Programming in Python[1]

Mattia Monga

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

Academic year 2025/26, I semester

PyQB
Monga
Software
git
Thonny

Lecture VI: Using the “naked” interpreter

A function computes a result

PyQB
Monga
Software
git
Thonny

- Returns a useful result

```
def concat_with_a_space(string1: str, string2: str) -> str:
   return string1 + ' ' + string2

# string1 is the _formal_ parameter
# 'foo' is the _actual_ parameter (like an assigment string1 =
↪  'foo')
print(concat_with_a_space('foo','bar'))
```

- Return None

```
def repeated_print(string: str, repetitions: int) -> None:
   for i in range(0, repetitions):
     print(string)

repeated_print('Hello, world!', 3)
```

- **Recursive call**:

```
def repeatedPrint(string: str, repetitions: int) -> None:
   if repetitions > 0:
     print(string)
     repeatedPrint(string, repetitions - 1)

repeatedPrint('Hello, world!', 3)
```


Functions are objects too

PyQB
Monga
Software
git
Thonny

One can assign functions to variables:

```
def cube(x: int) -> int:
   square = x * x
   return square * x

mycube = cube

print(mycube(3))
print(type(mycube))
```

And short functions can even be expressed as literal expressions (lambda expressions)

```
cube = lambda y: y*y*y
```

Example: Newton sqrt

Newton's method of successive approximations: whenever we have a guess g for the value of the square root of a number x, we can perform a simple manipulation to get a new guess closer to the actual square root by averaging g with $\frac{x}{g}$.

```
def newton_sqrt(x: float) -> float:
    guess = 1.0
    while not good_enough(guess, x):
        guess = improve_guess(guess, x)
    return guess
```

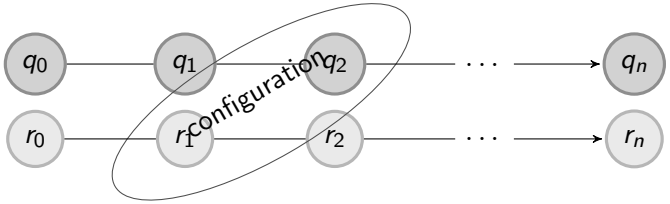

The pieces of software

- Python 3.13, with `pip` (on MS Windows be sure to select it): `https://www.python.org/downloads/`
- You need a text editor: suggested one is `https://thonny.org/`
- Git 2.30+ `https://git-scm.com/downloads`
- (optional, Win and Mac only) Github desktop `https://desktop.github.com/`

Homework assigments will be available via Github Classroom (you will need a Github account).
When you push (hand in) your solution, a suite of tests is run.

Software Configuration Management

Software Configuration Management like `git` are tools designed to track all the revisions of some set of software artifacts (files).



The system configuration itself evolves in different versions.
One can have multiple branches of evolution.
A motivating talk on why you should use tools like these in your scientific work.

Git

`git` is a powerful tool to manage all this complexity in a very efficient (and distributed) way. It is not an easy tool, however. A good tutorial is here. But for this course we use a very simplistic workflow:

1. Clone (copy) on your machine a repository `git clone ...` existing on a server (GitHub, probably);
2. Work on the artifacts (files)
3. Add the modified artifacts to the changeset you want to "publish" `git add ...` (this step is important: it makes you think **which** changes you want to "save" forever)
4. Commit (save) the changeset `git commit -m"message"` providing a comment about what have you done
5. Push the changeset on the origin server `git push`
6. (If someone else is working on the same artifacts you can sync with `git pull`)

All these steps are very easy (almost hidden, especially authentication) if you use Github desktop.

Thonny

Programs are data! File extension is conventionally `.py`

- To edit Python programs you need a **text editor**: something like `Notepad`, not `Word` (a word processor)
- `Thonny` is an editor designed for beginners and it provides an easy way for executing programs step-by-step
- Other good choices: `VS Code` `Atom` `Notepad++` `PyCharm` or any other universal text editor like `EMACS` or `vi`

Exercise

`https://classroom.github.com/a/_noVB5Ku`