University of Amsterdam
Communication Science

Day 1.
Introduction to Python & basic operations

Today

1. What will we do in this course?

2. The toolbox
    1. Python: introduction to a language
    2. Your new environment

3. Python basics

4. Operations in Python

5. Github

6. Teach each other!

Aims

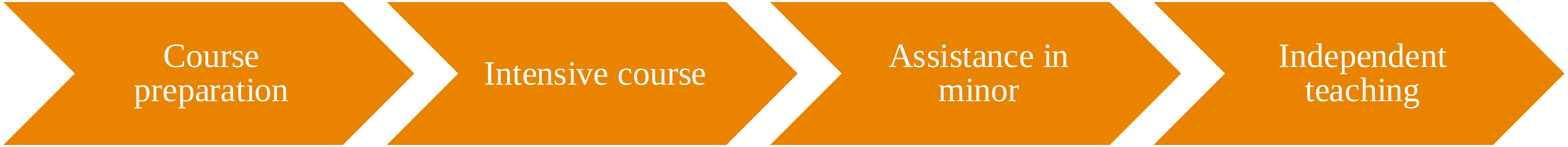
To create a larger pool of colleagues who can teach computational methods in the bachelor program (minor Communication in the Digital Society) and in the master program (assistance in Digital Analytics, Digital Journalism, Big Data)

What

Basics of Python + how to teach it

- The most common data types in programming,
- Simple control structures (loops, conditions and functions),
- Exploring and visualizing different data types,
- Handling errors in programming languages and debugging,
- Teaching methods in computational communication science.

How



Course preparation

- Book, videos, exercises
- November 2024

Intensive course

- 3 full-day sessions + homework
- 11, 18, 25 November 2024

Assistance in minor

- Teaching assistance in CCS1/CCS2/Research project
- Semester 2 (February-June 2025)

Independent teaching

- Teaching in minor
- February-June 2026

31 teaching hours*

10 teaching hours*

*Both parts are obligatory

Your new toolbox:

Introduction into Python

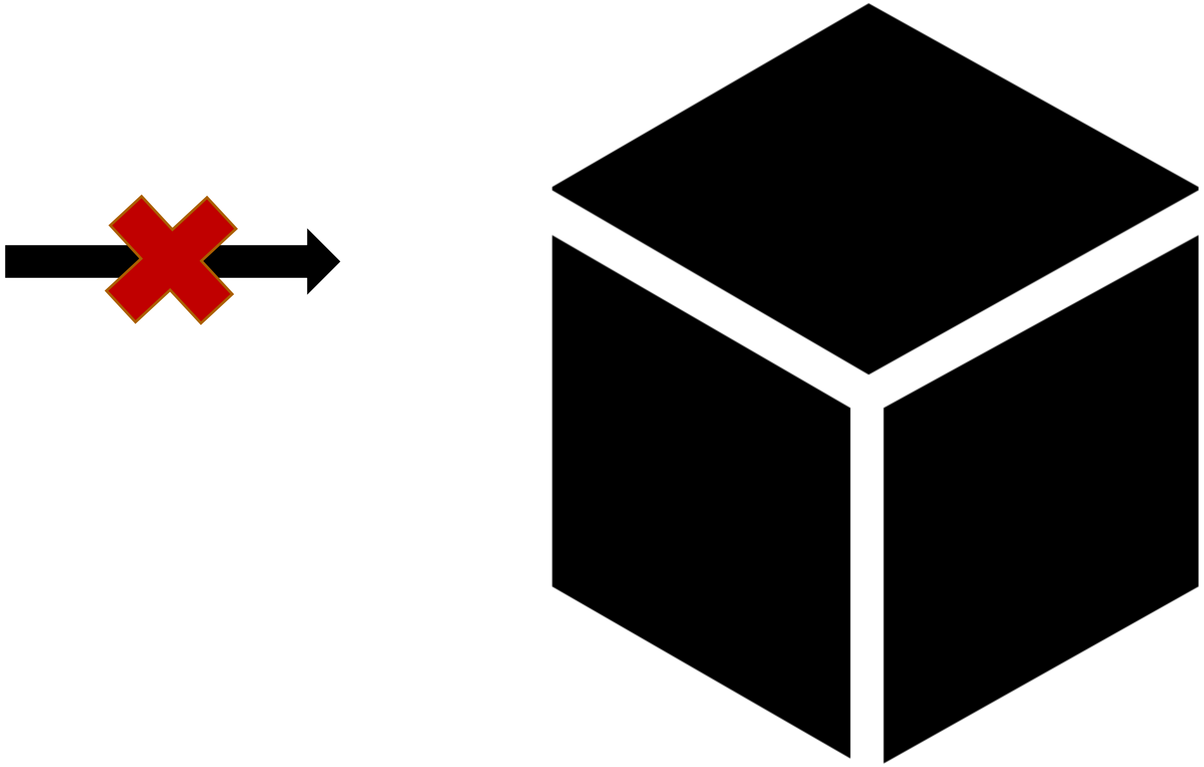
Why a programming language?

“Moreover, the tools we use can **limit the range of questions** that might be imagined, simply because they do not fit the affordances of the tool. Not many researchers themselves have the ability or access to other researchers who can build the required tools in line with any preferred enquiry. This then introduces serious limitations in terms of the scope of research that can be done.”

Vis (2013)

Advantages of programming your tools

- platform independent
- open-source – no blackbox
- free
- flexible

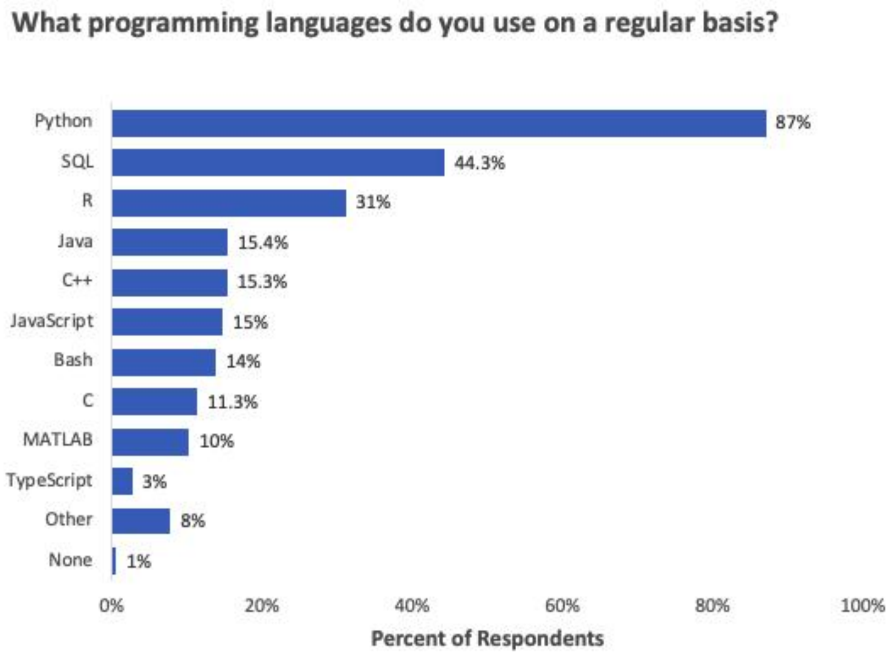


Python as a language

- language, not a program
- open source
- flexible (make you own functions)
- portable
- active community

Every data scientist has a tab open to Stack Overflow

Python and other languages



Note: Data are from the 2019 Kaggle ML and Data Science Survey. You can learn more about the study here: https://www.kaggle.com/c/kaggle-survey-2019.
A total of 19717 respondents completed the survey; the percentages in the graph are based on a total of 14762 respondents who provided an answer to this question.

BUSINESS BROADWAY
DATA SCIENCE | CUSTOMER ANALYTICS | MACHINE LEARNING

What is pip?

- Package manager for Python
- Allows you to install and manage additional packages
- You can run it
  in your notebook
  (`! pip install`)

bol.com

Jupyter – our environment

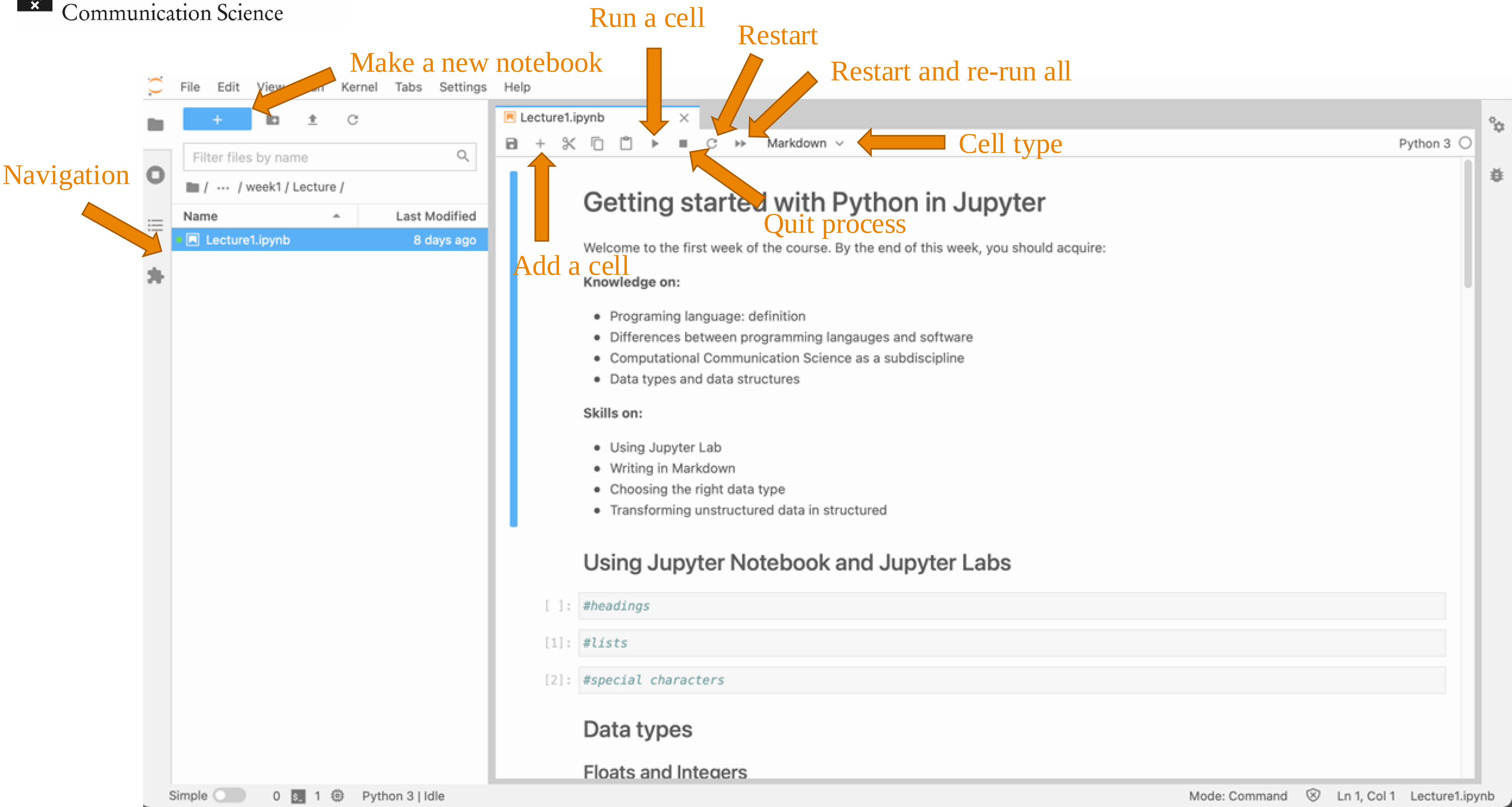
Web application in which you can make *notebooks*
that include live code and additional text

Combination of :

- Python code
- results of your code
- annotations that you can make in MarkDown

Other options

- VSCode – popular code editor, allows running Jupyter Notebooks
- Pycharm – popular among programmers, graphical interface
- Text editing programs: IDLE, EMACS…
- And many more…



Run a cell

Restart

Restart and re-run all

Make a new notebook

Cell type

Navigation

Quit process

Add a cell

Getting started with Python in Jupyter

Welcome to the first week of the course. By the end of this week, you should acquire:

Knowledge on:

- Programing language: definition
- Differences between programming langauges and software
- Computational Communication Science as a subdiscipline
- Data types and data structures

Skills on:

- Using Jupyter Lab
- Writing in Markdown
- Choosing the right data type
- Transforming unstructured data in structured

Using Jupyter Notebook and Jupyter Labs

```
[ ]: #headings
```

```
[1]: #lists
```

```
[2]: #special characters
```

Data types

Floats and Integers

Let’s start coding!

Python lingo

Basic data types	
int	37
float	1.75
bool	True, False
string	“Marijn”

Tips

“Joanna” and Joanna are not the same – what is the difference?

”5” and 5 are not the same – what is the difference?

“5” * 5 is not 25 – why? how to fix it?

Combining data

More advanced data types	
object	a = 5
	b = “5”
list	names = [”Marijn”, “Wouter”, “Edith”]
	postcodes = [1018, 1019, 1020]
dict	postdict = {“Marijn”: 1018, “Wouter”: 1019, “Edith”: 1020}

Working with lists and dicts

List	names[0]	the first entry (0)
	names[-1]	second-to-last entry
	names[:2]	first two entries (0,1)
	names[1:4]	entries 1, 2, 3
	names[1:]	entries starting with 1 until the end
	names.count(“name”)	frequency of name in the list
Dict	postdict[“Marijn”]	entry associated with key “Marijn”

We start counting with **0**

Combining data

Less common data types	
set	a collection of unique items {1,2,3}
tuple	a list that cannot be changed (1,2,3)
defaultdict	dict that returns “empty” when calling a non-existing key

…

Weekly Challenges – Pair programming (better together)

Two programmers work on code together

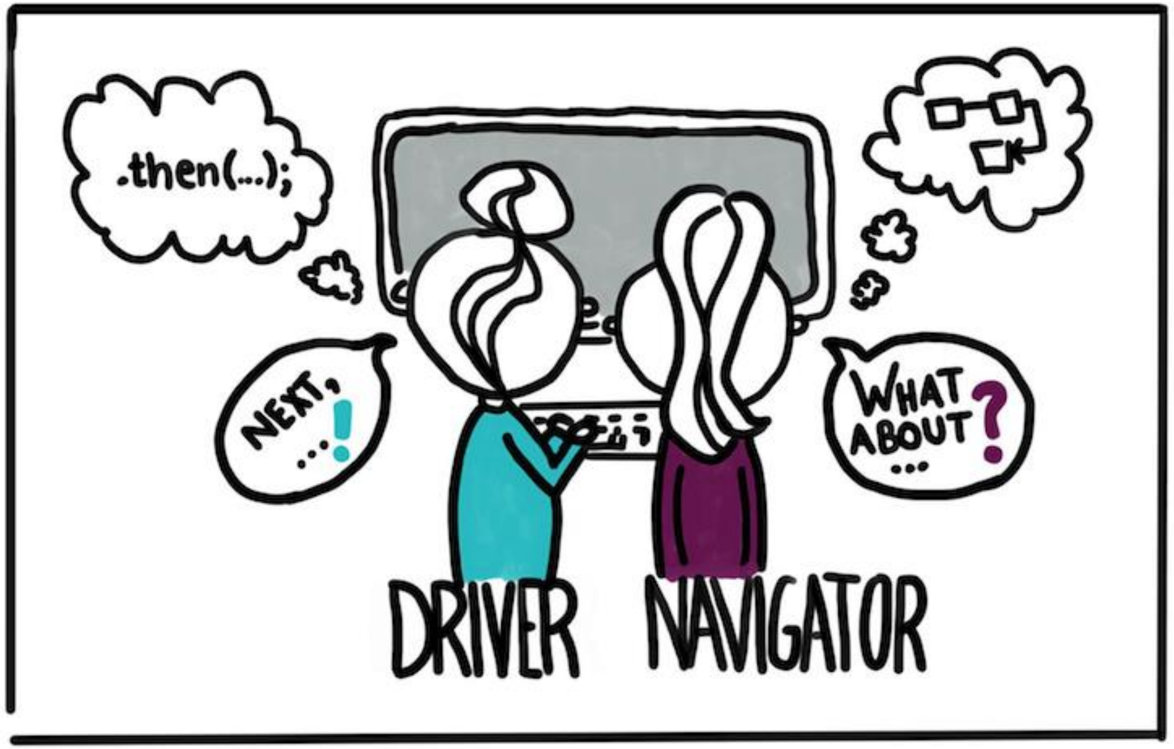
2 modes of thinking

Better solutions

Reflection

Focus

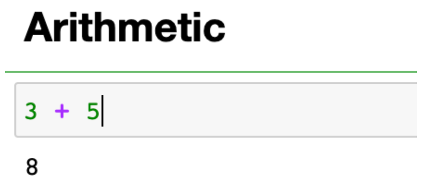
Code review on the go



Basic operators

Python operators

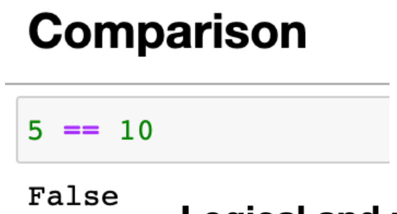
- Arithmetic (+, -, *, /, %, **, //)
- Comparison (==, !=, >, <, >=, <=)
- Logical and membership operators (and, or, not, in, not in)
- Assignment operators (=, +=)



Arithmetic

```
3 + 5
```

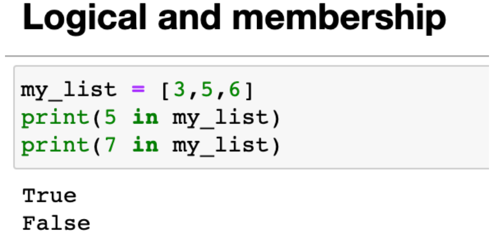
```
8
```



Comparison

```
5 == 10
```

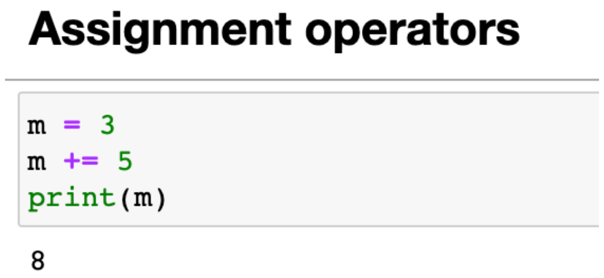
```
False
```



Logical and membership

```
my_list = [3,5,6]
print(5 in my_list)
print(7 in my_list)
```

```
True
False
```



Assignment operators

```
m = 3
m += 5
print(m)
```

```
8
```

Using conditional statements

Why?

Mandate the machine to do X or Y, depending on circumstances

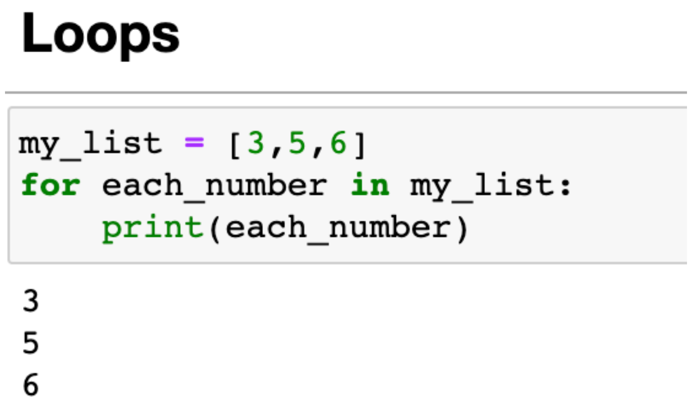
- Only print values larger than 5
- Only print something when two values match
- Generate a signal when a user subscription failed to come in on time

How?

- **if** – use to check if a condition is met
- **else** – use to specify what the machine should do if the initial condition is not met
- **elif** – use elif (else if) if there are more than two conditions that need to be checked
- **while –** use while to check if a condition still holds

Loops

Loops can be used to repeat a block of statements. They are executed once, indefinitely, or until a certain condition is reached.



Loops

```
my_list = [3,5,6]
for each_number in my_list:
    print(each_number)
```

```
3
5
6
```

There are three primary types of loops

- For – applies to all specified elements
- While – applies while a certain condition is true
- Repeat (do-while) – runs repeatedly until a certain condition is met

You will most often find yourself using for loops

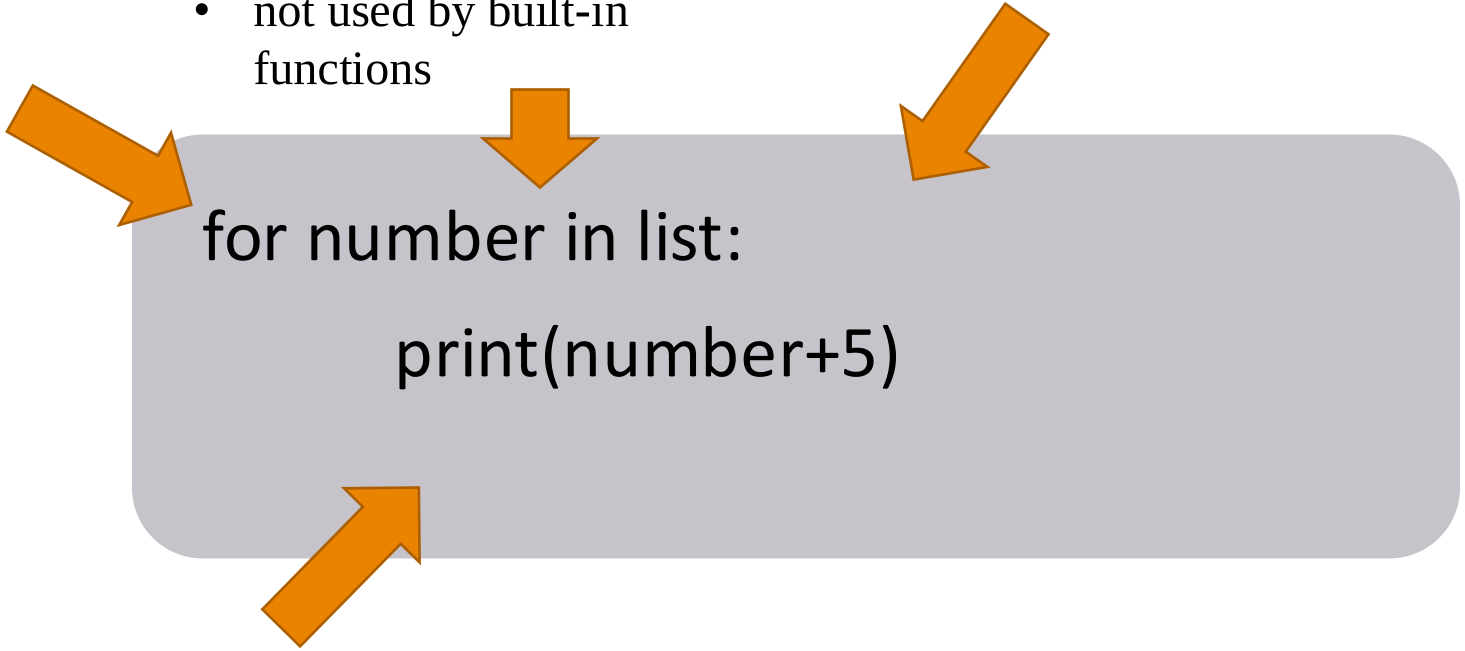
For loops dissected

For loops always start with the **for** statement which specifies which elements the for loop applies to.

An **arbitrary** designation for the list, dictionary or data frame element.

- no spaces
- not used by built-in functions

The name of your list, dictionary or dataframe. Note the **colon**, which leads into the next line.



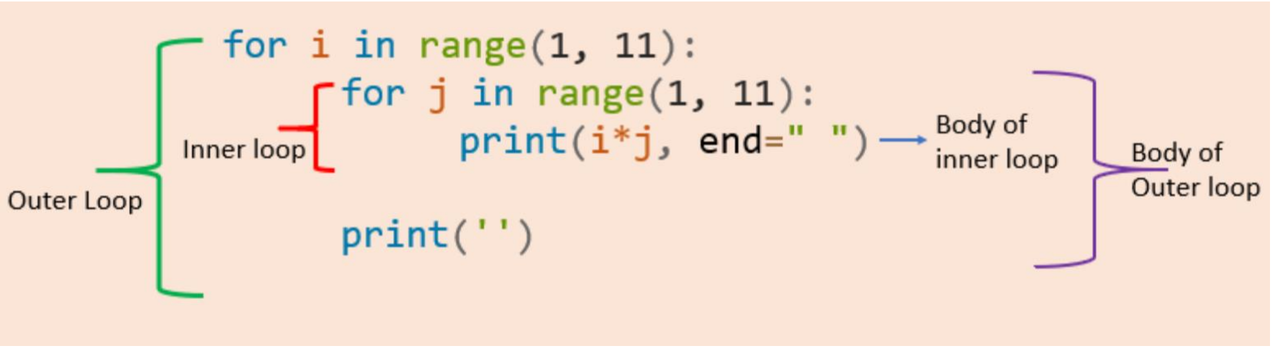
```
for number in list:
        print(number+5)
```

Specify what should happen to/with each element of the list, dictionary or dataframe. Note the **indent**!

Nested for loops

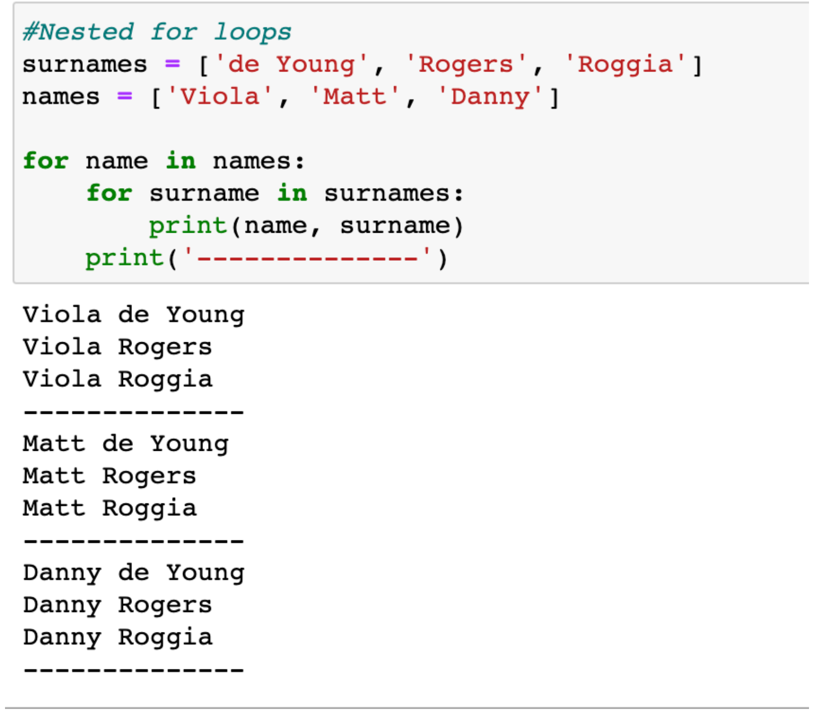
For loops can also be nested

- A loop (inner) inside the body of a loop (outer loop)
- Total iterations = no. iterations in inner loop * no. iteration in outer loop



```
for i in range(1, 11):
    for j in range(1, 11):
        print(i*j, end=" ")

    print('')
```



```
#Nested for loops
surnames = ['de Young', 'Rogers', 'Roggia']
names = ['Viola', 'Matt', 'Danny']

for name in names:
    for surname in surnames:
        print(name, surname)
    print('--------------')
```

```
Viola de Young
Viola Rogers
Viola Roggia
--------------
Matt de Young
Matt Rogers
Matt Roggia
--------------
Danny de Young
Danny Rogers
Danny Roggia
--------------
```

For loops for different data structures

The syntax for loops differs slightly based on the data structure used

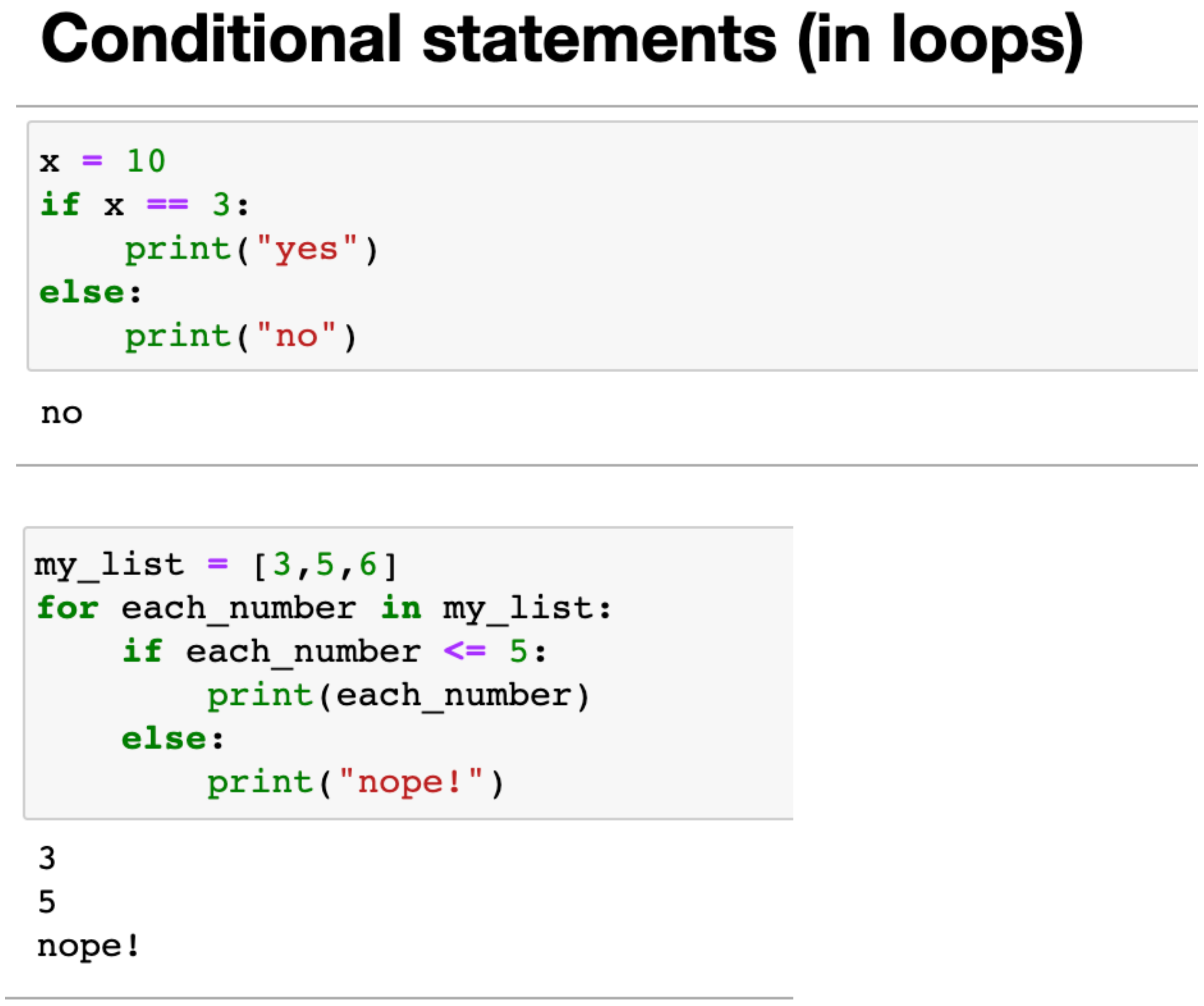
- It is the simplest for lists: “for x in list_name:”
- For dictionaries: use .items() and specify that the loop applies to both keys and values

Using conditional statements (in loops)

Why?

Mandate the machine to do X or Y, depending on circumstances

- Forms the basis of lots of software!
- Can get very complex
- Can be used in combination with for loops



Conditional statements (in loops)

```
x = 10
if x == 3:
    print("yes")
else:
    print("no")
```

```
no
```

```
my_list = [3,5,6]
for each_number in my_list:
    if each_number <= 5:
        print(each_number)
    else:
        print("nope!")
```

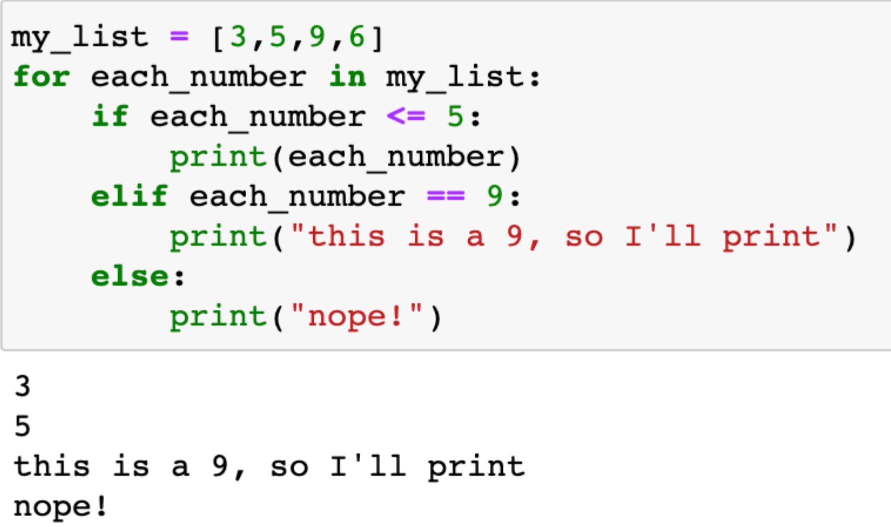
```
3
5
nope!
```

Using conditional statements (in loops)

Why?

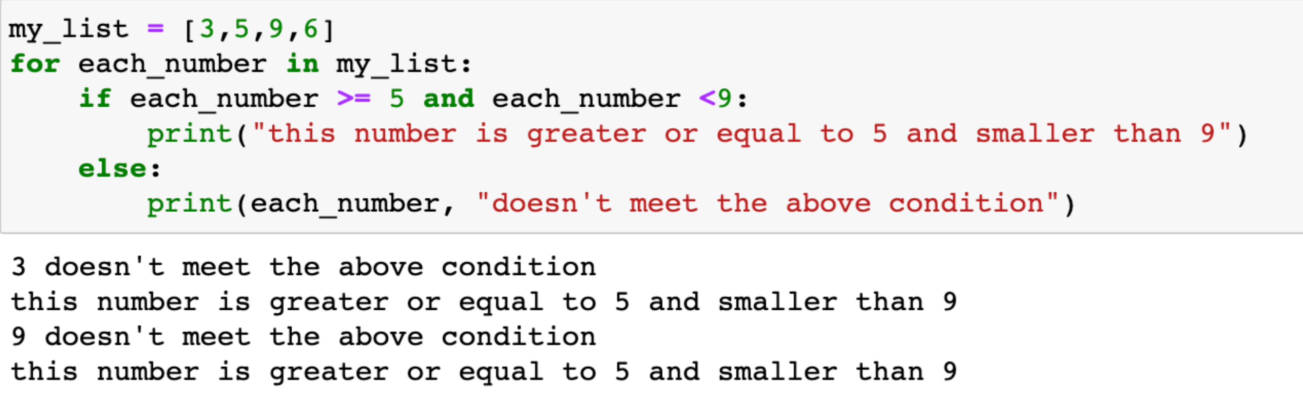
Mandate the machine to do X or Y, depending on circumstances

- Forms the basis of lots of software!
- Can get very complex
- Can be used in combination with any Boolean operator



```
my_list = [3,5,9,6]
for each_number in my_list:
    if each_number <= 5:
        print(each_number)
    elif each_number == 9:
        print("this is a 9, so I'll print")
    else:
        print("nope!")
```

```
3
5
this is a 9, so I'll print
nope!
```



```
my_list = [3,5,9,6]
for each_number in my_list:
    if each_number >= 5 and each_number <9:
        print("this number is greater or equal to 5 and smaller than 9")
    else:
        print(each_number, "doesn't meet the above condition")
```

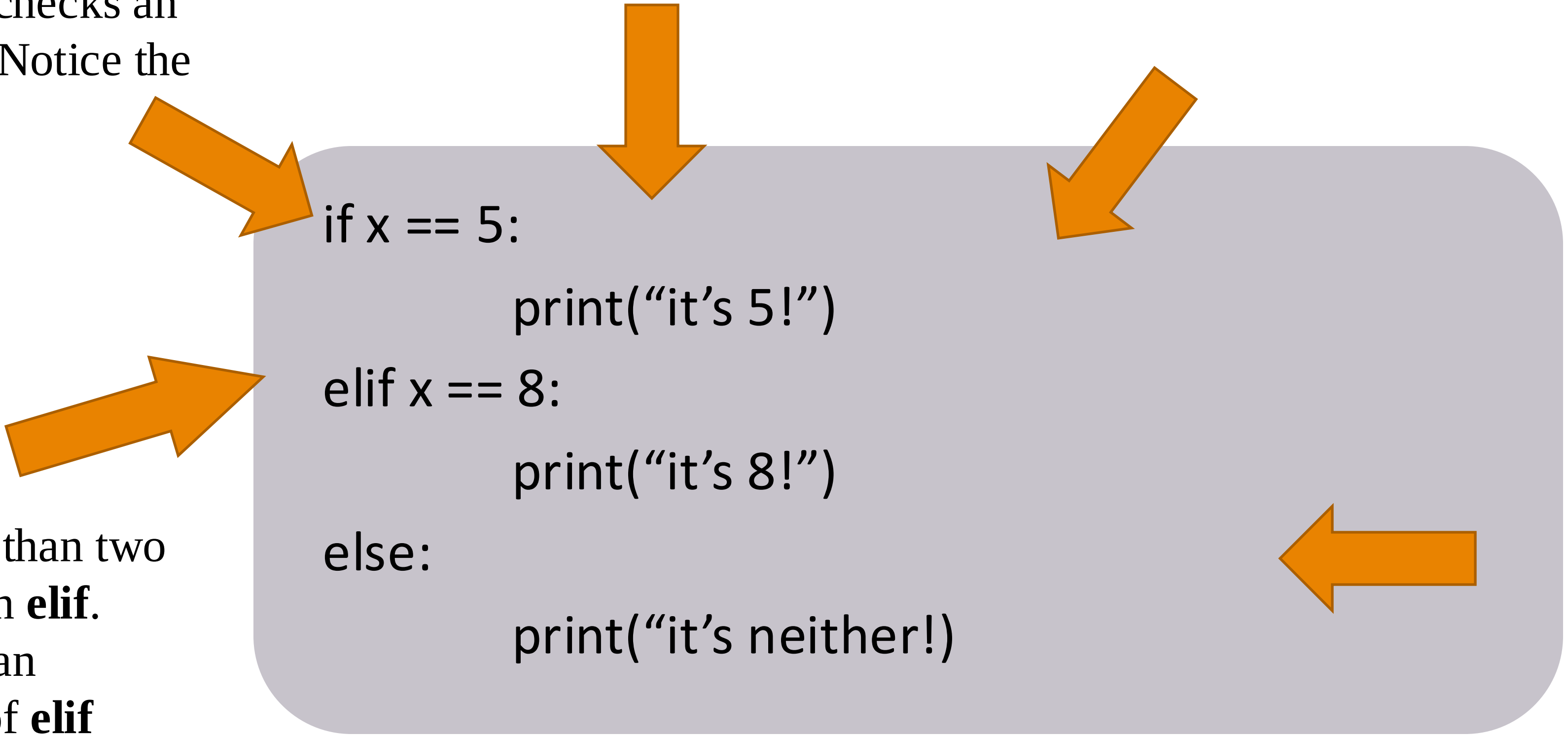
```
3 doesn't meet the above condition
this number is greater or equal to 5 and smaller than 9
9 doesn't meet the above condition
this number is greater or equal to 5 and smaller than 9
```

Conditional statements dissected

All conditional statements start with an **if** which checks an initial condition. Notice the **colon**!

The condition checked can be anything.

What should happen if the condition is met. Notice the **indent**!



```
if x == 5:
        print(“it’s 5!”)
elif x == 8:
        print(“it’s 8!”)
else:
        print(“it’s neither!)
```

Once you reach the final condition, use an **else** statement. The final condition is generally not explicitly noted. In this case, there appear to be three conditions.

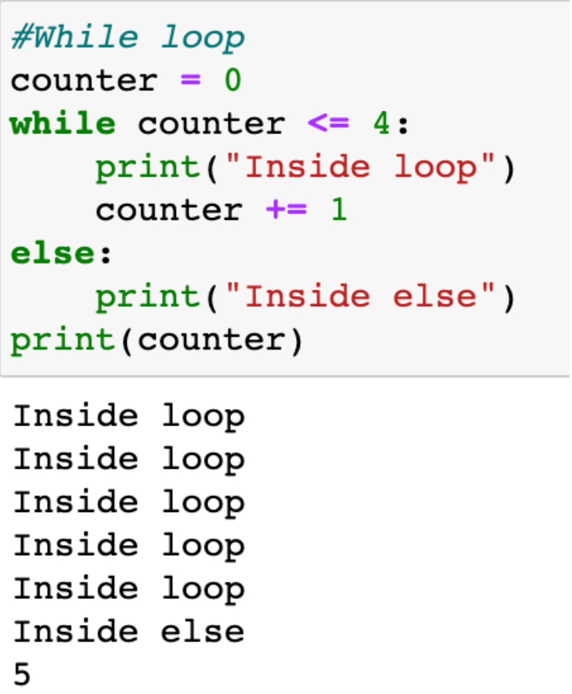
1) The number is a 5
2) The number is an 8
3) The number is neither

If there are more than two conditions, use an **elif**. You can include an infinite number of **elif** statements. The syntax is the same as for the **if** statement.

While loops

Loops can also execute while a certain condition is true

- A counter is something that you will frequently use as a reference variable
- Counters are commonly applied in while loops
- While loops are frequently used in combination with conditional statements



```
#While loop
counter = 0
while counter <= 4:
    print("Inside loop")
    counter += 1
else:
    print("Inside else")
print(counter)
```

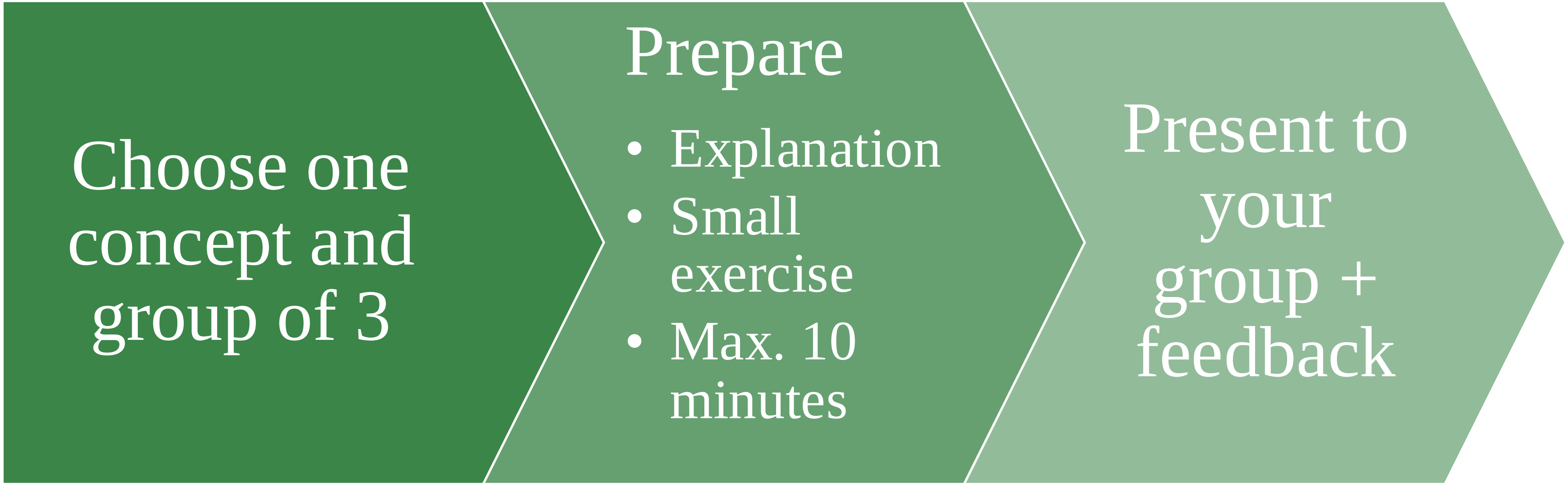
```
Inside loop
Inside loop
Inside loop
Inside loop
Inside loop
Inside else
5
```

How to teach this?

But how to teach it?

- **Live Coding**: demonstrate concepts by writing code in real-time. This shows students how to approach problem-solving and debugging naturally.
- **Relevant Data Sets**: work with data sets that relate to students' interests and study topic (e.g., movies, social media etc.) to make learning more engaging.
- **Encourage Pseudocode**: before diving into code, teach students to outline their logic in plain language. Pseudocode helps them think logically without getting bogged down in syntax.
- **Pair Programming**: let students work in pairs or small groups on certain exercises. This encourages collaboration, helps them learn from each other, and mimics real-world coding practices.
- **Code Reviews**: Ask students to review each other's code to learn from alternative approaches, catch mistakes, and build confidence in their ability to critique code constructively.

Try yourself



Choose one concept and group of 3

Prepare

- Explanation
- Small exercise
- Max. 10 minutes

Present to your group + feedback

15 minutes

Concepts

- What are different data types in Python?
- How can you organize data in Python?
- What are lists/dictionaries and how do you use them?
- What are loops?
- How to use conditions?

Next steps

- Review exercises, finish if necessary
- New readings + small preparatory exercises shared tomorrow
- Questions? → let us know ☺