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

IIC2233 Programación Avanzada (2021-2)

Tarea 0

Entrega

- **Tarea**
  - **Fecha y hora:** viernes 3 de septiembre de 2021, 20:00
  - **Lugar:** Repositorio personal de GitHub — Carpeta: `Tareas/T0/`
- `README.md`
  - **Fecha y hora:** viernes 3 de septiembre de 2021, 22:00
  - **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. DCCommerce

Luego de los largos meses en pandemia, las plataformas de ventas *online* han presentado un colapso en el registro de nuevos usuarios y en el sistema de publicaciones, impidiendo que gran parte de las personas puedan vender sus artículos. Además, informantes claves del Departamento de Ciencias y Computación (DCC) han concluido que este problema no será solucionado a la brevedad.

Por esto, el departamento te ha elegido a ti para que crees la plataforma *DCCommerce*, la cual buscará fomentar el comercio en la comunidad DCC y permitir el registro de usuarios sin miedo a que el programa colapse.

Figura 1: Logo de *DCCommerce*

2. Flujo del programa

DCCommerce es una plataforma de ventas para los estudiantes del DCC, que permite registrarse como usuario nuevo, crear publicaciones y comentarios, además de poder ver todas las publicaciones, entre otras cosas. La ejecución de este programa debe ser por consola, por lo que debes cuidar que todas las instrucciones y mensajes se visualicen de manera correcta.

Al iniciar el programa, se debe mostrar el Menú de inicio en la consola, donde podrás **Ingresar** con tu usuario registrado, **Ingresar como usuario anónimo** o **Registrar** tu nuevo usuario. En caso de seleccionar la primera opción, se debe ingresar el usuario y verificar su existencia en `usuarios.csv`, para luego dar paso al Menú Principal.

En el caso de que se escoja la opción de registrarse como un nuevo usuario, te pedirá el nombre y debes asegurarte que cumpla con las condiciones detalladas en Menú de inicio. Luego de eso, se procede a visualizar el Menú Principal.

Tanto los usuarios ingresados como los recién registrados podrán observar 2 opciones en el Menú Principal: ir al Menú de Publicaciones o ir al Menú de Publicaciones Realizadas. Si se ingresa como usuario anónimo, el programa debe redirigir directamente al Menú de Publicaciones y solo podrán acceder a este menú. En este menú, el usuario podrá visualizar el nombre de las publicaciones realizadas por otros usuarios, donde se puede seleccionar una publicación de la lista y poder ver el detalle de su fecha de creación, vendedor, precio, descripción y los comentarios. Si eres usuario anónimo, no podrás interactuar con las publicaciones, pero si estás registrado, tendrás la posibilidad de **agregar comentario**.

Por otro lado, si se decide ir al Menú de Publicaciones Realizadas, disponible sólo para usuarios registrados, se podrá observar todas las publicaciones personales realizadas hasta el momento, además de mostrar las siguientes opciones: **Crear nueva publicación** y **Eliminar publicación**. Para crear publicaciones, se debe ingresar los atributos correspondientes observados en el archivo `publicaciones.csv`.

La plataforma debe funcionar de tal manera que siempre se pueda volver al menú anterior.

3. Menús

La ejecución de *DCCommerce* se realizará mediante la interacción en consola a través de diferentes menús, los cuales se mostrarán dependiendo de las opciones que se elijan. 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 el que decidas mostrarlos queda a tu criterio, siempre y cuando se mantengan las opciones mínimas requeridas que se mostrarán a continuación.

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

3.1. Menú de inicio

Este será el primer menú que se mostrará al momento de ejecutar el programa. Deberá contar con las opciones de **Ingresar** como usuario registrado, **Registrar** nuevo usuario, **Ingresar como usuario anónimo** y **Salir** del programa.

Al elegir la opción de **Ingresar** como usuario registrado, se deberá solicitar el nombre de usuario y verificar que el nombre exista dentro de los usuarios registrados en el archivo `usuarios.csv`. Si el nombre de usuario ingresado no existe, se deberá notificar en consola y volver al Menú de inicio.

Si se elige la opción de **Registrar un usuario**, se pedirá el nuevo nombre de usuario verificando que este no se encuentre utilizado por otro usuario, que no incluya comas (`","`) y que tenga un mínimo de `MIN_CARACTERES` y máximo de `MAX_CARACTERES`[1], incluyendo ambos extremos en el intervalo. En caso de que el nombre ingresado no cumpla las condiciones, el programa deberá notificarlo y volver al Menú de inicio. Por otro lado, si el nombre es válido, este deberá ser añadido al archivo `usuarios.csv` y se deberá dirigir al Menú Principal.

Por último, si se desea **Ingresar como usuario anónimo**, no se pedirá nombre de usuario y se pasará directamente al Menú de Publicaciones. Dado esto, podrás notar que al ingresar como usuario anónimo, se tienen menos funcionalidades que al ingresar como un usuario registrado.

3.2. Menú Principal

Este menú se muestra una vez que se ingresa con un nombre de usuario válido y contendrá las opciones para ver el Menú de Publicaciones, Menú de Publicaciones Realizadas por el usuario activo y de volver al Menú de inicio. Todos los menús deben contener la opción de volver al menú anteriormente visitado, con el fin de que sea más sencillo navegar a través de *DCCommerce*.

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

```
--- ¡Bienvenid@s a DCCommerce! ---

*** Menu Inicio ***

Selecciona una opción:

[1] Ingresar sesión
[2] Registrar usuario
[3] Ingresar como usuario anónimo
[4] Salir

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

(a) Ejemplo de Menú de inicio

```
*** Menú Principal ***

[1] Menú de Publicaciones
[2] Menú de Publicaciones Realizadas
[3] Volver

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

(b) Ejemplo de Menú Principal

3.3. Menú de Publicaciones

Ya sea un usuario anónimo o un usuario registrado, en este menú se podrán ver todas las publicaciones existentes en *DCCommerce*, contenidas en el archivo `publicaciones.csv`. Se deberá mostrar sólo su nombre y estar ordenadas de forma descendente, es decir, las más nuevas primero y luego las más antiguas según su fecha de creación. Al seleccionar una de las publicaciones, se deberá desplegar su nombre, su fecha de creación, el nombre del vendedor que la publicó, su precio y su descripción, junto con los comentarios que han hecho los usuarios sobre esta (contenidos en `comentarios.csv`). Estos deberán mostrar (además del comentario), el nombre del vendedor que lo realizó y su fecha de creación. Los comentarios deberán ser mostrados en forma ascendente, es decir, que se muestren primero los comentarios más antiguos y luego los más nuevos.

Además, se deben agregar las opciones de **Agregar comentario** (sólo si el usuario está registrado) y de **Volver** al Menú de Publicaciones. Si un usuario registrado desea añadir un comentario, este se deberá guardar en `comentarios.csv` siguiendo correctamente el formato del archivo.

```
*** Menú de Publicaciones ***

[1] Pato de goma color amarillo
[2] Silla playera Lounge
[3] Cama dos plazas Cannon
[4] Juego Twister Nuevo
[5] Volver

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

Figura 3: Ejemplo de Menú de Publicaciones.

```
*** Pato de goma color amarillo ***

Creado: 2020/12/31 23:59:59
Vendedor: mmunoz12
Precio: $1100
Descripción: Pato de goma color amarillo especial para baños de burbujas.

Comentarios de la publicación:
2020/08/10 14:23:50 -- matiasmasjuan: Muy buena publicación!!!
2021/08/12 20:30:00 -- DCCollao: Lo recomiendo! buen producto.
2021/08/23 12:00:03 -- Emiliax16: Muy bueno!!

[1] Agregar comentario
[2] Volver

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

Figura 4: Ejemplo de publicación seleccionada por un usuario registrado.

3.4. Menú de Publicaciones Realizadas

En este menú se deberán desplegar todos los nombres de las publicaciones realizadas por el usuario hasta el momento. Además, se deberá dar la opción de **Crear** una nueva publicación y de **Eliminar** una publicación existente. Al crear una nueva publicación, se deberá pedir el nombre, el precio y su descripción, guardándolo correctamente en el archivo `publicaciones.csv`. Por otro lado, si se desea eliminar una publicación, se deberán mostrar todos los nombres de las publicaciones realizadas por el usuario junto con su fecha de creación. Finalmente, la publicación escogida debe ser borrada del archivo `publicaciones.csv`, junto con los comentarios asociados a esta publicación en `comentarios.csv`.

```
*** Menú de Publicaciones Realizadas ***

Mis publicaciones:
- Pato de goma color amarillo
- Juego Twister Nuevo

[1] Crear nueva publicación
[2] Eliminar publicación
[3] Volver

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

(a) Ejemplo de Menú de Publicaciones Realizadas.

```
¿Cuál publicación deseas eliminar?:

[1] Pato de goma color amarillo -- Creado el 2020/12/31 23:59:59
[2] Juego Twister Nuevo -- Creado el 2018/05/14 20:00:00
[3] Volver

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

(b) Ejemplo de opción Eliminar publicación.

4. Entidades

En esta sección se detallan las entidades que necesitarás para simular la plataforma DCCommerce. Para modelarlas correctamente, deberás utilizar competencias aprendidas en Introducción a la Programación. Las entidades principales son: Usuarios (de diversos tipos), Publicaciones y Comentarios.

4.1. Usuarios

Existen dos tipos de usuarios: usuarios **anónimos** y usuarios **registrados**. Los usuarios anónimos solo pueden entrar al programa y revisar las publicaciones. Al momento de ingresar, deberán dirigirse únicamente al **Menú de publicaciones**. Por otro lado, los usuarios registrados tienen la posibilidad de ahondar por todo el flujo del programa. Este usuario puede ver las publicaciones de otros usuarios, teniendo la opción de añadir nuevos comentarios. Además, tienen la posibilidad de ingresar a la opción **Ver publicaciones realizadas** en la cual podrán revisar sus publicaciones realizadas, crear nuevas publicaciones, o eliminar las existentes.

4.2. Publicaciones

Todos los usuarios pueden ver las publicaciones. Las publicaciones deben contener: **el nombre de la publicación, nombre de usuario del vendedor, precio, hora de creación y una breve descripción**. Los usuarios registrados tienen las opciones de crear y eliminar publicaciones. Las nuevas publicaciones deben ser agregadas en el archivo `publicaciones.csv`.

```
***Jordan 1 Shadow 2018 High Top Quality***

Creado: 2021/07/13 23:00:30
Vendedorx: catalina-arcila
Precio: $  75000
Descripción: Jordan 1 Shadow 2018 Talla 40 Mas Cordones Grises, están como nuevas

Comentarios de la publicación:
```

Figura 6: Ejemplo formato de publicaciones.

4.3. Comentarios

Los comentarios son las opiniones de publicaciones hechas por otros usuarios. Los comentarios pueden ser vistos por cualquier tipo de usuario, sin embargo, solo los usuarios registrados pueden agregar comentarios a las publicaciones. Cabe destacar que si creas un comentario nuevo, este debe ser agregado al historial de los comentarios de dicha publicación, por lo que deberás modificar el archivo `comentarios.csv`. La forma en la que muestres los comentarios queda a tu criterio, pero como mínimo **deberás incluir la fecha y hora de creación, nombre de usuario del que escribe el comentario y el contenido de cada comentario**. El formato de los comentarios debe ser parecida al siguiente:

```
Comentarios de la publicación:

2020/12/31 23:59:59, matiasmasjuan: Muy buena publicación!!!
2021/08/12 20:30:00, DCCollao: Lo recomiendo. Buen producto.
2021/08/23 12:00:03, Emiliax16: Muy bueno!!
```

Figura 7: Ejemplo de historial de comentarios en las publicaciones.

5. Archivos

Los siguientes archivos contienen la base de datos de usuarios, publicaciones y comentarios que usarás en tu programa. Cada vez que edites uno de estos archivos, deberás mantener el formato especificado para dicho archivo. Además, podrás notar que cada uno de los siguientes archivos viene con un **encabezado** o *header* en la primera linea. Esto indica a cual columna corresponde cada uno de los elementos de las siguientes filas separadas por comas (`","`). Por último, debes asegurarte de que los archivos se actualicen constantemente debido a cualquier interacción con el programa.

Importante: Al momento de leer y escribir en los archivos existe la posibilidad de que ciertos caracteres especiales, como la `ñ` o las tildes no se escriban de forma correcta. Para solucionar esto, existe un argumento en la función `open()` llamado *encoding* donde pueden especificar el formato de la lectura del caracter. Para esta tarea, será **obligatorio** el uso de `encoding="utf-8"`[2] ya que permite la correcta interpretación de estos caracteres.

5.1. `usuarios.csv`

Este archivo contiene todos los usuarios registrados en ***DCCommerce***. Cada fila contiene el nombre de usuario. 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
Gatochico
matiasmasjuan
igbasly
Christian-Klempau
DCCollao
```

[2]Para ver ejemplos de uso, pueden visitar la documentación.

5.2. `publicaciones.csv`

Este archivo contiene la información de cada publicación realizada en el programa. Cada fila contiene cinco datos importantes, un ***id*** diferente para cada publicación, **nombre de la publicación**, **nombre del usuario** que creó la publicación, la **fecha de creación** (en formato `yyyy/mm/dd hh:mm:ss`), el **precio** y una **descripción**, separados por una coma (`","`). Es necesario destacar que para obtener acceso a la fecha, será necesario hacer uso de la librería datetime. Además, pueden existir dos o más publicaciones con el mismo nombre, pero estas se diferenciarán por su id. Este identificador debe cumplir la restricción de ser único para cada publicación. Cabe mencionar que **la descripción puede contener comas**.

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

```
id_publicacion,nombre_publicacion,usuario,fecha_creacion,precio,descripcion
1,Balón de basketball,Juampisaez,2021/05/27 06:43:28,10000,Balón de tamaño 7.
2,Jeans,mmunoz12,2021/06/09 16:55:53,30000,Jeans talla L con poco uso en venta.
3,Bicicleta,nanglada,2021/05/25 05:34:28,350000,Vendo mi TREK marlin 5 por viaje.
4,Calefactor,matiasmasjuan,2021/06/08 13:21:46,50000,Calefactor para los días fríos.
```

5.3. `comentarios.csv`

Este archivo contiene la información de todos los comentarios del sistema. Cada línea posee cuatro datos relevantes para un comentario, separados por comas (`","`): El ***id*** **publicación**, el **usuario** que comenta, **fecha de emisión** (en formato `yyyy/mm/dd hh:mm:ss`) y el **contenido**. Cabe notar que **el contenido de un comentario puede poseer comas**, por lo que cualquier coma encontrada después de la que separa la fecha de emisión del contenido, es parte de este último [3].

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

```
id_publicacion,usuario,fecha_emision,contenido
1,matiasmasjuan,2021/05/24 01:32:23,Me interesa el computador! tienes algun mail?
2,knight-who-say-ni,2021/05/25 18:56:03,Lo quiero! que color es?
1,pedroriosg,2021/05/26 17:43:07,Nice! lo necesito para progra avanzada
3,catalina-arcila,2021/05/27 07:08:43,Sirve para jugar en linea?
```

5.4. `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. En tu tarea deberás **importar**[4] correctamente este archivo y utilizar los parámetros almacenados.

Particularmente, este archivo contiene el rango mínimo y máximo que deben tener el nombre de un nuevo usuario al momento de registrarse. Por lo tanto, deberás usar las variables de este archivo para verificar si el nuevo usuario que se quiere registrar es considerado válido o no. El contenido del archivo `parametros.py` es el siguiente:

[3] Podría ser útil el parámetro **maxsplit** del método split de strings.
[4] Para mas información revisar el material de modularización de la semana 0.

```python
# Mínimo de caracteres que debe tener el nombre de un nuevo usuario
MIN_CARACTERES = 1
# Máximo de caracteres que debe tener le nombre de un nuevo usuario
MAX_CARACTERES = 15
```

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[5]. En este curso te pediremos seguir un pequeño apartado de estas reglas, el cual puede ser encontrado en la **guía de estilo**.

- **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.

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 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,

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

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` o si se entrega incompleto en el mismo estado inicial. 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[6]. 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.

- ***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.

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

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.[7]

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

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 al siguiente correo: **bienestar.iic2233@ing.puc.cl**.

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 3.8.
- 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.

[7] Lamentablemente basado en una historia real.

- 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 2 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).