

LECTURE 2.2

Hello, Python

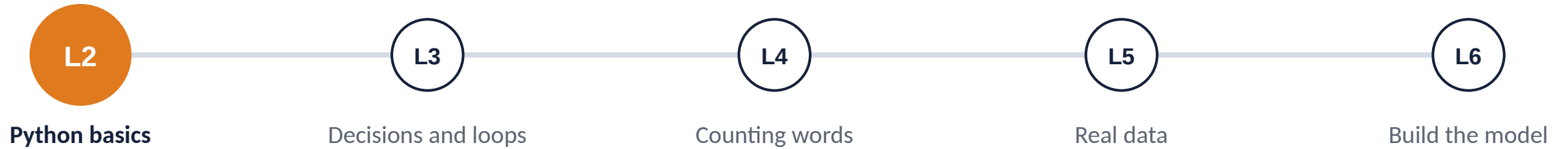
Your first lines of code: numbers, variables and text

Today's key line of code

```
print("Hello, Data Science!")
```

Where we are: Lecture 2, part 2 of 3

Part 2 of 3



Today you will

- 1 Show results with `print()`
- 2 Use Python as a calculator
- 3 Store values in variables
- 4 Work with text, the raw material of spam
- 5 Read error messages without panic

Why Python?

1

Reads like English

`print("hello")` does exactly what it says. Beginners get results on day one.

2

Free, everywhere

No license to buy. Runs offline in Jupyter, or online in Colab.

3

Built for data

Tools like pandas and scikit-learn, which we use to build the spam detector, are Python libraries.

print() shows a result

- `print()` displays whatever is inside the brackets
- Text goes inside quotes: `"Hello"`
- Numbers do not need quotes
- Each `print()` starts a new line

Code

```
print("Hello, Data Science!")  
print(2 + 3)  
print("2 + 3")
```

Output

```
Hello, Data Science!  
5  
2 + 3
```

Python as a calculator

- `+` `-` `*` `/` work as you expect
- `**` means power: `7 ** 2` is 7 squared
- `/` always gives a decimal: `7 / 2` is `3.5`
- `#` starts a comment: a note Python ignores

Code

```
# Python as a calculator
print(7 + 3)
print(7 * 3)
print(7 / 2)
print(7 ** 2)
```

Output

```
10
21
3.5
49
```

Variables store values

- `=` means "store this", not "is equal to"
- The name goes on the left, the value on the right
- Use clear names: `spam_found`, not `x`
- Names use lowercase and underscores, no spaces

Code

```
messages_today = 12
spam_found = 3
share = spam_found / messages_today
print(share)
```

Output

```
0.25
```

Watch it run: variables changing

WATCH FOR

- Each variable appears in Memory when it is first stored
- $x = x + 1$ uses the old value
- share does not update by itself
- The highlighted line is the next one to run

WATCH THE CODE RUN, STEP BY STEP

Variables change as lines run

= means store. Watch each variable appear and change in Memory.
The highlighted line is the NEXT line Python will run.

Click the video to play, click again to pause. Silent: talk the class through it.

Every value has a type

- **int**: whole numbers like **12**
- **float**: decimals like **0.25**
- **str** (string): text like **"free prize"**
- **bool**: **True** or **False**, answers to yes/no questions

Code

```
print(type(12))  
print(type(0.25))  
print(type("free prize"))  
print(type(True))
```

Output

```
<class 'int'>  
<class 'float'>  
<class 'str'>  
<class 'bool'>
```

Text is data too

- `len()` counts characters, spaces included
- `.lower()` makes all letters small, so FREE and free match
- `in` asks: does this text appear inside?
- This is the first building block of a spam filter

Code

```
message = "Claim your FREE prize now"
print(len(message))
print(message.lower())
print("free" in message)
print("free" in message.lower())
```

Output

```
25
claim your free prize now
False
True
```

f-strings mix text and values

- Put **f** before the quotes
- Anything inside **{ }** is calculated and inserted
- **{share:.0%}** shows a number as a percentage
- Perfect for clear, readable reports

Code

```
spam_found = 3
total = 12
print(f"{spam_found} of {total} messages were spam")
share = spam_found / total
print(f"That is {share:.0%} of the inbox")
```

Output

```
3 of 12 messages were spam
That is 25% of the inbox
```

Your turn

5 minutes

1

Store your first name in a variable called `name`

2

Store any text message in `message`

3

Print a sentence like: Nadia's message has 42 characters

4

Print whether the message contains "offer" (ignore capitals)

Code

```
name = "..."  
message = "..."  
print(f"...")  
print(...)
```

Starter code: replace the ... with your own code

A first look at errors (more in Lecture 2.3)

Read the last line of an error first. It names the problem.

What you typed	What Python says	What it means
<pre>total = 12 print(totl)</pre>	<pre>NameError: name 'totl' is not defined</pre>	Typo in a name. Python only knows names you created. Check spelling.
<pre>print("hello)</pre>	<pre>SyntaxError: unterminated string literal (detected at line 1)</pre>	A quote or bracket was opened but not closed. Close it.
<pre>print("Total: " + 12)</pre>	<pre>TypeError: can only concatenate str (not "int") to str</pre>	Cannot glue text and a number with +. Use an f-string.

Interactive lab: Variables Lab

In the lab you can

1 Memory boxes

Run code line by line and watch each box change.

2 Operators

Change a and b: every result and type updates live.

3 Text tools

Type a message: indexes, len, in and .lower() update.

4 f-strings

Move sliders and format numbers with :.2f and :.0%.

The screenshot shows the 'Variables Lab' interface. At the top, it says 'LECTURE 2.2 · INTERACTIVE LAB' and 'Variables Lab'. Below this, it says 'Watch values move into named boxes, test every operator, and play with text'. There are four tabs: '1 Memory boxes', '2 Operators', '3 Text tools', and '4 f-strings'. The 'Memory boxes' tab is selected. The main area is titled 'Memory boxes: run code one line at a time'. It explains that each variable is a labelled box in memory and that users should step through the code to watch which box changes, what it now holds, and what type of value it is. There is a code editor with the following code:

```
1 messages_today = 12
2 spam_found = 3
3 share = spam_found / messages_today
4 print(share)
5 spam_found = spam_found + 1
6 print(spam_found, share)
7 share = spam_found / messages_today
8 print(share)
```

Below the code editor are three buttons: 'Run next line', 'Run all', and 'Start again'. A message below the buttons says 'Press Run next line. The orange line is the one Python will run next.' To the right of the code editor are two panels: 'MEMORY' and 'OUTPUT'. The 'MEMORY' panel says 'Memory is empty. Boxes appear when a line stores a value.' The 'OUTPUT' panel is empty.

Open the lab ▶

Works offline in any browser
Labs_open_in_browser/
L2.2_Variables_Lab.html

Today's toolkit

<code>print(x)</code>	Show a value
<code># note</code>	Comment, ignored by Python
<code>name = value</code>	Store a value in a variable
<code>+ - * / **</code>	Arithmetic, <code>**</code> is power
<code>type(x)</code>	Which kind of value is this?

<code>len(text)</code>	Number of characters
<code>text.lower()</code>	All letters to lowercase
<code>"a" in text</code>	Does the text contain "a"? True/False
<code>f"{x} items"</code>	Insert values into text

Homework 2.2

Due before Lecture 2.3 • [date]

1 Set up

Create a notebook (Jupyter or Colab) named HW2_2_<your ID>. Put your name in a text cell at the top.

2 About me

Store your name, department and favourite number in variables. Print one f-string sentence using all three.

3 Predict, then run

For each line, write your prediction as a comment first, then run it: `print(10 / 4)`, `print(10 // 4)`, `print("ab" * 3)`, `print(len("data science"))`.

4 Message inspector

Store a real SMS you received (remove names and numbers). Print its length, its lowercase version, and whether it contains "free".

5 Break it on purpose

Create one error deliberately. Keep the error message and explain in one sentence what it means.

Submit on [LMS / Google Classroom]: your notebook file HW2_2_<your ID>.ipynb, or a Colab link set to "Anyone with the link can view".

Recap: Lecture 2.2 in 90 seconds

RECAP

Lecture 2.2 in 90 seconds

A short recap of today's lecture. Pause any time.

Click to play or pause. Watch it again at home to revise.