Computational Astrophysics

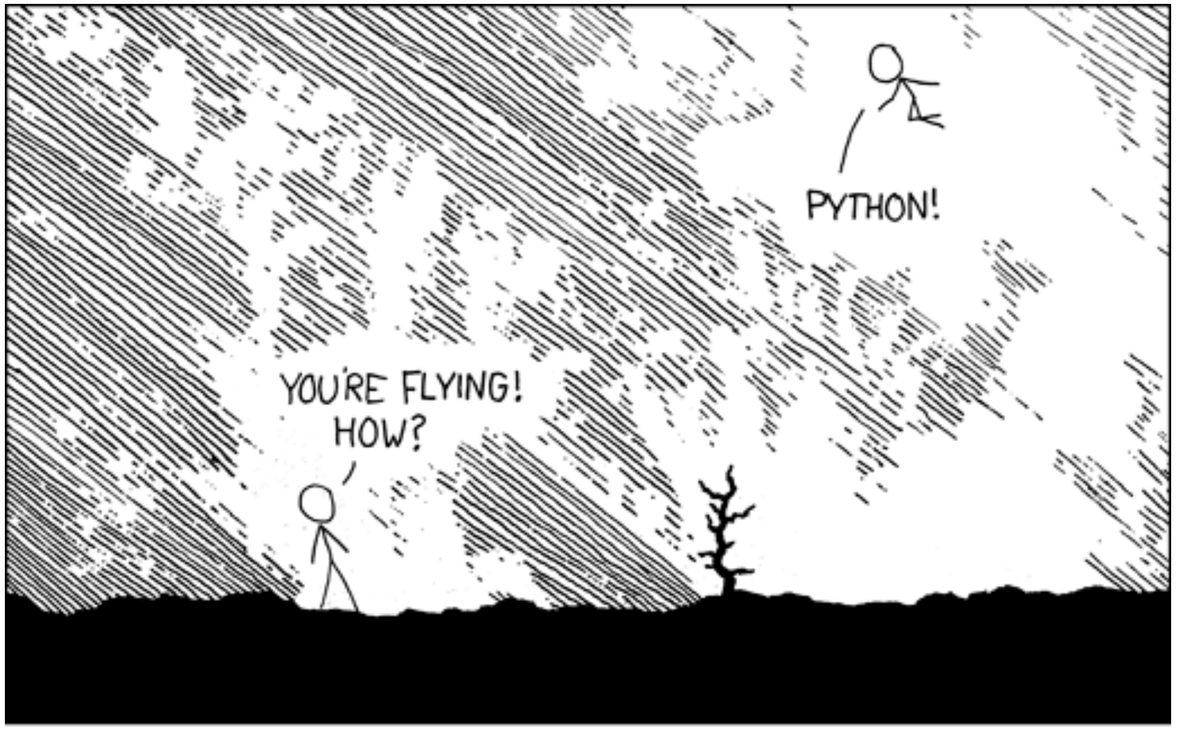
ASTR 660, Spring 2021
計算天文物理

Lecture 4

Instructor: Prof. Kuo-Chuan Pan
kuochuan.pan@gapp.nthu.edu.tw

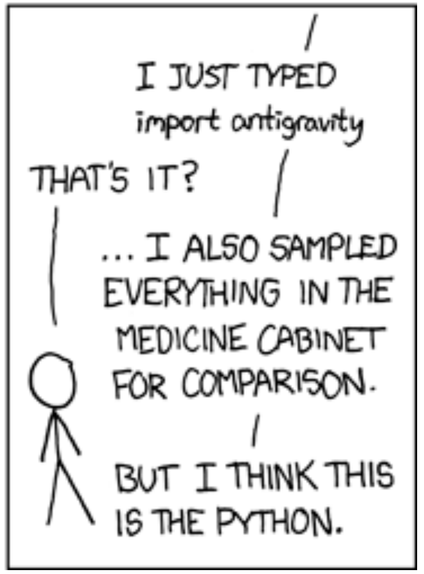
Class website

https://kuochuanpan.github.io/courses/109ASTR660_CA/


PYTHON!
YOU'RE FLYING! HOW?


I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE!
HELLO WORLD IS JUST print "Hello, world!"


I DUNNO... DYNAMIC TYPING? WHITESPACE?
COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE!
BUT HOW ARE YOU FLYING?


I JUST TYPED import antigravity
THAT'S IT?
... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.
BUT I THINK THIS IS THE PYTHON.

ref: xkcd.com

- A modern, interpreted, high-level, general purpose programming language.
- Expressive language: fewer codes
- Dynamically typed: No need to define the type of variables (disadvantage: slow)
- Interpreted: No need to compile (disadvantage: slow)
- Automatic memory management (disadvantage: memory leak)

- First released in 1991
- Purpose: improve the code readability
- Python 2.0 was released in 2000
- Python 3.0 was released in 2008 (not completely backward-compatible)

What makes python good for scientific computing?

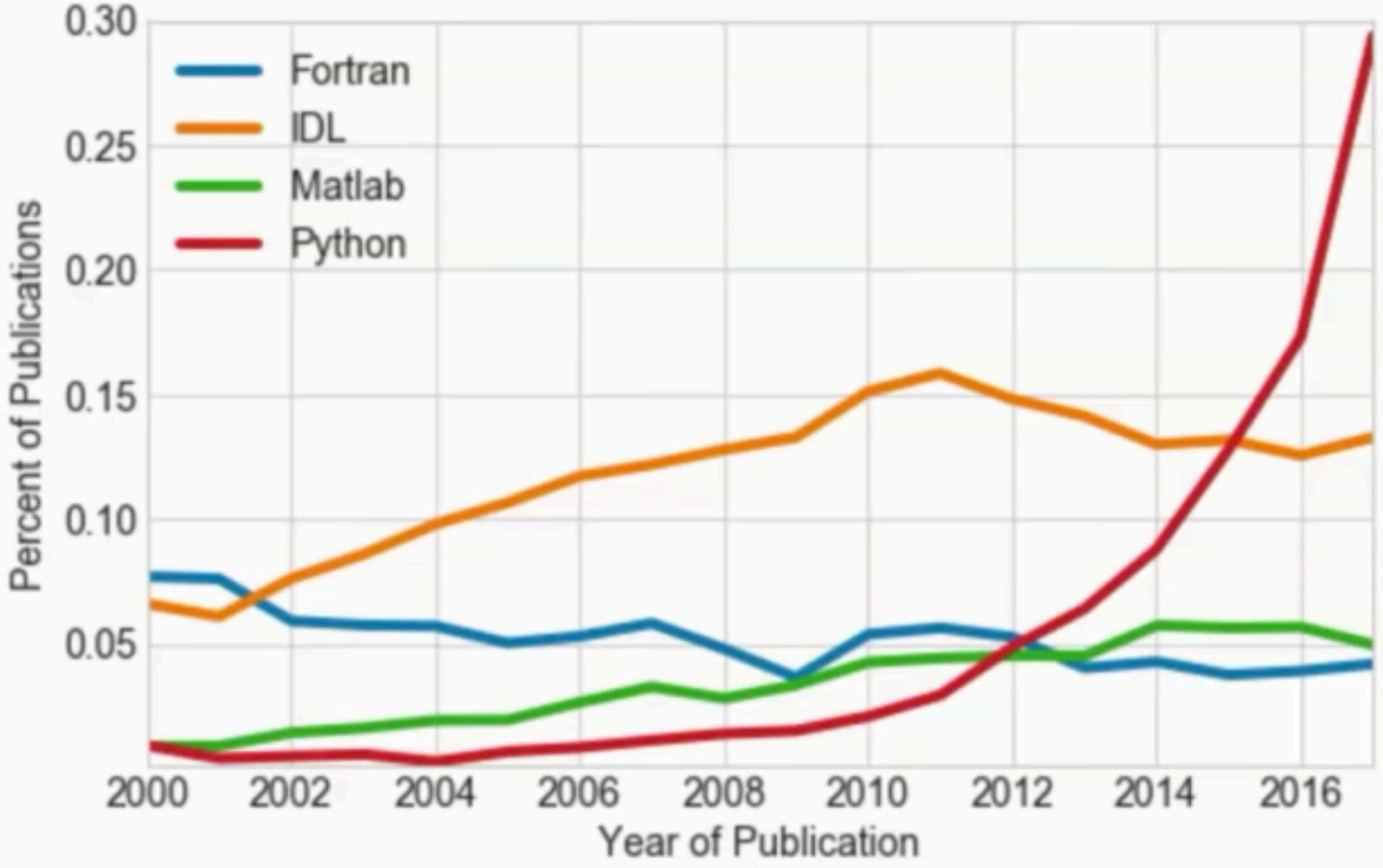
- Large community of users
- Plenty of scientific libraries and environments (ex. numpy, scipy, matplotlib, scikit-learn, astropy, …etc.)
- Good integration with highly optimized codes written in C and Fortran
- Good support for parallel programming (MPI) and GPU computing
- Open sourced

In this lecture, I assume

- You have attended the previous lectures.
- You have done the Fortran exercise
- You know basic Python programming (maybe, from other courses)

Python for Astronomers

Mentions of Software in Astronomy Publications:

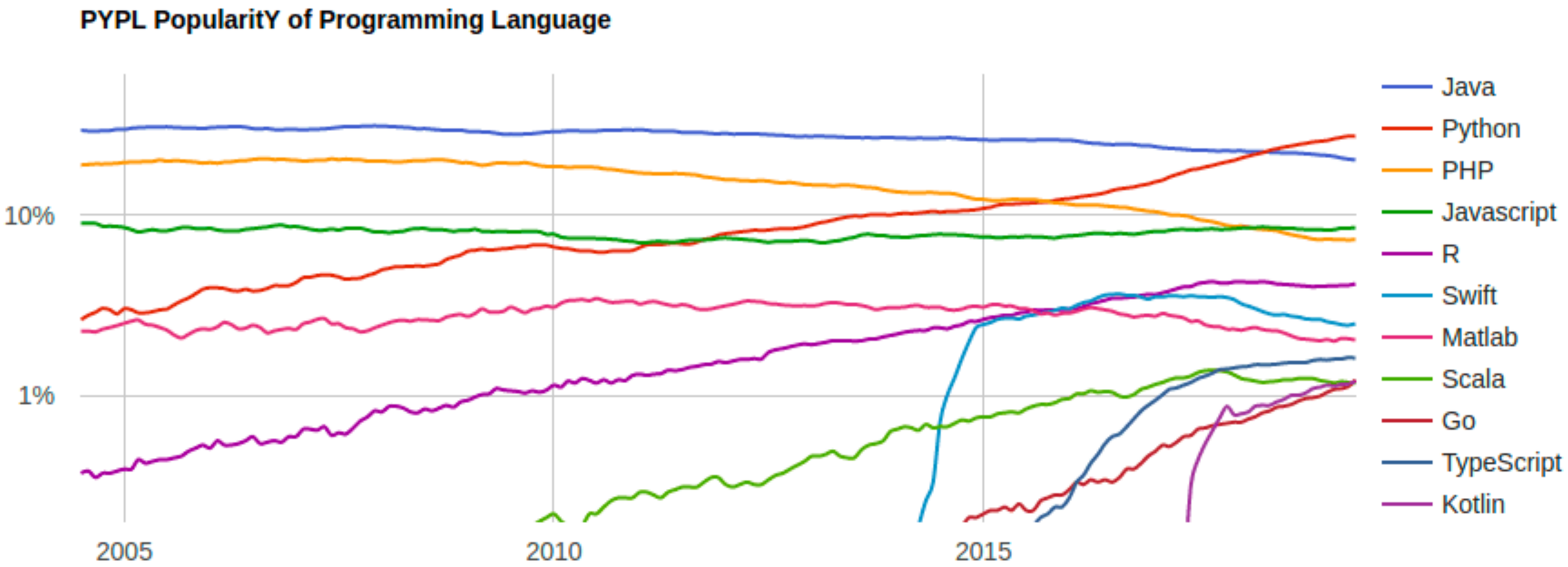


Compiled from NASA ADS (code).

Thanks to Juan Nunez-Iglesias, Thomas P. Robitaille, and Chris Beaumont.

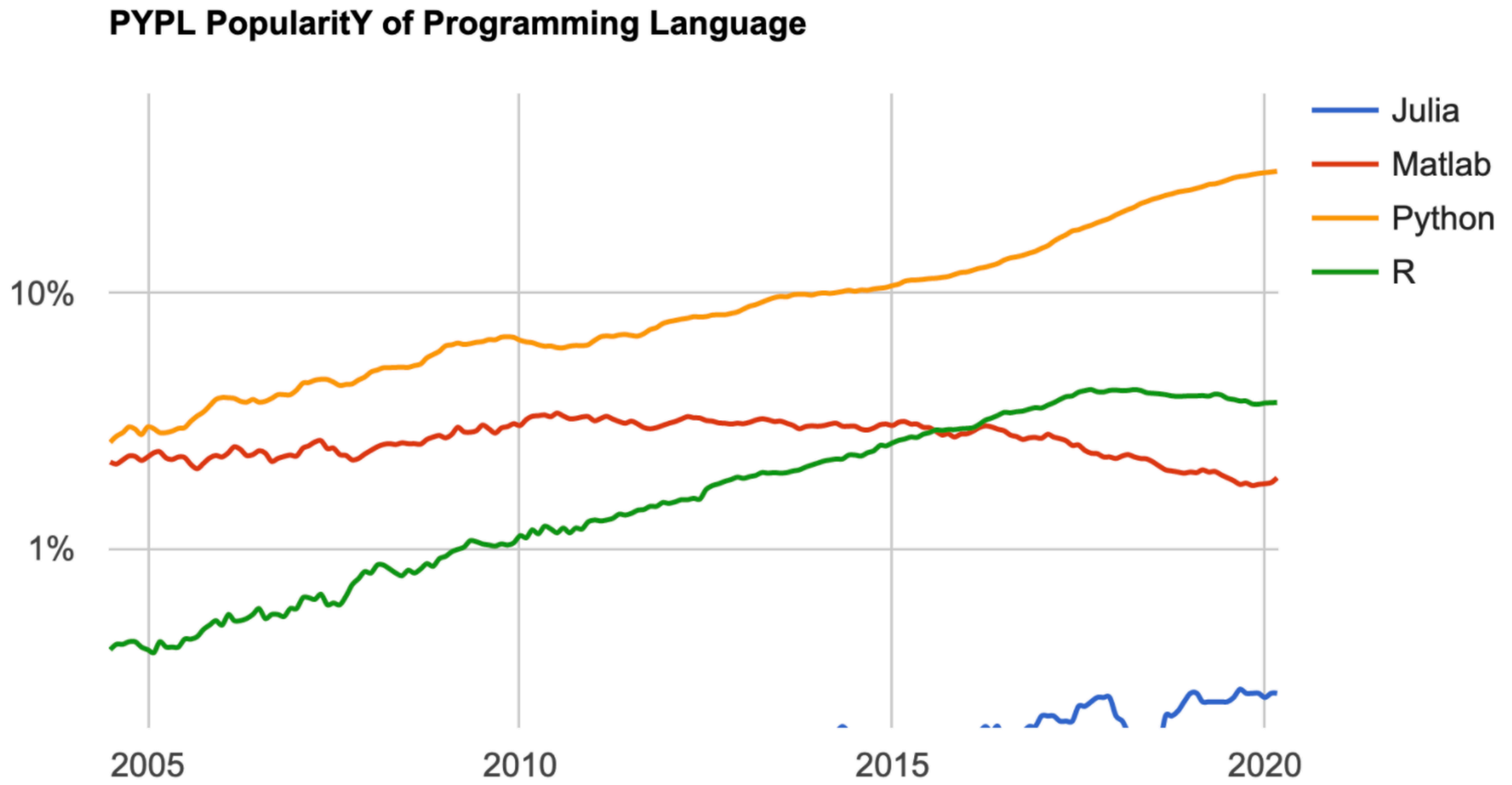
https://lasseschultebraucks.com/2017/08/14/why-python-is-probably-the-programming-language-for-dealing-with-big-data.html

Top Programming Language Trends in 2019



https://www.valuecoders.com/blog/technology-and-apps/programming-trends-2017-it-outsourcing-companies/

Scientific computing languages



Matlab starts to die, R gets more popular but seems to plateau.

Python Environment

- conda create -n compAstro python=3

Packages

- conda install -c conda-forge jupyterlab
- conda install numpy
- conda install scipy
- conda install matplotlib

PEP 8 -- Style Guide for Python Code

PEP:	8
Title:	Style Guide for Python Code
Author:	Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com>
Status:	Active
Type:	Process
Created:	05-Jul-2001
Post-History:	05-Jul-2001, 01-Aug-2013

REF: https://www.python.org/dev/peps/pep-0008/

REF: https://www.python.org/dev/peps/pep-0008/

- Indentation: Use 4 spaces per level (not a “tab”)
- Continue lines should align wrapped elements vertically

REF: https://www.python.org/dev/peps/pep-0008/

```
# Aligned with opening delimiter.
foo = long_function_name(var_one, var_two,
                         var_three, var_four)

# Add 4 spaces (an extra level of indentation) to distinguish arguments from the
rest.
def long_function_name(
        var_one, var_two, var_three,
        var_four):
    print(var_one)

# Hanging indents should add a level.
foo = long_function_name(
    var_one, var_two,
    var_three, var_four)
```

REF: https://www.python.org/dev/peps/pep-0008/

```
# Arguments on first line forbidden when not using vertical alignment.
foo = long_function_name(var_one, var_two,
    var_three, var_four)

# Further indentation required as indentation is not distinguishable.
def long_function_name(
    var_one, var_two, var_three,
    var_four):
    print(var_one)
```

REF: https://www.python.org/dev/peps/pep-0008/

- Indentation: Use 4 spaces per level (not a “tab”)
- Continue lines should align wrapped elements vertically
- Maximum line length < 79 characters
- Line break before a binary operator

REF: https://www.python.org/dev/peps/pep-0008/

```
# Yes: easy to match operators with operands
income = (gross_wages
          + taxable_interest
          + (dividends - qualified_dividends)
          - ira_deduction
          - student_loan_interest)
```

REF: https://www.python.org/dev/peps/pep-0008/

- Indentation: Use 4 spaces per level
- Continue lines should align wrapped elements vertically
- Maximum line length < 79 characters
- Line break before a binary operator
- Always use UTF-8 encoding
- Import on the top of the files (line by line)

REF: https://www.python.org/dev/peps/pep-0008/

```
Yes:  import os
      import sys

No:   import sys, os
```

```
from subprocess import Popen, PIPE
```

REF: https://www.python.org/dev/peps/pep-0008/

- Indentation: Use 4 spaces per level
- Continue lines should align wrapped elements vertically
- Maximum line length < 79 characters
- Line break before a binary operator
- Always use UTF-8 encoding
- Import on the top of the files (line by line)
- Avoid trailing white space anywhere

PEP 8 -- Style Guide for Python Code

PEP:	8
Title:	Style Guide for Python Code
Author:	Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com>
Status:	Active
Type:	Process
Created:	05-Jul-2001
Post-History:	05-Jul-2001, 01-Aug-2013

REF: https://www.python.org/dev/peps/pep-0008/

Problem Set 3

https://kuochuanpan.github.io/courses/109ASTR660_CA/

Next lecture

- Linear and Non-linear equations