There are two variants 2.x and since 2008 3.x (not backwards compatible)
- They are around 90% the same and if you learn one you should have few problems with the other.
- Up until last year we used a 2.x distribution of python, Specifically Python 2.7.3 contained within Enthoughts Canopy distribution https://www.enthought.com/products/canopy/ (https://www.enthought.com/products/canopy/), this will still be available on the teaching lab machines. ### However:
- According to the creators of python "Python 2.x is legacy, Python 3.x is the present and future of the language"
- The remaining libraries that we use in PH2150 have been "ported" into 3.x
- During this summer the College has migrated to Windows 10 and have installed 3.x on all of the teaching lab machines, . ### The Python Environment that has been installed is called Ananconda:
- Specifically, https://www.anaconda.com/download/ (https://www.anaconda.com/download/) 64-bit windows, python 3.6, Anaconda 5.1
- May be upgraded over the summer to Anaconda 2019, Python 3.7

The Python Environment

We are now going to take a look at Anaconda, but which ever environment you choose to run python in it requires two elements:

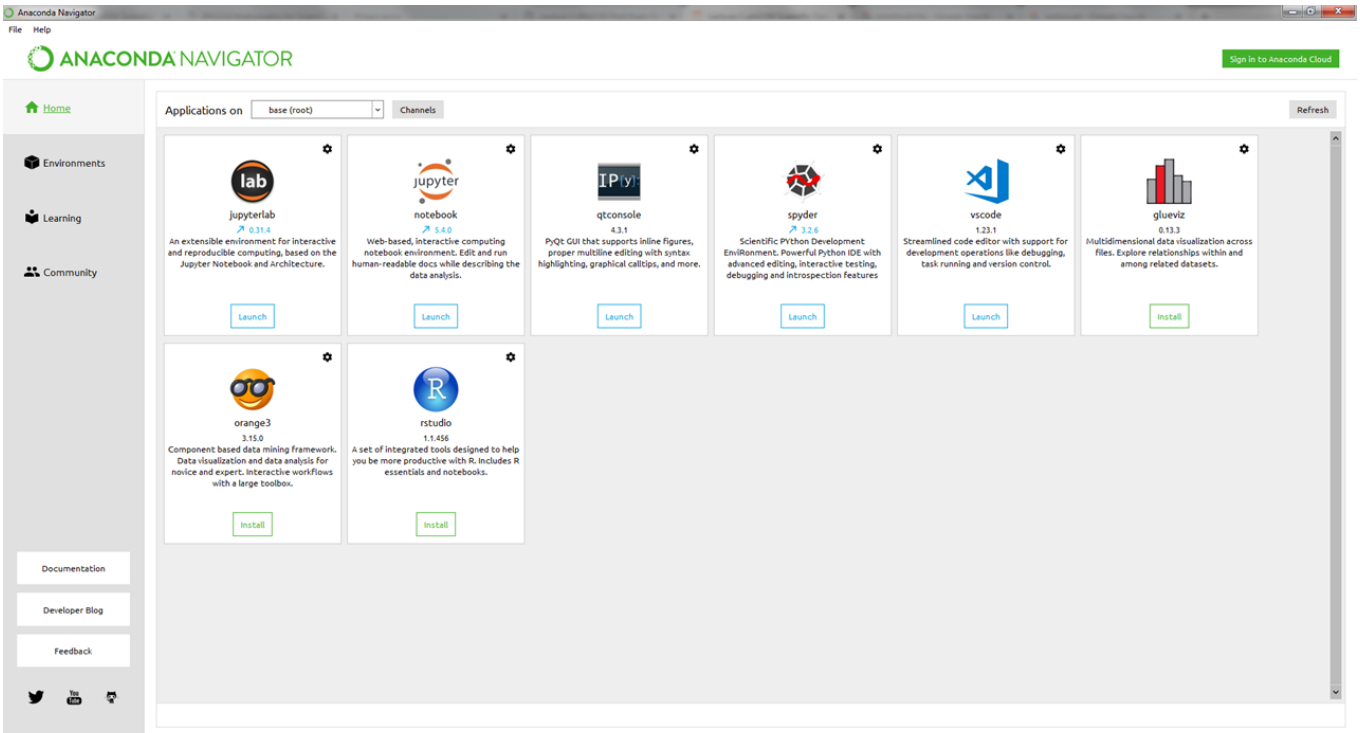
- A development Environment, where you edit code. This could be a basic text editor (not word processor), but it usual to use a text editor optimised for coding (this then will have useful features like recognising elements of the code and distinguishing them by colour, auto-completing commands, error checking, trace-back etc)
- The Interpreter, the Python shell, where the computer runs the code.
- These can be combined to form an integrated development environment (IDE) such as in Canopy, or Spyder (Scientific PYthon Development EnviRonment) within the Anaconda distribution.

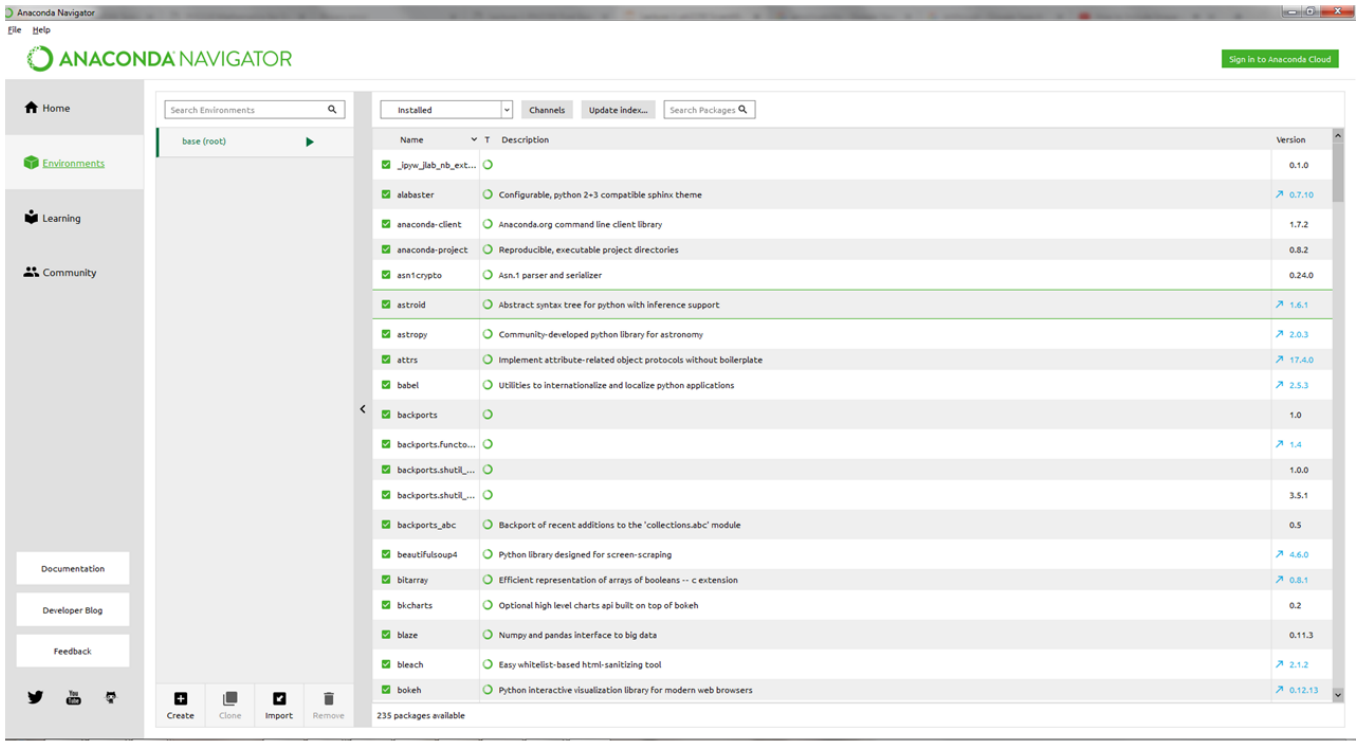
To open the environment lauch Anaconda Navigator from the start menu:

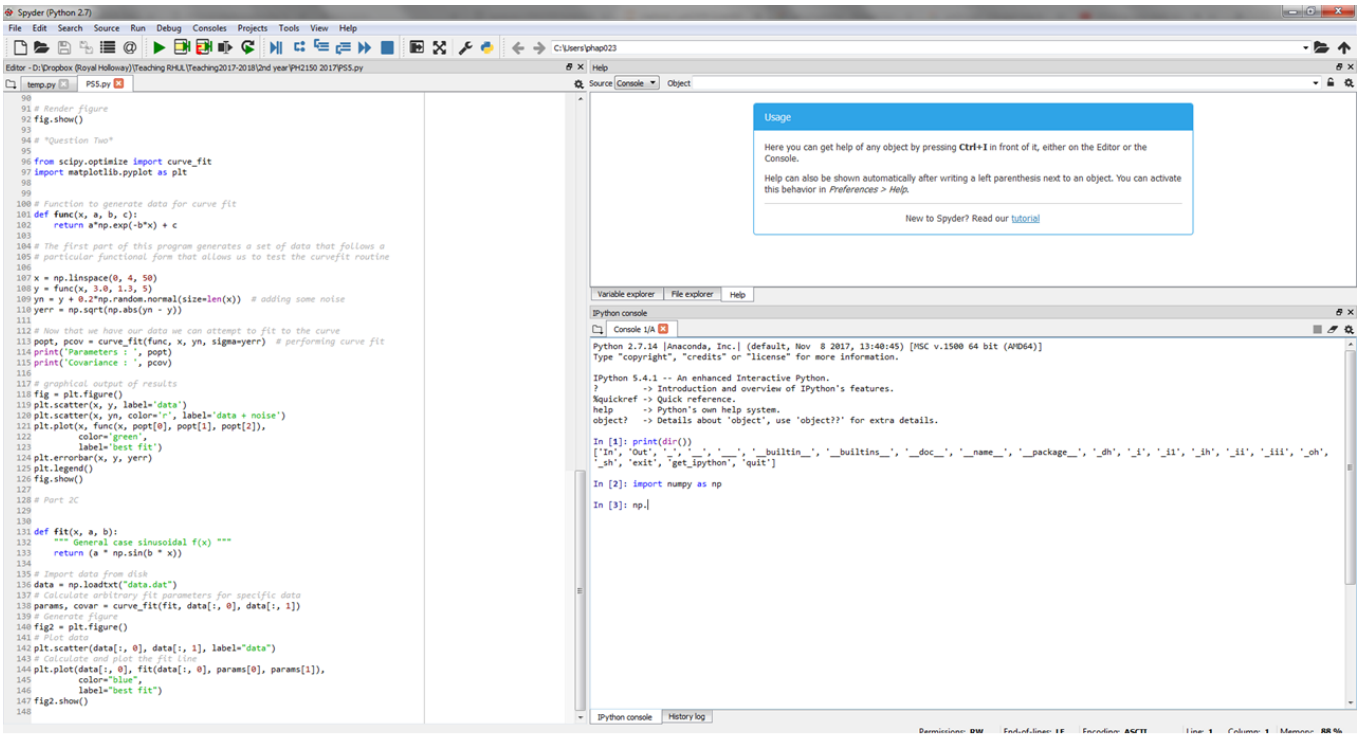
In [1]:

```
# An example of an interpreter
#Here the hash symbol signifies a comment and the interpreter skips the line after the
 #
from math import sqrt
x=7.2 # Here I am assigning the value 7.2 to the variable x
y=sqrt(x)
print('The square root of 7.2=',y)
```

```
The square root of 7.2= 2.6832815729997477
```


Anaconda Navigator
File Help
ANACONDA NAVIGATOR
Sign in to Anaconda Cloud
Home
Environments
Learning
Community
Documentation
Developer Blog
Feedback
Applications on
base (root)
Channels
Refresh
jupyterlab
0.31.4
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch
notebook
5.4.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch
qtconsole
4.3.1
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch
spyder
3.2.6
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features
Launch
vscode
1.23.1
Streamlined code editor with support for development operations like debugging, task running and version control.
Launch
glueviz
0.13.3
Multidimensional data visualization across files. Explore relationships within and among related datasets.
Install
orange3
3.15.0
Component based data mining framework. Data visualization and data analysis for novice and expert. Interactive workflows with a large toolbox.
Install
rstudio
1.1.456
A set of integrated tools designed to help you be more productive with R. Includes R essentials and notebooks.
Install


Anaconda Navigator
File Help
ANACONDA NAVIGATOR
Sign in to Anaconda Cloud
Home
Environments
Learning
Community
Documentation
Developer Blog
Feedback
Search Environments
base (root)
Installed
Channels
Update index...
Search Packages
Name
T
Description
Version
_ipyw_jlab_nb_ext...
0.1.0
alabaster
Configurable, python 2+3 compatible sphinx theme
0.7.10
anaconda-client
Anaconda.org command line client library
1.7.2
anaconda-project
Reproducible, executable project directories
0.8.2
asn1crypto
Asn.1 parser and serializer
0.24.0
astroid
Abstract syntax tree for python with inference support
1.6.1
astropy
Community-developed python library for astronomy
2.0.3
attrs
Implement attribute-related object protocols without boilerplate
17.4.0
babel
Utilities to internationalize and localize python applications
2.5.3
backports
1.0
backports.functo...
1.4
backports.shutil_...
1.0.0
backports.shutil_...
3.5.1
backports_abc
Backport of recent additions to the 'collections.abc' module
0.5
beautifulsoup4
Python library designed for screen-scraping
4.6.0
bitarray
Efficient representation of arrays of booleans -- c extension
0.8.1
bkcharts
Optional high level charts api built on top of bokeh
0.2
blaze
Numpy and pandas interface to big data
0.11.3
bleach
Easy whitelist-based html-sanitizing tool
2.1.2
bokeh
Python interactive visualization library for modern web browsers
0.12.13
Create
Clone
Import
Remove
235 packages available



Post Exam Notes and Problem Sheets

Until the moodle rollover will be stored on the departmental twiki:

https://twiki.ph.rhul.ac.uk/twiki/bin/view/Students/UnderGraduates/PostExamPython (https://twiki.ph.rhul.ac.uk/twiki/bin/view/Students/UnderGraduates/PostExamPython)

- Introduction Course Notes
- Notes on packages NumPy and SciPy
- Problem sheets 1-3 with accompanying material
- Uploaded Jupyter Notebooks contained code examples

During the course this material will be moved to moodle

Further reading

- The Python Software foundation: https://www.python.org/ (https://www.python.org/), The official web page of the Python programming language, contains tutorials, documents, reference library.
  - http://www.numpy.org/ (http://www.numpy.org/), the array
  - https://scipy.org/ (https://scipy.org/), scientific python library, documentation and tutorials
  - https://matplotlib.org/ (https://matplotlib.org/), Matplotlib is a Python 2D plotting library which produces publication quality figures (check-out the gallery)
  - http://www.python.org/dev/peps/pep-0008 (http://www.python.org/dev/peps/pep-0008) - Style guide for Python programming. Highly recommended.
- https://www.codecademy.com/ (https://www.codecademy.com/) self learning website
- https://software-carpentry.org/ (https://software-carpentry.org/) online tutorials
- http://https://stackoverflow.com/ (http://https://stackoverflow.com/), an online developer community, question and answer site
-

Books

This course is for solving physics problems, we get onto Numpy, SCipy very quickly, in some general programming books these are advanced topics (chapters 90+) or not covered at all. There are many good python books for the basic syntax, for the more physics relate problems I recommend:

- NEW.. Introduction ot Python for Science and Engineering, David J. Pine (Very close match to most topics)
- A Primer on Scientific Programming with Python, Hans Petter Langtangen
- Introduction to numerical programming, Titus Beu
- Numerical Methods in Engineering with Python, Jaan Kiusalaas and more general
- Beginning Python, from novice to professional, Magnus Hetland.
- Learn Python the Hard Way, Zed Shaw.
- Learn Python in one day and Learn it well, Jamie Chan

In [ ]: