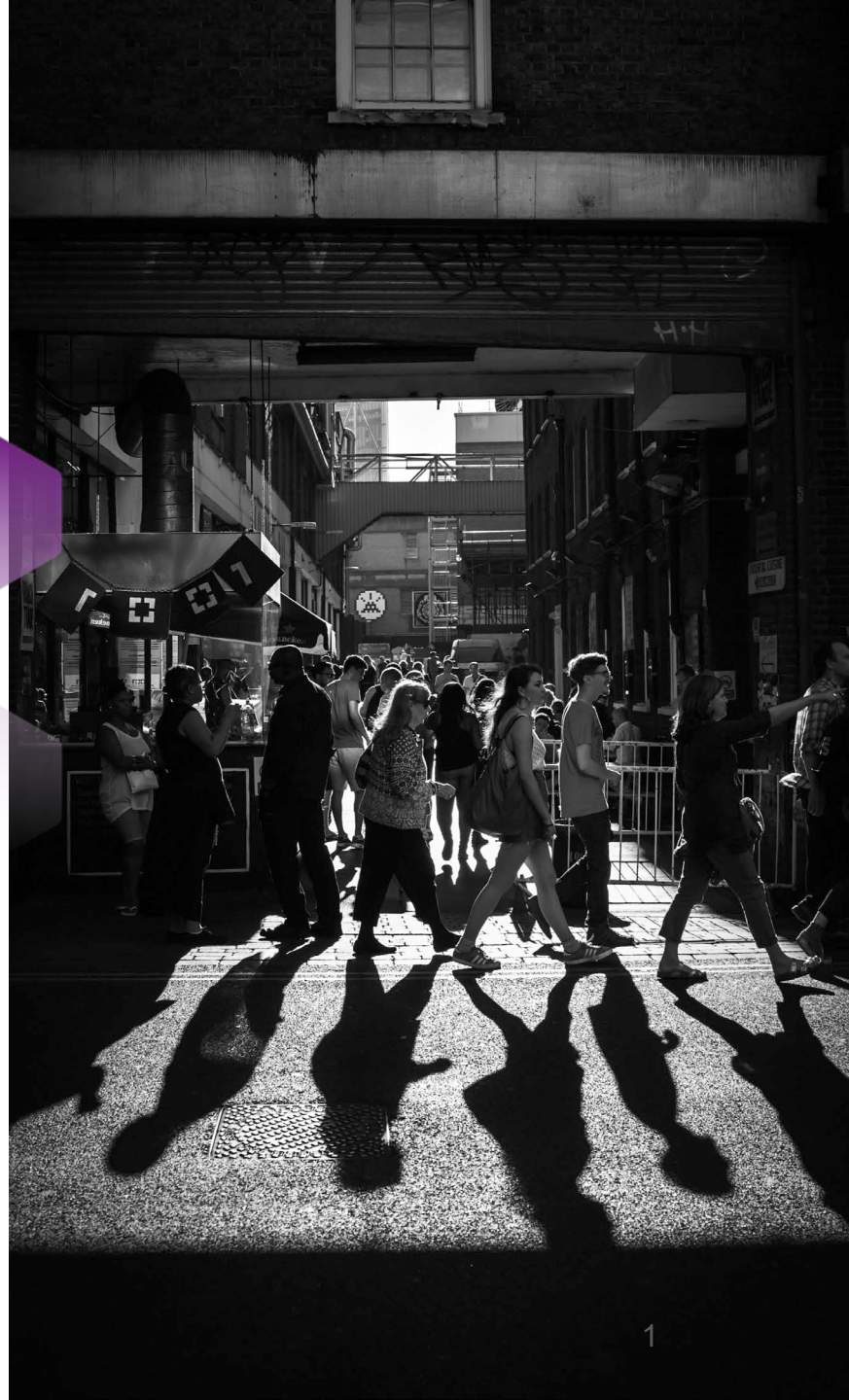
UK Data Service

Code Anxiety Club

Computational Social Science Training Team

UK Data Service



UKRI Economic and Social Research Council

Today's topic

Date (06/01): Introduction to Python and R: Your First Scripts

- Create your first coding script in Visual Studio Code (Python) or Rstudio (R)
- Master basic syntax and get to grips with functions
- Practical exercises using the coding packages 'pandas' (Python) or 'dplyr' (R) to explore a simple dataset

Consoles, Scripts, Notebooks, and the IDE

Thing	Where you see it	What it is	Use it for
Console	VS Code: Terminal Rstudio: Console pane	Type one line, see the result. Nothing is saved unless you copy it out.	Quick checks (“Did pandas load?”, 2+2).
Script (.py, .R)	VS Code: Editor tab in scripts/ folder Rstudio: Source pane, saved in scripts/	A plain text file that you can run any time – your actual program.	Build and run code.
Notebook (.ipynb, .Rmd)	VS Code: Jupyter notebook view Rstudio: R Markdown	Mix of code + text + plots in cells. Great for teaching/guides & reports.	Build and run code in chunks. Exploratory analysis.
IDE (VS Code/ Rstudio)	The whole app you’re using.	Your workbench: editor + console + file browser + plots in one place.	Opening the project folder, running scripts, viewing output. Built-in debugger.

Make your first script (Python)

1. File > New File
2. In the blank script, type:

```
print(“Hello, Code Anxiety Club!”)
name = “your name”
print(“Hi”, name)
```

3. File > Save As > hello_me.py
4. In the top-right hand corner of the screen click the play button to run the script

Make your first script (R)

1. New File > R Script
2. File > Save As > hello_me.R

```
print("Hello, Code Anxiety Club!")
name <- "your name"
print(paste("Hi", name))
```

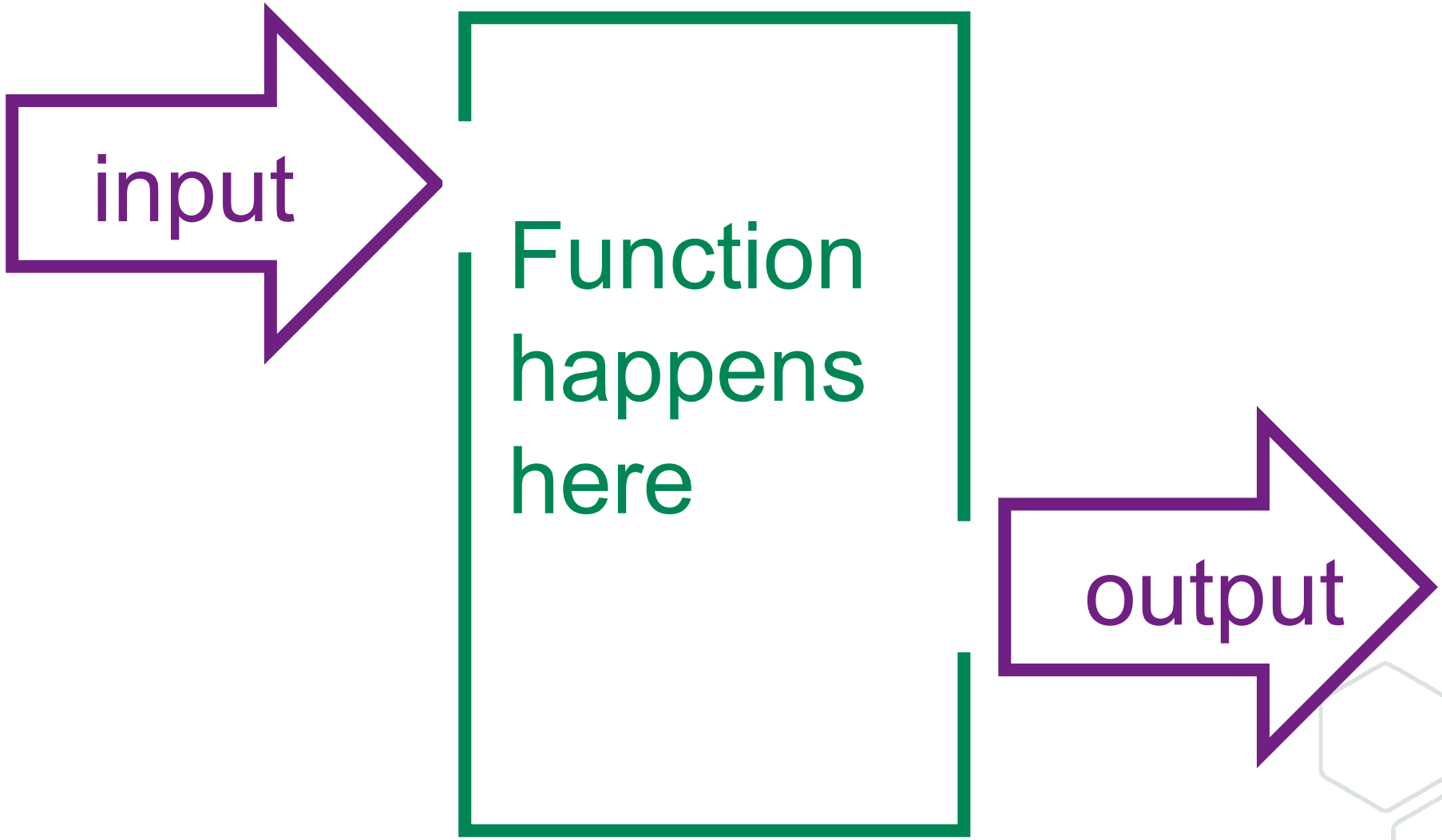
3. In the top-right hand corner of the source panel click the Source button to run the script

Basic Syntax: Variables, Comments, Print

Below are the building blocks that you'll use in most scripts…

Concept	Python	R
Variables	age = 34	age <- 34
Strings	name = "Anna"	name <- "Anna"
Reassigning variables	age = age + 1	age <- age + 1
Comments	# this is a comment	# this is a comment
Print statements	print("Hi", name)	print(paste("Hi", name))
Checking types	type(age)	class(age)

Functions: ‘magic’ transforming boxes



Functions: What are they?

- You've already used functions: print(), paste(), pd.read_csv()
- What is a function? Name + (inputs) → does work → returns a result

Two kinds

1. built-in/package functions: you call them
2. Your own: you define them, then call them the same way

Why write your own?

1. Avoid copy and paste
2. Clear names
3. Reuse with different inputs
4. Easier to test

Creating your own function

Python

```
def bmi(weight, height):
    return weight / (height ** 2)
```

- def starts it
- The colon and indentation make the body
- Return sends the result back
- result = bmi(70, 1.75)
- print(result)

R

```
bmi <- function(weight, height)
{
  weight / (height^2)
}
```

- function(…){…} is the shape
- R returns the last line
- Return is optional but fine to use
- result <- bmi(70, 1.75)
- print(result)

Load a Dataset (pandas/dplyr)

Python

```
import pandas as pd
df = pd.read_csv('superhero_dataset.csv')
```

R

```
library(dplyr)
df <- read.csv('superhero_dataset.csv')
```

Quick Explore: Head, Columns, Filter

- **Head**:
  Python: df.head()
  R: head(df)
- **Select columns**:
  Python: superhero_data_table[['height_cm', ‘age']]
  R: select(superhero_data_table,height_cm,age)
  Filter rows:
  Python: df[df["age"] > 30]
  R: filter(df, age > 30)

Practical Exercise

Python

1. Create scripts/add_two.py
2. Add:

```
# A tiny function and a couple of calls
def add_two(x):
    return x + 2

print(add_two(10))
n = 7
print(add_two(n))
```

R

1. Create scripts/add_two.R
2. Add:

```
# A tiny function and a couple of calls
add_two <- function(x) x + 2

print(add_two(10))
n <- 7
print(add_two(n))
```

UK Data Service

Thank you.

Jools Kasmire

Email j.kasmire@manchester.ac.uk