GEOG59950 Programming for Geographical Information Analysis: Core Skills

Contact: F.L.Pontin@leeds.ac.uk

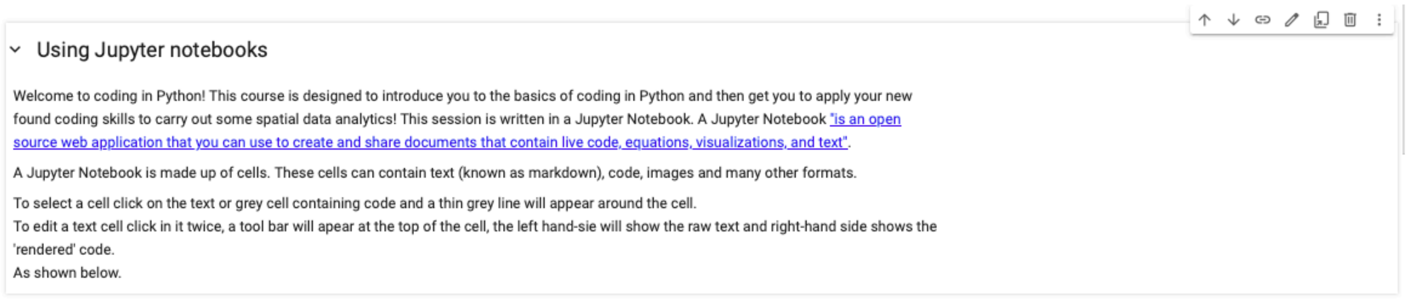
Exercise 2: Getting started

- Data types

Using Jupyter notebooks

Welcome to coding in Python! This course is designed to introduce you to the basics of coding in Python and then get you to apply your new-found coding skills to carry out some spatial data analytics! This session is written in a Jupyter Notebook. A Jupyter Notebook ["is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text"](https://realpython.com/jupyter-notebook-introduction/).

A Jupyter Notebook is made up of cells. These cells can contain text (known as markdown), code, images and many other formats. E.g. To select a cell click on the text or grey cell containing code and a thin grey line will appear around the cell and a small toolbar will appear on the top right-hand side of the cell.

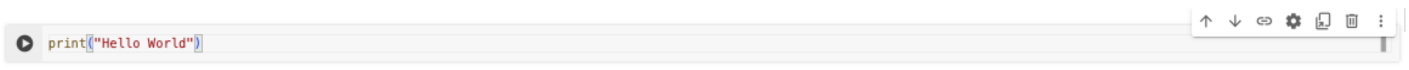


Using Jupyter notebooks

Welcome to coding in Python! This course is designed to introduce you to the basics of coding in Python and then get you to apply your new found coding skills to carry out some spatial data analytics! This session is written in a Jupyter Notebook. A Jupyter Notebook "is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text".

A Jupyter Notebook is made up of cells. These cells can contain text (known as markdown), code, images and many other formats.

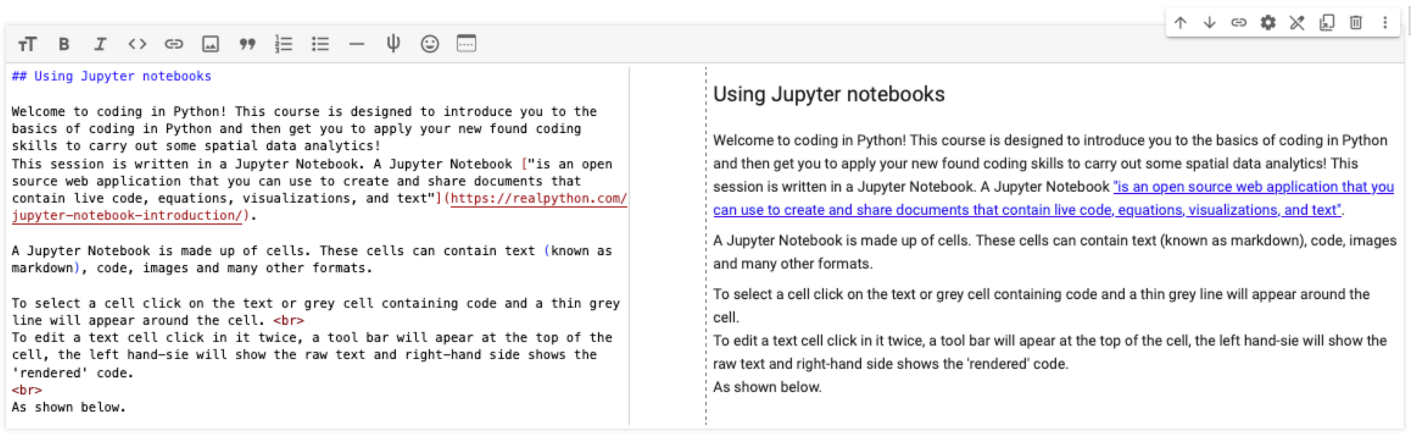
To select a cell click on the text or grey cell containing code and a thin grey line will appear around the cell.
To edit a text cell click in it twice, a tool bar will apear at the top of the cell, the left hand-sie will show the raw text and right-hand side shows the 'rendered' code.
As shown below.



```
print("Hello World")
```

Have a look at all the functions on the right-hand toolbar ('Move up', 'Move down', 'Copy link to cell', 'Edit', etc...)

To edit a text cell click in it twice, a toolbar will appear at the top of the cell, the left hand-side will show the raw text and right-hand side shows the 'rendered' code.



```
## Using Jupyter notebooks

Welcome to coding in Python! This course is designed to introduce you to the
basics of coding in Python and then get you to apply your new found coding
skills to carry out some spatial data analytics!
This session is written in a Jupyter Notebook. A Jupyter Notebook ["is an open
source web application that you can use to create and share documents that
contain live code, equations, visualizations, and text"](https://realpython.com/
jupyter-notebook-introduction/).

A Jupyter Notebook is made up of cells. These cells can contain text (known as
markdown), code, images and many other formats.

To select a cell click on the text or grey cell containing code and a thin grey
line will appear around the cell. <br>
To edit a text cell click in it twice, a tool bar will apear at the top of the
cell, the left hand-sie will show the raw text and right-hand side shows the
'rendered' code.
<br>
As shown below.
```

Using Jupyter notebooks

Welcome to coding in Python! This course is designed to introduce you to the basics of coding in Python and then get you to apply your new found coding skills to carry out some spatial data analytics! This session is written in a Jupyter Notebook. A Jupyter Notebook "is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text".

A Jupyter Notebook is made up of cells. These cells can contain text (known as markdown), code, images and many other formats.

To select a cell click on the text or grey cell containing code and a thin grey line will appear around the cell.
To edit a text cell click in it twice, a tool bar will apear at the top of the cell, the left hand-sie will show the raw text and right-hand side shows the 'rendered' code.
As shown below.

During the teaching session you will read through the Jupyter Notebok, running sections of code yourself to learn key data analysis skills. There will also be places to edit and write your own code within the notebook.

- Instructions and tasks for you to complete are in purple
- Where you have to write your own code answers are provided at the end of the section

The first few exercises are a recap from last week

Hello World

Following coding tradition the first thing we are going to program is to print the words "Hello World".

The `print()` function prints the specified message to your screen

The speech marks `" "` arround Hello World let python know you are typing text (also known as a string).

Run the code below

```
print("Hello World")
```

Basic maths

Lets do some basic maths
Run the code below

```
2+2
```

If we print 2+2 we get the same thing
Run the code below

```
print(2+2)
```

If we `print("2+2")` however we get the characters 2 + 2 as the speach marks tell Python we are typing text and not numerical characters

Run the code below and see for yourself

```
print("2+2")
```

Assigning values to variables

We can also assign the sum 2+2 to a variable.

In this case we have named the variable answer. We can now type answer instead of 2+2 every time.

To assign 2+2 to the variable answer run the code below

```
answer = 2+2
print(answer)
```

We can also add our created variable to a new equation
To add 5 to the variable answer run the code below

```
# we can also add our created variable to a new equation
answer + 5
```

NOTE: `#` When a hashtag is added to a line of code the rest of that line is not treated as code. This can be used to add comments to your code so you know what it is doing (especially useful when you come back to it later on!)

Lists

Remember lists are an ordered collection of one or more data item. Defined by square brackets `[]`

Look back at the lecture notes and create a list called fruit with the following data items: 'apple', 'pear', 'banana', 'strawberry' (note these are strings)

```
# Type code here
```

We can check that you have created a list using the `type` function

Enter `type(fruit)` into the cell below

```
# run this code
```

Lists and basic maths

We can also create a list of numbers.
Run the code below . What happens when we try to multiply the list by 3?

```
numbers = [1,3,5,7,9,11]
numbers*3
```

List comprehension

To multiply each item in the list (and create a new list of the results) we need to select each value `i` in the list: `[i*3 for i in numbers]`

I.e. for each element 1 to i in the list named numbers, multiply that element by 3

Run the code to see

```
# multiply element by 3, repeat for every element in the list 'numbers'
new_numbers = [i*3 for i in numbers]

# print the new list
print(new_numbers)
```

A very quick introduction to for loops

Making a list based on an old list is known as list comprehension. An alternative is a for loop as shown below.

We will come back to these later, so for now just **run the code and read the comments explaining what each step does.**

```
# create a new empty list
my_new_list = []

# for element in the list 'numbers':
for i in numbers:
    # multiply element by 3 and append the result to the new empty list
    my_new_list.append(i * 3)

# print the new list
print(my_new_list)
```

We can also multiply by another defined variable. For example we previously defined the variable `answer` (`answer = 2+2`)

```
# multiply element by variable 'answer', repeat for every element in the list '
[i*answer for i in numbers]
```

Commonly you will see `i` and `j` to define elements in a list. However this is just convention and you could use anything you wanted to refer to the elements in the list e.g. `elephants`

(Though typing `i` is a lot quicker)

```
# multiply element by 3, repeat for every element in the list 'numbers'
[elephants*3 for elephants in numbers]
```

First look at data frames

.head() and .tail() functions

To understand how to explore data frames and different data types in python we are going to use a set of data about passengers on the titanic. This is an example dataset built into the seaborn python package.

We will go into detail about reading in data and loading packages in the next exercise, **for now run the cell of code below.**

Note the `.head()` function shows the top 5 lines of the data frame.

```
# Import the seaborn package
import seaborn as sns

# load the titanic example dataset and save it as a dataframe named titanic
titanic = sns.load_dataset('titanic')

# look at the first 5 rows of the dataframe
titanic.head()
```

Note NaN denotes a cell containing no data - a null cell

Try entering and running `titanic.tail()` instead of `.head()` in the cell below. What view of the dataframe do you think you are now seeing?

```
# enter the instructed code here
```

A Quick description of the titanic data variables:

- ***survival:*** *If the passenger survived*
- ***PassengerId:*** *Unique Id of a passenger.*
- ***pclass:*** *Ticket class*
- ***sex:*** *Sex*
- ***Age:*** *Age in years*
- ***sibsp:*** *Number of siblings / spouses aboard the Titanic*
- ***parch:*** *Number of parents / children aboard the Titanic*
- ***ticket:*** *Ticket number*
- ***fare:*** *Cost of the passenger fare*
- ***cabin:*** *Cabin number*
- ***embarked:*** *Port of Embarkation*

Data frame columns

We might just want to get a list of the columns in the dataframe to give us a quick idea of what data we have present. To do this we can use the `.columns` function after we name the dataframe.

Enter `titanic.columns` in the cell below. . The columns listed should be the same as the columns in the `.head()` view of the dataframe.

```
# enter the code here
```

Referring to a column in a dataframe

If we want to select a single column of the dataframe we can also do that.

There are several ways to refer to a column in a pandas dataframe.

The easiest way is by putting the name of the column in square barckets and speech marks `[" "]` after the name of the dataframe. e.g. `dataframe_name["column_name"]`

Try to select the 'fare' column from the titanic dataframe.

```
# enter the code here
```

*Note: To save space only a snapshot of the column is shown and not all the rows

Data frame index

When we type `dataframe_name["column_name"]` we get the values of the column but we also see the index (the row names). In this case the rows are just numbered 0:890. But these could be other values such as passenger names.

Python indexing starts at 0 not 1. So the first row is row 0 and the first column column 0.

`.index` works the same as `.columns` but this time shows the row names.

Run `titanic.index` in the cell bellow

```
# enter the code here
```

We can see the index starts at 0, stops after 891 enteries and increases by a step of 1 for each row.

Data frame shape

To get the number of rows and columns of a dataframe we can use `.shape`

Run `titanic.shape` in the cell bellow

```
# enter the code here
titanic.shape
```

Data Types

Data types recap

As we covered in the lecture there are different types of data:

Objects: also known as strings or written characters/text in plain english e.g. Hello World

Intergers: Whole numbers e.g. 2, 57 or 109567835

Floats: A number with a decimal place e.g. 2.34534, 5.5 or 1.0

Boolean: True or False data type

Datetime: Values that are either a date, time or both e.g. 2019-10-31 09:26:03.478039 (9:26 am on Halloween 2019)

Category: A fintie list of text values E.g. London, Paris, Berlin, Rome (There are a finite number of captial cities)

Learn more about python data types using this realpython online resource

Checking the data type

We can check the data type of each column in a dataframe using the `.dtypes` funciton **Run the code below** and have a look at the data type of each of the columns. Are they all as you expected?

```
titanic.dtypes
```

`.info()` gives us slightly more information including:

- data types: (`.dtypes`)
- null counts: number of rows containing non-null values
- memory usage: how much computer memory the table uses (useful to know to stop your code running)

If a column has fewer non-null values than the total number of rows this indicates that data might be missing.
Run the code below

```
titanic.info()
```

Extra tasks

If you have time at the end of the session run the code below to load the penguins dataset and answer the following question using your new found coding skills.

- How many columns and rows does the penguins data frame have?
- What are the data types of the different columns in the penguindataframe?
- Is there any data obviously missing from the dataframe?
- Can you use markdown to produce a list fo the column names and variable types for the penguins data set as I have done above for the titanic data set

Work through the following:

- https://www.programiz.com/python-programming/comments
- https://www.programiz.com/python-programming/variables-constants-literals
- https://www.programiz.com/python-programming/variables-datatypes
- https://www.programiz.com/python-programming/operators

```
# load the penguins example dataset and save it as a dataframe named penguins
penguins = sns.load_dataset('penguins')
```

```
# How many columns and rows does the coffee data frame have?
penguins
```

```
# What are the data types of the different columns in the penguins dataframe?
# Is there any data obviously missing from the dataframe?
```

Use the following code to get a list of other dataframes you can explore `sns.get_dataset_names()`, can you produce a summary of what the data is showing and idenitfy where data is missing?

You might need to google the dataset to understand the variable names. You might want to create a markdown data summary similar to the one I created for the titanic dataset:

A Quick description of the titanic data variables:

- ***survival:*** *If the passenger survived*
- ***PassengerId:*** *Unique Id of a passenger.*
- ***pclass:*** *Ticket class*
- ***sex:*** *Sex*
- ***Age:*** *Age in years*
- ***sibsp:*** *Number of siblings / spouses aboard the Titanic*
- ***parch:*** *Number of parents / children aboard the Titanic*
- ***ticket:*** *Ticket number*
- ***fare:*** *Cost of the passenger fare*
- ***cabin:*** *Cabin number*
- ***embarked:*** *Port of Embarkation*

```
sns.get_dataset_names()
```