Auxiliar 12

Objetos y Clases

Profesor: Patricio Inostroza
Auxiliares: Valeria V. Franciscangeli, Pablo Skewes, Gustavo Rivera

1. Resumen

Clases: Una clase es un 'molde' que nos permite crear objetos similares. Parecido a lo que hace una estructura.

Objeto: Un objeto es una 'instancia' de una clase. Se guarda en una variable y sigue el patrón dado por la clase.

Para crear una clase, usamos la palabra clave 'class'.

Ejemplo:

```
class Fraccion:
    #Aquí va nuestro código
```

Y si queremos crear un objeto de esta clase hacemos lo siguiente:

```
unaFraccion = Fraccion(1, 2) #Nuestra fracción recibe un numerador y un denominador
```

Para que se cree correctamente el objeto, necesitamos un **constructor**, que es un método que se ejecuta automáticamente al hacer la llamada anterior. Este nos permite también almacenar variables que serán propias de cada objeto, llamadas **variables de instancia**.

```
class Fraccion:

    #Creamos el constructor. SIEMPRE se define como ___init___
    def ___init___(self, num, den):
        #Aquí trabajamos con nuestras variables de instancia, que en este caso son 2, num y den.
```

Noten el parámetro **self**. Este parámetro es una referencia a la instancia particular de la clase, y permite que dentro de la clase podamos diferenciar entre variables de instancia y otras que utilicemos.

Para almacenar una variable de instancia hacemos lo siguiente:

```
class Fraccion:

    #Creamos el constructor. SIEMPRE se define como ___init___
    def ___init___(self, num, den):
        #Aquí trabajamos con nuestras variables de instancia.
        self.numerador = num #variable de instancia numerador toma el valor del parámetro num
        self.denominador = den #variable de instancia denominador, toma el valor del parámetro den
```

También podemos crear funciones que sean propias de la clase, que se llamarán métodos. Los métodos también deben llevar el parámetro **self**, aparte de cualquier otro método que se necesite.

```
class Fraccion:

    #Creamos el constructor. SIEMPRE se define como ___init___
    def ___init___(self, num, den):
        #Aquí trabajamos con nuestras variables de instancia, que en este caso son 2, num y den.
        self.numerador = num #variable de instancia numerador toma el valor del parámetro num
        self.denominador = den #variable de instancia denominador, toma el valor del parámetro den

    #Creemos un método sume una fracción con otra.
    def sumar(self, f):
        assert type(f) == Fraccion
        num = self.numerador*f.denominador + self.denominador*f.numerador
        den = self.denominador*f.denominador
        return Fraccion(num, den)
```

Para llamar a los métodos, tenemos que haber instanciado un Objeto de la Clase. Luego, podemos llamar a los métodos usando el formato nombreObjeto.método(parámetros)

```
f1 = Fraccion(1,2) # 1/2
f2 = Fraccion(2,3) # 2/3

f3 = f1.sumar(f2) #Noten que el parámetro self no se llama.

#f3 resultará en una fraccion de numerador 7 y denominador 6.
```

1.1. Métodos mágicos

Los **Métodos Mágicos** *(Magic Methods)* son métodos especiales de una clase que se definen escribiendo doble guión bajo (___) antes y después de su nombre.

Estos métodos se diferencian del resto ya que no se invocan directamente, es decir, no se utilizan como objeto.___método___(), sino que se utilizan los operadores conocidos (+, -, *, **, /, %, etc.) para ello. Por ejemplo, si queremos utilizar el método ___add___(), solo debemos aplicar + entre dos objetos de la misma clase.

A continuación recordamos la tabla vista en clase de cátedra que resume algunos Métodos Mágicos útiles.

Operador	Expresión	Método
Menor que	p1 < p2	p1.__lt__(p2)
Menor o igual que	p1 <= p2	p1.__le__(p2)
Igual que	p1 == p2	p1.__eq__(p2)
Distinto	p1 != p2	p1.__ne__(p2)
Mayor que	p1 > p2	p1.__gt__(p2)
Mayor o igual que	p1 >= p2	p1.__ge__(p2)

Figura 1: Tabla de resumen Métodos Mágicos

Operador	Expresión	Método
Suma	p1 + p2	p1.__add__(p2)
Resta	p1 - p2	p1.__sub__(p2)
Multiplicación	p1 * p2	p1.__mul__(p2)
Potencia	p1 ** p2	p1.__pow__(p2)
División	p1 / p2	p1.__div__(p2)
Módulo	p1 % p2	p1.__mod__(p2)

Figura 2: Tabla de resumen Métodos Mágicos

2. Ejercicios

Problema 1 - Matrices

Aplicaremos nuestro conocimiento sobre Clases para crear una Clase *Matriz*. Para instanciar un objeto Matriz, se debe entregar una lista de listas de Python que representarán a la matriz. Se le pide que la Clase Matriz tenga los campos **M:** lista de listas que contiene los valores, **n_filas:** numero de filas de la matriz y **n_col:** numero de columnas de la matriz. Luego implemente los siguientes Métodos Mágicos:

- **___str___** : Representación de tipo string de la matriz.
- **___eq___**: Revisa si 2 matrices son iguales
- **___add___**: Suma de 2 matrices.
- **___mul___**: Define la multiplicación entre 2 matrices (propuesto: incluir multiplicación entre matriz y escalar).
- **___ne___:** Revisa si 2 matrices son distintas.
- **___sub___:** Resta entre 2 matrices
- **___pow___:** Potencia de una matriz (elevar una matriz a un número entero).

Indicaciones:

- Verifique para cada método que las matrices a operar no tengan problemas de dimensiones: por ejemplo, para poder sumar dos matrices se necesita que ambas tengan la misma dimensión.
- Para la multiplicación entre matrices, puede servirle la siguiente proposición:
  Dadas $A = (a_{ij}) \in \mathcal{M}_{nr}(\mathbb{K})$, $B = (b_{ij}) \in \mathcal{M}_{rm}(\mathbb{K})$ (es decir, A es una matriz de $n \times r$ y B es una matriz de $r \times m$, entonces el producto $C = A \cdot B$ se define como una matriz $C \in \mathcal{M}_{nm}(\mathbb{K})$ que cumple:
  $$c_{ij} = \sum_{k=1}^{r} a_{ik} b_{kj}, \quad \forall i \in \{1, ..., n\}, \forall j \in \{1, ..., m\},$$

Problema 2 - Polinomios

Ahora, crearemos una Clase Polinomio, como vimos en clases, y aplicaremos sobre esta los métodos mágicos aprendidos. Se le pide definir:

- **___str___**: Representación de tipo string del polinomio (vista en cátedra).
- **___eq___:** Revisa si 2 polinomios son iguales.
- **___ne___:** Revisa si 2 polinomios son distintos.
- **___add___:** Suma de 2 polinomios (vista en cátedra).
- **grado:** Entrega el grado del polinomio.
- **___mul___:** Define la multiplicación entre 2 polinomios (propuesto: incluir multiplicación entre un polinomio y un escalar) .
- **___sub___:** Resta entre 2 polinomios
- **___pow___:** Potencia de un polinomio (elevar un polinomio a un número entero) .

Indicaciones:

- Incluya en el constructor, una corrección de la lista de coeficientes entregada en caso de que hayan ceros a la derecha, es decir, si la lista de coeficientes es $L = [1, 1, 0, 1, 0, 0]$, entonces el polinomio generado debe ser $1 + x + x^3$, por lo que la lista que debemos guardar en el atributo *coef* es $L = [1, 1, 0, 1]$.
- Para la multiplicación de polinomios, le puede ser util recordar que si p y q son polinomios entonces $gr(p \cdot q) = gr(p) + gr(q)$.