Data Programming with Python

Lecture 1

Áine Byrne
aine.byrne@ucd.ie

School of Mathematics and Statistics

UCD ONLINE

Foreword

- These notes have been adapted from those of Andrew Parnell, who previously taught this course.
- These slides are subject to copyright. **Do not** distribute them elsewhere.

What will we cover in this module?

- In this module we will learn how to use the programming language **Python**
- Starting with no previously assumed Python knowledge, we will cover the structure of the language, how to deal with large data sets, produce plots, and access APIs to obtain data from the web
- We will mostly be using the `numPy`, `sciPy` and `pandas` libraries
- By the end of the module you should be very familiar with **obtaining and analysing data**, and **creating graphics** in Python

Why Python?

- Python is open access
- Python is a very popular language for handling big data sets and performing data cleaning (often known as **munging**)
- Many websites allow their data to be used via an **Application Programming Interface** (API), of which a great many are available in Python
- Python is quite easy to learn (compared to other C-like languages) and so it can be quick to get complicated code running
- Python is **faster** than R, but **slower** than other languages such as C/C++/etc. It also doesn't have the statistical package support of R

Module structure

- This module contains approximately **twelve** 45-minute lectures, each broken up into short videos of around 5-15 minutes in length.
- After most videos there are two **optional, non-assessed** exercises for you to complete.
- At the end of each week there are longer exercises for you to complete. These are **assessed** and worth 4% each. Your best 10 assignments will count towards 40% of the final grade.
- Around week 6 there will be a longer **project** due in at the end of week 12. This is worth 10% of the final grade
- A final **computer lab exam** will be worth 50% of the final grade
- You will find all of the material on Brightspace, separated by week.

Module structure 2

- I will use the standard UCD **grading scale**: `http://www.ucd.ie/t4cms/Undergraduate%20Grade%20Descriptors.pdf`
- It's easy to fall far behind in an online course. Make sure you carefully note the **due date** of the assessments given through Brightspace. In general, you can work ahead if you like but you can only be one week behind the lecture notes
- If you're running into problems please contact me as soon as possible. We have systems in place to help students who are falling behind
- If you are **sick or run into unforeseen problems**, please contact us and fill in an **extenuating circumstances** form: `www.ucd.ie/students/studentdesk/extenuating.html`

Slides

EX SC

- Most slides will contain bullet points and Python code. The code will look like this:

```
In[1]: print("Hello World!")
Hello World!
```

- The code that I have typed into Python will be in **blue** and the prompt and output will be in black
- If there are exercises associated with a slide, these will be in the top right corner labelled EX with a number after it indicating which exercise is to be done. If the exercises are **non-assessed** the EX will be in green and if they're **assessed** the EX is in red
- If there is a screencast demonstration associated with a slide, the top corner will contain SC followed by a number. The relevant screencast will be in the **screencast** folder

Communicating with us and your fellow students

- Please use the module **discussion boards** to post queries, thoughts, new discoveries, etc
- If you are stuck on an exercise then please **put your question on the discussion board**
- If you are answering someone else's problem please try **not** to post the solution. Instead try and guide them towards it
- Make sure you are familiar with the UCD plagiarism guidelines: `www.ucd.ie/t4cms/Plagiarism_Policy_Academic_Policy_2005.pdf`

Other resources for learning Python

EX1

- Will we be using the following books in the module (you don't need to buy any of them):
  - Python for Data Analysis by McKinney
  - A Primer on Scientific Programming with Python by Langtangen
  - Python Programming for the Absolute Beginner by Dawson
- There is also a very nice basic introduction at Code Academy: `http://www.codecademy.com/en/tracks/python`
- There is a nice tutorial and much more detail at the main Python website: `http://www.python.org`

Getting started

Canopy - Getting help - Differences between Python and R - A first session

Getting started

SC1

- We are going to use **Canopy** to do all our Python programming. It is free for academic use and is cross-platform too
- Canopy contains both an **Integrated Desktop Environment** (IDE) and most of the Python libraries (`NumPy`, `SciPy`) we will need for this module
- You can install it from: `www.enthought.com/products/canopy/`
- If you want to use your own Python IDE that's fine
- See screencast 1 for a walkthrough of installing Canopy

Getting help

- If you know the **name** of a command, simply type `help(command)` or `command?` at the command prompt and a description with examples will appear
- If you want to find which **methods** are available for a specific library you can type `help(library)`
- If you're **totally stuck** and don't know what to ask for, you can always use Google.
- Stack Overflow is good resource for when you're stuck `http://stackoverflow.com/questions/tagged/python`
- As we will be mostly using the pandas library the website `http://pandas.pydata.org` might also be useful

Three differences between Python and R

1. Python uses **indentation** rather than braces to structure code:

```
In[1]: if x < 3:
           y = 2
```

2. Python uses **pass-by-reference**:

```
In[2]: a = [1, 2, 3]
In[3]: b = a
In[4]: a.append(4)
In[5]: b
Out[5]: [1, 2, 3, 4]
```

3. Indexing starts from 0 (like C) rather than 1 (like R)

A first session

SC2

- In this session, we're going to learn how to use Python to analyse the **Diamonds** data set.
- You can follow along with what I'm doing by:
  - typing the code yourself,
  - downloading the code from Brightspace,
  - or watching the screencast associated with these slides
- As a first step, go and **download** the `diamonds.csv` file from the Brightspace **data** folder
- Save the file in a place where you can easily find it and **know the correct file path**
- If you get stuck, post in the **discussion forum** for week 1

Importing the data

SC2

To import the data set (`diamonds.csv`) use the commands:

```
In[1]: import pandas as pd
In[2]: diamonds = pd.read_csv('/path/to/diamonds.csv')
```

- The first command above is **very common** in Python. We import the Pandas library as `pd` because we are going to use that command often so giving it a shorter name will **save us** lots of typing!
- The second command **loads in** the CSV file. The column names correspond to the price (US$), carat, cut, colour (D=best), clarity (I1 worst, SI1, SI2, VS1, VS2, VVS1, VVS2, IF best), length/width/depth (x/y/z), depth, and table (width of the top relative to widest point))

Exploring the data

SC2

- We can now perform some basic **exploratory data analysis**:

```
In[1]: diamonds[:5]
In[2]: diamonds.cut.value_counts()
In[3]: pd.pivot_table(diamonds, 'price',
           index='color')
In[4]: diamonds.ix[0]
```

- I've truncated the output here for space but these should show
  - The first 5 **records**
  - A **count** of all the different values of the cut variable
  - A **table** of the average price indexed by colour
  - The first record **index**

Basic statistical modelling

SC2

- Now suppose we want to fit a **linear regression** model that tries to estimate price from the other variables
- We can use the `statsmodels` package:

```
In[1]: import statsmodels.formula.api as smf
In[2]: mod = smf.ols("price ~ carat",
           diamonds).fit()
In[3]: mod.summary()
```

- Note: an easy way to install extra packages is to go to Tools > **Package Manager** in Canopy
- The results should show that Carat is a **very good predictor** of price

A simple scatter plot

SC2

- We could now create a simple **scatter plot** of carat vs price to check the relationship

```
In[1]: diamonds.plot(x='carat',y='price',
           style='.')
```

- The default plot command for a `pandas` data frame is a line plot, but setting `style='.'` changes it to a scatter
- Note that the y-axis is **too big**. We can check why this is:

```
In[2]: diamonds.price.max()
In[3]: diamonds[diamonds.price ==
           diamonds.price.max()]
```

Congratulations!

EX1

- You have just **completed** your first Python session!
- You have learnt how to read in data sets, perform simple statistical analysis, and plot data
- See the **screencast 2** for a few more commands that can be run on the diamonds data
- When you are happy with the content in this first session, have a go at **assessed exercises 1**