

Python for Data Science and Machine Learning

School Year 2023-2024

IST

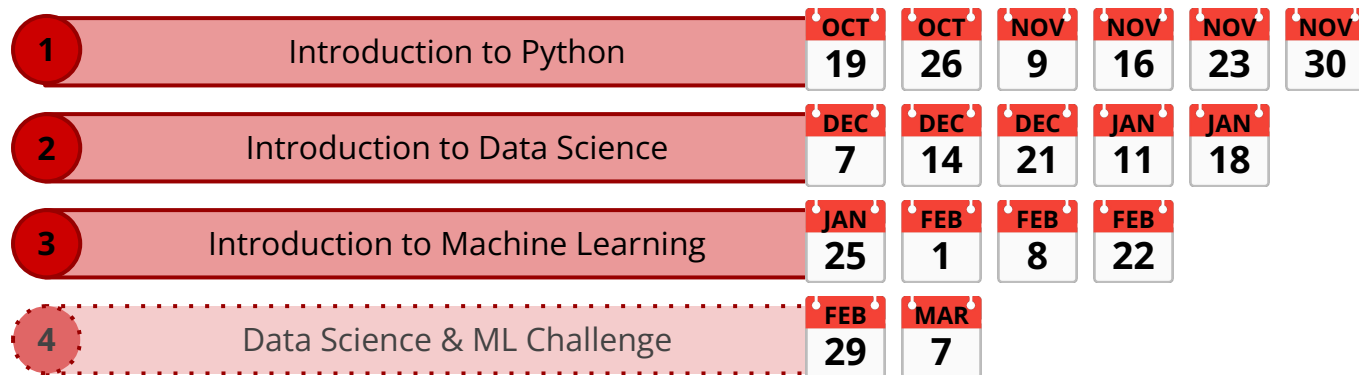
Introductions

- Alberto Spina
 - (2010) IST Alumni
 - (2019) MEng Computing - Imperial College London
 - (2023/current) Software Engineer - London

Introductions - Icebreaker

- What is your name?
- What grade are you in?
- What operating system do you use at home?
- Have you programmed before?

Course Structure



 = Core Topics  = Optional Topics

Course Objectives

- Learn to code using the **Python programming language**.
- Learn to use **Jupyter Notebooks** to write programs.
- Learn to use **Data Science** libraries to analyse datasets.
- Discover **Machine Learning** fundamentals, applying them to real-world challenges.

What is Coding?

Coding is the process of writing and creating **instructions in a programming language** to instruct a computer to **perform specific tasks** or functions.

What is Data Science?

Data science is the practice of **extracting insights** and knowledge **from data** using statistical methods and data analysis. It involves **data collection, cleaning, and analysis** to inform decision-making.

What is Machine Learning?

Machine learning is a branch of **artificial intelligence** that focuses on **developing algorithms and models** that enable **computers to learn and make predictions** or decisions from data without being explicitly programmed.

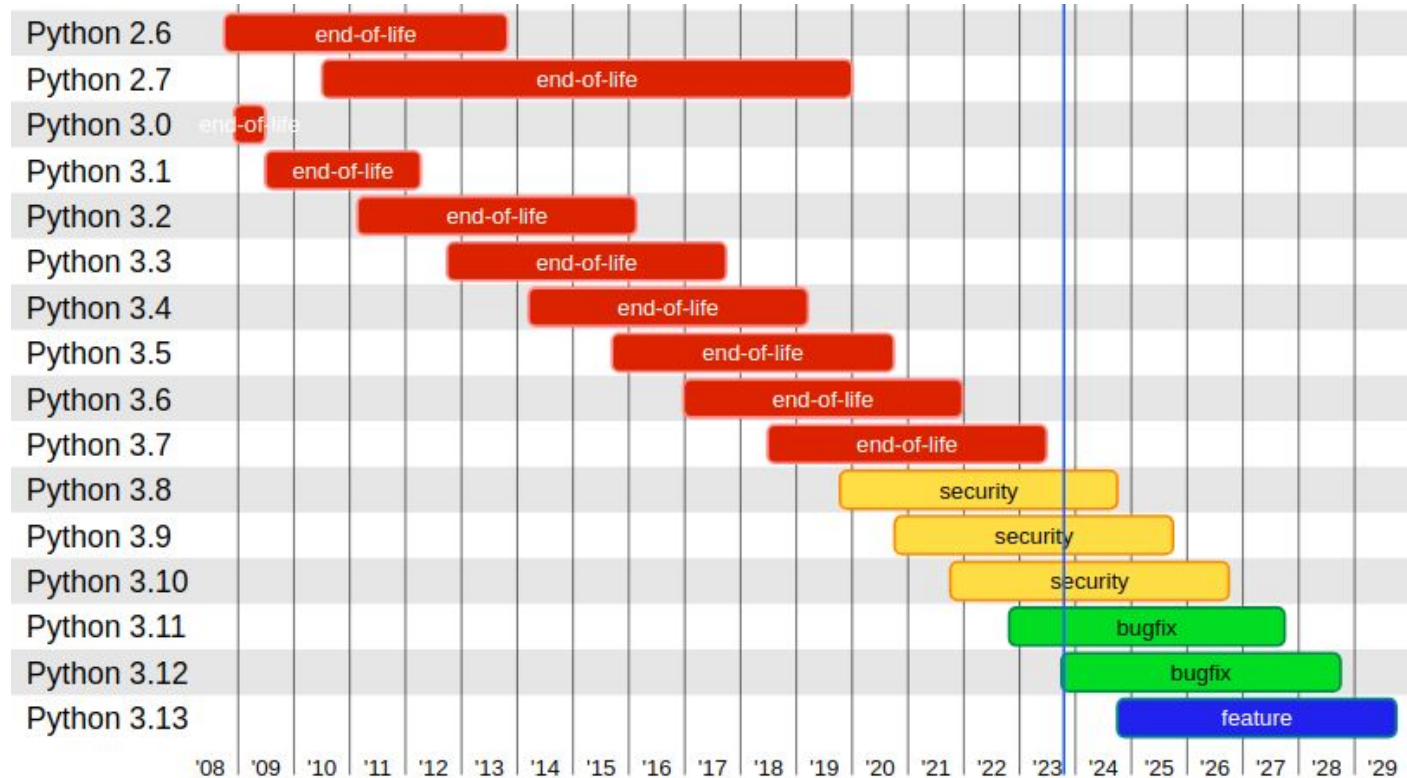
Programming Languages

- Python
- C
- C++
- Java
- PHP
- Javascript
- MATLAB
- Assembly
- ... many, many more!

Brief History of Python

- Python was created by Guido van Rossum in the late 1980s.
- It was first released in 1991 as Python 0.9.0.
- Python 3.12 just released this month.

Python versions



Why use Python?

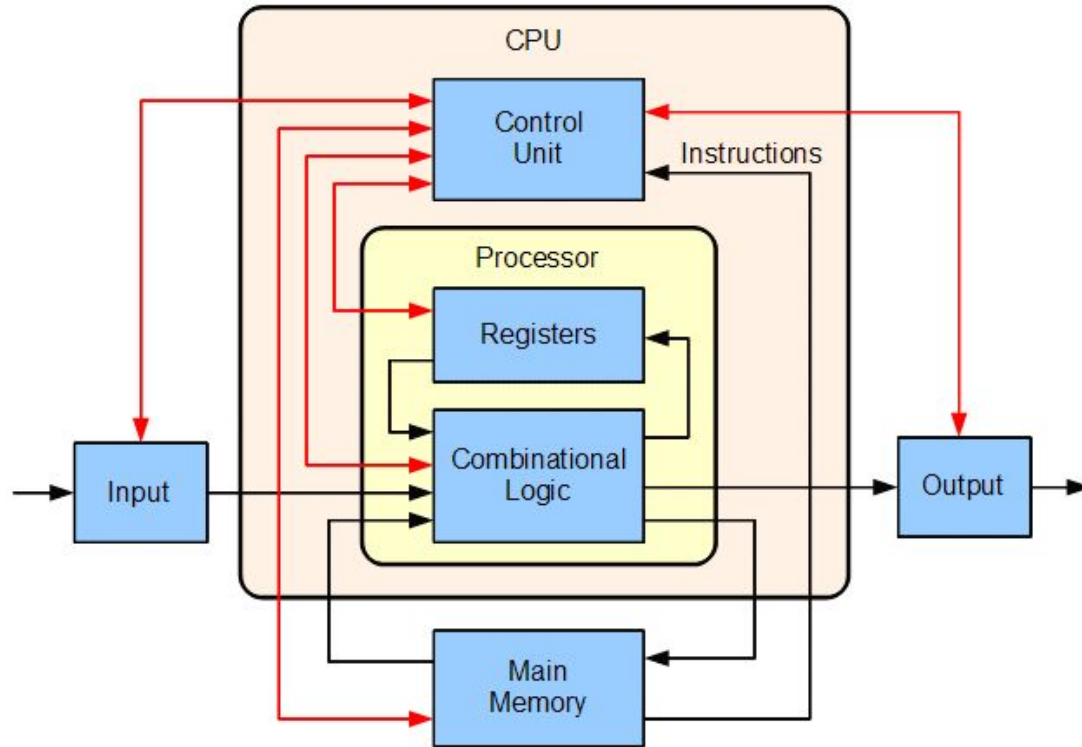
- **Simplicity**: Easy-to-read syntax.
- **Versatility**: Suitable for data analysis, and more.
- **Rich ecosystem**: Vast third party libraries and frameworks.
- **Community support**: Large and active community ensures access to resources and solutions.

What is a Program?

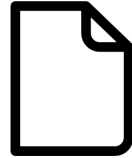
A program is a **set of instructions** or a sequence of code **written in a programming language** to tell a computer how to perform a specific task or solve a particular problem.

These **instructions** are designed to be **executed by the** computer's central processing unit (**CPU**).

Inside a CPU



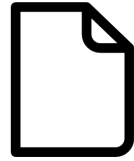
Machine Instructions (ARM Assembly)



program1.s

```
mov    r0, #42
mov    r1, #10
mov    r2, #5
add    r0, r0, r1
add    r0, r0, r2
```

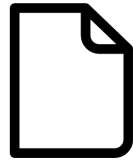
Machine Instructions (ARM Assembly)



program2.s

```
mov    r0, #5
mov    r1, #6
mov    r2, #7
cmp    r1, r2
movlt  r1, r2
cmp    r0, r1
movlt  r0, r1
```


Anatomy of a Python Program



program1.py

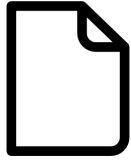
```
x = 42
```

```
y = 10
```

```
z = 5
```

```
x + y + z
```

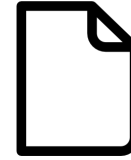
Python VS Assembly



program1.py

```
x = 42  
y = 10  
z = 5  
x + y + z
```

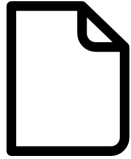
VS



program1.s

```
mov    r0, #42  
mov    r1, #10  
mov    r2, #5  
add    r0, r0, r1  
add    r0, r0, r2
```

Python VS Assembly



program3.py

```
print("Hello World!")
```

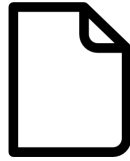
VS



program3.s

```
.LC0:
    .ascii "Hello, World!\000"
main:
    push    {r3, lr}
    movw    r0, #:lower16:.LC0
    movt     r0, #:upper16:.LC0
    bl      puts
    movs     r0, #0
    pop      {r3, pc}
```

Executing a Python Program



file.py

```
python file.py
```

```
$ python file.py  
Hello World!
```

Jupyter Notebook Setup

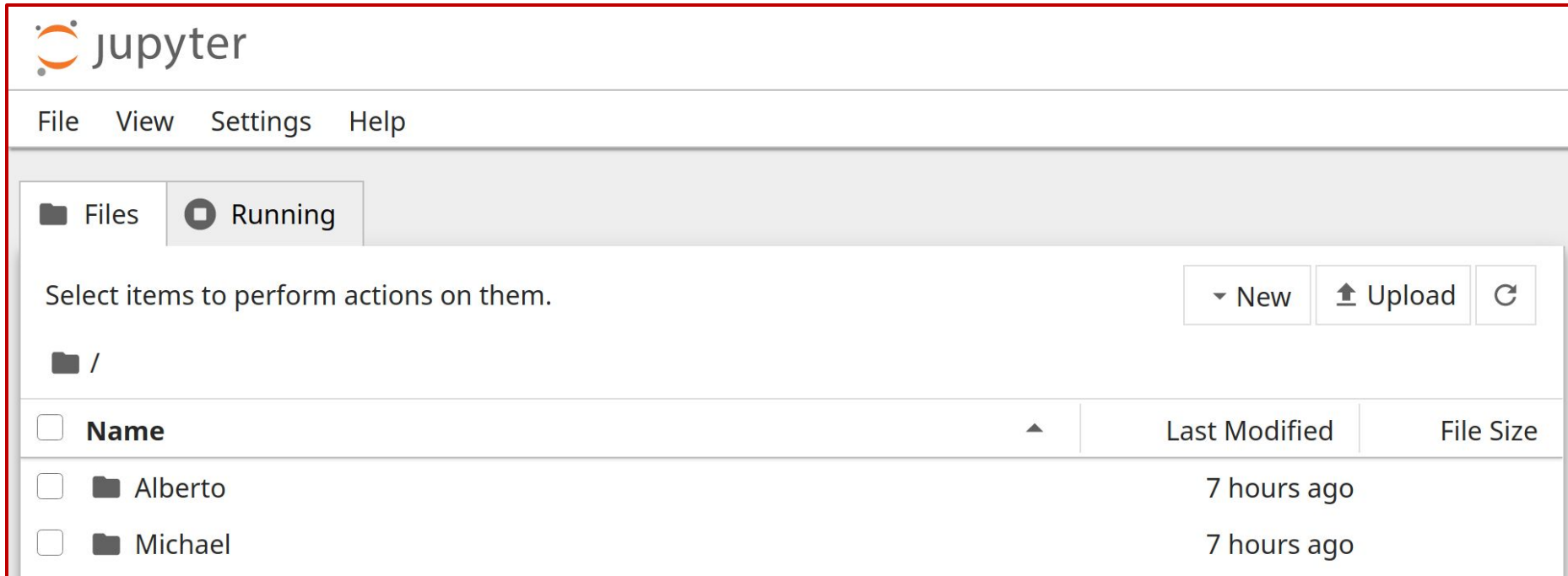


In a browser:

192.168.10.4:8888

Password: **ist**

Jupyter Notebook Setup



The screenshot displays the Jupyter Notebook web interface. At the top, the Jupyter logo is visible. Below it is a navigation bar with links for File, View, Settings, and Help. The main area is divided into two tabs: 'Files' and 'Running'. The 'Running' tab is active, showing a message 'Select items to perform actions on them.' and three buttons: 'New', 'Upload', and a refresh icon. Below this, a file browser shows a directory structure with a root folder '/' and two subfolders, 'Alberto' and 'Michael', both listed as '7 hours ago'.

jupyter

File View Settings Help

Files Running

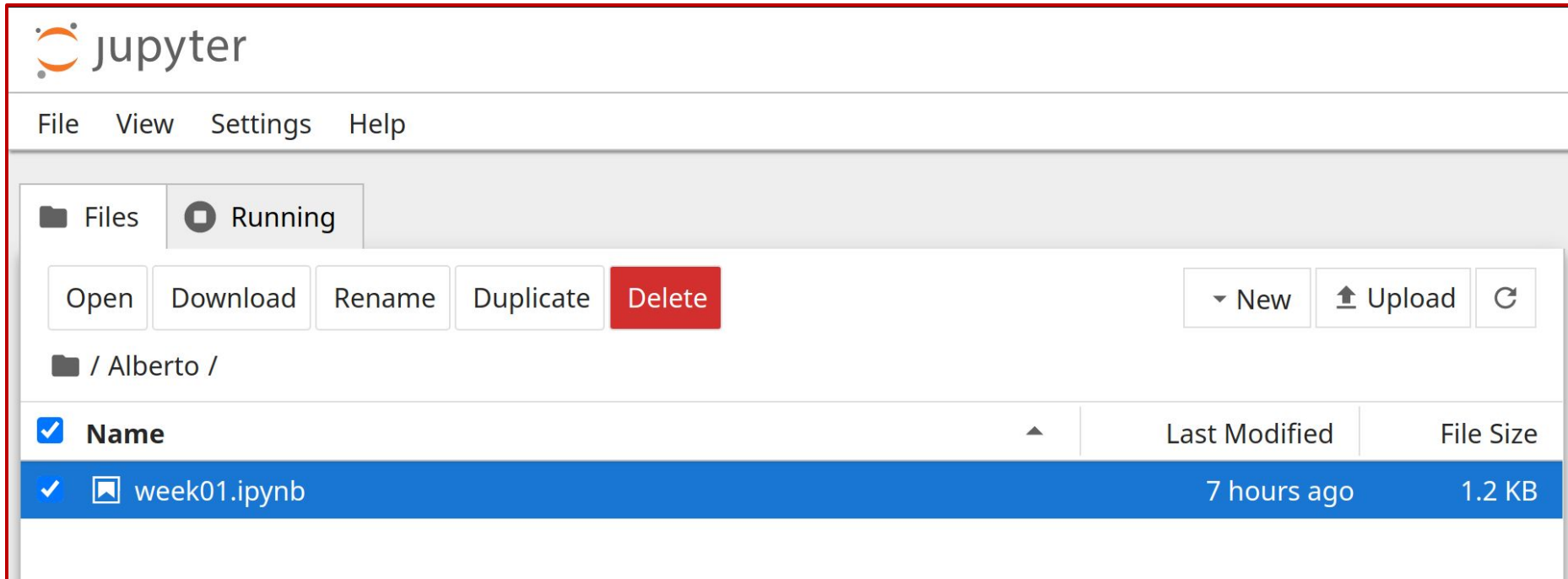
Select items to perform actions on them.

▼ New ⬆ Upload ↻


Ⓛ /

<input type="checkbox"/> Name	Last Modified	File Size
<input type="checkbox"/> Ⓛ Alberto	7 hours ago	
<input type="checkbox"/> Ⓛ Michael	7 hours ago	

Jupyter Notebook Setup



The screenshot displays the Jupyter Notebook web interface. At the top, the Jupyter logo is visible. Below it is a navigation bar with links for File, View, Settings, and Help. The main area is divided into two tabs: 'Files' (selected) and 'Running'. Under the 'Files' tab, there are buttons for Open, Download, Rename, Duplicate, and Delete. To the right of these buttons are buttons for New, Upload, and a refresh icon. Below the buttons, the current directory is shown as '/ Alberto /'. A table lists the files in the directory:

<input checked="" type="checkbox"/> Name	Last Modified	File Size
<input checked="" type="checkbox"/>  week01.ipynb	7 hours ago	1.2 KB


Jupyter Notebook Structure



Code Cell / Editor

Jupyter Notebook Structure

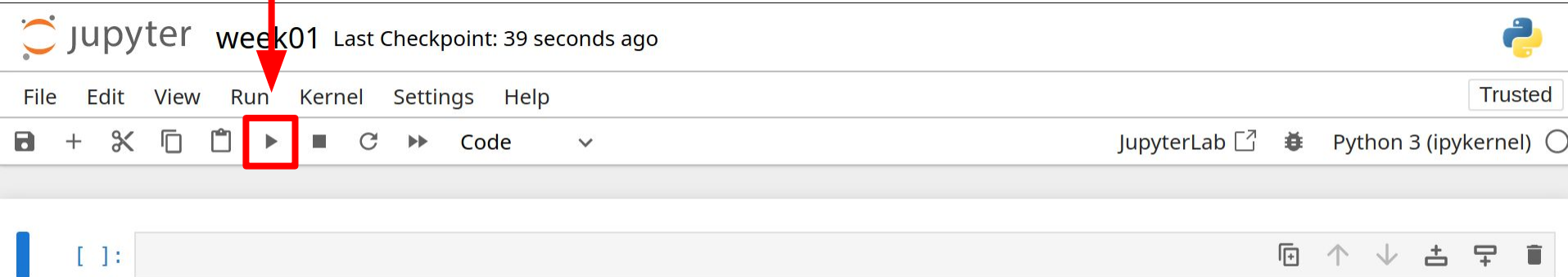
Add Cell



The screenshot displays the JupyterLab user interface. At the top, the text 'Jupyter week01' is followed by 'Last Checkpoint: 39 seconds ago'. On the right, there is a Python logo and a 'Trusted' badge. Below this is a menu bar with 'File', 'Edit', 'View', 'Run', 'Kernel', 'Settings', and 'Help'. A toolbar contains various icons, including a red-outlined square with a '+' sign, which is highlighted by a red arrow pointing down from the 'Add Cell' text above. Other icons in the toolbar include a scissors icon, a copy icon, a paste icon, a play icon, a square icon, a refresh icon, and a double play icon. To the right of the toolbar, it says 'JupyterLab' with an external link icon, a bug icon, and 'Python 3 (ipykernel)' with a circular icon. At the bottom, there is a code editor area with a blue vertical bar on the left and a text input field containing '[]:'. To the right of the input field are icons for copy, up, down, add, and delete.

Jupyter Notebook Structure

Run Cell



Variables

A variable is a named container that stores data or values.

```
x = 42  
y = "Hello"
```

Variable declarations must contain a variable name followed by an equals sign (=).

```
variable = "I am a variable"  
also_valid_variable_name = "I am also a variable"
```

Output

The **print** function can be used to display variables and values

```
print("Hello World!")  
print(123)
```

```
x = 42  
print(x)
```

```
y = "Hello"  
print(y)
```

Data Types

Python has 4 primitive data types:

int

42

1200

1_200

-3

float

3.14

0.00001

-2.1

str

"Hello"

"A"

"I am a full sentence!"

bool

True

False

The `type` function

You can use the **type** function to get the type of a variable

```
print(type(True))
```

```
print(type(42))
```

```
print(type("Hello"))
```

```
print(type(3.14))
```

Notebook TIP!

Jupyter Notebooks will automatically print the return value of the final line in a Notebook cell.

```
[12]: print(type(123))
```

```
<class 'int'>
```

```
[13]: type(123)
```

```
[13]: int
```

```
[14]: x = 1234
```

```
y = 4567
```

```
x
```

```
y
```

```
[14]: 4567
```

Changing the value of variables

You can mutate the value you assign to a variable

```
x = 42
print(x)

x = 200
print(x)

x = "Hello"
print(x)
```


Arithmetic Operations

You can perform arithmetic with variables

```
x = 9
y = 3
print(x + y)
print(x - y)
print(x * y)
print(x / y)
```

What is the output type of the division operation?

Order of Operations

What is the output of the following expression, and why?

```
x = 6 + 9 / 3 * 10  
print(x)
```

Order of Operations

What is the output of the following expression, and why?

```
x = 6 + 9 / 3 * 10  
print(x)
```

You can change the order of operations with parentheses:

```
x = (6 + 9) / 3 * 10  
print(x)
```

New Operators

What operation is the **%** operator performing?

```
x = 6 % 3
```

```
print(x)
```

```
y = 12 % 5
```

```
print(y)
```

New Operators

What operation is the **%** operator performing?

```
x = 6 % 3
```

```
print(x)
```

```
y = 12 % 5
```

```
print(y)
```

It's the remainder operator (also called **modulo**).

How does the modulo operator behave with floats?

Arithmetic with Strings

What happens if you try to perform arithmetic with strings?

```
x = "Hello"  
y = 3  
print(x * y)
```

```
x = "Hello"  
y = "World"  
print(x * y)
```

```
x = "Hello"  
y = 3  
print(x + y)
```

```
x = "Hello"  
y = "World"  
print(x + y)
```

Type Casting

You can convert from one type to another

```
x = "123"  
y = int(x)  
print(y)  
print(type(y))
```

```
x = "23.88"  
y = float(x)  
print(y)  
print(type(y))
```

Comparisons

- 5 is larger than 3

$$5 > 3$$

- -5 is larger than 9

$$-5 > 9$$

- 2 is the same as 2

$$2 == 2$$

- 2 is less than 6

$$2 < 6$$

Chaining Comparisons

- not (negation)

```
not True
```

```
not (5 < 3)
```

- and (both must be true)

```
(5 < 6) and (5 < 10)
```

- or (either must be true)

```
(5 < 3) or (5 < 10)
```

End of Class

See you all next week!