



8 de agosto de 2019

Actividad 00

Git y GitHub

Ayuda memoria de operaciones con Git

1. **Clonar:** `git clone <url del repositorio>`
2. **Revisar estado:** `git status`
3. **Agregar un archivo al *staging area*:** `git add file.py`
4. **Crear *commit*:** `git commit -m "Descripción del commit"`
5. **Deshacer cambios de un archivo en el *staging area*:** `git reset HEAD file_name`
6. **Deshacer último commit:** `git reset HEAD~1`
7. **Llevar versiones al repo remoto:** `git push`
8. **Traer versiones al repo local:** `git pull`

Parte 1: Bienvenid@ a IIC2233

El objetivo de esta parte es entender cuáles son las principales páginas relacionadas con el curso. Los requisitos para empezar son:

1. Haberte registrado en el curso según los avisos mandados.
2. Haber instalado Git en Windows (más info en [este enlace](#)). Si tienes macOS o Linux, muy probablemente ya venga instalado.

Syllabus

El *syllabus* es un **repositorio** de GitHub donde subiremos todos los enunciados de las tareas, actividades y ayudantías. Puedes entrar al *syllabus* desde el hipervínculo ubicado en la página principal del curso, o bien, en github.com/IIC2233/syllabus.

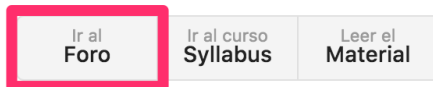
El *syllabus* tiene asociado un foro de *issues*. En este foro puedes preguntar sobre la materia o algo sobre los enunciados de las tareas. También los ayudantes podrán colocar avisos importantes, ya sean detalles administrativos o aclaraciones de enunciados, **por lo que es tu obligación estar atento al contenido de este foro**¹.

¹Existe un canal de [Telegram](#) (un cliente de mensajería, mejor que WhatsApp, que utilizamos mucho en el DCC) que avisa cuando una *issue* es marcada como importante. El canal es t.me/avanzadaissues y existe gracias a Enzo Tamburini.

IIC2233 @ UC

IIC2233 Programación Avanzada @
Pontificia Universidad Católica de Chile

[Ver el Curso en GitHub](#)
IIC2233 organization



Información del curso

Programa oficial completo en este [link](#)

Este curso enseña técnicas para diseñar, implementar, ejecutar y evaluar herramientas de software que resuelven problemas algorítmicos a partir de especificaciones detalladas. En particular, el curso enseña construcciones avanzadas de programación orientada a objetos, estructuras de datos fundamentales, diseño básico de algoritmos y técnicas de análisis.

Figura 1: Llegar al foro a través del *link* en la página principal.

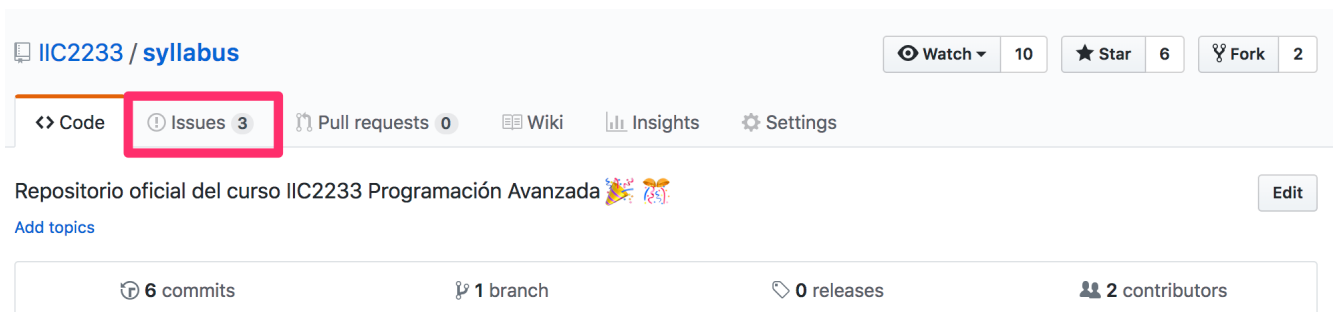


Figura 2: Llegar al foro a través del *syllabus*.

Repositorio de apuntes

Los apuntes con los contenidos del curso se encontrarán en un repositorio llamado “contenidos”, al que puedes llegar haciendo clic en un vínculo ubicado en la página principal del curso, o yendo directamente a github.com/IIC2233/contenidos.

Tu repositorio personal

Cuando te registraste en el curso te han creado un repositorio **personal** y **privado** donde deberás entregar tus actividades y tareas. Este se ubica en github.com/IIC2233/usuario-iic2233-2019-2, donde debes reemplazar “usuario” por tu usuario de GitHub.

Tu repositorio también tiene asociado un foro de *issues*. En él recibirás —en forma automatizada— el detalle de la corrección de tus actividades y tareas. **No respondas los mensajes ahí, puesto que nadie leerá lo que escribas.** Si tienes dudas sobre tu corrección usa el correo de los ayudantes o solicita una corrección mediante formulario.

IIC2233 / jecastro1-iic2233-2018-2 Private

Watch 5 Star 0 Fork 0

Code Issues 3 Pull requests 0 Insights Settings

Repositorio para Jaime Castro Edit

Add topics

2 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

Hernan4444 new template Latest commit 6e9244e 20 days ago

Actividades	new template	20 days ago
Tareas	new template	20 days ago
.gitignore	new template	20 days ago
README.md	new template	20 days ago

Figura 3: Ejemplo de repositorio personal.

IIC2233 / jecastro1-iic2233-2018-2 Private

Watch 5 Star 0 Fork 0

Code Issues 3 Pull requests 0 Insights Settings

Filters is:issue is:open Labels Milestones New issue

3 Open 0 Closed Author Labels Projects Milestones Assignee Sort

AC02	#3 opened 3 days ago by Hernan4444	
AC01	#2 opened 3 days ago by Hernan4444	
AC00	#1 opened 3 days ago by Hernan4444	

ProTip! Check team mentions with [team:IIC2233/profesores](#).

Figura 4: *Issues* del repositorio personal.

Parte 2: Ahora sí, bienvenid@ a IIC2233

El objetivo de esta parte es tener un primer contacto con el sistema de control de versiones que se utiliza en el curso.

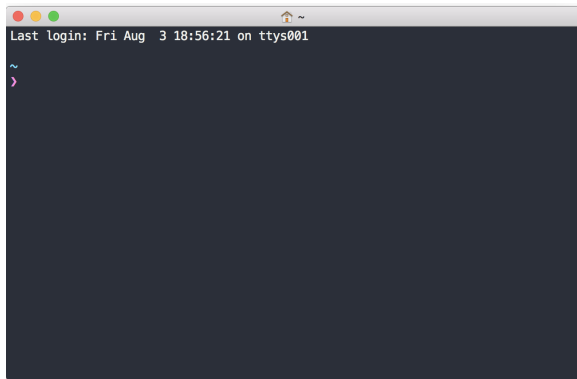
Aprender lo básico de la consola

Necesitamos aprender a navegar por las carpetas del computador usando la consola. En esta parte de la guía, sigue todos los pasos y verifica que te sale lo mismo que está aquí.

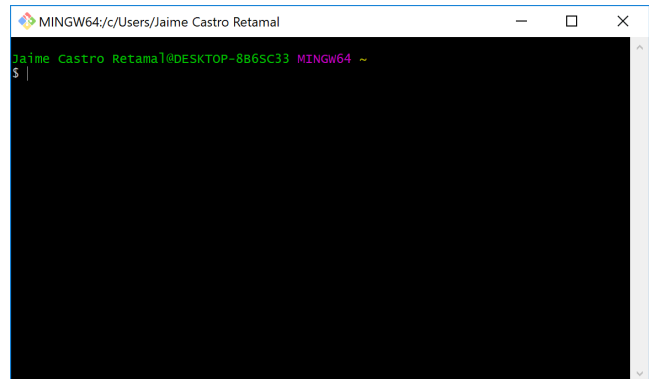
Para abrir la consola:

- **macOS o Linux:** busca el programa “Terminal” o similar.
- **Windows:** busca el programa “Git Bash”. Este programa es una consola que además implementa algunos de los comandos que podrías encontrar en Linux, por lo que es mucho más amigable que el “Símbolo de Sistema”.

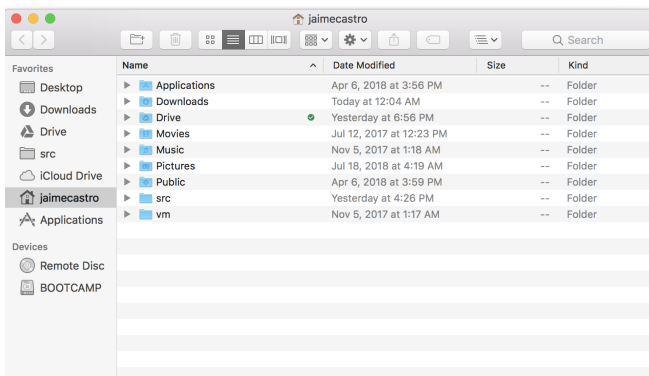
La consola en todo momento está ubicada en una carpeta. Cuando se abre, será la carpeta donde están todos tus datos, “*home*” o “*~*”. Dentro de ella, por ejemplo, se encuentra el escritorio o la carpeta de documentos.



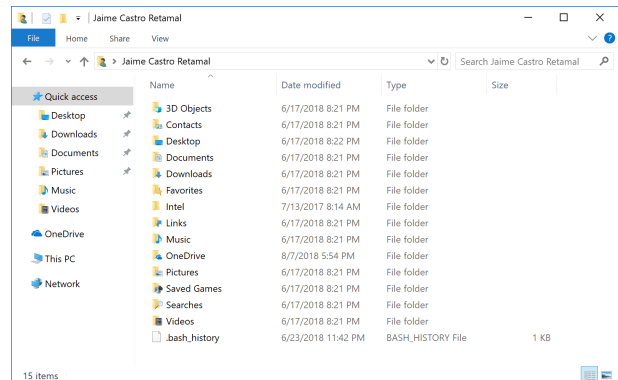
(a) Consola abierta en macOS



(b) Consola abierta en Windows

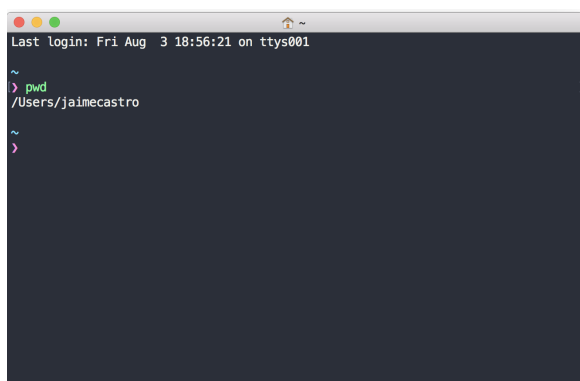


(a) “Home” en macOS



(b) “Home” en Windows

Muchas veces las consolas muestran en qué parte están ubicadas, pero el comando `pwd` también nos entrega esa información.



```
Last login: Fri Aug 3 18:56:21 on ttys001

~
> pwd
/Users/jaimecastro

~
>
```

(a) `pwd` en macOS estando en el “home”



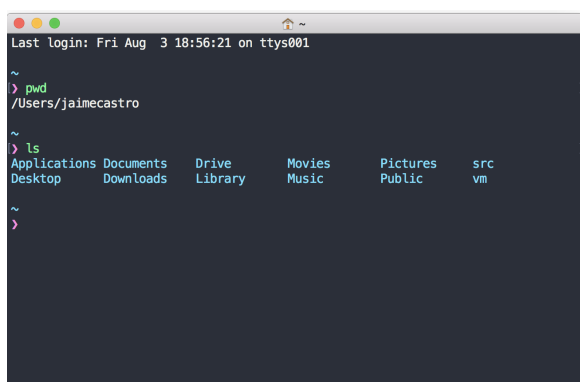
```
MINGW64/c/Users/Jaime Castro Retamal

Jaime Castro Retamal@DESKTOP-8B65C33 MINGW64 ~
$ pwd
/c/Users/Jaime Castro Retamal

Jaime Castro Retamal@DESKTOP-8B65C33 MINGW64 ~
$ |
```

(b) `pwd` en Windows estando en el “home”

También podemos saber qué es lo que contiene el directorio actual, con el comando `ls`.



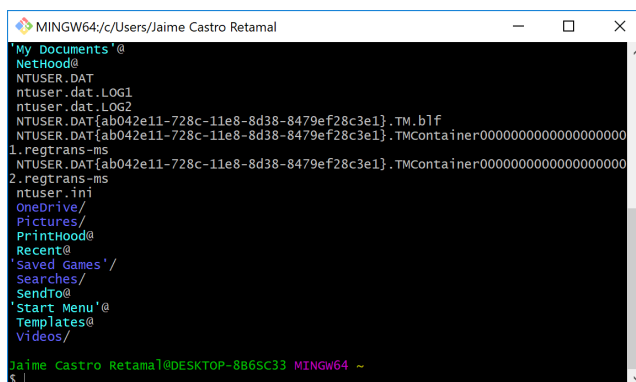
```
Last login: Fri Aug 3 18:56:21 on ttys001

~
> pwd
/Users/jaimecastro

~
> ls
Applications  Documents  Drive      Movies     Pictures   src
Desktop        Downloads  Library    Music      Public     vm

~
>
```

(a) `ls` en el “home” en macOS



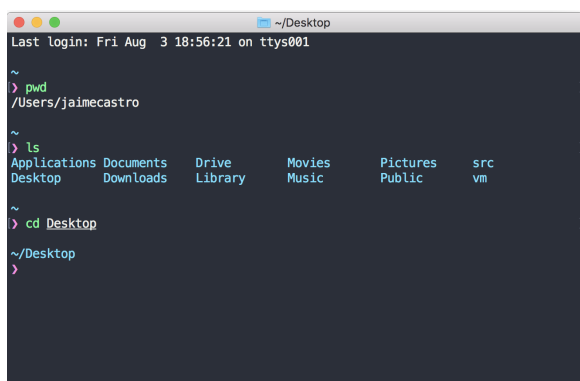
```
MINGW64/c/Users/Jaime Castro Retamal

My Documents'@
Nethood@
NTUSER.DAT
ntuser.dat.LOG1
ntuser.dat.LOG2
NTUSER.DAT{ab042e11-728c-11e8-8d38-8479ef28c3e1}.TM.b1f
NTUSER.DAT{ab042e11-728c-11e8-8d38-8479ef28c3e1}.TMContainer00000000000000000000
1.regtrans-ms
NTUSER.DAT{ab042e11-728c-11e8-8d38-8479ef28c3e1}.TMContainer00000000000000000000
2.regtrans-ms
ntuser.ini
OneDrive/
Pictures/
PrintHood@
Recent@
'Saved Games'/
Searches/
SendTo@
'Start Menu'@
Templates@
Videos/

Jaime Castro Retamal@DESKTOP-8B65C33 MINGW64 ~
$ |
```

(b) `ls` en el “home” en Windows

Supongamos que queremos mover la ubicación de la consola a la carpeta “Desktop”. Para ello, usamos el comando `cd` (*change directory*). En este caso, debemos teclear `cd Desktop`.



```
Last login: Fri Aug 3 18:56:21 on ttys001

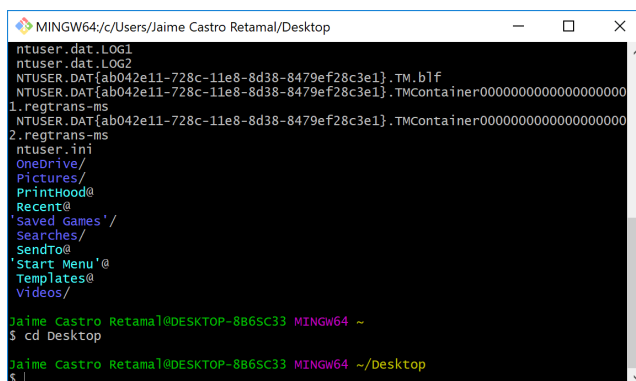
~
> pwd
/Users/jaimecastro

~
> ls
Applications  Documents  Drive      Movies     Pictures   src
Desktop        Downloads  Library    Music      Public     vm

~
> cd Desktop

~/Desktop
>
```

(a) `cd Desktop` en macOS



```
MINGW64/c/Users/Jaime Castro Retamal/Desktop

ntuser.dat.LOG1
ntuser.dat.LOG2
NTUSER.DAT{ab042e11-728c-11e8-8d38-8479ef28c3e1}.TM.b1f
NTUSER.DAT{ab042e11-728c-11e8-8d38-8479ef28c3e1}.TMContainer00000000000000000000
1.regtrans-ms
NTUSER.DAT{ab042e11-728c-11e8-8d38-8479ef28c3e1}.TMContainer00000000000000000000
2.regtrans-ms
ntuser.ini
OneDrive/
Pictures/
PrintHood@
Recent@
'Saved Games'/
Searches/
SendTo@
'Start Menu'@
Templates@
Videos/

Jaime Castro Retamal@DESKTOP-8B65C33 MINGW64 ~
$ cd Desktop

Jaime Castro Retamal@DESKTOP-8B65C33 MINGW64 ~/Desktop
$ |
```

(b) `cd Desktop` en Windows

Si nos queremos devolver a la carpeta que contiene a “Desktop”, usamos `cd ..` (los dos puntos significan “la carpeta que contiene la actual”).

```
Last login: Fri Aug 3 18:56:21 on ttys001

~
> pwd
/Users/jaimecastro

~
> ls
Applications  Documents  Drive      Movies     Pictures   src
Desktop       Downloads  Library    Music      Public     vm

~
> cd Desktop
~/Desktop
> cd ..
~
> |
```

(a) `cd ..` en macOS

```
MINGW64/c/Users/Jaime Castro Retamal

NTUSER.DAT{ab042e11-728c-11e8-8d38-8479ef28c3e1}.TMContainer000000000000000000000000
1.regtrans-ms
NTUSER.DAT{ab042e11-728c-11e8-8d38-8479ef28c3e1}.TMContainer000000000000000000000000
2.regtrans-ms
ntuser.ini
OneDrive/
Pictures/
PrintHood@
Recent@
'Saved Games'/
Searches/
SendTo@
'Start Menu'@
Templates@
Videos/

Jaime Castro Retamal@DESKTOP-8B65C33 MINGW64 ~
$ cd Desktop

Jaime Castro Retamal@DESKTOP-8B65C33 MINGW64 ~/Desktop
$ cd ..

Jaime Castro Retamal@DESKTOP-8B65C33 MINGW64 ~
$ |
```

(b) `cd ..` en Windows

Clonar repositorio personal

En esta parte, clonaremos tu repositorio personal para que puedas entregar tus actividades y tareas.

1. Asegúrate de que la consola esté dentro de la carpeta donde quieras tener tu repo.
2. Ve a la página de tu repo y copia el vínculo que permite clonar el repositorio.
3. En la consola, ejecuta el comando en la consola para clonarlo: `git clone url_copiada`.
4. Deberías ver que se creó la carpeta con los contenidos de tu repo personal.

En caso de que ya hayas copiado tu repositorio personal en otro lugar, simplemente puedes mover la carpeta completa a un directorio que prefieras.

Clonar otros repositorio importantes del curso

Aprovechando el vuelo, haz los mismos pasos del punto anterior pero para “*syllabus*” y para “*contenidos*”, ya que tendrás que estar pendiente de estos repositorios durante el semestre.

El primer *commit* y *push*

Haremos nuestro primer *commit* dentro del repositorio personal. Para trabajar en él, tu consola deberá estar dentro de la carpeta del repositorio.

1. Dentro del repositorio, hay un archivo llamado `README.md` creado. En este, hay información base de tu repositorio y se dejan espacios para que rellenes con tus datos. Abre el archivo con algún editor de texto y rellena los espacios con tus datos.
2. Ahora, vuelve a la consola. Revisa el estado del repositorio utilizando `git status`. Observa que tu archivo recién editado aparece bajo “*Untracked files*”.
3. Agrega el archivo `README.md` al *staging area* mediante `git add`. Recuerda que debes escribir el nombre del archivo después de este comando. Luego revisa el estado del repositorio y verifica que se haya agregado.
4. Utiliza `git commit` de los cambios realizados. Recuerda escribir un mensaje **descriptivo**. Luego revisa el estado del repositorio nuevamente.

5. Ahora, utiliza `git push` para escribir tus cambios en GitHub. Luego, revisa el estado del repositorio y después ve el contenido del repositorio en un *browser* (también conocido como navegador) y verifica tus cambios.

El segundo *commit* y *push*

Haremos un segundo *commit* dentro del repositorio personal.

1. En la carpeta `Actividades/AC00/` crea un archivo `bienvenida.py` cuyo código imprima en pantalla "Bienvenid@ a IIC2233".

Los siguientes pasos son iguales a la sección anterior, pero para el archivo `bienvenida.py`.
2. Revisa el estado del repositorio utilizando `git status`. Observa que tu archivo recién creado aparece bajo "*Untracked files*".
3. Agrega el archivo `bienvenida.py` al *staging area* mediante `git add`. Recuerda que debes escribir el nombre del archivo después de este comando. Luego revisa el estado del repositorio y verifica que se haya agregado.
4. Utiliza `git commit` de los cambios realizados. Recuerda escribir un mensaje **descriptivo**. Luego revisa el estado del repositorio.
5. Ahora, utiliza `git push` para escribir tus cambios en GitHub. Luego, revisa el estado del repositorio y después ve el contenido del repositorio en un *browser* (también conocido como navegador) y verifica tus cambios.

Parte 3: Actualizar tus repos locales

Hay un comando que no explicamos durante la clase, y es `git pull`. Ya hablamos de como agregar cambios locales y enviarlos a la versión remota de nuestro repositorio, pero no hemos hablado de como **traer cambios** del repositorio remoto a nuestro repositorio local.

Lo anterior, asume que posterior a que clonamos un repositorio, como "*syllabus*" o "contenidos", se hayan subido cambios que queremos ver en nuestro repositorio local. **La solución no es clonar cada vez, si no que actualizar la versión ya clonada.** Para eso es `git pull`.

Actualizar *syllabus*

Actualizaremos los contenidos del repositorio del curso *syllabus*. Para lograrlo, tu consola deberá estar dentro de la carpeta del repositorio.

- Primero, revisa el contenido de *syllabus* en tu computador, notarás que no hay muchos archivos, solo podrás ver este enunciado.
- Debemos esperar a que los ayudantes actualicen el repositorio *syllabus* del curso. Puedes revisar en github.com/IIC2233/syllabus en el navegador. Si ves que hay un archivo de extensión `.py` en la carpeta AC00, significa que ya es tiempo de actualizar tu repositorio local.
- Antes, en consola, revisa el estado del repositorio.
- Ahora, ejecuta el comando `git pull`. Este último debería descargar cambios que fueron subidos por los ayudantes a este repositorio.

- Si ves el contenido en tu computador nuevamente, te encontrarás con un nuevo archivo llamado `base.py`.

Ahora, deberás trabajar sobre el archivo entregado, pero lo harás en tu repositorio personal. Debes partir del archivo `base.py` que encontrarán en la carpeta `Actividades/AC00/` disponible en el *syllabus* después de actualizar. **Luego este copiarlo a su repositorio de alumno en la carpeta `Actividades/AC00/`, trabajar sobre él.**

Un poco de programación

Deberás usar clases para modelar **triángulos** y **cuadrados** en un plano cartesiano bidimensional. Para esto, será necesario guardar en qué coordenadas están situadas las figuras y cuáles son sus dimensiones. Ambas figuras deberán tener métodos para obtener su área y su perímetro. Para el caso de triángulo, debe haber un método que indique si es equilátero. Por último, ocupe el método `__str__` para que, al imprimir en pantalla una instancia de estas clases, se lea qué tipo de figura es la instanciada y cuáles son sus dimensiones.

Una vez que termines esto, debes subir todos tus cambios, como aprendiste en la Parte 2, a tu repositorio personal.

Notas

- Haz un *commit* cada vez que una idea lógica esté terminada.
- Recuerda que Python es un lenguaje con baterías incluidas: no dudes en utilizar el módulo *built-in* llamado `math`, puesto que ofrece múltiples funciones que podrían ser de utilidad para esta actividad.

Requerimientos

Esta sección aparecerá en todas las actividades en clase, y detalla todas las cosas que se espera realices o entregues al finalizar la actividad.

- Actualizar `README.md` de tu repositorio personal.
- *Script* `bienvenida.py` que imprima “Bienvenid@ a IIC2233”
- Clase `Triangulo` en archivo `base.py`
 - Crear el inicializador (método `__init__`)
 - Crear el método `obtener_area`
 - Crear el método `obtener_perimetro`
 - Crear el método `es_equilatero`
 - Crear el método `__str__`
- Clase `Cuadrado` en archivo `base.py`
 - Crear el inicializador (método `__init__`)
 - Crear el método `obtener_area`
 - Crear el método `obtener_perimetro`
 - Crear el método `__str__`
- Crear una instancia por cada clase

Entrega

- **Lugar:** En su repositorio privado de GitHub, en la **carpeta** `Actividades/AC00/`
- **Hora del *push*:** 16:30