Generador de descuentos

May 27, 2020

1 Generador de descuentos

1.1 Objetivos

- Incentivar nuevas compras del cliente en el establecimiento
- Fomentar el consumo de otros productos
- Fomentar el consumo de productos con más margen de beneficio

1.2 Entradas y Salidas

- **Entrada**: Lista de artículos que ha comprado el consumidor
- **Salida**: Lista de cupones descuento que imprimir junto al recibo de compra

[1]:
```
import re

from experta import *
```

1.3 Hechos

Definiremos a continuación los hechos que manejará el sistema.

[2]:
```
class Producto(Fact):
    """
    Producto que ha comprado un cliente.

    >>> Producto(nombre="pepsi", tipo="refresco de cola", cantidad=1)

    """
    pass

class Cupon(Fact):
    """
    Cupón a generar para la próxima compra del cliente.

    >>> Cupon(tipo="2x1", producto="pepsi")
```

```
    """
    pass
```

[3]:
```
class Promo(Fact):
    """
    Promoción vigente en el comercio.

    >>> Promo(tipo="2x1", **depende_de_la_promo)

    """
    pass

class Beneficio(Fact):
    """
    Define los beneficios que obtiene el comercio por cada producto.

    >>> Beneficio(nombre="pepsi", tipo="refresco de cola", ganancias=0.2)

    """
    pass
```

1.4 Objetivo 1

1.4.1 Incentivar nuevas compras del cliente en el establecimiento

Para esto no hay nada mejor que las típicas promociones **2x1**, **3x2**, etc.

Implementación

[4]:
```
class OfertasNxM(KnowledgeEngine):
    @DefFacts()
    def carga_promociones_nxm(self):
        """
        Hechos iniciales.

        Genera las promociones vigentes
        """
        yield Promo(tipo="2x1", producto="Dodot")
        yield Promo(tipo="2x1", producto="Leche Pascual")
        yield Promo(tipo="3x2", producto="Pilas AAA")

    @Rule(Promo(tipo=MATCH.t & P(lambda t: re.match(r"\d+x\d+", t)),
                producto=MATCH.p),
          Producto(nombre=MATCH.p))
    def oferta_nxm(self, t, p):
```

```
        """
        Sabemos que el cliente volverá para aprovechar
        la promoción, ya que hoy ha comprado el producto.
        """
        self.declare(Cupon(tipo=t, producto=p))
```

Pruebas Utilizaremos la función `watch` para ver qué está haciendo el motor durante la ejecución.

[5]:
```
watch('RULES', 'FACTS')
```

[6]:
```
nxm = OfertasNxM()
```

[7]:
```
nxm.reset()
```

```
INFO:experta.watchers.FACTS: ==> <f-0>: InitialFact()
INFO:experta.watchers.FACTS: ==> <f-1>: Promo(tipo='2x1', producto='Dodot')
INFO:experta.watchers.FACTS: ==> <f-2>: Promo(tipo='2x1', producto='Leche
Pascual')
INFO:experta.watchers.FACTS: ==> <f-3>: Promo(tipo='3x2', producto='Pilas AAA')
```

[8]:
```
nxm.declare(Producto(nombre="Dodot"))
```

```
INFO:experta.watchers.FACTS: ==> <f-4>: Producto(nombre='Dodot')
```

[8]:
```
Producto(nombre='Dodot')
```

[9]:
```
nxm.declare(Producto(nombre="Agua Mineral"))
```

```
INFO:experta.watchers.FACTS: ==> <f-5>: Producto(nombre='Agua Mineral')
```

[9]:
```
Producto(nombre='Agua Mineral')
```

[10]:
```
nxm.declare(Producto(nombre="Pilas AAA"))
```

```
INFO:experta.watchers.FACTS: ==> <f-6>: Producto(nombre='Pilas AAA')
```

[10]:
```
Producto(nombre='Pilas AAA')
```

[11]:
```
nxm.run()
```

```
INFO:experta.watchers.RULES:FIRE 1 oferta_nxm: <f-3>, <f-6>
INFO:experta.watchers.FACTS: ==> <f-7>: Cupon(tipo='3x2', producto='Pilas AAA')
INFO:experta.watchers.RULES:FIRE 2 oferta_nxm: <f-4>, <f-1>
INFO:experta.watchers.FACTS: ==> <f-8>: Cupon(tipo='2x1', producto='Dodot')
```

[12]:
```
nxm.facts
```

[12]:
```
FactList([(0, InitialFact()),
          (1, Promo(tipo='2x1', producto='Dodot')),
          (2, Promo(tipo='2x1', producto='Leche Pascual')),
          (3, Promo(tipo='3x2', producto='Pilas AAA')),
          (4, Producto(nombre='Dodot')),
          (5, Producto(nombre='Agua Mineral')),
          (6, Producto(nombre='Pilas AAA')),
          (7, Cupon(tipo='3x2', producto='Pilas AAA')),
          (8, Cupon(tipo='2x1', producto='Dodot'))])
```

1.5 Objetivo 2

1.5.1 Fomentar el consumo de otros productos

Para lograr este objetivo generaremos cupones con packs descuento. Ejemplo:

- Si compras una fregona y una mopa a la vez, tienes un 25% de descuento en ambos productos

Implementación

[13]:
```
class OfertasPACK(KnowledgeEngine):
    @DefFacts()
    def carga_promociones_pack(self):
        """Genera las promociones vigentes"""
        yield Promo(tipo="PACK", producto1="Fregona ACME", producto2="Mopa␣
↪ACME", descuento="25%")
        yield Promo(tipo="PACK", producto1="Pasta Gallo", producto2="Tomate␣
↪Frito", descuento="10%")

    @Rule(Promo(tipo="PACK", producto1=MATCH.p1, producto2=MATCH.p2,␣
↪descuento=MATCH.d),
          OR(
              AND(
                  NOT(Producto(nombre=MATCH.p1)),
                  Producto(nombre=MATCH.p2)
              ),
              AND(
                  Producto(nombre=MATCH.p1),
                  NOT(Producto(nombre=MATCH.p2))
              )
          )
    )
    def pack(self, p1, p2, d):
        """
        El cliente querrá comprar un producto adicional en su próxima visita.
        """
```

```
        self.declare(Cupon(tipo="PACK", producto1=p1, producto2=p2,␣
↪descuento=d))
```

Pruebas

[14]:
```
pack = OfertasPACK()
```

[15]:
```
pack.reset()
```

```
INFO:experta.watchers.FACTS: ==> <f-0>: InitialFact()
INFO:experta.watchers.FACTS: ==> <f-1>: Promo(tipo='PACK', producto1='Fregona
ACME', producto2='Mopa ACME', descuento='25%')
INFO:experta.watchers.FACTS: ==> <f-2>: Promo(tipo='PACK', producto1='Pasta
Gallo', producto2='Tomate Frito', descuento='10%')
```

[16]:
```
pack.declare(Producto(nombre="Tomate Frito"))
```

```
INFO:experta.watchers.FACTS: ==> <f-3>: Producto(nombre='Tomate Frito')
```

[16]:
```
Producto(nombre='Tomate Frito')
```

[17]:
```
pack.declare(Producto(nombre="Fregona ACME"))
```

```
INFO:experta.watchers.FACTS: ==> <f-4>: Producto(nombre='Fregona ACME')
```

[17]:
```
Producto(nombre='Fregona ACME')
```

[18]:
```
pack.run()
```

```
INFO:experta.watchers.RULES:FIRE 1 pack: <f-1>, <f-4>
INFO:experta.watchers.FACTS: ==> <f-5>: Cupon(tipo='PACK', producto1='Fregona
ACME', producto2='Mopa ACME', descuento='25%')
INFO:experta.watchers.RULES:FIRE 2 pack: <f-2>, <f-3>
INFO:experta.watchers.FACTS: ==> <f-6>: Cupon(tipo='PACK', producto1='Pasta
Gallo', producto2='Tomate Frito', descuento='10%')
```

Si compramos ambos productos de un pack no se nos debe generar la promoción, ya que en este caso el comercio perdería beneficio.

[19]:
```
pack.reset()
```

```
INFO:experta.watchers.FACTS: ==> <f-0>: InitialFact()
INFO:experta.watchers.FACTS: ==> <f-1>: Promo(tipo='PACK', producto1='Fregona
ACME', producto2='Mopa ACME', descuento='25%')
INFO:experta.watchers.FACTS: ==> <f-2>: Promo(tipo='PACK', producto1='Pasta
Gallo', producto2='Tomate Frito', descuento='10%')
```

[20]:
```
pack.declare(Producto(nombre="Fregona ACME"))
```

```
INFO:experta.watchers.FACTS: ==> <f-3>: Producto(nombre='Fregona ACME')
```

[20]:
```
Producto(nombre='Fregona ACME')
```

[21]:
```
pack.declare(Producto(nombre="Mopa ACME"))
```

```
INFO:experta.watchers.FACTS: ==> <f-4>: Producto(nombre='Mopa ACME')
```

[21]:
```
Producto(nombre='Mopa ACME')
```

[22]:
```
pack.run()
```

1.6 Objetivo 3

1.6.1 Fomentar el consumo de productos con más margen de beneficio

El truco para cumplir este objetivo es conocer qué beneficio se obtiene por cada producto, y si existe un producto del mismo **tipo** con un beneficio mayor, generar un cupón de descuento para ese producto que nos permita seguir ganando más.

Implementación

[23]:
```
class OfertasDescuento(KnowledgeEngine):
    @DefFacts()
    def carga_beneficios(self):
        """
        Define las beneficios por producto.
        """
        yield Beneficio(nombre="Mahou", tipo="Cerveza", ganancias=0.5)
        yield Beneficio(nombre="Cerveza Hacendado", tipo="Cerveza", ganancias=0.
↪9)

        yield Beneficio(nombre="Pilas AAA Duracell", tipo="Pilas AAA",␣
↪ganancias=1.5)
        yield Beneficio(nombre="Pilas AAA Hacendado", tipo="Pilas AAA",␣
↪ganancias=2)

    @Rule(Producto(nombre=MATCH.p1),
          Beneficio(nombre=MATCH.p1, tipo=MATCH.t, ganancias=MATCH.g1),
          Beneficio(nombre=MATCH.p2, tipo=MATCH.t, ganancias=MATCH.g2),
          TEST(lambda g1, g2: g2 > g1)
    )
    def descuento_producto_con_mayor_beneficio(self, p2, g1, g2, **_):
        """
        """
        diferencia_ganancia = g2 - g1
```

```
        self.declare(Cupon(tipo="DESCUENTO",
                           producto=p2,
                           cantidad=diferencia_ganancia / 2))
```

Pruebas

[24]:
```
descuento = OfertasDescuento()
```

[25]:
```
descuento.reset()
```

```
INFO:experta.watchers.FACTS: ==> <f-0>: InitialFact()
INFO:experta.watchers.FACTS: ==> <f-1>: Beneficio(nombre='Mahou',
tipo='Cerveza', ganancias=0.5)
INFO:experta.watchers.FACTS: ==> <f-2>: Beneficio(nombre='Cerveza Hacendado',
tipo='Cerveza', ganancias=0.9)
INFO:experta.watchers.FACTS: ==> <f-3>: Beneficio(nombre='Pilas AAA Duracell',
tipo='Pilas AAA', ganancias=1.5)
INFO:experta.watchers.FACTS: ==> <f-4>: Beneficio(nombre='Pilas AAA Hacendado',
tipo='Pilas AAA', ganancias=2)
```

[26]:
```
descuento.declare(Producto(nombre="Mahou"))
```

```
INFO:experta.watchers.FACTS: ==> <f-5>: Producto(nombre='Mahou')
```

[26]:
```
Producto(nombre='Mahou')
```

[27]:
```
descuento.run()
```

```
INFO:experta.watchers.RULES:FIRE 1 descuento_producto_con_mayor_beneficio:
<f-2>, <f-5>, <f-1>
INFO:experta.watchers.FACTS: ==> <f-6>: Cupon(tipo='DESCUENTO',
producto='Cerveza Hacendado', cantidad=0.2)
```

El sistema no debe generar cupón si se ha comprado el producto con mayor beneficio

[28]:
```
descuento.reset()
```

```
INFO:experta.watchers.FACTS: ==> <f-0>: InitialFact()
INFO:experta.watchers.FACTS: ==> <f-1>: Beneficio(nombre='Mahou',
tipo='Cerveza', ganancias=0.5)
INFO:experta.watchers.FACTS: ==> <f-2>: Beneficio(nombre='Cerveza Hacendado',
tipo='Cerveza', ganancias=0.9)
INFO:experta.watchers.FACTS: ==> <f-3>: Beneficio(nombre='Pilas AAA Duracell',
tipo='Pilas AAA', ganancias=1.5)
INFO:experta.watchers.FACTS: ==> <f-4>: Beneficio(nombre='Pilas AAA Hacendado',
tipo='Pilas AAA', ganancias=2)
```

[29]:
```
descuento.declare(Producto(nombre="Pilas AAA Hacendado"))
```

```
INFO:experta.watchers.FACTS: ==> <f-5>: Producto(nombre='Pilas AAA Hacendado')
```

[29]:
```
Producto(nombre='Pilas AAA Hacendado')
```

[30]:
```
descuento.run()
```

1.7 Juntándolo todo

Gracias a **Python** podemos utilizar herencia múltiple para unir nuestros distintos motores en uno y darle un mejor interfaz de usuario.

[31]:
```
class GeneradorCupones(OfertasNxM, OfertasPACK, OfertasDescuento):
    def generar_cupones(self, *nombre_productos):
        # Reiniciamos el motor
        self.reset()

        # Declaramos los productos que ha comprado el cliente
        for nombre in nombre_productos:
            self.declare(Producto(nombre=nombre))

        # Ejecutamos el motor
        self.run()

        # Extraemos las promociones generadas
        for fact in self.facts.values():
            if isinstance(fact, Cupon):
                yield fact
```

[32]:
```
ke = GeneradorCupones()
```

[33]:
```
[cupon for cupon in ke.generar_cupones("Pilas AAA", "Mahou", "Tomate Frito")]
```

```
INFO:experta.watchers.FACTS: ==> <f-0>: InitialFact()
INFO:experta.watchers.FACTS: ==> <f-1>: Beneficio(nombre='Mahou',
tipo='Cerveza', ganancias=0.5)
INFO:experta.watchers.FACTS: ==> <f-2>: Beneficio(nombre='Cerveza Hacendado',
tipo='Cerveza', ganancias=0.9)
INFO:experta.watchers.FACTS: ==> <f-3>: Beneficio(nombre='Pilas AAA Duracell',
tipo='Pilas AAA', ganancias=1.5)
INFO:experta.watchers.FACTS: ==> <f-4>: Beneficio(nombre='Pilas AAA Hacendado',
tipo='Pilas AAA', ganancias=2)
INFO:experta.watchers.FACTS: ==> <f-5>: Promo(tipo='2x1', producto='Dodot')
INFO:experta.watchers.FACTS: ==> <f-6>: Promo(tipo='2x1', producto='Leche
Pascual')
INFO:experta.watchers.FACTS: ==> <f-7>: Promo(tipo='3x2', producto='Pilas AAA')
INFO:experta.watchers.FACTS: ==> <f-8>: Promo(tipo='PACK', producto1='Fregona
ACME', producto2='Mopa ACME', descuento='25%')
```

```
INFO:experta.watchers.FACTS: ==> <f-9>: Promo(tipo='PACK', producto1='Pasta
Gallo', producto2='Tomate Frito', descuento='10%')
INFO:experta.watchers.FACTS: ==> <f-10>: Producto(nombre='Pilas AAA')
INFO:experta.watchers.FACTS: ==> <f-11>: Producto(nombre='Mahou')
INFO:experta.watchers.FACTS: ==> <f-12>: Producto(nombre='Tomate Frito')
INFO:experta.watchers.RULES:FIRE 1 pack: <f-12>, <f-9>
INFO:experta.watchers.FACTS: ==> <f-13>: Cupon(tipo='PACK', producto1='Pasta
Gallo', producto2='Tomate Frito', descuento='10%')
INFO:experta.watchers.RULES:FIRE 2 descuento_producto_con_mayor_beneficio:
<f-2>, <f-11>, <f-1>
INFO:experta.watchers.FACTS: ==> <f-14>: Cupon(tipo='DESCUENTO',
producto='Cerveza Hacendado', cantidad=0.2)
INFO:experta.watchers.RULES:FIRE 3 oferta_nxm: <f-10>, <f-7>
INFO:experta.watchers.FACTS: ==> <f-15>: Cupon(tipo='3x2', producto='Pilas AAA')
```

[33]:
```
[Cupon(tipo='PACK', producto1='Pasta Gallo', producto2='Tomate Frito',
descuento='10%'),
 Cupon(tipo='DESCUENTO', producto='Cerveza Hacendado', cantidad=0.2),
 Cupon(tipo='3x2', producto='Pilas AAA')]
```

[ ]:
```

```