# Laboratorio
# Práctica Final. Tipos de datos estructurados en Python, módulos y archivos

## 1. Objetivos

- Aprender a hacer programas que utilicen los diferentes objetos de datos estructurados que ofrece Python: cadenas, tuplas, listas y diccionarios.
- Conocer y utilizar los métodos (funciones) asociados a los datos estructurados en Python.
- Consolidar el desarrollo de programas en Python siguiendo las directrices de la programación modular utilizando funciones.
- Practicar la interacción de los programas con archivos de texto para leer y escribir datos.

## 2. Introducción

Esta práctica consiste en el desarrollo de un programa que implemente una red social. El programa debe cargar las bases de datos suministradas y, a partir de estas, implementar la funcionalidad básica de una red social y contar con persistencia de datos.

### 2.1. Requerimientos del programa

#### Requisitos funcionales

El programa por desarrollar debe ofrecer al usuario un menú con las siguientes opciones:

**Registrarse en la red social:** se le solicita al usuario que ingrese su nombre y contraseña. Se debe verificar que el nombre sea solo texto y que no haya sido registrado previamente en la red social. Para definir la contraseña se debe solicitar al usuario que la repita dos veces, en caso de que sea diferente se debe indicar.

**Iniciar sesión:** para iniciar sesión el usuario debe suministrar su nombre y contraseña. Se debe verificar que el nombre y la contraseña correspondan a un usuario registrado; si no corresponden, se le debe indicar al usuario que sus datos están errados, en caso de fallar dos veces el usuario se debe bloquear y sólo se podrá desbloquear pasados 10 minutos siempre y cuando ingrese la contraseña de forma correcta.

Una vez el usuario inicie sesión, se le debe informar sobre la cantidad de amigos, solicitudes de amistad y mensajes que tiene; además, se le debe presentar un menú con las siguientes opciones:

- **Ver usuarios registrados:** el usuario puede ver los nombres de los usuarios registrados en la red social.

- **Enviar una solicitud de amistad:** el usuario puede enviar una solicitud de amistad a alguno de los usuarios registrados. Para preservar la integridad de las bases de datos, se debe tener en cuenta las siguientes restricciones en la funcionalidad:

  **a.** No se puede enviar una solicitud de amistad a usuarios que no están registrados en la red social.
  **b.** No se puede enviar una solicitud de amistad a usuarios que estén en su lista de amigos.
  **c.** No se puede enviar una solicitud de amistad a otros usuarios más de una vez.
  **d.** No se puede enviar una solicitud de amistad a un usuario que haya solicitado ser su amigo. Lo más lógico es aceptar la solicitud que le han hecho.
  **e.** El usuario no se puede enviar una solicitud de amistad a sí mismo.

  **Nota:** Estos eventos se deben gestionar, en caso de ocurrir alguno de ellos se debe mostrar un mensaje de error que indique la restricción.

- **Ver solicitudes de amistad pendientes:** con esta opción se puede ver qué usuarios le han enviado una solicitud de amistad. También se debe ofrecer la funcionalidad de aceptar alguna de las solicitudes disponibles; en este sentido, si se acepta una solicitud, la misma debe ser eliminada del listado de solicitudes pendientes, y los dos usuarios (el que envió la solicitud y el que la aceptó) deben ser catalogados como amigos.

- **Ver mensajes:** esta funcionalidad le permite al usuario ver los mensajes que le han enviado sus amigos ordenados cronológicamente a partir del más reciente.

- **Enviar mensaje a un amigo**: con esta opción el usuario puede escribirle un mensaje a alguno de sus amigos. Se debe tener en cuenta que solo se puede enviar mensajes a usuarios registrados en la red social y que sean amigos de quien pretende enviar el mensaje. El mensaje debe contener fecha y hora (dia/mes/año, hora:minuto:segundo), además del nombre del usuario que envió el mensaje.

  Ayuda: investigar el módulo datetime de Python.

- **Solicitar un listado de usuarios que tengan intereses similares a los suyos** (dos o más). De esta manera, el usuario puede enviarle solicitudes de amistad a usuarios con intereses afines; para ello, se debe tener en cuenta que:
  **1.** Se debe agregar una nueva opción donde el usuario pueda ingresar sus gustos, estos se deben guardar en la base de datos userData.txt respetando el formato establecido.
  **2.** Esta opción solo debe mostrar otros usuarios compatibles que aún no son amigos del usuario.
  **3.** Se debe dejar claro que intereses comunes tienen, por ejemplo: gusto por los animales, deportes, etc.

- **Retirarse de la red social:** En esta opción el usuario se puede retirar de la red social, para hacerlo debe ingresar la contraseña que tenga por defecto. Los datos del usuario retirado se deben guardar (esto exige que se cree un dato en la base de datos que contenga el estado del usuario: activo, bloqueado, retirado…). En caso de ingresar tres veces la contraseña de forma incorrecta el usuario se debe bloquear y sólo se puede desbloquear pasados 10 minutos e ingresando correctamente la contraseña (en caso de bloqueo se debe volver al menú principal).
- **Cerrar sesión:** al seleccionar esta opción, se debe cerrar la sesión iniciada y el programa debe regresar al menú principal.

**Estadísticas de la red social:** al seleccionar esta opción se debe mostrar un menú que permitirá mostrar información de la red social, las opciones son las siguientes:

**a.** Histograma con la evolución de los registros mes a mes: En esta opción se debe graficar un histograma donde se muestre la información de las inscripciones en la red social en el tiempo.
**b.** Grafica de usuarios activos, usuarios retirados y bloqueados contra el tiempo (datos totales). Se debe graficar, en el tiempo, la evolución de usuarios activos y retirados.
**c.** Diagrama de torta con información del porcentaje y cantidad de usuarios por género.
**d.** Histograma con la distribución de los usuarios por edad. Se deben generar 3 histogramas en un subplot (consultar), donde se deje evidencia de la distribución de los usuarios activos, bloqueados y retirados.

**Salir del programa:** al elegir esta opción el usuario termina la ejecución de la aplicación.

Importante: cualquier cambio en la información de los usuarios debe quedar registrada en las bases de datos respetando los formatos establecidos.

### Bases de datos

Se le entregan los siguientes archivos: users.txt, userData.json y una carpeta con los mensajes escritos en la red social por cada usuario. Es necesario familiarizarse con la estructura de los archivos con el fin de que entienda cómo manipularlos.

Nota: Inspeccione los archivos manualmente para familiarizarse con su formato.

Es importante tener en cuenta que el archivo que se envía es el formato de la base de datos, observe que es necesario crear más usuarios en la base de datos para la red social.

### Requisitos no-funcionales

El programa desarrollado debe cumplir con los siguientes requisitos:

- Se sugiere (opcional) diseñar una clase para un usuario, con las propiedades que considere necesarias.

- Utilizar funciones en el desarrollo del programa, a criterio del programador, con sus respectivas especificaciones a manera de docstrings.
- Crear dos módulos (archivos fuente), uno para el programa principal que implementará los menús para la interacción con el usuario, y otro para las funciones que se encargarán de la administración de la información de las bases de datos.
- Utilizar strings, tuplas, listas y diccionarios.

**Importante:** El diseño debe tener una secuencia lógica, se deben mostrar los mensajes de error (por consola) que se requieran para que el flujo del programa sea adecuado. Tenga en cuenta que los mensajes de cada usuario se deben guardar en un archivo .txt con el ID del usuario y las claves se guardan en el archivo users.txt.

### 2.2. Etapas de desarrollo sugeridas

**Etapa 1:**

Realice un diagrama libre que indique la organización que le desea dar a los datos en el programa: tipos de variables, jerarquías, contenidos, etc. Este es el insumo principal para poder iniciar con el desarrollo del código; además diseñe un diagrama de flujo que muestre la funcionalidad general del programa.

**Etapa 2:**

Diseñe las funciones necesarias para cargar en las variables los datos de los archivos y para el proceso inverso, es decir, poder guardar en los archivos todo lo que esté en las variables respetando el formato definido.

**Etapa 3:**

Desarrolle las funcionalidades restantes del programa.

## 3. Evaluación

La evaluación se basará en los códigos enviados y una sustentación oral sobre los temas de la práctica.