

# Programming in Python[1]

Mattia Monga

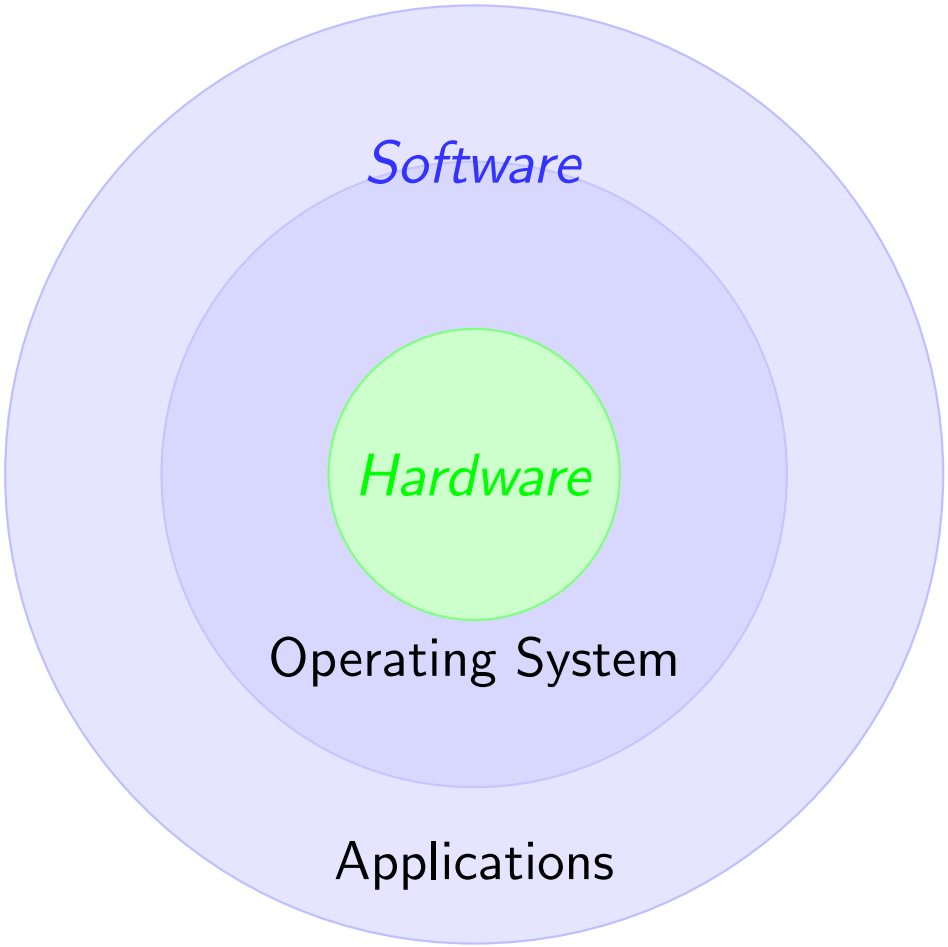
Dip. di Informatica
Università degli Studi di Milano, Italia
mattia.monga@unimi.it

Academic year 2022/23, I semester



# Lecture II: Fundamentals

# The onion model




- Operating System: it is the only program interpreted directly by the hardware; other pieces of software get interpreted by the virtual machine provided by it.
- Applications: programs (e.g., the python interpreter or python programs) executed within the protected environment created by the operating system.

# What we want to do




- Programming means to instruct an (automatic) interpret with a precise description of a computational process.
- (In fact, the only way to make a description precise is to specify exactly the interpreter)
- We use a software interpreter, itself a program interpreted by the operating system (the stack of interpreters can be much deeper).
- Our interpret (Python3) manipulates objects taken from types (that define which manipulations are possible), referred by variables, with special commands to ask the services provided by the operating system.

# Assignment




This is the fundamental statement for imperative programming:

- A name, known as variable, is needed to refer to objects.
  `professor = "Mattia"`
- = is not symmetrical, read it as becomes: Left-hand-side is always a variable, right-hand-side is an object, that can be either a literal or anything referred by another variable.
- A variable can change its value with another, following, assignment. Thus, the same variable may refer to different objects.
  `professor = "Violetta"`
- Basic objects (numbers, strings, Boolean values) are immutable (the variable change, not the object; different objects have always different identity)
- Tracking a program means to track the values of all the variables of a program during its execution.

# Type hints



Since Python 3.4 it is possible (and indeed desirable, especially for novices) to hint any reader of a program about the type of a variable.

- A variable has always a type (a string in this case) `professor = 'Mattia'`
- Type hints make clear the intention of the programmer (can be checked by external programs) `professor: str = 'Mattia'`
- Assigning to an object of another type is still possible (there is no syntax error raised), but it should be regarded with suspicion `professor = True`

# Basic operations




- Binary operators: `5 + 2`, they compute a new object by using the two objects on which they apply;
- Unary operators: `-(-5)`;
- Functions: `max`, they compute a new object by using an arbitrary number of objects (in general 0–..., `max` takes at least 1) passed as parameters (or arguments) when the function is called (`max(3, 6, something_else)`); sometimes the object computed is `None`;
- Syntactically appear as functions, but *commands* like `print("Hello!")` are actually used to request side effects in the executing environment.

Official Python docs (3.10)

# Different approaches


Problem: exchange the name of two objects (Chapter 1, last exercise).

- Know the basic syntax of variables and assignment =
- Know the semantics of what you write: assigning an object to a variable delete any previous assignment;

# Different approaches


Problem: exchange the name of two objects (Chapter 1, last exercise).

- Know the basic syntax of variables and assignment =
- Know the semantics of what you write: assigning an object to a variable delete any previous assignment;
- Natural strategy: use a temporary name to "save" the value during the exchange;

# Different approaches




Problem: exchange the name of two objects (Chapter 1, last exercise).

- Know the basic syntax of variables and assignment =
- Know the semantics of what you write: assigning an object to a variable delete any previous assignment;
- Natural strategy: use a temporary name to "save" the value during the exchange;
- "Fox" strategy: know language or library tricks For example Python has a "multiple assignment" construct `x, y = y, x`, or a special library function `swap(x, y)` could exist;

# Different approaches




Problem: exchange the name of two objects (Chapter 1, last exercise).

- Know the basic syntax of variables and assignment =
- Know the semantics of what you write: assigning an object to a variable delete any previous assignment;
- Natural strategy: use a temporary name to "save" the value during the exchange;
- "Fox" strategy: know language or library tricks For example Python has a "multiple assignment" construct `x, y = y, x`, or a special library function `swap(x, y)` could exist;
- "Hedgehog" strategy: study the problem in depth, e.g., if objects are numbers you can exploit arithmetic.

```
x = x + y
y = x - y
x = x - y
```

# Basic types

`bool` `False`, `True` Logical operations

`int` `1`, `-33`, `1_000_000` ... Arithmetic operations, no upper or lower limit

`float` `1.0`, `.1`, `1.2e34` ... Arithmetic operations, limited but you have `float('infinity')` (and `float('nan')`)

```
sys.float_info(max=1.7976931348623157e+308
↪   , max_exp=1024, max_10_exp=308,
↪  min=2.2250738585072014e-308,
↪  min_exp=-1021, min_10_exp=-307,
↪  dig=15, mant_dig=53,
↪  epsilon=2.220446049250313e-16,
↪  radix=2, rounds=1)
```

`str` `'aaaa\nthis is on a new line'`, `"bbb'b\"b"` ... Concatenation, alphabetical ordering, replication, ...

# Homework




Finish chapters 1, 1E, 2, 2X, 3, 4.
It shouldn't take more than a couple of hours, but exercising continuously is crucial.