Python for scientific research

Introduction

John Joseph Valletta

University of Exeter, Penryn Campus, UK

June 2017

Acknowledgements

- The workshop is funded by Exeter's researcher-led initiative award
- Thanks to Jeremy Metz for sharing his notes used in the Biomedical Informatics Hub, from which I borrowed some examples
- Last but not least, big thanks to Mario Recker, Thomas Holding, Warren Tennant and James Clewett for helping out putting this workshop together

Housekeeping

Day 1

08:45 - 10:30	Introduction to Python
10:30 - 11:00	Coffee/Tea break
11:00 - 12:30	Flow control
12:30 - 13:30	Lunch
13:30 - 15:00	Functions, modules and packages
15:00 - 15:30	Coffee/Tea break
15:30 - 17:00	Number crunching using Numpy/Scipy

Housekeeping

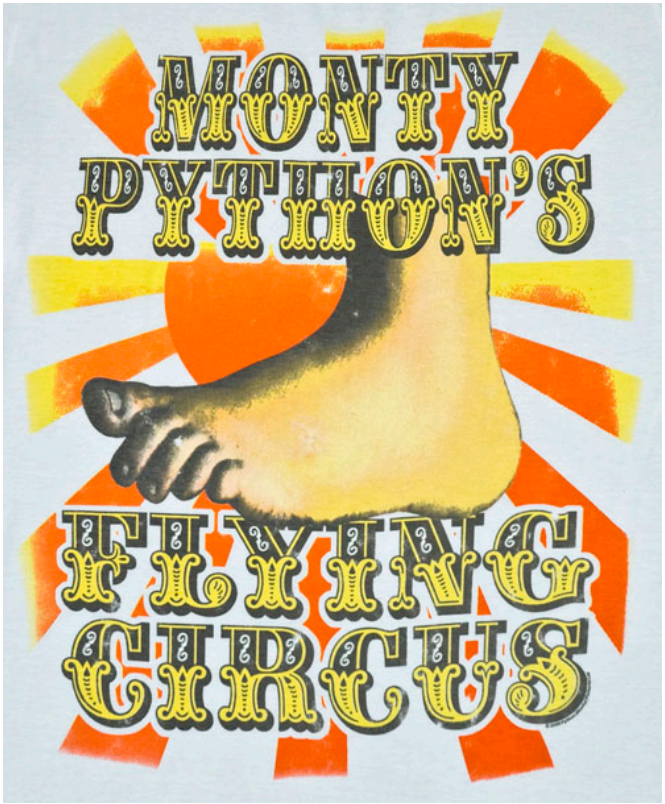
Day 2

09:00 - 10:30	Plotting with Matplotlib
10:30 - 11:00	Coffee/Tea break
11:00 - 12:30	Data analysis with Pandas
12:30 - 13:30	Lunch
13:30 - 15:00	Data visualisation with Seaborn
15:00 - 15:30	Coffee/Tea break
15:30 - 17:00	Advanced topics

References

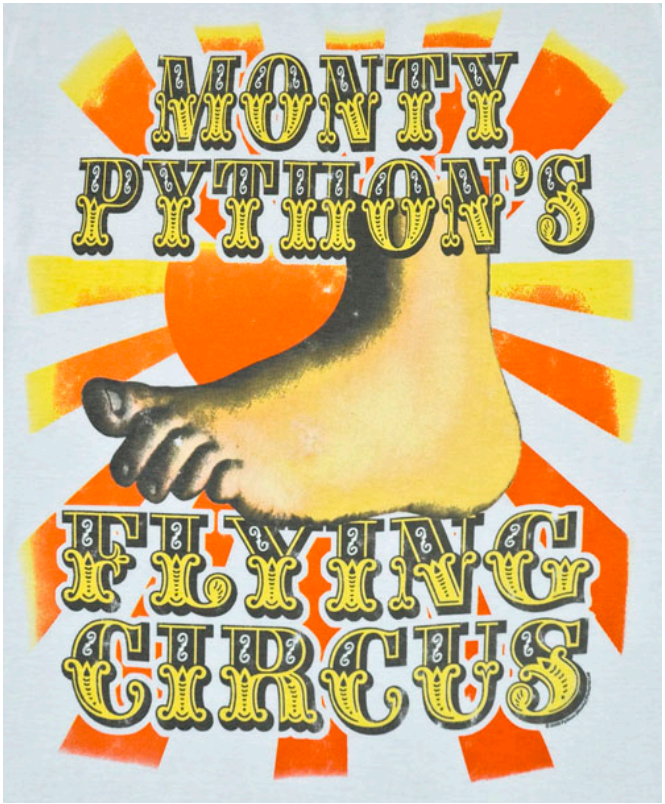
- A Byte of Python
- Think Python
- Python for Computational Science and Engineering
- A Primer on Scientific Programming with Python
- Introduction to Python for Econometrics, Statistics and Numerical Analysis

What is Python?



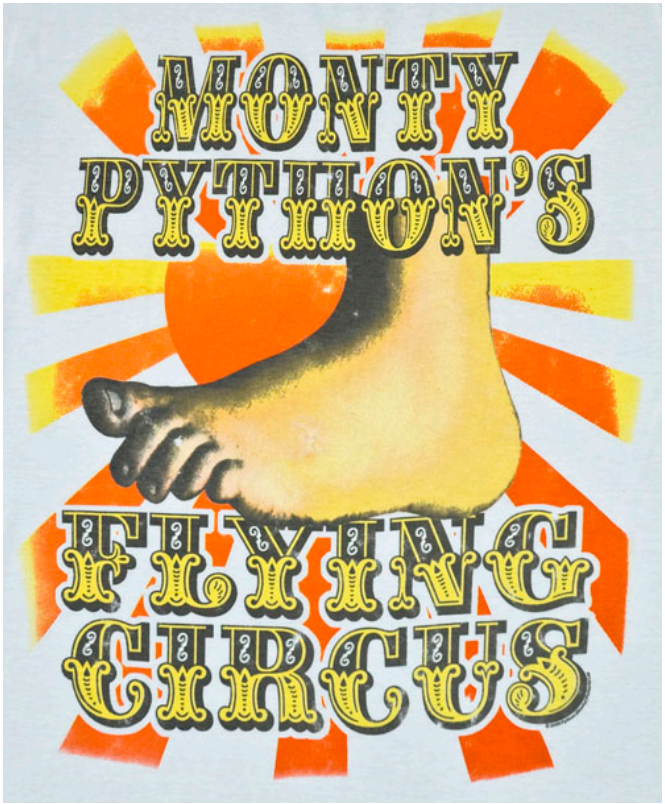
- A scripted high-level programming language created by Guido Van Rossum and named after Monty Python's Flying Circus
- Easy-to-use, versatile and with an emphasise on readability
- It has a minimalistic English-like syntax, relying on indentation instead of curly brackets, semicolons etc.

What is Python?



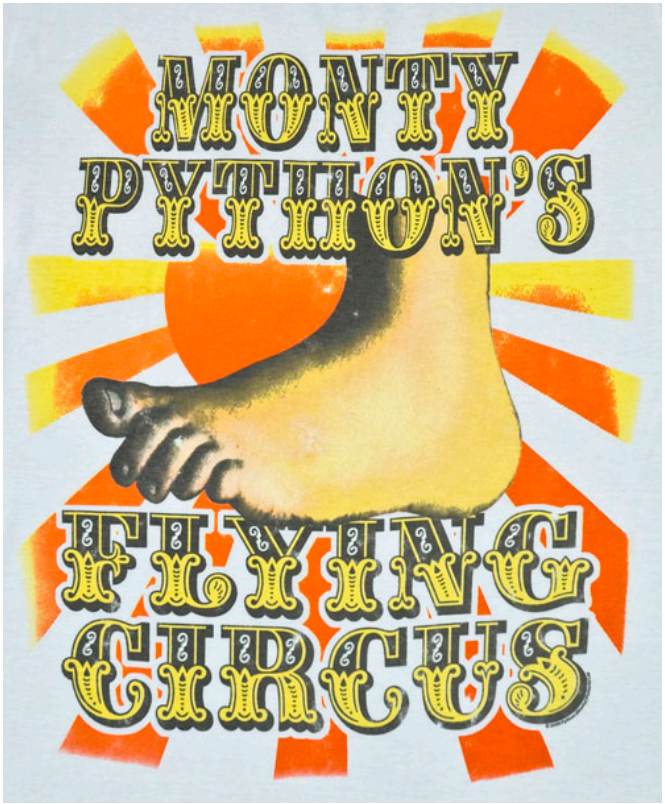
- A scripted high-level programming language created by Guido Van Rossum and named after Monty Python's Flying Circus
- Easy-to-use, versatile and with an emphasise on readability
- It has a minimalistic English-like syntax, relying on indentation instead of curly brackets, semicolons etc.

What is Python?



- A scripted high-level programming language created by Guido Van Rossum and named after Monty Python's Flying Circus
- Easy-to-use, versatile and with an emphasise on readability
- It has a minimalistic English-like syntax, relying on indentation instead of curly brackets, semicolons etc.

What is Python?



- A scripted high-level programming language created by Guido Van Rossum and named after Monty Python's Flying Circus
- Easy-to-use, versatile and with an emphasise on readability
- It has a minimalistic English-like syntax, relying on indentation instead of curly brackets, semicolons etc.

Why Python?

The TIOBE index is a measure of the popularity of programming languages

May 2017	May 2016	Change	Programming Language	Ratings	Change
1	1		Java	14.639%	-6.32%
2	2		C	7.002%	-6.22%
3	3		C++	4.751%	-1.95%
4	5	↑	Python	3.548%	-0.24%
5	4	↓	C#	3.457%	-1.02%
6	10	⇈	Visual Basic .NET	3.391%	+1.07%
7	7		JavaScript	3.071%	+0.73%
8	12	⇈	Assembly language	2.859%	+0.98%
9	6	↓	PHP	2.693%	-0.30%
10	9	↓	Perl	2.602%	+0.28%
11	8	↓	Ruby	2.429%	+0.09%
12	13	↑	Visual Basic	2.347%	+0.52%
13	15	↑	Swift	2.274%	+0.68%
14	16	↑	R	2.192%	+0.86%
15	14	↓	Objective-C	2.101%	+0.50%
16	42	⇈	Go	2.080%	+1.83%
17	18	↑	MATLAB	2.063%	+0.78%

Why Python?

- It is free! No licence costs
- Runs on all platforms (Mac, Windows, Linux)
- Because of it's ease of programming (e.g no neeed to worry about memory allocation), Python minimises development effort
- A huge number of libraries, written by an active community
- Python can "glue" together functions written in C/C++ and Fortran to speed things up (we can also call R and MATLAB functions)
- Compared to other high-level scientific languages such as MATLAB and R, Python offers a much wider range of additional functionality (e.g web and GUI development)

Why Python?

- It is free! No licence costs
- Runs on all platforms (Mac, Windows, Linux)
- Because of it's ease of programming (e.g no neeed to worry about memory allocation), Python minimises development effort
- A huge number of libraries, written by an active community
- Python can "glue" together functions written in C/C++ and Fortran to speed things up (we can also call R and MATLAB functions)
- Compared to other high-level scientific languages such as MATLAB and R, Python offers a much wider range of additional functionality (e.g web and GUI development)

Why Python?

- It is free! No licence costs
- Runs on all platforms (Mac, Windows, Linux)
- Because of it's ease of programming (e.g no neeed to worry about memory allocation), Python minimises development effort
- A huge number of libraries, written by an active community
- Python can "glue" together functions written in C/C++ and Fortran to speed things up (we can also call R and MATLAB functions)
- Compared to other high-level scientific languages such as MATLAB and R, Python offers a much wider range of additional functionality (e.g web and GUI development)

Why Python?

- It is free! No licence costs
- Runs on all platforms (Mac, Windows, Linux)
- Because of it's ease of programming (e.g no neeed to worry about memory allocation), Python minimises development effort
- A huge number of libraries, written by an active community
- Python can "glue" together functions written in C/C++ and Fortran to speed things up (we can also call R and MATLAB functions)
- Compared to other high-level scientific languages such as MATLAB and R, Python offers a much wider range of additional functionality (e.g web and GUI development)

Why Python?

- It is free! No licence costs
- Runs on all platforms (Mac, Windows, Linux)
- Because of it's ease of programming (e.g no neeed to worry about memory allocation), Python minimises development effort
- A huge number of libraries, written by an active community
- Python can "glue" together functions written in C/C++ and Fortran to speed things up (we can also call R and MATLAB functions)
- Compared to other high-level scientific languages such as MATLAB and R, Python offers a much wider range of additional functionality (e.g web and GUI development)

Why Python?

- It is free! No licence costs
- Runs on all platforms (Mac, Windows, Linux)
- Because of it's ease of programming (e.g no neeed to worry about memory allocation), Python minimises development effort
- A huge number of libraries, written by an active community
- Python can "glue" together functions written in C/C++ and Fortran to speed things up (we can also call R and MATLAB functions)
- Compared to other high-level scientific languages such as MATLAB and R, Python offers a much wider range of additional functionality (e.g web and GUI development)

Horses for courses

- Python is becoming the de facto standard for exploratory and interactive scientific research

BUT

- Python is no programming silver bullet
- Your application will ultimately dictate the tool (and a mixture of more than one language **is** ok). For example:
  - MATLAB excels at interfacing with hardware, e.g. connecting to configure an integrated circuit board or connecting to a
  - R is great for data wrangling and visualization and statistical modelling
  - (a probabilistic programming language) is an excellent choice for performing full Bayesian statistical inference

Horses for courses

- Python is becoming the de facto standard for exploratory and interactive scientific research

BUT

- Python is no programming silver bullet
- Your application will ultimately dictate the tool (and a mixture of more than one language **is** ok). For example:
  - MATLAB excels at interfacing with hardware, e.g. connecting to configure an integrated circuit board or connecting to a
  - R is great for data wrangling and visualization and statistical modelling
  - (a probabilistic programming language) is an excellent choice for performing full Bayesian statistical inference

Horses for courses

- Python is becoming the de facto standard for exploratory and interactive scientific research

  BUT

- Python is no programming silver bullet
- Your application will ultimately dictate the tool (and a mixture of more than one language **is** ok). For example:
  - MATLAB excels at interfacing with hardware, e.g generating hardware description language (HDL) code to configure an integrated circuit board or connecting to a data acquisition card
  - R is great for data wrangling and visualisation, and statistical modelling
  - Stan (a probabilistic programming language) is an excellent choice for performing full Bayesian statistical inference

Horses for courses

- Python is becoming the de facto standard for exploratory and interactive scientific research

  BUT

- Python is no programming silver bullet
- Your application will ultimately dictate the tool (and a mixture of more than one language **is** ok). For example:
  - MATLAB excels at interfacing with hardware, e.g generating hardware description language (HDL) code to configure an integrated circuit board or connecting to a data acquisition card
  - R is great for data wrangling and visualisation, and statistical modelling
  - Stan (a probabilistic programming language) is an excellent choice for performing full Bayesian statistical inference

Horses for courses

- Python is becoming the de facto standard for exploratory and interactive scientific research

  BUT

- Python is no programming silver bullet
- Your application will ultimately dictate the tool (and a mixture of more than one language **is** ok). For example:
  - MATLAB excels at interfacing with hardware, e.g generating hardware description language (HDL) code to configure an integrated circuit board or connecting to a data acquisition card
  - R is great for data wrangling and visualisation, and statistical modelling
  - Stan (a probabilistic programming language) is an excellent choice for performing full Bayesian statistical inference

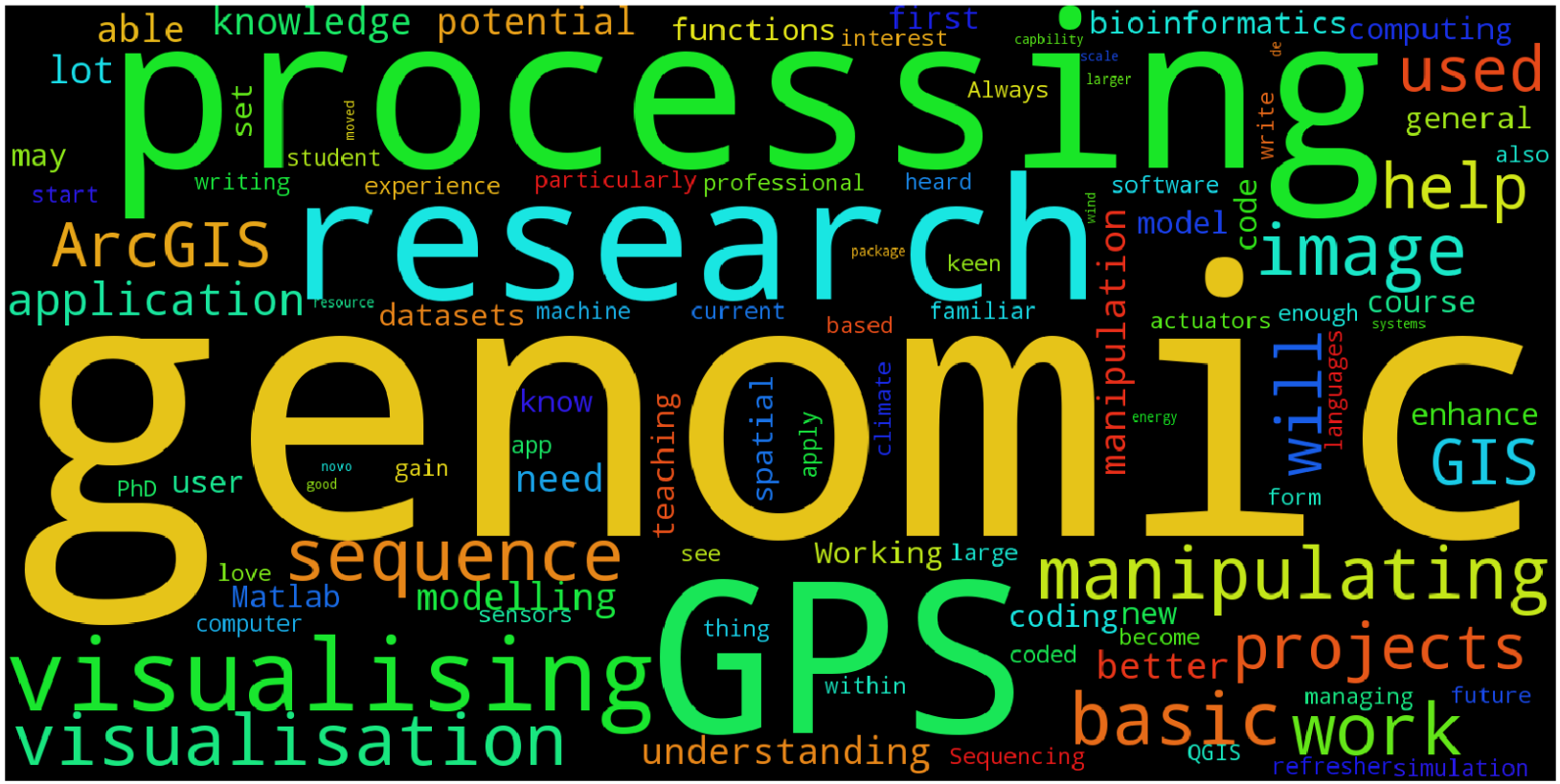
Horses for courses

- Python is becoming the de facto standard for exploratory and interactive scientific research

  BUT

- Python is no programming silver bullet
- Your application will ultimately dictate the tool (and a mixture of more than one language **is** ok). For example:
  - MATLAB excels at interfacing with hardware, e.g generating hardware description language (HDL) code to configure an integrated circuit board or connecting to a data acquisition card
  - R is great for data wrangling and visualisation, and statistical modelling
  - Stan (a probabilistic programming language) is an excellent choice for performing full Bayesian statistical inference

Why do *you* want to learn Python?



Executing Python code: No frills Python interpreter

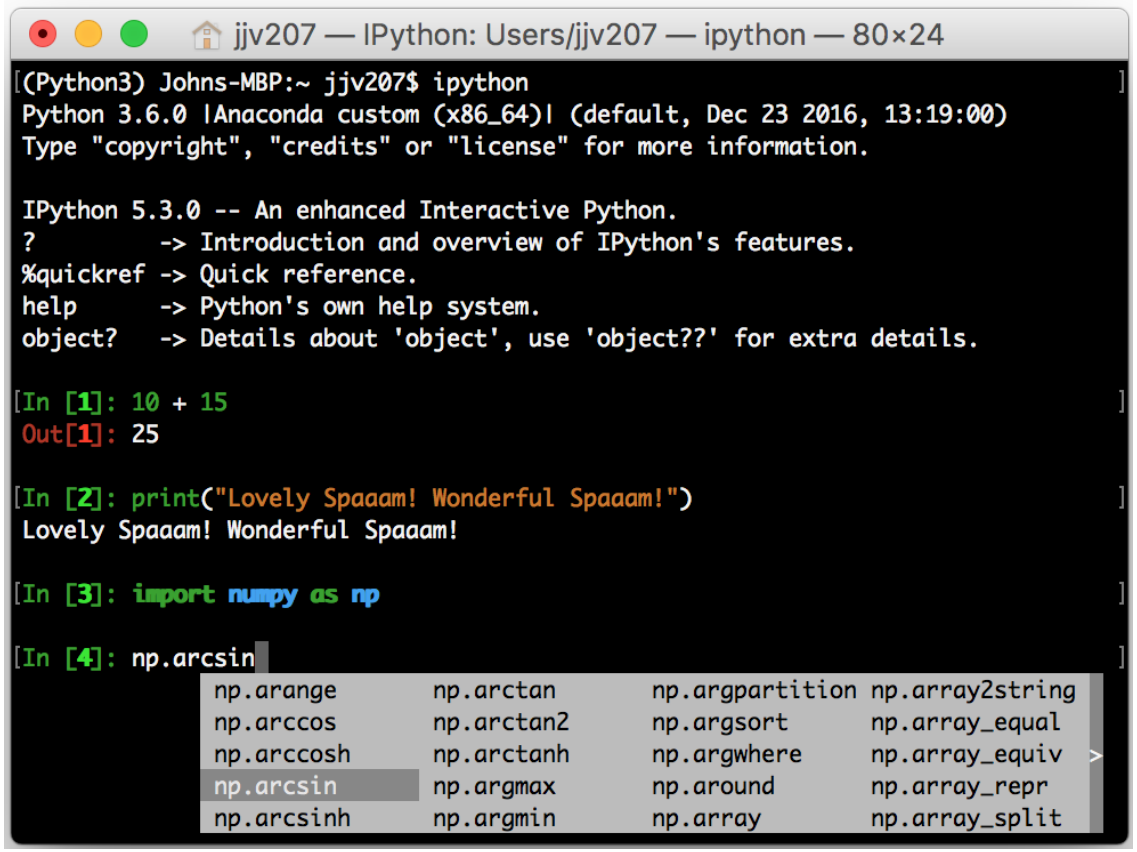
- Type `python` in your terminal window to invoke the interpreter
- Any Python code you type in is executed once you press enter

- Alternatively if your code is written in a text file, e.g `my_script.py`:

```
python my_script.py
```

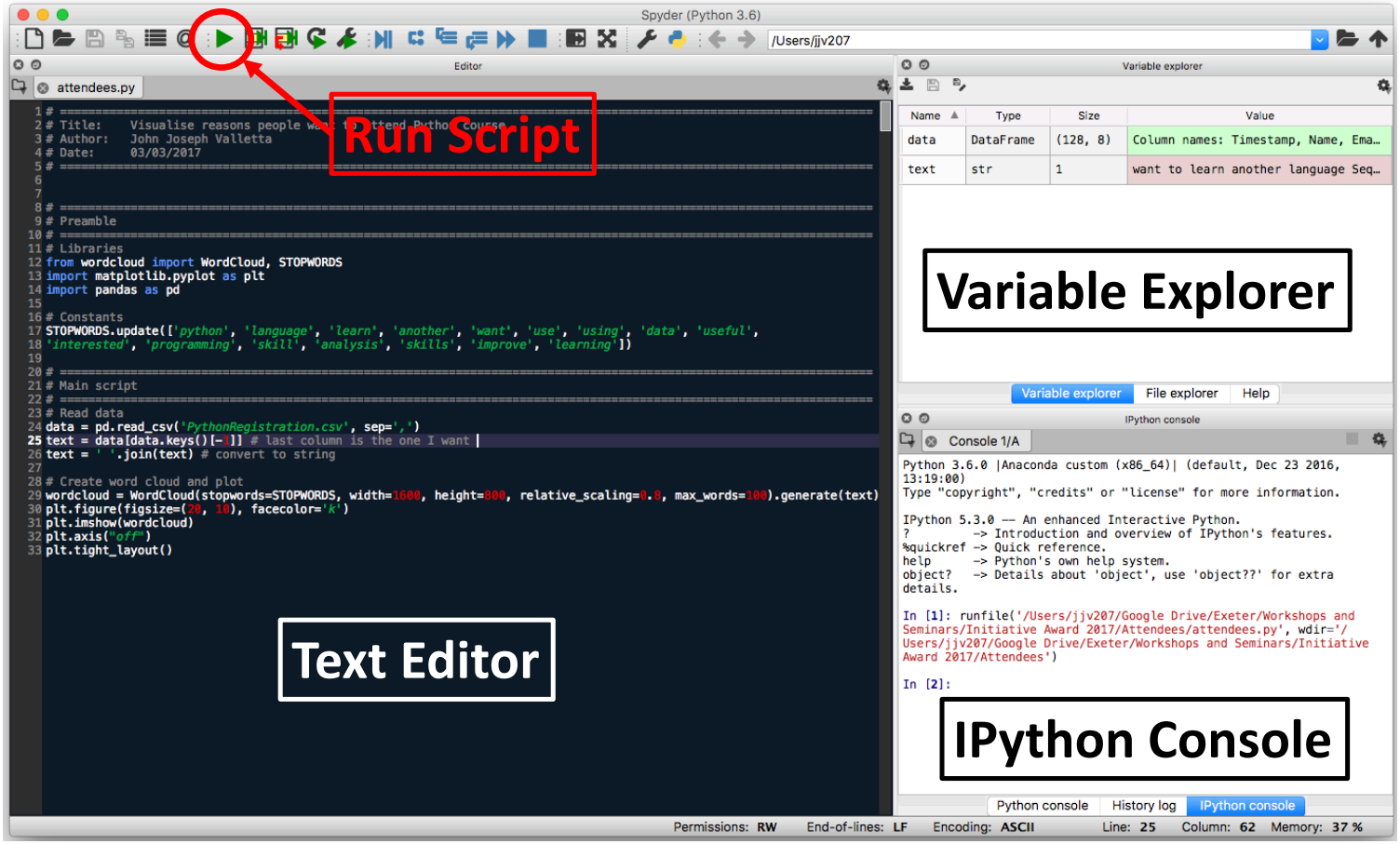
Executing Python code: IPython interpreter

- IPython is an interactive shell (similar to R Console), adding "frills" to the vanilla interpreter, such as:
  - syntax highlighting (making it easier to read code)
  - tab auto-completion (minimises typeos and lists available functions)



Executing Python code: Spyder IDE

- Spyder is an integrated development environment (IDE) for scientific computing, akin to RStudio and MATLAB
- One place to write, execute and debug code, and explore variables



Python 2.x vs 3.x

- Python 2.x and Python 3.x are the two main versions of Python
- Python 2.x is legacy, Python 3.x is the present and future of the language
- However, not all Python 3.x code is backwards-compatible
- Be aware of key differences between the two
- Here we will use Python 3.x, the language actively being developed

Python 2.x vs 3.x

- Python 2.x and Python 3.x are the two main versions of Python
- Python 2.x is legacy, Python 3.x is the present and future of the language
- However, not all Python 3.x code is backwards-compatible
- Be aware of key differences between the two
- Here we will use Python 3.x, the language actively being developed

Python 2.x vs 3.x

- Python 2.x and Python 3.x are the two main versions of Python
- Python 2.x is legacy, Python 3.x is the present and future of the language
- However, not all Python 3.x code is backwards-compatible
- Be aware of key differences between the two
- Here we will use Python 3.x, the language actively being developed

Python 2.x vs 3.x

- Python 2.x and Python 3.x are the two main versions of Python
- Python 2.x is legacy, Python 3.x is the present and future of the language
- However, not all Python 3.x code is backwards-compatible
- Be aware of key differences between the two
- Here we will use Python 3.x, the language actively being developed

Python 2.x vs 3.x

- Python 2.x and Python 3.x are the two main versions of Python
- Python 2.x is legacy, Python 3.x is the present and future of the language
- However, not all Python 3.x code is backwards-compatible
- Be aware of key differences between the two
- Here we will use Python 3.x, the language actively being developed

Installing Python

- The easiest way to get started is to download and install a cross-platform Python distribution such as:
  - Anaconda
  - Enthought Canopy
- These distributions contain several scientific libraries to get started
- Here we will use the Anaconda Python distribution and Spyder/IPython to write and run our code