Python

What is Python?

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic link; make it very attractive of Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python´s simple, easy to learn syntax emphasizes. Python supports module and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Python Test

Getting Started With Testing in Python

A unit test is a smaller test, one that checks that a single component operates in the right way. A unit test helps you to isolate what is broken in your application and fix it faster.

In this report we will begin by explaining how to perform a test with python testing. The operation of the program is very simple; all even numbers between a range of 1000 to 3000 will be shown.

The principal code is:

ejemplo.py - G:\ejemplo.py (3.7.3)

File Edit Format Run Options Window Help

```
#!/usr/bin/python
# -*- coding: utf-8 -*-

def pares(number1,number2):
    if number2 < number1:
        print(f"¡Le he pedido un número entero mayor o igual que {number1}!")
        print("False")
        return False
    else:
        for i in range(number1, number2 + 1):
            if i % 2 == 0:
                print(f"El número {i} es par.")
                print("True")
    return True

pares(1000,3000)
```

Explaining the code:

First we must import the python libraries.

```
ejemplo.py - G:\ejemplo.py (3.7.3)
File  Edit  Format  Run  Options  Window  Help
#!/usr/bin/python
# -*- coding: utf-8 -*-
```

Then we define the variables called number1 and number2.

```
def pares(number1,number2):
```

Since we define the variables, we perform the if statement, where we make the comparison, indicating if the number2 is less than number1, it will show the message on the screen indicating that we need to buy a greater value and not less in the value number2, and return false.

```
    if number2 < number1:
        print(f"¡Le he pedido un número entero mayor o igual que {number1}!")
        print("False")
        return False
```

If it is not this way then it will enter the else, where we assign a for with a value I that will be all the even numbers generated in a range, we generate our process that says the number1 entered and the number2 + 1. If so, the values I it is going to be divided between 2 and it must be == 0, this is the remainder in a division, if it shows zero it means that it is an even number. If everything is correct, it will print all the even numbers between that range, and it will return true.

```
    else:
        for i in range(number1, number2 + 1):
            if i % 2 == 0:
                print(f"El número {i} es par.")
                print("True")
    return True
```

To finish the main class of the code, we declare the vales of the rank that we want to generate their even numbers, mentioning that it will be between 1000 and 3000.

```
pares(1000,3000)
```

Pytest

Pytest supports execution of unittest test cases. The real advantage of pytest comes by writing pytest test cases. Pytest test cases are a series of functions in a Python file starting with the name test_.

Pytest has some other great features:

*Support for filtering for test cases

*Ability to rerun from the last failing test

Writing the test_pares test case example for pytest would look like this:

```
test_pares.py - G:\test_pares.py (3.7.3)
File  Edit  Format  Run  Options  Window  Help
import ejemplo

def test_1():
    response=ejemplo.pares(1000,3000)
    assert response == True

def test_2():
    response=ejemplo.pares(1000,90)
    assert response == False
```

Explaining the code:

First we must import the main class of the code.

```
test_pares.py - G:\test_pares.py (3.7.3)
File  Edit  Format  Run  Options  Window  Help
import ejemplo
```

After we declare the methods for each testing that will be generated, tests are performed for each function that will be executed.

First we define the first test_1 (): that in the main class is when the function is true. In this method it indicates that if the main code (example) we enter from 1000 to 3000. I will be true, because we are indicating the number2 greater than number1.

```
def test_1():
    response=ejemplo.pares(1000,3000)
    assert response == True
```

Then we define the second test_2 (): that in the main class is when the function is false or erroneous to what is asked mainly. In this method it indicates that if the main code (example) we enter a smaller number in the value of number2 as it shows from 1000 to 900. I will be false, because we are indicating the number2 must be greater than the number1.

```
def test_2():
    response=ejemplo.pares(1000,90)
    assert response == False
```

When we enter the pytest to perform the code testing. If the code has an error, it will show the error in the testing. Here the error is that we write the variable incorrectly when ordering it in the print.

```
Símbolo del sistema

C:\Users\Lalo\Desktop>pytest test_pares.py
========================================== test session starts ==========================================
platform win32 -- Python 3.7.2, pytest-4.4.0, py-1.8.0, pluggy-0.9.0
rootdir: C:\Users\Lalo\Desktop
collected 2 items

test_pares.py .F                                                                                    [100%]

================================================= FAILURES =================================================
_____________________________________________ test_2 _____________________________________________

    def test_2():
>       response=ejemplo.pares(1000,90)

test_pares.py:8:
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

number1 = 1000, number2 = 90

    def pares(number1,number2):
        if number2 < number1:
>           print(f"¡Le he pedido un número entero mayor o igual que {numero_1}!")
E           NameError: name 'numero_1' is not defined

ejemplo.py:6: NameError
====================================== 1 failed, 1 passed in 0.59 seconds ======================================
```

Now if we run the program without any errors, we will see that in the testing it already generates the code without any problem.

```
C:\Users\Lalo\Desktop>pytest test_pares.py
========================================== test session starts ==========================================
platform win32 -- Python 3.7.2, pytest-4.4.0, py-1.8.0, pluggy-0.9.0
rootdir: C:\Users\Lalo\Desktop
collected 2 items

test_pares.py ..                                                                                    [100%]

========================================== 2 passed in 0.21 seconds ==========================================
```

Coverage

Coverage.py is a tool for measuring code coverage of Python programs. It monitors your program, noting which parts of the code have been executed, then analyzes the source to identify code that could have been executed but was not.

Coverage measurement is typically used to gauge the effectiveness of tests. It can show which parts of your code are being exercised by tests, and which are not.

There are a few different ways to use coverage.py. The simplest is the command line, which lets you run your program and see the results. If you need more control over how your project is measured, you can use the API.

Some test runners provide coverage integration to make it easy to use coverage.py while running tests. For example, pytest has the pytest-cov plugin.

Executing the coverage:

Since it generates without any error, we can run the coverage to run the program and thus observe all the even numbers between the range that we mentioned in the main code (ejemplo.py), we use the command "coverage run ejemplo.py".

```
========================================== 2 passed in 0.21 seconds ==========================================

C:\Users\Lalo\Desktop>python-coverage run ejemplo.py
"python-coverage" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

C:\Users\Lalo\Desktop>coverage run ejemplo.py
El número 1000 es par.
```

Since we run it we will start to show all the even numbers between the selected range.

python™

Símbolo del sistema

```
C:\Users\Lalo\Desktop>python-coverage run ejemplo.py
"python-coverage" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

C:\Users\Lalo\Desktop>coverage run ejemplo.py
El número 1000 es par.
True
El número 1002 es par.
True
El número 1004 es par.
True
El número 1006 es par.
True
El número 1008 es par.
True
El número 1010 es par.
True
El número 1012 es par.
True
El número 1014 es par.
True
El número 1016 es par.
True
El número 1018 es par.
True
El número 1020 es par.
True
El número 1022 es par.
True
El número 1024 es par.
True
El número 1026 es par.
True
El número 1028 es par.
True
El número 1030 es par.
True
El número 1032 es par.
True
El número 1034 es par.
True
El número 1036 es par.
```

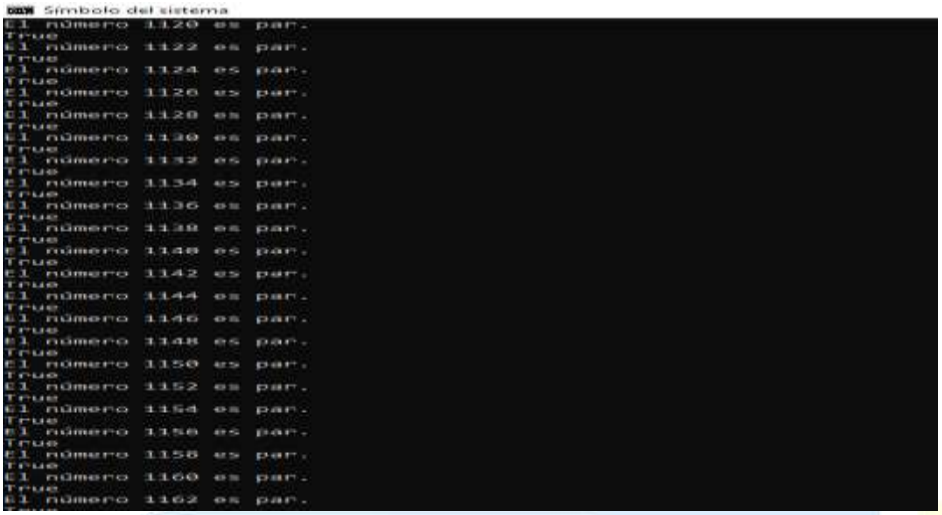
python™

Símbolo del sistema

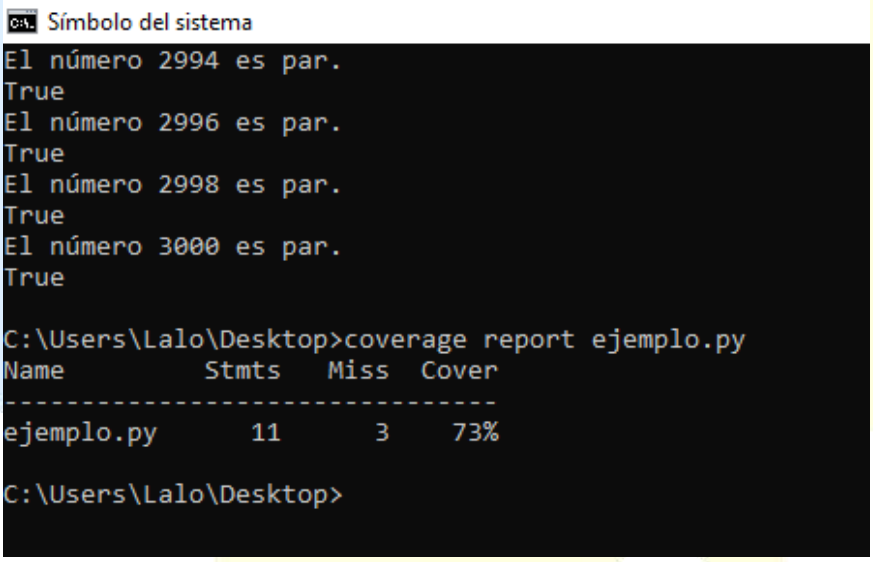
```
True
El número 1036 es par.
True
El número 1038 es par.
True
El número 1040 es par.
True
El número 1042 es par.
True
El número 1044 es par.
True
El número 1046 es par.
True
El número 1048 es par.
True
El número 1050 es par.
True
El número 1052 es par.
True
El número 1054 es par.
True
El número 1056 es par.
True
El número 1058 es par.
True
El número 1060 es par.
True
El número 1062 es par.
True
El número 1064 es par.
True
El número 1066 es par.
True
El número 1068 es par.
True
El número 1070 es par.
True
El número 1072 es par.
True
El número 1074 es par.
True
El número 1076 es par.
True
El número 1078 es par.
```

Símbolo del sistema

```
True
El número 1078 es par.
True
El número 1080 es par.
True
El número 1082 es par.
True
El número 1084 es par.
True
El número 1086 es par.
True
El número 1088 es par.
True
El número 1090 es par.
True
El número 1092 es par.
True
El número 1094 es par.
True
El número 1096 es par.
True
El número 1098 es par.
True
El número 1100 es par.
True
El número 1102 es par.
True
El número 1104 es par.
True
El número 1106 es par.
True
El número 1108 es par.
True
El número 1110 es par.
True
El número 1112 es par.
True
El número 1114 es par.
True
El número 1116 es par.
True
El número 1118 es par.
True
El número 1120 es par.
```



Already when it generates all the even numbers between the range of 1000 to 3000, and finally shows us the performance that the program had in a table, the cover was 73%.



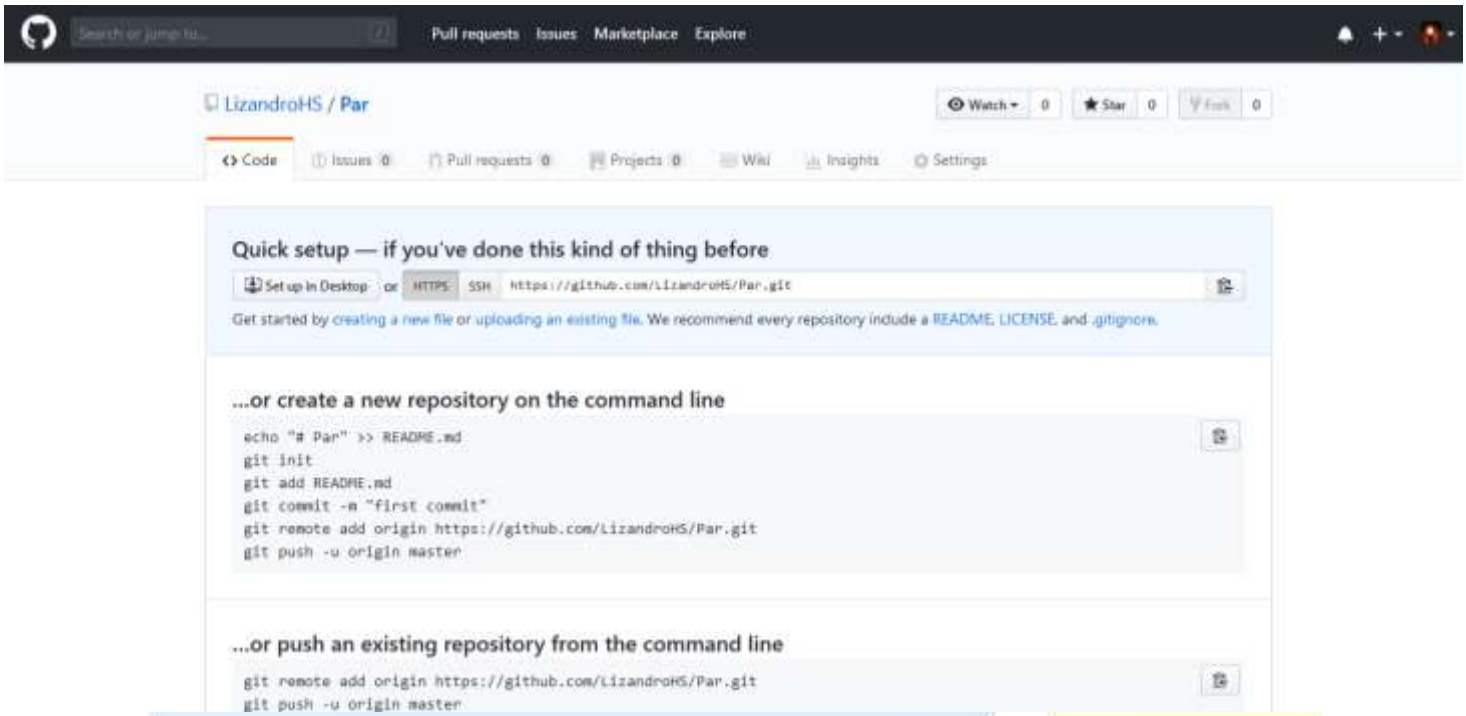
GitHub

To upload your projects to GitHub you must follow the following steps:

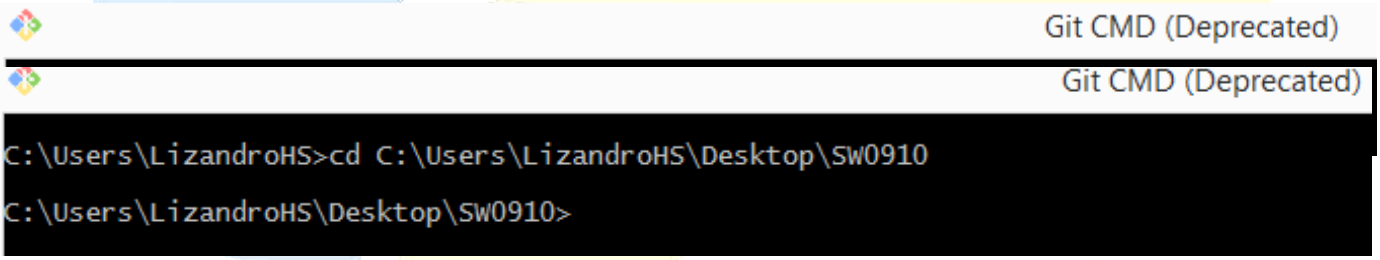
1. have or create an account on GitHub.
2. have the Git program installed.

3. Create a new project on GitHub.

python™

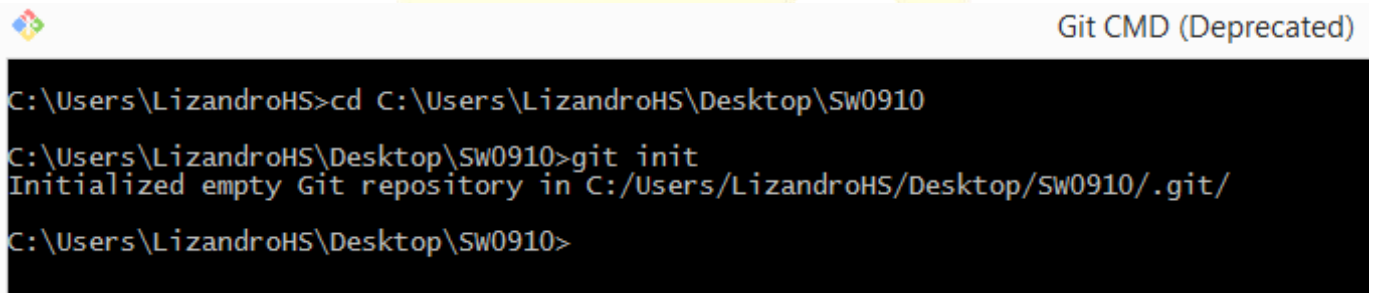


4. Enter the Git CMD console.



5. we are going to position ourselves within the folder of our project with the **cd** command after we enter the **project directory**.

6. initialize the repository with the **git init** command.



7. To add all the files of the project to the repository we execute the command **git add -A.**

Git CMD (Deprecated)

```
C:\Users\LizandroHS>cd C:\Users\LizandroHS\Desktop\SW0910

C:\Users\LizandroHS\Desktop\SW0910>git init
Initialized empty Git repository in C:/Users/LizandroHS/Desktop/SW0910/.git/

C:\Users\LizandroHS\Desktop\SW0910>git add -A

C:\Users\LizandroHS\Desktop\SW0910>
```

8. To see what is added to the repository, **git status** is executed

Git CMD (Deprecated)

```
C:\Users\LizandroHS>cd C:\Users\LizandroHS\Desktop\SW0910

C:\Users\LizandroHS\Desktop\SW0910>git init
Initialized empty Git repository in C:/Users/LizandroHS/Desktop/SW0910/.git/

C:\Users\LizandroHS\Desktop\SW0910>git add -A

C:\Users\LizandroHS\Desktop\SW0910>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   Python.docx
        new file:   ejemplo.py
        new file:   test_pares.py


C:\Users\LizandroHS\Desktop\SW0910>
```

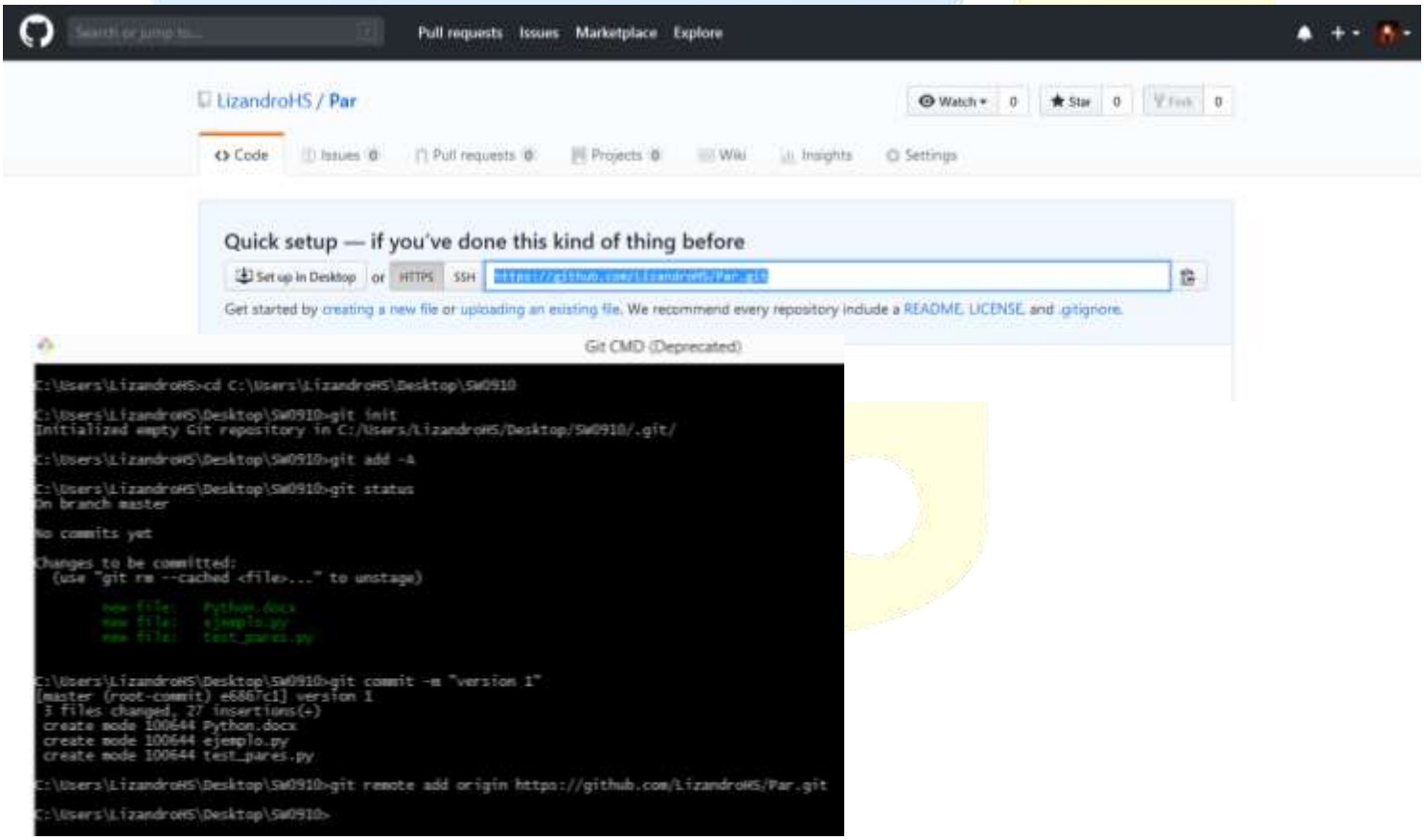
9. To initialize the local version of the project we execute the command **git commit -m "vesion 1".**

python™

Git CMD (Deprecated)

```
C:\Users\LizandroHS>cd C:\Users\LizandroHS\Desktop\SW0910

C:\Users\LizandroHS\Desktop\SW0910>git init
Initialized empty Git repository in C:/Users/LizandroHS/Desktop/SW0910/.git/

C:\Users\LizandroHS\Desktop\SW0910>git add -A

C:\Users\LizandroHS\Desktop\SW0910>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   Python.docx
        new file:   ejemplo.py
        new file:   test_pares.py


C:\Users\LizandroHS\Desktop\SW0910>git commit -m "version 1"
[master (root-commit) e6867c1] version 1
 3 files changed, 27 insertions(+)
 create mode 100644 Python.docx
 create mode 100644 ejemplo.py
 create mode 100644 test_pares.py

C:\Users\LizandroHS\Desktop\SW0910>
```

10. to assign the repository to the project we execute the command **git remote add origin** and copy and paste the **link** of the project that we created in GitHub.

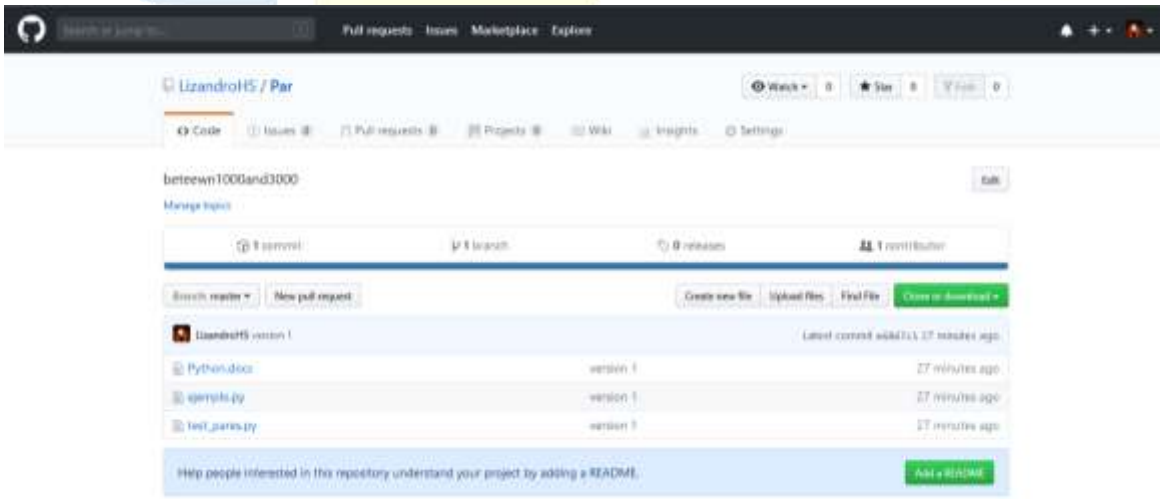


11. To make the copy in GitHub we execute the command **git push origin master.**

Git CMD (Deprecated)

```
C:\Users\LizandroHS>cd C:\Users\LizandroHS\Desktop\SW0910

C:\Users\LizandroHS\Desktop\SW0910>git init
Initialized empty Git repository in C:/Users/LizandroHS/Desktop/SW0910/.git/

C:\Users\LizandroHS\Desktop\SW0910>git add -A

C:\Users\LizandroHS\Desktop\SW0910>git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   Python.docx
        new file:   ejemplo.py
        new file:   test_pares.py


C:\Users\LizandroHS\Desktop\SW0910>git commit -m "version 1"
[master (root-commit) e6867c1] version 1
 3 files changed, 27 insertions(+)
 create mode 100644 Python.docx
 create mode 100644 ejemplo.py
 create mode 100644 test_pares.py

C:\Users\LizandroHS\Desktop\SW0910>git remote add origin https://github.com/LizandroHS/Par.git

C:\Users\LizandroHS\Desktop\SW0910>git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (5/5), 696.80 KiB | 11.06 MiB/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/LizandroHS/Par.git
 * [new branch]      master -> master

C:\Users\LizandroHS\Desktop\SW0910>
```

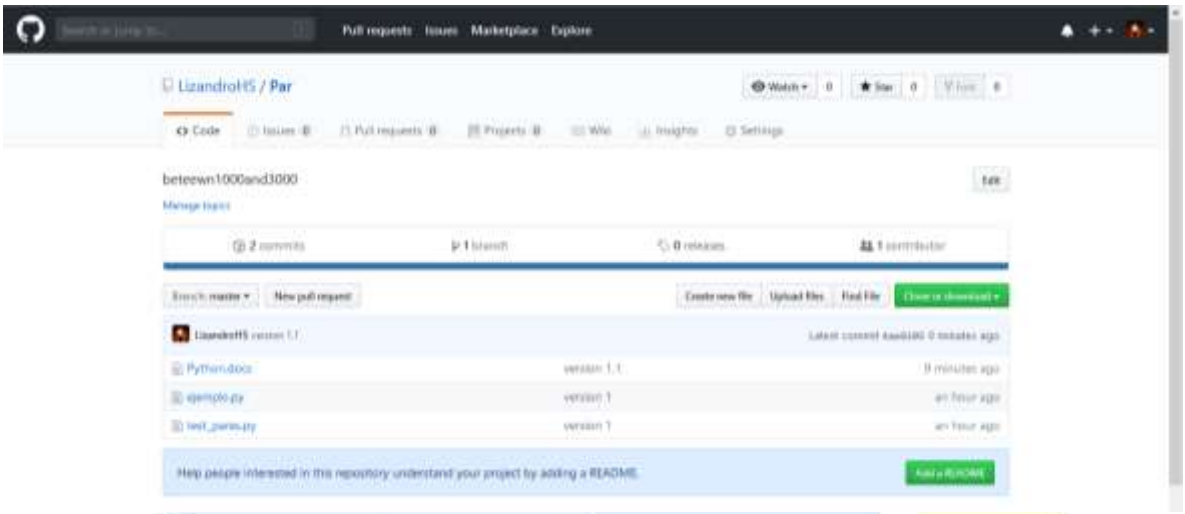
12. In the end we only refresh our GitHub project.



13. To make updates you must repeat the commands **git add -A**, **git commit -m "version n"** and finally **git push origin master**.

Git CMD (Deprecated)

```
On branch master
nothing to commit, working tree clean

C:\Users\LizandroHS\Desktop\SW0910>git status
On branch master
nothing to commit, working tree clean

C:\Users\LizandroHS\Desktop\SW0910>git add -A

C:\Users\LizandroHS\Desktop\SW0910>git status
On branch master
nothing to commit, working tree clean

C:\Users\LizandroHS\Desktop\SW0910>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   Python.docx

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\LizandroHS\Desktop\SW0910>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   Python.docx

no changes added to commit (use "git add" and/or "git commit -a")

C:\Users\LizandroHS\Desktop\SW0910>git add -A

C:\Users\LizandroHS\Desktop\SW0910>git commit -m "version 1.1"
[master 4ae8105] version 1.1
 1 file changed, 0 insertions(+), 0 deletions(-)

C:\Users\LizandroHS\Desktop\SW0910>git push origin master
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 1.72 MiB | 1.47 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/LizandroHS/Par.git
   e6867c1..4ae8105  master -> master

C:\Users\LizandroHS\Desktop\SW0910>
```



https://github.com/LizandroHS/Par.git

Team:
GUTIERREZ SERVIN GUADALUPE ELIZABETH
ESCANDON RODRIGUEZ JESSICA NAYELI
HERNANDEZ SUAREZ LIZANDRO