

# Práctico PP1 2021

Crear un programa en Python que tendrá 3 o 4 funciones:

- Suma, 3 parámetros, devuelve la suma de los 3.
- Resta, 2 parámetros, devuelve la resta de los 2.
- Producto, 4 parámetros, devuelve el producto de los 4.
- Imprimir, 2 parámetros, texto y valor, devuelve la impresión de los valores pasados.

Usaremos Visual Studio Code, con la extensión de Python para hacer la práctica.

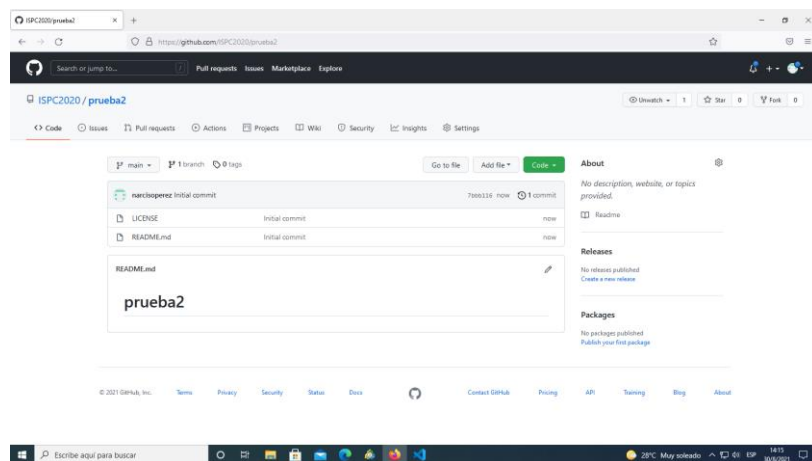
El equipo crea un Project en Github y agrega 2 al menos 2 issue por cada tarea a realizar.

Crear una rama por cada miembro.

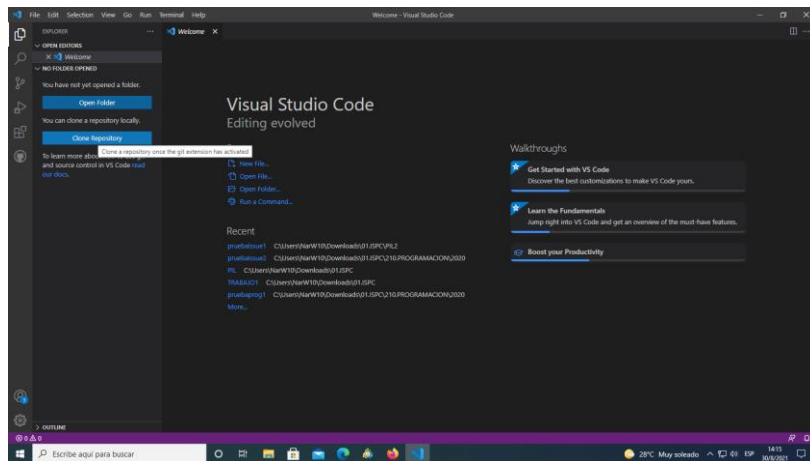
El primer integrante del grupo, creará el repositorio, creará un programa en Python con la función suma, y el main y subirá al repo el código producido

## Ejemplo de primer ciclo:

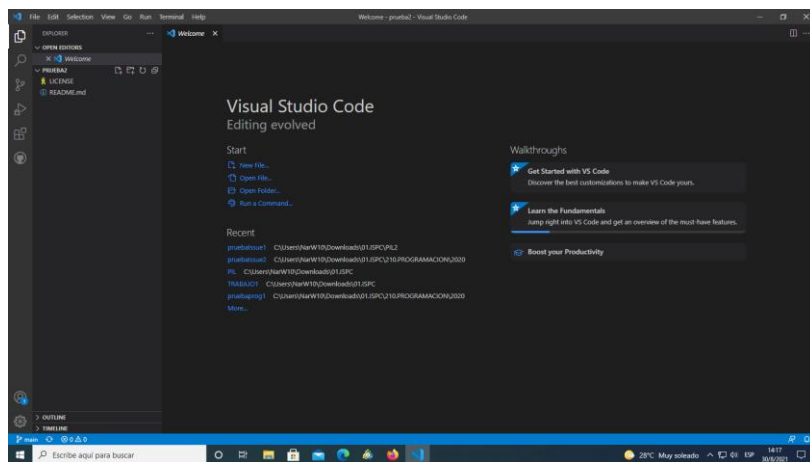
Crear repo, sincronizar remoto y local.



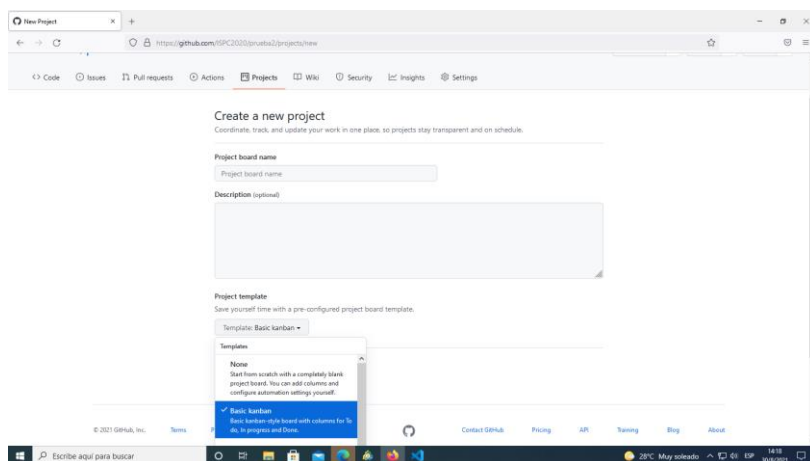
Clonar repo remoto



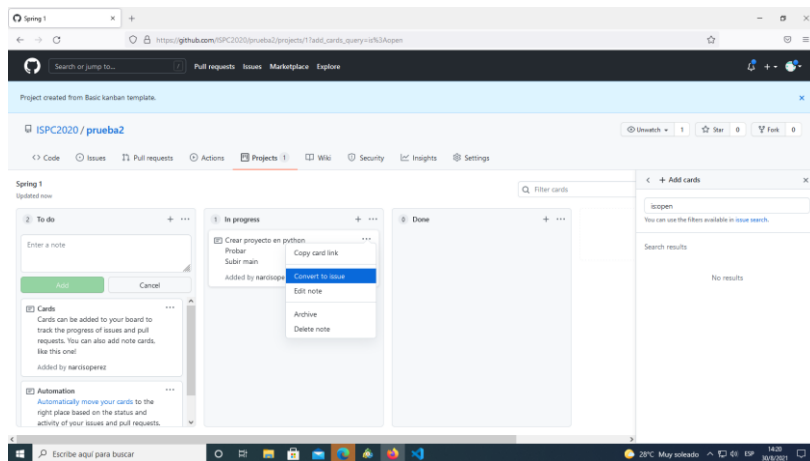
Remoto y local sincronizados.



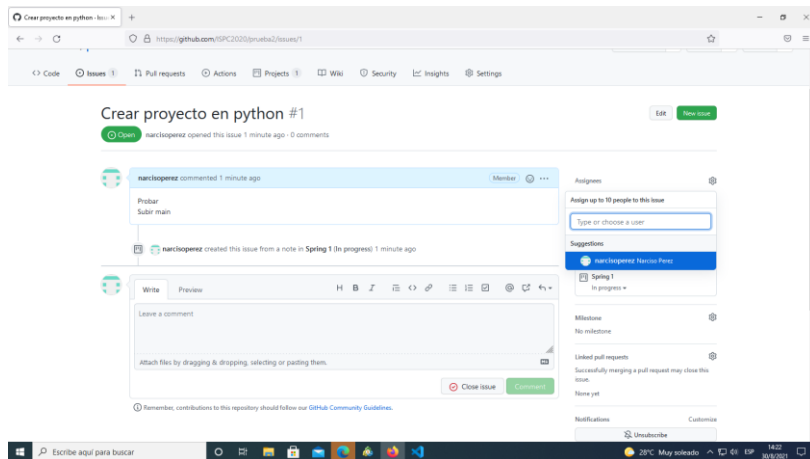
Cargar en Project las tareas.



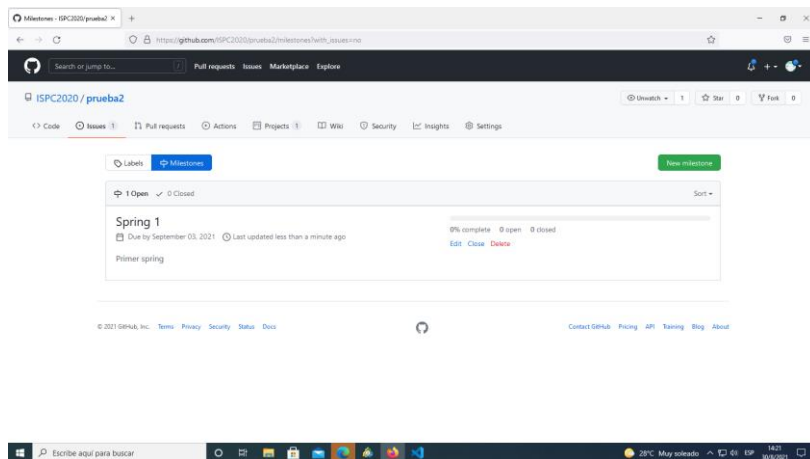
Cargar canvan en project. Crear issue de las tareas.

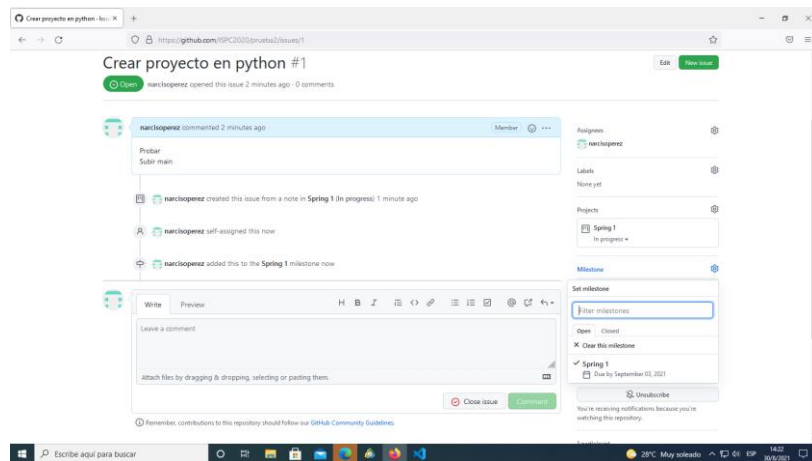


Asignar a miembro de equipo.



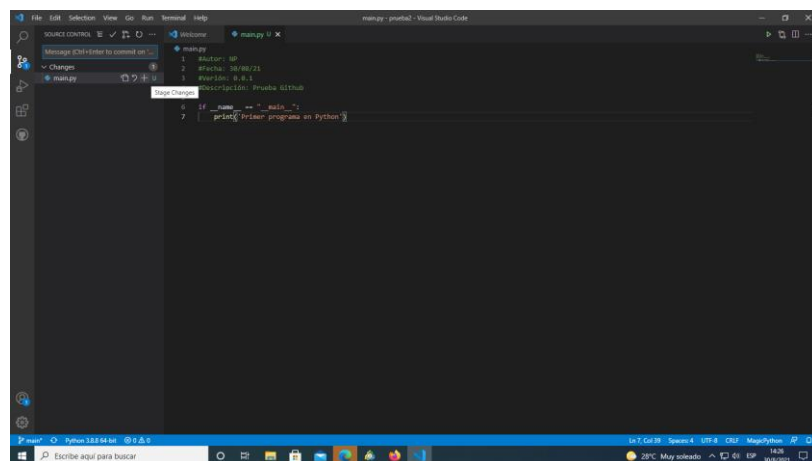
Crear un milestone.



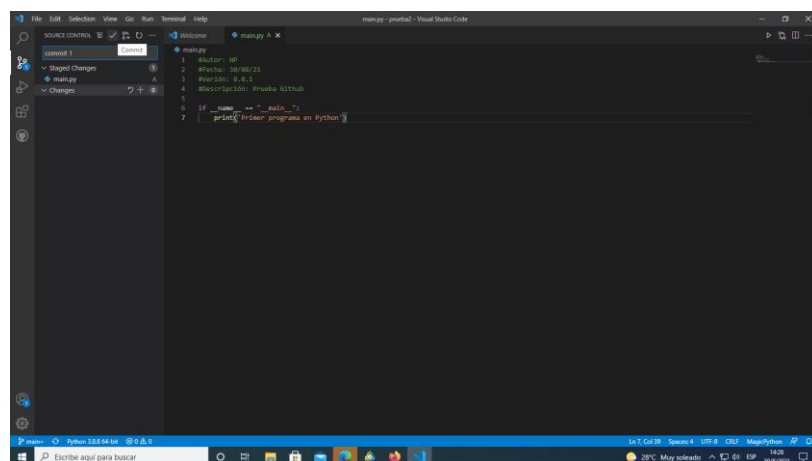


Programar estructura de programa con main. Poner un print y probar.

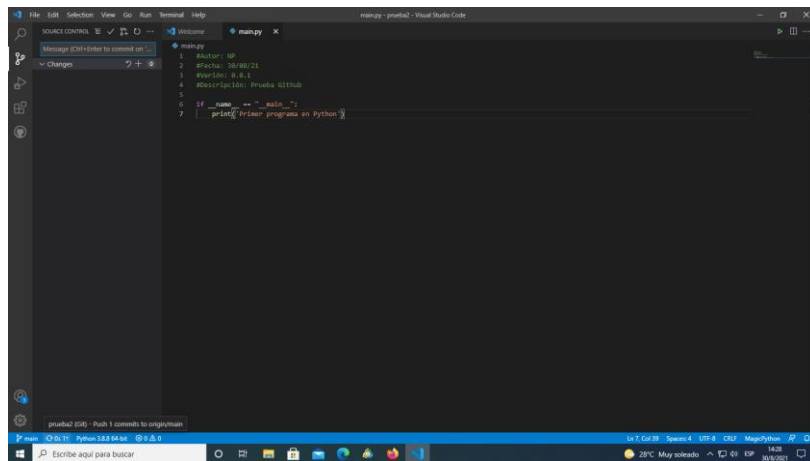
Hacer un “add”



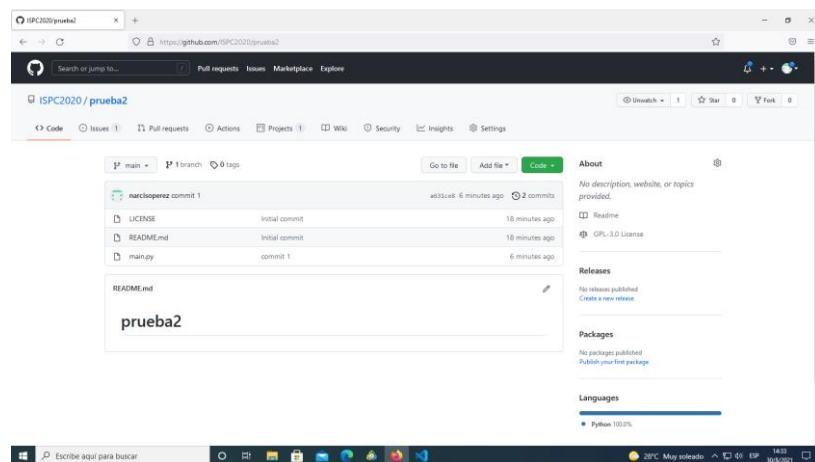
Hacer un “commit”



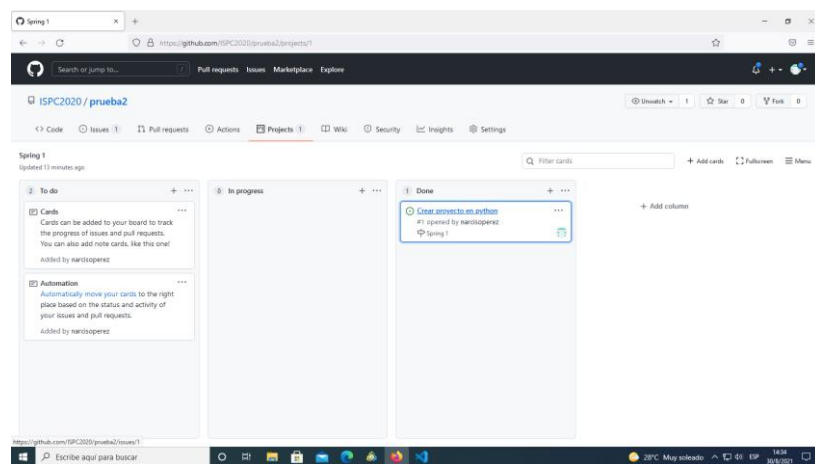
Hacer un “push”



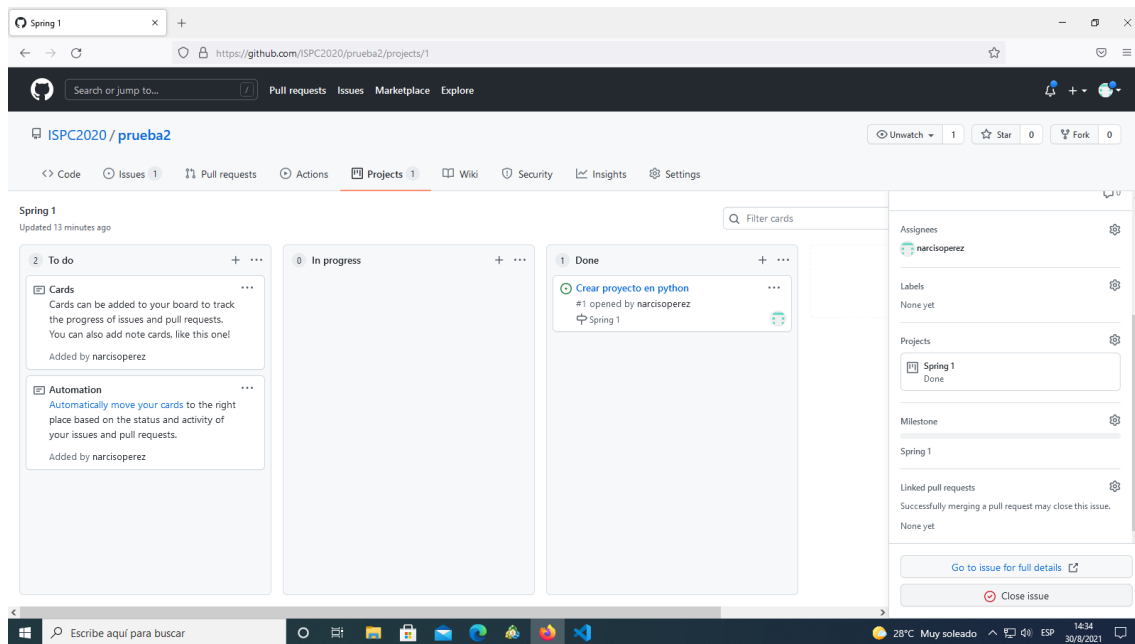
Verificar remoto actualizado con main.py



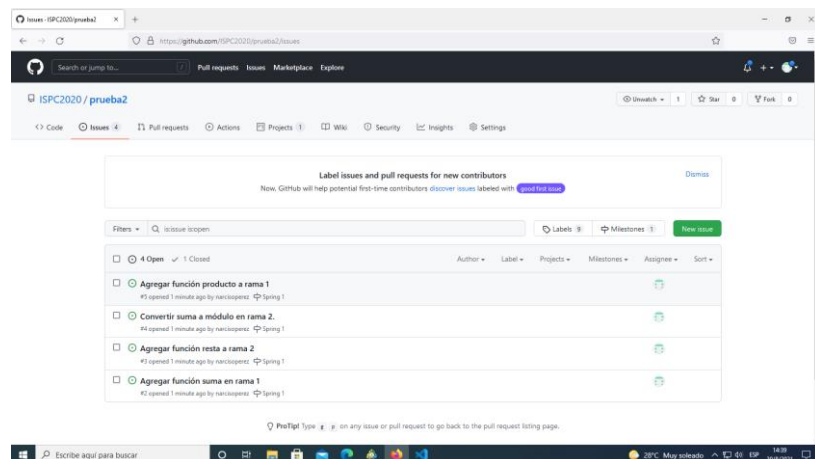
Pasar issue a Done



Cerrar issue

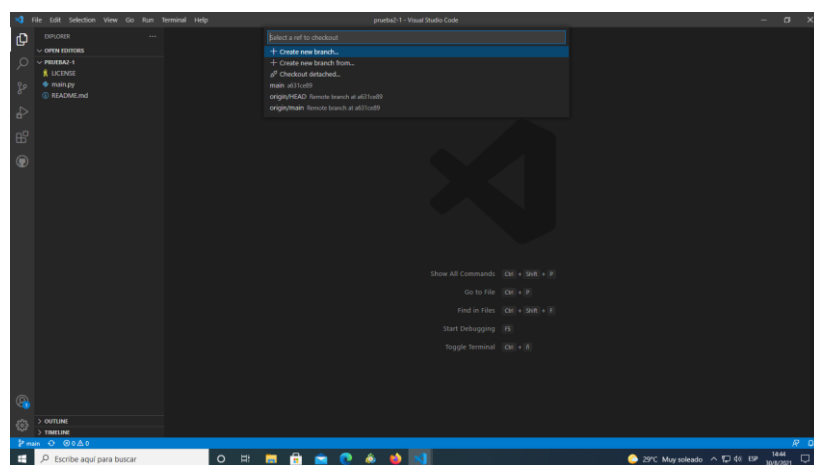
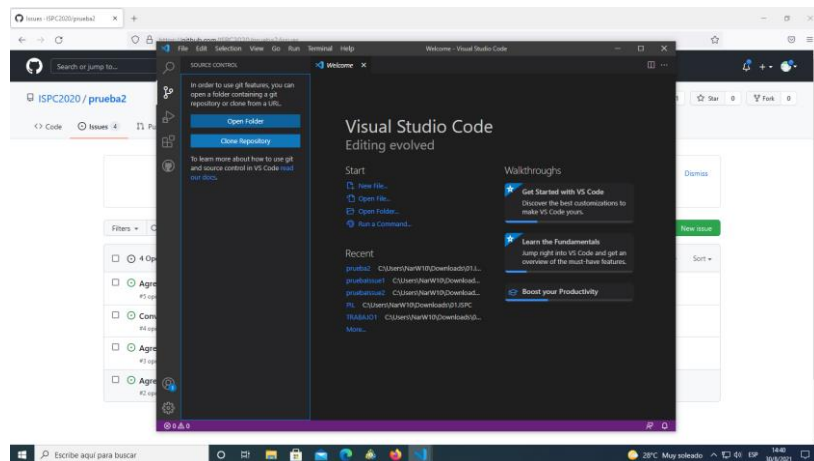


Crearemos todas las issue del Spring 1. Asignadas a milestone y a miembro.

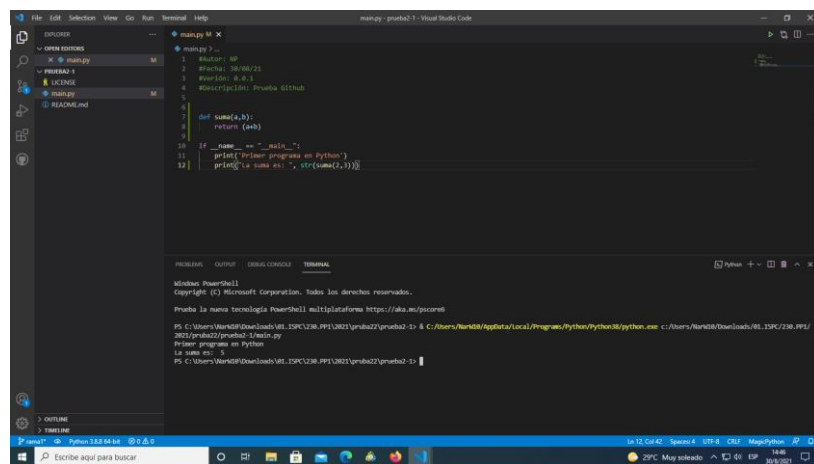


Crear rama1:

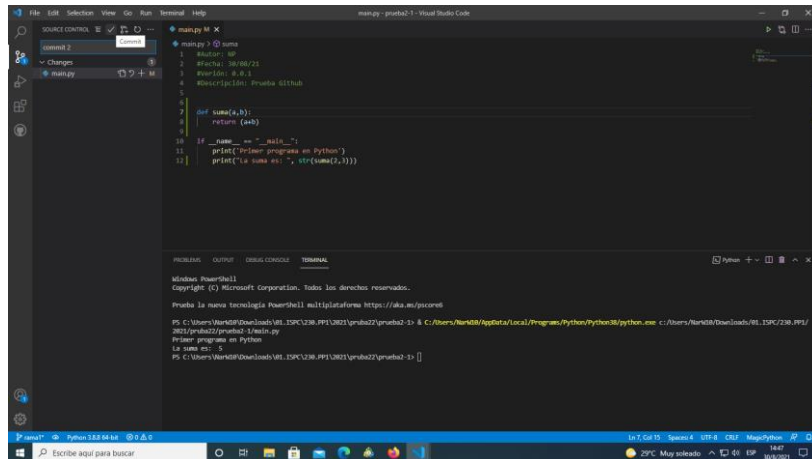
EL segundo integrante clonará el repo remoto en su equipo.



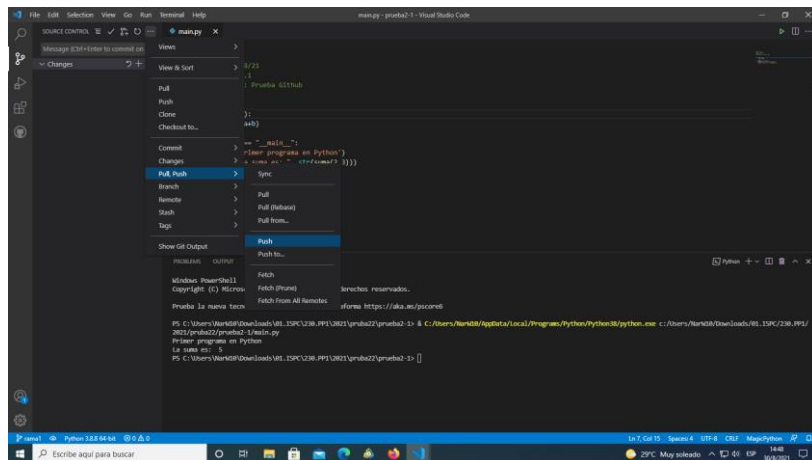
2do issue: agregar con suma, invocar con un print y probar.



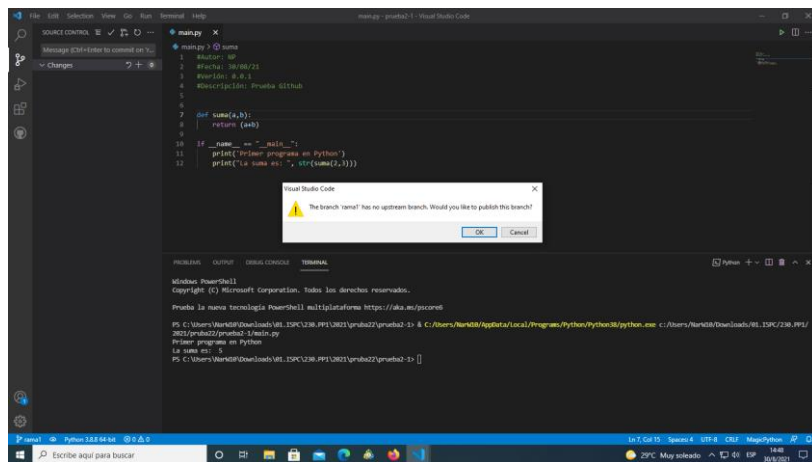
Hacer el commit



Hacer push.

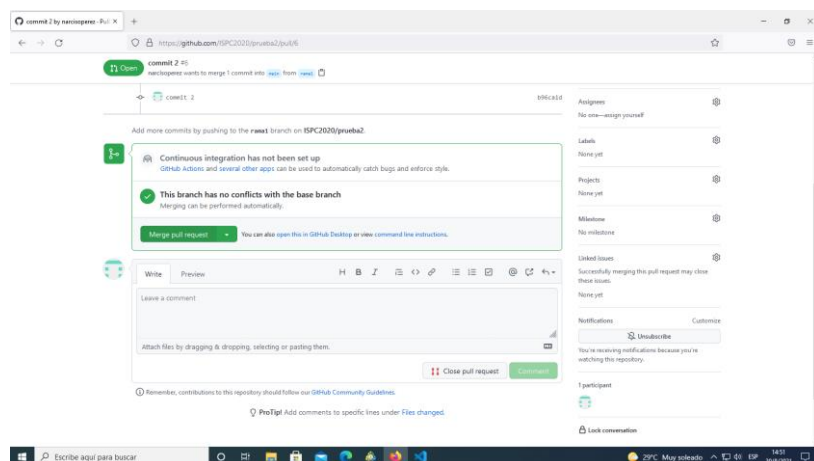
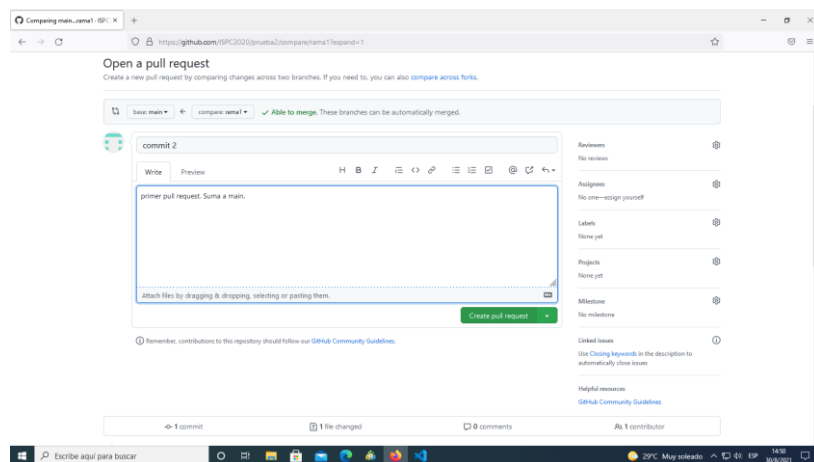
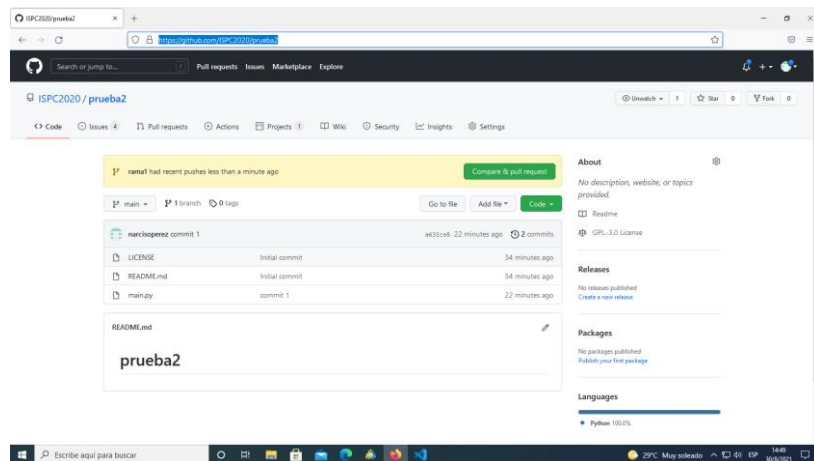


Como no existe la rama1 en el remoto, pide crearla.

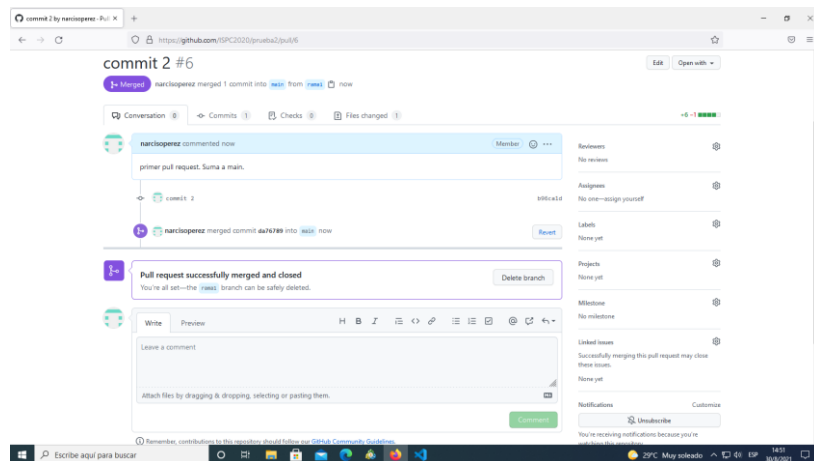


Primer pull request

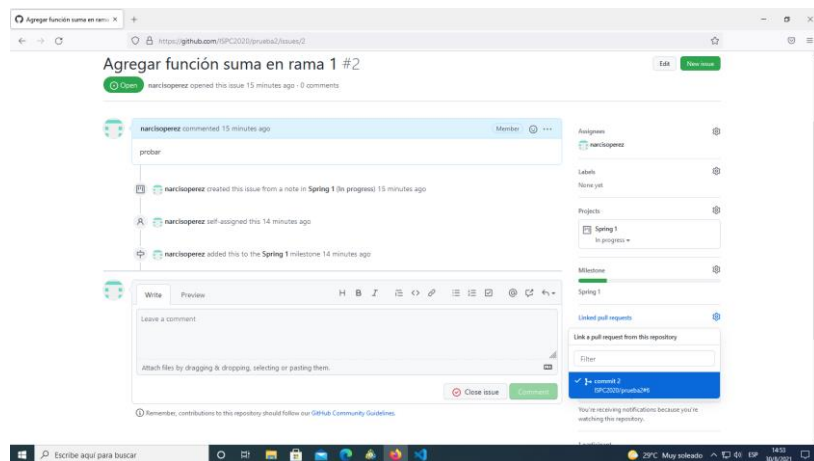




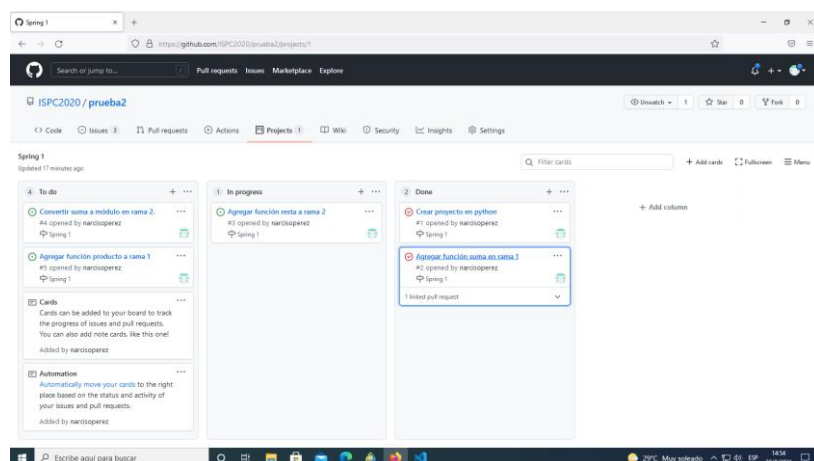
Confirmado merge.



Cerramos issue de suma

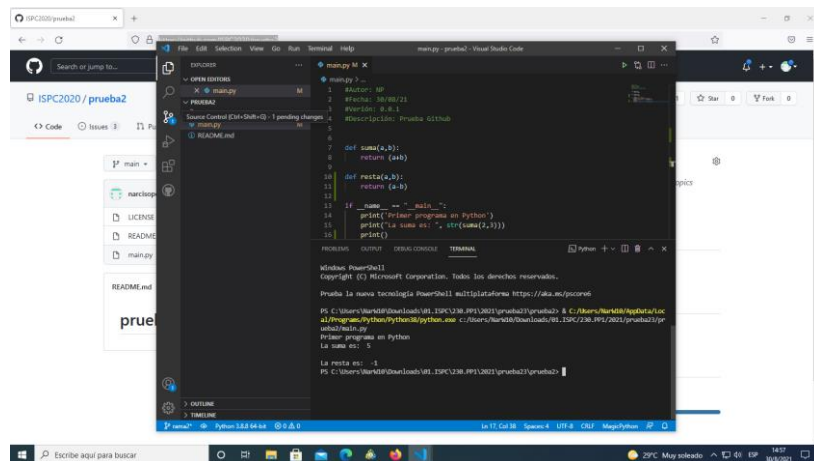


En project, suma a Done.

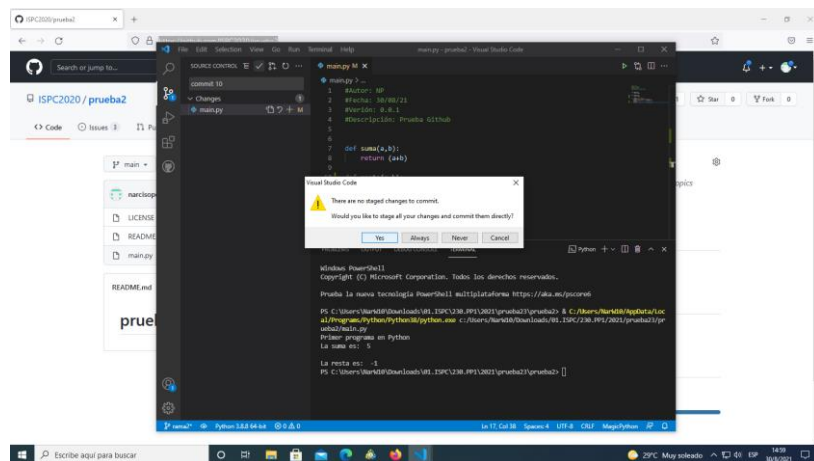


Crear rama2:

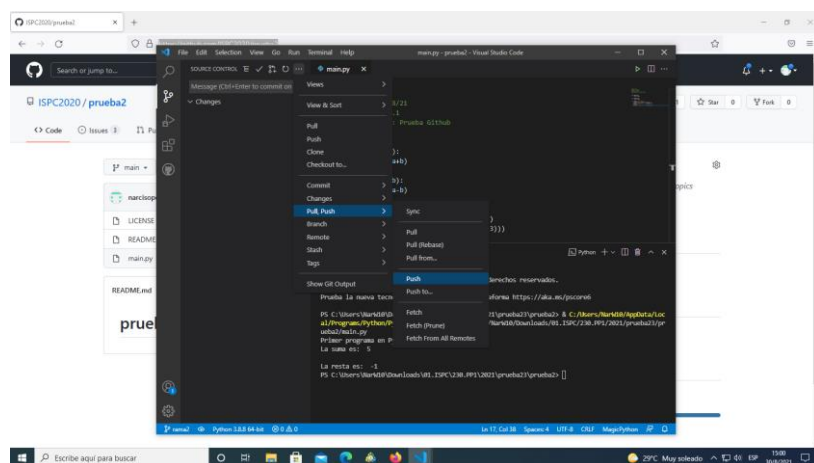
El integrante 3 clona repo en su equipo. Crea rama 2. Carga función resta. Prueba.

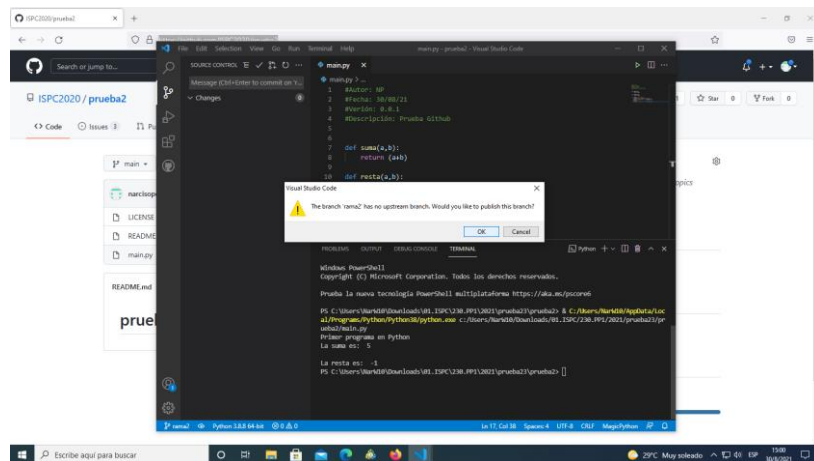


Hace commit. Esto corresponde al 3er issue: probar todo, invocar con un print y probar.

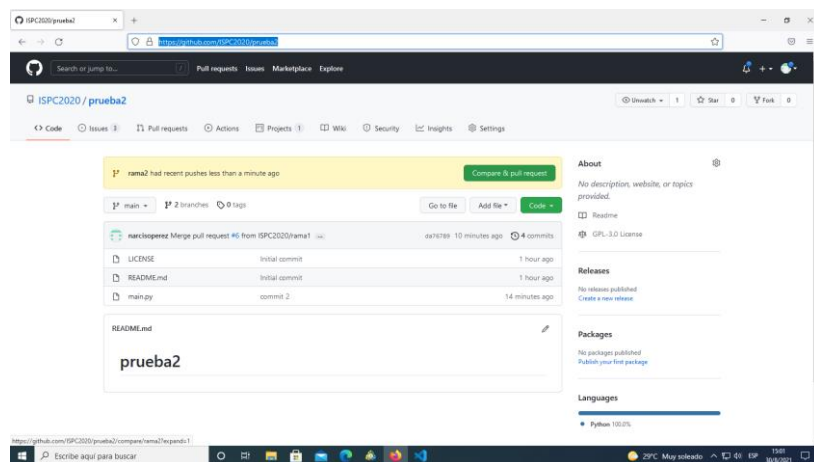


Hacer push.

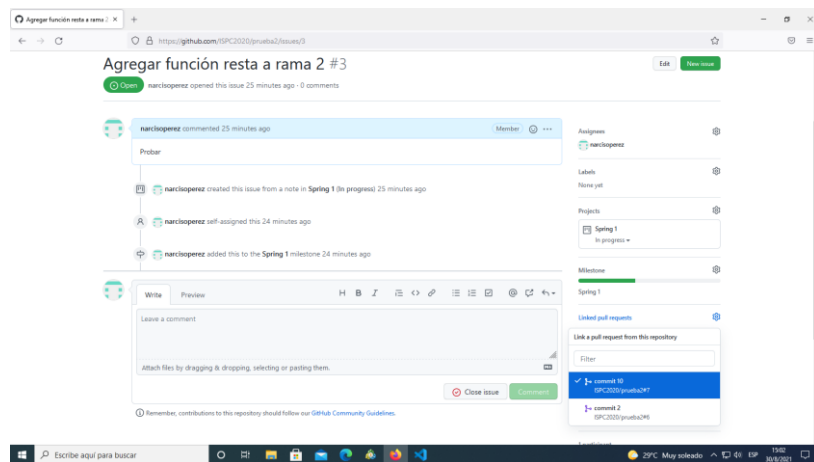




Hacer pull request a main



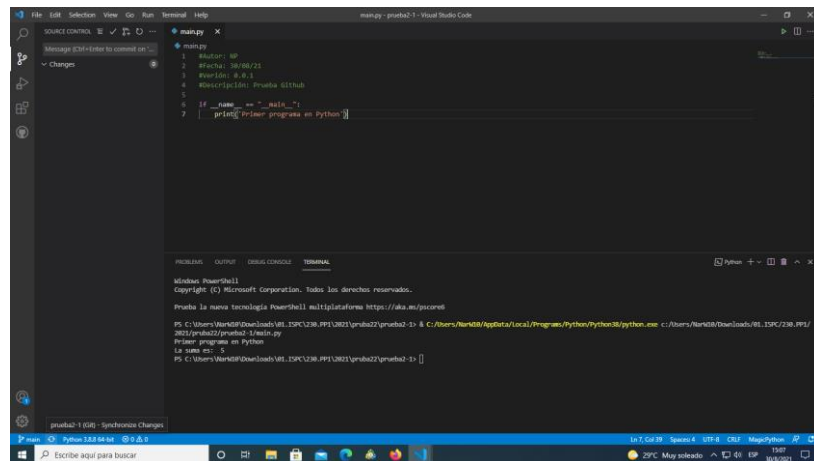
Linkear con tarea en Proyecto



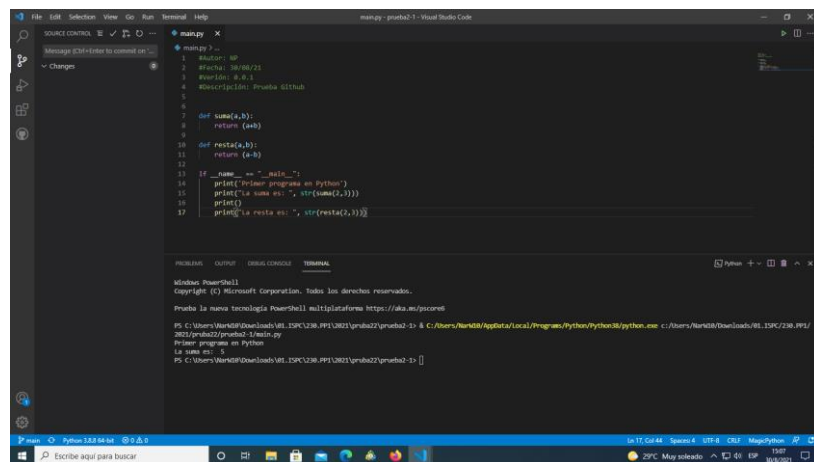
Proceder igualmente con producto.

Ojo: Rama1 esta desactualizado. Debe sincronizar con repo remoto.

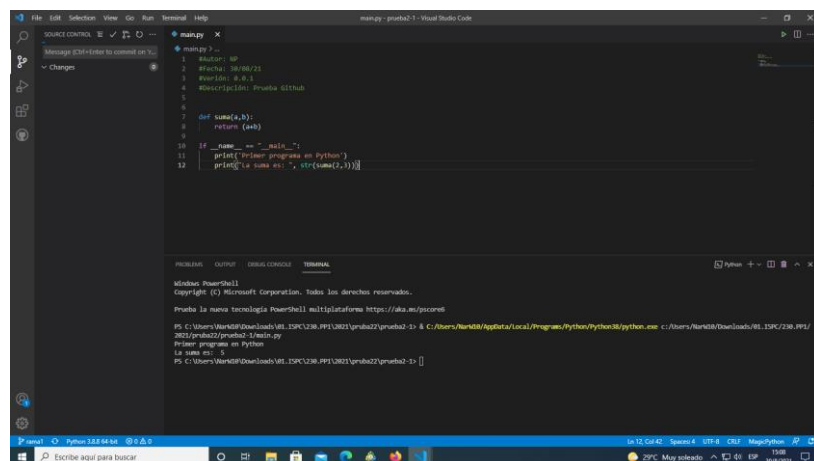
**SIEMPRE PENSAR EN ESTO, de lo contrario aparecerán inconsistencias.**



Se sincroniza main local con main remoto.

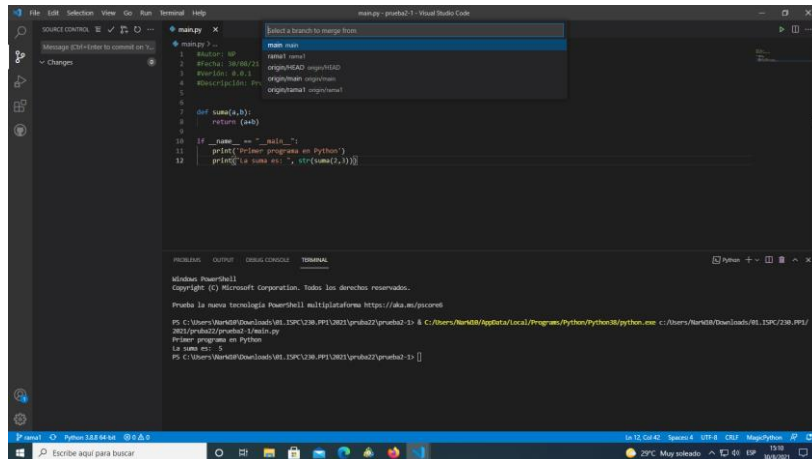


Debo actualizar rama 1 como main



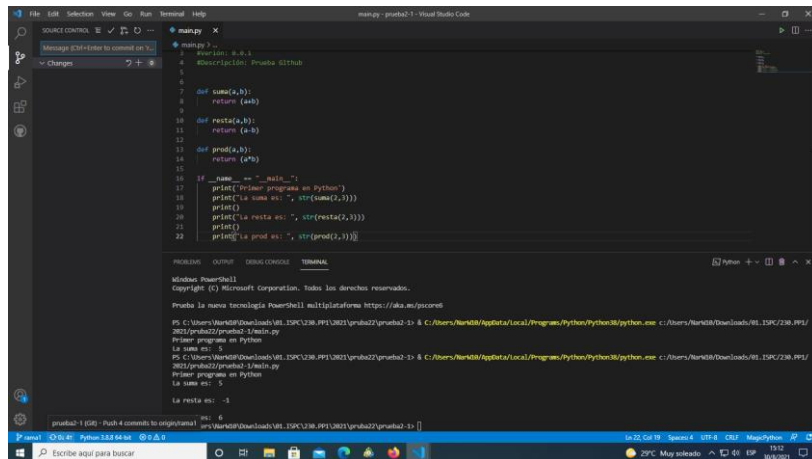
Pero rama 1 esta distinto el código. Debo hacer un merge con main local.

Hacemos un merge from main.



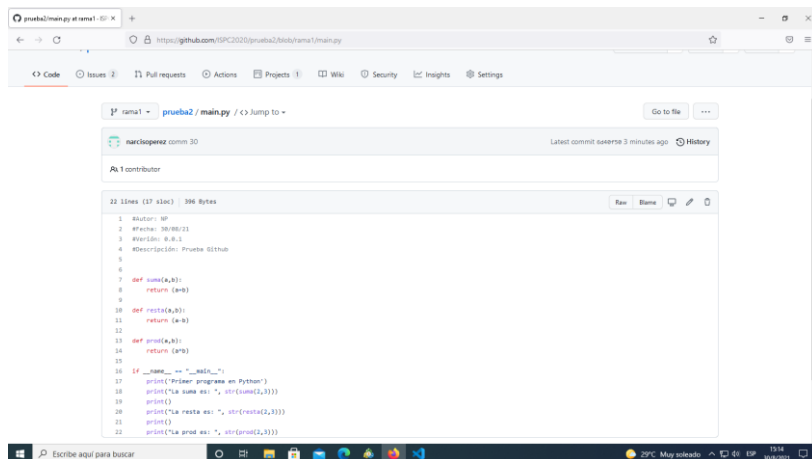
```
1 # -*- coding: utf-8 -*-
2 #fecha: 06/06/21
3 #Versión: 0.0.1
4 #Descripción: Prueba Github
5
6
7 def suma(a,b):
8     return (a+b)
9
10 def resta(a,b):
11     return (a-b)
12
13 def prod(a,b):
14     return (a*b)
15
16 if __name__ == "__main__":
17     print("Primer programa en Python")
18     print("La suma es: ", str(suma(2,3)))
19     print()
20     print("La resta es: ", str(resta(2,3)))
21     print()
22     print("La prod es: ", str(prod(2,3)))
```

Agregamos prod en rama1, probamos, hacemos commit y un push.

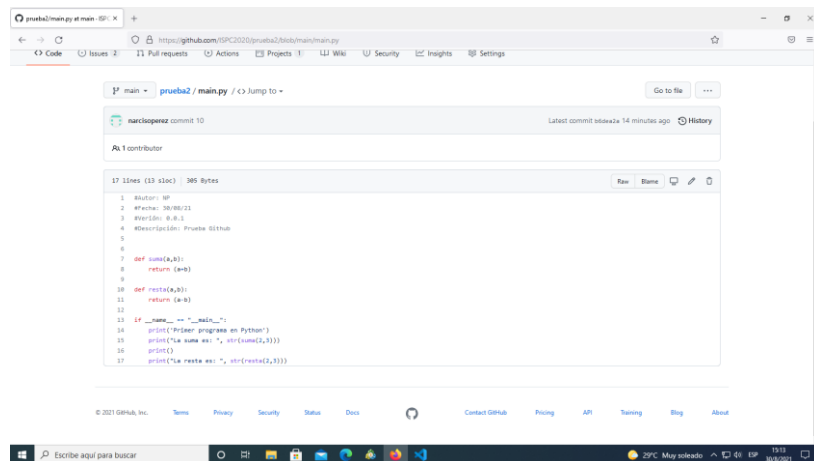


```
1 # -*- coding: utf-8 -*-
2 #fecha: 06/06/21
3 #Versión: 0.0.1
4 #Descripción: Prueba Github
5
6
7 def suma(a,b):
8     return (a+b)
9
10 def resta(a,b):
11     return (a-b)
12
13 def prod(a,b):
14     return (a*b)
15
16 if __name__ == "__main__":
17     print("Primer programa en Python")
18     print("La suma es: ", str(suma(2,3)))
19     print()
20     print("La resta es: ", str(resta(2,3)))
21     print()
22     print("La prod es: ", str(prod(2,3)))
```

Rama 1 remoto tiene prod, pero main remoto no. **OJO**



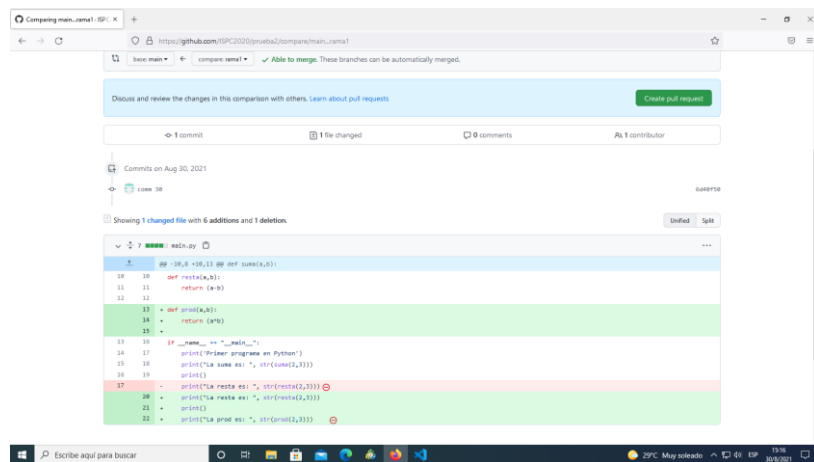
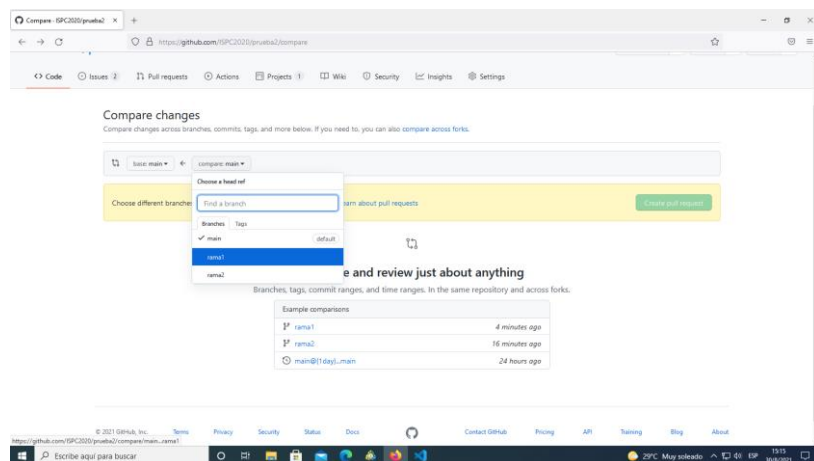
```
1 # -*- coding: utf-8 -*-
2 #fecha: 06/06/21
3 #Versión: 0.0.1
4 #Descripción: Prueba Github
5
6
7 def suma(a,b):
8     return (a+b)
9
10 def resta(a,b):
11     return (a-b)
12
13 def prod(a,b):
14     return (a*b)
15
16 if __name__ == "__main__":
17     print("Primer programa en Python")
18     print("La suma es: ", str(suma(2,3)))
19     print()
20     print("La resta es: ", str(resta(2,3)))
21     print()
22     print("La prod es: ", str(prod(2,3)))
```



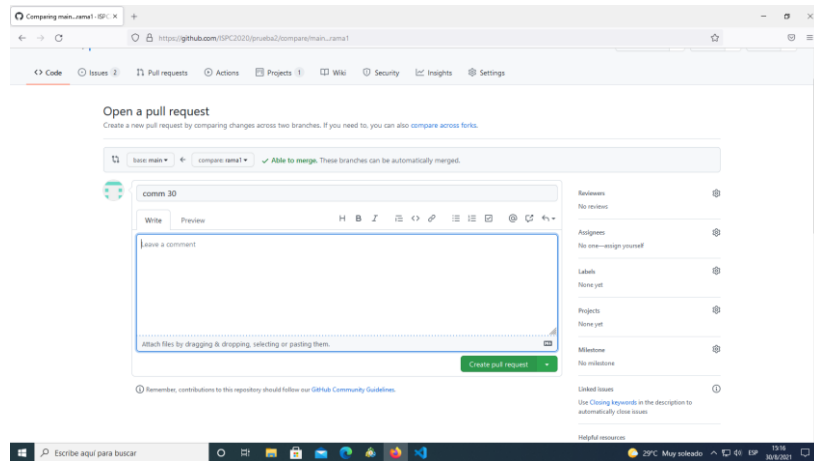
Hacemos pull request manual.

Observar: de rama1 a main.

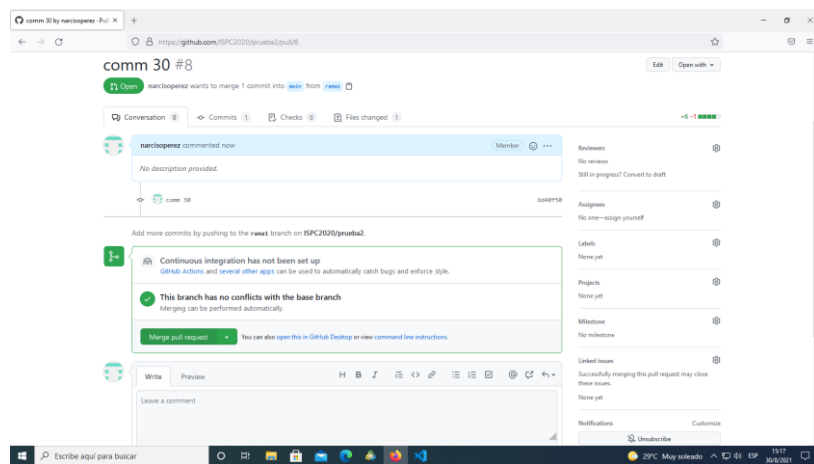
NOTA: debemos estar seguros de lo que hacemos o quedará corrupto main.



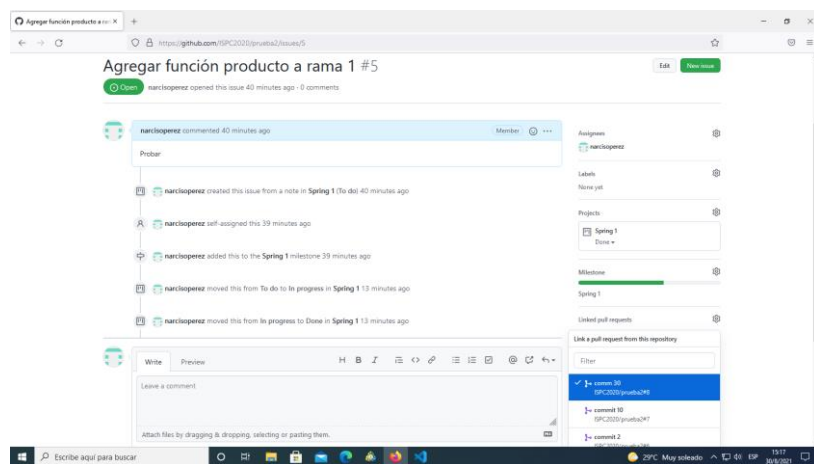
Crear un pull request



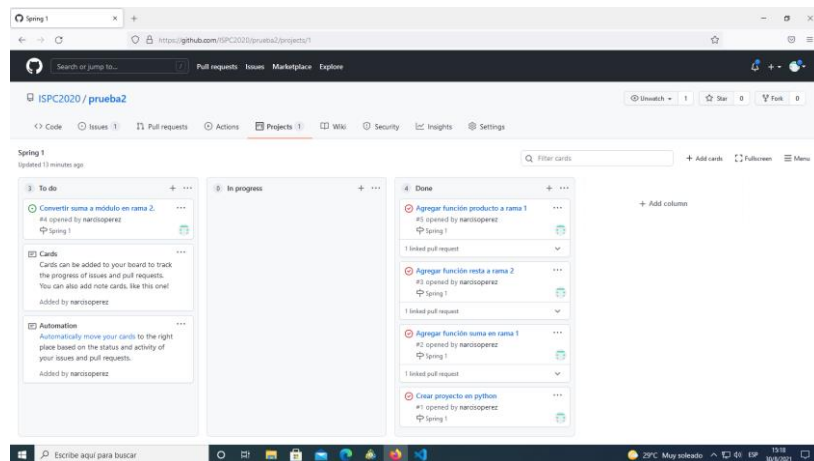
Validar el merge



Linkear pull request con issue prod y cerrar issue. Pasar a Done en project.

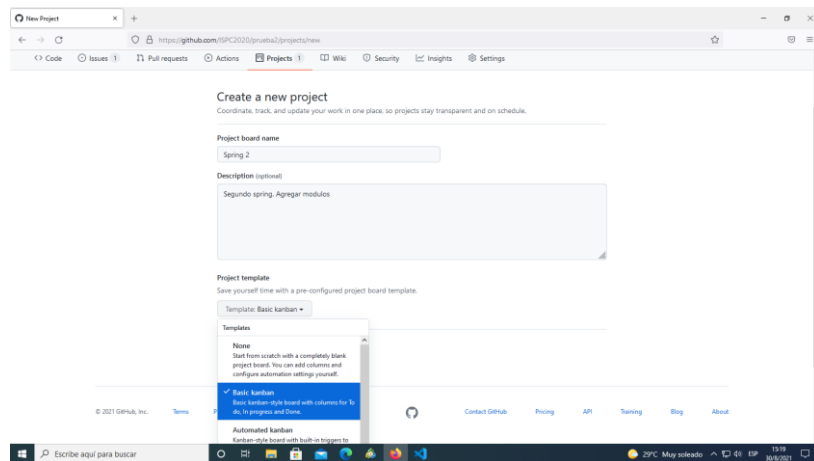




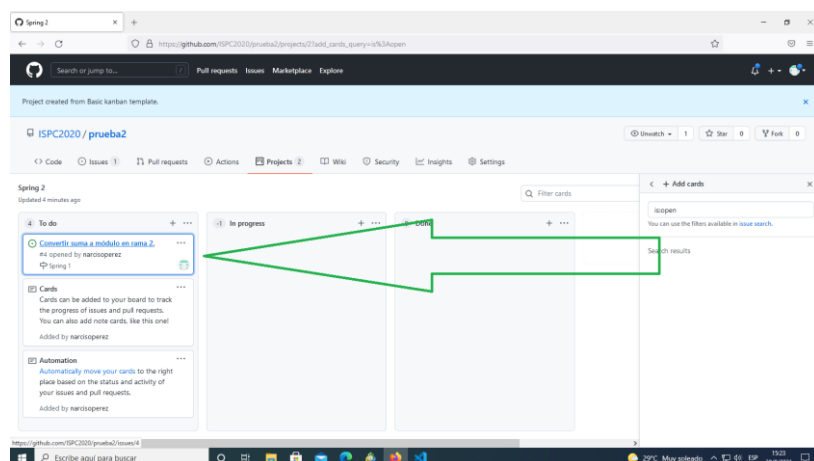


Cerramo el primer spring hasta acá.

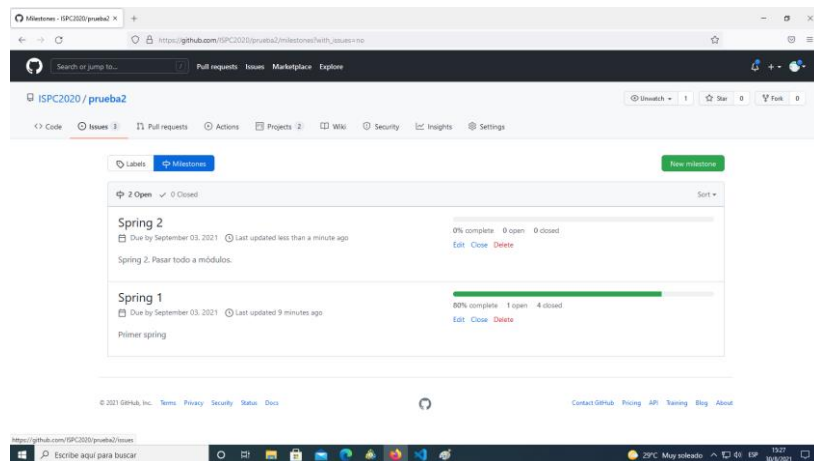
**Ejemplo de segundo ciclo:**



Quedó pendiente funcionalidad. La podemos agregar en el nuevo spring.



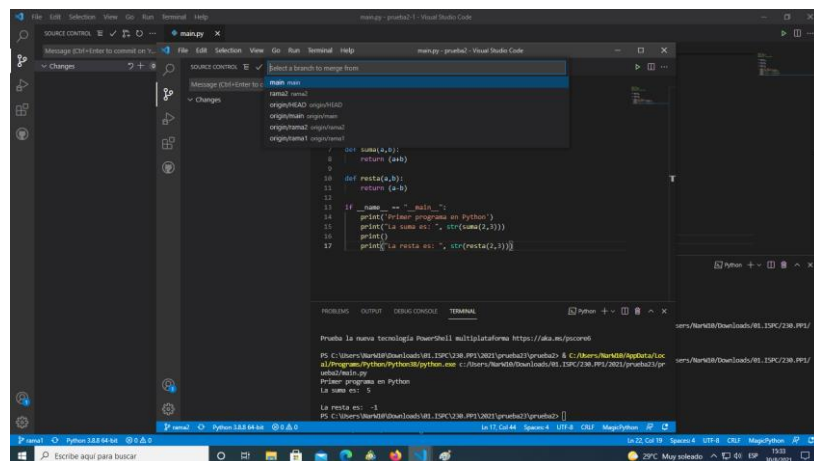
Nuevo milestone



4to issue: Asignado a rama 2, pasar resta a modulo, invocar con un print y probar.

Primero hacer sincronización.

Hacer un merge desde (“merge from”) main local a rama 2.

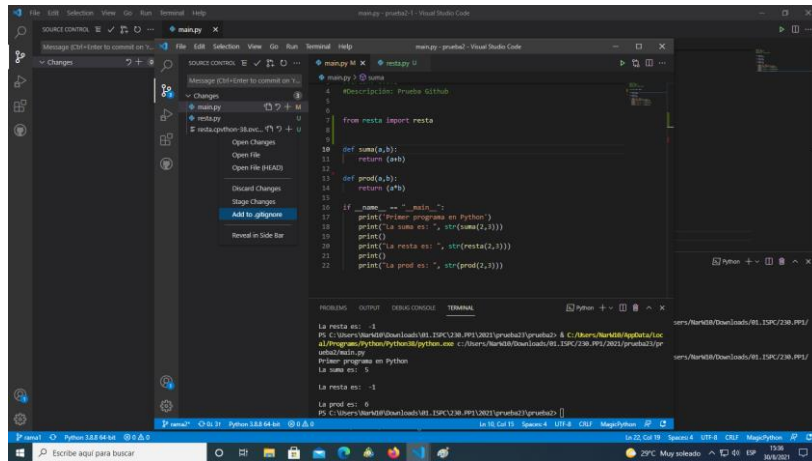


Crear archivo para módulo.

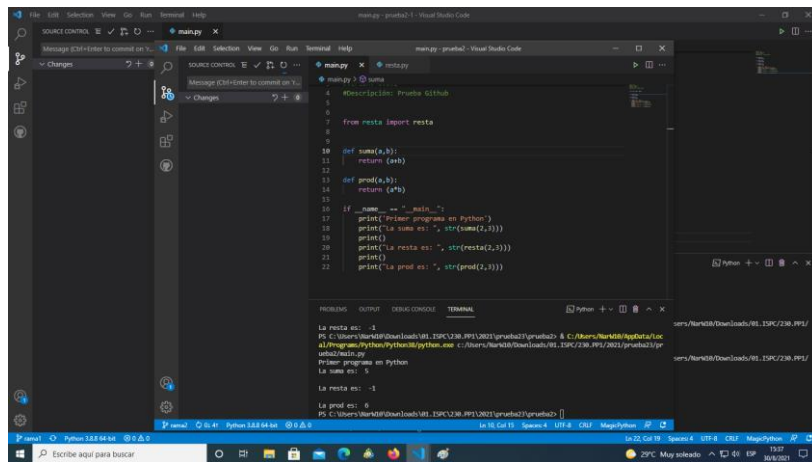
Pasar código a módulo.

Corregir en main.py

Probar

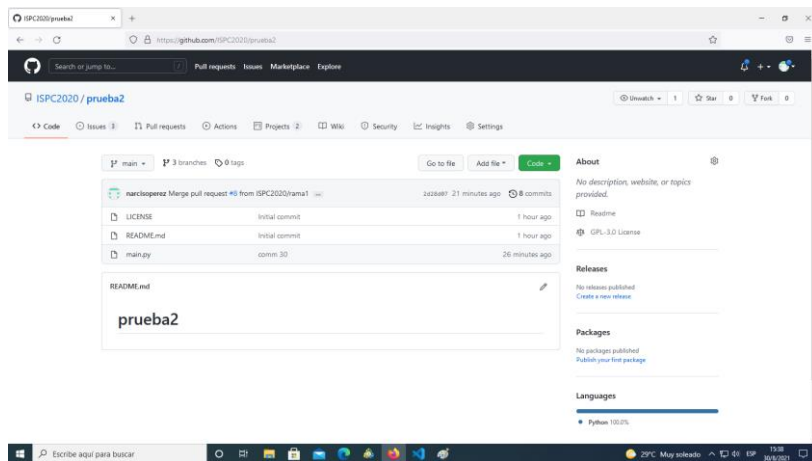


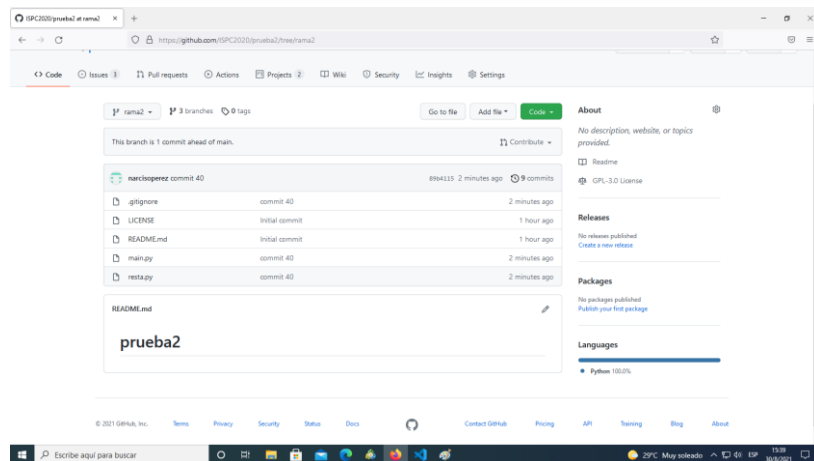
Hacer commit y push a rama 2.



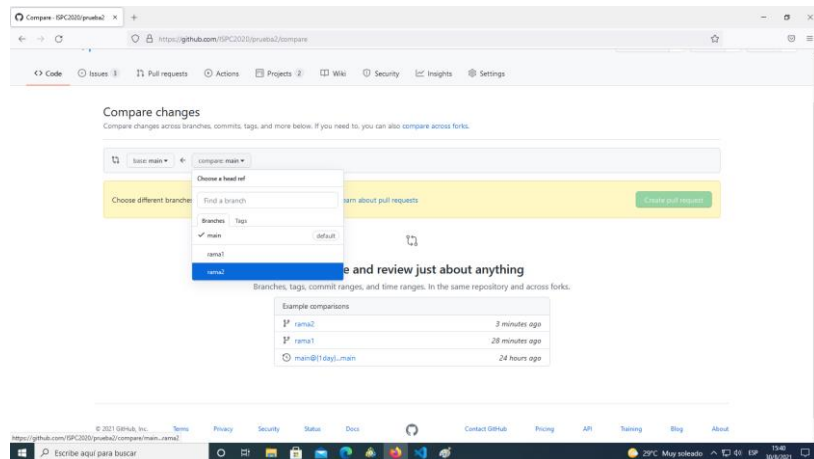
Hacer push a remoto.

Main no tiene aún los cambios, se debe hacer el pull request.





Hacer pull request a main remoto desde rama2



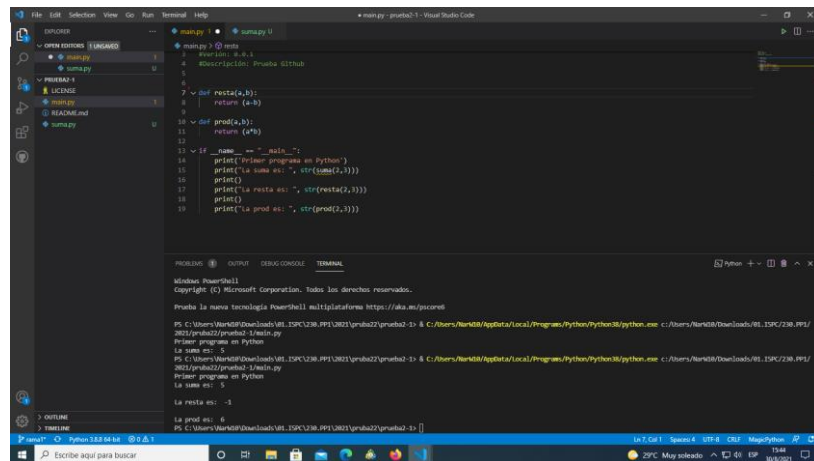
Lickear issue a pull request.

Cerrar issue.

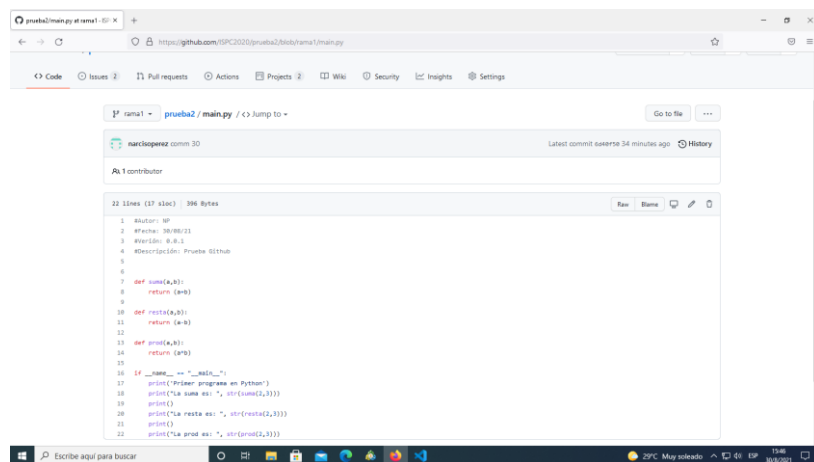
Pasar a Done en Project.

Veremos cómo se producen las corrupciones y como resolverlas.

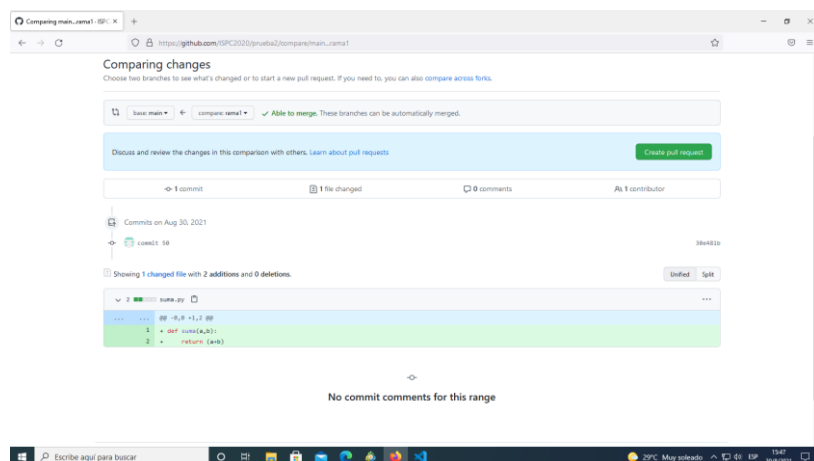
Si rama 1 crea el módulo suma, sin actualizar los correspondientes repositorios se producirán conflictos.

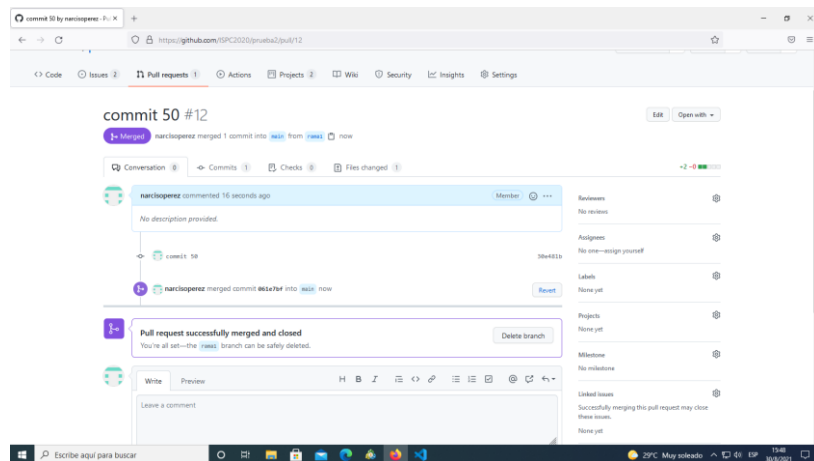


Main.py de rama 1 no está actualizado. Tiene resta, cuando rama 2 ya lo pasó a módulo.

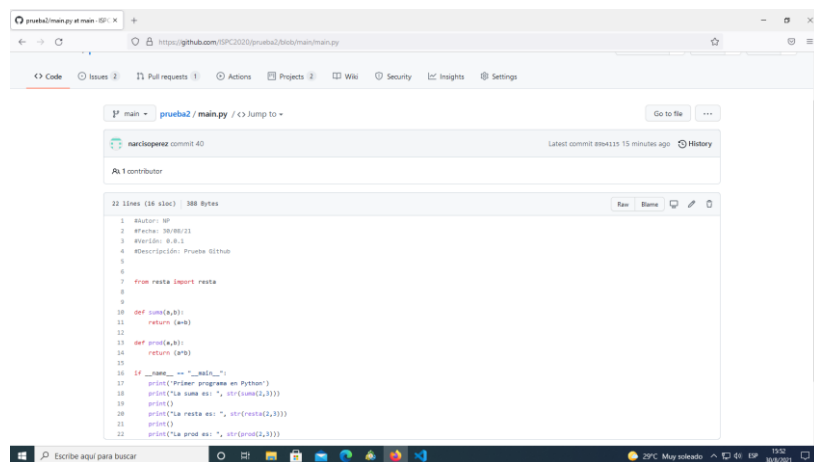


Pedimos un pull request de rama 1 a main

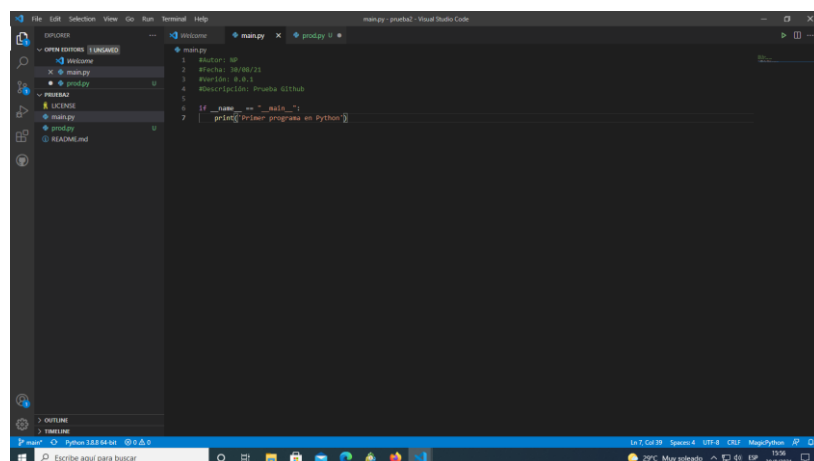


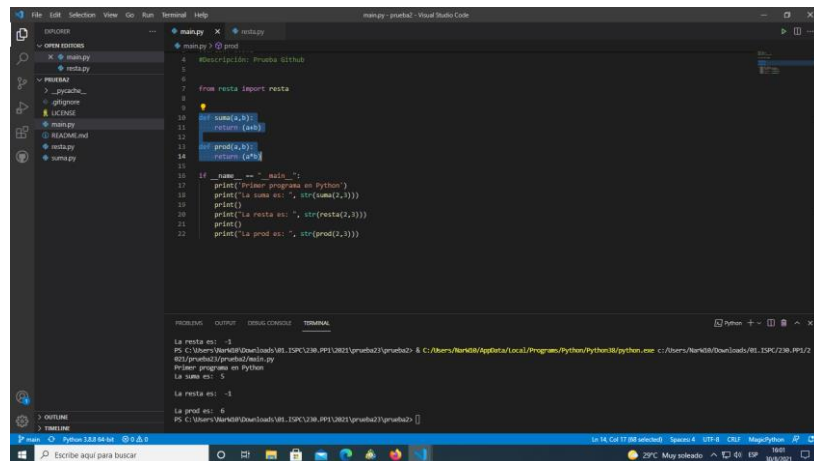


El main.py del main remoto no está correcto.



Para peor si el primero crea prod.py, sin hacer actualizaciones del repo.

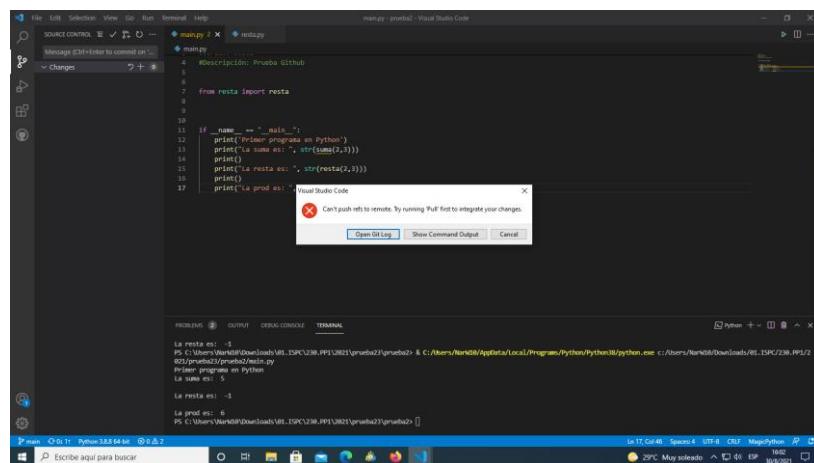




```
4 #descripcion: Prueba Github
5
6
7 from resta import resta
8
9
10 def suma(a,b):
11     return (a+b)
12
13 def prod(a,b):
14     return (a*b)
15
16 if __name__ == "__main__":
17     print("Primer programa en Python")
18     print("La suma es: ", str(suma(2,3)))
19     print()
20     print("La resta es: ", str(resta(2,3)))
21     print()
22     print("La prod es: ", str(prod(2,3)))
```

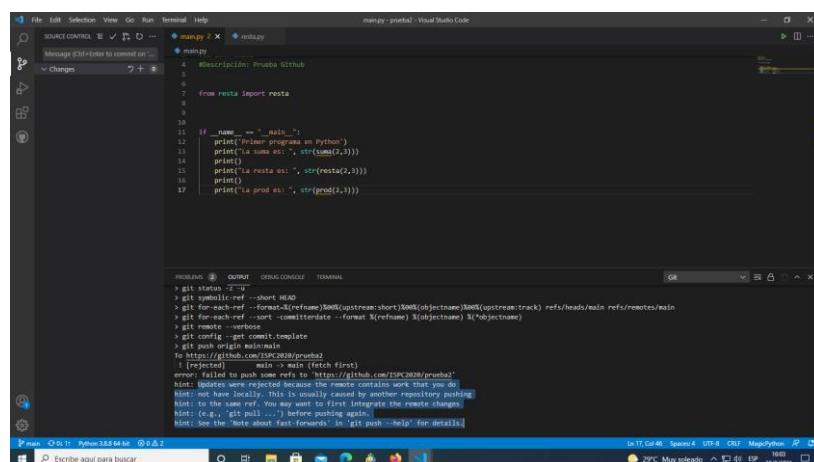
La resta es: -1  
PS C:\Users\Narciso\Downloads\URL\_EJERCICIOS\Python2\Python2> & C:\Users\Narciso\AppData\Local\Programs\Python\Python38\python.exe c:\Users\Narciso\Downloads\URL\_EJERCICIOS\Python2\Python2/main.py  
Primer programa en Python  
La suma es: 5  
La resta es: -1  
La prod es: 6  
PS C:\Users\Narciso\Downloads\URL\_EJERCICIOS\Python2\Python2>

Aparecen las inconsistencias, porque se rompió la lógica de trabajo.



Can't push refs to remote. Try running 'Pull' first to integrate your changes.

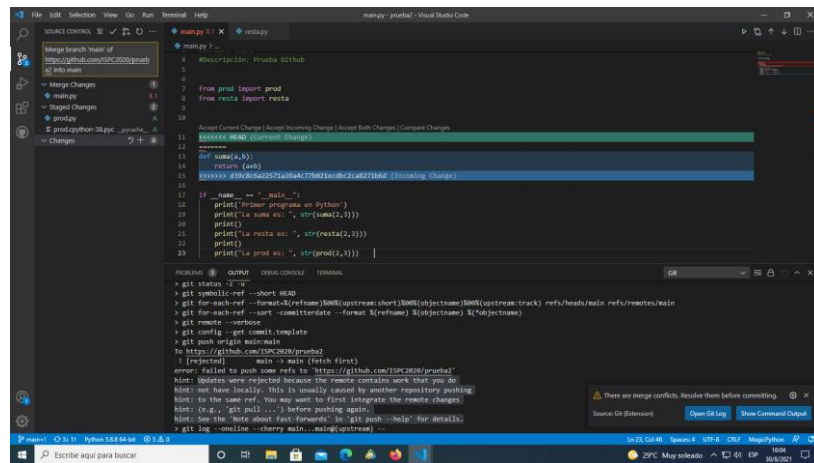
Open Git Log Show Command Output Cancel



```
> git status
> git symbolic-ref --short HEAD
> git for-each-ref --format=%(objectname)%00%(upstream:short)%00%(objectname)%00%(upstream:track) refs/heads/main refs/remotes/main
> git remote --verbose
> git config --get commit.template
> git push origin main:main
To https://github.com:ESP8266/prueba2
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'https://github.com:ESP8266/prueba2'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.
```

Me dice que debería hacer un pull, o sea traer del remoto para tener lo que está en el main remoto.

Pero al hacerlo me indicará lo que está diferente y debo ser precavido en las acciones.



Próxima clase veremos conflictos y como resolverlos.