

# Trabajo Práctico Final

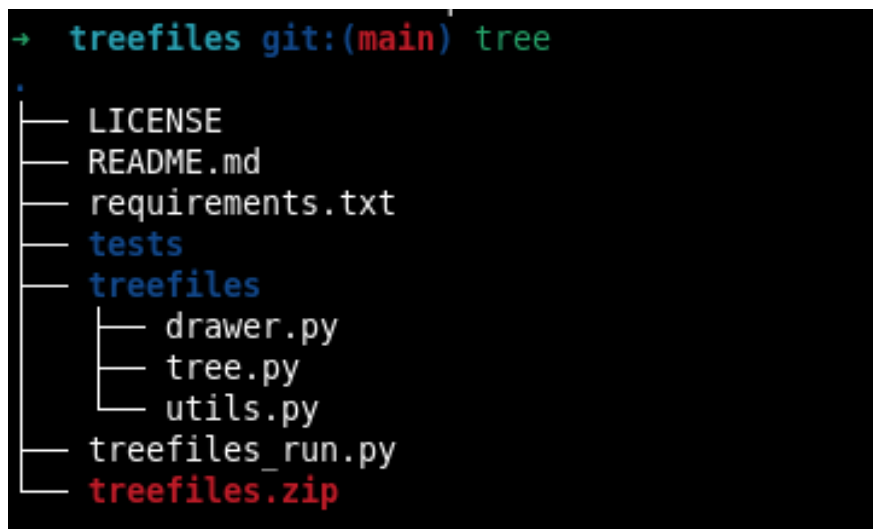
Python Intermedio - UNLAR 2020

19 de diciembre de 2020

## 1. Ejercicio

Subir a la plataforma todo el código desarrollado.


El objetivo es desarrollar una utilidad (simplificada) que simule el comportamiento del comando "tree" de Linux. El resultado es mostrar algo así:



```
→ treefiles git:(main) tree
.
├── LICENSE
├── README.md
├── requirements.txt
├── tests
├── treefiles
│   ├── drawer.py
│   ├── tree.py
│   └── utils.py
├── treefiles_run.py
└── treefiles.zip
```

Desarrollar esa utilidad será algo complejo para hacerlo en el tiempo que tenemos por lo que vamos a simplificarlo. En primer lugar vamos a liberarnos de la tarea de consultar el árbol de directorio del sistema operativo, y vamos a simularlo.

Luego vamos a crear la posibilidad de que el usuario pueda elegir como presentar el árbol, si es que lo hacemos al estilo de unix, como el que se mostró en la imagen o lo hacemos en un formato ascii como el de la siguiente imagen:



```
(venv) → treefiles git:(4db0219) python treefiles_run.py /home/eamanu -f ascii
home
+-- eamanu
+-- Documents
|   +-- ejemplo
|   |   +-- ejemplo.py
|   |   +-- ejemplo2.py
|   +-- carpeta_vacia
+-- Videos
+-- Pictures
```

Dentro del comprimido se encuentra un archivo requirements.txt que es el que tiene todas las dependencias necesarias para que el programa funcione. De ahí no deben modificar nada.

Dentro de la carpeta *treefiles* se encuentran todos los archivos que necesitarán para que el programa funcione. Pero OJO hay errores y partes incompletas.

Leean atentamente el docstring del código porque puede dar algunas pistas, incluso hasta la solución.

Se debe hacer que el programa reciba parametros para ello van a utilizar argparse para facilitar esa tarea.

EL programa debe pedir un "path" que es el que se va a listar (PERO RECUERDEN AUNQUE SE PIDA EL PATH ESTÁ SIMULADO, ASÍ QUE NO IMPORTA QUÉ VALOR VA ACA) y otro parámetro que me indique el formato.

Por último hacer al menos 3 test unitarios para las funciones

Entonces las tareas son:

- crear la función `get_simulate_folder()` que me devuelva un formato conocido dependiendo si quiero imprimir en formato unix o ascii, y que le guste a la biblioteca Ver docstring.
- Corregir los errores.
- Implementar lo que haga falta
- Hacer que la utilidad reciba los parametros: folder y format
- Hacer test unitarios.