Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2233 Programación Avanzada (2021-1)

Tarea 0

Entrega

- **Tarea**
  - **Fecha y hora:** miércoles 31 de marzo de 2021, 20:00
  - **Lugar:** Repositorio personal de GitHub — Carpeta: `Tareas/T0/`
- `README.md`
  - **Fecha y hora:** jueves 1 de abril de 2021, 20:00
  - **Lugar:** Repositorio personal de GitHub — Carpeta: `Tareas/T0/`

Objetivos

- Aplicar competencias asimiladas en *Introducción a la Programación* para el desarrollo de una solución a un problema.
- Familiarizarse con el proceso de entrega de tareas y uso de buenas prácticas de programación.
- Procesar *input* del usuario de forma robusta, manejando potenciales errores de formato.
- Trabajar con archivos de texto para leer, escribir y procesar datos.
- Escribir código utilizando paquetes externos (*i.e.* código no escrito por el estudiante), como por ejemplo, módulos que pertenecen a la librería estándar de Python.

Índice

1. *DCConecta2*

Las recientes caídas de grandes aplicaciones de mensajería, como Whatsapp y Messenger, han provocado saturación y demoras en otras aplicaciones, como Telegram, debido a la repentina alza en la demanda. Tras quedarse sin forma de comunicación entre ayudantes, el DCC ha decidido tomar riendas en el asunto.

Como departamento, han decidido encargarte crear un prototipo para un nuevo sistema de mensajería basado en archivos `.csv` llamado ***DCConecta2***. En este nuevo sistema podrás registrarte, hablar con tus contactos y chatear en grupos, entre otras cosas.

Figura 1: Logo de *DCConecta2*

2. Flujo del programa

DCConecta2 corresponde a un cliente de mensajería instantánea, cuya navegación por los distintos Menús se realizará mediante la interacción de un usuario con la terminal.

Al ejecutar el programa, lo primero que se mostrará es el Menú de Inicio, donde el usuario tendrá la opción de **registrarse** con un nombre de usuario nuevo, **iniciar sesión** con un nombre de usuario ya registrado o simplemente **salir** del programa. Una vez dentro del programa, el usuario podrá acceder al Menú de Chats donde tendrá la opción de revisar sus contactos, grupos o volver al menú de inicio.

Desde el Menú de Contactos, el usuario deberá ser capaz de enviar mensajes a **usuarios que se encuentren entre sus contactos** o **agregar nuevos usuarios a sus contactos** ingresando un nombre de usuario válido. Por otro lado, en el Menú de Grupos el usuario podrá **ver todos sus grupos**, **ver el historial de ese grupo**, **crear grupos** y **abandonar grupos**.

Toda la información sobre usuarios, contactos, grupos y mensajes será recuperada de una base de datos almacenada en archivos `.csv`, donde **cualquier modificación deberá ser registrada para un futuro ingreso al programa**.

3. Tipos de chats

Existirán dos tipos de *chats* en *DCConecta2*, los cuales te permitirán tener una grata experiencia y comunicarte de diferentes formas con tus amigues.

3.1. Chats regulares

En estos Chats podrás conversar directamente con algún usuario que esté previamente en tu lista de contactos. Al seleccionar el nombre de usuario de esta persona en el Menú de Contactos, se desplegará el historial de conversaciones que hayas tenido con ese usuario hasta entonces y podrás continuar la conversación, o enviar un comando que te permitirá volver atrás. Cabe destacar que si envías un mensaje nuevo, este debe ser agregado al historial de la conversación con ese contacto, por lo que deberás modificar el archivo `mensajes.csv` y luego deberás volver a mostrar el historial de conversación.

La forma en la que muestres el historial de mensajes queda a tu criterio, pero como mínimo deberás incluir la **fecha y hora de envío**, **emisor** y el **contenido de cada mensaje**. Además de esto, los mensajes se deberán desplegar en orden ascendente, mostrando los más antiguos arriba, y los más nuevos abajo. A continuación se presenta un ejemplo de *chat* regular:

```
****** Tu historial con Juampisaez ******

2015/07/06 11:57:35, Juampisaez: Hola, como estas?
2015/07/06 12:02:01, MatiasDuhalde: Bien! Y tu?
2015/07/06 12:08:10, Juampisaez: Biennn
2015/07/06 12:08:12, Juampisaez: Tengo que hacer la tarea 0
2015/07/06 12:12:22, MatiasDuhalde: Ohh
2015/07/06 12:12:27, MatiasDuhalde: Me encanta avanzada

****** Fin del historial con Juampisaez ******
```

Figura 2: Ejemplo de historial de mensajes de Chat Regular

3.2. Grupos

En las conversaciones de grupo podrás enviar mensajes a múltiples usuarios que estén en el mismo grupo que tú. El grupo tendrá un **nombre único**, un mínimo de **dos usuarios** y no tendrá máximo de participantes.

En el Menú de Grupos se desplegarán todos los grupos a los que pertenezcas y podrás seleccionar alguno escribiendo el nombre del grupo en la terminal. Al igual que en los Chats regulares, al seleccionar el grupo se mostrará todo el historial de conversación correspondiente al grupo, y podrás enviar nuevos mensajes o ciertos comandos para volver atrás o abandonar el grupo. Además, deberás tener la opción de abandonar un grupo en cualquier momento mediante un comando especial. Los mensajes que envíes deberán agregarse al historial de la conversación del grupo, modificando el archivo **mensajes.csv** y volviendo a mostrar el historial de conversación del grupo cada vez que se envíe un mensaje.

La forma en la que muestres el historial de mensajes queda a tu criterio, pero como mínimo deberás incluir el **nombre del grupo**, la **fecha y hora de envío**, **emisor** y el **contenido de cada mensaje**. Además de esto, los mensajes se deberán desplegar en orden ascendente, mostrando los más antiguos arriba, y los más nuevos abajo. A continuación se presenta un ejemplo de *chat* de grupo:

```
****** Historial de Los Teletubbies ******

2020/04/03 10:42:12, lily416: Hola a todos!
2020/04/03 10:43:23, csantiagopaz: Hola!!
2020/04/03 10:43:56, Pablok98: Helloo
2020/04/03 10:44:18, Tsbalart: Holaaaa
2020/04/03 10:46:16, baoss: Como estan??
2020/04/03 10:42:12, MatiasDuhalde: Todo bien!
2020/04/03 10:42:14, MatiasDuhalde: Tanto tiempo!

****** Fin del historial de Los Teletubbies ******
```

(a) Ejemplo de historial de mensajes de Grupo

4. Menús

Para permitir la correcta navegación del usuario dentro del *DCConecta2*, deberás implementar un sistema de **menús** que facilite la interacción por consola. Cada menú muestra opciones disponibles al usuario, cuya decisión debe ser **recibida** y **procesada** de forma correcta.

Todos los menús deberán ser **a prueba de errores**, es decir, tu programa no debe caerse al ingresar una opción inválida y debe responder de forma acorde. El formato en que decidas mostrarlos quedará a tu criterio, pero debes incluir las opciones de **volver al menú anterior** y **salir del programa** cuando corresponda.

También puedes crear distintos submenús si lo consideras necesario, pero como mínimo tu programa deberá implementar los siguientes:

4.1. Menú de Inicio

Este será el primer menú que se mostrará al iniciar el programa. Se deben mostrar las opciones de **Iniciar Sesión** con un usuario ya registrado, **Registrar** a un nuevo usuario o **Salir del programa**.

Al momento de **Iniciar Sesión**, se deberá solicitar el **nombre de usuario** y verificar que el nombre exista dentro de los usuarios registrados en el archivo `usuarios.csv` (más información en Archivos). Si el usuario ingresado es inexistente, se deberá notificar al usuario y volver al Menú de Inicio.

Si se elige la opción de **registrar** un usuario, se pedirá el **nombre de usuario**, verificando que el nombre **no** esté actualmente siendo utilizado por otro usuario, que tenga un largo **entre 3 y 15** caracteres (ambos extremos incluidos y no incluya comas (`","`). En caso de que el nombre ingresado no cumpla con estas condiciones, se deberá notificar al usuario y volver al Menú de Inicio. Por otro lado, si el nombre inscrito es válido, deberás añadirlo dentro del archivo `usuarios.csv`

Una vez que hayas logrado iniciar sesión o haber registrado un nuevo usuario, deberás dirigirte al Menú de Chats con la misma cuenta.

4.2. Menú de Chats

En este menú, el usuario podrá seleccionar los distintos Tipos de chats que desea revisar. En particular, deberás mostrar las opciones de **ver contactos**, **ver grupos** o **volver al menú de inicio**.

A continuación se muestran ejemplos de **Menú de Inicio** y **Menú de Chats**:

```
***** Menu de Inicio *****

Selecciona una opción:

[1] Crear usuario
[2] Iniciar sesión
[0] Salir

Indique su opción (0, 1 o 2): (input del usuario)
```

(a) Ejemplo de Menú de inicio

```
***** Menu de Chats *****

Selecciona una opción:

[1] Ver contactos
[2] Ver grupos
[0] Volver

Indique su opción (0, 1 o 2): (input del usuario)
```

(b) Ejemplo de Menú de Chats

4.2.1. Menú de Contactos

Al seleccionar la opción de **Ver contactos** desde el Menú de Chats, ingresarás al Menú de Contactos. Este deberá permitir las siguientes operaciones:

1. Mostrar todos los contactos del usuario: En caso de seleccionar esta opción, se mostrará un listado con todos los contactos del usuario, junto a la opción de seleccionar uno. Al seleccionar un contacto, se desplegará el historial de los mensajes enviados entre ambos usuarios ordenados de forma ascendente (es decir, mientras más nuevo sea el mensaje, más abajo se ubicará en consola).

   Luego de haber mostrado todo el historial de conversaciones, cualquier *input* que se ingrese se enviará como un nuevo mensaje de parte del usuario. Si se ingresa el *input* `VOLVER_FRASE` [1], deberás volver al menú anterior con el listado de contactos.

2. Añadir contacto: Al seleccionar esta opción, deberás ingresar el **nombre de usuario** del contacto al cual deseas agregar. Si la opción es válida y el usuario existe, deberás añadir ese nombre dentro de los contactos del usuario y viceversa. En caso de que la opción ingresada sea errónea y el nombre no exista, se deberá notificar al usuario y volver a desplegar el menú actual.

3. Volver al **Menú de Chats**.

A continuación se presenta un ejemplo del Menú de Contactos:

```
***** Menu de Contactos *****

Selecciona una opción:

[1] Ver contactos
[2] Añadir contacto
[0] Volver

Indique su opción: (input del usuario)
```

(a) Ejemplo de Menú de contactos

```
***** Ver Contactos *****

Selecciona un usuario para
ver tus conversaciones,
o 0 para volver atrás:

[1] matiasmasjuan
[2] igbasly
[3] DCCollao
[0] Volver

Indique su opción: (input del usuario)
```

(b) Ejemplo de listado de contactos.

4.2.2. Menú de Grupos

Al seleccionar la opción de **Ver grupos** desde el Menú de Chats, ingresarás al Menú de Grupos. Este deberá permitir las siguientes operaciones:

[1] Las palabras escritas en `ESTE_FORMATO` son parámetros que tendrás que escribir e importar desde el archivo `parametros.py`

1. Mostrar todos los grupos del usuario: En caso de seleccionar uno, se mostrará un listado de todos los grupos a los que perteneza el usuario, junto con la opción de seleccionar uno.

   Al seleccionar un grupo, se desplegará el historial de todos los mensajes enviados al grupo ordenados de forma ascendente. A continuación, cualquier *input* que sea ingresado se enviará como mensaje y deberá almacenarse en el archivo `mensajes.csv`.

   Si se ingresa como mensaje el *input* `VOLVER_FRASE`, deberás dirigirte nuevamente al menú anterior con el listado de grupos. En caso de ingresar el *input* `ABANDONAR_FRASE`, el usuario abandonará el grupo, enviando al grupo un último mensaje de forma **automática** que indique: *El usuario <Usuario> ha abandonado el grupo.* Una vez que hayas abandonado el grupo, deberás dirigirte nuevamente al menú anterior.

2. Crear Grupo: Al seleccionar esta opción, deberás ingresar el nombre del grupo, verificando que no exista un grupo con ese mismo nombre, tenga como mínimo **1 carácter** y no contenga comas (`","`). Posteriormente, deberás ingresar todos los usuarios que deseas ingresar al grupo separados por `";"` según el siguiente formato: `usuario1;usuario2;...`. Además, deberás verificar que el nuevo grupo cuente con un **mínimo** de **2** usuarios y que todos los usuarios a añadir existan en el programa. En caso de no cumplirse alguna de estas condiciones, deberás notificar al usuario y volver al menú actual.

3. Volver al **Menú de Chats**.

A continuación se presenta un ejemplo del Menú de Grupos:

```
***** Menu de Grupos *****

Selecciona una opción:

[1] Ver grupos
[2] Crear grupo
[0] Volver

Indique su opción: (input del usuario)
```

(a) Ejemplo de Menú de grupos

```
***** Ver Grupos *****

Selecciona un grupo para
ver tus conversaciones,
o 0 para volver atrás:

[1] Los Teletubbies
[0] Volver

Indique su opción: (input del usuario)
```

(b) Ejemplo de listado de grupos.

5. Archivos

Los siguientes archivos contienen la base de datos de usuarios, contactos, mensajes y grupos que usarás en tu programa. Cada vez que edites uno de estos archivos, deberás mantener el formato especificado para dicho archivo. Debes asegurarte de que los archivos se actualicen constantemente debido a cualquier interacción con el programa.

5.1. `usuarios.csv`

Este archivo contiene una lista de todos los usuarios registrados en **DCConecta2**. Al iniciar sesión debes verificar que el nombre ingresado se encuentre en este archivo. Cuando un usuario es registrado correctamente, debe quedar guardado en este archivo.

Un ejemplo del archivo `usuarios.csv` es el siguiente:

```
usuario
lily416
```

```
Dnpoblete
DCCollao
Gatochico
igbasly
```

5.2. `contactos.csv`

Este archivo contiene la información de los contactos de todos los usuarios del programa. Cada fila contiene el nombre de un usuario y el de un contacto suyo, separados por una coma (`","`). Cabe mencionar que las relaciones de contactos son simétricas, por lo que si un usuario_1 posee como contacto al usuario_2, el usuario_2 también tendrá como contacto al usuario_1.

Un ejemplo del archivo `contactos.csv` es el siguiente:

```
usuario,contacto
lily416,matiasmasjuan
lily416,igbasly
matiasmasjuan,lily416
igbasly,lily416
```

5.3. `grupos.csv`

Este archivo contiene la información de los integrantes de cada grupo existente.Cada fila contiene el nombre de un grupo y el de uno de sus integrantes, separados por una coma (`","`). Por lo tanto, cada grupo deberá poseer tantas filas como integrantes hayan en él. Cabe mencionar que cada grupo deberá respetar la cantidad de integrantes mínima.

Un ejemplo del archivo `grupos.csv` es el siguiente:

```
grupo,integrante
El Padrino,lily416
Hagamos como que socializamos,csantiagopaz
Hagamos como que socializamos,Dnpoblete
El Padrino,Juampisaez
```

5.4. `mensajes.csv`

Este archivo contiene la información de todos los mensajes del sistema. Cada línea posee cinco datos relevantes para un mensaje, separado por comas (`","`): El **tipo de chat** que puede ser `regular` o `grupo`, el **emisor**, el **receptor**, **fecha de emisión** y el **contenido**. Cabe notar que el cuerpo de un mensaje puede poseer comas, por lo que cualquier coma encontrada después de la que se utiliza para separar la fecha de emisión del cuerpo es parte de este último [2].

Un ejemplo del archivo `mensajes.csv` es el siguiente:

```
tipo_chat,emisor,receptor,fecha_emision,contenido
regular,MatiasDuhalde,Juampisaez,2015/07/06 11:57:35,Hola, como estas?
regular,csantiagopaz,MatiasDuhalde,2015/08/18 08:33:35,Tengo que hacer la tarea 0
grupo,baosse,Los otros,2015/10/17 22:16:06,Me encanta avanzada
```

[2] Podría ser útil el parámetro **maxsplit** del método split de strings.

```
regular,Dnpoblete,manearaya,2015/10/26 02:41:11,Tanto tiempo!
grupo,Dnpoblete,Paella familiar,2015/11/10 03:15:51,No quiero cuarentena
```

5.5. parametros.py

Al momento de escribir programas de mayor complejidad, como lo son las tareas del curso, una buena práctica es **parametrizar** ciertos datos o variables que pueden variar entre ejecuciones de este, o que dependen de la estructura de archivos del computador del usuario. El tener todos los parámetros almacenados en un único archivo permite identificarlos y modificarlos de forma simple y rápida.

Para esta tarea te entregamos un archivo `parametros.py` ya rellenado. En este archivo se encontrarán los parámetros mencionados anteriormente en el enunciado (en `ESTE_FORMATO`), en donde cada línea almacena una constante con su respectivo valor.

Particularmente, este archivo contiene las frases claves que deberá detectar tu programa para **volver atrás** estando dentro de una conversación y **sacar** a un usuario de un grupo. Por lo tanto, deberás usar las variables de este archivo para verificar si efectuar estas acciones o no. El contenido del archivo `parametros.py` es el siguiente:

```python
# Si el programa detecta que se ingresó este input en una conversación de grupo,
# deberá sacar al usuario del grupo.
ABANDONAR_FRASE = "\\salir"
# Si el programa detecta que se ingresó este input en cualquier conversación,
# Debe volver al menú anterior
VOLVER_FRASE = "\\volver"
```

6. Buenas Prácticas

Se espera que durante todas las tareas del curso, se empleen buenas prácticas de programación. Esta sección detalla dos aspectos que deben considerar a la hora de escribir sus programas, que buscan mejorar la forma en que lo hacen.

- **PEP8**:

  PEP8 es una guía de estilo que se utiliza para programar en Python. Es una serie de reglas de redacción al momento de escribir código en el lenguaje y su utilidad es que permite estandarizar la forma en que se escribe el programa para sea más legible[3]. En este curso te pediremos seguir un pequeño apartado de estas reglas, el cual puede ser encontrado en la **guía de estilos**.

- **Modularización**:

  Al escribir un programa complejo y largo, se recomienda organizar en múltiples módulos de poca extensión. Se obtendrá puntaje si ningún archivo de tu proyecto contiene más de 400 líneas de código.

Normalmente, estos dos aspectos son considerados como descuentos. A manera de excepción, para esta tarea serán parte del puntaje de la tarea, buscando que los apliques y premiando su correcto uso. Ten en cuenta que en las siguientes tareas, funcionarán como cualquier otro descuento de la sección Descuentos.

[3] Símil a como la ortografía nos ayuda a estandarizar la forma es que las palabras se escriben.

7. README

Para todas las tareas de este semestre deberás redactar un archivo `README.md`, un archivo de texto escrito en Markdown, que tiene por objetivo explicar su tarea y facilitar su corrección para el ayudante. Markdown es un lenguaje de marcado (como LaTeX o HTML) que permite crear un texto organizado y simple de leer. Pueden encontrar un pequeño tutorial del lenguaje en este ***link***.

Un buen `README.md` debe **facilitar la corrección de la tarea.** Una forma de lograr esto es explicar de forma breve y directa el **idioma** en qué programaste (puedes usar inglés o español) y **qué cosas fueron implementadas, y qué cosas no**, usualmente **siguiendo la pauta de evaluación**. Esto permite que el ayudante dedique más tiempo a revisar las partes de tu tarea que efectivamente lograste implementar, lo cual permite entregar un *feedback* más certero. Para facilitar la escritura del `README`, se te entregará una **plantilla** (*template*) a rellenar con la información adecuada.

Finalmente, como forma de motivarte a redactar buenos `README`s, todas las tareas tendrán **décimas de des-descuento** si el ayudante considera que tu `README` fue especialmente útil para la corrección. Estás décimas anulan décimas de descuento que les hayan sido asignadas hasta un máximo de cinco.

8. Descuentos

En todas las tareas de este ramo habrá una serie de descuentos que se realizarán para tareas que no cumplan ciertas restricciones. Estas restricciones están relacionadas con malas prácticas en programación, es decir, formas de programar que hacen que tu código sea poco legible y poco escalable (difícil de extender con más funcionalidades). Los descuentos tienen por objetivo que te vuelvas consciente de estas prácticas e intentes activamente evitarlas, lo cual a la larga te facilitará la realización de las tareas en éste y próximos ramos donde tengas que programar. Los descuentos que se aplicarán en esta tarea serán los siguientes:

- `README`: (1 décima)

  Se descontará una décima si no se indica(n) los archivos principales que son necesarios para ejecutar la tarea o su ubicación dentro de su carpeta. También se descontará una décima si es que no se hace entrega de un `README`. Esto se debe a que este archivo facilita considerablemente la recorrección de las tareas.

- **Formato de entrega**: (hasta 5 décimas)

  Se descontarán hasta cinco décimas si es que no se siguen reglas básicas de la entrega de una tarea, como son el uso de groserías en su redacción, archivos sin nombres aclarativos, no seguir restricciones del enunciado, entre otros[4]. Esto se debe a que en próximos ramos (o en un futuro trabajo) no se tiene tolerancia respecto a este tipo de errores.

- **Cambio de líneas**: (hasta 5 décimas)

  Se permite cambiar **hasta 20 líneas de código** por tarea, ya sea para corregir un error o mejorar una funcionalidad. Este descuento puede aplicarse si se requieren cambios en el código después de la entrega (incluyendo las entregas atrasadas). Dependiendo de la cantidad de líneas cambiadas se descontará entre una y cinco décimas.

- **Adicionales**: (hasta 5 décimas)

  Se descontarán hasta cinco décimas a criterio del ayudante corrector en caso de que la tarea resulte especialmente difícil de corregir, ya sea por una multitud de errores o porque el programa sea especialmente ilegible. Este descuento estará correctamente justificado.

[4] Uno de los puntos a revisar es el uso de *paths* relativos, para más información revisar el siguiente **material**.

- ***Built-in* prohibidos**: (entre 1 a 5 décimas)

  En cada tarea se prohibirán algunas funcionalidades que Python ofrece y se descontarán entre una y cinco décimas si se utilizan, dependiendo del caso. Para cada tarea se creará una *issue* donde se especificará qué funcionalidades estarán prohibidas. Es tu responsabilidad leerla.

- **Malas prácticas**: (hasta 5 décimas)

  Al igual que los *built-ins* prohibidos, también se prohibirán ciertas malas practicas y se descontarán entre una y cinco décimas si se realizan. Para cada tarea se creará una *issue* donde se especificará qué malas prácticas estarán prohibidas. Es tu responsabilidad leerla y preguntar en caso de tener dudas sobre las malas prácticas establecidas.

- **Entrega atrasada**: (entre 5 a 20 décimas)

  Las tareas serán recolectadas automáticamente y no se considerará ningún avance realizado después de la hora de entrega. Sin embargo, se puede optar por entregar la tarea de forma atrasada y se descontarán 5 a 20 décimas dependiendo de cuánto tiempo de diferencia haya entre la hora de entrega y la entrega atrasada.

- **Des-descuento**: (entre 1 a 5 décimas)

  Finalmente, se des-descontarán hasta 5 décimas por un `README` especialmente útil para la corrección de la tarea.

En la **guía de descuentos** se puede encontrar un desglose más específico y detallado de los descuentos.

9. `.gitignore`

Cuando estés trabajando con repositorios, muchas veces habrán archivos y/o carpetas que no querrás subir a la nube. Por ejemplo, puedes estar trabajando con planillas de Excel muy pesadas, o tal vez estás utilizando un Mac y no quieres subir la carpeta `__MACOSX`, o el archivo `.DS_Store`, entre otros.

Una posible solución es simplemente tener cuidado con lo que subes a tu repositorio. Sin embargo esta "solución" es extremadamente vulnerable al error humano y podría terminar causando que subas muchos *gigabytes* de archivos y carpetas no deseados a tu repositorio.[5]

Para solucionar esto, `git` nos da la opción de crear un archivo `.gitignore`. Éste es un archivo **sin nombre, y con extensión `.gitignore`**, en el cual puedes detallar **archivos y carpetas a ser ignoradas por `git`**. Esto quiere decir que todo lo especificado en este archivo **no será subido a tu repositorio accidentalmente**, evitando los problemas anteriores.

En esta ocasión el uso de este archivo **no será evaluado**, pero se recomienda su realización para que aprendan a crearlo y utilizarlo ya que será evaluado en las siguientes tareas.

Se recomienda utilizar el archivo `.gitignore` para ignorar archivos en tu entrega, específicamente el enunciado, los archivos indicados en Archivos y todos los que no sean pertinentes para el funcionamiento de tu tarea. El archivo `.gitignore` debe encontrarse dentro de tu carpeta `T00`. Puedes encontrar un ejemplo de `.gitignore` en el siguiente ***link***.

10. Importante: Corrección de la tarea

Acá se deben indicar los aspectos importantes de la corrección, se debe actualizar según la pauta de cada tarea.

[5] Lamentablemente basado en una historia real.

Para esta tarea, el carácter funcional del programa será el pilar de la corrección, es decir, **sólo se corrigen tareas que se puedan ejecutar**. Por lo tanto, se recomienda hacer periódicamente pruebas de ejecución de su tarea y *push* en sus repositorios.

Cuando se publique la distribución de puntajes, se señalará con color **amarillo** cada ítem que será evaluado a nivel de código, todo aquel que no esté pintado de amarillo significa que será evaluado si y sólo si se puede probar con la ejecución de su tarea.

En tu archivo `README.md` deberás señalar el archivo y la línea donde se encuentran definidas las funciones o clases relacionados a esos ítems.

Finalmente, si durante la realización de tu tarea se te presenta algún problema o situación que pueda afectar tu rendimiento, no dudes en contactar al ayudante jefe de Bienestar a su **correo**.

11. Restricciones y alcances

- Esta tarea es **estrictamente individual**, y está regida por el **Código de honor de Ingeniería**.
- Tu programa debe ser desarrollado en Python 3.7.
- Tu programa debe estar compuesto por uno o más archivos de extensión `.py`.
- Si no se encuentra especificado en el enunciado, supón que el uso de cualquier librería Python está prohibido. Pregunta en la *issue* especial del **foro** si es que es posible utilizar alguna librería en particular.
- Debes adjuntar un archivo `README.md` **conciso y claro**, donde describas los alcances de tu programa, cómo correrlo, las librerías usadas, los supuestos hechos, y las referencias a código externo. **Tendrás hasta 48 horas después del plazo de entrega** de la tarea para subir el `README` a tu repositorio.
- Tu tarea podría sufrir los descuentos descritos en la **guía de descuentos**.
- Entregas con atraso de más de 24 horas tendrán calificación mínima (1,0).
- Cualquier aspecto no especificado queda a tu criterio, siempre que no pase por sobre otro.

Las tareas que no cumplan con las restricciones del enunciado obtendrán la calificación mínima (1,0).