

Maquina Expendedora con Python

De la rosa Vázquez Josué

PROGRAMACIÓN ORIENTADA A OBJETOS (POO)

PROYECTO MAQUINA EXPENDEDORA

Simulación de Máquina Expendedora Crane National 167

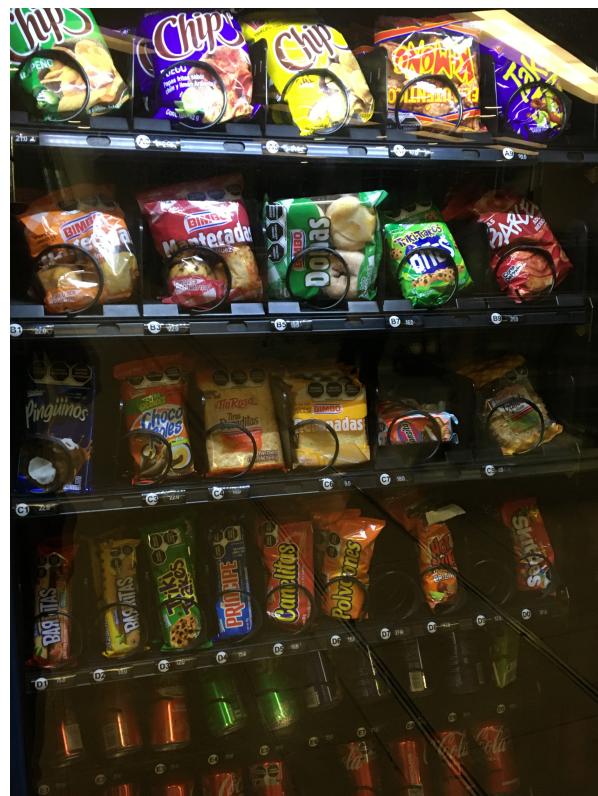
Objetivo

Desarrollar una simulación en Python que represente una **maquina expendedora** de snacks y refrescos, el modelo en el que se basa este proyecto es el, **Crane National 167**, se integraran características reales como, detección de producto agotado, calculo de cambios y manejo para opciones refrigeradas, implementando la programación orientada a objetos en Python y el uso de **clases** como **Producto** , **MaquinaExpededora** , para reflejar el funcionamiento real, así desarrollar habilidades de resolución de problemas reales en Python, como es el caso de las maquinas de venta automatizada.

Introducción

Las máquinas expendedoras han transformado nuestra forma de acceder a productos de consumo inmediato, desde productos simples como soda, frituras, productos empaquetados de fácil venta y que no puedan implicar una perdida de dinero a corto plazo, como son productos con caducidades bajas, sin embargo también en la actualidad, se han implementado diseños nuevos y muy futuristas de este tipo de maquinas, en los que se incluyen bastantes nuevas funcionalidades y variaciones también como son diferentes tipos de comida mas elaborada, maquinas de cafe, etc.

Este proyecto simula el funcionamiento del modelo **Crane National 167**, una máquina expendedora que destaca por su versatilidad y su sistema de refrigeración para snacks, golosinas y refrescos.



Evidencia sobre el modelo de máquina expendedora

Modelo: Crane National 167

- **Descripción:** Máquina estándar con diseño modular para snacks y golosinas.
- **Especificaciones principales:**
 - Capacidad: 336 artículos de golosinas, 189 snacks y 53 pasteles.
 - Dimensiones: 182.88cm x 100cm x 90cm.

- Funcionalidad: Alfanumérica para selección, bandejas ajustables, precios variables, opción de refrigeración

<https://store-d4ib26ho61.mybigcommerce.com/content/National-167-168-Programming-Guide.pdf>

Desarrollo del Proyecto

Investigación de Campo

¿Por qué hacer este proyecto?

1. **Demanda comercial:** Las máquinas expendedoras son una solución práctica para entornos como oficinas, escuelas y centros comerciales.
2. **Problemática:**
 - Muchas máquinas tienen interfaces complicadas para usuarios.
 - La gestión del inventario suele requerir intervención manual.
 - Escasez de opciones simuladas para experimentación y diseño personalizado.
3. **Objetivo práctico:** Crear un simulador que permita explorar mejoras en interfaz, gestión de inventario automatizada, y funciones como control remoto o diagnósticos.

El modelo Crane National 167

Es una máquina versátil y ampliamente usada, lo que la hace ideal para estudiar procesos de dispensación automatizada y gestión de inventarios.

Desarrollo Técnico

Clases Principales

1. Clase `Producto`

- **Atributos:** `nombre`, `precio`, `stock`, `refrigerado`.
- **Métodos:**
 - `revisar_disponibilidad()`

2. Clase `Bandeja`

- **Atributos:** `productos` (lista de objetos `Producto`), `capacidad`.
- **Métodos:**
 - `agregar_producto(producto)`
 - `retirar_producto(nombre_producto)`

3. Clase `MaquinaExpededora`

- **Atributos:** `bandejas`, `dinero_ingresado`, `cambio_disponible`.
- **Métodos:**
 - `insertar_dinero(cantidad)`
 - `seleccionar_producto(numero_bandeja, nombre_producto)`
 - `calcular_cambio()`
 - `mostrar_productos()`

Funcionalidades Clave

- **Gestión de Inventario:**
Monitorear niveles de stock por bandeja y producto.
- **Procesamiento de Pagos:**
Cálculo automático de cambio y validación de fondos suficientes.
- **Interfaz de Usuario Simulada:**
Menús interactivos que permitan al usuario elegir productos por código.

- **Diagnósticos:**

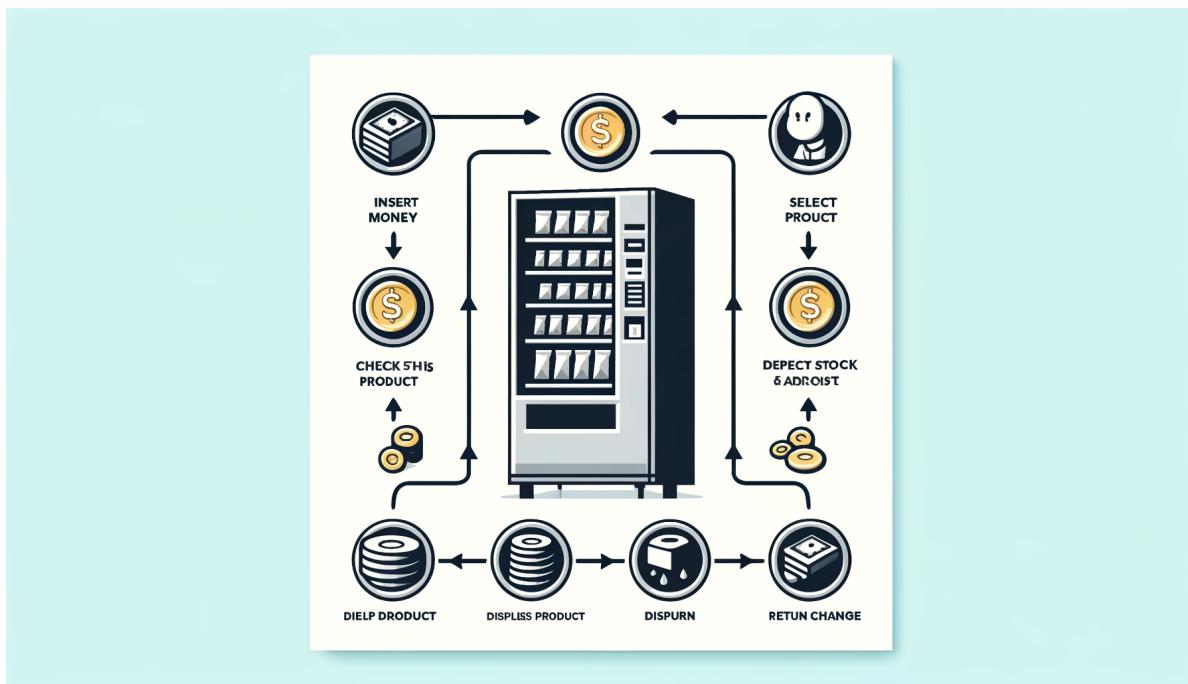
Identificar bandejas vacías y realizar mantenimiento preventivo.

Simulación de Flujo

Diagrama de Flujo del Proceso

El flujo típico para un usuario será:

1. Ingresar dinero.
2. Seleccionar producto por código (alfanumérico).
3. Validar si hay suficiente dinero y stock.
4. Dispensar el producto y devolver el cambio.



Implementación y Desarrollo

Inicio del Programa

1. Inicializar la máquina expendedora:

- Crear bandejas y productos.
- Definir el cambio disponible.

2. Definir variables principales:

- `dinero_ingresado` para registrar el dinero del usuario.
- `opcion_usuario` para manejar el flujo del menú.

Ciclo del Menú

1. Mostrar Menú Principal:

- Opciones:
 1. Mostrar productos.
 2. Insertar dinero.
 3. Seleccionar producto.
 4. Cancelar compra y devolver dinero.
 5. Salir.

2. Leer `opcion_usuario`:

- Validar entrada (debe ser un número entre 1 y 5).

Clases del Programa

1. Clase `Producto`

- **Atributos:** `nombre`, `precio`, `stock`, `refrigerado`.
- **Métodos:**
 - `revisar_disponibilidad()`

La clase `Producto` se basa en los atributos de:

- `nombre` del producto, que sea un `string` que indique simple el nombre del producto
- `precio` del producto, que sera un numero entero o decimal dependiendo del producto
- `stock` del producto, este también sera un valor numérico, entero que indica la cantidad en existencia del producto
- `refrigerado` este sera un valor booleano, `True` o `False` indicando si el producto debe ir en refrigeración o no, como es el caso de las sodas

Los métodos de la clase `Producto`:

- `revisar_disponibilidad()` este método revisa la disponibilidad del stock del producto, para saber si se puede vender o no.

```
class Producto:  
    def __init__(self, nombre, precio, stock, refrigerado=False):  
        self.nombre = nombre  
        self.precio = precio  
        self.stock = stock  
        self.refrigerado = refrigerado  
  
    def revisar_disponibilidad(self):  
        if self.refrigerado:  
            print(f"{self.nombre} - ${self.precio} (Disponibles)  
        else:  
            print(f"{self.nombre} - ${self.precio} (Disponibles)  
  
sabritas = Producto("sabritas", 22.50, 4)  
cocacola = Producto("coca cola", 23, 5, True)  
doritos = Producto("doritos", 23.14, 0)  
  
sabritas.revisar_disponibilidad() #sabritas - $22.5 (Disponibles)
```

```
cocacola.revisar_disponibilidad() #coca cola - $23 (Disponibles  
doritos.revisar_disponibilidad() #doritos - $23.14 (Disponibles
```

2. Clase `Bandeja`

- **Atributos:** `productos` (lista de objetos `Producto`), `capacidad`, `refrigerada`.
- **Métodos:**
 - `agregar_producto(producto)`
 - `seleccionar_producto(nombre)`

En la clase `Bandeja` estará basada en recibir los atributos de:

- `capacidad`, valor numérico que indica la capacidad de la bandeja
- `refrigerada`, si la bandeja debe tener refrigeración o no, valor booleano
- `productos`, como un array de objetos.

Los métodos de la clase `Bandeja` :

- `agregar_producto(producto)` , este método se basa en agregar los productos recibiendo el producto a agregar, agregándolos en un array que simula la bandeja.
- `retirar_producto(nombre_producto)` , este método simula el retiro de un producto del stock de la bandeja.

```
class Bandeja:  
    def __init__(self, capacidad, refrigerada=False):  
        self.capacidad = capacidad  
        self.refrigerada = refrigerada  
        self.productos = [] #Array de productos en la bandeja  
  
    def agregar_producto(self, producto):  
        # Para agregar productos  
        pass
```

```
def retirar_producto(self, nombre_producto):  
    # Para retirar un producto  
    pass
```

Bibliografía

Downey, A. (2015).

Think Python: How to Think Like a Computer Scientist (2nd ed., Version 2.4.0).
Green Tea Press.

Disponible en: <http://facweb.cs.depaul.edu/sjost/it211/documents/think-python-2nd.pdf>

Sweigart, A. (2020).

Beyond the Basic Stuff with Python: Best Practices for Writing Clean Code.
No Starch Press.

Disponible en: <https://nostarch.com/beyond-basic-stuff-python>

