Grado en Ingeniería Informática

Inteligencia Artificial

Curso 2024/2025

Universidad de Jaén

Práctica 1:
Python

Índice

1. Introducción a programación dirigida a objetos en Python

La definición por nuestra parte de una nueva clase se traduce en la aparición de un nuevo tipo de dato (la clase) del que podremos declarar nuevas instancias (los objetos) y almacenarlos en la memoria del ordenador e incluso organizarlos en vectores o utilizarlos como parámetros de funciones, etcétera.
La declaración de una nueva clase se lleva a cabo utilizando las distintas técnicas relacionadas con la abstracción ya citadas en teoría: descomposición modular, especificación, ocultamiento de información y parametrización.
En Python las clases se declaran en un fichero, el cual contendrá tanto la declaración de la clase (que atributos y que métodos tiene), como la implementación de la misma (implementación de los métodos).

1.1. Ocultamiento de información

La programación orientada a objetos permite restringir el acceso a los atributos y métodos de una clase; así, algunos pueden ser accesibles (públicos) desde otros métodos y funciones externos, o bien estar disponibles sólo desde los métodos propios de la clase (atributos y métodos privados).
El siguiente código muestra una clase para la cual:

- El constructor es el método __init__
- Dentro del constructor se declaran los atributos de los objeto. Un subrayado indicaría que el atributo es protegido. Dos subrayados que el atributo es privado.
- También se definen el resto de métodos, que en este caso son públicos.

En la siguiente página podemos ver un ejemplo de objeto.

```
class Libro():  1 usage
    def __init__(self, tituloInicial, autorInicial): #este es el constructor
        self.__titulo = tituloInicial #se crea el atributo titulo (privado)
        self.__autor = autorInicial #s crea el atributo autor (privado)

    def get_titulo(self):  1 usage
        return self.__titulo # si se quiere acceder a atributos o métodos del objeto
                             # se tiene que usar el self.

    def set_titulo(self, valor):
        self.__titulo = valor

    def get_autor(self):  1 usage
        return self.__autor

    def set_autor(self, valor):
        self.__autor = valor

    def get_titulo_y_autor(self):  1 usage
        return self.get_titulo() + " y " + self.get_autor()

libro = Libro( tituloInicial: "Python",  autorInicial: "Yo") # se crea el objeto libro

print(libro.get_titulo_y_autor()) #el parámetro self del objeto no se usa en las llamadas
```

1.2 Instanciación de objetos

Una vez declarada una nueva clase, esto es, un nuevo tipo de datos, puede ser utilizada para declarar variables de dicho tipo. Dichas variables son los objetos que se utilizarán a lo largo de la ejecución del programa.
La instanciación de un objeto se realiza mediante una llamada al constructor de la clase. Si utilizamos la clase Libro declarada previamente podríamos definir las siguiente variable con un objeto libro:

```
libro = Libro( tituloInicial: "Python",  autorInicial: "Yo") # se crea el objeto libro
```

Es importante tener en cuenta que a la hora de crear una instancia de un objeto ocurren realmente dos cosas: por un lado, se reserva memoria para almacenar su estado (atributos); por otro, se deben inicializar sus atributos o recursos que utiliza en general.

1.3 Paso de mensajes

Finalmente, entre la definición de un nuevo objeto y su destrucción, podremos operar con él mediante llamadas a sus métodos públicos. La llamada a un método es lo que en Programación Orientada a Objetos se denomina "paso de mensaje" a un objeto.
En el ejemplo anterior son los métodos del objeto get_titulo, set_titulo, get_autor, set_autor y get_titulo_y_autor().

Para llamar a un método de un objeto, lo hacemos mediante el operador "." Tal y como puede verse en el siguiente ejemplo:

```
print(libro.get_titulo_y_autor())
```

1.4 Relación de generalización especialización

Una de las relaciones más características dentro de la POO es la de generalización-especialización, que se traduce a la hora de programar en la utilización de herencia para derivar nuevas subclases a partir de una superclase.
Dado que el concepto de herencia lleva asociado otro conjunto de conceptos más complejos, haremos en esta práctica una primera aproximación que consistirá exclusivamente en la derivación de nuevas subclases.
Cuando hablamos de especialización de una clase nos referimos al hecho de que, a la hora de plantear nuestro diseño, por algún motivo en particular, observamos que existen objetos de la clase que tienen características y/o comportamientos específicos además de las características y comportamiento de la clase a la que pertenecen. En esta situación, podemos utilizar la relación de herencia para derivar una clase "especializada" que añada estado y/o comportamiento al de la clase padre. De esta forma, reutilizamos el código de esa clase padre y añadimos nuevos atributos o métodos que complementen o sobrecarguen el comportamiento de los objetos de la nueva clase hija.
La derivación de una subclase se realiza fácilmente de la siguiente forma:

```
class ComponenteOrdenador():  1 usage  new *
    def __init__(self, precio = 0):  new *
        self.__precio = precio


class Microprocesador(ComponenteOrdenador):  1 us
    def __init__(self,precio = 0, nucleos = 1):
        super().__init__(precio)
        self.__nucleos = nucleos
```

Hay que recordar que la clase derivada únicamente tendrá acceso a los atributos y métodos públicos o protegidos de la superclase, pero no tendrá acceso a los privados.
Es importante que no demos cuenta que en python, una subclase **no llamará a un método o a un constructor de la clase padre, si no se hace explícitamente**. Por lo tanto, si no se hace, no se crearán ni inicializarán los atributos de la clase padre.

1.5 Almacenamiento y recuperación de datos en ficheros

Para leer y escribir datos en un fichero podemos utilizar el formato CSV, en esta sección revisaremos los conceptos claves para la lectura de ficheros.
Antes de trabajar con un fichero es necesario abrirlo, para abrir un fichero usaremos la instrucción open.

```
fileobj = open(filename,mode)
```

fileobj es un objeto asociado al fichero que hemos abierto, *filename* es el nombre del fichero y *mode* indica el modo en el que se abre.
El modo tiene dos letras. La primera letra indica la operación (r para lectura, w para escritura, x para escritura pero solo si el fichero no existe, a indica añadir al final si el fichero ya existe). La segunda letra indica el tipo de fichero (t para texto, b para binario).
Una vez abierto el fichero, se pueden realizar las operaciones de lectura y escritura.
Es importante que una vez hayamos acabo de operar con el fichero lo cerremos con el método close.

```
fileobj.close()
```

Si queremos escribir un fichero de texto, tendríamos que hacer las siguientes operaciones:

```
texto = "Lorem ipsum dolor sit amet, consectetur elit, sed"

nombreFichero = "texto.txt"
modo = "wt"

fichero = open(nombreFichero,modo)

fichero.write(texto)
#print(texto, file=fichero)

fichero.close()
```

Para leer de un fichero de texto, podemos usar los métodos, read(), readline(), readlines(). read() leería un fichero completo y devolvería dicha lectura,

readline() leería linea a linea. Si está al final del fichero devolverá una cadena vacía (la cual, si se evalúa como si fuera un valor lógico, equivaldría a False). Readlines() leería el fichero entero, pero devolvería una lista de cadenas de caracteres donde cada elemento de la lista es una linea del fichero.
A continuación vemos un ejemplo de código que lee de un fichero.

```
nombreFichero = "texto.txt"
modo = "rt"

fichero = open(nombreFichero,modo)

lineas = fichero.readlines()

for linea in lineas:
    print(linea)

fichero.close()
```

Existe una estructura alternativa para operar con los ficheros, de forma que, el fichero sea automáticamente cerrado una vez hemos terminado de operar con el. Para ello usaremos la estructura siguiente:

```
with open('relativity', 'wt') as fout:
```

El primer ejemplo que hemos visto, quedaría de la siguiente forma:

```
texto = "Lorem ipsum dolor sit amet, consectetur elit, sed"

nombreFichero = "texto.txt"
modo = "wt"

with open(nombreFichero,modo) as fichero:
    fichero.write(texto)
```

1.6. Actividad

Diseñar un programa en Python para resolver los siguientes puntos:

1. Definir una clase Alumno, que contenga las siguientes características: Nombre, y primer apellido, DNI y correo electrónico. Además crear los métodos para consultar los valores de las características anteriormente nombradas. Y por último, escribir el constructor para que inicialice todos sus campos a unos valores que se le pasen como parámetros.
2. Definir un método que lea una serie de valores que se introduzcan por teclado y cree un nuevo objeto de la clase Alumno, inicializado con dichos

valores. Definir otro método que muestre los datos de ese alumno por pantalla.

3. Definir una clase Alumno_IA, subclase de la anterior, que contenga los siguientes campos adicionales: grupo de teoría, grupo de prácticas, nota asociada a las prácticas 1, 2, 3 y 4. Defina un contructor similar a la de la clase Alumno y los métodos para poder consultar estos nuevos campos
4. Definir un método que lea desde teclado las 4 notas de prácticas de un alumno de IA, actualice los campos de notas del alumno, y calcule su nota de prácticas como la media de esos 4 valores. Mostrar el resultado por pantalla.
5. Se dispone de un archivo "datos.txt" (en PLATEA) con datos sobre alumnos (un alumno por línea). Definir los métodos necesarios para leer el contenido de dicho archivo, y volcar en un nuevo archivo "impares.txt" una lista con los alumnos cuyo número de DNI sea impar.

2. Normativa de entrega

- Se creará un documento (en formato pdf) que incluirá una descripción breve, en lenguaje natural, sobre cómo se ha resuelto cada uno de los ejercicios planteados, sin incluir código fuente. Extensión máxima recomendada: 3 páginas.
- Solo se admitirá el formato **PDF**. No se corregirán guiones en cualquier otro formato diferente al indicado.
- El documento incluirá una portada con:
  - Identificación de los dos alumnos (Nombre, apellidos y DNI).
  - Identificación del guión.
- El documento incluirá un índice con las secciones y subsecciones del documento.
- El nombre del fichero tendrá el siguiente formato "*Ape11-Ape12-Ape21-Ape22-GuionX.pdf*" donde
  - Ape11 es el primer apellido del primer alumno.
  - Ape12 es el segundo apellido del primer alumno.
  - Ape21 es el primer apellido del segundo alumno.
  - Ape22 es el segundo apellido del segundo alumno.
  - X es el número del guión de prácticas.
- Se generará un fichero comprimido (en formato zip) que incluya el documento y el código asociado a ambas actividades.

3. Entrega y evaluación

La puntuación máxima de la práctica será de 1 punto. La entrega se llevará a cabo a través de la actividad correspondiente en PLATEA. El plazo de entrega termina el 13 de febrero de 2025 a las 23:59 horas.