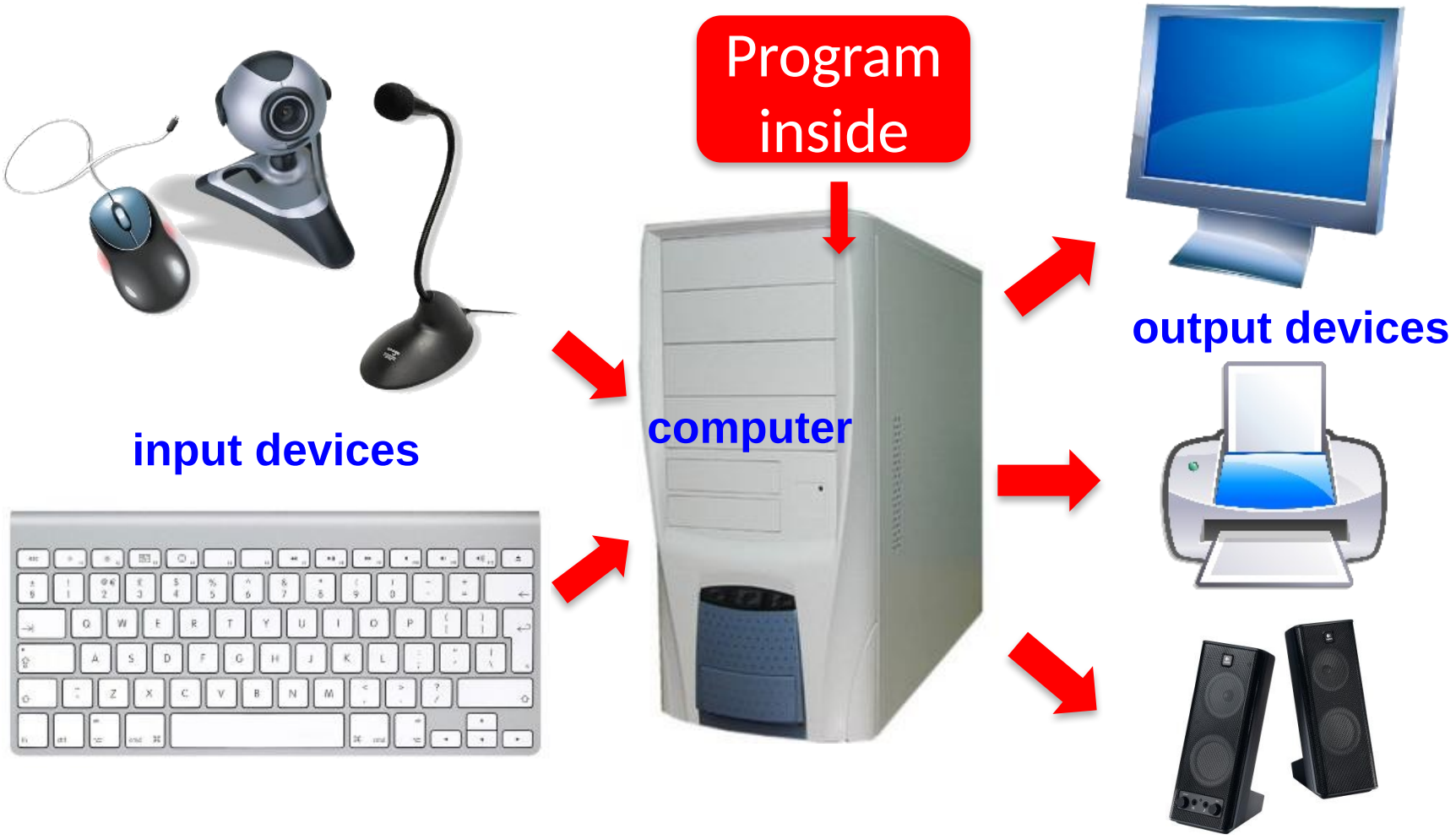
Applied Programming 1

Lab Introduction

Degree on Bioinformatics

Lluís Padró
Computer Science Department – UPC
padro@cs.upc.edu

Interacting with computers

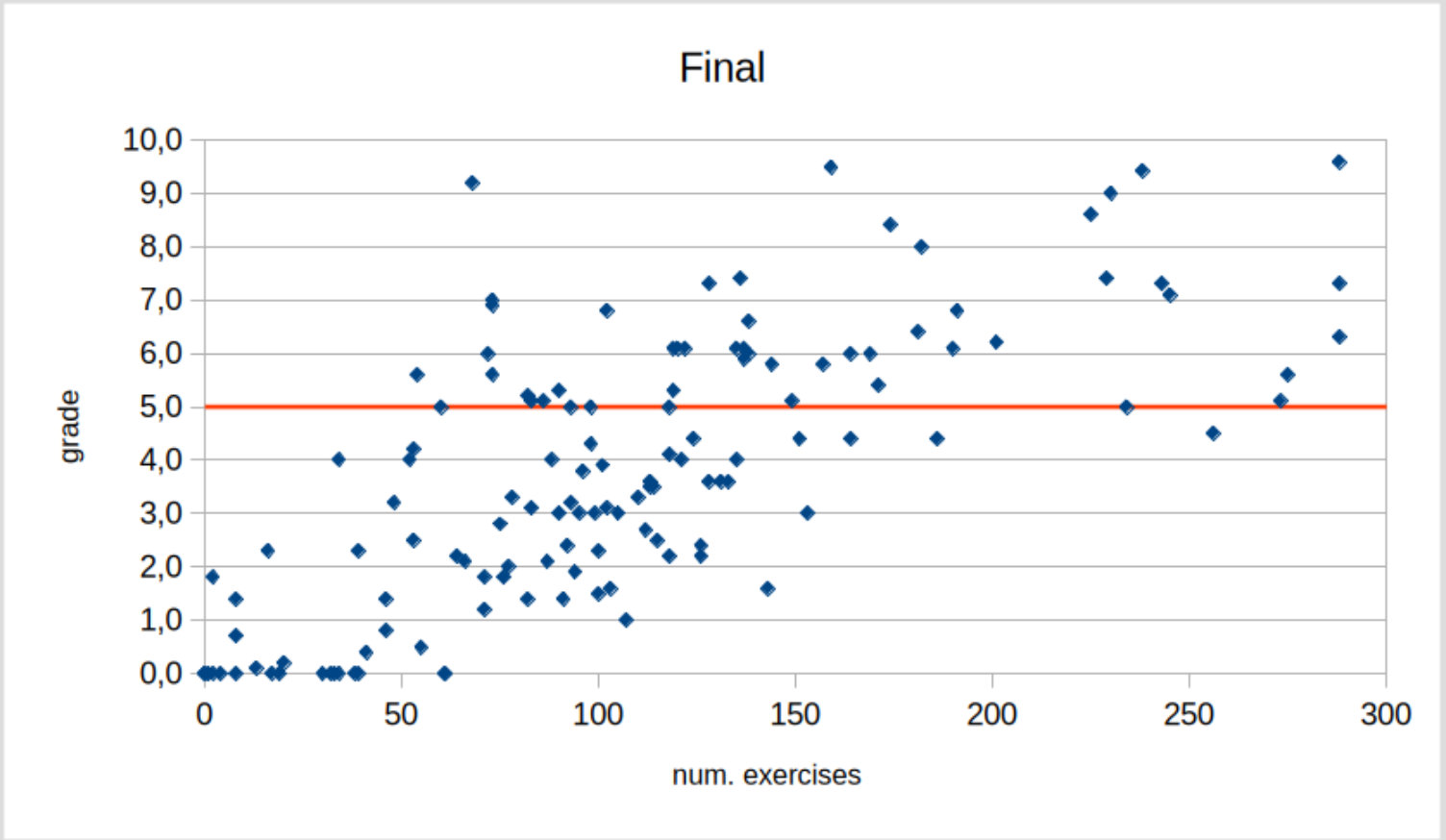


Learning to program

- Programming is a skill which can be learnt, but not taught.
- Many levels of your brain need to be trained to work together:
  - Solve complex tasks combining very simple operations
  - Pay attention to tiny details obvious to humans but not to computers.
  - Understanding of existing algorithms.
  - Combine simpler algorithms into bigger ones.
- Similarly to riding a bicycle, reading music, or speaking a foreign language, the only way of learning to program is by doing. And practicing. **A lot**.

Learning to program

Correlation plot of number of solved exercises vs course grade



#Exercises	Success rate	Avg. Grade
<70	8%	1,4
70-140	35%	3,9
140-200	71%	5,7
>200	93%	7,0

Correlation: 0,73

Python

- Python is an flexible and widely used programming language, allowing
  - Scripting (disposable small on-place code fragments for simple tasks)
  - Structured and object-oriented programming
  - Some functional programming
  - Allows interactive use:

    REPL interpreter – Read-Evaluate-Print Loop

Python program example (1)

```
from yogi import scan

# Ask the user her name and greet her
print("I am a python program, what is your name ?")
name = scan(str)
print(f"Hello {name}, nice to meet you.")
```

Python program example (2)

```
from yogi import scan


# Ask the user to input 2 numbers, then
# do some computations with them
print("Please, enter 2 numbers")
x = scan(int)
y = scan(int)
s = x + y
p = x * y
print("The sum of", x, "and", y, "is", s)
print("and their product is", p)
```

Python program example (3)

```
from yogi import scan


# Ask the user to input 1 numbers, then
# print a short multiplication table
print("Please, enter a number")
n = scan(int)
print(n, "x 1 =", n*1)
print(n, "x 2 =", n*2)
print(n, "x 3 =", n*3)
print(n, "x 4 =", n*4)
print(n, "x 5 =", n*5)
```

Python programs *vs* REPL

- A Python program is written in a text file, with extension `.py`, which can be executed by the interpreter

Myprogram.py

```
from yogi import scan
# Ask the user to input 2 numbers
print("Please, enter 2 numbers")
x = scan(int)
y = scan(int)
# do some computations with the numbers given by the user
s = x + y
p = x * y
# print results to the display for the user to see
print("The sum of", x, "and", y, "is", s)
print("and their product is", p)
```

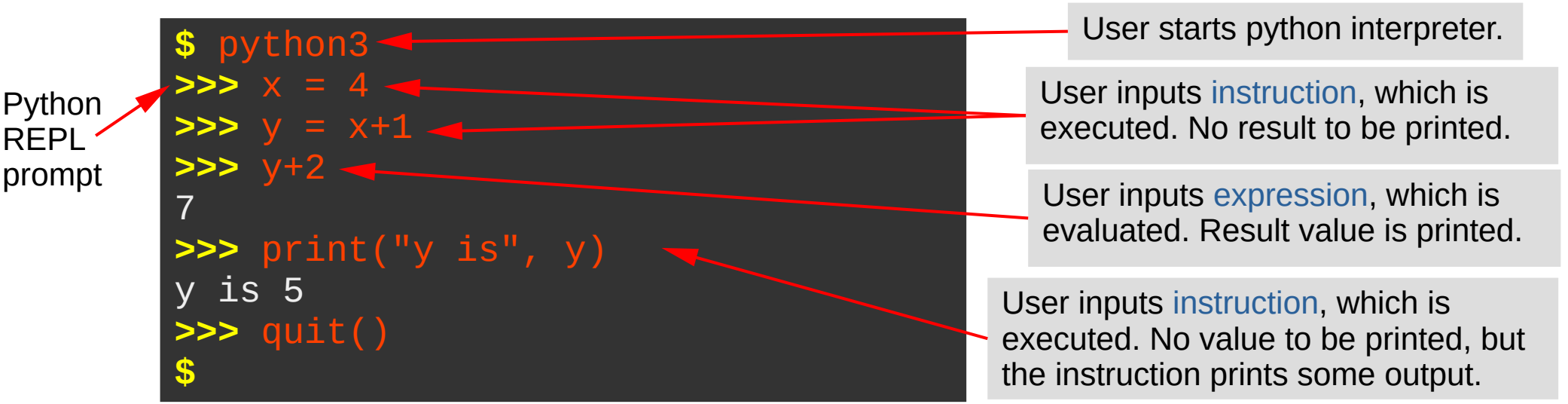
```
$ python3 Myprogram.py
Please, enter 2 numbers
4 6
The sum of 4 and 6 is 10
and their product is 24
$
```

prompt

User input

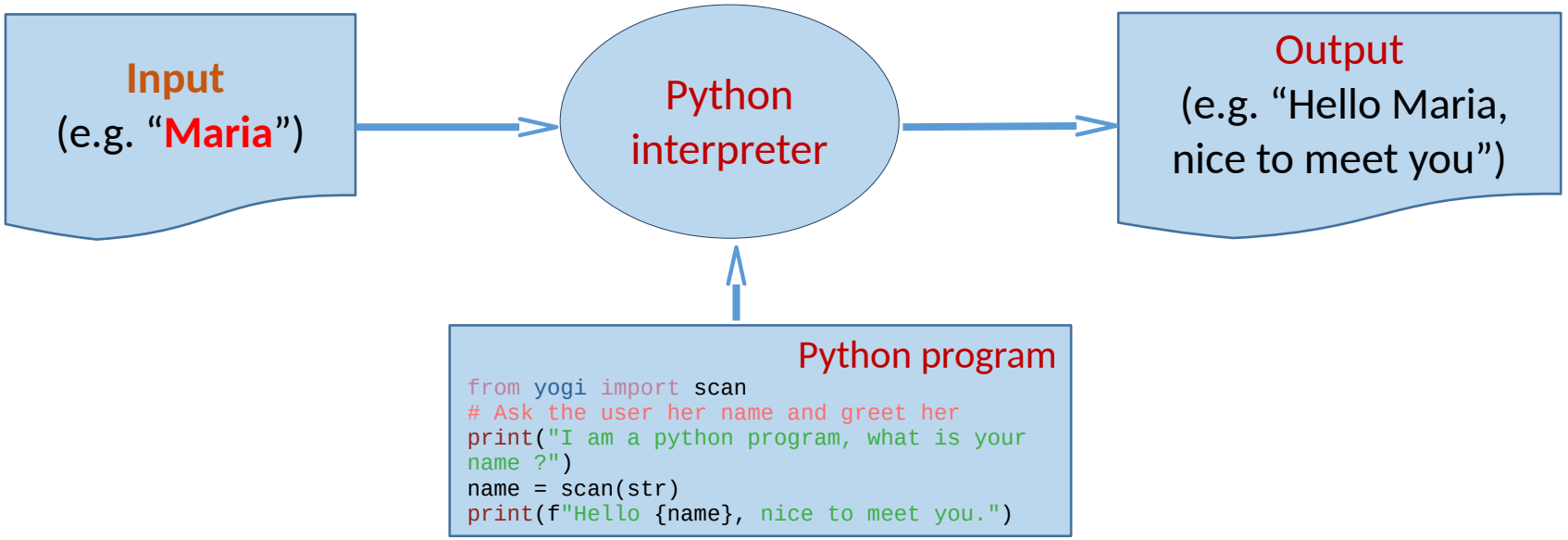
Python programs and REPL

- Python can also be used as a command line instruction interpreter. In this mode, the interpreter follows the Read-Evaluate-Print Loop (REPL)
  - The interpreter reads an instruction/expression from the user.
  - The interpreter evaluates the instruction/expression.
  - If the instruction/expression yields a value, it is printed.
  - The above steps are repeated in a loop.



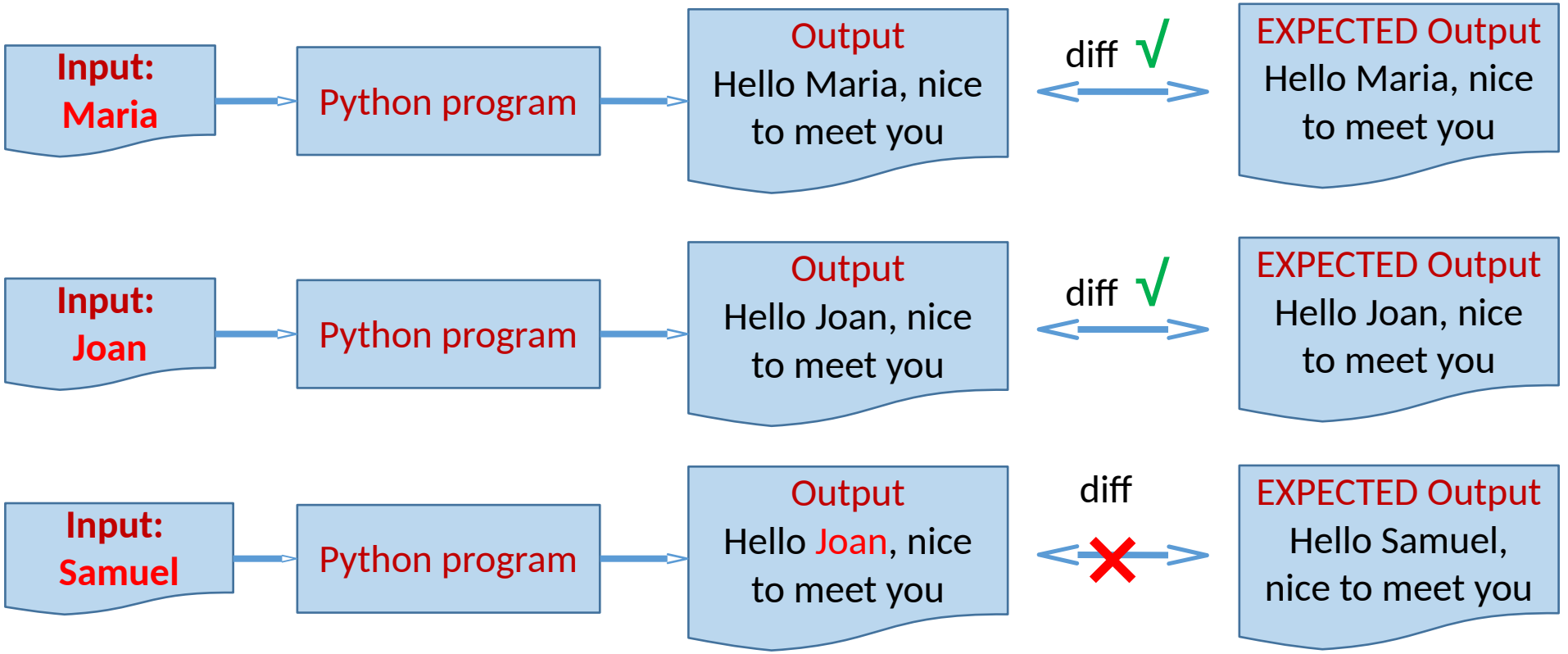
Running python programs

- Computer processors do not understand high level programming languages (Java, Python, C++, etc), which are designed to be closer to humans.
- To be run in a computer, programs need to be in executable (a.k.a. *binary*) form.
- Python is an interpreted language, which means that there is a program (the python interpreter) that reads the code and tells the processor what to do.



Running python programs

- Once the programm is written, we have to check whether it works as expected, comparing the actual output with what we intended to do



Linux

Linux desktop and command line

- In Linux, you have a desktop similar to that of any other O.S.
- Most tasks (copying or renaming a file, moving it to a different folder, create a new folder, etc) can be performed using the graphical desktop interface
- However, we are going to write command-line interface programs, which need to be run in a command line interpreter (also known as console, terminal, or shell)
- From the console, you can run commands to execute any program, or to handle files (copy, rename, move, etc).

Basic shell commands

- A terminal has, in a given moment, one and only one current working directory (i.e. the folder we have currently open).
- Shell commands are always referred to the current working directory

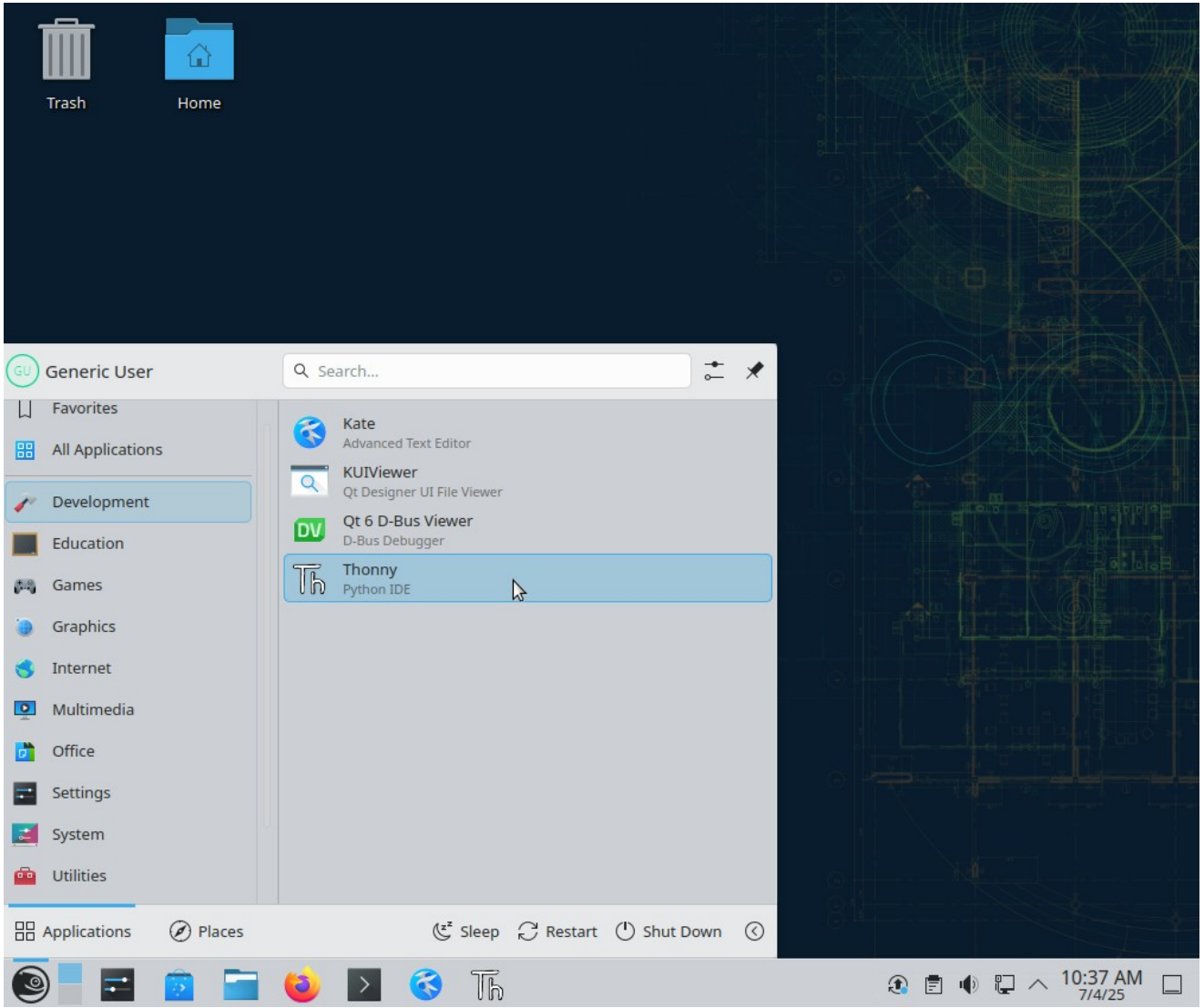
`cd dirname`	Open folder called *dirname*
`cd ..`	Close current folder and go up to parent
`pwd`	Print current working directory
`ls`	List contents of current directory
`mkdir dirname`	Create a new folder named *dirname*
`rmdir dirname`	Remove folder *dirname*
`cp file file2`	Copy *file1* to *file2*
`mv file1 file2`	Rename *file1* to *file2*
`rm file1`	Remove *file1*
`more file1`	Show content of *file1*

Extensive and detailed step-by-step tutorial on shell commands for newbies:
http://linuxcommand.org/

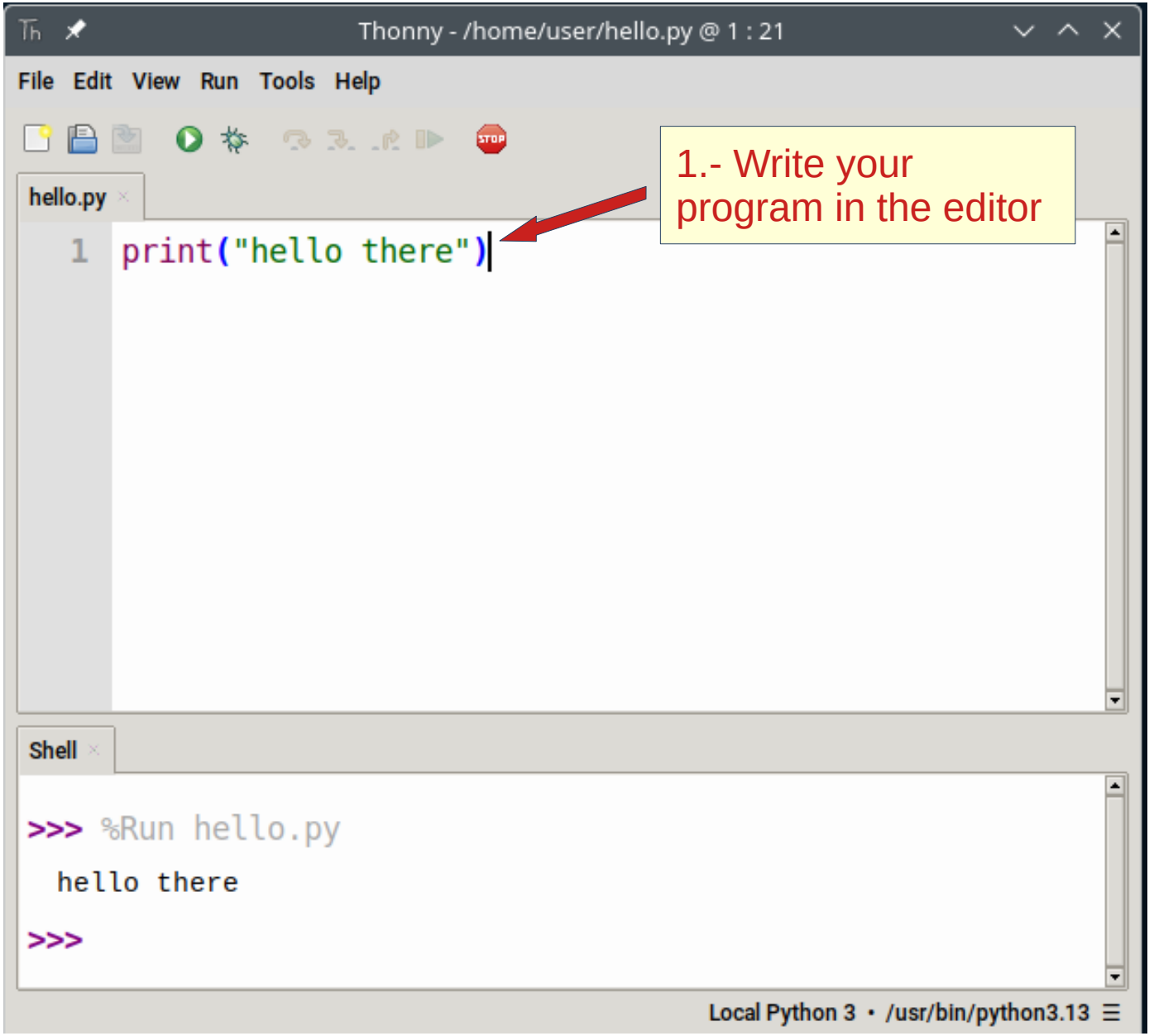
Writing programs

Setting up programming environment

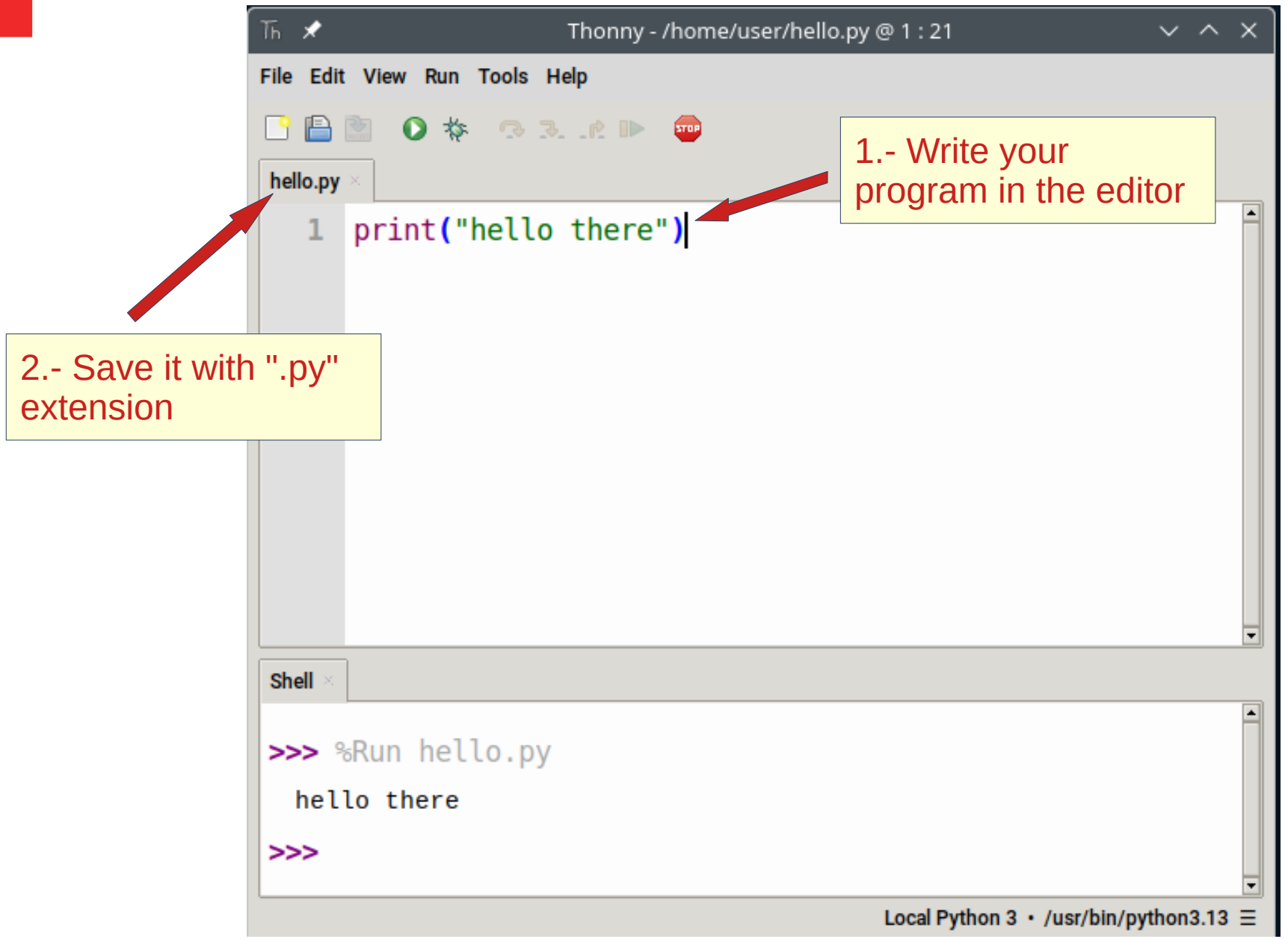
- Programs must be written on a plain text editor
- Linux offers several of them (emacs, kwrite, TextEditor, ...)
- We recommend `thonny`, already available in the provided OpenSuse virtual machine.



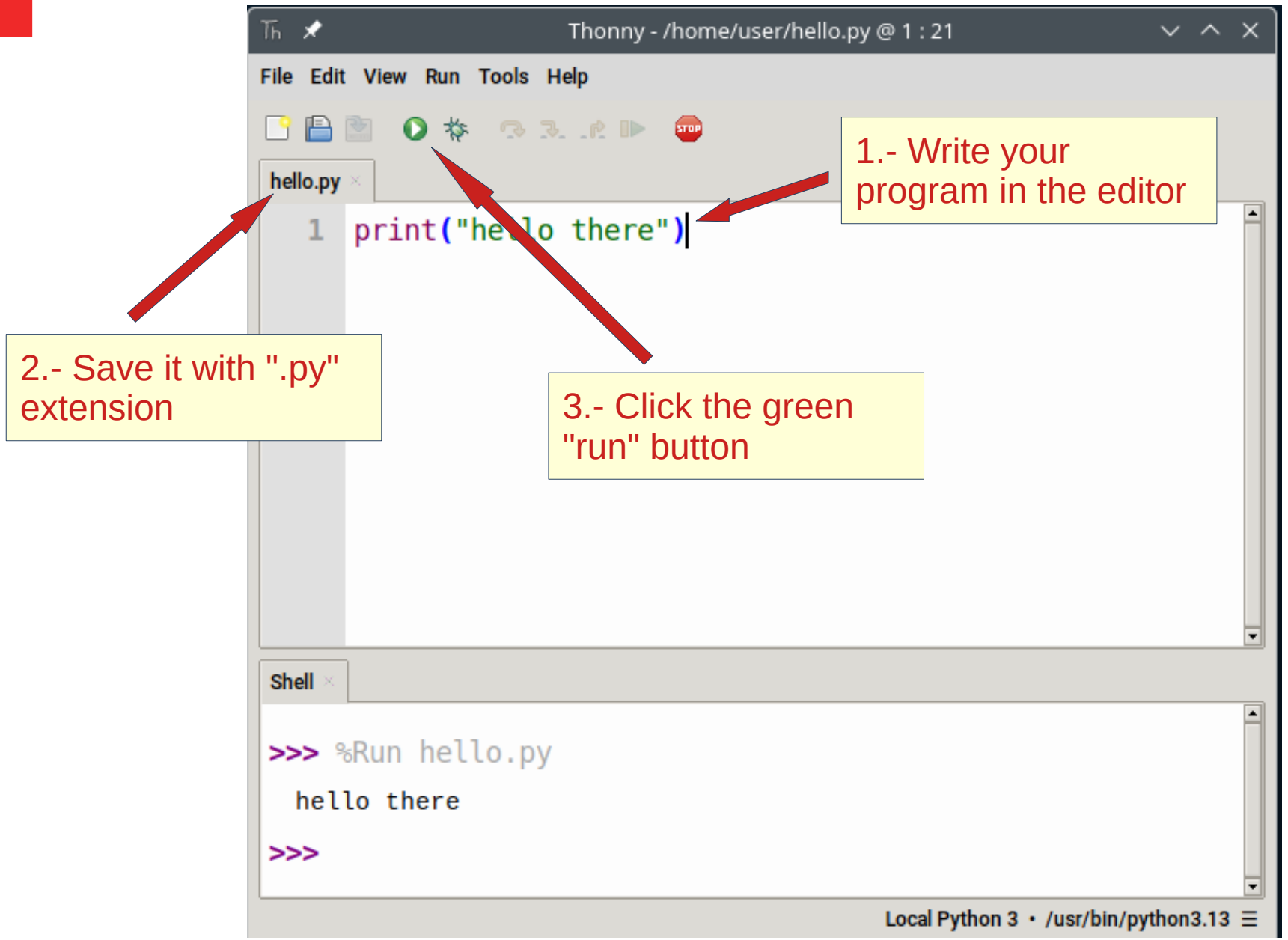
Writing a program



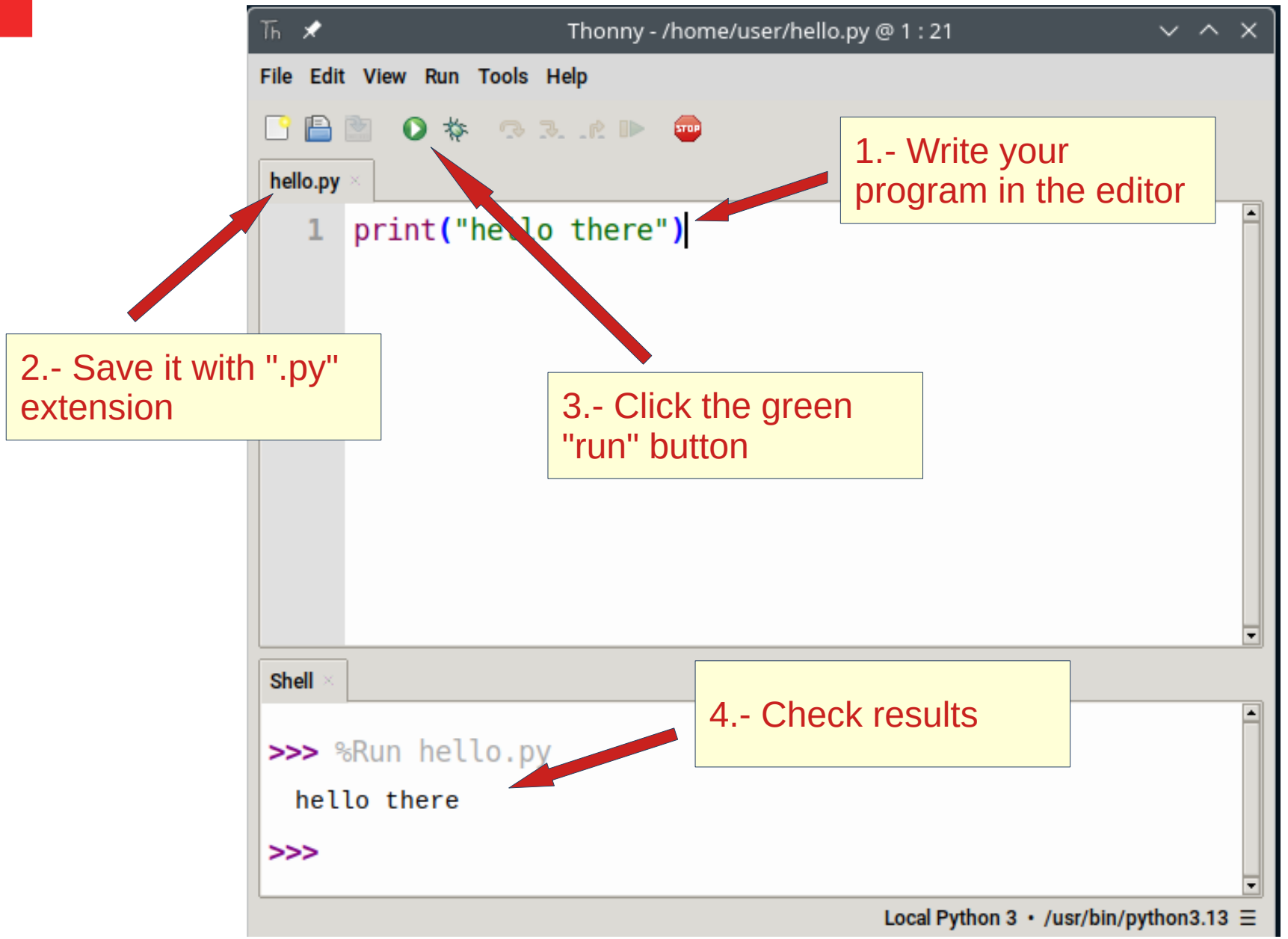
Writing a program



Writing a program



Writing a program



Thonny

- Thonny also allows to run your program in a terminal, so you can type any input.
- Thonny includes a useful *debugger* which will allow you to follow the steps of your program and find errors-

Details to come in Lab sessions

Organize your work

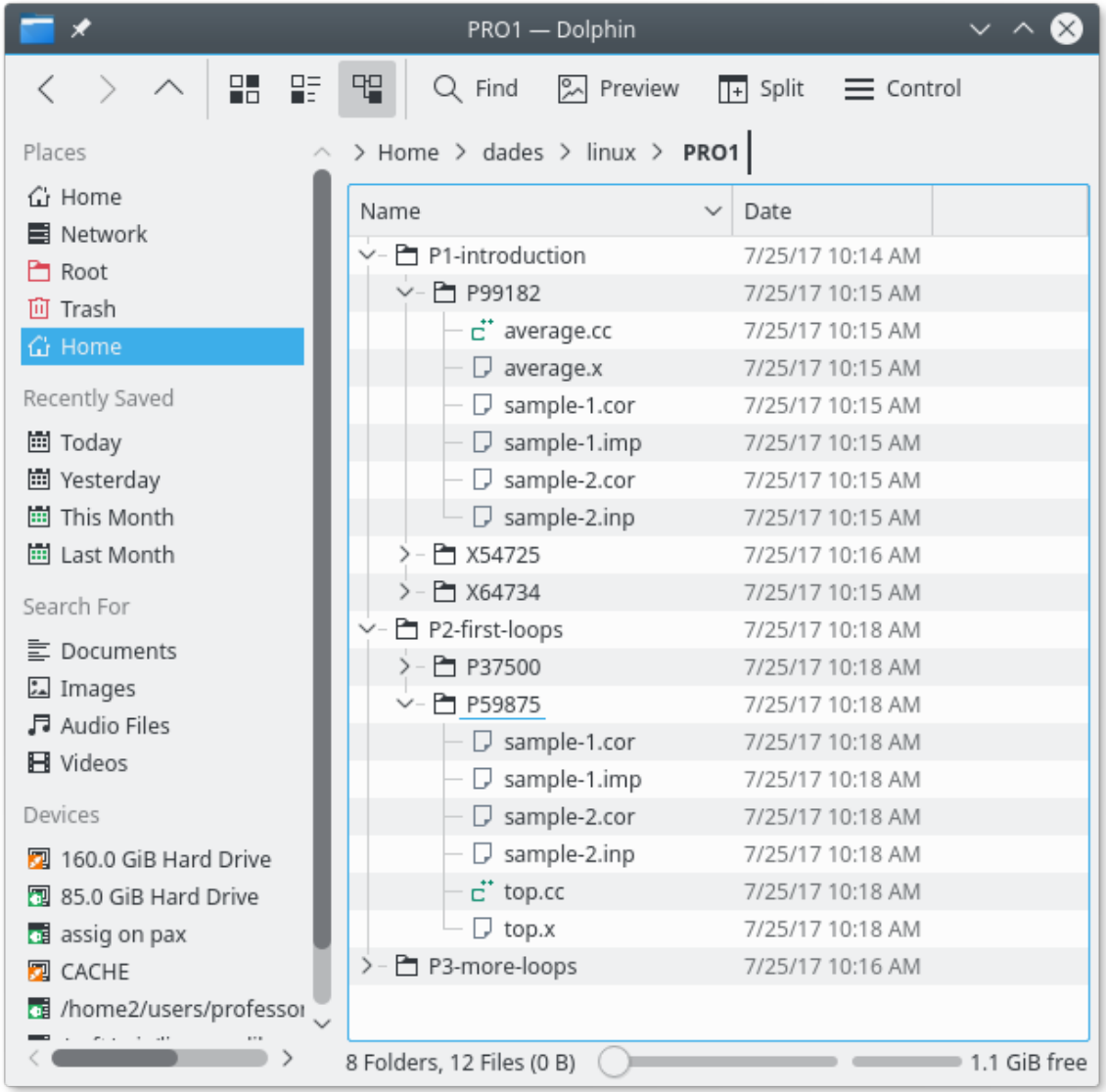
Organize you exercises

- During the course there will be two lab exams.
- There are dozens of exercises in the course lab.
- Exercises are organized in lists, by course chapters.
- Solving more exercises means better practice, and better preparation to face the exams

- It is crucial to have exercises organized to avoid getting lost.

Organize you exercises

- We recommend having a folder for each problem.
- It is also useful to group problem folders depending on the list they belong to.
- In each problem folder, you can have the statement, the python program, and its input and output files.



The *Jutge*

Automatic scoring of programs

- https://jutge.org is the environment where we will test the lab exercises and we will take the course exams.
- You have been invited to this course. Find it in the list, and click “enroll this course”.
- You can submit your programs to the jutge and find out whether they work.
- You can also download the input files and expected outputs for each problem, to check them in your PC.