Chapter 06: Coupon Generator (Expert System)

MSc. Juan Antonio Castro Silva

May 27, 2020

1 Introduction

In this challenge, you need to develop a coupon generator system based on expert systems, using the experta framework.

1.1 Requirements

- Use the experta framework
- Get the input data from database (clients, purchases, products, packs, promo, among others)
- Create web forms to feed the system (optional)
- Create a web service in python with flask
- Store the generated coupons in a database, and add related information (client id, creation date, due date - fecha de vencimiento, among others).

1.2 Project description

Experta is a Python library for building expert systems rule based.

1.3 Version

experta 1.9.4

1.4 Installation

```
pip install experta
```

1.5 Reference

`https://pypi.org/project/experta/#description`

1.6 Source code example

`https://github.com/nilp0inter/experta`

2 Generador de descuentos

2.1 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

2.2 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
2 from experta import *
```

2.3 Hechos

Definiremos a continuación los hechos que manejará el sistema.

```
3 class Producto(Fact):
4     """
5     Producto que ha comprado un cliente.
6     >>> Producto(nombre="pepsi", tipo="refresco de cola", cantidad=1)
7     """
8     pass
9
10 class Cupon(Fact):
11     """
12     Cup n a generar para la pr xima compra del cliente.
13     >>> Cupon(tipo="2x1", producto="pepsi")
14     """
15     pass
16 class Promo(Fact):
17     """
18     Promoci n vigente en el comercio.
19     >>> Promo(tipo="2x1", **depende_de_la_promo)
20     """
21     pass
22
23 class Beneficio(Fact):
24     """
25     Define los beneficios que obtiene el comercio por cada producto.
26     >>> Beneficio(nombre="pepsi", tipo="refresco de cola", ganancias=0.2)
27     """
28     pass
```

2.4 Objetivo 1

Incentivar nuevas compras del cliente en el establecimiento.

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

2.4.1 Implementación

```
29 class OfertasNxM(KnowledgeEngine):
30     @DefFacts()
31     def carga_promociones_nxm(self):
32         """
33         Hechos iniciales.
34         Genera las promociones vigentes
35         """
36         yield Promo(tipo="2x1", producto="Dodot")
37         yield Promo(tipo="2x1", producto="Leche Pascual")
38         yield Promo(tipo="3x2", producto="Pilas AAA")
39 
40     @Rule(Promo(tipo=MATCH.t & P(lambda t: re.match(r"\d+x\d+", t)),
41                 producto=MATCH.p),
42           Producto(nombre=MATCH.p))
43     def oferta_nxm(self, t, p):
44         """
45         Sabemos que el cliente volver  para aprovechar
46         la promoci n, ya que hoy ha comprado el producto.
47         """
48         self.declare(Cupon(tipo=t, producto=p))
```

2.4.2 Pruebas

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

```
49 watch('RULES', 'FACTS')
50 nxm = OfertasNxM()
51 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')
```

```
52 nxm.declare(Producto(nombre="Dodot"))
53 nxm.declare(Producto(nombre="Agua Mineral"))
54 nxm.declare(Producto(nombre="Pilas AAA")
```

```
INFO:experta.watchers.FACTS: ==> <f-4>: Producto(nombre='Dodot')
INFO:experta.watchers.FACTS: ==> <f-5>: Producto(nombre='Agua Mineral')
INFO:experta.watchers.FACTS: ==> <f-6>: Producto(nombre='Pilas AAA')
```

```
55 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-1>, <f-4>
INFO:experta.watchers.FACTS: ==> <f-8>: Cupon(tipo='2x1', producto='Dodot')
```

```
56 nxm.facts
```

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

2.5 Objetivo 2

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

2.5.1 Implementación

```
57 class OfertasPACK(KnowledgeEngine):
58     @DefFacts()
59     def carga_promociones_pack(self):
60         """Genera las promociones vigentes"""
61         yield Promo(tipo="PACK", producto1="Fregona ACME", producto2="Mopa ACME",
               descuento="25%")
62         yield Promo(tipo="PACK", producto1="Pasta Gallo", producto2="Tomate Frito
              ", descuento="10%")
63
64     @Rule(Promo(tipo="PACK", producto1=MATCH.p1, producto2=MATCH.p2, descuento=
           MATCH.d),
65           OR(
66               AND(
67                   NOT(Producto(nombre=MATCH.p1)),
68                   Producto(nombre=MATCH.p2)
69               ),
70               AND(
71                   Producto(nombre=MATCH.p1),
72                   NOT(Producto(nombre=MATCH.p2))
73               )
74           )
75     )
76     def pack(self, p1, p2, d):
```

```
77          """
78          El cliente querr  comprar un producto adicional en su pr xima visita.
79          """
80          self.declare(Cupon(tipo="PACK", producto1=p1, producto2=p2, descuento=d))
```

2.5.2 Pruebas

```
81 pack = OfertasPACK()
82 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%')
```

```
83 pack.declare(Producto(nombre="Tomate Frito"))
84 pack.declare(Producto(nombre="Fregona ACME"))
```

```
INFO:experta.watchers.FACTS: ==> <f-3>: Producto(nombre='Tomate Frito')
INFO:experta.watchers.FACTS: ==> <f-4>: Producto(nombre='Fregona ACME')
```

```
85 pack.run()
```

```
INFO:experta.watchers.RULES:FIRE 1 pack: <f-4>, <f-1>
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.

```
86 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%')
```

```
87 pack.declare(Producto(nombre="Fregona ACME"))
88 pack.declare(Producto(nombre="Mopa ACME"))
```

```
INFO:experta.watchers.FACTS: ==> <f-3>: Producto(nombre='Fregona ACME')
INFO:experta.watchers.FACTS: ==> <f-4>: Producto(nombre='Mopa ACME')
```

```
89 pack.run()
```

2.6 Objetivo 3

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.

2.6.1 Implementación

```
90 class OfertasDescuento(KnowledgeEngine):
91     @DefFacts()
92     def carga_beneficios(self):
93         """
94         Define las beneficios por producto.
95         """
96         yield Beneficio(nombre="Mahou", tipo="Cerveza", ganancias=0.5)
97         yield Beneficio(nombre="Cerveza Hacendado", tipo="Cerveza", ganancias=0.9
               )
98
99         yield Beneficio(nombre="Pilas AAA Duracell", tipo="Pilas AAA", ganancias=
               1.5)
100        yield Beneficio(nombre="Pilas AAA Hacendado", tipo="Pilas AAA", ganancias
               =2)
101
102    @Rule(Producto(nombre=MATCH.p1),
103          Beneficio(nombre=MATCH.p1, tipo=MATCH.t, ganancias=MATCH.g1),
104          Beneficio(nombre=MATCH.p2, tipo=MATCH.t, ganancias=MATCH.g2),
105          TEST(lambda g1, g2: g2 > g1)
106    )
107    def descuento_producto_con_mayor_beneficio(self, p2, g1, g2, **_):
108        """
109        """
110        diferencia_ganancia = g2 - g1
111        self.declare(Cupon(tipo="DESCUENTO",
112                           producto=p2,
113                           cantidad=diferencia_ganancia / 2))
```

2.6.2 Pruebas

```
114 descuento = OfertasDescuento()
115 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)
```

```
116 descuento.declare(Producto(nombre="Mahou"))
```

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

```
117 descuento.run()
```

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

```
118 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)
```

```
119 descuento.declare(Producto(nombre="Pilas AAA Hacendado"))
```

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

```
120 descuento.run()
```

2.7 Juntándolo todo

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

```
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
```

```
ke = GeneradorCupones()
cupones = [cupon for cupon in ke.generar_cupones("Pilas AAA", "Mahou", "Tomate Frito")]
print("cupones:", cupones)
```

```
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-1>, <f-11>
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')
```

```
cupones: [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')]
```

Download Descuentos.ipynb