Facultad de Ciencias Exactas, Físicas y Naturales

Departamento de Informática

Tecnicatura Universitaria en Programación Web

Asignatura

Programación Orientada a Objetos

Tarea de Investigación

Métodos aplicados a Listas en Python – Sobrecarga de Operadores – Metodo de Ordenamiento.

Realizado por:

Federico Vera Frassinelli
Nro. Reg. E 014 – 56

Juan Marcelo Luna
LCC 21631

Año: 2023

Introducción

La sobrecarga de operadores (Overloading) es una herramienta que nos ofrece alguno de los lenguajes de programación orientados a objetos en este caso (Python), esto nos permite poder crear código con una legibilidad mayor ya que usamos los operadores nativos del lenguaje para hacer comparaciones y operaciones entre objetos que nosotros hemos creado además poder escribir menos código para desarrollar un problema y de forma muy sencilla ya que para sobrecargar un operador lo hacemos de forma análoga a la que normalmente se crea un método de una clase.
Un operador que funciona de una manera cuando se aplica a una instancia de una clase puede funcionar de manera diferente en instancias de otra clase. También podemos anular los operadores sobrecargados en las subclases. Por ejemplo, podemos hacer que los operadores de comparación funcionen de manera diferente en una superclase y sus subclases.

El método sort() permite ordenar Listas en forma Ascendente y Descendente. Este método ordena una lista "in-place", lo cual significa que la lista muta o cambia directamente en memoria sin crear copias adicionales.

Palabras Claves: *Listas, Python, Métodos, Sobrecarga, Overloading, Sort*

Bibliografía
Mark Lutz – Learning Python – Fifth edition – O'Reilly (2013)
Gaston C. Hillar – Learning Object-Oriented Programming-Packt Publishing (2015)

Sitios WEB
Python Sitio Oficial. 3. Modelo de datos 3.3.8. Emulando tipos numéricos.
URL: https://docs.python.org/es/3.10/reference/datamodel.html
Python Sitio Oficial. 6. Expresiones. 6.7. Operaciones Aritméticas binarias.
URL: https://docs.python.org/es/3.10/reference/expressions.html
Python Sitio Oficial. Tabla de Contenido. Listas. Sort.
URL: https://docs.python.org/es/3.10/library/stdtypes.html?highlight=sort#list.sort
Python Sitio Oficial. Tabla de Contenidos. How to – Ordenar.
URL: https://docs.python.org/es/3.10/howto/sorting.html?highlight=sort
Python Sitio Oficial. Tabla de Contenido. Porque List.Sort() no retorna la lista ordenada.
URL:https://docs.python.org/es/3.10/faq/design.html?highlight=sort#why-doesn-t-list-sort-return-the-sorted-list

DESARROLLO

Sobrecarga del Operadores

En realidad, la "sobrecarga de operadores" simplemente significa interceptar operaciones integradas en los métodos de una clase: Python invoca automáticamente sus métodos cuando aparecen instancias de la clase en operaciones integradas, y el valor de retorno de su método se convierte en el resultado de la operación correspondiente. Aquí hay una revisión de las ideas clave detrás de la sobrecarga:

- La sobrecarga de operadores permite que las clases intercepten las operaciones normales de Python.
- Las clases pueden sobrecargar todos los operadores de expresión de Python.
- Las clases también pueden sobrecargar las operaciones integradas, como la impresión, las llamadas a funciones, el acceso a atributos, etc.
- La sobrecarga hace que las instancias de clase actúen más como tipos incorporados.
- La sobrecarga se implementa proporcionando métodos con nombres especiales en una clase.

En otras palabras, cuando se proporcionan ciertos métodos con nombres especiales en una clase, Python los llama automáticamente cuando aparecen instancias de la clase en sus expresiones asociadas.

Su clase proporciona el comportamiento de la operación correspondiente para los objetos de instancia creados a partir de ella.

Como se sabe, los métodos de sobrecarga de operadores no son muy requeridos y, por lo general, no tienen valores predeterminados (aparte de algunos que algunas clases obtienen del objeto); si no se hace el código o se hereda uno, solo significa que su clase no admite la operación correspondiente. Sin embargo, cuando se usan, estos métodos permiten que las clases emulen las interfaces de los objetos incorporados y, por lo tanto, parezcan más consistentes.

Métodos comunes de sobrecarga de operadores

Operator	Method	Expression
+ Addition	`__add__(self, other)`	`a1 + a2`
- Subtraction	`__sub__(self, other)`	`a1 - a2`
* Multiplication	`__mul__(self, other)`	`a1 * a2`
@ Matrix Multiplication	`__matmul__(self, other)`	`a1 @ a2` (*Python 3.5*)
/ Division	`__div__(self, other)`	`a1 / a2` (*Python 2 only*)
/ Division	`__truediv__(self, other)`	`a1 / a2` (*Python 3*)
// Floor Division	`__floordiv__(self, other)`	`a1 // a2`
% Modulo/Remainder	`__mod__(self, other)`	`a1 % a2`
** Power	`__pow__(self, other[, modulo])`	`a1 ** a2`
<< Bitwise Left Shift	`__lshift__(self, other)`	`a1 << a2`
>> Bitwise Right Shift	`__rshift__(self, other)`	`a1 >> a2`
& Bitwise AND	`__and__(self, other)`	`a1 & a2`
^ Bitwise XOR	`__xor__(self, other)`	`a1 ^ a2`
| (Bitwise OR)	`__or__(self, other)`	`a1 | a2`
- Negation (Arithmetic)	`__neg__(self)`	`-a1`
+ Positive	`__pos__(self)`	`+a1`
~ Bitwise NOT	`__invert__(self)`	`~a1`
< Less than	`__lt__(self, other)`	`a1 < a2`
<= Less than or Equal to	`__le__(self, other)`	`a1 <= a2`
== Equal to	`__eq__(self, other)`	`a1 == a2`
!= Not Equal to	`__ne__(self, other)`	`a1 != a2`
> Greater than	`__gt__(self, other)`	`a1 > a2`
>= Greater than or Equal to	`__ge__(self, other)`	`a1 >= a2`
[index] Index operator	`__getitem__(self, index)`	`a1[index]`
in In operator	`__contains__(self, other)`	`a2 in a1`
(*args, ...) Calling	`__call__(self, *args, **kwargs)`	`a1(*args, **kwargs)`

Todos los métodos de sobrecarga tienen nombres que comienzan y terminan con dos guiones bajos para mantenerlos distintos de otros nombres que defina en sus clases. Las asignaciones de nombres de métodos especiales a expresiones u operaciones están predefinidas por el lenguaje Python y documentadas en su totalidad en el manual del lenguaje estándar y otros recursos de referencia.

Todos los métodos de instancia anteriores tienen la misma declaración. Python pasa la instancia especificada en el lado derecho del operador como un argumento, que generalmente se denomina como other. Así tenemos self y other como argumentos para el método de instancia.

Los métodos de sobrecarga de operadores se pueden heredar de las superclases si no se definen, como cualquier otro método. Los métodos de sobrecarga de operadores también son opcionales, si no codifique ni hereda uno, esa operación simplemente no es compatible con su clase, e intentarlo generará una excepción. Algunas operaciones integradas, como la impresión, tienen valores predeterminados (heredados de la clase de objeto implícita en Python 3.X), pero la mayoría de los integrados fallan para las instancias de clase si no está presente el método de sobrecarga del operador correspondiente.

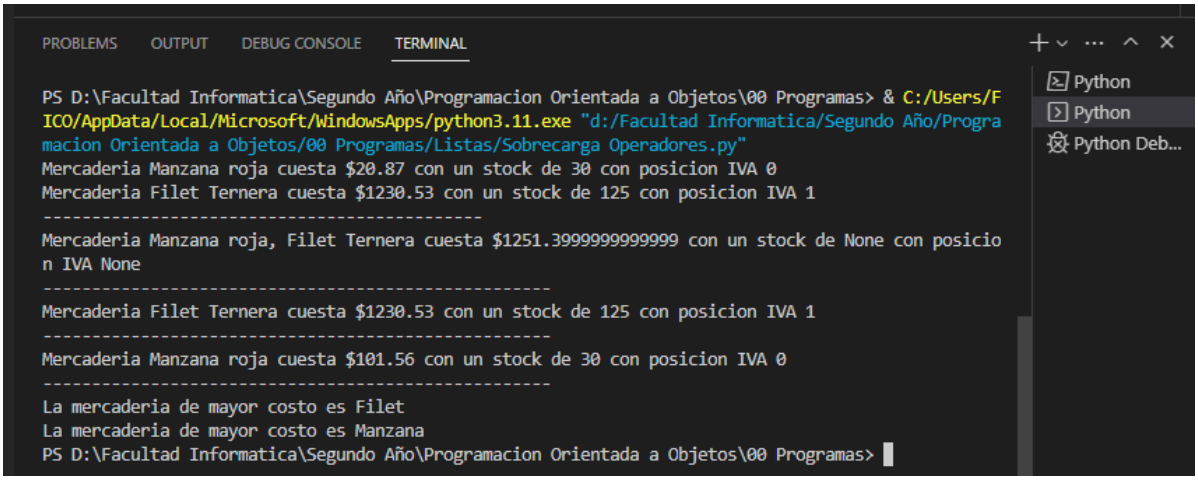
A continuación, se verán algunos de los métodos adicionales en la Tabla

```
# Se define una clase Mercaderia para poder trabajar con ella

class Mercaderia:
    pass

    def __init__(self, nom, cos, stock, iva):
        self.__nom = nom
        self.__cos = cos
        self.__stock = stock
        self.__iva = iva

# Se hace la sobrecarga de __str__ para facilitar la impresion

    def __str__(self):
```

```
        return 'Mercaderia' +self.__nom +' cuesta $' +str(self.__cos) +' con un stock de ' +str(self.__stock)+' con posicion IVA '
+str(self.__iva)



# Se realiza la sobrecarga del operador + (en donde se va a crear
# un objeto carrito que sume los nombre y los costos de 2 objetos
# de la clase mercaderia)

    def __add__(self, p):
        TempN = self.__nom +', ' + p.__nom
        TempC = self.__cos + p.__cos
        TempS = None
        TempI = None
        return Mercaderia(TempN, TempC, TempS, TempI)

# Se realiza la sobrecarga del operador O (en nuestro ejemplo, se le
# pide a este metodo que nos muestre la Mercaderia que tiene el mayor costo )

    def __or__(self, p):
        if(self.__cos > p.__cos):
            return Mercaderia(self.__nom, self.__cos, self.__stock, self.__iva)
        else:
            return Mercaderia(p.__nom, p.__cos, p.__stock, p.__iva)

# Se realiza la sobrecarga del operador incremento += (en nuestro ejemplo,
# se incrementa por medio de este metodo el costo de la Mercaderia
# en un valor flotante p)

    def __iadd__(self, p):
        self.__cos = self.__cos + float (p)
        return self

# Se realiza la sobrecarga del operador de comparacion mayor que >
# (en nuestro ejemplo se comparan los costos de las 2 mercaderias)

    def __gt__(self, p):
        if self.__cos > p.__cos:
            return True
        else:
            return False



# Se generan 2 objetos de la clase Mercaderia con sus atributos
# para probar las sobrecargas realizadas

manzana=Mercaderia('Manzana roja', 20.87, 30, 0)
filet=Mercaderia('Filet Ternera', 1230.53, 125, 1)

print(manzana)
print(filet)
print('-----------------------------------------------')



# Se genera un carrito con manzana y filet para probar
# el operador sobrecargado +

carrito = manzana + filet

print (carrito)
print('-----------------------------------------------------')



# Se genera una funcion O con manzana y filet para probar
# el operador sobrecargado O

costoso = manzana | filet
print (costoso)
print('-----------------------------------------------------')

# Se genera una funcion incremental con manzana para probar
# el operador sobrecargado +=

manzana += 80.69
print (manzana)
```

```python
print('-----------------------------------------------------')

# Se comparan las mercaderias como estan utilizando el operador
# sobrecargado > y luego se incrementa el costo de una de ellas
# para que supere a la otra utilizando el operador sobrecargado +=

if (filet>manzana):
    print('La mercaderia de mayor costo es Filet')
else:
    print('La mercaderia de mayor costo es Manzana')

manzana+=50000

if (filet>manzana):
    print ('La mercaderia de mayor costo es Filet')
else:
    print('La mercaderia de mayor costo es Manzana')
```

Ejecución del Código



¿Qué es el método sort () en Python?

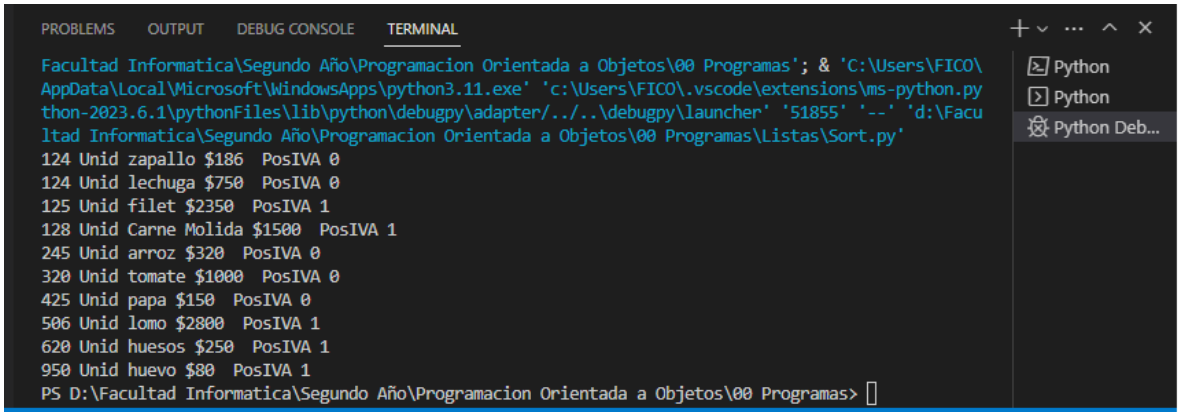
Este método toma una lista y le otorga un orden determinado. Dicho método no posee un valor de retorno. El método sort () puede tomar dos argumentos opcionales llamados key y reverse. key tiene el valor de la función que será llamada en cada ítem de la lista. Es posible utilizar la función len () cómo el valor para el argumento key. key=len indicarán al programa ordenar la lista de nombres por longitud, del más pequeño al más largo. reverse=True ordenará la lista en orden alfabético inverso.

Ejemplo del metodo sort() utilizando sobrecarga de operador < (__lt__)

```python
class Mercaderia:
    pass
    def __init__(self, nom, cos, stock, iva):
        self.__nom = nom
        self.__cos = cos
        self.__stock = stock
        self.__iva = iva
    def get_stock(self):
        return self.__stock
```

```
    def get_nombre(self):
        return self.__nom
    def get_costo(self):
        return self.__cos
    def get_iva(self):
        return self.__iva
    def __lt__(self, otra_mercaderia):
        return self.__stock < otra_mercaderia.get_stock()
mercaderias = [Mercaderia("papa", 150, 425, 0), Mercaderia("zapallo", 186, 124, 0), Mercaderia("huevo", 80, 950, 1),
           Mercaderia("huesos", 250, 620, 1), Mercaderia("tomate", 1000, 320, 0), Mercaderia("lechuga", 750, 124, 0),
           Mercaderia("Carne Molida", 1500, 128, 1), Mercaderia("arroz", 320, 245, 0), Mercaderia("lomo", 2800, 506, 1),
           Mercaderia("filet", 2350, 125, 1)]
mercaderias.sort()
# Imprimir lista de mercaderias ordenada por stock
for merc in mercaderias:
    print(f"{merc.get_stock()} Unid {merc.get_nombre()} ${merc.get_costo()}  PosIVA {merc.get_iva()}")
```

Ejecución del Código



Conclusiones

En el ámbito de la POO, la sobrecarga de métodos se refiere a la posibilidad de tener dos o más métodos con el mismo nombre, pero distinta funcionalidad. Es decir, dos o más métodos con el mismo nombre realizan acciones diferentes y el compilador usará una u otra dependiendo de los parámetros usados.

La sobrecarga de operadores quiere decir que se pueden redefinir algunos de los operadores existentes en Python para que actúen de una determinada manera, definida por el programador, con los objetos de una clase determinada.

El método sort() solo funciona con listas y ordena la propia lista seleccionada. No posee un valor de retorno, este metodo tiene los argumentos opcionales key y reverse.
key tiene el valor de una función que será llamada en cada ítem de la lista.
reverse tiene un valor booleano de True o False.