CHECKPOINT 6 – EJERCICIO TEÓRICO

01 - ¿Para qué usamos Classes en Python?

Las clases son mapeadores de objetos, por lo que las clases esencialmente te dan la capacidad de crear un modelo para objetos, lo que significa que las clases pueden tener datos dentro de ellas y también pueden tener funciones y pueden agregar comportamiento.

Las clases se usan para agrupar objetos y que estos trabajen en conjunto cuando se hace una llamada desde el exterior de la clase. Al hacer una llamada desde el exterior, se le proporciona unos valores y son con estos valores son con los que van a trabajar los objetos que hay dentro de la clase.

Las clases se pueden usar tantas veces como se necesite desde cualquier parte del código, por lo que son reutilizables y se ahorra mucho código y tiempo, tanto para crearlo como para hacer mantenimiento o cambios a lo largo del tiempo de uso.

La estructura básica de una clase según la guía de estilo de Python, los nombres de las clases se escriben con la letra inicial mayúscula y se usa Pascal Case, es decir la palabra “class” seguido del nombre de la clase con la primera letra en mayúscula y finalmente dos puntos “:”. Usando sangría (tal como se hace en las condicionales, bucles y funciones) se añaden los objetos que pertenecerán a la clase.

```
class NombreClase():
```

Ejemplo simple donde se ha definido la clase y se han añadido atributos generales que pertenecen a la clase y por lo tanto serán comunes a todos los objetos que creemos con ella:

```
class MoviesCharacters:
  character = "James Bond"
  type = "Action"
```

02 - ¿Qué método se ejecuta automáticamente cuando se crea una instancia de una clase?

El método __init__ (dunder init) es el que es llamado automáticamente por el intérprete de Python y se utiliza para realizar cualquier inicialización que sea necesaria para la instancia. Es algo que está disponible para las clases y la forma en que funcionan las funciones constructoras como init es que

esta es una función que se llamará automáticamente cada vez que crees una instancia de la clase.

Se usa para asignar valores iniciales a los atributos de una instancia de la clase.

Este método sirve para crear los atributos que van a poder tener el resto de objetos que hay dentro de la clase y se le proporciona al momento de hacer una llamada desde el exterior de la clase.

La estructura del método __init__ es la siguiente:

```
def __init__(self, dato1, dato2):
  self.dato1 = dato1
  self.dato2 = dato2
```

Un ejemplo práctico podría ser:

```
class Debt:
    def __init__(self, friend, money):
        self.friend = friend
        self.money = money


    def formatter(self):
        return f'{self.friend} debe: {self.money}€'
```

03 - ¿Cuáles son los tres verbos de API?

Los tres verbos principales son GET, POST y DELETE.

- GET: Sirve para consultar un recurso, es decir, pedir y traer información o datos. Una de las principales características en este caso es que no debe causar efectos secundarios en el servidor, no debe producir nuevos registros, ni modificar los ya existentes, Esto quiere decir, que no importa cuántas veces hagamos una petición GET, los resultados obtenidos serán los mismos.
  Por ejemplo, si se desea ver la información de una BBDD o los datos de un usuario se usaría GET.

- POST: Sirve para crear recursos nuevos. Cada llamada con POST debería producir un nuevo recurso. Normalmente, la acción POST se dirige a un recurso que representa una colección, para indicar que el nuevo recurso debe agregarse a dicha colección.

Por ejemplo, POST /idiomas para agregar un nuevo recurso a la colección idiomas.

- DELETE: Tal como se intuye por su nombre, este verbo sirve para eliminar datos o información, se puede usar tanto para eliminar un dato en concreto o para eliminar un grupo grande de datos (colecciones) con una sola orden.

04 - ¿Es MongoDB una base de datos SQL o NoSQL?

MongoDB es una base de datos (BBDD) de tipo NoSQL.

Una BBDD del tipo NoSQL quiere decir que no trabaja de forma obligatoria en lenguaje SQL, sino que tienen un uso más flexible y dinámico, permitiendo usar diferentes modelos de datos. Gracias a ello tiene una mejor adaptación a todo tipo de aplicaciones. Además, las opciones son open source por lo cual, no requieren pago de licencia.

Por lo tanto, es una BBDD no relacional que no cuenta con un identificador que relacione un conjunto de datos con otro y donde la información es organizada generalmente como documentos no requiriendo que los datos estén estructurados para poder manipularlos.

Además, permiten guardar datos de cualquier tipo sin requerir una verificación, y realizan consultas utilizando lenguaje JSON (JavaScript Object Notation).

05 - ¿Qué es una API?

API es el acrónimo de "application programming interface" o interfaz de programación de aplicaciones, en castellano.

Básicamente, es un lenguaje que nuestra aplicación puede usar y comunicarse con otra aplicación de terceros ya que es un grupo de datos estructurados que logra comunicarse entre diferentes sistemas o aplicaciones, pudiendo crear de esta forma, un vínculo entre servicios para transportar información o dar diferentes tipos de órdenes.

Por lo tanto, podemos decir que una API es un servicio similar a un sitio web o un servidor con el que puedes comunicarte, pero en lugar de recibir una página web, recibes datos.

Por lo general las APIs están estructuradas en JSON.

06 - ¿Qué es Postman?

Postman es una plataforma API para construir y utilizar APIs. Se trata de una aplicación que dispone de herramientas nativas para diversos sistemas operativos como Windows, Mac y Linux.

Se trata de una aplicación muy útil que permite comunicarse con API externas, siendo una forma muy útil para crear aplicaciones y pasar datos de un lado a otro.

Sin importar el tipo de curso que estés haciendo, puedes usar Postman, ya que no es específico de un lenguaje de programación o un marco, porque lo único que hace es preocuparse solo por la solicitud de datos. Debido a ello es realmente útil y es una herramienta que de uso prácticamente diario si se realiza desarrollo de API.

Tiene una interfaz gráfica con diferentes opciones para poder crear APIs y probar su funcionamiento. Tiene capacidad para trabajar con HTTP y HTTPS con lo que ayuda a poder depurar las APIs, también tiene varios métodos de solicitud disponibles, GET, POST o DELETE entre otras.

07 - ¿Qué es el polimorfismo?

El término polimorfismo tiene origen en las palabras poly (muchos) y morfo (formas), y aplicado a la programación hace referencia a que los objetos pueden tomar diferentes cambios y/o formas, lo cual significa que puede comportarse de manera diferente en distintos contextos.

Por lo cual podemos decir que el polimorfismo hace que una llamada se pueda comunicar con diferentes clases (accediendo también a sus objetos) y de esta forma poder recibir una respuesta diferente según se necesite. En otras palabras, dos objetos de diferentes clases pueden tener métodos con el mismo nombre, y ambos métodos pueden ser llamados con el mismo código, dando respuestas diferentes

Para implementar el polimorfismo, simplemente podemos definir diferentes métodos que compartan el mismo nombre pero que realicen operaciones distintas.

08 - ¿Qué es un método dunder?

En Python, los métodos dunder son métodos especiales que tienen como estructura, encerrar el nombre entre dos guiones bajos, tanto al principio como al final del nombre, con lo que quedaría de la siguiente forma __nombre__. Dunder es la abreviatura de double underscore.

El método Dunder en Python significa que se trata de un método que se le proporciona directamente desde el lenguaje Python, por lo que es importante que no se sobrescriban ni se cambien los datos, simplemente usándolo en el programa de la manera prevista.

Los métodos Dunder sirven para hacer algún comportamiento especial y concreto en los objetos que hay dentro de una clase.

Son métodos que permiten a las instancias de una clase interactuar con las funciones y operadores incorporados en el lenguaje. Típicamente, los métodos dunder no son invocados directamente por el programador, haciendo que parezca que son llamados por arte de magia. Por eso, a veces también se hace referencia a los métodos dunder como "métodos mágicos".

Sin embargo, los métodos dunder, esto no ocurre asi, simplemente son llamados implícitamente por el lenguaje, en momentos específicos

Ejemplos de métodos Dunder son los siguientes:

- __init__: Uno de los más importantes, es el que inicia la clase y trae los datos de la llamada del exterior.
- __repr__: Devuelve una cadena del objeto y se usa mayormente para depurar y desarrollar las clases.
- __str__: Devuelve una cadena de texto del objeto.

09 - ¿Qué es un decorador de python?

Los decoradores son funciones que modifican el comportamiento de otras funciones.

Es una función que puede cambiar otra función sin necesidad de modificar el código de la función original. Ayuda a extender y reutilizar funciones de forma sencilla sin tener que añadir nuevas.

Un decorador no es más que una función la cual toma como input una función y a su vez retorna otra función.

Un decorador debe empezar siempre con una arroba @ seguido del nombre de la función a la que se le va a asociar, sin necesidad de poner paréntesis o dos puntos finales.

Debajo del comienzo del decorador se escribe la función en concreto que va a ejecutarse.

Ejemplo:

```
class Garage:
  def __init__(self, size):
    self._size = size
```

```
    self.cars = []


  @property
  def size(self):
      return self._size


  def open_door(self):
    return "The door opens"
```