

# Welcome to Foundations of Python

Hamish Gibbs

# Course aims

- A foundation in Python programming.
  - *Variables, data structures, control logic, functions, classes.*
- An Introduction to popular Python tools for data science.
  - *pandas, matplotlib, sklearn.*
- A hands-on data science challenge.
  - Predicting the price of AirBnBs in London.

# Also: tools of ‘professional’ Python

- Using an Integrated Development Environment
  - *VSCode*
- Using collaboration tools
  - *git*, *GitHub*

# Course format

- *This is a short course!*
- Days 1 – 3: lectures and practicals.
  - Practical sessions rely on existing Python tutorials.
  - Once you have worked through a practical, try to change the examples it provides until you “*get it*”.
  - We are using a variety of open source Python learning resources.
- Day 4: Working together on a collaborative challenge.

# Schedule

- **Day 1:** Python basics
  - Variables, data structures (`list`, `dict`), control logic (`if`, `for`, `while`).
- **Day 2:** Abstraction & composition
  - Functions (`def`), Classes (`class`).
  - *Also:* Using `.py` files, not `.ipynb` in VSCode.

# Schedule

- Day 3: Python data science
  - `pandas`, `numpy`, `matplotlib`.
  - *Also:* Collaboration with `git` and GitHub.
- Day 4: Challenge: regression analysis
  - Predicting the price of London AirBnBs using `Inside AirBnB` data.

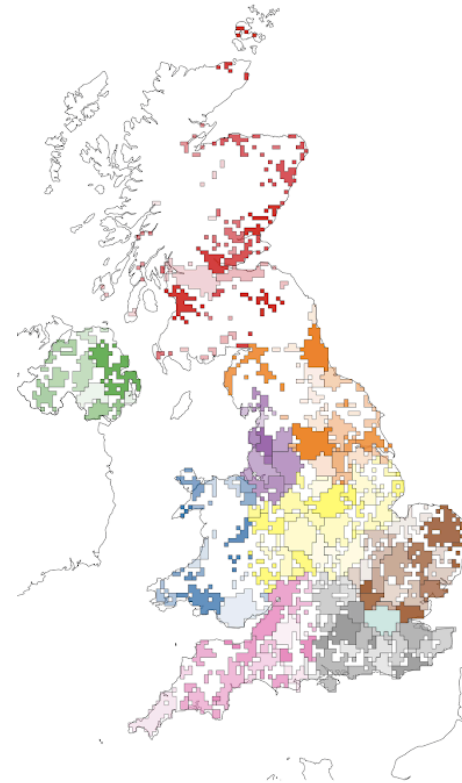
# About you

- Programming experience?
- Statistics experience?
- Any installation problems?

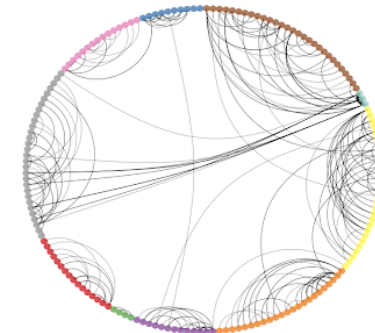
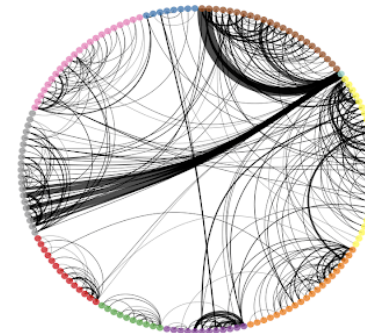
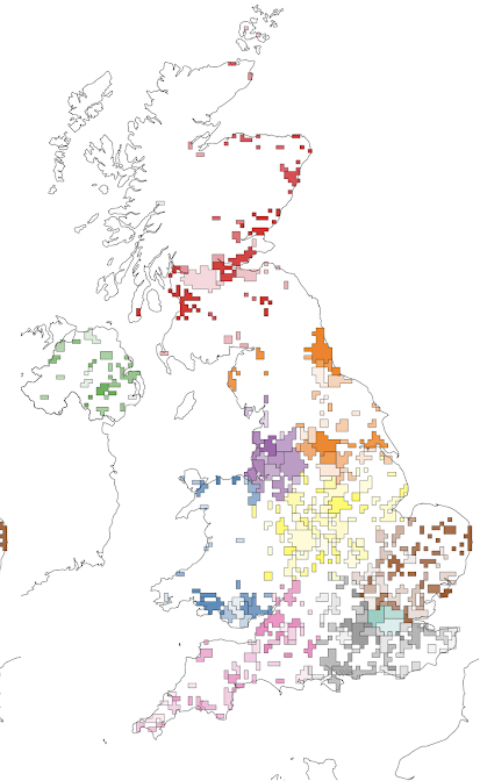
# About me

- Final year PhD student
  - *Supervised by James Cheshire*
- My research interests:
  - *Human mobility, disease transmission, bias & uncertainty*
- Python experience:
  - *9 years*

March 10, 2020



April 13, 2020





# Learning python



# Learning python

- Practice is the most important ingredient to becoming a good programmer.
- It is easier to “practice” if you find *personally compelling* reasons to use Python.
  - Coursework, side projects, random curiosity, automating things in your life.
- Programming is all about trial and error.

# AI

- New AI programming assistants:
  - *Chat GPT, GitHub Copilot, Copilot Chat.*
- I recommend using them all, especially as a study aid.
  - **Bad idea:** Using AI to *generate* code you can't understand.
  - **Good idea:** Using AI to *explain* code you can't understand.

**Any questions?**

# Basic data types

## Integer

```
1 as.integer(10) # R
```

```
1 int(10) # Python
```

## Float

```
1 as.numeric(10.3) # R
```

```
1 float(10.3) # Python
```

## String

```
1 as.character("Hello") # R
```

```
1 str("Hello") # Python
```

## Type checking

```
1 class("Hello") # R
```

```
1 type("Hello") # Python
```

# Tutorial #1: Variables

- Variables, expressions, and statements
- Core concepts:
  - Variable assignment and basic math

```
1 minute = 20  
2 minute + 32
```

- Working with strings

```
1 first = '100'  
2 second = '150'  
3 print(first + second)
```

- Subsetting strings

```
1 word = 'Python'  
2 word[0]
```

# Tutorial #2: Lists

- [An informal introduction to python §3.1.3](#)
- Core concepts:
  - List indexing

```
1 squares = [1, 4, 9, 16, 25]
2 squares[0]
```

- List manipulation

```
1 squares + [36, 49, 64, 81, 100]
2 squares.append(100)
```

- List mutability

```
1 rgb = ["Red", "Green", "Blue"]
2 rgba = rgb
3 id(rgb) == id(rgba) # they reference the same object
```

# One more data structure: tuples

- A **tuple** is an immutable collection of values.

```
1 coord = (0, 1) # a single tuple
2 coords = [coord, coord] # a list of tuples
```

- Unlike a **list**, the values in a **tuple** cannot be changed.
  - This means no **.append()** or **.sort()** methods (which both change the values in a **list**).
- Tuples are faster than lists and good when you have a collection of values that won't have to change once it has been created.



# Just one more data structure: set

- A **set** is an unordered collection with no duplicate elements.

```
1 set1 = {0, 1} # a set
2 set2 = set([0, 1]) # creating a set from a list
```

- A **set** is mutable (its values can be changed).
- It is very quick to check if an element is in a **set**:

```
1 1 in {0, 1} # I'm faster!
2 1 in [0, 1]
```

- This doesn't matter for a collection of 2 values, but matters a lot for larger collections.

# Tutorial #3: Dictionaries

- Dictionaries §1

- *To start, only work on the first section of this tutorial!*
- *If you “get it,” try some of the exercises in Tutorial #1.*

- Core concepts:

- The `{key: value}` format of dictionaries

```
1 eng2sp = dict()  
2 eng2sp['one'] = 'uno'  
3 eng2sp['one']
```

- The `.items()`, `.keys()`, and `.values()` methods

# Using Python

- [Interactive shell](#)

```
1 >>> print("Hello")  
2 Hello
```

- *Clunky, ephemeral, hard to use for anything ‘real’.*

- [Google Colab](#)

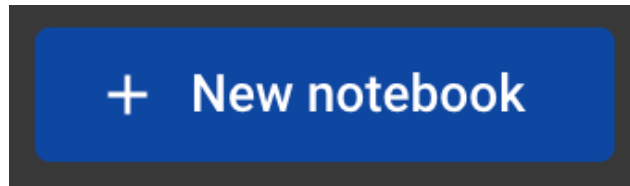
- Google-hosted version of a [Jupyter Notebook](#)
- *A very easy way to start!*

- [Visual Studio Code](#)

- A “serious” integrated development environment.
- *Good for larger projects, collaboration.*

# Setting up Colab

- Does everyone have a Google account?
- Go to [colab.research.google.com](https://colab.research.google.com).
- Open a new notebook.



- Rename your notebook.
- Get started!