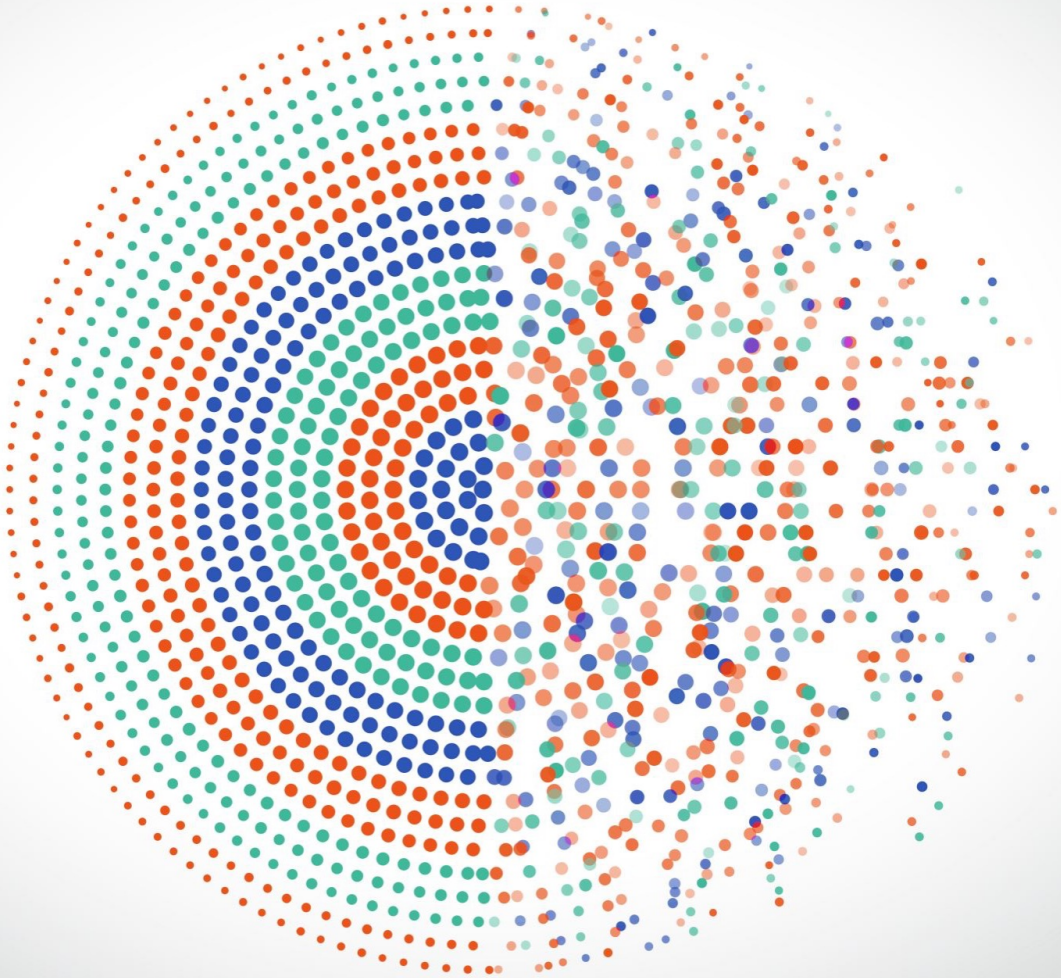
Agent-Based Models for Infectious Disease

Thalia Seale



Plan for Today

Intro to epidemiological modelling

What mechanisms are involved in disease spread?

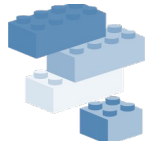
Live demonstration.

Python 101

Quick review of principles of coding in Python

No expectation of previous coding experience

Break



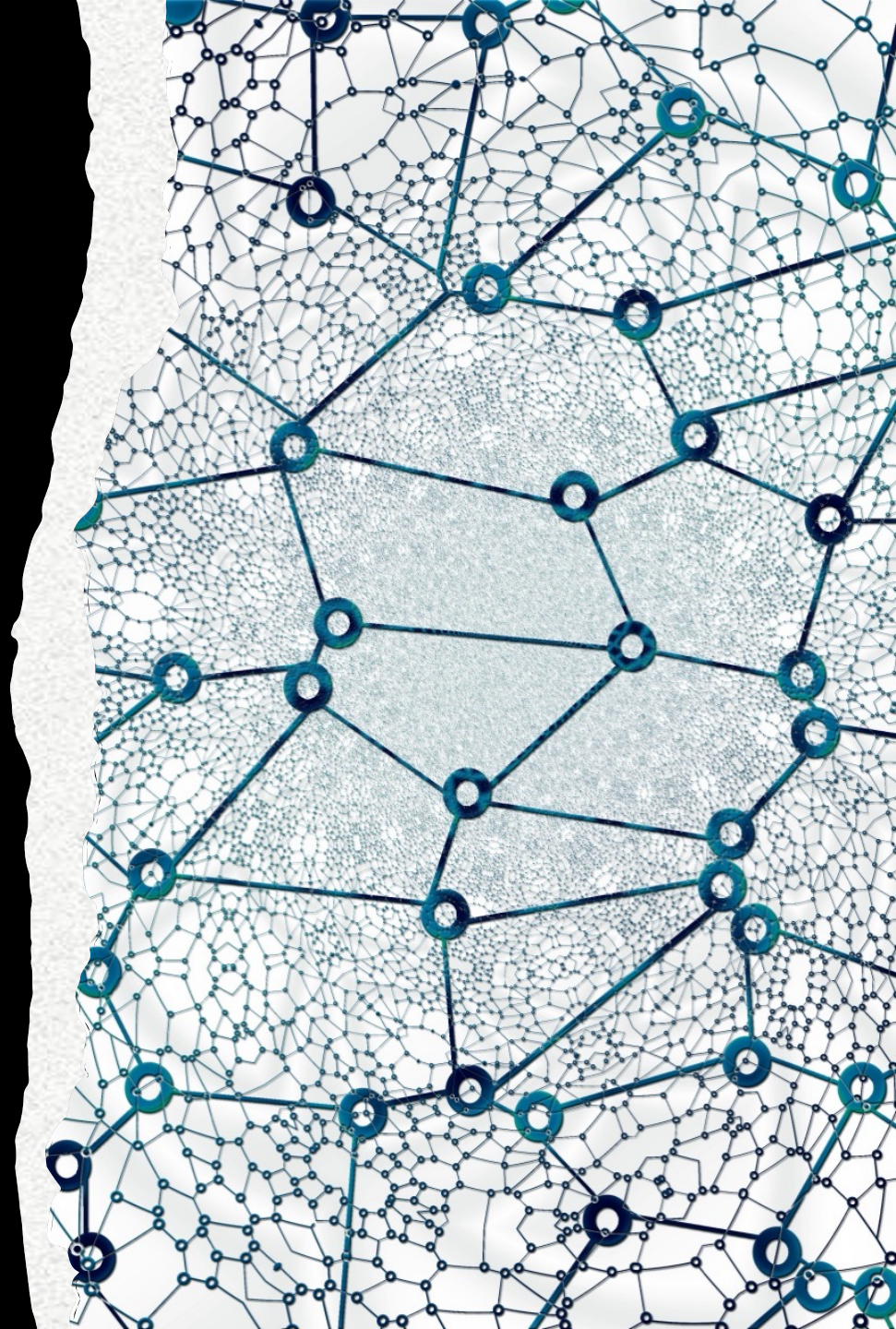
Build an ABM in Python

Range of worksheet difficulties available

Experiments!

Use the coded models to explore different scenarios

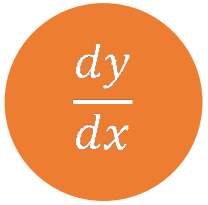
Epidemiological Modelling



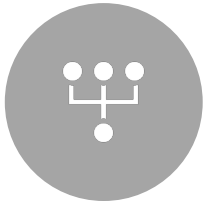
Introduction to Epidemiological Modelling

- What is epidemiology?
  - Epidemiology is the study of the patterns, distribution and determinants of health and disease within populations.
  - Overall aims:
    - Understand causes of disease
    - Predict its course
    - Develop control methods
- We can use mathematical/statistical models to try to understand these patterns.

Example model approaches:



DIFFERENTIAL EQUATIONS



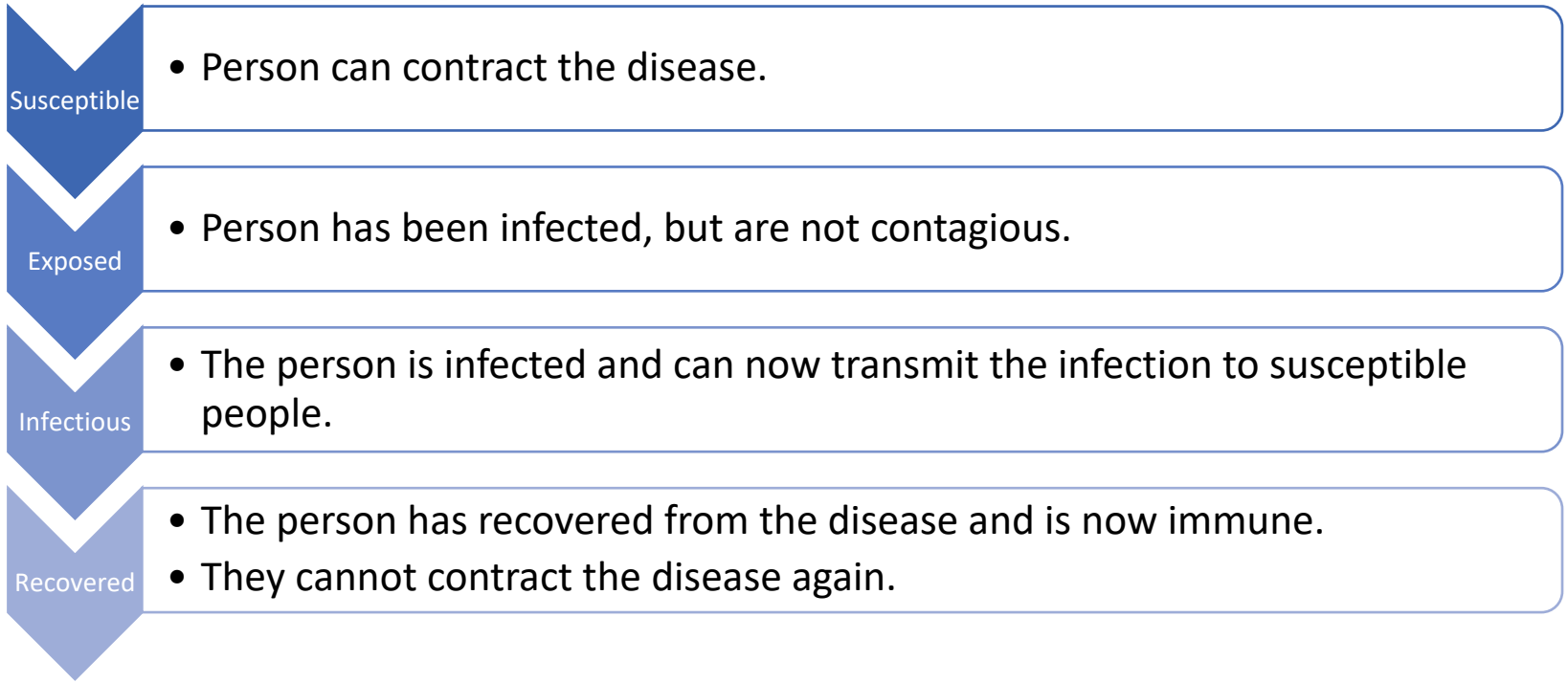
BRANCHING PROCESSES

AGENT-BASED

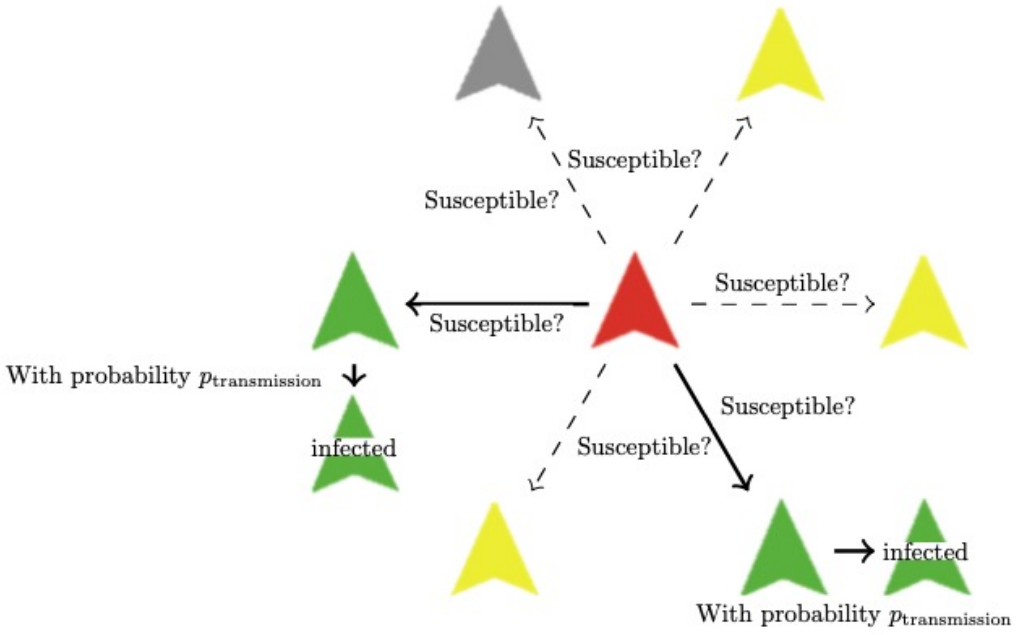
Compartment Modelling

- Provides a framework for describing disease dynamics by dividing the population into categories denoting possible disease states.

SEIR model



Agent-based models (ABMs)

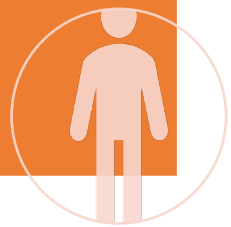


- ABMs attempt to systems by modelling the interactions and behaviours of its components individually.
  - Facilitates integration of a variety of data sources.
  - Allows investigation of all levels of a system.
  - Assumes behaviour of the system emergent from components.

ABMs- Model Components

- The indivIDuals composing the population or system being modelled.
- State variables
  - Variables determining the properties of an agent, such as age or colour, and behavioural attributes, such as movement pattern.

Agents



- The main procedures that comprise an ABM.
- Context
  - The entity or set of entities by which a submodel is performed.

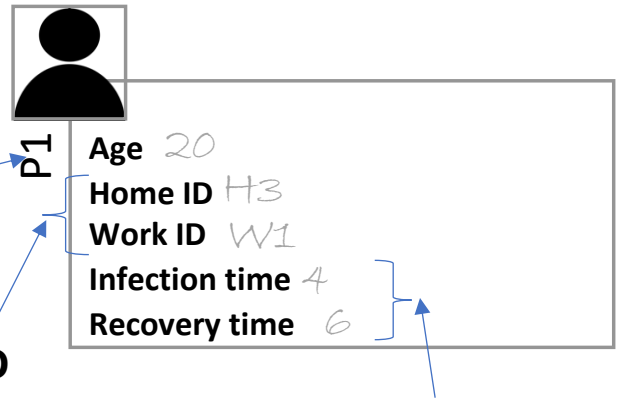
Subroutines

Demonstration

- Each of you gets to be an “agent” in the simulation!
- On your desk each of you has:
  - A die- this is to create randomness in the simulation.
  - A character profile- this will be a record of the “state variables” of your agent.
  - A game piece- you will use this to show the location of your agent.
  - Paper clips- this will be placed on your piece to indicate the disease compartment of your piece.
- Typically, we randomly generate the state variables of each agent since we might not have adequate data on what the starting state of the simulation is, so we want to explore the evolution of the simulation given a variety of different starting points.

Demonstration- Character Card

Agent Name
The number after P will dictate scheduling. We will perform all subroutines starting with P1 and so on.



Home/Work ID
Records the location where your agent lives and works.

Infection/Recovery time
Number of days your agent will take to become infectious/recover from illness

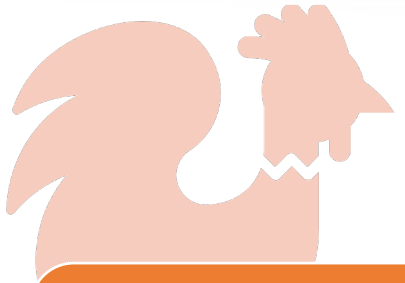
State Variable	Generation Procedure
Age	1. Roll the die. 2. $Age = 10 \times dice\ roll.$
Home ID	1. Roll the die 4 times. 2. Let x be the number of the largest roll. 3. $Home\ ID = x$
Work ID	1. Roll die once. 2. If the value is ≤ 3 the $Work\ ID = W1$, otherwise $= W2$.
Infection/recovery time	Value of a single dice roll.

Demonstration- Setup

- We select 5 people to begin as infected. These agents will have a red paperclip.
- Everyone else will be susceptible and have a green paperclip.
- Place your piece at your home location.

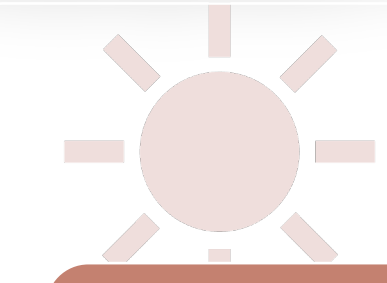
Demonstration- Schedule

- Go to work if age >= 20.
- Become infected if dice roll <2.



Morning

- If your agent is susceptible and there is an infectious person at your location, roll a dice. If it is < 2 then your agent is now exposed.
- If you are exposed or infectious reduce infection/recovery time by 1. If your time = 0, then if you are exposed, you are now infectious. If you are infectious you are now recovered.



Day

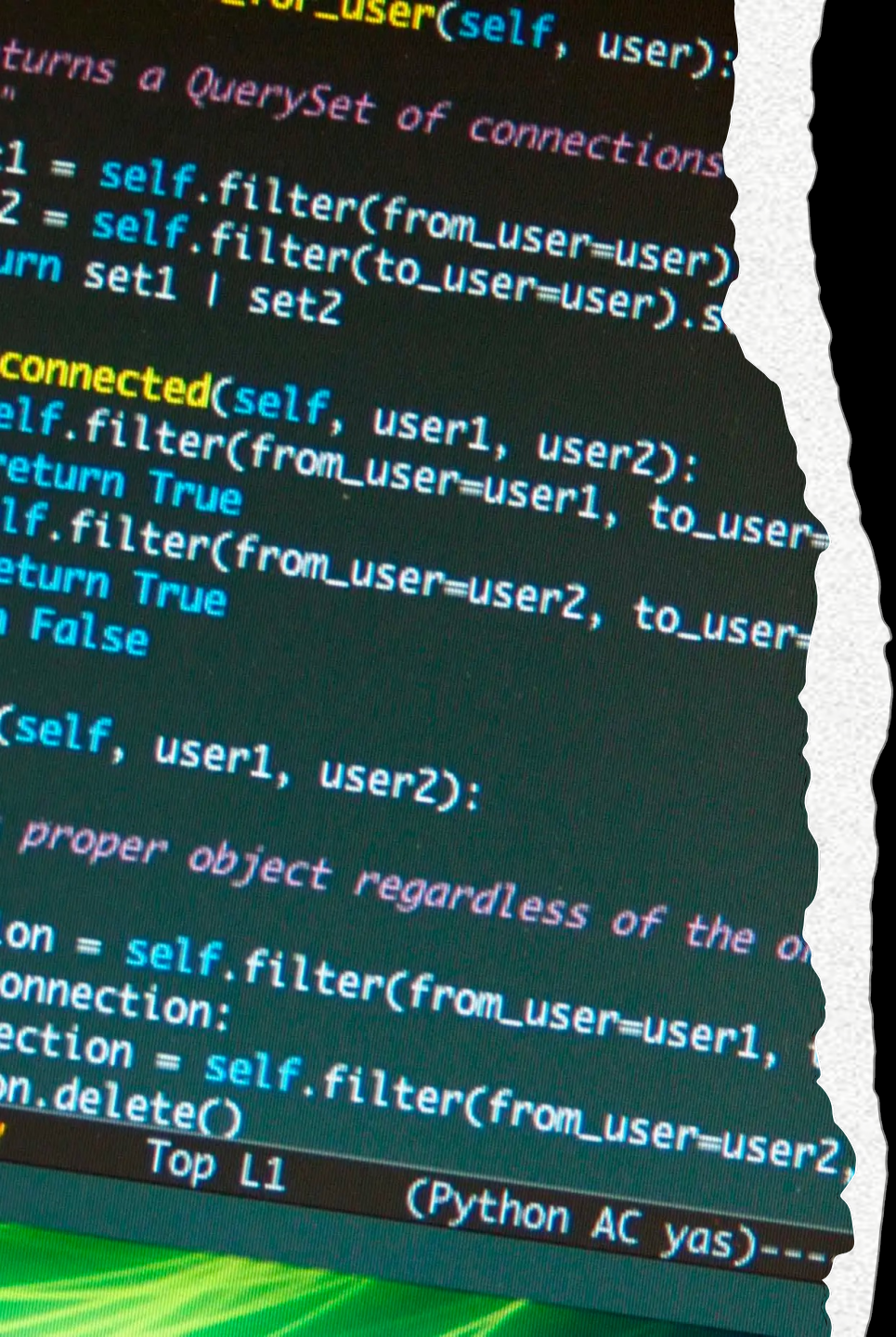
- If your agent is aged >= 20, then move it to its place of work.
- If your agent is susceptible and there is an infectious person at your location, roll a dice. If it is < 2 then your agent is now exposed.

Evening

- If your agent was at work, move it back home.
- If your agent is susceptible and there is an infectious person at your location, roll a dice. If it is < 2 then your agent is now exposed.

Demonstration- Debrief

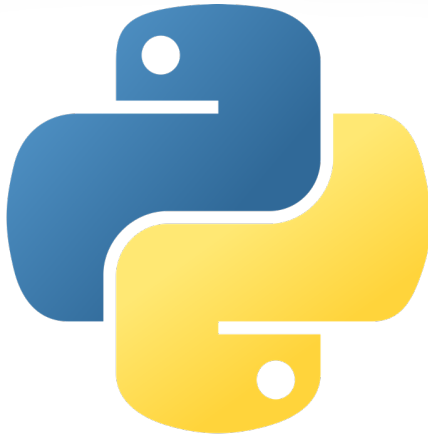
- Can you describe some of the main assumptions used by the model?
  - How realistic were they?
  - How do you think different assumptions could have affected the outcome?
- Think about the kinds of questions about patterns in disease. Discuss what you could have tracked/measured in the simulation to answer these questions.
- How could you have improved the simulation?


_user(self, user):
turns a QuerySet of connections
1 = self.filter(from_user=user)
2 = self.filter(to_user=user).s
urn set1 | set2
connected(self, user1, user2):
elf.filter(from_user=user1, to_user=
return True
lf.filter(from_user=user2, to_user=
eturn True
False
(self, user1, user2):
proper object regardless of the o
on = self.filter(from_user=user1,
onnection:
ection = self.filter(from_user=user2,
n.delete()
Top L1
(Python AC yas)---

Python 101

What is Python? And why are we using it?

- One of the most popular programming languages.
- General purpose.
- The various common applications span the scope of what we are trying to do:
  - Data science
  - Data analysis
  - Scientific programming
- Supports several programming paradigms, e.g., object-oriented, functional.



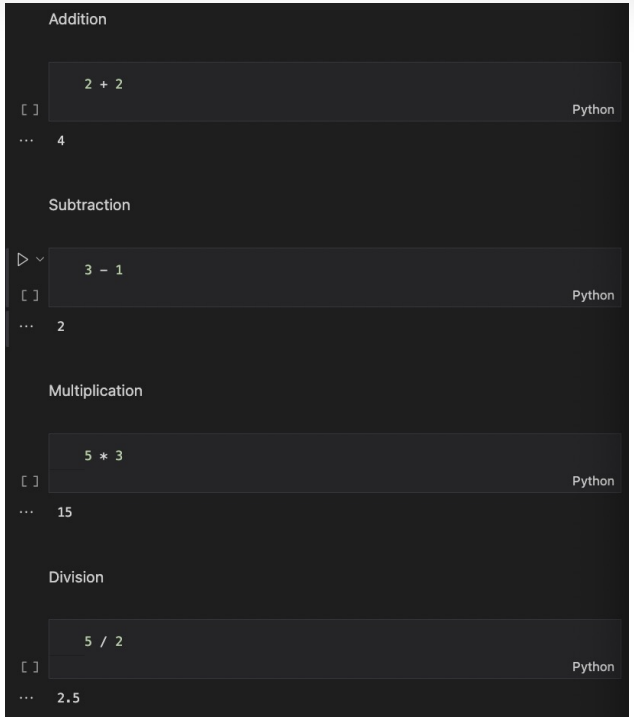
python™

Accessing Python

- Typically, Python can be installed on common operating systems (iOS, Windows, Linux etc.).
- For this masterclass, we will be using Python via the Jupyter Notebook interface on Google Collab for convenience.
- Jupyter Notebook combines Python code “chunks” with Markdown text.

Basic Arithmetic

Operator	Name	Example
+	Addition	2 + 2
–	Subtraction	3 – 1
*	Multiplication	5 * 3
/	Division	5 / 2



Variables

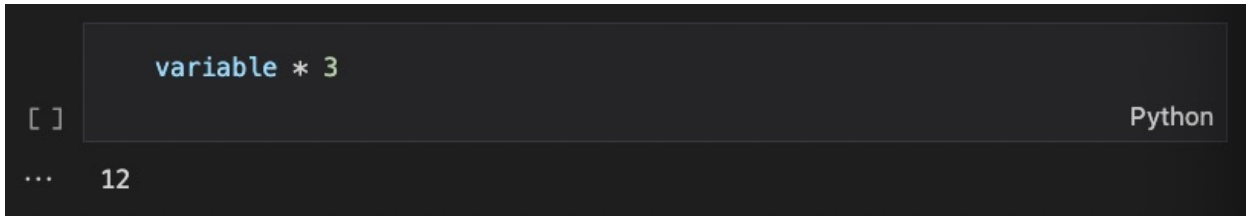
- Variables store information that may be referenced later.
- How to declare a Python variable:

```
variable = 4
```

Variable name

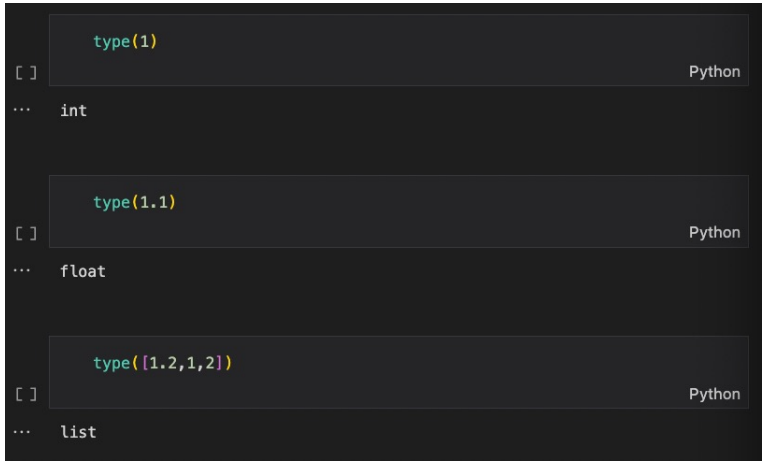
value of the variable

- If you reference the variable later, Python will run the computation replacing the name of the variable with the value associated with it.



Variable Types

Group	Name
Numeric Types:	**int**, **float**, complex
Sequence Types:	**list**, tuple, **range, array**
Mapping Type:	dict
Set Types:	set, frozenset
Boolean Type:	**bool**
Binary Types:	bytes, bytearray, memoryview
None Type:	NoneType



Boolean and Conditional Programming

- Boolean variables are either True or False.
- These can be used with conditional statements if or while.
- E.g.

```
lol = True # boolean variable

if lol:
  print("Haha")

rofl = False # boolean variable

if rofl:
  print("Hahaha")
```

[ ] Python

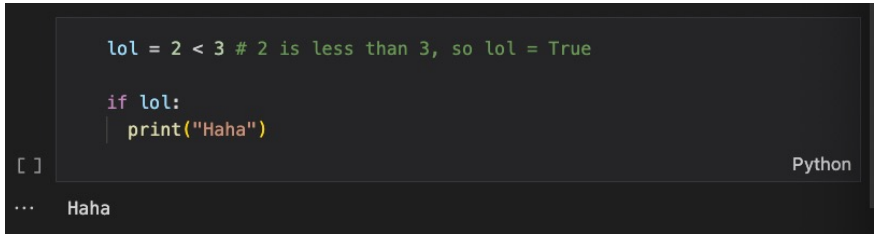
```
Haha
```

The colon that the indented block should beindicates executed if true

- Since the lol is True, “Haha” is printed, but since rofl is False, “Hahaha” is not.

Comparison Operators

Comparison operators can also be used to make Boolean variables.



Operator	Meaning
>	greater than
<	smaller than
>=	greater than or equal to
<=	smaller than or equal to
==	is equal
!=	is not equal

Lists

- Lists is a data type that contains multiple elements.
- They elements may not have the same type.
- Ways to create a list:

```
a_list = [1,2,4,8]
another_list = list()
```
[ ] Python

- Adding to a list

```
a_list = [1,2,4,8]
another_list = list()
```
[ ] Python

- Referencing an element in the list. (Lists are indexed from 0.)

```
a_list[3]
```
[ ] Python

... 8

Loops

- `for` loops iterate over the objects in a list.
- `while` loops repeat a block of code provided a specified condition is met.

Dummy variable- the for loop substitutes a value in the list in place of this variable

List that is iterated over

Functions

- Functions are sections of code that perform a specific task and can be reused.
- Gives us the option to reuse and parameterise code.
- There are a lot of inbuilt functions in Python, and ones that may be added from packages.
- We can also define our own functions using `def`
- `return` means that the function outputs.

```
def hello():
  return("hello")

hello()
```

```
'hello'
```

Name of the function

Parameters taken by the function these will only exist inside the function

```
def emoji(expression,number):
  for i in range(number):
    if expression == "happy":
      print(":)")
    else:
      print(":(")
```

Python

```
emoji("happy",2)
```

Python

```
:)
:)
```

```
emoji("sad",1)
```

Python

```
:(
```

When the function is called, it will run this code with the parameters replaced by inputs.

Classes

- We can make our variable types by defining our own classes.
- Classes provide a blueprint for an object.

Class name

Variables required when you instantiate the class

```
class Dog:

    def __init__(self, name):
        self.name = name
        self.tricks = []

    def add_trick(self, trick):
        self.tricks.append(trick)
```

[ ] Python

This function adds a trick to the list of tricks for the dog

These are characteristic variables that will be associated with any dog

To use a function associated with the class, use the syntax `class_inst.function(parameters)`

```
d = Dog('Fido') # instance of a dog
d.add_trick('roll over') # adding a trick

print(d.name,d.tricks)
```

[ ] Python

Fido ['roll over']

You can access the variables associated with the instance of the class using the syntax `class_inst.variable`

Packages

- Packages allow us to important functions and classes that others have built to extend the functionality of the language.
- Popular packages include:
  - **Numpy-** Scientific computing
  - **Pandas-** Data analysis
  - **Matplotlib-** Plotting
  - **Pytorch-** Machine learning and AI

You can give the package a short nickname

When you call an object from the library, use the syntax `package.object`

You can also just import a specific part of the package, e.g. a single function