PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2233 Programación Avanzada (I/2018)

Tarea 4

Entrega

- Entregable **obligatorio**
  - **Fecha/hora**: 19 de mayo del 2018, 23:59 horas.
  - **Lugar:** Google Form `https://goo.gl/forms/pTchURhdCQSh3Lev1`
- Tarea **obligatoria**
  - **Fecha/hora:** 27 de mayo del 2018, 23:59 horas.
  - **Lugar:** GitHub – Carpeta: `Tareas/T04/`
- `README.md` **obligatorio**
  - **Fecha/hora:** 28 de mayo del 2018, 23:59 horas.
  - **Lugar:** GitHub – Carpeta: `Tareas/T04/`

Objetivos

- Modelar e implementar un problema de simulación de eventos discretos.
- Aplicar elementos del paradigma OOP para solucionar un problema, que puede incluir uso de *properties*, *overriding*, herencia y *class methods*.
- Aplicar elementos del paradigma de programación funcional para solucionar un problema.
- Escribir *docstrings* para documentar el funcionamiento de un programa.
- Definir, utilizar y documentar supuestos que no contradigan el enunciado, para manejar casos que no estén especificados de forma exhaustiva.
- Resolver un problema altamente parametrizado para ofrecer flexibilidad en la creación y exploración de distintos escenarios.

Índice

1. Introducción

La universidad donde estudia Javier Pentos está a pocos metros del parque de diversiones más malo de Chile: *Nebiland*. Cada vez que Javier intenta planificar una salida con sus amigos a este parque, las responsabilidades universitarias le arruinan su plan. Para poder acercarse a una experiencia divertida, sólo le queda una opción: programar una simulación de un parque de diversiones y así poder asistir —al menos virtualmente— durante sus largas jornadas de estudio. Sin embargo, él no está en *Programación avanzada*, por lo que te deja la tarea a ti de resolverlo.

2. Descripción del problema

Nebiland es un parque que aún sigue abierto gracias a los distintos grupos de **Personas** que lo visitan o trabajan en él.

Por un lado, existen los **Clientes**, que son todos aquellos que deciden pagar una entrada para entretenerse y divertirse en el parque. Los clientes se clasifican según su edad, en **Adultos** y **Niños**, quienes tienen distinto comportamiento dentro del parque y distinto nivel de cansancio según las atracciones de diversión que visitan.

Por otro lado, existen los **Empleados**, quienes son encargados de hacer que los clientes tengan la mejor experiencia posible dentro de *Nebiland*. Los empleados se dividen de acuerdo a sus tareas en: **Operadores**, **Técnicos** y **Limpiadores**.

Dentro del parque, razón por la que una gran cantidad de personas deciden visitarlo, se encuentran las **Atracciones**. Cada una tiene distintas características y restricciones de acceso.

Por último, la cadena *SlowDely* tiene un monopolio de **Restaurantes** en *Nebiland*. En cada uno de estos restaurantes, niños y adultos recargan sus energías por medio de alimento.

A lo largo del día ocurren distintos eventos en *Nebiland*, tanto esperados como no esperados. Javier Pentos quiere entender cómo sería pasar una semana dentro del parque, para poder analizar todos los eventos que ocurran y poder comentar con amigos cómo sería pasar una semana en el parque, donde cada día se atiende de 10:00 hrs. a 19:00 hrs.

Para la simulación de este escenario tendrás que generar los mismos eventos que ocurren en *Nebiland* **mediante simulación por eventos discretos**. Cada uno de esos eventos sigue una distribución aleatoria, cuya implementación podrás encontrar en el módulo `random`, de la librería estándar de Python.

3. Entidades [35 %]

En la simulación, es posible identificar dos tipos de entidades distintas: las **Personas** (empleados y clientes) y las **Instalaciones** (atracciones y restaurantes).

3.1. Personas

Todas las personas tienen un nombre (*string*) y un apellido (*string*).

3.1.1. Clientes

Estos son los usuarios de las diferentes atracciones de *Nebiland*. Cada uno de ellos puede ser descrito por información que los diferencia de otros clientes. *Nebiland* sólo atiende clientes que han hecho una reserva; por lo tanto, se te entrega un archivo `arrivals.csv` con las horas de llegada de cada adulto para una semana,

donde se incluye el día y hora de llegada, presupuesto y la cantidad de niños que trae a su cargo.

Como información extra de cada cliente, deberás simular las siguientes variables e información:

- **Estatura:** el valor de esta variable distribuye normal $\mathcal{N}(\mu_{\text{rango edad}}, \sigma_{\text{rango edad}})$. Los clientes que entran a *Nebiland* no pueden medir menos de 60 cm ni más de 190 cm, ya que personas de otras alturas pueden salir heridas del parque y perjudicar su imagen.
  - Adulto: $\mu_{\text{adulto}} = 170$ cm. y $\sigma_{\text{adulto}} = 8$ cm.
  - Niño: $\mu_{\text{niño}} = 120$ cm. y $\sigma_{\text{niño}} = 15$ cm.
- **Energía:** representa el nivel de energía del cliente. Este valor es un índice que varía entre 0 y 1, y se ve reducido por el uso y espera en filas de las atracciones de diversión. Un cliente puede aumentar su energía descansando o comiendo en un restaurante. Al llegar al parque, los clientes cuentan con un nivel de energía de 1 y en ningún momento es mayor a este valor. En cada evento que modifica el nivel de energía de un adulto o de alguno de sus niños a cargo, si este llega a 0,1, entonces con 50 % de probabilidad todo el grupo se va del parque. Si el nivel de energía de algún miembro del grupo llega a 0, todo el grupo se va del parque.
- **Hambre:** representa qué tan hambriento está un cliente. Este valor es un índice que varía entre 0 y 1. Todos los clientes llegan a *Nebiland* con un nivel de hambre aleatorio que distribuye uniforme de 0,01 a 0,25 (incluyendo ambos extremos).
- **Paciencia:** representa el tiempo máximo que un cliente está dispuesto a esperar en una fila de acceso a cualquiera de las atracciones. Para cada cliente c, se tiene que el valor de esta variable distribuye normal $\mathcal{N}(\mu_{\text{paciencia}_c}, \sigma_{\text{paciencia}})$, donde los parámetros son:
  - $\mu_{\text{paciencia}_c} = (10 + \text{energía}_c \times 30)$ min
  - $\sigma_{\text{paciencia}} = 5$ min

  Una persona que supera su límite de paciencia sólo se retira de la fila si lo próximo que hará es distinto de ponerse en otra fila de una atracción (ver 3.1.1 para ver qué decisiones puede tomar un cliente).
- **Náuseas:** representa qué tan mareado está un cliente debido a las atracciones del parque. Al llegar al parque todos parten en el nivel 0 de náuseas, llegando como máximo al nivel 150. Si el nivel de náuseas de un cliente es estrictamente superior a 90 al momento de subirse a una atracción, entonces tiene un 60 % de probabilidades de vomitar. Cada vez que alguien vomita, su nivel de náuseas queda en 50. El nivel de náuseas es incrementado por los siguientes eventos:
  - Niño sube a una atracción: aumenta en 10 las náuseas del niño.
  - Adulto sube a una atracción: aumenta en 5 las náuseas del adulto.
  - Comer en un restaurante: aumenta en 30 el nivel de náuseas siempre y cuando el cliente se haya subido a una atracción justo antes de haber decidido ir al restaurante a comer.

Adultos

- Son clientes con 18 años o más. Cada uno de ellos tiene un presupuesto disponible para gastar en el parque, entre boletos para atracciones, comestibles y bebestibles.
- Tienen niños a su cargo. Cuando el adulto llega al parque, debe tener dinero suficiente para pagar su entrada, la de los niños de los que es responsable, el ingreso a las atracciones y la comida que ingerirán. Los precios para restaurantes y atracciones se definen, respectivamente, en los archivos `restaurants.csv` y `attractions.csv`.
- Al bajarse de una atracción pierden 0,15 de energía y aumentan en 0,05 su hambre.

- Los adultos toman todas las decisiones de qué hacer y cuándo hacer algo en *Nebiland*. Las decisiones se toman al entrar al parque o al finalizar una actividad. Cada vez que se toma una decisión, el grupo se demora 10 minutos en comenzar a ejecutarla. Las actividades posibles son:

  - **Hacer fila para atracción:** existe un 70 % de probabilidad de tomar esta decisión. En este caso, el adulto elige cualquiera de las atracciones que aún no ha visitado de manera inversamente proporcional al largo de la fila de espera de cada atracción. Esto es, mientras más corta es la fila, mayor probabilidad tiene de escoger esa atracción. En caso de que ya haya visitado todas las atracciones, comienza a repetirlas.
  - **Restaurante:** existe una probabilidad de r % de elegir esta opción. En este caso, el adulto elige con igual probabilidad cualquiera de los restaurantes a los que puede ir a comer con sus niños a cargo. Esto disminuye el hambre de todos los miembros del grupo en 0,06 y aumenta la energía de cada uno en 0,2. En caso de que no haya un restaurante disponible, elige la opción "descansar" y luego vuelve a tomar una decisión.

    $$r = \text{hambre promedio del grupo a cargo del adulto} \times 30$$

  - **Descansar:** si no elige ninguna de las opciones anteriores, entonces decide descansar. En este caso, el adulto decide descansar con sus niños por una hora, aumentando así la energía de cada uno en 0,2. No olvides que la energía posee un valor máximo.

- En el momento que un adulto no puede costear ninguna atracción para él y los niños que vienen a su cargo, se retira del parque.

Niños

- Son clientes que tienen entre 1 y 17 años.
- Al bajarse de una atracción pierden 0,05 de energía y aumentan su hambre en 0,1.
- Al bajarse de una atracción, un niño que no vomitó llorará con una probabilidad inversamente proporcional a su edad. En caso de llorar, su energía disminuye en 0,1 y la de su adulto responsable en 0,2.

3.1.2. Empleados

Todas las personas que trabajan dentro del parque, son considerados empleados.

- Todos los empleados atienden durante todo el horario en que el parque está abierto.
- Cuentan con un horario de colación, durante el cual dejan de trabajar por una hora.

  Para tomarse esta hora libre, primero deben terminar lo que estaban haciendo. Pueden elegir cualquier hora del día para descansar. Además, cada empleado elige su hora del día de manera independiente de los demás empleados. La probabilidad de elegir cierto horario varía según el cuadro 1.

Rango horario	Probabilidad
10:00 – 10:59	5 %
11:00 – 11:59	5 %
12:00 – 12:59	30 %
13:00 – 13:59	20 %
14:00 – 14:59	10 %
15:00 – 15:59	10 %
16:00 – 16:59	5 %
17:00 – 17:59	5 %
18:00 – 18:59	10 %

Cuadro 1: Probabilidad horaria de colaciones

Operador

- Un operador es quien atiende a los clientes.
- Existen 3 operadores asignados a la portería y venta de *tickets* del parque. Sólo puede haber uno de estos operadores en horario de colación a la vez.
- Existe un único operador asignado por atracción.
- El operador designado a la portería solo admite a una cantidad X de personas en el parque. Mientras la cantidad de personas en el parque sea menor a X, el operador designado a la portería permitirá el acceso de nuevos visitantes. Esta cantidad (X) se calcula como:

$$\left(\sum_{i=1}^{\text{cantidad\_instalaciones}} \text{capacidad\_instalación}_i \right) \times 1{,}2$$

Limpiador

- Un limpiador es quien se encarga del aseo de las atracciones del parque.
- Existe un limpiador por cada dos atracciones en el parque.
- Cuando una atracción supera su límite de suciedad, se llama a un limpiador, quien espera que la atracción esté detenida para hacer el aseo.
- Los llamados a hacer aseo son atendidos en orden FIFO (*First in, first out*).
- Un limpiador se demora 15 minutos en desplazarse de un lugar a otro.
- El tiempo invertido en la limpieza depende de la atracción y del nivel de suciedad (ver la sección 3.2.1).

Técnico

- Un técnico es quien repara o efectúa la mantención de las atracciones.
- Hay un técnico por cada tres atracciones en el parque.
- Los llamados de reparación y mantención son atendidos en orden FIFO.

- Cada vez que la atracción esté a punto de iniciarse, el operador debe revisar si está averiada o si fallará mientras esté en funcionamiento. En caso de que esté averiada o vaya a fallar, el operador debe evacuar la atracción y llamar al técnico asociado.
- Inmediatamente luego de llamar al técnico, y hasta que éste termine su trabajo, la atracción está en estado *Fuera de servicio*. Durante este tiempo, los clientes no pueden utilizar ni elegir la atracción, pero sí pueden permanecer en la fila de espera.
- Un técnico se demora 15 minutos en desplazarse de un lugar a otro.
- Un técnico se demora 20 minutos en realizar la reparación o mantención de la atracción.

3.2. Instalaciones

Las instalaciones en *Nebiland* se pueden clasificar en dos: atracciones de diversión y los restaurantes de *SlowDely*. Cada instalación puede ser descrita por un nombre y una capacidad máxima de clientes permitidos simultáneamente.

3.2.1. Atracciones de diversión

Las atracciones son lo más importante del parque, ya que entre más atracciones, más clientes querrán visitar el parque. Todas las atracciones están operativas en el mismo horario del parque. Puedes encontrar un listado público con todas las atracciones disponibles en *Nebiland* en el archivo `attractions.csv`. Cada atracción puede ser descrita por las siguientes características presentes en el listado público:

- **Nombre:** identificador único de cada atracción.
- **Tipo:** acuáticas, aéreas o terrestres.
- **Costo:** diferente para adultos y niños. El adulto responsable de un grupo debe pagar lo correspondiente al grupo completo para poder disfrutar de la atracción.
- **Capacidad:** cantidad máxima de personas que pueden participar en una vuelta del juego.
- **Duración:** cantidad de minutos que dura una vuelta del juego.
- **Operador:** quién se encarga de atender al público, además de cobrar la entrada.
- **Altura mínima permitida:** sólo se pueden poner en su fila de espera los grupos de clientes donde todos sus integrantes superen esa altura.
- **Tiempo entre fallas:** cantidad de tiempo en minutos que pasará hasta la siguiente falla. Este valor distribuye de manera exponencial con una tasa f, la cual se calcula para cada atracción como $f = (\text{capacidad} \times \text{duración})^{-1}$.
- **Límite de suciedad:** indica qué tan sucia puede llegar a estar una atracción. La suciedad se mide por un índice de suciedad que aumenta a medida que los clientes usan la atracción o vomitan en ella. Por cada persona que se sube a la atracción, la suciedad aumentará en una unidad. En caso de que alguien vomite, este índice aumentará en 40 unidades.
- **Tiempo máximo entre clientes:** tiempo máximo que se esperará a que la atracción se llene para partir. Si se supera este tiempo, la atracción partirá aunque queden puestos libres en ella.
- **Tiempo de limpieza:** tiempo en minutos que el limpiador se demora en dejar limpia la atracción. Se calcula como:

  $$\text{mín}(\text{suciedad} - \text{límite de suciedad}, \text{tiempo máximo}) \text{ minutos}$$

  El parámetro *tiempo máximo* es por defecto 10.

3.2.2. Restaurantes

Los clientes van a un restaurante a descansar y alimentarse. Es por esto que *Nebiland* ha querido hacer de este servicio lo más expedito posible. Cada restaurante es de autoservicio, y cuenta con la siguiente información descriptora:

- **Nombre:** denominación de cada restaurante, que podría estar repetida.
- **Capacidad:** cantidad máxima de personas que pueden servirse comida a la vez.
- **Costo:** diferente para adultos y niños.
- **Juegos asociados:** cada restaurante exige que hayas pasado por al menos uno de los juegos que tiene asociado. Así se previene que todos se llenen o que los clientes vayan al parque solo a comer.
- **Tiempo de preparación:** distribuye exponencial con media de 6 minutos para platos de adultos y media 4 minutos para plato de niños.
- **Tiempo máximo para comer:** como es autoservicio, una vez que cada integrante del grupo tiene su pedido, hay un límite de tiempo en minutos permitido por grupo para estar en el recinto. Todos los grupos pasan ese tiempo máximo en el comedor.

4. Eventos externos al parque [12 %]

- **Lluvia:** la dirección meteorológica señala que el tiempo entre lluvias, distribuye de manera exponencial con parámetros $\lambda = \frac{1}{20}$ (*lluvias/días*). Cuando llueve, las atracciones acuáticas se cierran y sus operadores están disponibles para cubrir horarios de colación de otros operadores. Además, sólo son permitidos en el parque adultos con máximo un niño a cargo, para así controlar los accidentes por descuido ante la lluvia.
- **Invasión *Ruziland*:** lunes por medio existe una probabilidad de 25 % de que el parque sea invadido por su archienemigo. Cuando esto ocurre la popularidad del parque disminuye; por lo tanto, el 40 % de los visitantes adultos deciden ser cuidadosos y abandonan el parque junto a todos sus niños a cargo. Además, el tiempo entre fallas de las atracciones se reduce a la mitad.
- **Día colegio:** cada día sábado, existe un 50 % de probabilidad de que un colegio haya reservado el parque exclusivamente para sus alumnos. En ese día de paseo, el parque sólo admitirá grupos de visitas donde por cada adulto hayan al menos 10 niños. En este día los niños no necesitan estar acompañados por un adulto en las atracciones, excepto en los restaurantes, donde tiene que estar disponible el adulto responsable para que pague las comidas. En cualquier otro caso, el presupuesto se divide equitativamente por cada niño. Los adultos entran gratis, pero no pueden subirse a los juegos, por lo que sólo acompañan a los niños en los restaurantes.

5. Registro de eventos [14 %]

Para tener un registro de todo lo sucedido en el parque, deberás crear un archivo (de extensión `.txt`) que contenga **todos los eventos ocurridos durante la simulación**. Este archivo debe tener como nombre `log.txt`, y se actualizará a medida que los eventos ocurran.
Cada *log* debe contener:

- **Iteración**: indica el número de la simulación actualmente ejecutada.
- **Día de ocurrencia**: indica el día en el cual se generó el evento.
- **Tiempo**: indica la hora de ocurrencia en la cual se generó el evento.

- **Evento**: breve descripción del evento ocurrido.
- **Entidad(es) afectada(s)**: listado de entidades afectadas (*e.g.* Operador, Atracción, Limpiador, Cliente, etcétera). Se debe indicar el tipo de entidad, no la información de la entidad en específico.
- **Descripción de la entidad**: indica el nombre de la entidad generadora del evento.

El formato que debe tener el archivo `log.txt` es el siguiente:

Iteración | Día de ocurrencia | Tiempo | Evento | Entidad(es) afectada(s) | Descripción de la entidad

Importante: en esta parte se evalúa que el evento **esté registrado correctamente en el archivo**. Si implementas un evento (código), debes asegurarte que se guarde su información en `log.txt`.

6. Estadísticas [14 %]

El *output* o salida de la simulación son conocidos como medidas de desempeño, ya que posteriormente permiten comparar resultados de la simulación para los distintos *inputs*. En esta sección, deberás desplegar en consola estadísticas de las medidas de desempeño de la simulación. Específicamente, tendrá que desplegar:

1. Tiempo promedio de espera en fila de acceso a las atracciones
2. Promedio de llantos diarios por atracción
3. Total de personas que se van del parque a causa de pérdida de energía
4. Energía promedio de las personas[1] al momento de salir del parque
5. Total de dinero ganado por el parque en días de colegio
6. Total de fallas en las atracciones provocadas por una Invasión *Ruziland*
7. Atracción con mayor cantidad de fallas totales
8. Total de personas que no pudieron comer en los restaurantes[2]
9. Tiempo promedio que pasa un grupo en un restaurante
10. Tiempo máximo que una atracción permaneció en estado Fuera de Servicio
11. Tiempo promedio perdido[3] por persona en fila de acceso
12. Promedio de llamados totales a hacer aseo y reparación por Limpiador y Técnico respectivamente
13. Dinero total perdido por el parque en entradas debido a eventos externos
14. Dinero total no gastado por los visitantes

Para el cálculo de estas estadísticas, su simulación debe aceptar como parámetro la cantidad de iteraciones que simulará la semana del parque de diversiones. Posteriormente, el resultado final será el promedio de las estadísticas obtenida en cada simulación. Este parámetro es por defecto 5 y debe estar en el archivo `parameters.py`.

Ten presente que, en esta tarea, les entregamos las reglas principales de la simulación. Todos los casos bordes o partes no especificadas quedan a tu criterio, **mientras no contradiga algún punto del enunciado**. Por lo tanto, las estadísticas entre dos alumnos podrían presentar ligeras diferencias en base a las decisiones tomadas.

[1] Sólo cuentan las personas que no hayan salido por falta de energía.
[2] Debido a las varias restricciones de este.
[3] Se considera perdido todo el tiempo el que la persona esperó si esta no se termina subiendo a la atracción.

7. Parámetros [5 %]

Durante la simulación existen distintos datos que se mantienen **constantes a lo largo de la misma**. Estos datos se conocen como **parámetros**, y se caracterizan por ser valores que influyen en los resultados de la simulación.

Para esta tarea, todos aquellos parámetros de la simulación deben ser definidos en un archivo (`.py`) separado, que tenga por nombre `parameters.py`. De esta manera, podrán ser modificados con facilidad, permitiendo evaluar la simulación bajo distintos valores iniciales.

Considere que su tarea **no debe contener ningún valor numérico en su código**, todos esos valores estarán en `parameters.py`.

8. Archivos

En esta simulación se entregarán cuatro archivos en formato CSV. Recuerda que el orden de las columnas en los archivos CSV está dado por el *header*. Estos poseen la información de las personas inscritas para ingresa al parque, información de las atracciones, restaurantes y las asociaciones que tiene cada restaurante a cada atracción. Tu tarea **debe** utilizar estos archivos para realizar la simulación de forma correcta.

8.1. Llegadas

La información de los tiempos de llegada de cada adulto se encuentra en el archivo `arrivals.csv`, y cada fila representa la llegada de un adulto al parque.

Nombre	Tipo de dato	Explicación
`day`	`string`	Día de llegada. Puede ser "Lunes", "Martes", ... , "Domingo".
`time`	`string`	Tiempo de llegada. Este valor siempre estará entre las "10:00" y "19:00".
`budget`	`integer`	Presupuesto del adulto.
`children`	`integer`	Cantidad de niños que llegan asignados al adulto.

8.2. Atracciones

La información de las **atracciones** se encuentra en el archivo `attractions.csv`, y cada fila representa a una atracción.

Nombre	Tipo de dato	Explicación
id	integer	Identificador único de cada atracción.
name	string	Nombre de la atracción.
type	string	Tipo de atracción. Puede ser "Acuática", "Aérea" o "Terrestre".
adult_cost	integer	Costo de entrada de un adulto.
child_cost	integer	Costo de entrada de un niño.
capacity	integer	Capacidad máxima de clientes permitidos simultáneamente.
duration	integer	Duración en minutos de la atracción
min_height	integer	Altura mínima de la atracción para ingresar.
dirt_limit	integer	Límite de suciedad de la atracción.
max_time	integer	Tiempo máximo entre cliente.

8.3. Restaurantes

La información de los **restaurantes** se encuentra en el archivo `restaurants.csv`. Cada fila del archivo representa a un restaurante.

Nombre	Tipo de dato	Explicación
id	integer	Identificador único de cada restaurante.
name	string	Nombre del restaurante.
adult_cost	integer	Costo del almuerzo para un adulto en el restaurante.
child_cost	integer	Costo del almuerzo para un niño en el restaurante.
capacity	integer	Capacidad máxima de clientes permitidos simultáneamente.
max_duration	integer	Tiempo máximo para comer en el restaurante.

8.4. Asociaciones

La información de las **asociaciones** de cada restaurante a un conjunto de atracciones se encuentra en el archivo `associations.csv`, y cada fila representa a las asociaciones entre un restaurante y una atracción. El restaurante puede estar asociado a múltiples atracciones y una atracción puede estar asociada a múltiples restaurantes.

Nombre	Tipo de dato	Explicación
restaurant_id	integer	Identificador del restaurante.
attraction_id	integer	Identificador de la atracción.

9. Generar datos [5 %]

Para esta tarea, la edad y nombres de cada cliente y empleado **deben** ser generados de forma aleatoria durante la simulación. Para la edad, puedes utilizar cualquier distribución de probabilidad que estimes apropiada; para los nombres, estos **deben** ser generados mediante la librería *Faker* que podrás encontrar en este enlace: https://github.com/joke2k/faker.

10. Documentación [6 %]

La documentación de software son todos aquellos comentarios que acompañan al código, con la finalidad de explicar cómo opera y cómo usarlo. En general, una buena documentación describe cada funcionalidad al usuario, y lo asiste en su uso. Por esta razón, es importante que la documentación sea lo más clara posible. Es importante recordar que un código se leerá muchas más veces de lo que se escribe.

En programación se utiliza una herramienta conocida como *docstring*, que consiste en un *string* encargado de documentar un segmento específico del código. En Python, existen normas establecidas para documentar el código de manera eficiente (PEP257).

Todos los módulos de la tarea deberán estar documentados. Cada método y función debe incluir su explicación e indicar el tipo de dato de los argumentos recibidos y retornados. Los métodos de la clase `object`, como `__str__`, `__add__` no requieren documentación, a excepción de `__init__` que si lo requerirá. A continuación se muestra un segmento de código con método de clases documentados. Puede utilizar la librería `typing` para apoyar su documentación.

```
class Auto:
    def __init__(self, patente):
        """
        patente: string
        """
        self.patente = patente

    def __str__(self):
        return self.patente

class Estacionamiento:
    def __init__(self, encargado, autos_dentro):
        """
        encargado: object Encargado
        autos_dentro: list[Auto]
        """
        self.encargado = encargado
        self.autos_dentro = autos_dentro

    def agregar_auto(self, auto):
        """
        Agrega un auto a la lista de autos_dentro
        Retorna None
        """
        self.autos_dentro.append(auto)

    def retirar_auto(self, patente):
        """
        Busca un auto según su patente y lo retira del estacionamiento
        patente Es un String
        Retorna un objeto clase Auto o None
        """
        index = 0
        while index < len(self.autos_dentro):
            if self.autos_dentro[index].patente == patente:
```

```
            return self.autos_dentro.pop(index)
        index += 1
    return None
```

11. Requisitos [6 %]

Se evaluará la calidad de tu código. En particular, tu código debe contener lo siguiente:

- *Properties*

  Uso apropiado de *properties* para el *setter* y/o *getter* de algunas características de las entidades.

- *Modelar la realidad*

  Se espera que tu código represente lo mayor posible la realidad. En particular, se evaluará en el código que las entidades no tengan "permisos adicionales" en las otras entidades; es decir, que una clase no pueda acceder, editar y/o eliminar atributos o métodos de otra clase cuando no corresponde.

  Por ejemplo, si un cliente guarda una instancia de la atracción dentro de sus atributos cuando se sube a una atracción, esta persona podría llegar a editar atributos de la atracción como su duración, pero esa facultad para editar no lo tiene el cliente.

  Para este requisito, no debes usar *referencia circular* cuando no corresponde en la modelación de clases. En caso de dudas, revisar el anexo del material de clases de simulación: `https://github.com/IIC2233/contenidos/blob/master/semana-08/Anexo-Evitar-referencias-circulares.ipynb`.

12. Entregable [3 %]

Para esta tarea deberás responder el *form* en donde se solicita:

- Entregar una lista de cada evento que su **simulación por evento discreto** deberá considerar. **Escribe 1 evento por línea**, es decir, no todos separados por comas.
- Explicar como piensan modelar en su tarea el evento *Día colegio*. Es importante mencionar las entidades relacionas, eventos involucrados y explicar con sus palabras como llevar el evento al código.

13. Restricciones y alcances

- Tu programa debe ser desarrollado en Python v3.6.
- Esta tarea es estrictamente individual, y está regida por el Código de Honor de la Escuela: Clickear para Leer.
- Tu código debe seguir la guía de estilos descrita en el `PEP8`.
- Todos los eventos deben ser calculados según una **simulación por eventos discretos**. Eventos desarrollados con cualquier otro tipo de simulación será considerado incorrecto y no serán evaluados.
- Si no se encuentra especificado en el enunciado, asume que el uso de cualquier librería Python está prohibida. Pregunta en el foro si es que es posible utilizar alguna librería en particular.

- El ayudante puede castigar el puntaje[4] de tu tarea, si le parece adecuado. Se recomienda ordenar el código y ser lo más claro y eficiente posible en la creación algoritmos.
- Debes adjuntar un archivo `README.md` donde comentes sus alcances y el funcionamiento del sistema (*i.e.* manual de usuario) de forma *concisa* y *clara*. **Tendrás hasta 24 horas después de la fecha de entrega** de la tarea para subir el `README.md` a tu repositorio.
- Crea un módulo para cada conjunto de clases. Divídelas por las relaciones y los tipos que poseen en común. **Se descontará hasta un punto si se entrega la tarea en un solo módulo[5].**
- No cumplir con alguno de los ítems de la sección **Requisitos** vendrá con un descuento asociado.
- Cualquier aspecto no especificado queda a tu criterio, siempre que no pase por sobre otro.

Tareas que no cumplan con las restricciones señaladas en este enunciado tendrán la calificación mínima (1.0).

[4] Hasta -5 décimas.
[5] No agarres tu código de un solo módulo para dividirlo en dos; separa su código de forma lógica