

Roteiro de Atividade Prática

Nome: Henrique Ferreira Da Silva

Turma: 2B

Atividade 1: Classe Simples – Lampada

Objetivos

Antes de resolver este exercício, é importante entender o que é uma classe e como criar objetos a partir dela. Uma classe é definida usando a palavra-chave `class`, seguida do nome da classe e dois-pontos. Dentro da classe, podemos definir métodos (funções) e atributos (dados).

Descrição do exercício:

Utilizando a linguagem de programação Python, crie uma classe `Lampada` com um atributo `estado` (ligado/desligado) e um método para alterar seu estado.

Tempo estimado: 15 minutos

Lista de materiais

- Computador com internet;
- Caderno para anotações;
- 1 caneta.

Procedimento experimental

1. Analise o exemplo-base para criação do código:

```
class Lampada:
    def __init__(self):
        self.estado = False # Começa desligada
# Uso da classe
lampada = Lampada()
```

```
print(lampada.alterar_estado()) # Liga a lâmpada  
print(lampada.alterar_estado()) # Desliga a lâmpada
```

2. Agora, crie uma definição para alterar o seu estado entre ligada e desligada de acordo com o exemplo disponibilizado.
3. Anote o código desenvolvido nas linhas seguintes e envie por meio do AVA.

```
4. class Lampada:  
5.     def __init__(self):  
6.         self.estado = False  
7.  
8.     def desligar(self):  
9.         if self.estado:  
10.            self.estado = False  
11.            print("Lâmpada desligada.")  
12.        else:  
13.            print("A lâmpada já está desligada.")  
14.  
15.     def ligar(self):  
16.         if not self.estado:  
17.             self.estado = True  
18.             print("Lâmpada ligada.")  
19.         else:  
20.             print("A lâmpada já está ligada.")  
21.  
22. def escolher_opcao(lampada):  
23.     while True:  
24.         opcao = input("Digite 1 para ligar a lâmpada, 0 para desligar ou  
                qualquer outra tecla para sair: ")  
25.         if opcao == '1':  
26.             lampada.ligar()  
27.         elif opcao == '0':  
28.             lampada.desligar()  
29.         else:  
30.             print("Saindo do programa...")  
31.             break  
32.  
33. lampada = Lampada()  
34. escolher_opcao(lampada)
```

1.

Atividade 2: Encapsulamento com Atributos Privados

Objetivos

Entenda que encapsulamento em Python é mais uma convenção do que uma restrição técnica. Atributos privados são definidos com um sublinhado duplo (`__`). Eles não são acessíveis diretamente de fora da classe, promovendo o encapsulamento.

Descrição do exercício:

Crie uma classe `Contador` que mantenha um valor interno privado e tenha métodos para incrementar, decrementar e obter esse valor.

Tempo estimado: 15 minutos

Procedimento experimental

1. Analise o exemplo base para criação do código:

```
class Contador:
    def __init__(self):
        self.__valor = 0

# Uso da classe
contador = Contador()
contador.incrementar()
contador.incrementar()
print(contador.get_valor()) # Deve mostrar 2
contador.decrementar()
print(contador.get_valor()) # Deve mostrar 1
```

2. Agora, a partir do código analisado, crie as definições para incrementar, decrementar e obter esse valor.
3. Anote o código desenvolvido nas linhas seguintes e envie por meio do AVA.

```
class Contador:
```

```
def __init__(self):
    self.__valor = 0

def incrementar(self):
    self.__valor += 1

def decrementar(self):
    self.__valor -= 1

def obter_valor(self):
    return self.__valor
def escolher_opcao(contador):
    while True:
        opcao = input("Digite 1 para incrementar, -1 para decrementar ou qualquer outra
tecla para sair: ")
        if opcao == '1':
            contador.incrementar()
            print("Contador atual:", contador.obter_valor())
        elif opcao == '-1':
            contador.decrementar()
            print("Contador atual:", contador.obter_valor())
        else:
            print("Saindo do programa...")
            break
contador=Contador()
escolher_opcao(contador)
```

Atividade 3: Uso de Getters e Setters

Objetivos

Getters e Setters são métodos usados para obter e definir o valor de atributos privados. Eles são úteis para adicionar lógica adicional durante a obtenção ou a definição de um valor, como validações.

Descrição do exercício:

Crie uma classe Termômetro que armazene a temperatura em graus Celsius, mas permita definir e obter a temperatura em Fahrenheit.

Tempo estimado: 10 minutos

Procedimento experimental

1. Analise o exemplo-base para criação do código:

```
class Termometro:
    def __init__(self, temperatura=0):
        self.__temperatura_celsius = temperatura

    def get_temperatura_fahrenheit(self):
        return (self.__temperatura_celsius * 9/5) + 32

# Uso da classe
termometro = Termometro()
termometro.set_temperatura_fahrenheit(68)
print(termometro.get_temperatura_fahrenheit()) # Deve mostrar 68
```

2. Agora, a partir do código analisado, crie as definições para que a temperatura também possa ser obtida em Celsius.
3. Anote o código desenvolvido nas linhas seguintes e envie por meio do AVA.

```
4. class Termometro:
5.     def __init__(self, temperatura=0):
6.         self.__temperatura_celsius = temperatura
7.
8.     def set_temperatura_celsius(self, temperatura):
9.         self.__temperatura_celsius = temperatura
10.
11.    def set_temperatura_fahrenheit(self, temperatura):
12.        self.__temperatura_celsius = (temperatura - 32) * 5/9
13.
14.    def get_temperatura_celsius(self):
15.        return self.__temperatura_celsius
16.
17.    def get_temperatura_fahrenheit(self):
18.        return (self.__temperatura_celsius * 9/5) + 32
19.
20. termometro = Termometro()
21. termometro.set_temperatura_fahrenheit(32)
22. print(termometro.get_temperatura_celsius())
```

