Pasos hacia la programación con Python (III)

Introducción

Por primera vez nos vamos a adentrar en el mundo de la programación orientada a objetos (object oriented programming), también conocida como OOP o POO. Es uno de los paradigmas de programación más relevantes de la actualidad y a ha dado lugar a la creación de programas como Java o C# directamente enfocados a trabajar bajo esta lógica.

En el caso de Python estamos ante un lenguaje multipropósito y por ello permite también orientación a objetos. Otro paradigma de programación que permitiría, sería por ejemplo el de la programación funcional.

Esta sección representa una introducción al uso de objetos y de este paradigma. Veremos los conceptos de clase, objeto e instancia, así como lo que es un método o una variable.

Para la teoría desarrollada en esta sección nos basamos en el libro “Python para todos” de Raúl Gonzalez Duque, en concreto el capítulo “Orientación Objetos”. También usaremos recursos del tutorial oficial de Python: https://docs.python.org/es/3.8/tutorial/classes.html así como del tutorial https://aprendepython.es/core/modularity/oop/

Orientación a Objetos (1)

Tenéis disponible esta introducción a POO en el repositorio de GitHub python_basics_2. Esta en el archivo introduccion_objetos_1.py

ejercicio1.py

Vamos a crear una clase llamada Persona. Sus atributos de instancia son: nombre, edad y DNI. Construye los siguientes métodos para la clase:

- Un constructor .
- mostrar(): Muestra los datos de la persona.
- esMayorDeEdad(): Devuelve un valor lógico indicando si es mayor de edad.

Instancia tres personas diferentes. Un niño, una mujer adulta y un hombre anciano. Tras instanciarlos asignales algunos atributos propios de cada uno.

ejercicio2.py

Crear una clase llamada calculadora. La clase debe tener una variable de instancia llamada Ans que se inicialice por defecto como None. En esta variable se guarda siempre el resultado de la última operación realizada. Definir los métodos suma, resta, división, multiplicación y potencia tal que tomen dos números y realicen la operación correspondiente. No deben devolver el valor, solo printar el resultado por pantalla y asignarlo a la variable de instancia Ans.

ejercicio3.py

Crear una clase llamada ReaderJSON. Debes definir la clase de tal forma que pueda ser usada de manera sencilla para la lectura de archivos JSON. La clase debe tener al menos el atributo json_path que será indicado a través del constructor. Así mismo debes crear el método read() que devuelve un diccionario con el contenido del JSON al que apunta el path.

Prueba a leer un archivo JSON con este reader (puedes crear uno de manera sencilla para el ejercicio). Después cambia en el objeto el atributo json_path para apuntar a otro archivo JSON diferente (debes crearlo tu también) y comprueba si has logrado que el método read te devuelva el contenido del nuevo archivo.

Bonus: Implementar una cache. Para evitar lecturas innecesarias puedes asignar el contenido de del JSON a una variable que se llame JSON_content cuando se llama al método read y usar esa variable para devolver el valor ya almacenado en el objeto en sucesivas llamdas. Trata de replicar el comportamiento exacto que teníamos sin cache.

Orientación a Objetos (2)

Tenéis disponible esta introducción a POO en el repositorio de GitHub python_basics_2. Esta en el archivo introduccion_objetos_2.py

ejercicio4.py

Crear las clases TrianguloEquilatero, Cuadrado y Rectangulo. Las clases deben tener como variable de clase el numero de lados (no es necesario que sea una variable oculta). El constructor debe tomar en cada caso la base y la altura y en el caso del cuadrado el lado. Implementar los métodos perímetro y área para cada una de ellas de tal forma que devuelvan estas magnitudes.

Instanciar un objeto de cada clase con unos atributos de instancia concretos y comprobar el funcionamiento de los métodos.

Nota: *Un triangulo equilátero tiene todos los lados iguales. Por ello su base es igual al lado y su perímetro será 3*base.*

ejercicio5.py

Crea una clase llamada Estudiante. Define como atributos de instancia nombre, edad y curso. Estos argumentos deben ser pasados al constructor. El constructor debe tener también un argumento de instancia llamado nota_media que se inicia por defecto como None. Este argumento de instancia debe ser oculto (privado). Definir los getters y setters para este atributo. Definir un método que se llame esta_aprobado que devuelva True o False en función de la nota media del alumno (aprobado por encima de 5).

ejercicio6.py

Crea una clase llamada Cuenta que tendrá los siguientes atributos: titular (que es una Persona, objeto de clase Persona creada en el ejercicio1 anterior) y cantidad (puede tener decimales). El titular será obligatorio y la cantidad es opcional (para ello debes usar los argumentos tipo keyword vistos anteriormente en funciones). El atributo cantidad no se puede modificar directamente, sólo ingresando o retirando dinero. Para ello deberás crear este atributo como oculto usando __. Construye los siguientes métodos para la clase:

- Un constructor.
- Los getters para cada uno de los atributos.
- mostrar(): Muestra los datos de la cuenta.
- ingresar(cantidad): se ingresa una cantidad a la cuenta, si la cantidad introducida es negativa, no se hará nada.
- retirar(cantidad): se retira una cantidad a la cuenta. La cuenta puede estar en números rojos.

Instancia una cuenta asociada a un objeto Persona y comprueba que todos los métodos funcionan como es debido.

ejercicio7.py

Crear una clase llamada Usuario que tiene los atributos de instancia: conectado (True o False e inicializado por defecto como False), nombre, empresa y rol (que puede ser lectura, escritura o admin), todos estos atributos debe recibirlos el constructor. Además el constructor declara una variable de instancia llamada inicio_conexion por defecto como 0 (esta variable no debe recibirla el constructor, si no, declarase dentro de él).

Además como variable de clase se declara la variable server_url = "www.miempresa.es/server1/mysql/" que debe ser de tipo oculto (privado) y la variable contraseña_server = "server1234" también de tipo oculto.

Crear un método que se llame set_inicio_conexion que llame al la función de Python time() y asigne el valor que devuelve esa función a la variable de clase inicio_conexion (recordar que para hacer esto debéis importar la función time usando: ***from time import time***).

Crear un método llamado activar_conexion que requiera la contraseña, compruebe que es correcta y si lo es cambie el valor de la variable conectado a True y haga una llamada al método set_inicio_conexion.

Crear getters y setters para las variables de tipo oculto.

Crear un método de clase que se llame activar_respaldo. Al llamar a este método se deben cambiar las variable de clase (lo que afectará a todos los objetos) __server_url y __contraseña_server. Para ello debes usar los setters definidos anteriormente para estas variables.

Una vez creada la clase comprueba que todo funciona como es debido creando una instancia de la clase Usuario.

Comprueba que solo tienes acceso a las variables ocultas a través del getter.

Comprueba que el método activar_conexion funciona como es debido y que modifica las variables de instancia descritas.

Comprueba que el método activar_respaldo funciona como es debido y que modifica las variables de clase descritas.