

# ***Programación Avanzada***

## **II C2233**

Nebil Kawas – Jaime Castro – Felipe del Río



# Agenda

- Programa del curso
- Equipo docente
- Herramientas del curso
- Recomendaciones para pasar el curso
- Actividad para aprender a usar Git y GitHub

# Programa



**<https://iic2233.github.io>**

**Página del curso**

**Este curso enseña técnicas para diseñar, implementar, ejecutar y evaluar software que resuelva problemas a partir de especificaciones detalladas.**

# Objetivos

1. **Descomponer problemas complejos**, para diseñar sus soluciones.
2. Crear **diseños orientados a objetos**.
3. Aplicar conceptos de **OOP** y **estructuras de datos**, para diseñar y escribir programas complejos en Python, pudiendo **extender este conocimiento** a distintos lenguajes.
4. **Usar herramientas de programación estándares**; técnicas de programación; y un entorno de desarrollo de software para editar, ejecutar y depurar programas.
5. **Generar software desde cero**, con código de alto nivel y calidad, y con interfaces gráficas totalmente funcionales.

# Contenidos del curso

## Fundamentos de programación

- Estructuras de datos *built-ins*
- Programación funcional
- Programación orientada a objetos
- Estructuras de datos con nodos
- Excepciones
- *Testing*

## Herramientas y patrones de programación

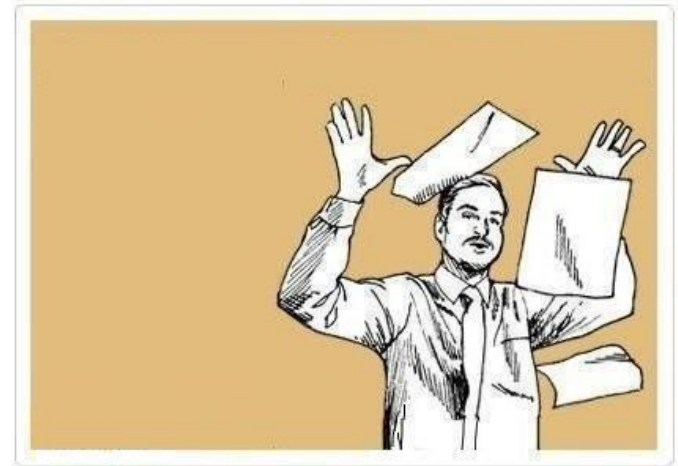
- *Threading*
- Interfaces Gráficas
- I/O (*strings, bytes, serialización*)
- *Networking*
- *Webservices & Regex*

# ***Programación avanzada vs.***

***Exploratorio de computación***



***Introducción a la programación***





# Si no recuerdan IIC1103

En el *syllabus* hay una lista de contenidos para refrescar la memoria:

[github.com/IIC2233/syllabus/blob/master/Contenidos-IIC1103.md](https://github.com/IIC2233/syllabus/blob/master/Contenidos-IIC1103.md)

# Comenzar su formación como desarrolladores de software



# Metodología



FLIPPED

---

CLASSROOM

# ***Flipped Classroom***



# ¿Cómo funciona?

## Antes de la clase

### Viernes en la noche

Los profesores suben material para que lo **estudien** y resuelvan dudas en las *issues* del foro del curso.

### Martes 14:00

Habrà una ayudantía para **reforzar** los contenidos y para resolver dudas con los ayudantes.

### Miércoles

El coordinador subirá **puestos asignados** en el [syllabus](#), para la clase del día siguiente.

# ¿Cómo funciona?

Durante la clase: jueves 14:00 — 16:40

## Primera media hora

Se abre una sesión de **repaso** de los contenidos de la semana, para resolver dudas con el profesor.

## Después

Habrà una **actividad** para aplicar los contenidos y **resolver dudas** con los ayudantes y el profesor.

# Todas las clases tienen **una** evaluación

Esta podrá ser, según  
calendario:

- La **actividad** de ese día,  
o bien,
- Un **control** que  
empezará a las 16:15,  
aproximadamente.

---



# Actividades evaluadas

- Serán de contenidos no evaluados en alguna actividad anterior, **incluyendo siempre** los de la semana.
- Los ayudantes **pasarán lista**. Si están marcados como ausentes, tendrán un 1.0 en la actividad.
  - Es su responsabilidad verificar que los ayudantes los hayan puesto presentes.
- **Tendrán hasta las 16:40 para entregar la actividad**. Luego, el profesor podrá quedarse para resolver dudas y/o mostrar la solución de la actividad, si es que hay interesados.
- No se recuperan actividades.

# Controles

- Serán de contenidos no evaluados en algún control anterior, **incluyendo siempre** los de la semana. Además, tendrá una pregunta relacionada con la actividad no evaluada de ese día.
- Enfocados en conceptos y en lectura de código.
- Normalmente **empezarán a las 16:15**, y terminarán a las 16:40.
- No se recuperan controles.

# Tareas

Objetivo: resolver un problema complejo

Lunes 00:01

Publicación

**Lean primero**, y luego empiecen a desarrollar la solución.

Viernes o sábado

Entregable pequeño

El entregable es para ver cómo van, y es obligatorio.  
Recibirán un *feedback* general dentro de 2 días.

Plazo de entrega final

Entrega de código

Sigan haciendo su tarea hasta el plazo designado.

# Examen

- El curso contempla un examen escrito de conocimientos relevantes.
- Habrán preguntas de conceptos, modelación, lectura de código, y escritura de código.
- Será el **7 de diciembre a las 15:30**, y durará ~3 horas.

# Nota del curso

- 14 evaluaciones en clases (**EC**)
- 5 o 6 tareas (**T**)
- 1 examen final (**E**)

$$\mathbf{NC} = 0,35 \times \mathbf{EC} + 0,45 \times \mathbf{T} + 0,20 \times \mathbf{E}$$

# Condiciones para aprobar

- Para aprobar el curso, debe cumplir con todas estas condiciones:
  - **EC**  $\geq 3,95$
  - **T**  $\geq 3,95$
  - **E**  $\geq 3,50$
- Si el alumno cumple con todas las condiciones anteriores,  
**NF = NC**. En caso contrario, **NF = min(3,9 ; NC)**

# Cálculo de nota de tareas (T)

**Si inscribe tarea opcional**

$$\frac{\sum_{i=0}^4 p_i \times t_i}{\sum_{i=0}^4 p_i}$$

**En otro caso**

$$\frac{\sum_{i=0}^5 p_i \times t_i}{\sum_{i=0}^5 p_i}$$

Nota tarea $t_i$	Ponderador $p_i$
$t_{00}$	60
$t_{01}, t_{02}, t_{03}, t_{04}$	100
$t_{05}$ (opcional, con inscripción)	100

# Cálculo de nota de evaluaciones en clases (EC)

- Es el promedio simple de las notas obtenidas en estas evaluaciones.
- Si asisten a la evaluación, la nota mínima que obtendrán es 1,5.
- En ese cálculo eliminaremos:
  - Las dos peores notas.
  - La tercera peor nota si faltaron a lo más a 2 evaluaciones en clases.



# Evaluaciones: otras consideraciones

- La inasistencia a alguna de las evaluaciones (actividad, control y examen) se evalúa con nota 1.0.
- **NO se borrará ninguna otra evaluación aparte de las estipuladas anteriormente.**
- La nota **NF** se calculará con un decimal.  
El resto de las notas serán calculadas con dos decimales.

# Correcciones y recorreciones

- Las notas de una evaluación se publican **a más tardar 15 días hábiles** después de haberse realizado.
- Tendrán **una semana** para corregir después de que se publiquen las notas.
- Las evaluaciones se recorren a través de un formulario que les publicarán los ayudantes para cada caso.

# Solicitud de corrección

- Recuerde que la corrección la revisa un ser humano.
- La nota **puede bajar**.
- **Debe estar correctamente justificada**. No se aceptarán cosas del tipo:
  - “Me dieron medio puntaje. No tengo todo, pero merezco más”.
  - “No respeté el PEP-8, pero nunca me habían descontado antes”.
  - “Si bien no puse nada en mis clases, me deberían dar puntaje por haberlas definido”.
  - “Aunque la tarea dice que solo se corrige lo que se ve en la interfaz gráfica, deberían corregir lo que hice aunque no se vea en la pantalla”.
- La solicitud debe indicar cuáles fueron los puntos mal corregidos y por qué están mal corregidos.

# ¿Qué pasa si no estoy de acuerdo con la respuesta de la corrección?

- En este caso, deberán solicitar mediante un *form* que los profesores evalúen el caso.
- La respuesta va a demorar, aunque haremos el mejor esfuerzo.
- Esto implica la re-revisión completa de la evaluación.
- El resultado de este proceso es inapelable.

La recorreción del examen y de la última tarea es **presencial**, el día **lunes 10 de diciembre**.

**NO HAY OTRA FECHA**

# Integridad académica

*“Cualquier situación de copia en alguna evaluación tendrá como **sanción un 1,1 final en el curso**. Esto sin perjuicio de sanciones posteriores que estén de acuerdo a la Política de Integridad Académica de la Escuela de Ingeniería y de la Universidad, que sean aplicables para el caso.”*

También aplica la política de integridad académica del Departamento de Ciencia de la Computación (DCC), disponible como anexo en el programa del curso.

# Integridad académica

- Deben **indicar la fuente** de cualquier código que encuentren en internet y que usen en sus tareas y/o actividades.
- Deben indicar si están usando código del material del curso o de las ayudantías.
- Si no lo hacen, se considerará **plagio**.

# Integridad académica

- Las evaluaciones de este curso se consideran **estrictamente individuales**, a menos que se indique lo contrario.
- Compartir todo o parte de la respuesta a una evaluación del curso, o indicar inequívocamente cómo llegar a ella, se considera **copia**.



# Integridad académica

/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/4/raw/ [redacted] (68%)	<div><div></div></div>	/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/4/raw/ [redacted] (73%)	<div><div></div></div>
<a href="#">4-71</a>	<div><div></div></div>	<a href="#">2-66</a>	<div><div></div></div>
<a href="#">95-111</a>	<div><div></div></div>	<a href="#">90-106</a>	<div><div></div></div>
<a href="#">74-91</a>	<div><div></div></div>	<a href="#">69-86</a>	<div><div></div></div>
<a href="#">115-132</a>	<div><div></div></div>	<a href="#">110-127</a>	<div><div></div></div>

```
/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/4/raw/ [redacted]
>>> file: LongJump.py
# [redacted]

print("***** Long Jump Information System *****")
print("Please enter the names of competitors. (Press return when done.)")
print("Competitor no. 1:")
competitor = input()
b,c,g,h,d,k = 1,0,0,0,[],0
maxi,competitors = [],[competitor]
while True:
    b += 1
    print("Competitor no. "+str(b)+":")
    competitor = input()
    if competitor == "":break
    else:
        competitors.append(competitor)
print("Please enter the distances for each competitor.")
for each in competitors:
    print("Competitor " + each + " sep="")
    at1 = input("Attempt 1:\n")
    at2 = input("Attempt 2:\n")
    at3 = input("Attempt 3:\n")
    x = (at1+at2+at3).lower()
    if (at1+at2+at3).find("oul") != -1:
        x = (at1+at2+at3).lower()
    d.append(at1)
    d.append(at2)
    d.append(at3)
    maxi.append(max(eval(at1),eval(at2),eval(at3)))
```

```
/home/ubuntu/Projects/work/2015/uct-csc1010h/tutorials/4/raw/ [redacted]
>>> file: LongJump.py
# [redacted]

print("***** Long Jump Information System *****")
print("Please enter the names of competitors. (Press return when done.)")
print("Competitor no. 1:")
competitor = input()
b,c,g,h,d,k = 1,0,0,0,[],0
maximums,competitors = [],[competitor]
while True:
    b += 1
    print("Competitor no. "+str(b)+":")
    competitor = input()
    if competitor == "":break
    else:
        competitors.append(competitor)
print("Please enter the distances for each competitor.")
for each in competitors:
    print("Competitor " + each + " sep="")
    attempt1 = input("Attempt 1:\n")
    attempt2 = input("Attempt 2:\n")
    attempt3 = input("Attempt 3:\n")
    g = (attempt1+attempt2+attempt3).lower()
    if (attempt1+attempt2+attempt3).find("oul") != -1:
        g = (attempt1+attempt2+attempt3).lower()
    d.append(attempt1)
    d.append(attempt2)
    d.append(attempt3)
    if "foul" in g:
        maximums.append(max(eval(attempt1),eval(attempt2),eval(attempt3)))
    else:
        d.remove("foul")
        if not "foul" in d:
```

# Fechas

**<https://iic2233.github.io/calendario/>**

# Cuerpo docente



Nebil (S1)



Jaime (S2)



Felipe (S3)

# Ayudantes jefes



Hernán



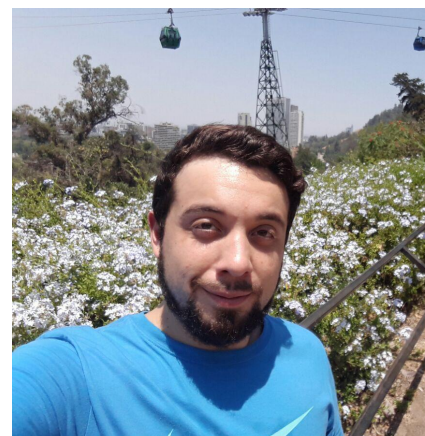
Fernando



Rodrigo



Enzo



Benjamín

# Ayudantes

## Tareas

- **Pablo Flores**
- **Jessica Hormazabal**
- **Camilo López**
- **Ricardo Schilling**
- Joaquín Araya
- Javiera Bao
- José T. Caraball
- Camila Chávez
- Daniela Concha
- Ignacio Contreras
- Fernando de Diego
- Agustín Krebs
- Vicente Lira
- Norman Oriden
- Francisco J. Ossandón
- Pablo Rodríguez
- Juan Schuwirth

## Docencia

- **Franco Bruña**
- **Gabriel Lyon**
- **Pablo Olea**
- Vicente Aguila
- Juan Aguillón
- Paula Arellano
- Fernando Duarte
- Jacques Hasard
- Paul Heinsohn
- Santiago Laguna
- Benjamín Martínez
- Dante Pinto
- Nicolás Quiroz

# Consultas

- Administrativas:

**iic2233@ing.uc.cl**

- Sobre contenidos del curso, enunciados y pautas:

<https://github.com/IIC2233/syllabus/issues>

**NO MANDEN MAILS A LAS DIRECCIONES PERSONALES**

**A menos que sea algo personal**

# Herramientas del curso



# Python 3.6

<https://www.python.org/>

<https://zen-of-python.info/>





Guido van Rossum, creador de Python, en la convención OSCON 2006. Fuente: [Wikipedia](#).

# PEP8

Guía de estilo

---

# PEP8

- *Python Enhancement Proposal 8* es la guía de estilo de Python
- Se usa para hacer más legible y consistente el código
- En las tareas controlaremos el respeto por algunos aspectos de esta guía de estilo
- <https://www.python.org/dev/peps/pep-0008/>

# PEP8: algunas cosas

- **Nombres de variables descriptivos.**
- *Imports* al comienzo del módulo.
- Un espacio después de “,” y a cada lado de los operadores.
- Líneas de máximo 80 caracteres (incluyendo espacios).
- **PyCharm no respeta esta restricción por defecto, deben configurarlo.**
- **No** usar *tabs*. Usar (4) espacios para indentar.

# CamelCase y snake\_case

```
CONST_PI = 3.1415
```

```
class ClaseDeEjemplo:  
    def __init__(self, parametro):  
        self.variable_de_ejemplo = parametro  
  
    def metodo_de_ejemplo(self):  
        return 1 + 1 == 2
```

**El código se lee más veces  
de lo que se escribe, y que  
es otro el que lo va a leer.**

# Modularización

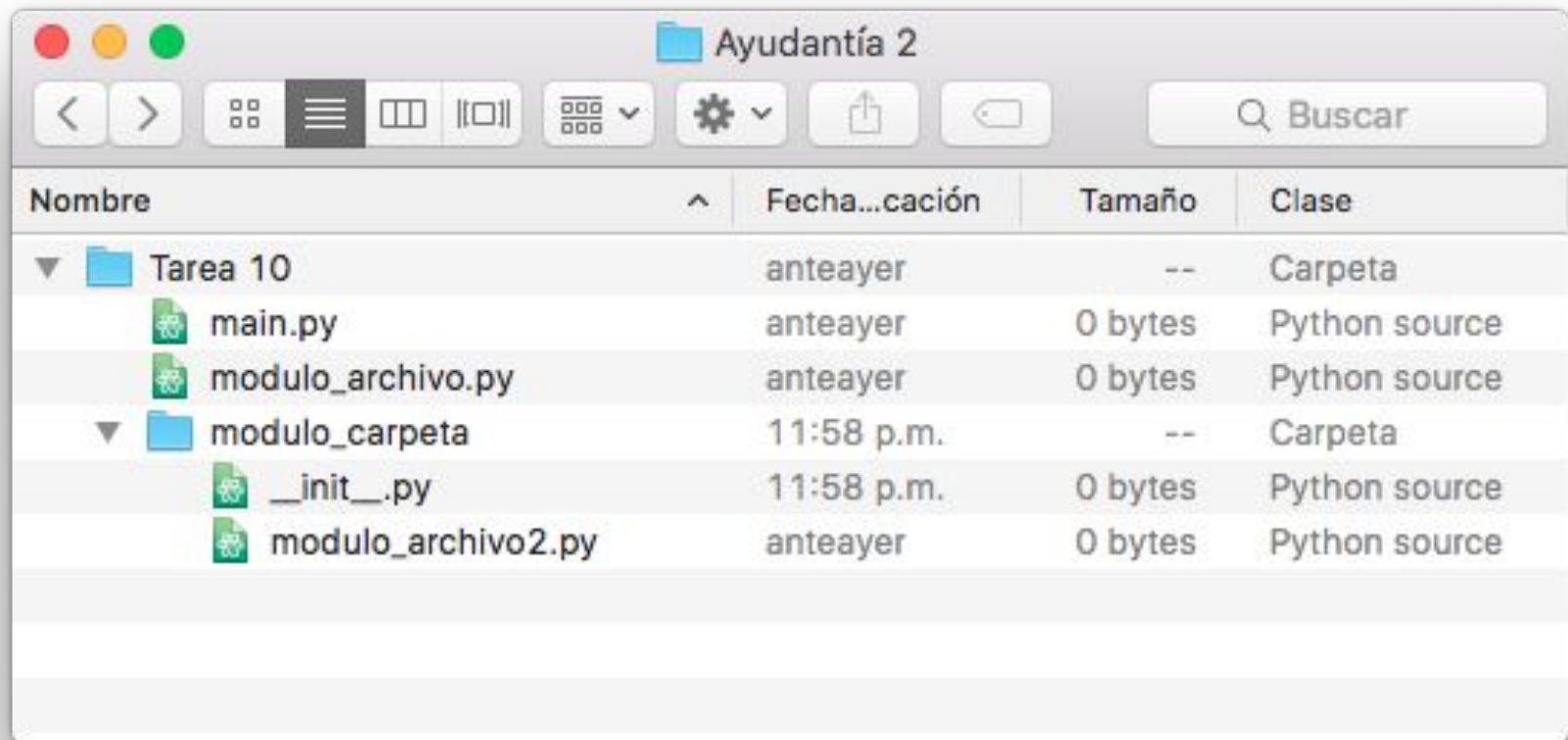
—

# Modularización: ¿por qué?

- Cuando un programa crece, se hace inviable mantenerlo en un solo archivo:
  - El mantenimiento es difícil
  - El trabajo en equipo es difícil
  - Es desordenado
- Un módulo es un archivo de Python normal, y puede tener:
  - Variables
  - Métodos
  - Clases



# Modularización



# Cómo usar módulos

Importándolo entero

```
import modulo_archivo
```

```
if __name__ == '__main__':
```

```
    variable_tipica = modulo_archivo.VALOR_FIJO
```

```
    objeto_tipico = modulo_archivo.Clase()
```

```
    modulo_archivo.funcion()
```

# Cómo usar módulos

Importándolo entero con un alias

```
import modulo_archivo as ma
```

```
if __name__ == '__main__':  
    variable_tipica = ma.VALOR_FIJO  
  
    objeto_tipico = ma.Clase()  
  
    ma.funcion()
```

# Cómo usar módulos

Importando lo necesario

```
from modulo_archivo import VALOR_FIJO, Clase, funcion
```

```
if __name__ == '__main__':  
    variable_tipica = VALOR_FIJO  
  
    objeto_tipico = Clase()  
  
    funcion()
```

# Cómo usar módulos

- Cuando se importa un módulo se ejecuta todo el código en él.
- Para evitar que se ejecute código de un módulo al ser importado se utiliza el siguiente `if`:

```
# Código del módulo
```

```
if __name__ == '__main__':
```

```
    # Mucho código escrito
```

# Cómo **NO** usar módulos

Importando todo sin referencia al módulo

```
from modulo_archivo import *
```

```
if __name__ == '__main__':  
    variable_tipica = VALOR_FIJO  
    objeto_tipico = Clase()  
    funcion()
```



# Cómo **NO** usar módulos

- **Evita** crear módulos que se llamen igual a los que vienen incluidos en Python.
- ¿Cómo Python busca los módulos?
  1. Módulo de la librería estándar
  2. Módulo en la misma carpeta
  3. Módulo en el directorio de instalación

# Jupyter Notebook





# Jupyter Notebook

- Es una aplicación web que permite crear documentos interactivos con código, gráficos y texto explicativo.
- Es el formato de los apuntes del curso.
- Se recomienda **bajar los apuntes e interactuar con el código**, no solo leerlo desde la página.
- **No deben usarlo para programar sus actividades ni tareas.**
- Instrucciones para instalar: <https://jupyter.org/install.html>.



Google

+



**stackoverflow**

# ¿Cómo buscar soluciones?

python [versión] [librería] [duda]



¡En inglés!

¿Cómo imprimir una cola con Python? **X**

python 3.6 collections print queue



# ¿Cómo buscar soluciones?

python [versión] [error]



¡En inglés!

NameError: name "MiVariable" is not defined



NameError: name \* is not defined





python3.5 NameError: name \* is not defined



[Todos](#) [Videos](#) [Maps](#) [Imágenes](#) [Noticias](#) [Más](#) [Preferencias](#) [Herramientas](#)

Sez de 95,800 resultados (0.50 segundos)

[In Python3.5:NameError: name 'image\\_to\\_string' is not defined](#)

[https://stackoverflow.com/.../in-python3-5nameerror-name-image...](https://stackoverflow.com/.../in-python3-5nameerror-name-image-...) ▼ Traducir esta página

11 jun. 2017 - Please post your source code so we can look over the code and get more details. Also your error is caused by a variable declaration without a ...

[oop - Python3 NameError: name 'method' is not defined - Stack Overflow](#)

[https://stackoverflow.com/.../python3-nameerror-name-method-is...](https://stackoverflow.com/.../python3-nameerror-name-method-is-...) ▼ Traducir esta página

18 mar. 2016 - consider you have the function **defined** in the global scope: def recursive(x): if (x>5): print (x) recursive(x - 1). you would simply call this with ...

[input\(\) error - NameError: name '...' is not defined - Stack Overflow](#)

[https://stackoverflow.com/.../input-error-nameerror-name-is-not...](https://stackoverflow.com/.../input-error-nameerror-name-is-not-...) ▼ Traducir esta página

14 ene. 2014 - input\_variable = input ("Enter your name: ") print ("your name is" + input\_variable) ...  
input ("Enter your name: ") File "<string>", line 1, in <module> **NameError: name 'dude' is not defined**  
... I did what Kevin said and it is version 2.7.5! ... If you are using **Python 3.x**, raw\_input has been renamed to input .

[python NameError: name 'file' is not defined in python 3.5 - Stack ...](#)

[https://stackoverflow.com/.../python-nameerror-name-file-is-not...](https://stackoverflow.com/.../python-nameerror-name-file-is-not-...) ▼ Traducir esta página

26 nov. 2015 - Traceback (most recent call last): File "c:\python3.5\lib\runpy.py", line .... python 3.x from  
is Q: python **NameError: name 'file' is not defined** But ...

[python 3.x - NameError: name 'value' is not defined - Stack Overflow](#)

[https://stackoverflow.com/.../nameerror-name-value-is-not-defined...](https://stackoverflow.com/.../nameerror-name-value-is-not-defined-...) ▼ Traducir esta página

3 abr. 2014 - **NameError: name 'value' is not defined** ... A variable defined in a function isn't visible outside the function. ... answered Apr 5 '14 at 2:39.

[NameError: global name 'unicode' is not defined - in Python 3 - Stack ...](#)

[https://stackoverflow.com/.../nameerror-global-name-unicode-is...](https://stackoverflow.com/.../nameerror-global-name-unicode-is-...) ▼ Traducir esta página

9 nov. 2013 - **Python 3** renamed the unicode type to str , the old str type has been replaced by bytes . if  
isinstance(unicode or str, str): text = unicode or str ...

# Git



# ¿Qué es Git?

Git es un sistema distribuido de control de versión, gratuito y *open source*, diseñado para manejar de pequeños a enormes proyectos de forma rápida y eficiente<sup>1</sup>.



---

<sup>1</sup> <https://git-scm.com/>



# Ventajas

- Versiones disponibles en cualquier momento.
- Control de cambios.
- Programar versiones en paralelo y luego juntarlas.
- Múltiples *backups* de sus programas.
- Trabajo en equipo fluido, sin problemas como en Dropbox.

Se usa en la vida real.

Es **obligatorio** conocerlo.

# ¿Qué es GitHub?

Es una plataforma para alojar proyectos, usando el sistema de control de versiones Git.

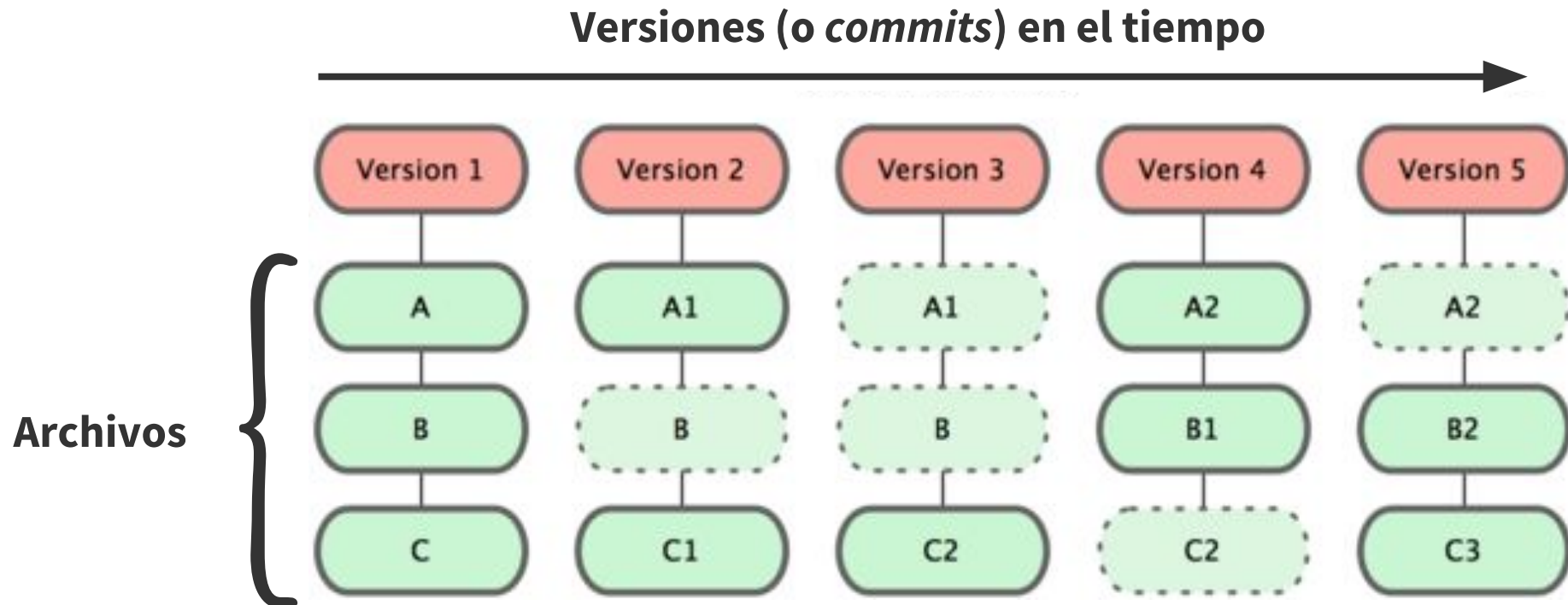


# ¿Cómo funciona Git y GitHub?



# Conceptos

1. **Commit o versión:** un estado, que contiene archivos y carpetas con cierto contenido.



# Conceptos

2. ***Working directory***: carpeta local con la que se trabaja directamente, creando, cambiando o eliminando archivos.
3. ***Staging area***: creaciones, modificaciones o eliminaciones de archivos que serán incluidas en un nuevo *commit* o versión.
4. **Repositorio local (o repo)**: carpeta local que contiene todas las versiones o *commits*.
5. **Repositorio remoto**: servidor, en nuestro caso GitHub, que contiene todas las versiones o *commits*. **¡Este es el lugar que los ayudantes revisarán!**

# Lo primero es lo primero

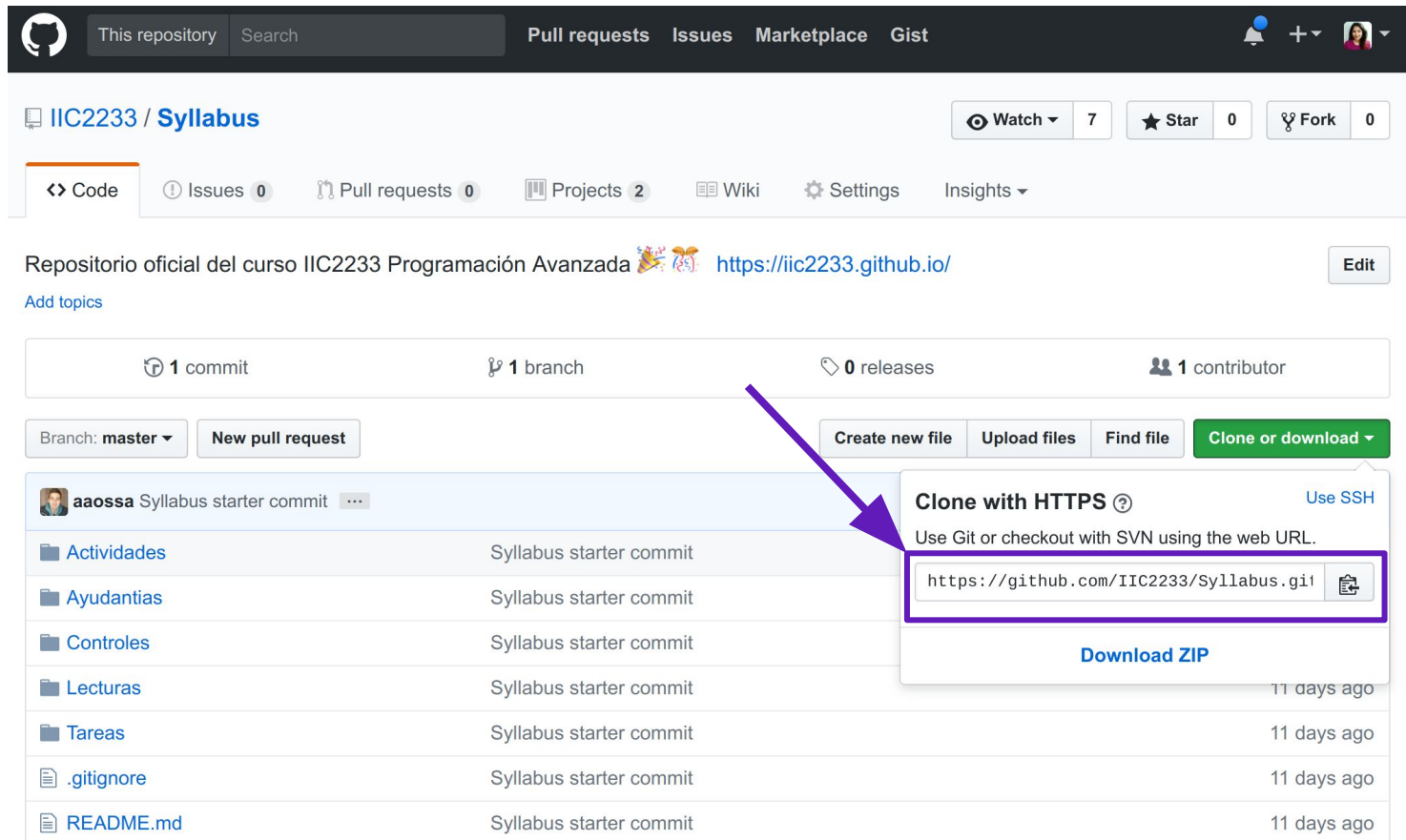
Clonaremos un repositorio

- Para empezar a hacer cosas con un repositorio, hay que tener uno.
- Podríamos crear uno, pero mejor usemos el que les dieron los ayudantes.

---

# Obtener dirección

1. Haberse registrado en el curso correctamente
2. Ir a <https://github.com/IIC2233/<mi-usuario>-iic2233-2018-2>



The screenshot shows the GitHub repository page for **IIC2233 / Syllabus**. The repository has 7 watches, 0 stars, and 0 forks. The main content area shows the repository description: "Repositorio oficial del curso IIC2233 Programación Avanzada" with a link to <https://iic2233.github.io/>. Below this, there are statistics: 1 commit, 1 branch, 0 releases, and 1 contributor. The "Clone or download" button is highlighted with a purple arrow pointing to the "Clone with HTTPS" dropdown menu. The dropdown menu shows the URL `https://github.com/IIC2233/Syllabus.git` and a "Download ZIP" button. The repository files list includes: Actividades, Ayudantias, Controles, Lecturas, Tareas, .gitignore, and README.md, all from the "Syllabus starter commit" 11 days ago.

File	Commit	Time
Actividades	Syllabus starter commit	11 days ago
Ayudantias	Syllabus starter commit	11 days ago
Controles	Syllabus starter commit	11 days ago
Lecturas	Syllabus starter commit	11 days ago
Tareas	Syllabus starter commit	11 days ago
.gitignore	Syllabus starter commit	11 days ago
README.md	Syllabus starter commit	11 days ago



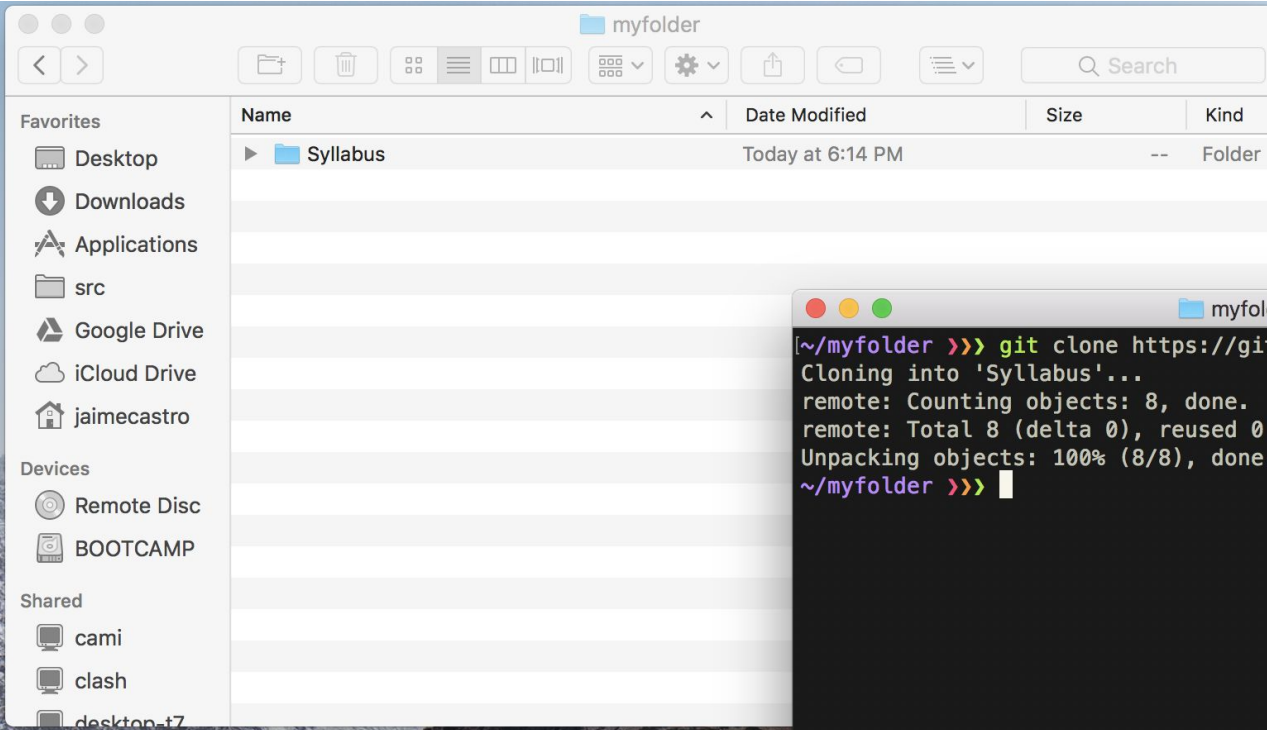
# Clonar el repositorio

Escribir en la consola

```
git clone url_que_copiaron
```

Recuerden estar en la carpeta en la que quieren mantener el repo.

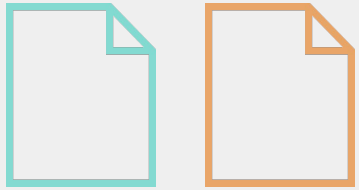
# Clonar el repositorio



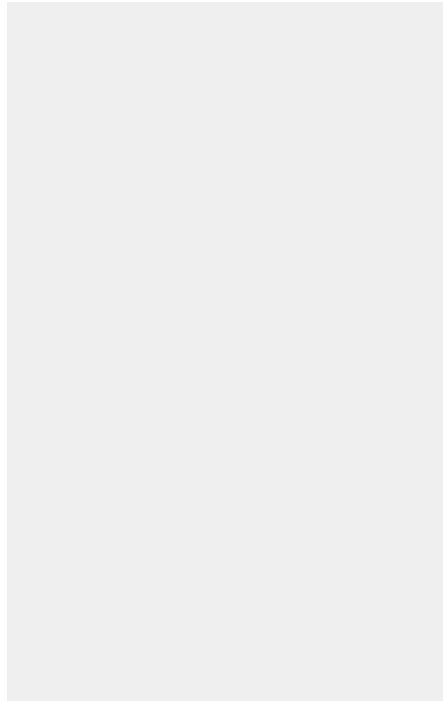
```
myfolder — -zsh — 80x23
[~/myfolder >>>] git clone https://github.com/IIC2233/Syllabus.git
Cloning into 'Syllabus'...
remote: Counting objects: 8, done.
remote: Total 8 (delta 0), reused 0 (delta 0), pack-reused 8
Unpacking objects: 100% (8/8), done.
~/myfolder >>> |
```

# Estado inicial

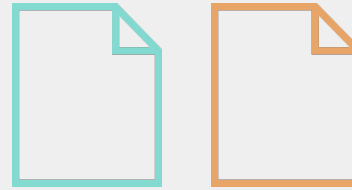
*Working directory*



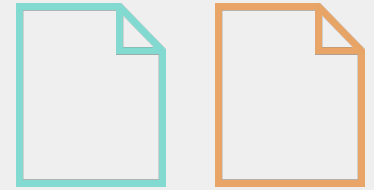
*Staging area*



**Repositorio local**  
(Última versión)



**Repositorio remoto**  
(Última versión)



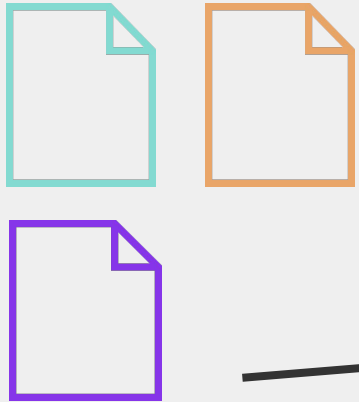
# Creé un nuevo archivo. ¿Cómo lo “subo” al repositorio remoto?

*Working directory*

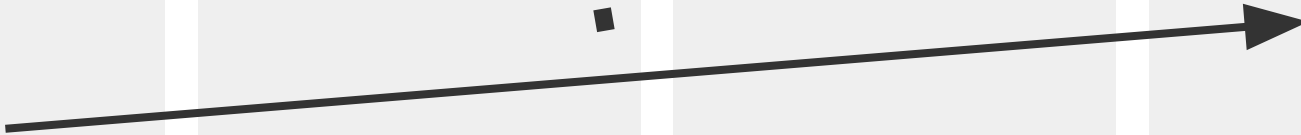
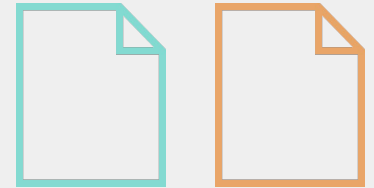
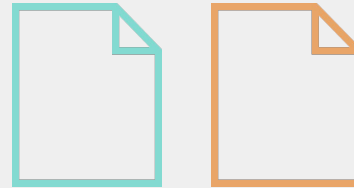
*Staging area*

Repositorio local  
(Última versión)

Repositorio remoto  
(Última versión)



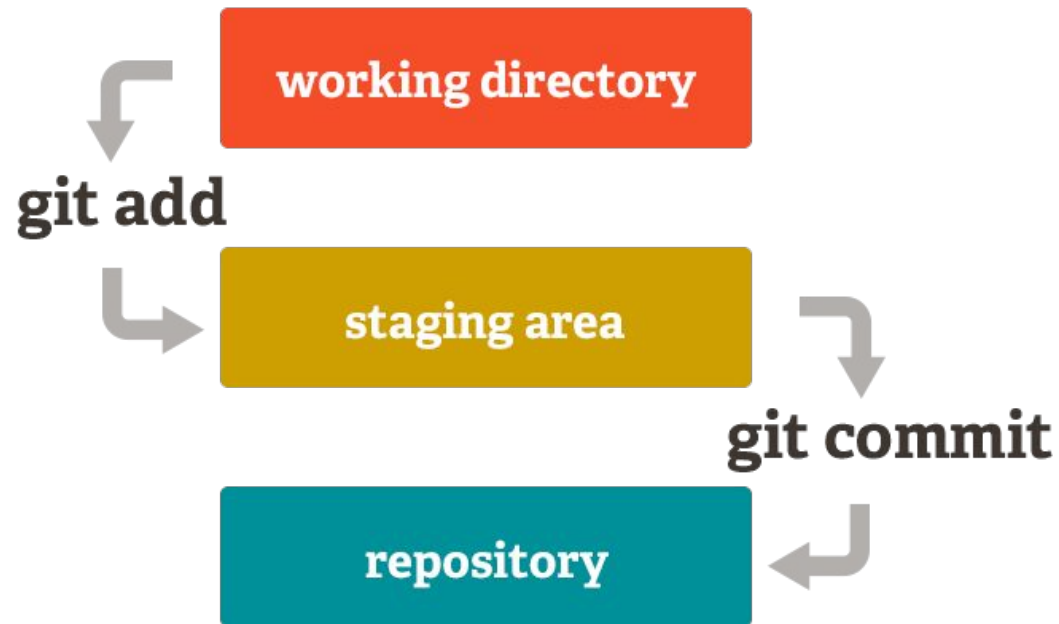
?



# Creé un nuevo archivo. ¿Cómo lo “subo” al repositorio remoto?

1. Crear una versión con el nuevo archivo en el repositorio local.
2. Hacer que mi repositorio remoto tenga las versiones que tengo en el repositorio local.

# Crear una versión



# Crear nueva versión

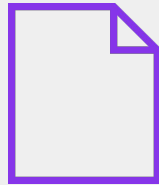
Primero, agregamos el archivo al *staging area*

*Working directory*

*Staging area*

Repositorio local  
(Última versión)

Repositorio remoto  
(Última versión)

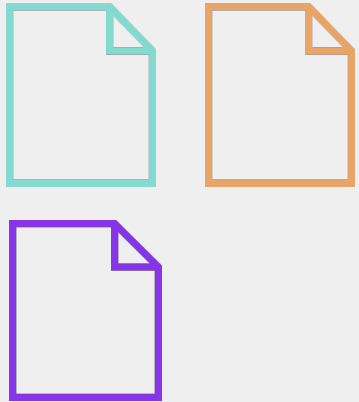


```
git add archivo.py
```

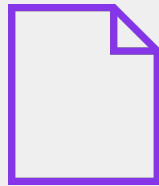
# Crear nueva versión

## Ahora sí, creamos la versión

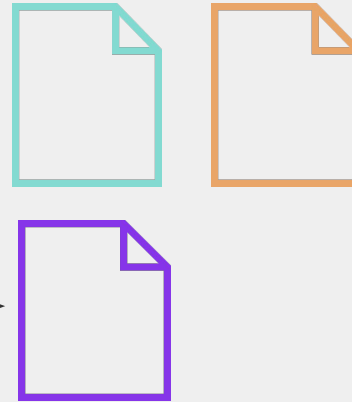
*Working directory*



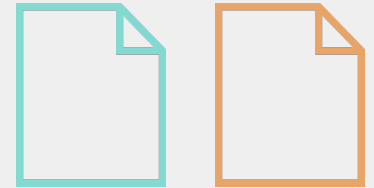
*Staging area*



Repositorio local  
(Última versión)



Repositorio remoto  
(Última versión)



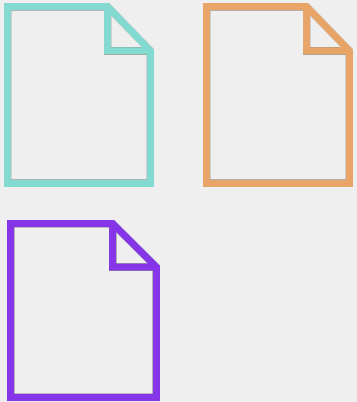
```
git commit -m "Mensaje descriptivo"
```



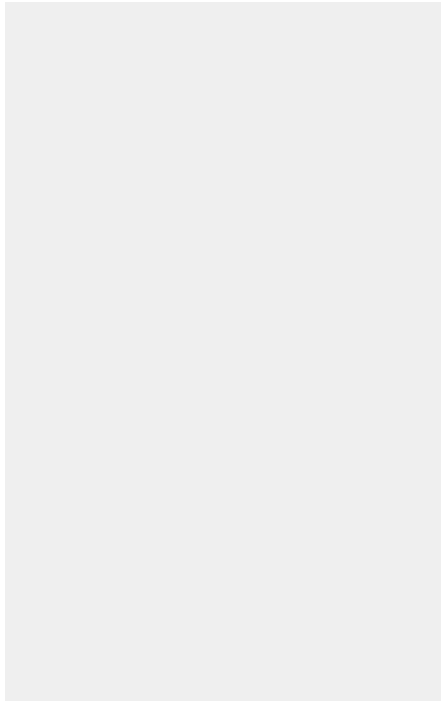
# Crear nueva versión

## Así queda con la nueva versión creada

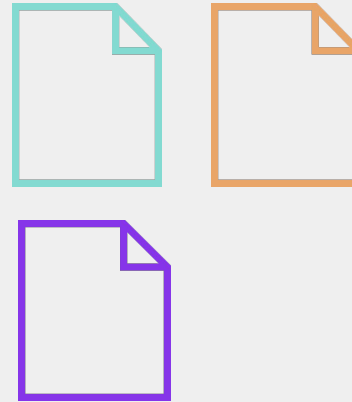
*Working directory*



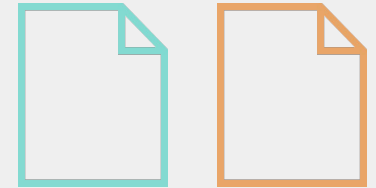
*Staging area*



**Repositorio local**  
(Última versión)



**Repositorio remoto**  
(Última versión)

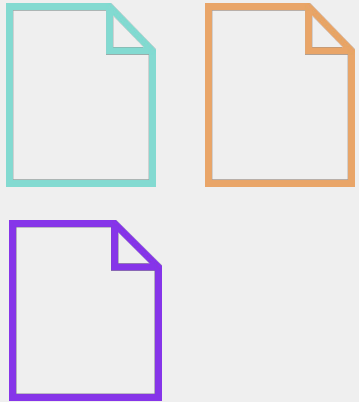


Los mensajes son **muy importantes**. Son una ayuda a ustedes en el futuro.

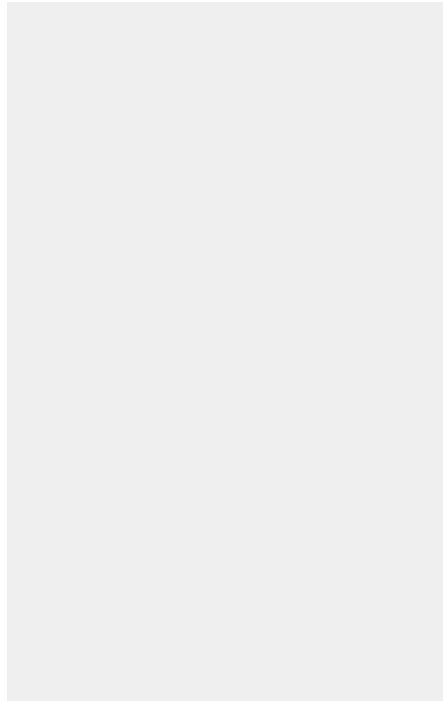
Revisen esta [guía de estilo](#)

# Subir lo que tengo en el repo local a GitHub

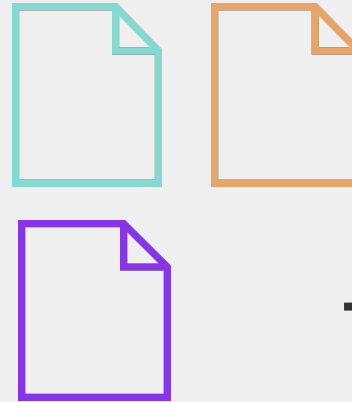
*Working directory*



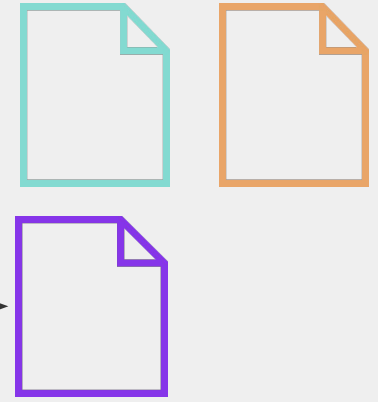
*Staging area*



**Repositorio local**  
(Última versión)



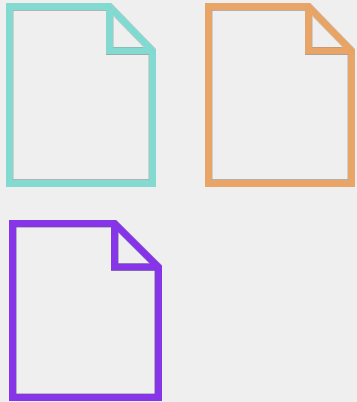
**Repositorio remoto**  
(Última versión)



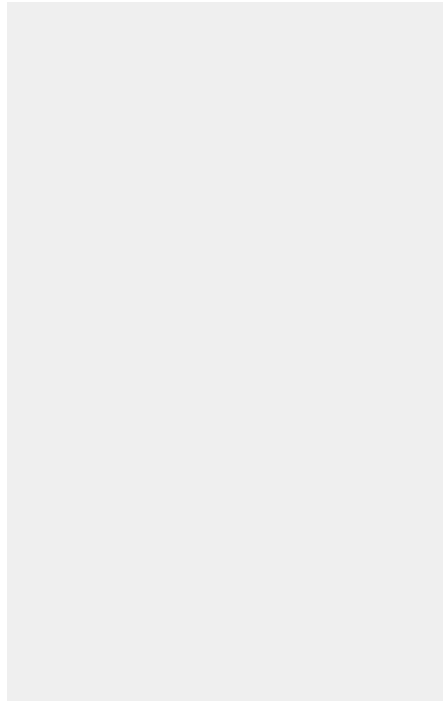
**git push**

# Subir lo que tengo en el repo local a GitHub

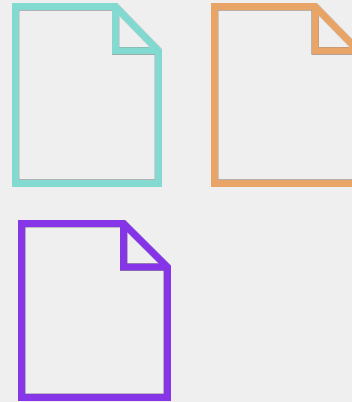
*Working directory*



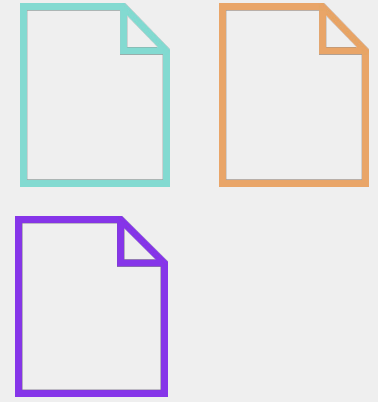
*Staging area*



Repositorio local  
(Última versión)



Repositorio remoto  
(Última versión)



Para las entregas, se toma **la hora del *push* a GitHub.**

## **Sacar algo del *staging area***

Son las 16:38. Las instrucciones dicen que no debo subir el archivo `VeryHeavyFile.txt` que pesa 100 MB.

Hice `git add --all` y solo me queda un minuto para poder subir la actividad.



*I'm having a panic attack.*

```
git reset HEAD file_name
```

**Ya hice *commit***





```
git reset HEAD~
```



# IN CASE OF FIRE

 git commit

 git push

 leave building

**Siempre** hagan  
*commit y push*  
de su trabajo.

- Cada vez que avancen en algo importante de su actividad o tarea.
- Si llevan programando más de media hora.
- Cada vez que paren de programar para dedicarse a otra cosa.

---

**De verdad:**  
**Siempre hagan**  
***commit y push***  
**de su trabajo.**

- Tener su trabajo en GitHub es una copia de seguridad.
- *Shit happens:*
  - Accidentes con líquidos.
  - Robos en Deportes.
  - Fallas de *hardware* o *software*.
  - Cortes de internet.
  - Echar a perder la tarea.
  - Y muchas otras cosas.

---

# Sitios útiles

- [www.git-scm.com](http://www.git-scm.com)
- [Una metaguía de Git](#)
- [Una guía de estilo de \*commits\*](#)

# Otras recomendaciones



# Recomendaciones

- Leer el enunciado cuando lo entreguen, para empezar a programar a más tardar el miércoles de esa semana.
- Buscar más en Google.
- Estudiar e interactuar con el material de clases.
- Ir a las ayudantías.
- Estudiar el ramo en serio desde el principio.
- Ser estratégico con las tareas.
- Dedicarle tiempo a otros ramos.
- Dormir.

# Actividad

1. En el *syllabus*, vayan a la carpeta “Actividades” y descarguen el enunciado de la actividad 0 (AC00).  
<https://github.com/IIC2233/syllabus>
2. Trabajen en la actividad hasta las 16:40.