IMPERIAL

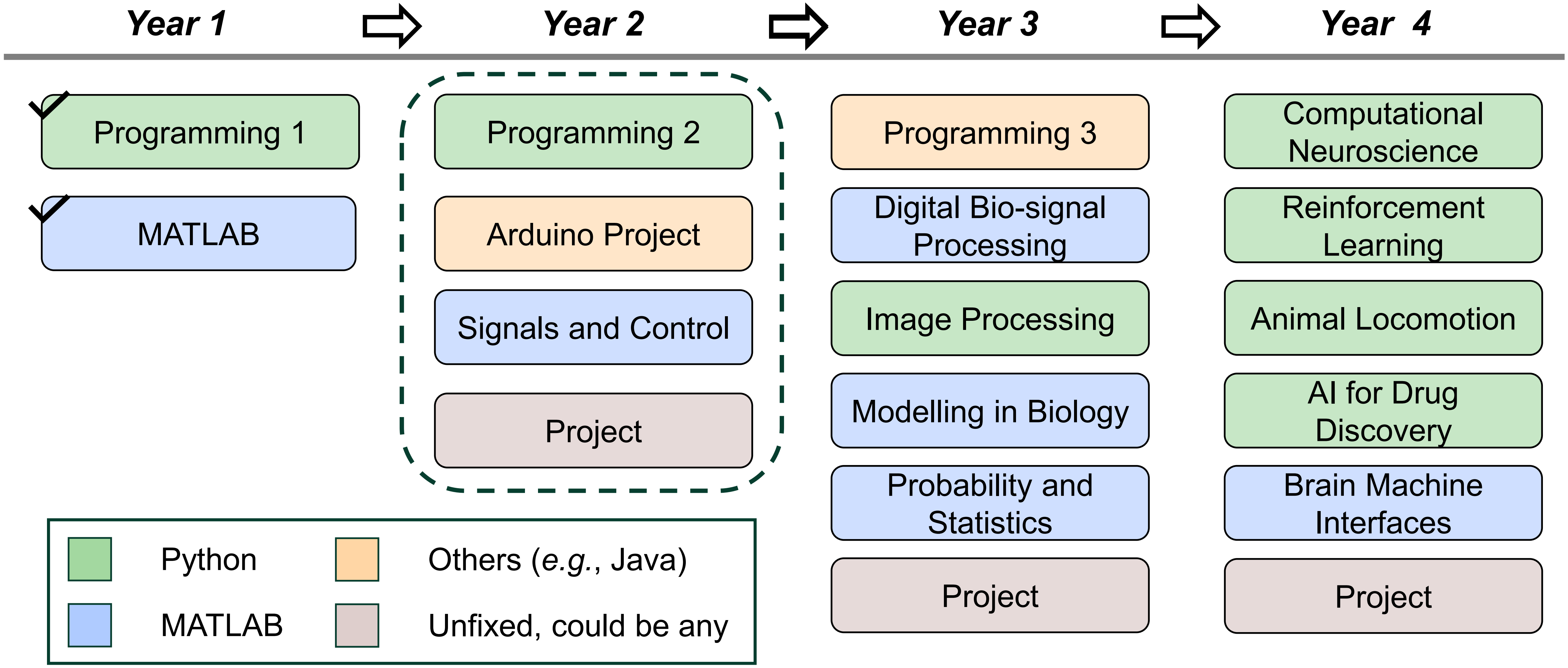
BIOE50010 – Programming 2

Computer Lab 1: Revision of Programming 1

Binghuan Li, Maria Portela, Gauthier Boeshertz, Samuel George-White, Yilin Sun, Kamrul Hasan, Wenhao Ding, Siyu Mu, Lito Chatzidavari

5 October, 2025

An Indictive Timeline



Year 1	Year 2	Year 3	Year 4
✓ 🟩 Programming 1	🟩 Programming 2	🟧 Programming 3	🟩 Computational Neuroscience
✓ 🟦 MATLAB	🟧 Arduino Project	🟦 Digital Bio-signal Processing	🟩 Reinforcement Learning
	🟦 Signals and Control	🟩 Image Processing	🟩 Animal Locomotion
	🟫 Project	🟦 Modelling in Biology	🟩 AI for Drug Discovery
		🟦 Probability and Statistics	🟦 Brain Machine Interfaces
		🟫 Project	🟫 Project

🟩	Python	🟧	Others (*e.g.*, Java)
🟦	MATLAB	🟫	Unfixed, could be any

* Information retrieved from the Module Descriptor 2025-26. Indictive only.

The Programming 2 Course

Learning Outcomes*

- **Write**, **debug**, **compile**, and **run** programs using Python;
- Use **data structures** appropriately;
- Know how to create and use **algorithms**;
- Explain and apply concepts of **object-oriented programming**.

Envision: *“Coding as fluently as writing an email.”*

Assessment Modes*

- 1 timed assignment (50%) + 1 live programming test (50%).
- To be soon communicated by the module leader directly to you.

* Information retrieved from the Module Descriptor 2025-26. Indictive only.

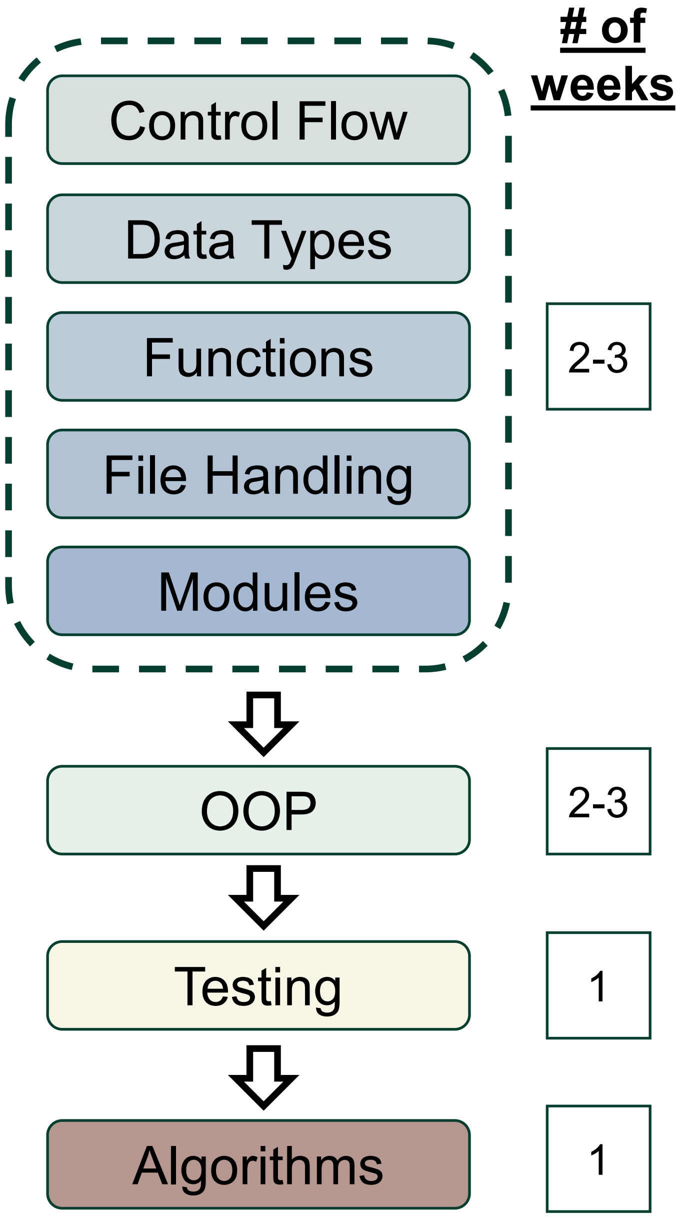
Rough Structure & Rationale

Lectures (2 hours × 9 weeks*)

- Introduction to general coding concepts with **definitions** and **live coding examples**.
- **Aim:** to enhance your understanding of core concepts and techniques.

Labs (2 hours × 10 weeks*)

- **Exercises** to apply the concepts from lectures delivered in self-learning and peer learning fashion.
- **Aim**: apply coding concepts in a practical setting.



# of weeks

Control Flow
Data Types
Functions
File Handling
Modules
2-3

OOP
2-3

Testing
1

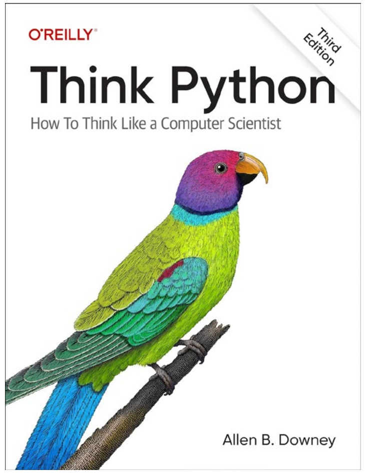
Algorithms
1

* As per module timetable on CELCAT Calendar. Indictive only.

Resources & References

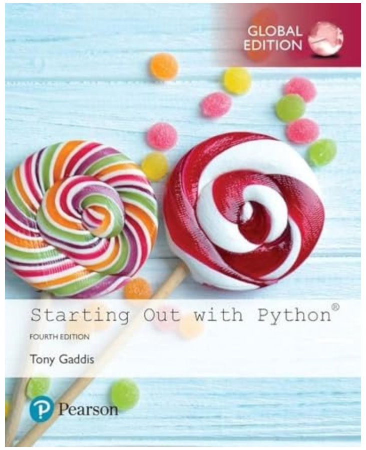
Where to Seek Help?

- **Module leader:** Dr James Choi `<j.choi@imperial.ac.uk>`
- **GTAs:** Questions will be actively monitored on **Ed Discussion** ed discussion
  - General programming advices (within or beyond this course) are welcomed.
- Python 3.13.5 documentation and Python's built-in **`help()`** function
- Textbooks, online resources, **weekly example notebook**.



Think Python 2e,
by A. Downey

An 'official' textbook - rigorous and comprehensive, yet as informative as a dictionary, allowed for use during exams.



Starting Out with Python,
by T. Gaddis

An 'unofficial' textbook - friendly for Python beginners with intuitive explanations, though sometimes shallow for advanced coders.

Will It Be Tough?

Me internally: "… probably he/she only wrote 'Hello, World!' once." 🙄

> ***"There are only two kinds of languages: the ones people complain about and the ones nobody uses."***
>
> Bjarne Stroustrup, creator of C++

> ***"The only way to learn a new programming language is by writing programs in it."***
>
> Dennis Ritchie, creator of C

> ***"Talk is cheap. Show me the code."***
>
> Linus Torvalds, creator of Linux & Git

How To Nail Programming 2?

- **Syntax, syntax, syntax**
- Don’t rush – but please keep up! Save your work, log your progress, make the most of your time.
- *“Why doesn’t my code work?”* isn’t a helpful question. Be specific, so others can help you debug.
- Use **Stack Overflow / generative AI models** wisely – the goal is learning, not just finishing first. Be responsible for your work!
- Working code is the best code.

✓ **Correct syntax**

```
for i in range (0,10):
␣␣␣␣print(i)
```

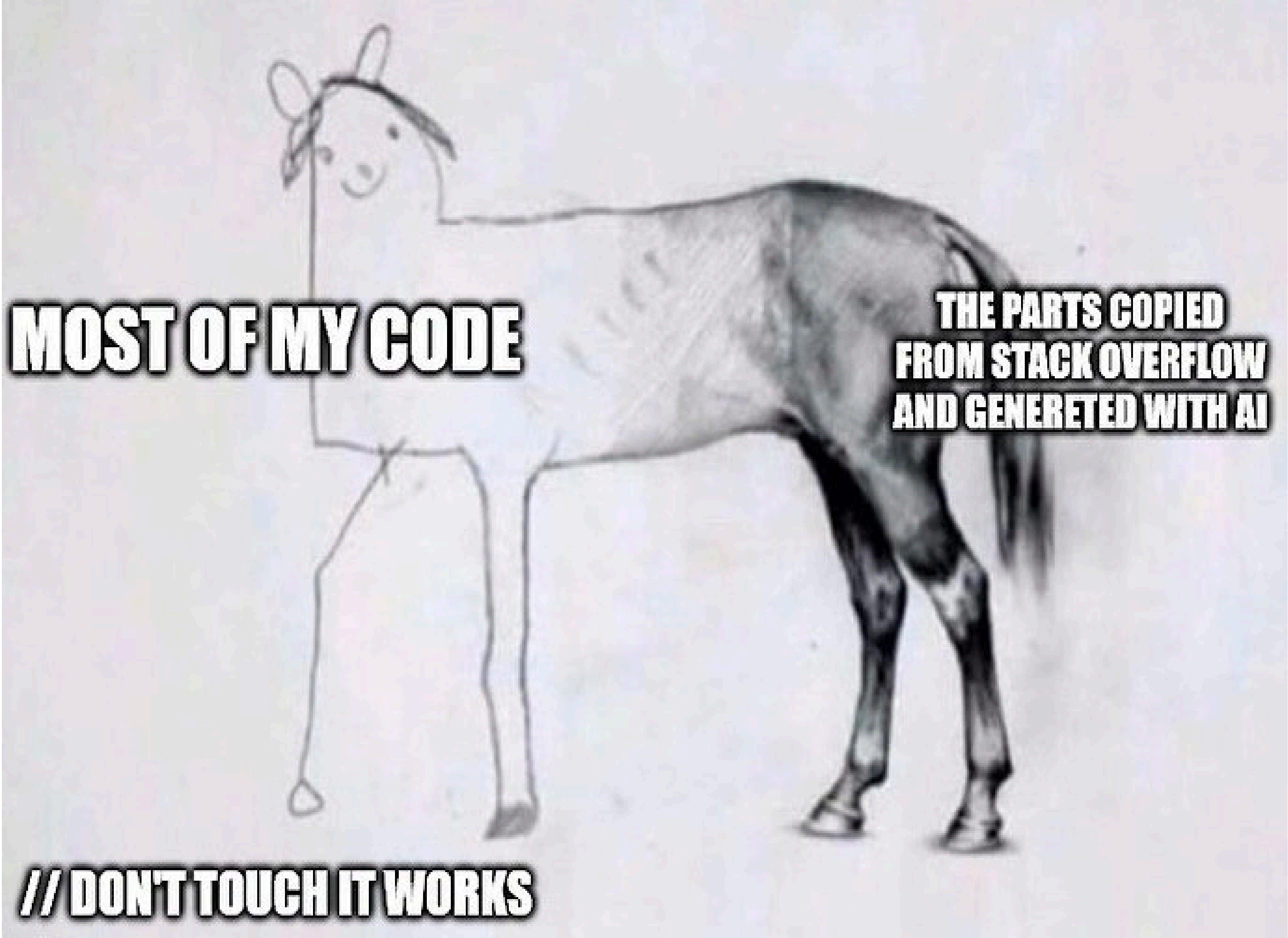
✕ **Erroneous syntax: indentation**

- not indented

```
for i in range (0,10):
print(i)
```

- Inconsistent indentation

```
for i in range (0,10):
␣␣␣␣print(“i equals to”)
␣␣␣␣␣␣␣␣print(i)
```

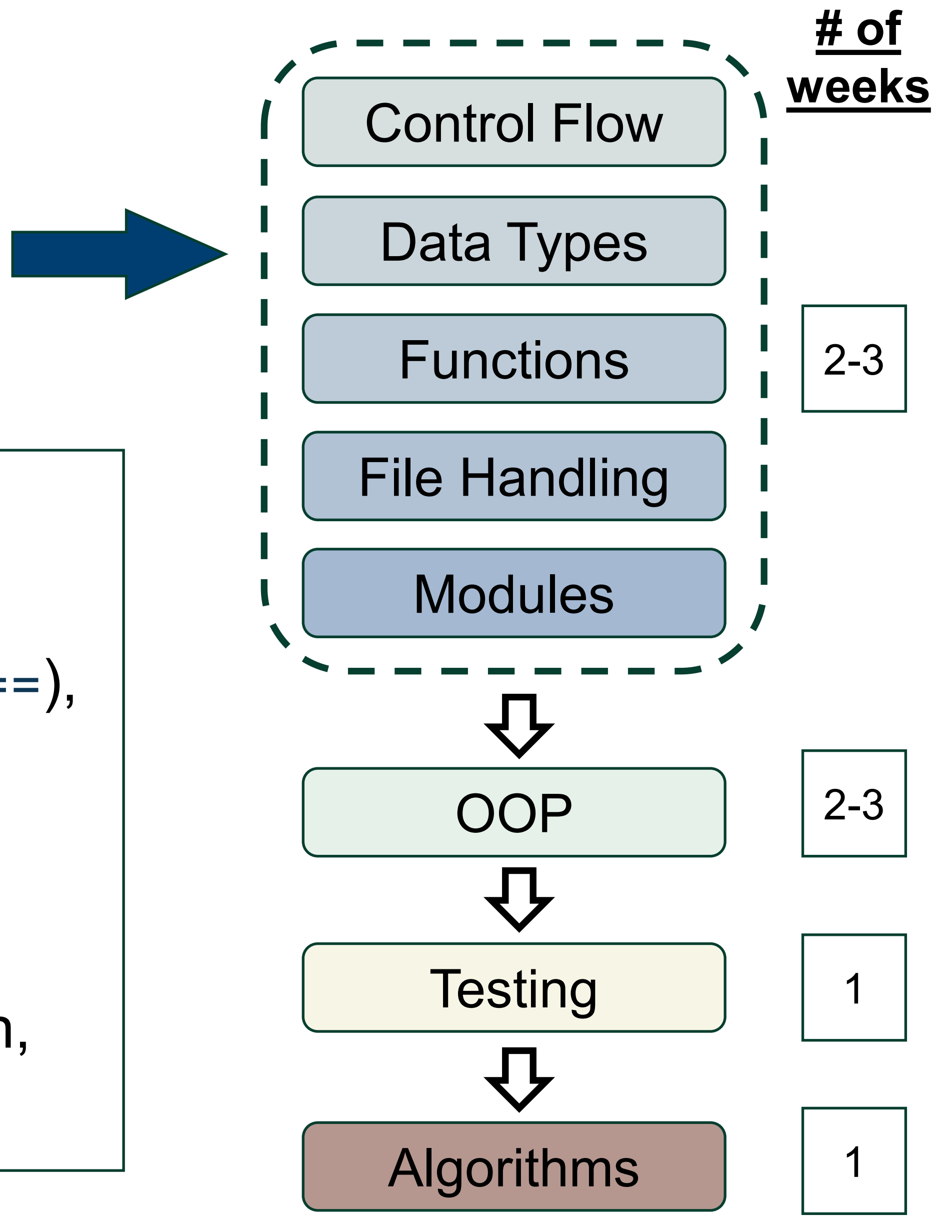

MOST OF MY CODE
THE PARTS COPIED FROM STACK OVERFLOW AND GENERETED WITH AI
// DON'T TOUCH IT WORKS

Progress Check

Week 1:
we are here

Revision Points (from Programming 1)

- **Data types:** `int`, `str`, `list`, `dict`, …
- **Operations:** arithmetic (`+`, `-`), comparison (`==`), logical (`True`, `False`)
- **Control flows:** `if…elif…else` condition, `while` condition, `for` loop
- **Functions and scopes:** definition a function, pass and return data to/from function



# of weeks

Control Flow
Data Types
Functions
File Handling
Modules
2-3

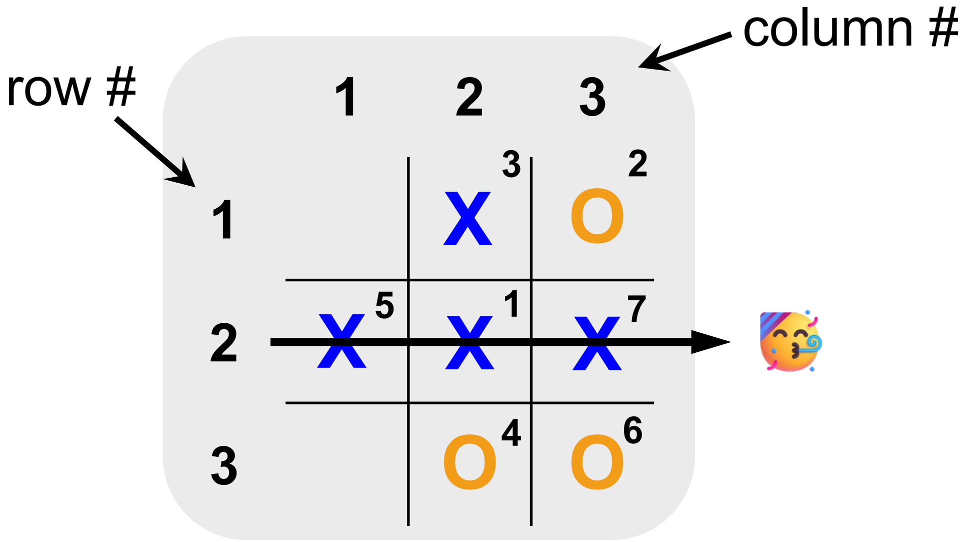
OOP
2-3

Testing
1

Algorithms
1

Your task today: Tic Tac Toe

A 3×3 game board



2 players:
X and O play in turn

step 1	player **X**	row **2**	col **2**
step 2	player **O**	row **1**	col **3**
step 3	player **X**	row **1**	col **2**
step 4	player **O**	row **3**	col **2**
step 5	player **X**	row **2**	col **1**
step 6	player **O**	row **3**	col **3**
step 7	player **X**	row **2**	col **3**

game over, player **X** win!

Note: A ***tie*** occurs when the board is full and neither player has won.

Your task today

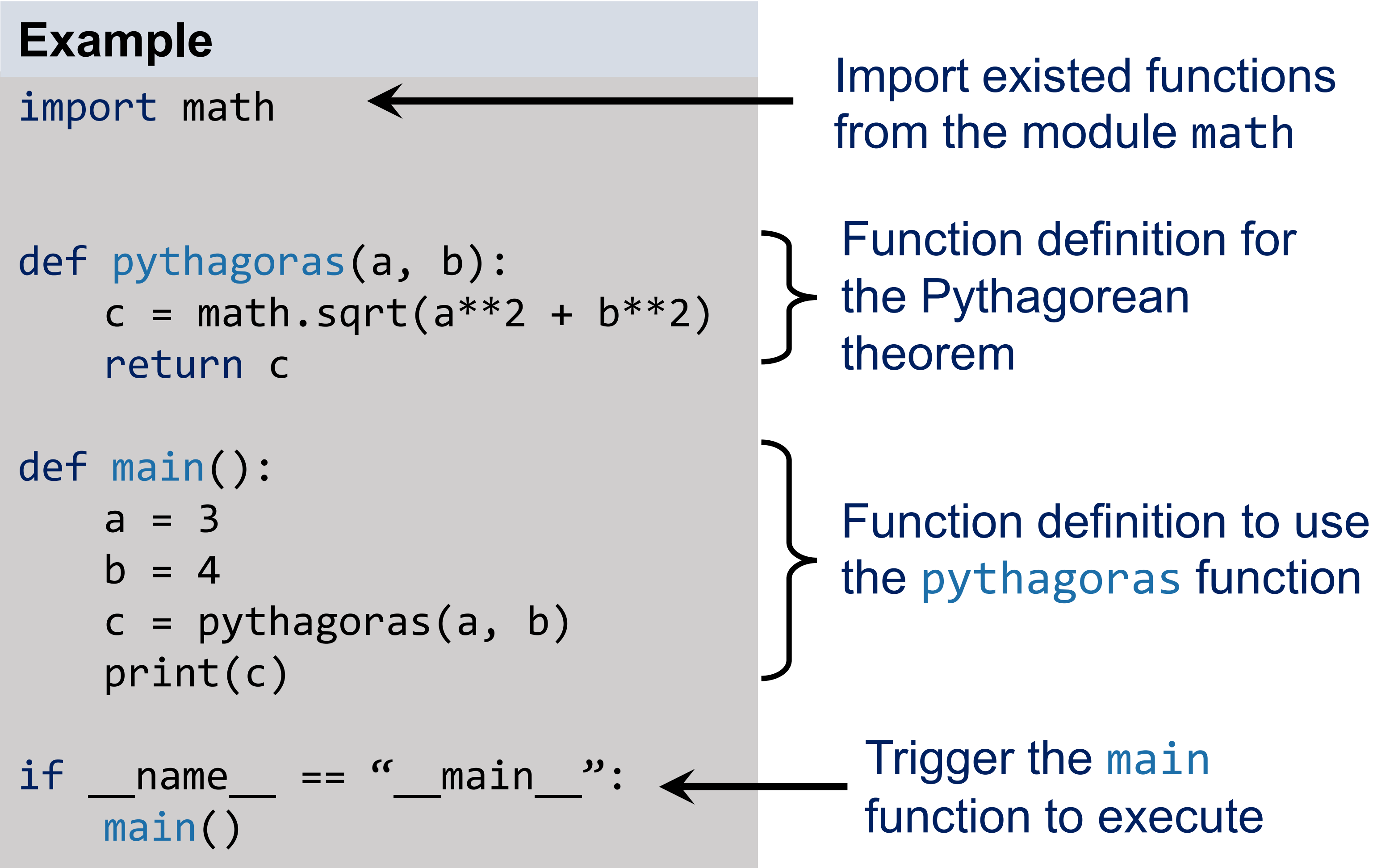
Write a Python programme to realise the game Tic Tac Toe. **Modularise** your programme (using functions), and your code should consider the following aspects:

1. How many steps do you need? Draw yourself a flowchart on a piece of paper, it may include…
   - **Format** a 3×3 board;
   - **Switch**/set a player, **take** the move;
   - **Update** the cells in the 3×3 board;
   - **Check** if the termination condition reached: X/O win the game? Tie?
2. If you code this flow using functions, how many functions you may need? (i.e., how many functions are reusable?)
3. Error/exception handling: check user input – when to accept or reject?

Modular Programming

Modules (**functions**) are put together to make up one executable program.

- Functions are separately defined, hence, reusable
- Functions are triggered serially in a main script (caller)



Example

```
import math


def pythagoras(a, b):
    c = math.sqrt(a**2 + b**2)
    return c

def main():
    a = 3
    b = 4
    c = pythagoras(a, b)
    print(c)

if __name__ == “__main__”:
    main()
```

Import existed functions from the module `math`

Function definition for the Pythagorean theorem

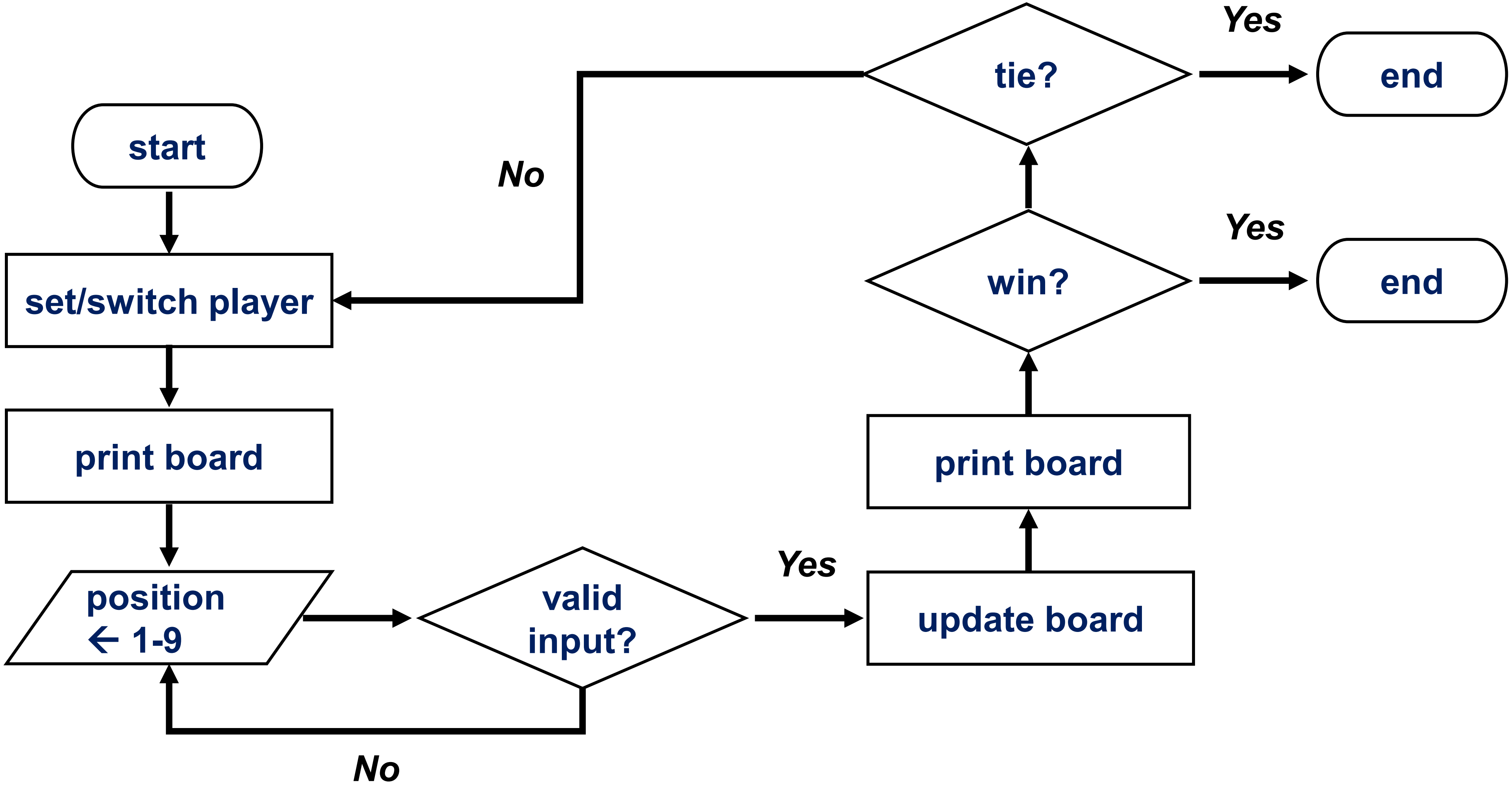
Function definition to use the `pythagoras` function

Trigger the `main` function to execute

Questions?

That's it for now.

You can now proceed to the Lab 1 exercise.


start
set/switch player
print board
position
← 1-9
valid
input?
No
Yes
update board
print board
win?
Yes
end
tie?
Yes
end
No