Open in Colab

Lecture 1: Introduction

Coding: Introduction to Python - Applied Economics

Laboratory: Coding - Accounting, Finance and Business Consulting

Coding

- In this course you will learn the fundamentals of coding
- Coding is how we communicate with computers
  - **Allows you to tell a computer what actions to perform**
  - Making you able to exploit the opportunities offered by computers
- We will mainly focus on data analysis
  - Digital revolution brought unprecedented amount of data and analytics is the key to extract knowledge/value from it
- Coding defines an **algorithm** to be performed by the computer
  - An algorithm is a set of instructions for solving a problem or accomplishing a task

This Course - Timetable and Office Hours

Coding: Introduction to Python
Monday
16:30 - 18:30

Laboratory: Coding
Tuesday
12:30 - 14:30

Office Hours:
See on my website

Dsea, via del Santo 33, first office on the left entering the Levi Cases inner garden

This Course - Objective and Topics

An introductory course giving you the tools to carry out a simple data analysis in Python

Tentative list of topics:

- Setting up your python environment

- Introduction to algorithms
- Intro to data analysis
- Numpy and main data types
- Pandas and dataframes
- Data import and export
- Loops and Iteration
- Data description and visualization
- Data analysis example

This Course - Evaluation - Notions

Pass/Fail exam at the LEM

Closed-ended multiple-choice questions

- Testing your knowledge of the language
  - Given `var_a = 'True'`, what type of data is the variable `var_a` ?
  1. Boolean
  2. String
  3. Integer
  4. Float

This Course - Evaluation - Understanding

Pass/Fail exam at the LEM

Closed-ended multiple-choice questions

- Testing the understanding of the language
  - what is the output of `var_a = 1; var_b = 2; var_c = var_a + var_b; print(var_c)` ?
  1. 1
  2. 2
  3. 3
  4. "error"

This Course - Evaluation - Errors

Pass/Fail exam at the LEM

Closed-ended multiple-choice questions

- Testing the ability to spot errors in the code
  - what is the first error that will be raised by `var_a = 1; var_b = "two"; var_c = var_a + var_b; print(var_d)` ?

1. `NameError: name 'var_d' is not defined
2. TypeError: unsupported operand type(s) for +: 'int' and 'str'
3. TypeError: unsupported operand type(s) for +: 'float' and 'str'
4. SyntaxError: EOL while scanning string literal

This Course - How to prepare?

The course is not designed to be absorbed passively

Clearly the basis is reading, understanding and being able to recall the material

- The material is going to be available in Notebook format (`.ipynb` files) that include explanations, code, and code output
  - You will be able to run the code and modify it to test your understanding

This Course - How to prepare?

- As you progress with the material you should experiment with the tools and syntax we cover
  - Simply reading the material will not be enough
  - Coding is a practical endeavour
    - You can gain "muscle" memory from doing things instead of just memorizing concepts
- I will be proposing exercises for each topic
  - Solving them is very important to test your understanding
  - Crucial to get prepared for the exam
  - Usually teach you how to do basic (and crucial) tasks that you might need in other courses

This Course - Books and Additional Material

The course relies on material from a variety of sources, such as books, software documentation, online tutorials, blogposts, etc.

The main sources of the material in this course are two ***great and free*** books:

Python for Data Analysis by Wes McKinney

Python Data Science Handbook by Jake VanderPlas

During the course, I will provide links to additional material that might help in understanding the topics covered

Coding, what are we talking about?

Coding is the process of writing instructions for a computer to perform

- When we code we define an **algorithm** to be performed by the computer
- An algorithm is a set of instructions for solving a problem or accomplishing a task
- A data analysis is a set of algorithms to be performed on data:
  - Importing data
  - Cleaning data
  - Describing data
  - Visualizing data

Coding, what are we talking about?

- In this process you usually start with some questions that you want to answer with the data
  - Define the steps needed to answer those questions
  - Write the code and run it to get the answers
  - While doing so you will learn more about the data and the questions you are asking
    - Adjust your original questions and the steps needed to answer them
  - This is an iterative process that leads to the final analysis
  - The final analysis will be a set of algorithms that you can run on new data to get the answers to your questions
  - Answers to your questions will be descriptive statistics, plots, tables, statistical tests, etc.

Computers, a Simplification

- Computers basic pieces are
  - Memory: used to store information
  - Central Processing Unit (CPU): used to perform operations on the information stored in memory
- Numbers (integers) are the only thing that computer memory can store and the CPU can manipulate

Computer, Code and Interpreters

- **How can we program a computer to execute a specific algorithm?**
  - To be executed, an algorithm is converted into a sequence of instructions that are valid for a specific CPU (Intel processors, ARM processors, etc.)
  - The instructions are written in some **high-level language** (C, Python, R, etc.) that is easier to understand for humans
  - An intereter (or compiler) transforms the high-level language into machine code (a sequence of numbers) that can be executed the computer

    - In this course we will define a series of these tasks, and write in the Python language to be executed by the computer

Representing Complex Data as Integers

- If computer can only store and process integers, how come that we have and manipulate text, images, videos, etc. on our devices?
    - Computer scientists have devised clever rules (or conventions, or **standards**) to encode a specific type of data into a sequence of integers
    - Data that is analog by nature (ex an image, a sound, etc) must first be digitized:
        - Bitmap images encode images as a grid of pixels, each pixel is a number representing the color
        - Text is encoded as a sequence of numbers representing the characters in the text
- Always losing some information in the process
    - The more information you want to store, the more memory you need
    - The level of current technology allows us to have indistinguishable quality from the original analog data e.g., retina display

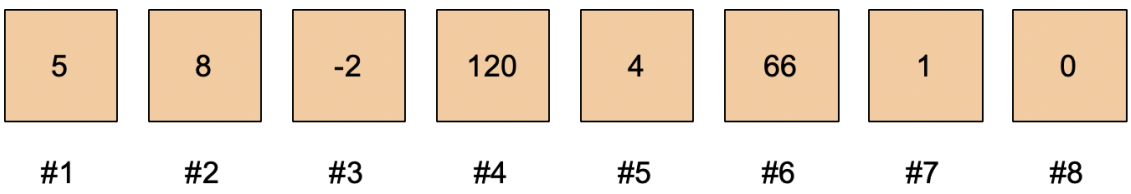
CPU can only do basic operations

- CPU can only perform basic operations on integers
    - Addition, subtraction, multiplication, division, etc.
    - Comparison: equal, greater than, less than, etc.
    - Logical operations: and, or, not, etc.

A Very Simple Representation of a Computer

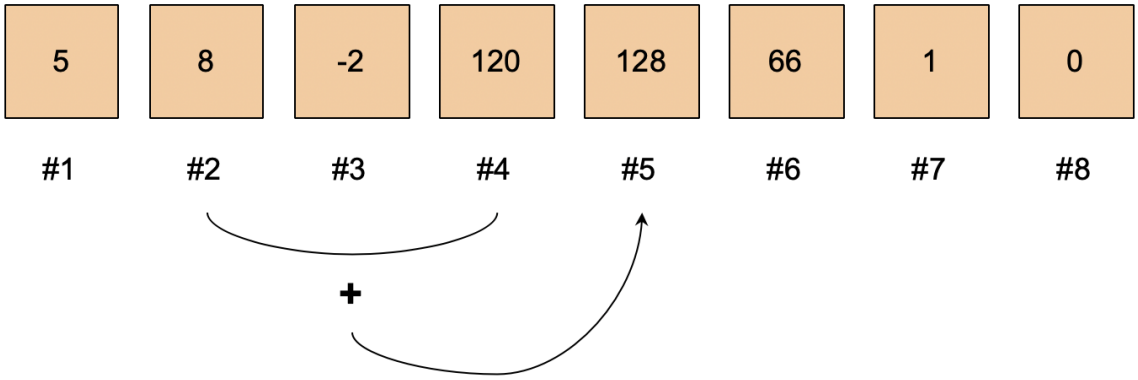
To understand how a computer works, we can think of a very simple representation:

- **Memory** is represented by a set of boxes
    - Each box has a label and contains a number
    - The label is the address of the box
- **CPU** is represented by Tom
    - Tom must sequentially execute instructions that operate on the box content
    - Tom can do a lot of these operations on the box content, never does mistakes and never gets tired



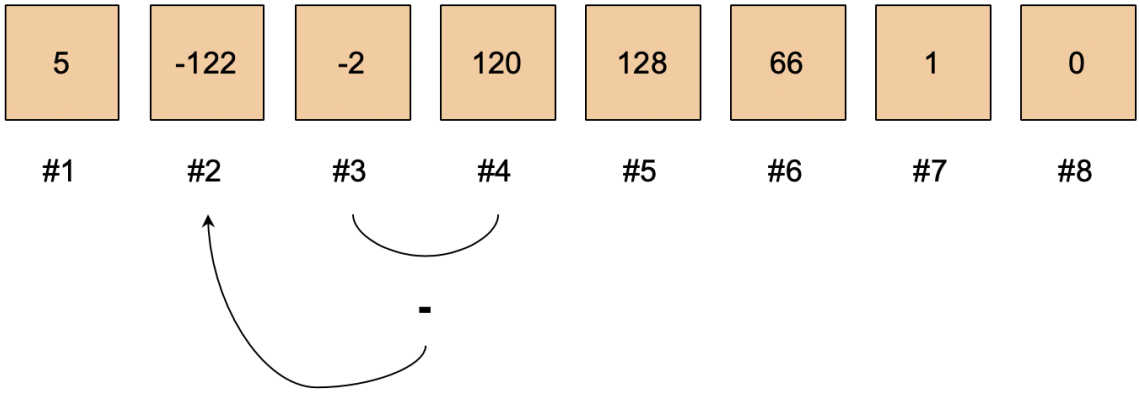
A Very Simple Representation of a Computer

- Add the number contained in box #2 to the number contained in box #4 and place the result in box #5



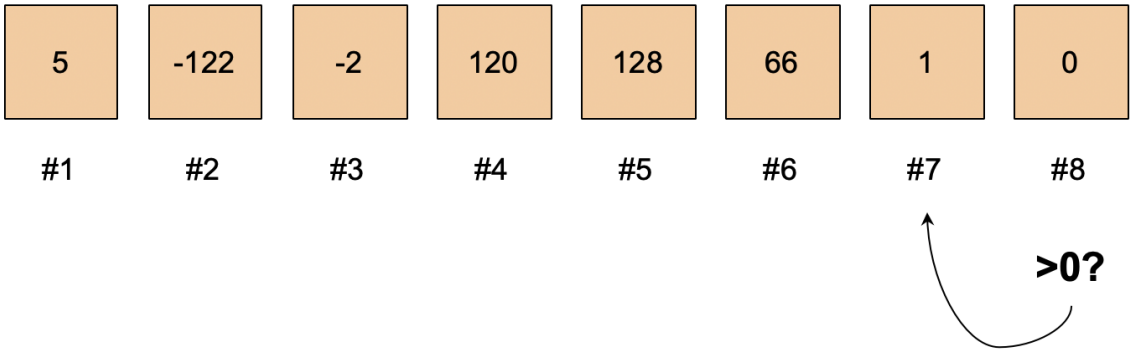
A Very Simple Representation of a Computer

- Another example:
  - Subtract the number contained in box #4 to the number contained in box #3 and place the result in box #2



A Very Simple Representation of a Computer

- Another example:
  - If box #8 is greater than zero, put number 10 in box #7. Do nothing otherwise



Writing an algorithm

- How can we perform **complex** operations on **complex** data?
  - We need to know the actions you want the computer to perform
  - The order (and cases) those actions should be performed in
- A well thought-out, step-by-step solution to the problem
  - Writing an algorithm is "finding a way" to get to our goal using the operations offered by the language
  - To code you need:
    - A thorough understanding of the problem
    - A well thought-out, step-by-step solution to the problem
    - Knowledge of a computer language to implement the solution

Coding is Problem Solving Plus Implementation

The usual phases of algorithm's writing are:

1. **Analyze** the problem
2. **Devise** different solutions to the problem
3. **Design** an approach that will solve
4. **Implement** that design - Translate the algorithm into a programming language
5. **Test** to see if it works

Pseudocode

- During the design of an algorithm you should be able to decompose it in well-thought series of steps
  - This is called **Pseudocode** - simply an implementation of the algorithm in plain English
  - Not actual programming language — can’t be compiled or interpreted by the computer
  - Allows you to focus on the logic of the algorithm - no distraction by details of the programming language

    - It describes the logic/steps of the algorithm - implementation becomes a simple translation into code

A Simple Algorithm

- Let's say that our goal is to implement the division operation on our computer
    - We want our computer to be able to divide two numbers m/n and return the results: quotient and remainder
- We only have the computer as described above:
- Can only store store integers
- Can only perform addition, subtraction and conditional checks on integers

A Very Simple Representation of a Simple Algorithm

First, we need to define how the numbers of our problem will be stored in the boxes

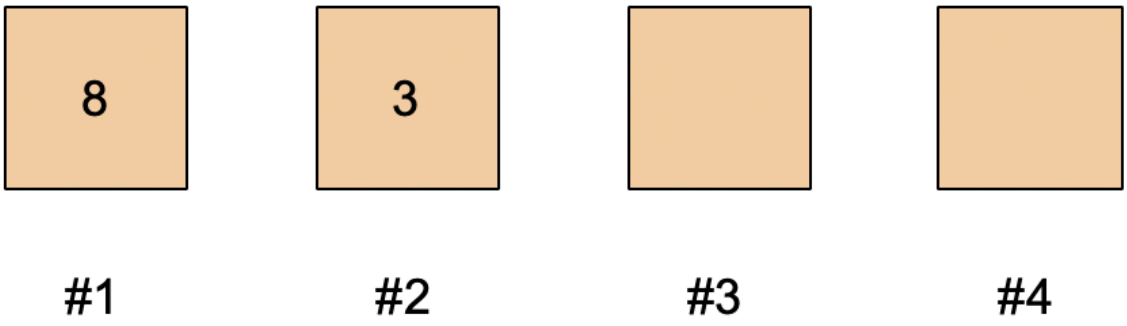
- **Inputs**:
    - Content of box #1 is called **dividend**
    - Content of box #2 is called **divisor**
- **Outputs**:
    - Content of box #3 is called **quotient**
    - Content of box #4 is called **remainder**

- The only instructions that Tom can perform are:
    - Place a number in a box
    - Sum/Subtract
    - Compare if a number contained in a box is greater than another

A Simple Algorithm

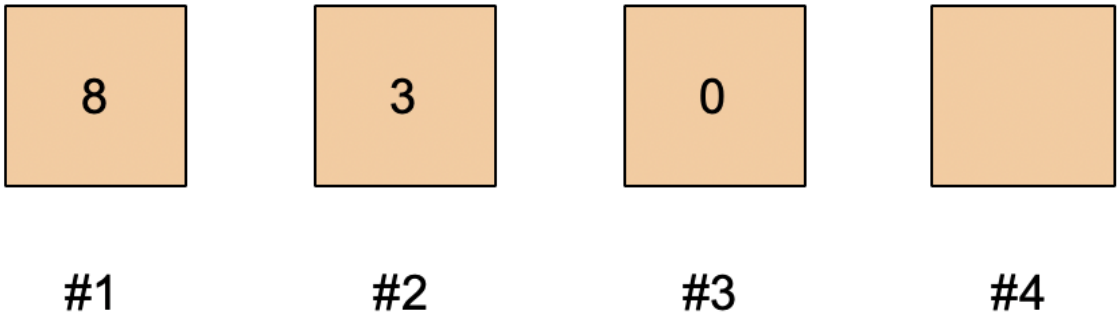
Computer scientists have devised this clever algorithm to perform division:

1. Put 0 in box #3
2. Put the content of box #1 in box #4
3. If content of #4 is >= than content of #2
    A. Subtract #2 to #4 and place the result in #4
    B. Add 1 to #3 and place the result in #3
    C. Return to step 3
4. HALT



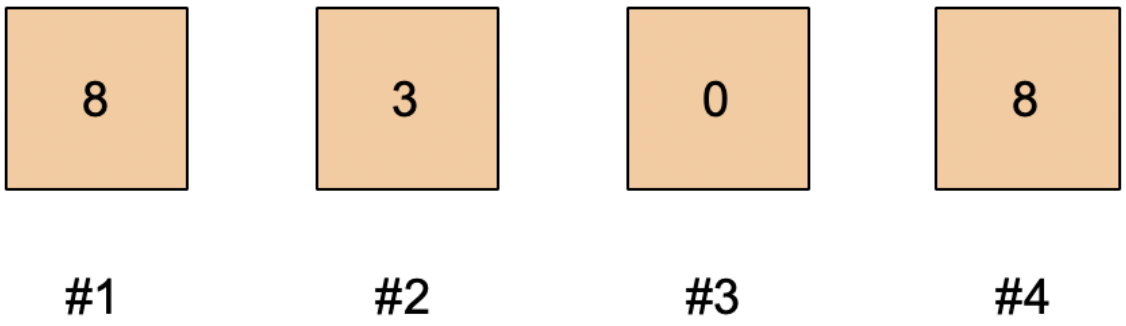
A Simple Algorithm

1. **Put 0 in box #3**
2. Put the content of box #1 in box #4
3. If content of #4 is >= than content of #2
    A. Subtract #2 to #4 and place the result in #4
    B. Add 1 to #3 and place the result in #3
    C. Return to step 3
4. HALT



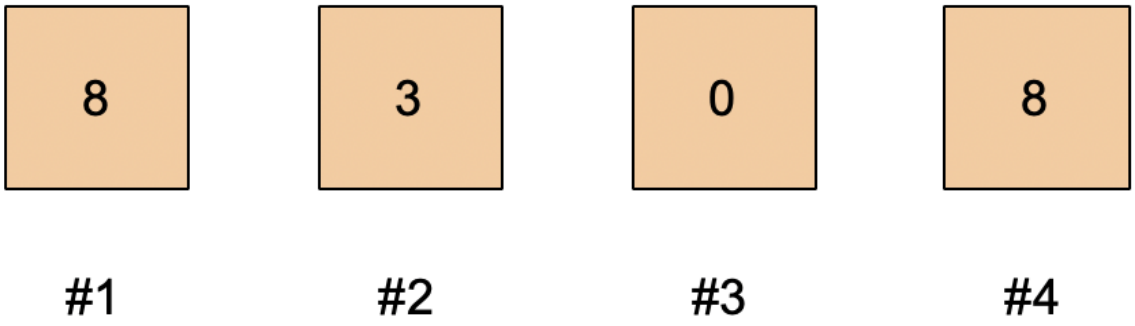
A Simple Algorithm

1. Put 0 in box #3
2. **Put the content of box #1 in box #4**
3. If content of #4 is >= than content of #2
    A. Subtract #2 to #4 and place the result in #4
    B. Add 1 to #3 and place the result in #3
    C. Return to step 3
4. HALT



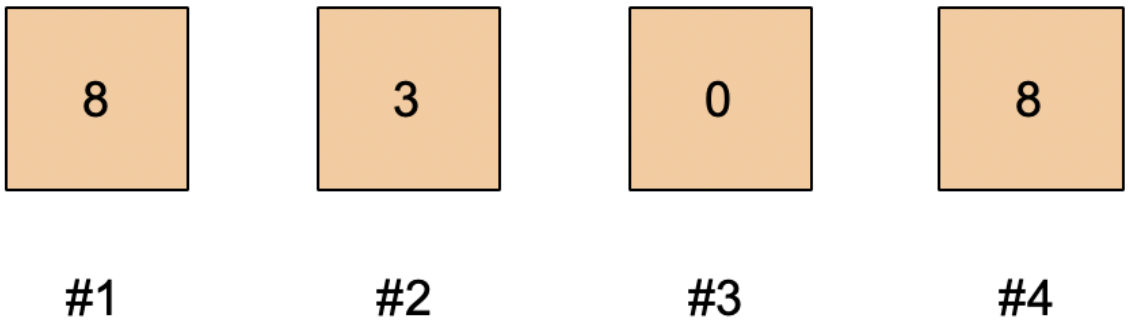
A Simple Algorithm

1. Put 0 in box #3
2. Put the content of box #1 in box #4
3. **If content of #4 is >= than content of #2**
    A. Subtract #2 to #4 and place the result in #4
    B. Add 1 to #3 and place the result in #3
    C. Return to step 3
4. HALT



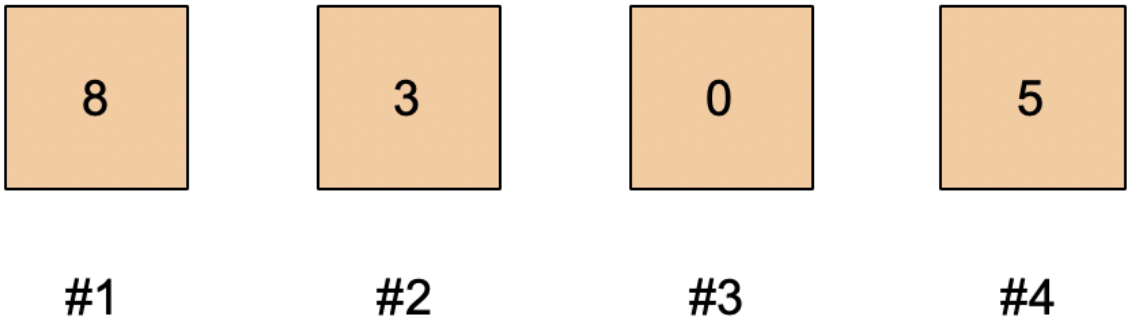
A Simple Algorithm

1. Put 0 in box #3
2. Put the content of box #1 in box #4
3. If content of #4 is >= than content of #2
    A. **Subtract #2 to #4 and place the result in #4**
    B. Add 1 to #3 and place the result in #3
    C. Return to step 3
4. HALT



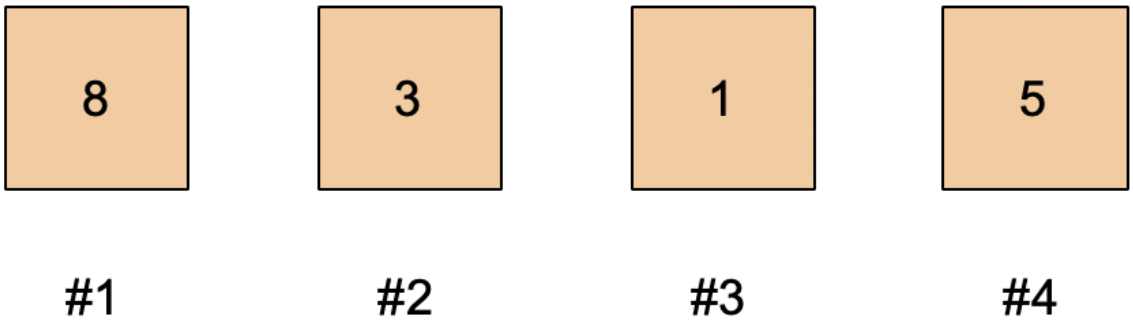
A Simple Algorithm

1. Put 0 in box #3
2. Put the content of box #1 in box #4
3. If content of #4 is >= than content of #2
    A. **Subtract #2 to #4 and place the result in #4**
    B. Add 1 to #3 and place the result in #3
    C. Return to step 3
4. HALT



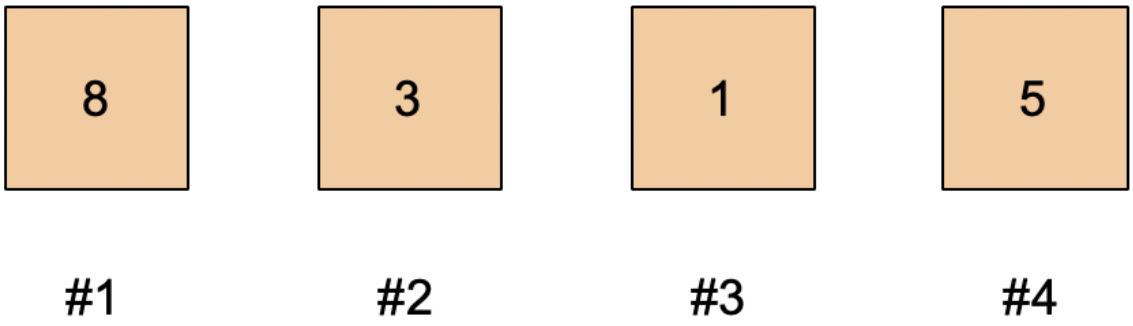
A Simple Algorithm

1. Put 0 in box #3
2. Put the content of box #1 in box #4
3. If content of #4 is >= than content of #2
    A. Subtract #2 to #4 and place the result in #4
    B. **Add 1 to #3 and place the result in #3**
    C. Return to step 3
4. HALT



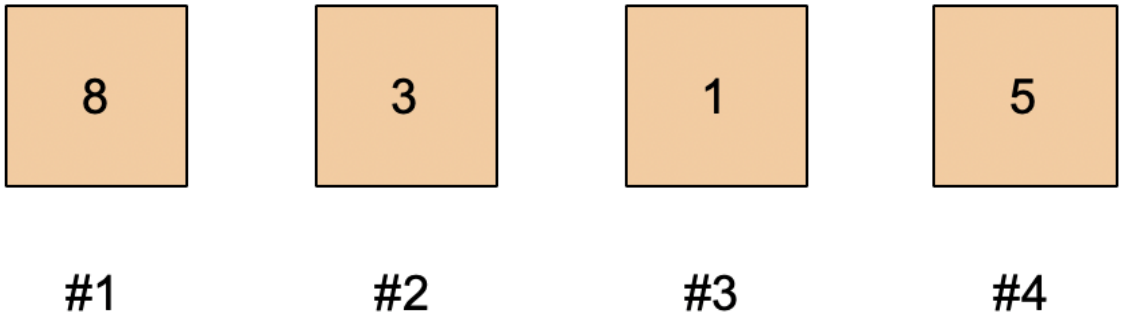
A Simple Algorithm

1. Put 0 in box #3
2. Put the content of box #1 in box #4
3. If content of #4 is >= than content of #2
    A. Subtract #2 to #4 and place the result in #4
    B. Add 1 to #3 and place the result in #3
    C. **Return to step 3**
4. HALT



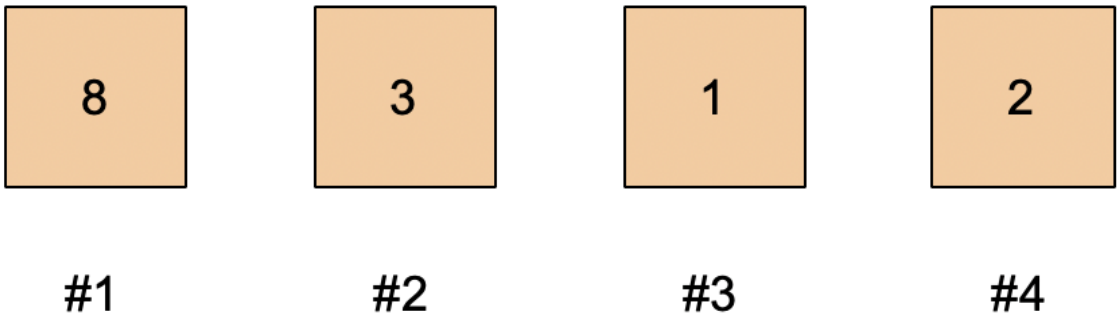
A Simple Algorithm

1. Put 0 in box #3
2. Put the content of box #1 in box #4
3. **If content of #4 is >= than content of #2**
    A. Subtract #2 to #4 and place the result in #4
    B. Add 1 to #3 and place the result in #3
    C. Return to step 3
4. HALT



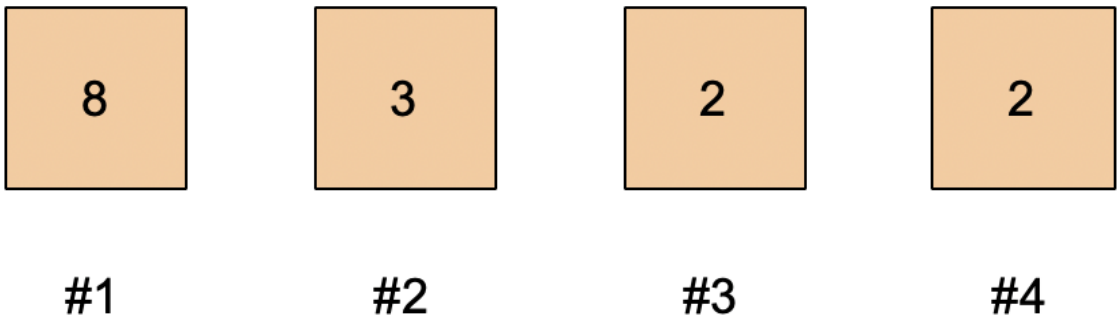
A Simple Algorithm

1. Put 0 in box #3
2. Put the content of box #1 in box #4
3. If content of #4 is >= than content of #2
    A. **Subtract #2 to #4 and place the result in #4**
    B. Add 1 to #3 and place the result in #3
    C. Return to step 3
4. HALT



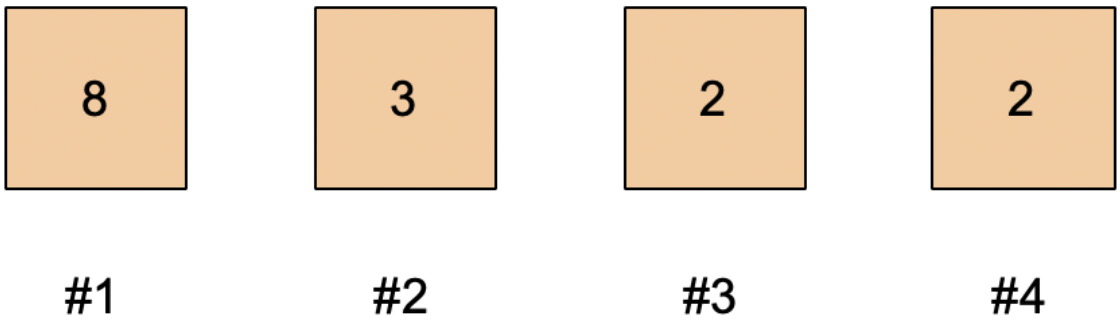
A Simple Algorithm

1. Put 0 in box #3
2. Put the content of box #1 in box #4
3. If content of #4 is >= than content of #2
    A. Subtract #2 to #4 and place the result in #4
    B. Add 1 to #3 and place the result in #3
    C. **Return to step 3**
4. HALT



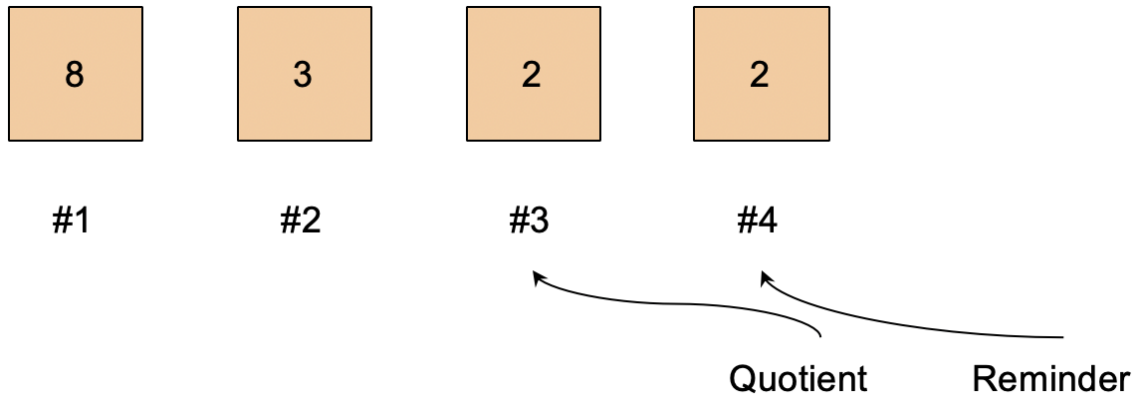
A Simple Algorithm

1. Put 0 in box #3
2. Put the content of box #1 in box #4
3. **If content of #4 is >= than content of #2**
    A. Subtract #2 to #4 and place the result in #4
    B. Add 1 to #3 and place the result in #3
    C. Return to step 3
4. HALT



A Simple Algorithm

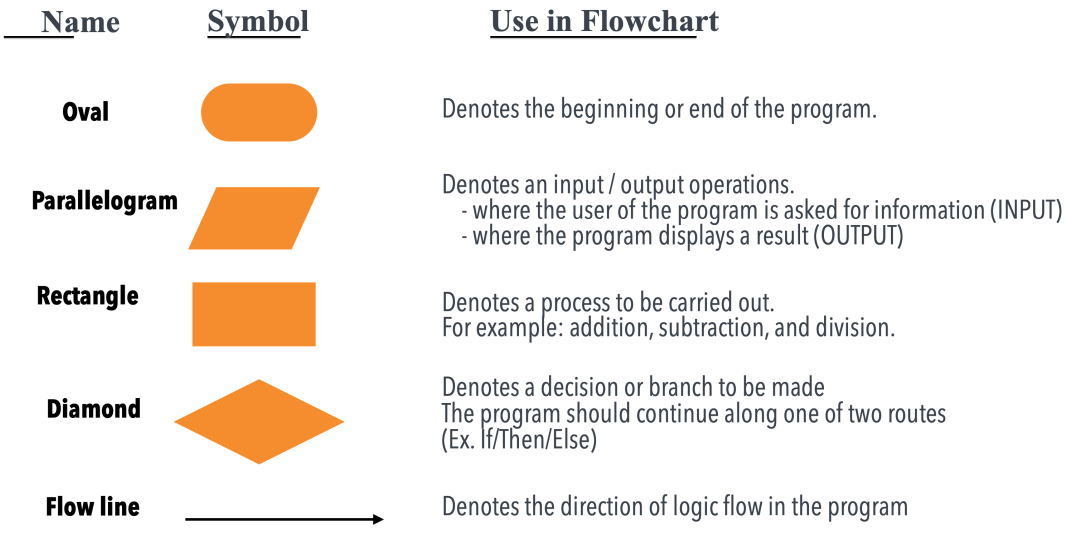
1. Put 0 in box #3
2. Put the content of box #1 in box #4
3. If content of #4 is >= than content of #2
    A. Subtract #2 to #4 and place the result in #4
    B. Add 1 to #3 and place the result in #3
    C. Return to step 3
4. **HALT**



Flowchart

- The algorithm can also be written as a FLOW CHART
- The **flow chart** is a graphical rapresentation of the (a picture that helps organize your thoughts)
- It uses a collection of basic symbols that are used to organize your algorithm
- These symbols are connected by arrows that show how the algorithm “flows”

Flowchart Symbols



An everyday example

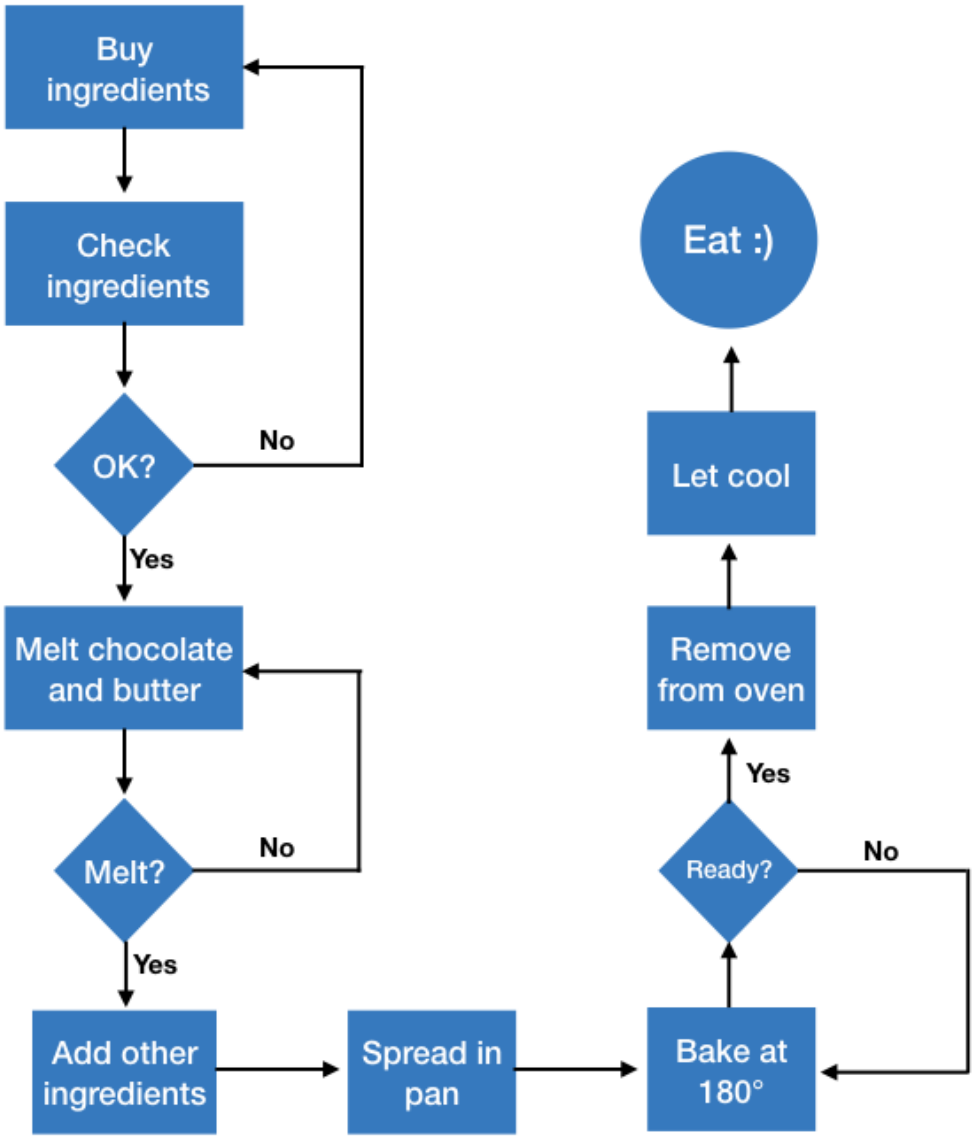
Write a simple algorithm for making a chocolate cake:

Ingredients: 4 oz. chocolate, 1 cup butter, 2 cups sugar, 3 eggs, 1 tsp. vanilla, 1 cup flour

Instructions: Melt chocolate and butter. Stir sugar into melted chocolate. Stir in eggs and vanilla. Mix in flour. Spread mix in greased pan. Bake at 180 degrees Celsius for 40

minutes or until inserted stick comes out almost clear. Cool in pan before eating.

A Possible Solution



In summary

Coding is a way to operationalize the achievement of a certain (computational) goal

- Identify the tasks to be performed
- Figure out how how these tasks can be achieved using basic operations that the computer can do
- These basic operations can follow a single path or multiple paths, depending on conditions
    - Pseudocode and flowcharts are extremely useful at this stage

- Usually there is no just one way to perform a task, but each have their trade-offs:
  - Faster to write/slower to execute
  - Faster to execute/using more memory

In summary

- Contrary to the examples above you will have Python as your tool
  - Lots of built-in operations and libraries that already address a multitude of use-cases
    - Still need a lot of work to adapt to each research question and dataset
    - Reality is messy $\rightarrow$ data that represents it is messy
  - Continuously improved by a very active community
    - The productivity of IT led huge amount of resources and very bright people to be devoted to it)
- In this course we will get to know some basic (but pretty powerful) operations and libraries, and how they can be combined together

Homework

1. Draw the flowchart for the division algorithm presented in the slides
2. Draw the flow-chart of an algorithm that finds the maximum of three numbers A, B and C provided by the user
3. Draw the flow-chart of an algorithm that returns the sum of three consecutive numbers starting from N, which is provided by the user. Ex: if N=5 the output will be 5+6+7 = 18
4. Draw the flow-chart of an algorithm that sums the numbers given by the user, until the user provides zero. Then, the algorithm ends getting the result of the sum

Notes

Book references:
Ch.1 of Python for Data Analysis
Preface of Python Data Science Handbook

Office Hours:
See on my website:

Author: Duccio Gamannossi degl'Innocenti
Web: https://www.dgdi.me/
Mail: duccio.gamannossi@unipd.it