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

IIC2233 Programación Avanzada (2024-2)

Bases Generales de Tareas

1. *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[1]. En este curso te pediremos seguir un pequeño apartado de estas reglas, el cual se detallarán en la sección Descuentos.

- **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. Además, no se aceptará el uso de punto y coma (`";"`) dentro del código como forma de reducir la cantidad de líneas de este. En caso de utilizar punto y coma (`";"`), se reordenará el código para eliminar este carácter y luego se revisará el número de líneas.

Normalmente, estos dos aspectos son considerados como descuentos. A manera de excepción, para la primera tarea (T1) serán parte del puntaje de la evaluación, buscando que los apliques y premiando su correcto uso. Ten en cuenta que en las siguientes tareas (T2, T3 y T4), funcionarán como cualquier otro descuento de la sección Descuentos.

2. 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 que programaste (puedes usar inglés o español), incluir las referencias, e indicar **qué cosas fueron implementadas, se encuentran incompletas, presentan errores o no fueron implementadas**, 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.

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

Al iniciar cada Tarea, encontrarás en el repositorio de Syllabus dos archivos `README.md` y `README_inicial.md`. El primero contendrá las actualizaciones a la tarea aplicadas hasta ese momento. El segundo es la base desde la cual deberás construir tu propio archivo `README`.

Se **recomienda fuertemente** ir completando este archivo a medida que se desarrolla la tarea. De este modo, se asegura la creación de un buen y completo `README.md`. Aprovecha este documento para dejar un registro de las funcionalidades que has completado y las que se encuentran incompletas o presentan un error, así tendrás más claridad de qué puntos de la tareas has completado y cuáles presentan problemas.

3. `.gitignore`

Para todas las tareas **deberás utilizar un** `.gitignore` para ignorar ciertos archivos indicados. Este archivo deberá estar dentro de la carpeta de la tarea correspondiente (`T1/`, `T2/`, `T3/` o `T4/`). Puedes encontrar un ejemplo de `.gitignore` en el siguiente ***link***.

En cada tarea se indicará cuáles son los archivos específicos a ignorar. Además, se espera que ignores todo archivo auto-generado por las interfaces de desarrollo o los entornos virtuales de Python, como por ejemplo: la carpeta `__pycache__`, `ipynb_checkpoints`, `.idea`, `.DS_STORE`, `.vscode`, etc.

Para este punto es importante que hagan un correcto uso del archivo `.gitignore`, es decir, los archivos no **deben** subirse al repositorio debido al archivo `.gitignore` y no debido a otros medios.

4. 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:

4.1. PEP8 (hasta 4 décimas)

PEP8 es el estándar que se utiliza para programar en Python, por lo que es importante que se sigan las convenciones. Todo tipo de detalle sobre estas convenciones se encuentran en la **guía de estilo**. De los cuales sólo se verificarán:

- No exceder el máximo de 100 caracteres por línea.
- Uso de variables declarativas y aclarativas.
- Uso adecuado de *CamelCase* y *snake_case*.
- Espacio después de la coma `","`.
- Uso de espacios para indentación y no de tabulaciones.
- Uso de variables globales.
- Utilizar correctamente los import (no hacer `import *`). Para más información de cómo importar archivos, revisar los contenido del curso.

La cantidad de décimas a descontar será la siguiente (acorde a lo establecido arriba):

- 2 décimas si no se cumple sólo 1 punto.

- 3 décimas si no se cumplen 2 puntos.
- 4 décimas si no se cumplen 3 o más puntos.

La búsqueda de estos tipos de errores tampoco debe ser una búsqueda exhaustiva, pero es altamente probable que los *linters* se los muestren. Si sólo ven un error de algún tipo siendo que en el resto de los casos no ocurre, entonces se puede perdonar.

4.2. `README` (hasta 5 décimas)

- Se descontará 1 décima si no se indica(n) los archivos principales que son necesarios para ejecutar la tarea o su ubicación dentro de su carpeta.
- Se descontará hasta 4 décimas si se sube el `README`, pero este se encuentra incompleto, sin especificar los aspectos implementados, incompletos y no implementados o sin mencionar posibles errores dentro del código. Para más información, pueden revisar la sección de README.

4.3. Modularización (5 décimas)

Se descuentan 5 décimas si uno o más archivos .py exceden las 400 líneas de código. En caso de utilizar punto y coma (`";"`) para reducir la cantidad de líneas, se reordenará el código para eliminar este carácter y luego se revisará el número de líneas.

4.4. Uso de `.gitignore` (hasta 5 décimas)

El `.gitignore` es un archivo que permite indicar cuales archivos deben ser ignorados del repositorio, es decir, que no son detectados dentro de este al momento de querer hacer `git add`. Para más información visitar **este link**.

Este descuento sólo se aplica si en el enunciado de la tarea se especifica que **utilicen el `.gitignore`** o se indica que **no deben subir algún archivo en específico**. En caso de solicitar que usen el `.gitignore`, el descuento puede ser por uno de los 3 siguientes casos:

- **5 décimas** si no está el `.gitignore` y sube los archivos a ignorar de todos modos.
- **2 décimas** si no está el `.gitignore` pero no están subidos los archivos a ignorar. Este descuento es porque no se puede evaluar el correcto uso de `.gitignore`.
- **2 décimas** si sube igual los archivos a ignorar a pesar de tener el archivo `.gitignore`. Este descuento es porque no cumple con el objetivo del `.gitignore` o no lo utiliza correctamente.

Es importantes destacar que el archivo `.gitignore` debe estar dentro de la carpeta de la tarea (`T1/`, `T2/`, `T3/` o `T4/`), no se aceptará que lo pongan en otra parte aunque funcione.

4.5. Formato de entrega (hasta 10 décimas)

Se descontarán hasta diez décimas si es que no se siguen las reglas básicas de la entrega de una tarea, como son: el uso de groserías en su redacción, archivos sin nombres aclarativos, uso correcto de *paths* que permitan la ejecución de la tarea en cualquier computadora, no seguir restricciones del enunciado, entre otros. Esto se debe a que en próximos ramos (o en un futuro trabajo) no se tiene tolerancia respecto a este tipo de errores.

4.6. Cambio de líneas (hasta 5 décimas)

Se permite cambiar **hasta 20 líneas de código** por tarea en **únicamente archivos Python (.py)** para corregir un error. El cambio de líneas debe pedirse oportunamente **mediante el formulario de**

recorrección, si no se pide, no se cambiarán líneas en el código entregado. 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.

Finalmente, este descuento **solo aplica** para partes de tareas que sean corregidas manualmente. Toda tarea o parte de tarea que presente corrección automatizada **no permitirá cambio de línea**.

4.7. *Built-in* prohibidos (hasta 5 décimas)

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

4.8. Malas prácticas (hasta 5 décimas)

Dentro de la programación, existen ciertos códigos o prácticas que en primera instancia pueden funcionar, pero fácilmente pueden gatillar un error o dificultad en la programación cuando el código a confeccionar aumenta en su magnitud. Incurrir en estas prácticas implicará un descuento en la nota. Las prácticas que se penalizarán son:

1. ***Hard-coding* en el código:** es la práctica de ingresar valores directamente en el código fuente del programa en lugar de parametrizar desde fuentes externas. En toda tarea que se definan parámetros, deberás crear un archivo externo donde estén los valores de dichos parámetros, en vez de ingresar en cada parte del código el valor elegido. Más información en este **enlace**.
2. **No especificar excepción**: dentro de los contenidos a explorar en el curso, veremos el bloque `try/except` para poder interceptar e intervenir errores de ejecución en el código. **Siempre se debe explicitar el error en específico que deseamos intervenir**. Se penalizará el uso de `except:` o `except Exception:` para interceptar cualquier clase de error en vez de especificar el error en particular que se desea intervenir.
3. **Uso de conector de *path* incorrecto**: cuando se genera un *path* hacia un archivo en otra ubicación, se debe utilizar `os.path.join`, `Pathlib` o el conector `'/'` para confeccionar la ruta y asegurar que funcione en todo sistema operativo. Se penalizará si se utiliza otro conector, por ejemplo `'\'`, dado que esos conectores no funcionan en todo sistema operativo.

5. Des-descuentos

5.1. `README.md` (hasta 5 décimas)

Una de las principales herramientas que ayudan a la documentación y corrección de las tareas es el `README.md`. Con el fin de fomentar que completen dicho archivo con todo lo solicitado: archivo a ejecutar, qué ítems lograron realizar, referencias, etc., se otorgará un des-descuento de hasta **5 décimas** en los **descuentos misceláneos** de su tarea. Este "des-descuento" permite reducir el descuento recibido en la evaluación, pero no es una bonificación a la nota. A modo de ejemplo:

- Si su tarea tenía 10 décimas de descuento misceláneos, no tiene descuentos adicionales, y hace un excelente `README.md` para tener 5 décimas de des-descuento. Finalmente su tarea queda con 5 décimas de descuento.
- Si su tarea tenía 2 décimas de descuento misceláneos, no tiene descuentos adicionales, y hace un excelente `README.md` para tener 5 décimas de des-descuento. Finalmente su tarea queda con 0 décimas de descuento.

- Si su tarea tenía 2 décimas de descuento misceláneos, 3 de formato de entrega y hace un excelente `README.md` para tener 5 décimas de des-descuento. Finalmente su tarea queda con 3 décimas de descuento que son las del formato de entrega.

El cómo aplicar este des-descuento quedará a criterio del cuerpo docente, pero con tener un README.md que: (1) indique el archivo a ejecutar y la ubicación de los archivos necesarios para la ejecución del programa; (2) indique mediante emojis el estado de cada ítem y (3) solo contenga textos que aporten al README, es decir, elimine todo texto que no sea un porte a la corrección de tarea; será merecedor de algunas décimas de des-descuento.

5.2. *Commits* (2 décimas)

Otra herramienta fundamental en el curso es Git, que permite guardar avances parciales del trabajo sin necesidad de crear múltiples versiones de los archivos. Aunque su uso principal es para la entrega de evaluaciones (actividades y tareas), se otorgará un des-descuento de 2 décimas a quienes la utilicen más allá de la entrega final. En particular, si durante el desarrollo de la tarea, es decir, desde la publicación del enunciado hasta el último *commit* a revisar, se detectan 3 o más *commits* distintos, se otorgará el des-descuento.

Este criterio es binario: se cumple con 3 o más *commits*, o no se cumple; por lo que no se otorgará un des-descuento parcial para quienes tengan 1 o 2 *commits* durante la realización de la evaluación. Además, solo se espera que los *commits* sean diferentes dentro de las fechas indicadas, pero no se revisará la distancia de tiempo entre cada *commit*, el mensaje de cada uno o los cambios considerados en cada *commit*.

6. Cálculo del descuento

Finalmente, los descuentos finales aplicados sobre una tarea se calcularán de la siguiente forma:

```
max(min(descuentos_miscelaneos - des_descuentos, 10), 0) +
          descuento_por_usar_algo_prohibido + formato_de_entrega
```

Donde:

- `descuentos_miscelaneos`: es la suma entre los descuento de PEP8, Readme, Modularización, Uso de `.gitignore` y descuentos adicionales.
- `des_descuentos`: corresponde a las décimas de des-descuento obtenidas por el README entregado y si se cumple la cantidad de *commits* solicitada.
- `descuento_por_usar_algo_prohibido` es el descuento asociado a utilizar alguna librería o "Built-in" no autorizado.
- `formato_de_entrega` es el descuento asociado a no respetar el Formato de entrega.

7. Política de atraso

Se aceptarán entregas **hasta 48 horas** después de la hora de oficial, aplicándose una **penalización a la nota máxima** dependiente de la cantidad de días de atraso desde la hora de entrega oficial.

Las tareas entregadas con hasta 24 horas (un día) de atraso tendrán nota máxima **6.0**, mientras que las tareas con hasta 48 horas (dos días) de atraso tendrán nota máxima **4.0**. Cabe señalar que estas penalizaciones son a la nota máxima y **no son un descuento directo**, es decir, en caso de entregar tarde y obtener una calificación menor a la nota máxima penalizada, la nota obtenida no se verá afectada.

En otras palabras, la nota final de cada tarea atrasada se calculará como:

$$\mathbf{T_i} = \text{mín}\left(\mathbf{T_i^a}, \texttt{techo}^{\mathbf{a}}\right)$$

Donde $\mathbf{T_i^a}$ es la nota obtenida en la entrega atrasada y $\texttt{techo}^{\mathbf{a}}$ es la nota máxima dado los días de atraso **a** (definida anteriormente).

Cupón: Este cupón le permite al estudiante disminuir la penalización del atraso de una tarea en 24 horas. De este modo, si el estudiante entrega con dos días de atraso, tiene la posibilidad de usar el cupón para que se le aplique la penalización correspondiente a solo un día y, de la misma manera, si entrega con un día de atraso, puede utilizar el cupón para eliminar del todo la penalización y acceder a la nota máxima. En consecuencia, este cupón **no extiende el plazo de entrega**, solo disminuye la penalización.

Durante el semestre, cada estudiante dispondrá de dos (2) cupones que podrán ser utilizados en conjunto para una sola tarea (eliminando la penalización de dos días de atraso), o cada uno en tareas distintas.

Noticar atraso o uso de cupón

Una vez finalizado el plazo de cada tarea, se enviará un aviso, por Canvas, que incluirá un formulario para notificar cualquier entrega atrasada y/o uso de cupón. **Es responsabilidad del estudiantado responder el formulario dentro del plazo indicado en cada aviso o no se considerará la entrega atrasada y/o uso de cupón.**