Desafios atuais sobre tudo o que aprendemos até agora em Python.

Hoje, vamos praticar e aprimorar nossos conhecimentos sobre em Python!

Utilizado o que aprendemos nas aulas anteriores:

Controle de Estoque e Vendas de uma Empresa de Produtos:

Uma **empresa de vendas de eletrônicos** deseja controlar o estoque de seus produtos, registrar as vendas realizadas e calcular o valor total das vendas.

O sistema deve ser capaz de armazenar as informações sobre o nome do produto, o preço unitário, a quantidade em estoque e a quantidade de unidades vendidas.

Controle de Estoque e Vendas de uma Empresa de Produtos:

Os requisitos já foi feito pelos desenvolvedores e o gerente de projeto no qual foi identificado que irá utilizar nos próximos Slides:

Variáveis Iniciais:

O programa começa com a definição de quatro listas vazias para armazenar informações essenciais.

A lista produtos será utilizada para armazenar os nomes dos produtos.

A lista precos armazenará os preços unitários de cada produto.

A lista estoques guardará as quantidades de cada item disponíveis em estoque, enquanto a lista vendas manterá a quantidade de unidades vendidas de cada produto.

Estas listas serão fundamentais para o gerenciamento do estoque e das vendas.

Laço while para Inserção de Dados:

O programa utilizará um laço while True para continuamente pedir ao usuário informações sobre os produtos. O laço permitirá que o usuário insira o nome do produto, o preço unitário, a quantidade em estoque e as unidades vendidas.

Caso o nome do produto seja deixado em branco (pressionando Enter sem digitar nada), o programa sairá do laço e calculará os resultados finais.

Se o produto já estiver na lista, o programa irá atualizar a quantidade em estoque e as vendas. Se for um produto novo, o programa irá adicionar o nome do produto, o preço unitário, a quantidade em estoque e as unidades vendidas às respectivas listas. Este processo garante que o estoque e as vendas sejam atualizados conforme o usuário for inserindo os dados dos produtos.

Cálculos:

O programa deve realizar alguns cálculos importantes para fornecer as informações ao usuário.

Primeiro, ele calculará o estoque total, que é a soma das quantidades de todos os produtos em estoque.

Em seguida, o programa calculará o valor total das vendas, multiplicando a quantidade de unidades vendidas de cada produto pelo seu preço unitário. O valor total das vendas será exibido ao final, proporcionando uma visão clara do faturamento.

Além disso, o programa identificará o produto com a maior demanda, ou seja, aquele que teve a maior quantidade de unidades vendidas, facilitando a análise do desempenho dos produtos.

Finais:

Após os cálculos, o programa exibirá o estoque atual de cada produto, mostrando o nome do produto e a quantidade em estoque.

Também será exibido o valor total de vendas de cada produto, fornecendo uma visão detalhada de quanto cada item gerou em faturamento.

Por fim, o programa mostrará o nome do produto mais vendido, juntamente com o total de unidades que foram comercializadas.

Essas informações são essenciais para que a empresa tenha um controle efetivo sobre o estoque e as vendas, possibilitando decisões de reposição e estratégias de marketing.

```python
# Variáveis iniciais
produtos = []  # Lista para armazenar os nomes dos produtos
precos = []    # Lista para armazenar os preços dos produtos
estoques = []  # Lista para armazenar a quantidade em estoque de cada produto
vendas = []    # Lista para armazenar a quantidade vendida de cada produto

# Boas-Vindas
print("Bem-vindo ao sistema de controle de estoque e vendas.")
print("Você pode adicionar produtos, inserir preços, quantidades em estoque e quantidades vendidas.")
print("Para sair, basta deixar o nome do produto em branco e precionar Enter.")

# Loop para coletar os dados dos produtos
while True:
    # Solicita os dados do produto
    nome_produto = input('\n'"Digite o nome do produto/item (ou pressione Enter para sair do programa): ")

    # Se o usuário deixar o nome do produto em branco, o loop é interrompido
    if nome_produto == "":
        break  # Sai do loop se a entrada estiver vazia

    # Solicita o preço do produto
    try:
        preco_produto = float(input(f"Qual o preço Unitário? {nome_produto}: R$ "))
    except ValueError:
        # Se o valor informado não for um número válido, exibe uma mensagem de erro
        print("Entrada inválida! O preço deve ser um número.")
        continue

    # Solicita a quantidade em estoque
    try:
        estoque_produto = int(input(f"Digite a quantidade em estoque do {nome_produto}: "))
    except ValueError:
        # Se o valor informado não for um número inteiro válido, exibe uma mensagem de erro
        print("Entrada inválida! A quantidade deve ser um número inteiro.")
        continue

    # Solicita a quantidade vendida
    try:
        quantidade_vendida = int(input(f"Digite a quantidade vendida do {nome_produto}: "))
    except ValueError:
        # Se o valor informado não for um número inteiro válido, exibe uma mensagem de erro
        print("Entrada inválida! A quantidade vendida deve ser um número inteiro.")
        continue
```

```python
    # Verifica se o produto já existe na lista de produtos
    if nome_produto in produtos:
        # Se o produto já estiver na lista, atualiza os valores
        index = produtos.index(nome_produto)
        estoques[index] += estoque_produto
        vendas[index] += quantidade_vendida
    else:
        # Se for um produto novo, adiciona os dados às listas
        produtos.append(nome_produto)
        precos.append(preco_produto)
        estoques.append(estoque_produto)
        vendas.append(quantidade_vendida)

# Cálculos
valor_total_vendas = 0
produto_mais_vendido = None
maior_quantidade_vendida = 0

# Exibe o estoque e o valor total das vendas para cada produto
for i in range(len(produtos)):
    # Cálculo do valor total de vendas
    valor_vendas_produto = vendas[i] * precos[i] # Multiplica a quantidade vendida pelo preço unitário
    valor_total_vendas += valor_vendas_produto # Acumula o valor total das vendas

    # Verifica qual produto tem a maior demanda
    if vendas[i] > maior_quantidade_vendida:
        maior_quantidade_vendida = vendas[i] # Atualiza a maior quantidade vendida
        produto_mais_vendido = produtos[i] # Atualiza o nome do produto mais vendido

    # Exibe o estoque e o valor das vendas do produto
    print(f"\nProduto: {produtos[i]}")
    print(f"Estoque: {estoques[i]}") # Exibe a quantidade em estoque
    print(f"Quantidade vendida: {vendas[i]}") # Exibe a quantidade vendida
    print(f"Valor total das vendas: R$ {valor_vendas_produto:.2f}") # Exibe o valor total das vendas do produto

# Exibe o valor total de vendas e o produto mais vendido
print(f"\nValor total de vendas de todos os produtos: R$ {valor_total_vendas:.2f}")
print(f"O produto mais vendido foi: {produto_mais_vendido}, com {maior_quantidade_vendida} unidades vendidas.")

# Finalização
print('\n'"Obrigado por utilizar nosso programa! Em breve iremos ter interface.")
```

IMPORTANTE:

Obtendo um código profissional e com maior flexibilidade:

Iremos realizar um código com um algoritmo e construção avançado deste programa, somente para visualização e futuramente em outras aulas iremos dar continuidade em nossos aprendizados, como (Dicionários, tuplas, classes e funções).

Lembrete:
O código ainda não foi modularizado, com as classes e funções separadas em arquivos distintos para melhorar a organização do código. Também não contém interface e banco de dados.

```python
class Produto:
    # Classe que representa um produto com informações de nome, preço, estoque e vendas.

    def __init__(self, nome, preco, estoque, vendas):
        # Inicializa um objeto Produto com nome, preço, estoque e vendas
        self.nome = nome
        self.preco = preco
        self.estoque = estoque
        self.vendas = vendas

    def atualizar_estoque(self, quantidade):
        # Atualiza o estoque do produto
        self.estoque += quantidade

    def atualizar_vendas(self, quantidade):
        # Atualiza as vendas do produto
        self.vendas += quantidade

    def valor_total_vendas(self):
        # Calcula o valor total de vendas do produto
        return self.vendas * self.preco


class SistemaControleEstoque:
    # Classe que representa o sistema de controle de estoque e vendas.

    def __init__(self):
        # Inicializa o sistema com um dicionário vazio para armazenar os produtos e valores de vendas
        self.produtos = {}  # Dicionário para armazenar produtos, chave: nome do produto, valor: objeto Produto
        self.valor_total_vendas = 0  # Inicializa o valor total das vendas como 0
        self.produto_mais_vendido = None  # Inicializa a variável para armazenar o produto mais vendido
        self.maior_quantidade_vendida = 0  # Inicializa a quantidade vendida máxima como 0

    def adicionar_produto(self, nome, preco, estoque, vendas):
        # Adiciona ou atualiza um produto no sistema.
        if nome in self.produtos:
            # Se o produto já existe no dicionário, atualiza estoque e vendas
            produto = self.produtos[nome]
            produto.atualizar_estoque(estoque)  # Atualiza o estoque do produto
            produto.atualizar_vendas(vendas)  # Atualiza as vendas do produto
        else:
            # Se o produto não existe, cria um novo objeto Produto e adiciona ao dicionário
            produto = Produto(nome, preco, estoque, vendas)
            self.produtos[nome] = produto  # Adiciona o novo produto no dicionário

    def calcular_vendas_totais(self):
        # Calcula o valor total das vendas e determina o produto mais vendido.
        for produto in self.produtos.values():
```

```python
        for produto in self.produtos.values():
            # Soma o valor total das vendas de cada produto
            self.valor_total_vendas += produto.valor_total_vendas()

            # Verifica se a quantidade vendida desse produto é maior que a anterior
            if produto.vendas > self.maior_quantidade_vendida:
                self.maior_quantidade_vendida = produto.vendas  # Atualiza a maior quantidade vendida
                self.produto_mais_vendido = produto  # Atualiza o produto mais vendido

    def exibir_relatorio(self):
        # Exibe o relatório com as informações de todos os produtos.
        for produto in self.produtos.values():
            print(f"\nProduto: {produto.nome}")
            print(f"Estoque: {produto.estoque}")
            print(f"Quantidade vendida: {produto.vendas}")
            print(f"Valor total das vendas: R$ {produto.valor_total_vendas():.2f}")

        # Exibe o valor total das vendas e o produto mais vendido
        print(f"\nValor total de vendas de todos os produtos: R$ {self.valor_total_vendas:.2f}")
        if self.produto_mais_vendido:
            print(f"O produto mais vendido foi: {self.produto_mais_vendido.nome}, com {self.maior_quantidade_vendida} unidades vendidas.")


def boas_vindas():
    # Exibe as boas-vindas e informações iniciais do sistema.
    print("Bem-vindo ao sistema de controle de estoque e vendas.")
    print("Você pode adicionar produtos, inserir preços, quantidades em estoque e quantidades vendidas.")
    print("Para sair, basta deixar o nome do produto em branco e precionar Enter.")


def obter_entrada_produto():
    # Solicita e valida as entradas do usuário para um produto.
    nome_produto = input('\n' "Digite o nome do produto/item (ou pressione Enter para sair do programa): ")
    if nome_produto == "":
        return None  # Retorna None para indicar que o usuário quer sair

    # Solicita o preço do produto
    try:
        preco_produto = float(input(f"Qual o preço Unitário? {nome_produto}: R$ "))
    except ValueError:
        print("Entrada inválida! O preço deve ser um número.")
        return obter_entrada_produto()  # Chama recursivamente para tentar novamente

    # Solicita a quantidade em estoque
    try:
        estoque_produto = int(input(f"Digite a quantidade em estoque do {nome_produto}: "))
```

```python
def obter_entrada_produto():
        estoque_produto = int(input(f"Digite a quantidade em estoque do {nome_produto}: "))
    except ValueError:
        print("Entrada inválida! A quantidade deve ser um número inteiro.")
        return obter_entrada_produto()  # Chama recursivamente para tentar novamente

    # Solicita a quantidade vendida
    try:
        quantidade_vendida = int(input(f"Digite a quantidade vendida do {nome_produto}: "))
    except ValueError:
        print("Entrada inválida! A quantidade vendida deve ser um número inteiro.")
        return obter_entrada_produto()  # Chama recursivamente para tentar novamente

    return nome_produto, preco_produto, estoque_produto, quantidade_vendida


def main():
    # Função principal que executa o fluxo do programa.
    sistema = SistemaControleEstoque()  # Instancia o sistema de controle de estoque

    boas_vindas()

    while True:
        # Coleta os dados do produto
        dados_produto = obter_entrada_produto()
        if dados_produto is None:
            break  # Sai do loop se o nome do produto for deixado em branco

        nome_produto, preco_produto, estoque_produto, quantidade_vendida = dados_produto
        sistema.adicionar_produto(nome_produto, preco_produto, estoque_produto, quantidade_vendida)

    # Calcula as vendas totais e o produto mais vendido
    sistema.calcular_vendas_totais()

    # Exibe o relatório final
    sistema.exibir_relatorio()

    # Finalização
    print('\nObrigado por utilizar nosso programa! Em breve iremos ter interface.')


# Chama a função principal para iniciar o programa
if __name__ == "__main__":
    main()
```

Dúvidas