




# Challenge Python - Terminal TO-DO list

**Nombre del Proyecto:** [usuario]-python-examen

Esta prueba consiste en la creación de una TO-DO lista o **agenda de tareas** que pueda ser utilizada desde la terminal y ofrezca cierto grado de persistencia. Esta **será de temática y estilo libre** conteniendo los requisitos que se explican a continuación.

## Stack obligatorio

- Versión Python  $\geq 3.08$ , ([Python](#))
- Librería Colorama,  
(Tutorial:   Aprende Colorama en Python: Colores y Estilos en tu Terminal )

## Detalle del Challenge

Los requisitos funcionales de la prueba son:

- El programa deberá mostrar un menú por terminal que permita interactuar con las opciones del mismo. El menú deberá ser claro y amigable con el usuario.
- Previo al arranque del programa, la terminal deberá quedar totalmente limpia.
- El programa deberá permitir realizar las operaciones CRUD, con persistencia pero sin base de datos.
- [OPCIONAL]: El menú contará con sub-menús dentro de las opciones principales donde el usuario podrá interactuar fácilmente (volver atrás, adelante...)

Los requisitos técnicos de la prueba son:

- Manejo de archivos para las operaciones de lectura, escritura, actualización y eliminación de contenido. Estos archivos serán en formato JSON o CSV
- Gestión de errores personalizados con el objetivo de mantener siempre el programa en ejecución hasta que el usuario decida salir del mismo.
- [OPCIONAL]: Los datos que se manejen durante la ejecución del programa se guardarán en formato JSON y una vez que el usuario decida terminar la ejecución, se guardarán en formato CSV. Este CSV se volverá a cargar al inicio del programa y pasará a JSON para volver a guardar los datos del usuario durante la ejecución.
- [OPCIONAL]: Incluir un archivo **.gitignore** con el entorno ignorado.



## Ejemplo de posible menú

```
-----Menu-----  
1. Add new note  
2. Display all notes  
3. Search for a note by title  
4. Search for a note by content  
5. Update a note  
6. Delete a note  
0. Exit  
  
[+] Enter your choice:
```

## Requisitos obligatorios

- La aplicación deberá contar con al menos tres entidades o clases (ej. entidades y casos de uso) de libre implementación excluyendo el menú como posible implementación de clase. Así como, con al menos dos carpetas y el habitual punto de entrada, main.py
- Es obligatorio la creación de un environment aislado.
- Uso de, al menos una vez, la librería Colorama, arriba mencionada.
- Se precisa de la creación de un archivo README.md donde se indiquen los pasos a seguir para probar la aplicación, así como, de al menos, un ejemplo orientativo de su uso a modo de guía de usuario.
- Código documentado.
- La terminal debe de estar y mantenerse limpia de errores durante la ejecución del programa.

## Importante

- No está permitido el uso de ninguna base de datos.
- No está permitida la descarga de ninguna dependencia adicional a las indicadas.



## Evaluación

El objetivo final de la prueba es subir el código a un repositorio personal (Gitlab o similar) con la solución desarrollada.

Esta prueba evalúa principalmente cuatro áreas de conocimientos. En primer lugar, y como bloque principal, programación orientada a objetos. En segundo lugar, modularización del código, y correcta implementación de los principios DRY, y SRP. En tercer lugar, el correcto dominio de algunas de las librerías in-built de python más importantes (como son, entre otras, os, sys, pickle, date, time, math, etc...)

Ninguna de las anteriores es obligada ni quizá están todas las necesarias por esta aplicación recogidas en la lista anterior, por lo que te recomendamos planificar bien el proyecto y consultar la documentación oficial de Python en todo momento. Y por último, el dominio de las convenciones, sintaxis y prácticas habituales de Python.

La nota de evaluación se adjudicará atendiendo al siguiente criterio de evaluación:

- **POO: 4 puntos**
  - Definición de clases y objetos
  - Uso de métodos y funciones dentro de las clases
- **Modularización: 2 puntos**
  - Estructura modular del proyecto
  - Uso de principios DRY y SRP
- **Librerías: 1 punto**
- **Python general: 3 puntos**
  - Convenciones y buenas prácticas
  - Manejo de errores y excepciones
- **Puesta en común y defensa del proyecto\***
- **Puntos comodín por [Opcionales]\*\*: 2 puntos**
  - Gestión avanzada de archivos: 1 punto
  - Uso de submenús y navegación mejorada: 0.5 puntos
  - Inclusión de .gitignore: 0.5 puntos

*\* El desconocimiento flagrante durante la exposición de parte del código puede llevar a la consideración de “copia” o “plagio” del mismo, implicando la inmediata anulación de la prueba con resultado de 0 puntos de evaluación.*

*\*\* Los puntos opcionales, considerados puntos comodín, ayudarán al alumno a alcanzar la máxima puntuación de la prueba y podrán ser utilizados en cualesquiera sean los bloques de evaluación a excepción de la puesta en común.*

## Contacto

Si durante la realización de la prueba encuentras alguna duda o necesitas aclaraciones, no dudes en contactar con tu formador asignado.