Introducción a la Programación Orientada a Objeto con Python

Guillermo Palma

Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información

CI-2692 Laboratorio de Algoritmos II

Outline

Objetivo

Se quiere introducir algunos de los principios básicos de la Programación Orientada a Objetos con Python, para poder crear nuevos tipos de datos, específicamente para crear colas, pilas, árboles, tablas de hash, colas de prioridad, entre otras estructuras que se verán durante el curso.

Definición de la POO

De Wikipedia tenemos la siguiente definición [1]:

> *La programación orientada a objetos, es un paradigma de programación basado en el concepto de "objetos", los cuales son estructuras de datos que contienen data, en forma de campos conocidos como **atributos**; y código en forma de procedimientos, conocidos como **métodos**. Una característica de los objetos es que los procedimientos de un objeto, pueden acceder y modificar los campos de datos de los objetos con los cuales ellos están asociados.*

[1] https://en.wikipedia.org/wiki/Object-oriented_programming

Terminología de la POO

Clase: Mecanismo para definir un nuevo tipo de datos, que contiene un conjunto de datos (**atributos**) y procedimientos (**métodos**).

Instancia: Un objeto de cierta clase. Ej. un objeto *p* de la clase Persona es una instancia de la clase Persona.

Variable de clase: Variables que comparten todas las instancias de una clase.

Variable de instancia: Variables que contiene datos de los objetos (atributos) y están asociadas a la instancia actual de la clase.

Método: función o procedimiento definido en una clase.

Objeto : Una instancia única de una estructura de datos que está definida por su clase.

Ejemplo: Clase **Persona**

Clase: Se define el tipo de datos **Persona**.

Instancias: *Fabian* y *Diana* son instancias de la clase **Persona**.

Variables de clase: Número de instancias de tipo persona creadas, en este caso el valor es 2.

Variable de instancia o atributos:

- Nombre
- Cédula de identidad
- Fecha de Nacimiento
- Trabajos
- Sueldo

Métodos:

- Obtener nombre
- Cambiar nombre
- Obtener atributo *X* (C.I., Fecha Nacimiento, etc)
- Cambiar atributo *Y* (C.I., Fecha Nacimiento, etc)
- Calcular edad
- Dar aumento de sueldo

Sobre la POO en Python

- Python soporta múltiples paradigmas de programación, incluyendo la programación orientada a objetos.
- En Python todos los tipos de datos que son parte del lenguage, como números, *strings*, listas, diccionarios, tuplas, archivos, conjuntos, entre otros, son objetos.
- Los principios de la programación orientada a objetos están implementados de manera clara y sencilla en Python.

Implementación en Python de la clase **Persona**

- La declaración *class* se usa para crear una nueva clase. Creamos una *clase* **Persona**:

```
1 class Persona(object):
2     "Clase que representa al tipo persona"
```

- Se agrega una *variable de clase* para contar todas las personas creadas, es decir, contar todas las instancias de tipo **Persona** que se están ejecutando:

```
1 class Persona(object):
2     "Clase que representa al tipo persona"
3     contador = 0
```

Implementación en Python de la clase **Persona** cont.

- El primer método es *__init__* que es llamado constructor porque es el primer método que se ejecuta cuando se crea una instancia de **Persona**.
- El parámetro *self* es una referencia al objeto **Persona** que se esta creando, para poder tener acceso a sus atributos.
- Se muestran los *atributos* con que cuenta la clase **Persona**.
- Se aumenta el contador de instancias de clase **Persona** en 1

```
1 def __init__(self, nombre, ci, fechaNac, trabajos=None,
      sueldo=0.0):
2         self.nombre = nombre
3         self.ci = ci
4         self.fechaNac = fechaNac
5         self.trabajos = trabajos
6         self.sueldo = sueldo
7         Persona.contador += 1
```

Implementación en Python de la clase **Persona** cont.

- *Método* para aumentar el sueldo en 10 %

```
1 def dar_aumento_sueldo(self):
2     self.sueldo += self.sueldo * 0.1
```

- *Método* para obtener la edad de la persona

```
1 def obtener_edad(self):
2     return relativedelta(date.today(), self.fechaNac).years
```

- *Método* para agregar un nuevo trabajo

```
1 def agregar_trabajo(self, nuevo):
2     if self.trabajos: self.trabajos.append(nuevo)
3     else: self.trabajos = [nuevo]
```

Implementación en Python de la clase **Persona** cont.

Método para mostrar los atributos de un objeto de tipo **Persona**

```
def mostrar_persona(self):
    print("\nNombre: "+self.nombre)
    print("C.I.: "+self.ci)
    print("Fecha Nacimiento: {:%d, %b %Y}".format(self.fechaNac))
    print("Trabajos: "+ ", ".join(self.trabajos) if self.trabajos
          else "Desempleado")
    print("Sueldo: "+str(self.sueldo))
```

```
1  class Persona(object):
2      "Clase que representa al tipo persona"
3      contador = 0
4      def __init__(self, nombre, ci, fechaNac, trabajos=None,
       sueldo=0.0):
5          self.nombre = nombre
6          self.ci = ci
7          self.fechaNac = fechaNac
8          self.trabajos = trabajos
9          self.sueldo = sueldo
10         Persona.contador += 1
11
12     def dar_aumento_sueldo(self):
13         self.sueldo += self.sueldo * 0.1
14
15     def obtener_edad(self):
16        return relativedelta(date.today(), self.fechaNac).years
17
18     def agregar_trabajo(self, nuevo):
19        if self.trabajos: self.trabajos.append(nuevo)
20        else: self.trabajos = [nuevo]
```

Implementación en Python de la clase **Persona** cont.

Para acceder a los atributos se puede hacer uso de las siguientes funciones:

- **getattr(obj, name[, default])**: retorna el valor del atributo *name* del objeto *obj*.
- **setattr(obj,name,value)**: asigna el valor *value* al atributo *name* del objeto *obj*.
- **hasattr(obj,name)**: chequea si el atributo *name* forma parte del objeto *obj*.

Prueba de la clase **Persona**

Se crean dos instancias de la clase **Persona**, estos son los objetos *diana* y *fabian*.

```
1 from datetime import date
2 from persona import Persona
3
4 if __name__ == '__main__':
5   fabian = Persona("Fabian Sun", "25345654", date(1990, 12, 18)
      , ["Economista", "Consultor"], 3500)
6
7   diana = Persona("Diana He", "89345654", date(1989, 06, 29))
```

Prueba de la clase **Persona** cont.

Se muestran los atributos de los objetos, así como el número de instancias creadas.

```
1     fabian.mostrar_persona()
2     diana.mostrar_persona()
3     print("Total personas: "+str(diana.contador))
```

Prueba de la clase **Persona** cont.

- Se obtiene el valor del atributo *nombre* del objeto *diana* y se calcula su edad.

```
print("{0} tiene la edad de {1:d}".format(getattr(diana, "
    nombre"), diana.obtener_edad()))
```

- Se modifica la cédula de la **Persona** *diana*.

```
setattr(diana, "ci", "29345654")
```

- A la **Persona** *diana* se le agrega un nuevo trabajo.

```
diana.agregar_trabajo("Mototaxista")
```

- Se le asigna un sueldo la **Persona** *diana* y luego se le aumenta el sueldo.

```
setattr(diana, "sueldo", 1000)
diana.dar_aumento_sueldo()
```

Prueba de la clase **Persona** cont.

```
1  from datetime import date
2  from persona import Persona
3
4  if __name__ == '__main__':
5    fabian = Persona("Fabian Sun", "25345654", date(1990, 12, 18)
       , ["Economista", "Consultor"], 3500)
6    diana = Persona("Diana He", "89345654", date(1989, 06, 29))
7    fabian.mostrar_persona()
8    diana.mostrar_persona()
9    print("Total personas: "+str(diana.contador))
10   print("{0} tiene la edad de {1:d}".format(getattr(diana, "
       nombre"), diana.obtener_edad()))
11   setattr(diana, "ci", "29345654")
12   diana.agregar_trabajo("Mototaxista")
13   setattr(diana, "sueldo", 1000)
14   diana.dar_aumento_sueldo()
15   diana.mostrar_persona()
```

Resumen

- Se dio la definición de la POO junto con algunos de sus principales elementos.
- Se realizó la implementación de una clase en Python.
- Se mostró como instanciar un objeto de una clase.
- Se ilustró como acceder a los atributos de un objeto y como aplicar sus métodos.
- Se aplicó la POO en Python para crea un nuevo tipo de datos llamado *Persona*.