

ejemplosClase1

March 15, 2022

#

Ejemplos clase Módulo 1

La siguiente notebook contiene ejemplos de la clase del Módulo 1 de la Unidad Curricular **Programación Digital Avanzada** de la carrera de *Ingeniería Biomédica* de la UTEC.

Profesor Adjunto: Mag. Bioing. Baldezzari Lucas

V2022

0.1 Instalando Python

Lo primero que tenemos que hacer para utilizar Python es descargar el intérprete.

Descargamos la versión deseada desde python.org/downloads.

0.2 Anaconda

¿Qué es? → Plataforma Open Source para trabajar con Python (y R) en el procesamiento de datos y ciencias de la computación.

Ventajas → Librerías pre instaladas de Python mayormente utilizadas para el análisis de datos y cálculos numéricos.

Administración de enviroments → Permite la administración de diferentes ambientes de trabajo utilizando el gestor conda. Es posible instalar diferentes versiones de Python, de librerías, etc.

0.2.1 Instalación

Ingresa en anaconda.com/products/individual y descarga la última versión.

0.2.2 Creando un ambiente nuevo

La forma de crear un ambiente nuevo es

```
>> conda create --name miAmbiente
```

Donde el comando *create* le dice a conda que queremos crear un nuevo ambiente utilizando un nombre dado por el argumento *-name* seguido del nombre que deseamos.

Si quisieramos especificar alguna versión de Python en particular, debemos agregar el argumento *python=3.x*. Por ejemplo, si queremos que en el environment se instale la versión 3.10 de Python, haríamos.

```
>> conda create --name miAmbientePy310 python=3.10
```

Empezando con Anaconda Si necesitan profundizar en la instalación y manejo de environments usando Anaconda pueden entrar a [Getting started with conda](#).

Cheatsheet Siempre es bueno tener una “hoja de ayuda” para poder repasar o recordar los comandos más utilizados. Pueden ver la misma en [Conda Cheatsheet](#).

0.3 Jupyter notebook

¿Qué es? → Plataforma Open Source para el desarrollo, documentación y visualización de código, datos, gráficos, entre otros.

Ventajas → Se ejecuta en un explorador web. Permite escribir texto, código ejecutable, gráficos, imágenes, y más.

Environments → Es posible utilizar environments diferentes –creados previamente en nuestra pc- para cada notebook de trabajo.

Ejecutando código con el kernel de Jupyter Es posible ejecutar código y scripts completos de Python usando el Kernel de Jupyter Notebook.

```
[1]: print("Hola Mundo")
```

Hola Mundo

```
[2]: num = range(0,20)
     [num for num in num if num%3 == 0]
```

```
[2]: [0, 3, 6, 9, 12, 15, 18]
```

Editando nuestra Jupyter Notebook. Es posible agregar imágenes, links, diferentes tamaños en la tipografía, diferentes tipografías, cambiar los colores de las fuentes, entre otras cosas utilizando los comandos **Markdown**.

Una guía oficial puede verse en [Markdown Cells](#)

0.4 Sistema de control de versiones (VCS)

- Herramientas que ayudan a rastrear y gestionar cambios en códigos de software.
- Posibilita ir “hacia atrás en el tiempo” y comparar y/o volver a versiones anteriores.
- Trazabilidad. Los cambios de cada desarrollador –sin importar que tan pequeños fueran y el momento en que fueron hechos- son visibles por todo el equipo de desarrollo.
- Flujos independientes de trabajo mediante ramas.

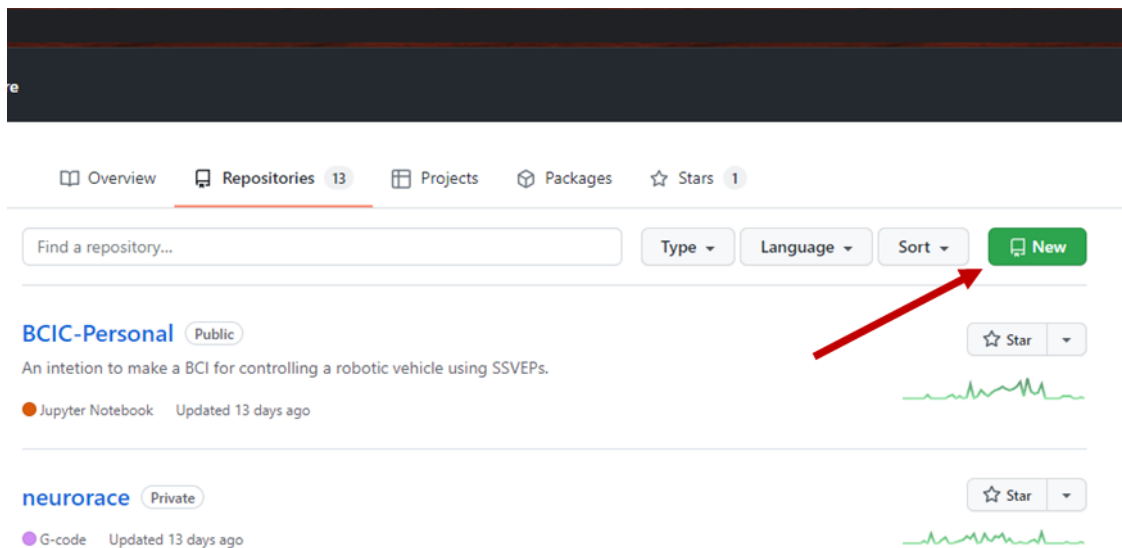
Click [aquí](#) para descargar Git. Siempre es bueno tener a mano la [documentación oficial](#).

0.4.1 Github

- Integración con Git. Es una plataforma con funcionalidades que facilitan el uso de Git y con herramientas adicionales (ej. asignación de tareas mediante issues).
- Alojamiento gratuito. Repositorios públicos y privados.
- Forma sencilla de compartir proyectos y portafolios.
- Posibilidad de usar Github Copilot.

Creando nuestro repositorio La forma más sencilla (y la cual recomiendo al iniciarse en el mundo de git) es crear un repositorio directamente en Github y luego lo clonamos en nuestra PC.

Para esto lo primero que tenemos que hacer es ir a github.com y crear un nuevo repositorio haciendo click en *new*. Ver la imagen debajo.



Se nos abrirá una página donde podremos colocar el nombre de nuestro repositorio, una pequeña descripción, si queremos que sea Público o Privado.


Por otro lado, nos permite seleccionar si queremos que se agregue un archivo *Readme.md*. En general, este archivo es utilizado como una página de *presentación* del repositorio. De momento **no** crearemos el archivo readme.

La figura debajo muestra la creación de un repositorio llamado *repoPrueba* el cual es público.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

 lucasbaldezzari ▾

Repository name *

/ repoPrueba ✓

Great repository names are short and memorable. Need inspiration? How about [shiny-engine](#)?

Description (optional)

Esto es un repositorio de prueba.



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.



Add a README file

This is where you can write a long description for your project. [Learn more.](#)




Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)



Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)





This will set  main as the default branch. Change the default name in your [settings](#).

Create repository

Una vez que hacemos click en *Create repository* se nos abrirá una nueva ventana con información la cual debemos prestar atención.

Veamos la siguiente figura,


Quick setup — if you've done this kind of thing before

 Set up in Desktop or  HTTPS  SSH 

Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.


...or create a new repository on the command line

```
echo "# repoPrueba" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/lucasbaldezzari/repoPrueba.git
git push -u origin main
```



...or push an existing repository from the command line

```
git remote add origin https://github.com/lucasbaldezzari/repoPrueba.git
git branch -M main
git push -u origin main
```



...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Vemos *cuatro* secciones. En la primera, marcada con un color diferente, nos muestra el link (en este caso es `https://github.com/lucasbaldezzari/repoPrueba.git`) al repositorio con el cual podemos *clonar* nuestro repo haciendo desde la consola,

```
C:\myFolder> git clone https://github.com/lucasbaldezzari/repoPrueba.git
```

El comando anterior nos creará una carpeta `repoPrueba` que contendrá el repositorio creado (de momento vacío). Si todo salió bien deberíamos ver algo similar a esto,

```
D:\repos>git clone https://github.com/lucasbaldezzari/repoPrueba.git
Cloning into 'repoPrueba'...
warning: You appear to have cloned an empty repository.
```

A partir de aquí podemos empezar a llenar nuestro repositorio tanto local (en nuestra PC) como remotamente (en github). A medida que avancemos en el curso iremos viendo como trabajar con nuestro repositorio, como sincronizar archivos y ver el historial de cambios sobre los mismos.