Universidad Nacional de General Sarmiento

Licenciatura en Sistemas

Trabajos Práctico: "Peguele al precio"

Introducción a la Programación

Segundo semestre, 2023

Resumen: Realizar un programa sobre un juego en el que el usuario deba adivinar el precio de ciertos productos en 60 segundos.

Integrantes: Tomás Menegol tomas.menegol@gmail.com

Leonardo Mendoza leonardoemendoza1@gmail.com

Universidad Nacional de General Sarmiento

1. Introducción (Resumen explicando de qué se trata el trabajo enfocándose en el problema que se intenta resolver de ser necesario se pueden incluir dibujos o capturas de pantalla).

1. El trabajo se trata de generar 6 funciones provistas en el archivo funcionesVACIAS.py que después se integran en el archivo PRINCIPAL.py del juego para hacerlo funcionar. Estas funciones involucran trabajar y manipular los datos de la lista de productos que conforman el juego.

2. Desarrollo (Esta sección está destinada a explicar cómo se resolvió el problema presentado en la introducción, aquí se deben incluir las principales dificultades encontradas, las soluciones a ellas y las decisiones tomadas)

2.1 Descripción general: (Resumen general de la solución completa).

2.1 Las funciones se fueron armando teniendo en cuenta primero que nada, que es lo que debían retornar. Posterior a eso, si nos pedian productos aleatorios sabiamos que ibamos a tener que usar funciones como "random.randint", y si nos pedían tener en cuenta el "margen" sabíamos que iba a involucrar un "If" que comparara la diferencia de precios absolutos contra este margen. El resto de la resolución, fue solo estructurar donde usar estas funciones o evaluar si se requieren ciclos.

2.2 Funcionalidades principales: (Breve descripción de qué se trata y cómo se resolvió la funcionalidad. En las siguientes secciones detallar las funciones implementadas incluyendo por cada función: La idea general de la función, por qué se hizo, el código comentado (utilizando tipografía adecuada) y una descripción de los parámetros que toma y los valores que devuelve/modifica)

2.2 Lectura() no toma parámetros, solo abre un archivo .txt con productos y retorna una lista de listas (con productos organizados en "nombre, precio económico, precio premium"). Esto se resolvió abriendo el archivo, convirtiendo sus contenidos a string, y después iterando sobre este string guardando sus caracteres en una variable "char". Cuando se encuentra una coma, los caracteres acumulados se añaden a una lista correspondiente al producto individual, y se borra "char" para empezar a guardar la siguiente tanda de caracteres. Cuando se encuentra con un salto de línea, se convierten los precios de string a entero, y se guarda la lista del producto individual con todos los demás productos en otra lista. Posterior a esto se reinicia la lista del producto individual para almacenar al siguiente. Al final de todas las iteraciones, se retorna está lista de productos.

```
def lectura():
    with open("productos.txt") as productosTxt:
        productosStr = productosTxt.read() #Lee el Txt con productos y lo vuelve string

    producto = []
    char = ""
    listaProductos = []

    for i in productosStr:
        if i != "\n" and i != ",": #Guarda temporalmente todo lo que haya antes de una
coma o salto de linea
            char += i
        else:
            producto.append(char) #Cuando encuentra una coma o un salto de linea, guarda
```

```
las letras o cifras que recorrio hasta el momento en una lista del producto individual
            char = ""
            if i == "\n": #Si es un salto de linea, guarda esa lista del producto en una lista de todos los productos, y continua buscando mas
                producto[1], producto[2] = int(producto[1]), int(producto[2]) #Convierte los strings de precio de producto en numeros antes de guardarlos
                listaProductos.append(producto)
                producto = []

    return listaProductos #Retorna una lista de listas (lista de productos individuales con su nombre y precios)
```

buscarProducto(listaProd) toma como parámetro una lista de productos y retornar uno solo al azar con la estructura: "nombre, tipo (económico o premium), precio". Para esto, nos paramos en la lista de productos, sobre un índice random dado por randint entre 0 y el largo de la lista - 1 (porque las listas comienzan en 0, no en 1, lo que les resta un elemento del largo). Con este índice random de la lista, tenemos un elemento random con estructura "nombre, precio económico, precio premium". Para convertir este elemento, elegimos aleatoriamente entre 1 y 2 que precio usar (el precio económico se encuentra en el índice 1 del producto, y el premium en el índice 2). Finalmente, se modifica la lista para mostrar cuál es el precio que se eligió aleatoriamente, y se retorna el producto modificado.

```
def buscar_producto(listaProd):
    prodRandom = listaProd[random.randint(0,len(listaProd)-1)] #Busca un indice aleatorio en la lista de productos y se lo asigna a productoRandom
    premiumOEconomico = random.randint(1, 2) #Da un indice random (1 o 2) para decidir si se va a usar el precio premium (guardado en el indice 2 del producto) o economico

    if premiumOEconomico == 1 : #En base al indice anterior, se asigna al producto "economico" o "premium" dependiendo de que precio se va a usar
        prodRandomPrice = [prodRandom[0], "(economico)", prodRandom[1]]
    else :
        prodRandomPrice = [prodRandom[0], "(premium)", prodRandom[2]]
    return prodRandomPrice #Retornamos una lista con un producto aleatorio, con un precio aleatorio (entre premium o economico)
```

dameProducto(listaProd, margen) toma como parámetros una lista de productos y un margen, y retorna un producto aleatorio asegurando que por lo menos existan otros 2 en la lista con un precio dentro del margen. Para hacer esto, tomamos 3 productos aleatorios de la función anterior y comparamos si su diferencia de precios está dentro del margen. Si la diferencia de alguno de los 3 productos está dentro del margen respecto de los otros 2, retornamos ese producto. Si ninguno de todos los productos está dentro del margen con respecto a los otros, se reinicia el ciclo, invocando 3 nuevos productos aleatorios.

```
def dameProducto(listaProd, margen):
    while True: #Creamos un ciclo infinito donde generaremos 3 productos aleatoriamente hasta que encontremos el correcto
        prod1 = buscar_producto(listaProd)
        prod2 = buscar_producto(listaProd)
        prod3 = buscar_producto(listaProd)
```

```
        #Comparamos la diferencia de precios entre los 3 productos. El primero que
encontremos que tenga una diferencia de precios menor al margen (respecto de los otros 2
productos), sera el que usemos
        if abs(prod1[2] - prod2[2]) <= margen and abs(prod1[2] - prod3[2]) <= margen:
            return prod1
        if abs(prod2[2] - prod1[2]) <= margen and abs(prod2[2] - prod3[2]) <= margen:
            return prod2
        if abs(prod3[2] - prod1[2]) <= margen and abs(prod3[2] - prod2[2]) <= margen:
```

esUnPrecioValido(precio, listaProd, margen) toma como parámetros un precio, la lista de productos y un margen. Retorna true o false si hay o no 3 productos en la lista que tengan un costo (sea económico o premium) dentro del margen respecto al precio. Para hacer esto, simplemente iteramos en la lista comparando tanto el precio económico como el premium con el precio dado y sumamos cuantos encontramos. Si encontramos 3 productos que cumplen con el margen, retornamos true.

```
def esUnPrecioValido(precio,listaProd,margen):
    cont = 0
    for elem in listaProd:
        if abs(elem[1] - precio) <= margen or abs(elem[2] - precio) <= margen : #Se fija
que haya productos que tengan menos que el margen de diferencia con el precio
            cont += 1
            if cont == 3: #Si hay 3 de esos productos, retorno true
                return True
    return False
```

procesar(prodPrincipal, prodCandidato, margen) toma como argumentos un producto principal, un producto candidato y un margen. Con estos retorna un 0 (o el precio del producto candidato), dependiendo si el precio del producto candidato tiene una diferencia mayor al margen con el producto principal (o no) respectivamente. Para hacer esto solo calculamos la diferencia de precios respecto al margen y retornamos el precio o el 0 según corresponda.

```
def procesar(prodPrincipal, prodCandidato, margen):
        if  abs(prodPrincipal[2]  -  prodCandidato[2])  <=  margen:    #Busca  que  haya  una
diferencia menor al margen entre el precio del producto y el precio del candidato
            return prodCandidato[2] #Si la diferencia es menor al margen, retorna el precio
del candidato
    else: return 0 #Caso contrario, retorna 0
```

dameProductosAleatorios(producto, listaProd, margen) toma como argumentos un producto, la lista de productos, y el margen. En base a eso devuelve 6 productos (el de parámetro + 5 aleatorios) con un precio aleatorio garantizando que por lo menos 2 de esos 5 están dentro del margen respecto al producto de referencia. Para esto, elegimos 5 productos aleatorios con un precio aleatorio, y comparamos si están dentro del margen. Si encontramos por lo menos dos, retornamos toda la lista con estos productos. Si no encontramos como mínimo 2, reiniciamos el ciclo buscando 5 nuevos productos. Adicionalmente, verificamos que ninguno de los productos aleatorios que vamos a retornar sea repetido entre sí o con el producto que nos dieron de referencia con la función "sonProductosDiferentes".

```
def sonProductosDiferentes(listaDeProductosAleatorios):
    listaParaComparar = listaDeProductosAleatorios  #Tengo dos listas, una que me dieron por parametro, y una para comparar
    productos = 0
    for elem in listaDeProductosAleatorios:
        for prod in listaParaComparar:
            if  elem[0] == prod[0]:
                productos +=1   #Cada vez que yo encuentro un producto de la lista que me pasaron en la lista de comparacion, sumo 1.
    if productos == len(listaDeProductosAleatorios): return True #Si ninguno de los productos se repitio, el largo de la lista y la cantidad de productos que encontre deben ser iguales
    else: return False #Caso contrario, algún producto se repitio


def dameProductosAleatorios(producto, listaProd, margen):
    while True: #Hago un ciclo infinito en caso de no encontrar productos con un precio similar reiniciar el ciclo
        prod1 = buscar_producto(listaProd)  #Busco 5 productos aleatorios, que aleatoriamente tengan precio economico o premium
        prod2 = buscar_producto(listaProd)
        prod3 = buscar_producto(listaProd)
        prod4 = buscar_producto(listaProd)
        prod5 = buscar_producto(listaProd)
        cont = 0
        prodAleatorios = [producto, prod1, prod2, prod3, prod4, prod5]
        if sonProductosDiferentes(prodAleatorios): #Me aseguro de que ninguno de los productos aleatorios por chance sean iguales entre si, o iguales al producto que me dieron por parámetro. Si son iguales, reinicia el ciclo y agarro nuevos productos aleatorios.
            if abs(producto[2] - prod1[2]) <= margen: #Con cada producto verifico si esta dentro o no del margen.
                cont+=1
            if abs(producto[2] - prod2[2]) <= margen:
                cont+=1
            if abs(producto[2] - prod3[2]) <= margen:
                cont+=1
            if abs(producto[2] - prod4[2]) <= margen:
                cont+=1
            if abs(producto[2] - prod5[2]) <= margen:
                cont+=1

            if cont >= 2: #Si hay como minimo 2 productos dentro del margen, retorno toda la lista. Quizas haya mas de 2 productos aleatorioamente
                return [producto, prod1, prod2, prod3, prod4, prod5]
```

3. Conclusiones: (Comentarios puntuales pensados finalizada la realización del trabajo)

3. El trabajo nos pareció muy interesante como desafío para poner en práctica en un escenario real lo aprendido en clase. También nos interesó mucho aprender como funciona y se implementa pygame en un juego funcional.