LEARN CODING

ale66

github.com/ale66/learn-coding

WELCOME

- Introduction 👋
- Materials ⬇️

PLAN

1. What is Coding? Why should I learn it?
2. A tour of computers
3. A tour of [online] learning resources
4. A peek at topics

WHAT IS CODING?

WHAT IS CODING?

A creative activity where computers are instructed directly to perform useful, step-by-step operations.

YES, BUT HOW?

In music we go from *imagined* sounds to music score to execution (press piano keys, blow air in trumpet, harp on strings) to actual, perceived sounds.

Humans communicate on three levels: natural language, the music score and the execution.

METHAPHORICALLY

- the music score represents the code.
- execution on a specific instrument represents executable code, e.g., file chrome.exe on your computer
- the hearing *experience* represents the changes that take place on your data.

Coding in some ways is like music composition

UN-METAPHORICALLY...

- an informal language will describe algorithms on paper, on whiteboard etc.
- a formal language, Python or SQL or Markdown, will describe code.
- special interpretation/compilation software will take code and execute it.
- we need to supply *data* and store results.

ALGORITHMS, BY EXAMPLE

the Farenheit temperature in Naples can be obtained by

1. taking the current temperature in Celsius degrees
2. rescale it by $\frac{9}{5}$
3. re-center it by adding 32

CODE

```
1 my_celsius = int(input('Please enter the current temperature in Naples:'))
2
3 my_farenheit = (my_celsius * 9/5) + 32
4
5 print(f'Today we have {my_farenheit} Farenheit degrees in Naples')
```

Reading this syntax requires training.

IMPLEMENTATION: COMPILATION

```
1 >python -m py_compile my_converter.py -o converter.exe
2 >converter.exe
3 >Please enter the temperature in Naples now:
```

file converter.exe is for computers, not humans.

IMPLEMENTATION: INTERPRETATION

```
1  >python my_converter.py
2  >Please enter the current temperature in Milan:
```

COMPUTERS

COMPUTERS

Electronics, networks

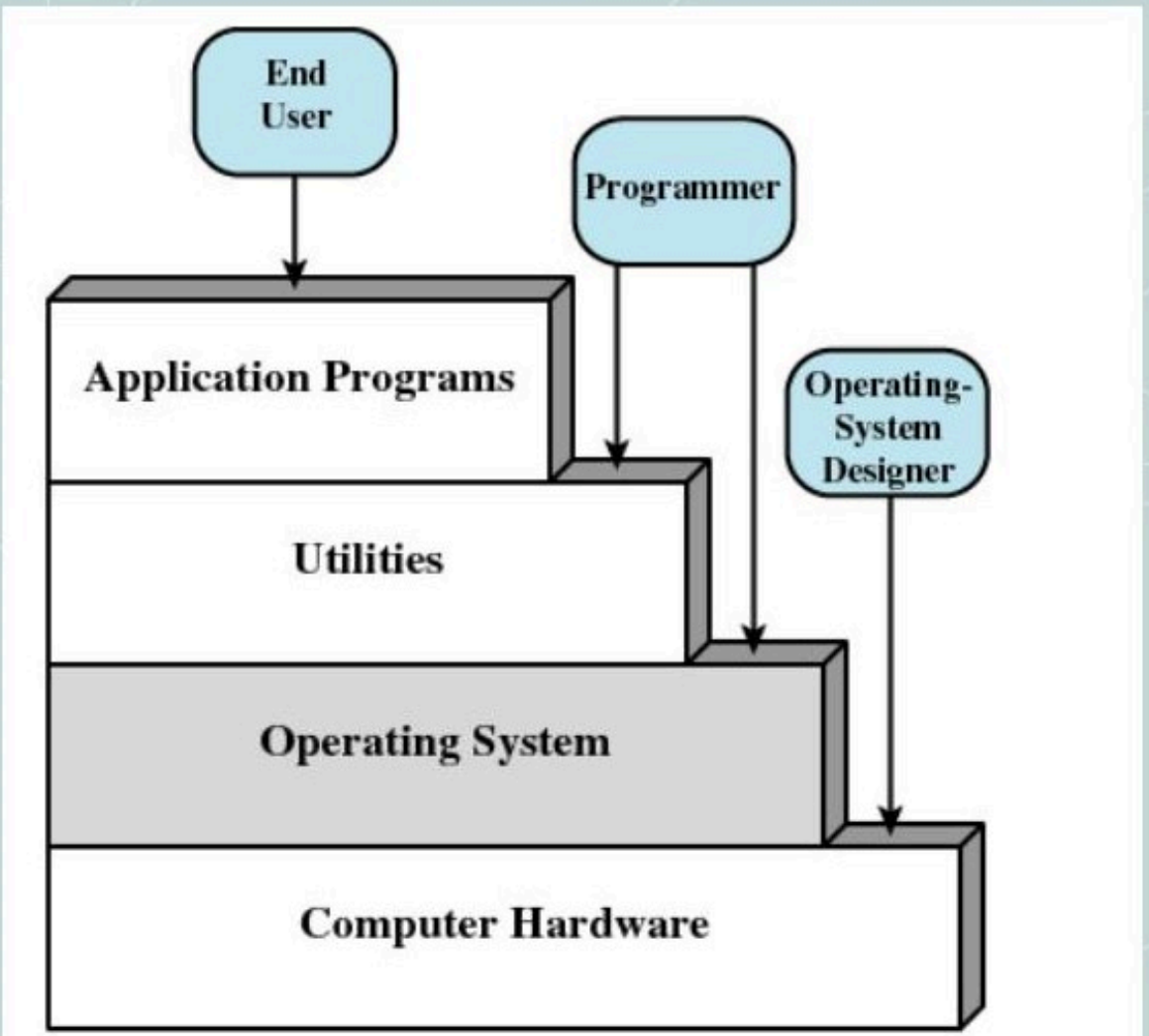
Operating system

Software

OPERATING SYSTEMS

- computers/smartphones come with a pre-loaded set of executable files that create the *operating system environment*
- create an abstract view of the computer: specific hardware details are now trasparent
- special abstraction: **the file system**
- all of them offer basic functionalities for coding
- let's get started with the **file system** and the **command line interface**

Layers of computer system



AND WHAT'S A FILE?

- computer memory is best seen as a long ribbon where, at different times, we write sequences of bytes, called *files*
- a file is i) a unique name, ii) a memory address inside the compuhardware iii) a sequence of bits, the actual content, iv) ownership information (for later) and v) a format that guides the interpretation of the bits: are they color pixels? Characters? Numbers?

THE FILE SYSTEM

- the Os shows files are organised in a hyerachical structure of *manila folders*

- hyerarchy is not about importance. It rather creates *locality*

THE FILE SYSTEM

- a special file, called folder, contains the names and physical addresses of the files within
- each folder contains two special (and secret) files: `.` and `..`
- file `.` contains a reference to the actual positioning of the folder on the memory device
- file `..` contains a reference to the *containing* folder

THE CLI: COMMAND-LINE INTERFACE

- iOs: `Terminal`
- Win: `Cmd` or `Powershell` or `Windows terminal`
- Linux/Android: `sh` or `bash`

Structure:

```
1  >pwd
2  >dir
3  >notepad my_converter.py
4  >cd ..
5  >dir
6  >pwd
7  >notepad my_converter.py
```

PATHS

File must have unique names

No repeated names in the same folder

Ok to repeated names in different folders, how?

absolute path: `C:\Users\ale\git\learn-coding\00-create_platform\my_converter.py`

relative path: `.\my_converter.py`

More relative paths:

```
1  >pwd
2  C:\Users\ale\git\learn-coding\10-first_steps
3  >python ..\00-create_platform\my_converter.py
```