How to use the tests effectively

Introduction

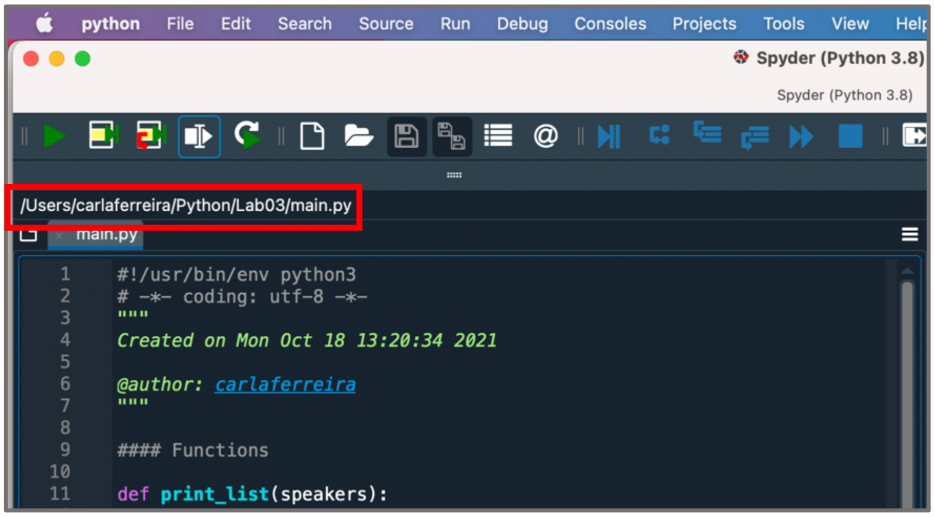
In this curricular unit we use the Mooshak system to validate the programming work. Essentially, Mooshak performs a battery of tests defined by the teaching team for each project and compares the results obtained with each student program with the expected results. If the results are exactly the same, the program passes the tests. If there are some differences, even if apparently insignificant, the program fails the tests. It is intended, with this type of evaluation, to promote the rigor when developing code.

One implication of using this system is that students must be meticulous in assessing the results of their programmes to successfully submit them. A single white space, or a wrong punctuation signal in a message are sufficient for the program to fail the tests. Sometimes the results generated by the programs are relatively extensive. Therefore, it is quite difficult to detect inconsistencies just by visually inspecting results. For this reason, the best approach is to use automatic tools to help us compare the results. This document explains how we can do this.

1. Open a terminal in MacOS or a command line in Windows

2. Locate the folder of your project (Figure 1)

The Spyder IDE shows the complete path for your program, as shown in Figure 1.


python File Edit Search Source Run Debug Consoles Projects Tools View Help
Spyder (Python 3.8)
Spyder (Python 3.8)
/Users/carlaferreira/Python/Lab03/main.py
main.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 18 13:20:34 2021

@author: carlaferreira
"""

#### Functions

def print_list(speakers):
```

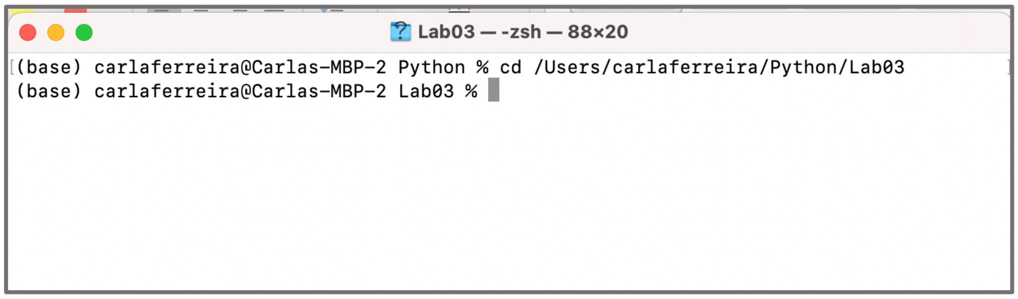
Figure 1 – Spyder indicates the project's path

3. Use the console to change your work context to the folder where you have the project

Use the command **cd** (that means **c**hange **d**irectory) followed from the full path to the folder where you have your project. In the example in Figure 1, the complete command is:

```
$ cd /Users/carlaferreira/Python/Lab03
```

As you can see in Figure 2, the context changes to the folder /Users/carlaferreira/Python/

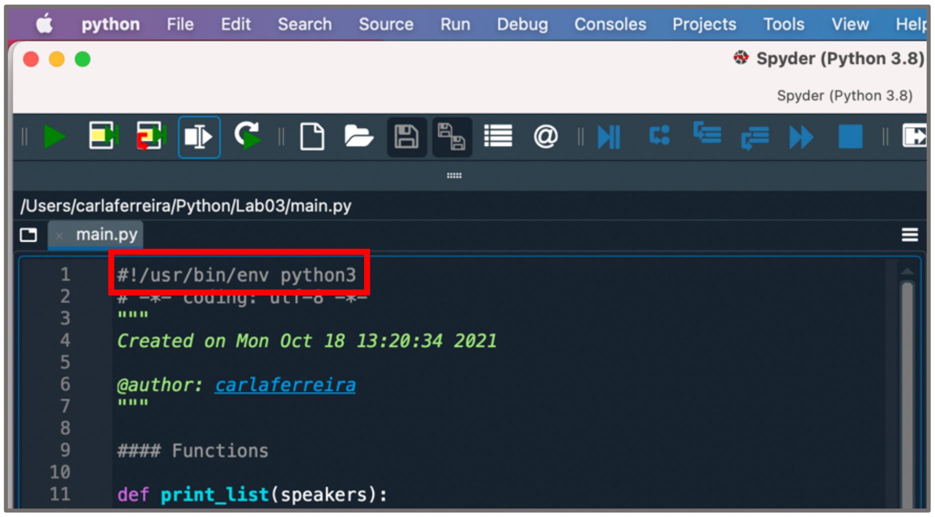


```
(base) carlaferreira@Carlas-MBP-2 Python % cd /Users/carlaferreira/Python/Lab03
(base) carlaferreira@Carlas-MBP-2 Lab03 %
```

Figure 2 – Change of the work context

4. Execute your Python program in the command line

To execute your program in the command line we need to know where the Python interpreter is installed. That information is shown in the automated generated initial comments by the Spyder IDE for each python file (see Figure 3.

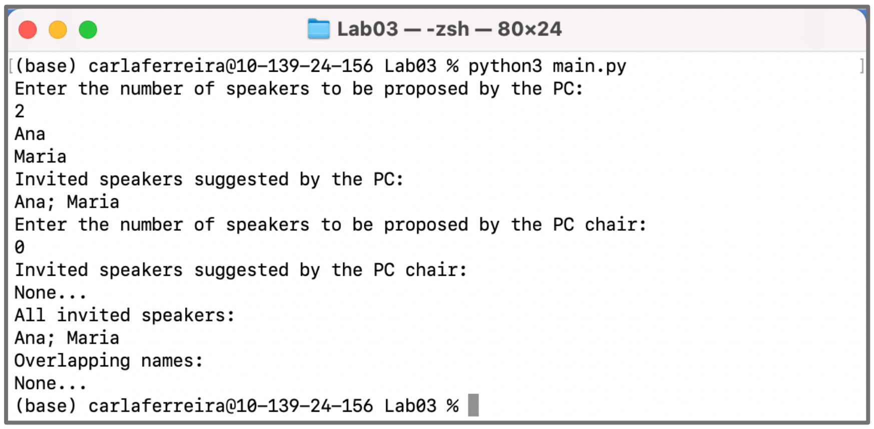

python File Edit Search Source Run Debug Consoles Projects Tools View Help
Spyder (Python 3.8)
Spyder (Python 3.8)
/Users/carlaferreira/Python/Lab03/main.py
main.py

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Oct 18 13:20:34 2021

@author: carlaferreira
"""

#### Functions

def print_list(speakers):
```

Figure 3 – Path to the Python interpreter

The command to execute a program that is stored in file `main.py` is the following:

```
$ /usr/bin/env python3 main.py
```



Lab03 — -zsh — 80×24

```
(base) carlaferreira@10-139-24-156 Lab03 % python3 main.py
Enter the number of speakers to be proposed by the PC:
2
Ana
Maria
Invited speakers suggested by the PC:
Ana; Maria
Enter the number of speakers to be proposed by the PC chair:
0
Invited speakers suggested by the PC chair:
None...
All invited speakers:
Ana; Maria
Overlapping names:
None...
(base) carlaferreira@10-139-24-156 Lab03 %
```

Figure 4 – Executing program main.py in the command line

As you can see in Figure 4, the execution of a program in the command line is similar to its execution within Spyder.

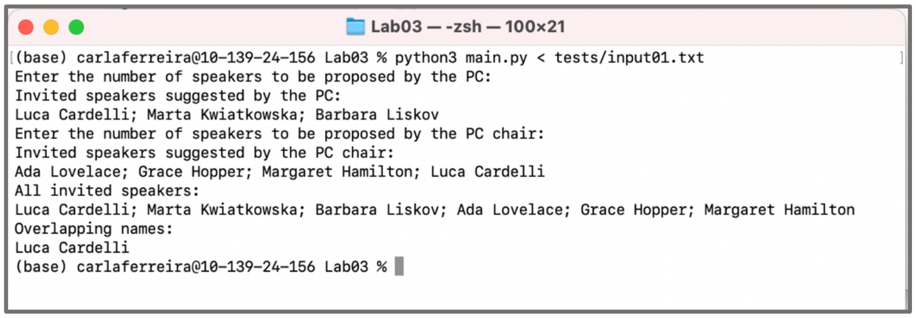
5. Reading the inputs directly from a file

Instead of typing inputs, we can read those inputs directly from a text file. That's exactly what Mooshak will do. Note that in this case we have all the tests stored in a folder called `tests`, which is inside the folder where we have our python program `main.py`. In this example, we will use the first test by adding to our command:

```
< tests\input01.txt
```

This part of the command redirects the contents of `input01.txt` to the console. It's like you're writing your content with your keyboard! But much faster and without mistakes. Try it:

```
$ /usr/bin/env python3 main.py < tests\input01.txt
```



Lab03 — -zsh — 100×21

```
(base) carlaferreira@10-139-24-156 Lab03 % python3 main.py < tests/input01.txt
Enter the number of speakers to be proposed by the PC:
Invited speakers suggested by the PC:
Luca Cardelli; Marta Kwiatkowska; Barbara Liskov
Enter the number of speakers to be proposed by the PC chair:
Invited speakers suggested by the PC chair:
Ada Lovelace; Grace Hopper; Margaret Hamilton; Luca Cardelli
All invited speakers:
Luca Cardelli; Marta Kwiatkowska; Barbara Liskov; Ada Lovelace; Grace Hopper; Margaret Hamilton
Overlapping names:
Luca Cardelli
(base) carlaferreira@10-139-24-156 Lab03 %
```

Figure 5 – Executing main.py with input redirectement

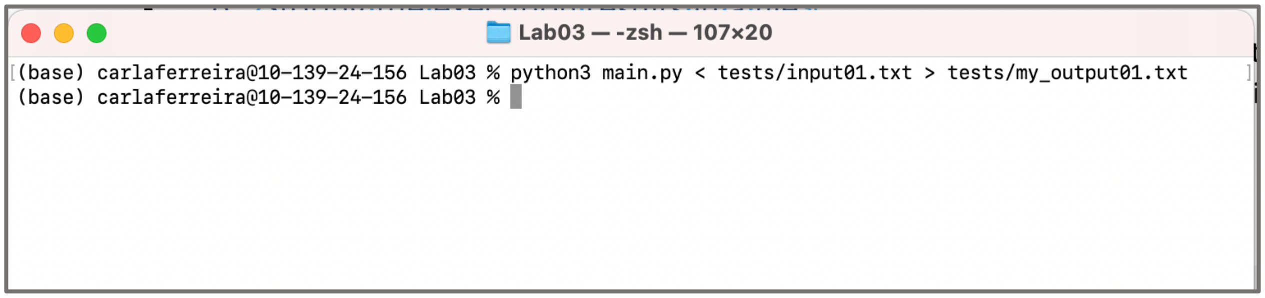
6. Storing the execution results in a file

Notice how test 1 was run and the result shown on the console. The next step is to write these results to a file. Add the redirect to the results file that you want to create to the end of the previous command:

```
> tests\my_output01.txt
```

The command to execute is :

```
$ /usr/bin/env python3 main.py < tests\input01.txt > tests\my_output01.txt
```



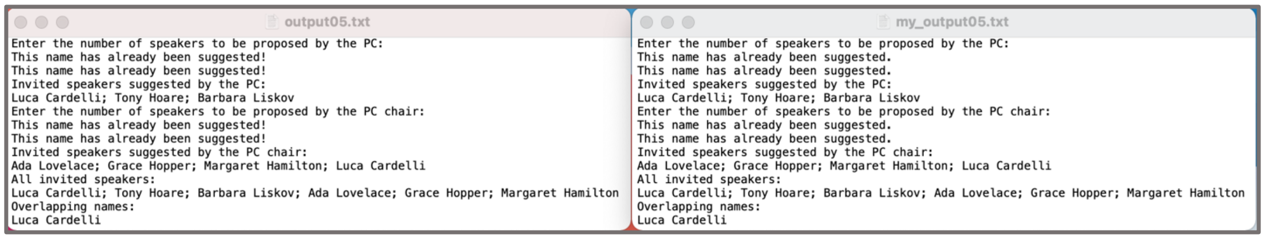
Lab03 — -zsh — 107×20

```
(base) carlaferreira@10-139-24-156 Lab03 % python3 main.py < tests/input01.txt > tests/my_output01.txt
(base) carlaferreira@10-139-24-156 Lab03 %
```

Figure 6 – Program execution with redirection of the output to a file

7. Compare your program result with the expected result

If we open both files and place them side by side, we may detect a small error in our output. Can you find the differences in Figure 7? One of them is easy to find. The other one is harder. We really need a tool to detect these differences.



output05.txt

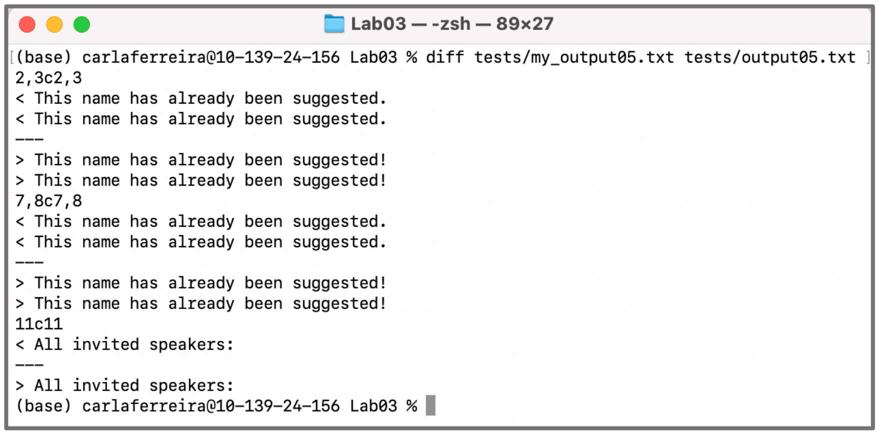
```
Enter the number of speakers to be proposed by the PC:
This name has already been suggested!
This name has already been suggested!
Invited speakers suggested by the PC:
Luca Cardelli; Tony Hoare; Barbara Liskov
Enter the number of speakers to be proposed by the PC chair:
This name has already been suggested!
This name has already been suggested!
Invited speakers suggested by the PC chair:
Ada Lovelace; Grace Hopper; Margaret Hamilton; Luca Cardelli
All invited speakers:
Luca Cardelli; Tony Hoare; Barbara Liskov; Ada Lovelace; Grace Hopper; Margaret Hamilton
Overlapping names:
Luca Cardelli
```

my_output05.txt

```
Enter the number of speakers to be proposed by the PC:
This name has already been suggested.
This name has already been suggested.
Invited speakers suggested by the PC:
Luca Cardelli; Tony Hoare; Barbara Liskov
Enter the number of speakers to be proposed by the PC chair:
This name has already been suggested.
This name has already been suggested.
Invited speakers suggested by the PC chair:
Ada Lovelace; Grace Hopper; Margaret Hamilton; Luca Cardelli
All invited speakers:
Luca Cardelli; Tony Hoare; Barbara Liskov; Ada Lovelace; Grace Hopper; Margaret Hamilton
Overlapping names:
Luca Cardelli
```

Figure 7 – Comparing output files (on the left the expected result, on the right the obtained result)

In Mac OS there is a command that compares the contents of two text files (see Figure 8).

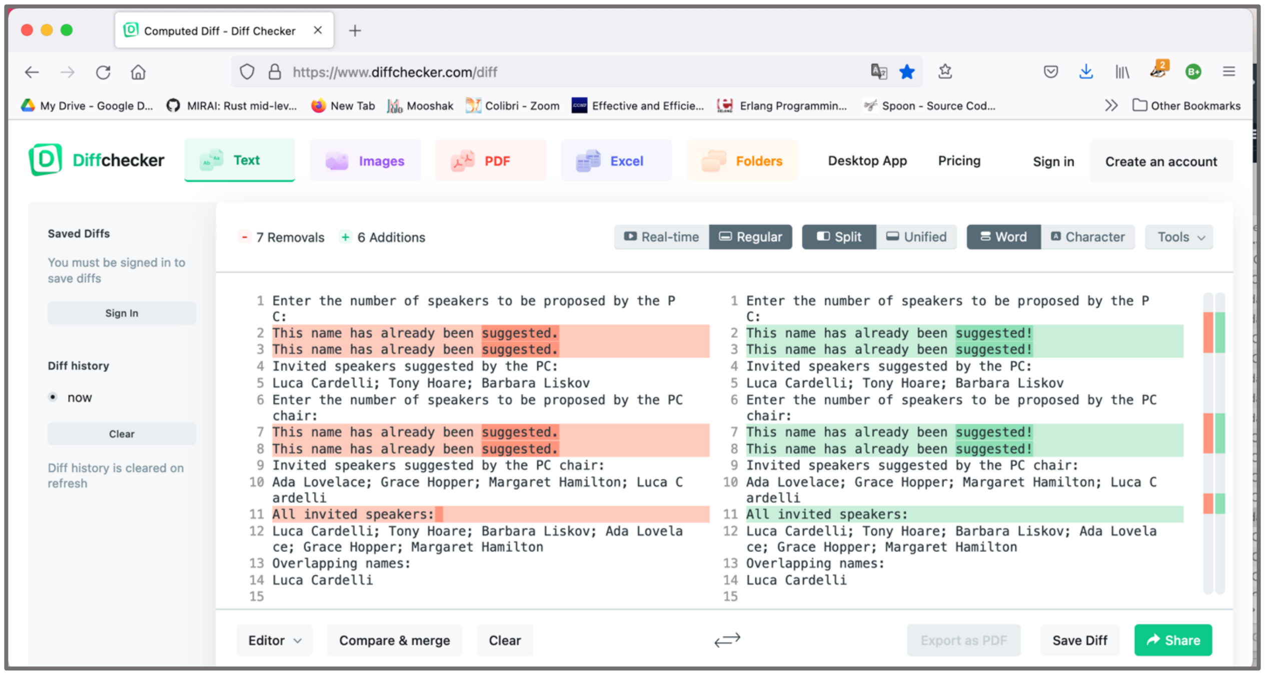


Lab03 — -zsh — 89×27

```
(base) carlaferreira@10-139-24-156 Lab03 % diff tests/my_output05.txt tests/output05.txt
2,3c2,3
< This name has already been suggested.
< This name has already been suggested.
---
> This name has already been suggested!
> This name has already been suggested!
7,8c7,8
< This name has already been suggested.
< This name has already been suggested.
---
> This name has already been suggested!
> This name has already been suggested!
11c11
< All invited speakers:
---
> All invited speakers:
(base) carlaferreira@10-139-24-156 Lab03 %
```

Figure 8 – Using command diff

An alternative is the diffchecker tool, available online https://www.diffchecker.com/. In Figure 9 we have, on the left, the expected result and on the right the available. As you can see, in this case, problems were detected in both rows of the result. In the first case, there was a wrong punctuation signal. On the next line an extra space appears at the end of line 2. This error, in particular, would be difficult to detect without using a tool. Now, you should go fix your program, test and repeat the process until there are no more differences.



```
 1 Enter the number of speakers to be proposed by the P
   C:
 2 This name has already been suggested.
 3 This name has already been suggested.
 4 Invited speakers suggested by the PC:
 5 Luca Cardelli; Tony Hoare; Barbara Liskov
 6 Enter the number of speakers to be proposed by the PC
   chair:
 7 This name has already been suggested.
 8 This name has already been suggested.
 9 Invited speakers suggested by the PC chair:
10 Ada Lovelace; Grace Hopper; Margaret Hamilton; Luca C
   ardelli
11 All invited speakers: 
12 Luca Cardelli; Tony Hoare; Barbara Liskov; Ada Lovela
   ce; Grace Hopper; Margaret Hamilton
13 Overlapping names:
14 Luca Cardelli
15
```

```
 1 Enter the number of speakers to be proposed by the P
   C:
 2 This name has already been suggested!
 3 This name has already been suggested!
 4 Invited speakers suggested by the PC:
 5 Luca Cardelli; Tony Hoare; Barbara Liskov
 6 Enter the number of speakers to be proposed by the PC
   chair:
 7 This name has already been suggested!
 8 This name has already been suggested!
 9 Invited speakers suggested by the PC chair:
10 Ada Lovelace; Grace Hopper; Margaret Hamilton; Luca C
   ardelli
11 All invited speakers:
12 Luca Cardelli; Tony Hoare; Barbara Liskov; Ada Lovela
   ce; Grace Hopper; Margaret Hamilton
13 Overlapping names:
14 Luca Cardelli
15
```

Figure 9 – Using a diff graphical tool to detect differences in files