Relatório do bootcamp - 2

Thassiana C. A. Muller

Revisão de sintaxe e lógica de Python:

Já programei em Python então os vídeos foram úteis para reforçar os principais conceitos da linguagem. **Em anexo no Trello junto a esse documento estão todos os códigos utilizados para estudo, e também o exemplo desse pdf para execução**.

Para exemplificar os conhecimentos, exponho um exemplo comentado de controle de estoque utilizando a maioria dos conceitos expostos nos vídeos[1]

Inicialmente, é criado a classe Produto e três métodos para ela, o primeiro para inicializar um produto com as suas características e o segundo para retornar suas características cruas e o terceiro para retornar suas características de maneira mais formatada. Vale lembrar que, nesse caso, para utilizar os métodos é preciso, primeiramente, instanciar um objeto da classe.

```python
class Produto:
    def __init__(self, nome, preco, categoria): #construtor de Produto
        self.nome = nome
        self.preco = preco
        self.categoria = categoria

    def __repr__(self): # metodo para mostrar representação detalhada
        return f"nome={self.nome}, preco={self.preco}, categoria={self.categoria}"

    def __str__(self): # metodo para mostrar as caracteristicas do Produto
        return f"Nome: {self.nome}, Preco: {self.preco}, Categoria: {self.categoria}"
```

[1] PYTHON 3 Curso Rápido 🐍 Parte #1 2020 - 100% Prático! ,
PYTHON 3 Curso Rápido 🐍 Parte #2 2020 - 100% Prático!

Aqui é criada a função que irá adicionar um ou mais produtos em um inventário, ela recebe uma lista de produtos e um número variável de dicionários com as características dos produtos. Assim, ocorre uma iteração em cada dicionário para que seja criado um objeto da classe Produto e adicionado a uma lista de produtos.

```
# funcao para adicionar produtos a uma lista chamada inventario
def adicionar_produto(inventario, *args):
    for p in args:
        produto = Produto(p['nome'], p['preco'], p['categoria'])
        inventario.append(produto)
```

Essa próxima função calcula o valor total de produtos em uma lista chamada inventário, para fazer isso, a função reduce recebe 3 argumentos, o primeiro é uma lambda que irá atualizar o valor de total a cada iteração, o segundo é a lista de produtos que o reduce vai aplicar a função do primeiro argumento e o terceiro é o valor inicial da acumulação do reduce

```
# funcao para calcular a soma cumulativa de inventario utilizando reduce e a lambda
def calcular_valor_total(inventario):
    return reduce((lambda total, produto: total + produto.preco), inventario, 0)
```

Aqui é criado a função que filtra produtos de uma lista pelo seu atributo categoria.

A função filter recebe 2 argumentos, uma lambda, que retornará verdadeiro se a categoria do produto for a categoria informada, e uma lista de produtos a serem verificados. Após isso, ocorre um typecast para list e é retornado a lista de produtos da categoria informada

```
# funcao utilizando filter para filtrar os produtos por categoria
def filtrar_por_categoria(inventario, categoria):
    return list(filter(lambda produto: produto.categoria == categoria, inventario))
```

Seguindo com a criação de funções para o estoque, temos a responsável por aplicar um desconto em todos os produtos de uma lista.

A função map recebe 2 argumentos, uma lambda, que irá retornar uma instância de produto idêntica à mapeada, com exceção do preço, que será atualizado de acordo com o desconto desejado, e uma lista de produtos que serão avaliados pela lambda. Por fim, é realizado o typecast para que seja retornado corretamente uma lista de produtos

```
# funcao para mapear um lambda em uma nova lista de descontos
def aplicar_desconto(inventario, desconto):
    return list(map(lambda produto: Produto(produto.nome, produto.preco * (1 - desconto),
produto.categoria), inventario))
```

A próxima função retorna as informações dos produtos de uma lista utilizando método __str__ de cada produto de uma lista

```
# funcao para exibir um inventario
def exibir_inventario(inventario):
    return [str(produto)[2] for produto in inventario]
```

E por último, temos a função principal que irá utilizar as outras funções para testar se está tudo funcionando corretamente.

Como não quero que ela execute sempre que eu utilizar esse código em outros arquivos deixo a restrição : if __name__ == "__main__": main()

```
def main():
    inventario = []

    adicionar_produto(inventario,
                      {"nome": "Camiseta", "preco": 50.0, "categoria": "Roupas"},
                      {"nome": "Calca", "preco": 80.0, "categoria": "Roupas"},
                      {"nome": "Tenis", "preco": 120.0, "categoria": "Calcados"},
                      {"nome": "Bone", "preco": 30.0, "categoria": "Acessorios"}
    )
```

[2] str(produto) é a maneira idiomática e recomendada de converter um objeto para sua representação em string. Isso respeita o princípio de encapsulamento, escondendo os detalhes de implementação interna do método __str__ do usuário

```
    inventario_exibicao = exibir_inventario(inventario)
    print("\nInventario:")
    for item in inventario_exibicao:
        print(item)
    valor_total = calcular_valor_total(inventario)
    print(f"Valor total do inventario: R$ {valor_total:.2f}\n")

    roupas = filtrar_por_categoria(inventario, "Roupas")
    print(f"Produtos na categoria 'Roupas': {roupas}\n")

    inventario_com_desconto = aplicar_desconto(inventario, 0.1)
    print("Inventario com desconto:")
    for produto in inventario_com_desconto:
        print(produto)


if __name__ == "__main__":
    main()
```

Saída do terminal:

```
[Running] python -u "d:\Drive Google\UTFPR\Lamia\Atividade 2\exemplo.py"

Inventario:
Nome: Camiseta, Preco: 50.0, Categoria: Roupas
Nome: Calca, Preco: 80.0, Categoria: Roupas
Nome: Tenis, Preco: 120.0, Categoria: Calcados
Nome: Bone, Preco: 30.0, Categoria: Acessorios
Valor total do inventario: R$ 280.00

Produtos na categoria 'Roupas': [nome=Camiseta, preco=50.0, categoria=Roupas, nome=Calca, preco=80.0,
categoria=Roupas]

Inventario com desconto:
Nome: Camiseta, Preco: 45.0, Categoria: Roupas
Nome: Calca, Preco: 72.0, Categoria: Roupas
Nome: Tenis, Preco: 108.0, Categoria: Calcados
Nome: Bone, Preco: 27.0, Categoria: Acessorios
```