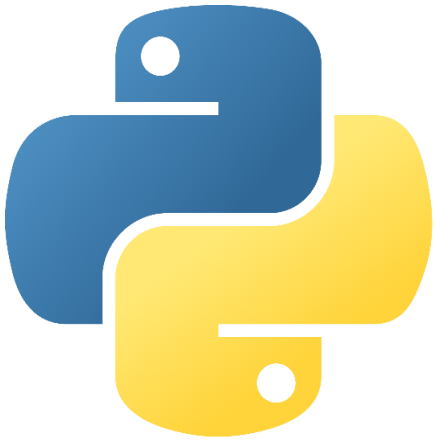

EJERCICIOS DE PYTHON IV

Clases y Objetos

- Ptogramación orientada a objetos (POO)
- Clases y Objetos
- Atributos instancia
- Métodos
- Método __init__()
- Método __str__()
- Método __de__()
- Colaboración de clases
- Atributos de clase
- Composición

Codo a Codo

2024

1) **Ejercicio 1:** Implementar una clase llamada Estudiante que tendrá como atributos (variables) su nombre, su apellido, dni y dos métodos (funciones), uno de dichos métodos inicializará los atributos y el siguiente método los mostrará en pantalla. Definir dos objetos de la clase Estudiante e incorporar una variable de clase (piernas).

2) **Ejercicio 2:** Implementar una nueva clase llamada Estudiante. Esta clase tendrá como atributos su nombre y su nota. Definir los métodos para inicializar sus atributos, imprimirlos y mostrar un mensaje que indique: “Promocionó” (nota >= 7), “Rinde final” (nota >= 4) o “Desaprobó”.
   Definir tres objetos de la clase Alumno, cada uno con una condición de aprobación distinta.

3) **Ejercicio 3:** Crear una clase que represente una Materia de la universidad. Definir como atributos su nombre, carrera, duración en meses y un atributo de clase booleano para definir que todas las materias no son promocionables. Desarrollar un método __init__ para incializarlos. Crear un método para imprimir los datos del objeto, luego sustituirlo por el método __str__(). Crear dos objetos. Eliminar uno de ellos a través del método __del__().

4) **Ejercicio Integrador:** Implementar una clase llamada Empleado, que posee un atributo de clase (nro_empleados) que lleva la cuenta de los objetos instanciados.
   Cada objeto de la clase Empleado posee un legajo, nombre completo y un sueldo.
   Definir métodos para inicializar sus atributos, definir su categoría (variable de clase), procesar su eliminación de la memoria y para mostrar un texto con la categoría asignada.
   La categoría es "Full Time" para los legajos comenzados en "F", "Part time" para los legajos comenzados en "P" y "Contratado" para los comenzados en "C", para el resto la categoría es vacía.
   En el programa principal instanciar distintos objetos de la clase Empleado y mostrar sus características encolumnadas. Al salir del programa eliminarlos de la memoria.

5) **Ejercicio 5 (colaboración de clases):** El comedor de la universidad tiene un sistema de beneficios a través del cual los alumnos pueden sumar puntos o canjearlos por menús. El comedor necesita al final del día un reporte de la cantidad de puntos que sus alumnos poseen. Crear 3 alumnos cuyos atributos son su nombre y la cantidad de puntos que poseen, inicializado en 0. Crear un método que simule las operaciones de ingresar puntos y de canjear puntos.

6) **Ejercicio 6 (colaboración de clases):** Se debe implementar un programa que permita jugar a un juego de dados. El juego sigue las siguientes reglas:
   - Se tiran tres dados al mismo tiempo.
   - Si los tres dados muestran el mismo valor, el jugador gana.
   - Si no se cumplen las condiciones del punto 2, el jugador pierde.

   El programa debe constar de las siguientes partes:
   a) Una clase llamada **Dado** que debe tener los siguientes métodos:
      - **tirar():** Este método debe generar un valor aleatorio entre 1 y 6 y asignarlo como el valor del dado.
      - **imprimir():** Este método debe imprimir el valor actual del dado.
      - **retornar_valor():** Este método debe devolver el valor actual del dado.
   b) Una clase llamada **JuegoDeDados** que debe tener los siguientes métodos:
      - **__init__():** Este método debe inicializar tres instancias de la clase Dado como atributos (dado1, dado2 y dado3).

- **jugar():** Este método debe simular una jugada del juego. Para ello, debe tirar los tres dados, imprimir los resultados y determinar si el jugador ganó o perdió según las reglas mencionadas anteriormente.
- **simular_jugadas(cantidad):** Este método debe simular una cantidad especificada de jugadas del juego. Debe llamar al método **jugar()** la cantidad de veces especificada y mostrar los resultados.

Resultado esperado: El programa debe simular 20 jugadas del juego de dados y mostrar los resultados de cada jugada.

7) **Ejercicio 7 (variables de clase):** Se debe implementar una clase **Alumno** que almacene la información de un estudiante, incluyendo un código de alumno, un nombre y su nota final en una materia. Además, la clase **Alumno** debe mantener dos listas de variables de clase, una de ellas llamada aprobados, que almacenará los códigos de los alumnos aprobados (nota mayor o igual a 4) y otra llamada reprobados que almacenará los códigos de los alumnos reprobados (nota menor a 4).
   La clase Alumno debe tener los siguientes métodos:
   - **__init__(self, codigo, nombre, nota):** Este método inicializa un objeto de tipo Alumno con el código, nombre y nota proporcionados. Además, invoca el método definir_estado() para determinar si el alumno está aprobado o reprobado.
   - **__str__(self):** Este método devuelve una representación en forma de cadena del objeto Alumno, que incluye el código, nombre y nota.
   - **definir_estado(self):** Este método determina si el alumno está aprobado o reprobado, y actualiza las listas aprobados y reprobados en consecuencia.

   A continuación, crear cuatro objetos de tipo Alumno con notas distintas y almacenarlos en una lista. Posteriormente, mostrar los datos de los 4 alumnos y finalmente mostrar la lista de los códigos de los alumnos aprobados y además los códigos de los alumnos reprobados, uno debajo del otro.

8) **Ejercicio 8 (composición):** Se debe implementar una clase **Materia** que gestione los datos de las materias de una universidad. Cada instancia de Materia debe tener los siguientes atributos: código, nombre y duración en meses.
   Además, la clase **Materia** debe implementar el método **__str__** para mostrar sus datos. La duración en meses debe ser representada de la siguiente manera: "Cuatrimestral" (4 meses), "Anual" (8 meses) o "A confirmar" (en el resto de los casos).
   Luego, se debe definir la clase **Carrera**, que administra una lista de materias. Cada materia en la lista es un objeto de la clase Materia.
   La clase **Carrera** debe tener los siguientes métodos:
   - **__init__(self, nombre, materias=[]):** Este método inicializa una carrera con un nombre y una lista opcional de materias. Por defecto, la lista de materias está vacía.
   - **agregar(self, mat):** Este método permite agregar una materia (representada por un objeto de la clase Materia) a la lista de materias de la carrera.
   - **__str__(self):** Este método devuelve una representación en forma de cadena de la carrera, incluyendo su nombre y el listado de materias.

   A continuación, crear una Materia e instanciar una carrera con esa materia. Luego agregar se agregan materias adicionales e imprimir el listado de materias de la carrera.