Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación
IIC2233 — Programación Avanzada 2024-1

14 de Marzo de 2024
Actividad NO evaluada

Actividad 0
Entorno de trabajo

Entrega

- **Lugar:** Repositorio personal de GitHub — Carpeta: `Actividades/AC0`
- **Fecha máxima de entrega:** 18 de Marzo 20:00
- **Ejecución de actividad**: La Actividad será ejecutada **únicamente** desde la terminal del computador. **Los *paths* relativos utilizados en la Actividad deben ser coherentes con esta instrucción, y no pueden modificarse.**

Introducción

Ayudaremos a una tienda a digitalizar su experiencia de compra, permitiendo a un usuario agregar los productos disponibles a su canasta. En esta primera actividad nos familiarizaremos con las herramientas básicas de control de entorno de trabajo. Estas serán utilizadas durante todo el semestre. Además, introduciremos los archivos de *testing*, que ayudan a corroborar el avance y puntaje de cada actividad.

Flujo del programa

Esta actividad consta de tres partes, en las cuales se te pedirá que implementes funciones que permitan cargar información y manejar una sesión de compras.

1. Parte 1 – Repositorio en Github

Para empezar a trabajar, necesitaremos acceso a dos repositorios: el Syllabus del curso y el repositorio personal. Deberás crear tu repositorio personal siguiendo las instrucciones enviadas por el equipo docente (si ya crearon su repositorio correctamente lo podrán encontrar dentro de la página de GitHub del ramo). Luego, utilizando los comandos de git, deberás mantener en tu computador la versión más actualizada de `github.com/IIC2233/Syllabus/`. En ella encontrarás los archivos de la Actividad 0.

Archivos de código

En el directorio de la actividad encontrarás los siguientes archivos con código:

- Crear Entregar `main.py`: Deberás crear y completar este archivo según lo pedido.
- No modificar `test.py`: Este archivo de Python contiene los tests que podrás utilizar para ir viendo si lo desarrollado hasta el momento cumple con lo pedido.

- No modificar `utils/menu_print.py`: Este archivo de Python contiene funciones necesarias para visualizar la compra.
- No modificar `utils/items.dcc`: Contiene los datos de los ítems en el formato `'nombre, precio, puntos'`.

2. Parte 2 – Cargar datos

Para que puedas implementar correctamente las funcionalidades, te entregamos las siguientes clases **ya implementadas** en el módulo `entities.py`:

- No modificar `class Item`:
  Clase que representa un producto de la tienda. Incluye sólo el constructor:
  - No modificar `def __init__(self, nombre: str, precio: int, puntos: int) -> None`:
    Este es el inicializador de la clase, y asigna los siguientes atributos:

`self.nombre`	Un `str` que representa el nombre del producto.
`self.puntos`	Un `int` que representa la cantidad entera de puntos que acumularía un usuario al comprar el producto.
`self.precio`	Un `int` que representa el precio original del producto.

- No modificar `class Usuario`:
  Clase que representa a un usuario, el cual puede adquirir productos en la tienda.
  - No modificar `def __init__(self, esta_suscrito: bool) -> None`:
    Este es el inicializador de la clase, y asigna los siguientes atributos:

`self.suscripcion`	Un `bool` que representa si el usuario cuenta o no con suscripción, duplica los puntos de cada item agregado a la canasta.
`self.canasta`	Una `list` con instancias de `Item`.
`self.puntos`	Un `int` que representa la cantidad entera de puntos acumulados del usuario.

  - No modificar `def agregar_item(self, item: Item) -> None`:
    Método de clase que agrega un objeto del tipo `Item` a la lista `self.canasta`. Si el usuario cuenta con suscripción, se duplican los puntos del producto.
  - No modificar `def comprar(self) -> None`:
    Método de encargado de comprar todos los productos de la canasta, traspasando los puntos de cada producto comprado a los puntos del usuario. Finalmente, deja la canasta vacía.

Para cargar los datos y utilizar las clases anteriores, deberás crear las siguientes funciones dentro del archivo `main.py`:

- Crear `def cargar_items() -> list`:
  Esta función debe extraer la información sobre los productos disponibles desde archivo items.dcc. Finalmente, debe retornar una lista con una instancias de `Item` por producto disponible en el archivo.
- Crear `def crear_usuario(tiene_suscripcion: bool) -> Usuario`:
  Método que instancia un objeto de la clase `Usuario`, imprime información sobre este y lo retorna. Para imprimir al usuario, debe usarse el método correspondiente en `utils/menu_print.py`.

3. Parte 3 – Ejecución del programa

Para poder visualizar la simulación, debes agregar el siguiente bloque de código al final del archivo `main.py`, y completar lo indicado por las líneas comentadas, con tal de cumplir con lo pedido.

```
if __name__ == "__main__":
    # 1) Crear usuario (con o sin suscripcion)
    # 2) Cargar los items
    # 3) Imprimir todos los items usando los módulos de pretty_print
    # 4) Agregar todos los items a la canasta del usuario
    # 5) Imprimir la canasta del usuario usando los módulos de pretty_print
    # 6) Generar la compra desde el usuario
    # 7) Imprimir el usuario usando los módulos de pretty_print
```

La primera línea indica que el *script* contenido en el archivo `main.py` será ejecutado directamente al ser llamado desde la terminal. Todo el código que esté contenido dentro de esa instrucción condicional será ejecutado linealmente.

Para probar tu código completo de la actividad, puedes ejecutar estas líneas en la terminal dentro del directorio `Actividades/AC0`:

- `python main.py`: Ejecuta el código en la terminal, según el flujo del programa realizado.
- `python test.py`: Ejecuta los tests de prueba para comprobar si el código cumple con lo pedido en el enunciado.

Si es que el comando `python` no te funciona, prueba con `py`, `python3`, `python3.11` o `py3`.

4. Entregar

Para finalizar la Actividad 0 debes subir el archivo Entregar `main.py` a tu repositorio personal, en la ruta `Actividades/AC0`.

Notas

- Recuerda que la ubicación de tu entrega es en **tu repositorio de Git**.
- Se recomienda completar la actividad en el orden del enunciado.
- Si aparece un error inesperado, ¡léelo! Intenta interpretarlo.

Objetivos de la actividad

- Crear y modificar archivos en Python (.py).
- Importar y usar módulos de archivos en Python (.py).
- Familiarizarse con los comandos de Git.
- Descargar y actualizar contenido del repositorio del curso.
- Crear, modificar y actualizar el repositorio personal.
- Familiarizarse con la metodología de avance mediante *testing*.