Aluno: Rodrigo de Paula Pedrosa

Turma: 2024.1

E-mail: 202311035761@alunos.estacio.br ou d2_rodrigod2@hotmail.com

Professora: Jhonatan Alves

Disciplina: Conhecendo novos paradigmas

Missão prática do 4º nível de conhecimento.

Microatividades e Códigos

Microatividades

1, 2, 3, 4, e 5

Pessoa.py

```
class Pessoa:
    def __init__(self, nome, dataNascimento, cpf, rg, status=False):
        self.__nome = nome
        self.__dataNascimento = dataNascimento
        self.__cpf = cpf
        self.__rg = rg
        self.__status = status

    @property
    def nome(self):
        return self.__nome

    @nome.setter
    def nome(self, nome):
        self.__nome = nome

    @property
    def dataNascimento(self):
        return self.__dataNascimento

    @dataNascimento.setter
    def dataNascimento(self, dataNascimento):
        self.__dataNascimento = dataNascimento

    @property
    def cpf(self):
        return self.__cpf

    @cpf.setter
    def cpf(self, cpf):
        if len(cpf) != 14:
            raise ValueError("O CPF deve conter 14 caracteres.")
        self.__cpf = cpf

    @property
    def rg(self):
        return self.__rg

    @rg.setter
    def rg(self, rg):
        self.__rg = rg
```

---

main_pessoa.py

```
from Pessoa import Pessoa

pessoa = Pessoa(
    nome="Vivian Souza",
    dataNascimento="07/11/1977",
    cpf="598.247.956-57",
    rg="92648175-1",
    status=False
)

print(f"Nome: {pessoa.nome}, Data de Nascimento: {pessoa.dataNascimento}, CPF: {pessoa.cpf}, RG: {pessoa.rg}, Status: {pessoa.status}")


try:
    pessoa.cpf = "000.001.002"
except ValueError as e:
    print(e)

pessoa.cpf = "598.247.956-57"
print(f"Nome: {pessoa.nome}, Data de Nascimento: {pessoa.dataNascimento}, CPF: {pessoa.cpf}, RG: {pessoa.rg}, Status: {pessoa.status}")
```

Missão Prática | Conhecendo novos paradigmas 💻

Calculadora.py

```
class Calculadora:
    def __init__(self, valorA=0, valorB=0, operacao=''):
        self.__valorA = valorA
        self.__valorB = valorB
        self.__operacao = operacao

    @property
    def valorA(self):
        return self.__valorA

    @valorA.setter
    def valorA(self, valorA):
        self.__valorA = valorA

    @property
    def valorB(self):
        return self.__valorB

    @valorB.setter
    def valorB(self, valorB):
        self.__valorB = valorB

    @property
    def operacao(self):
        return self.__operacao

    @operacao.setter
    def operacao(self, operacao):
        self.__operacao = operacao

    def validarOperacao(self):
        operacoes_validas = ['+', '-', '*', '/']
        return self.__operacao in operacoes_validas

    def calcular(self):
        if not self.validarOperacao():
            print(f"Operação inválida: {self.__operacao}. As operações válidas são +, - , *, /. ")
            exit()

        if self.__operacao == '+':
            return self.__valorA + self.__valorB
        elif self.__operacao == '-':
            return self.__valorA - self.__valorB
        elif self.__operacao == '*':
            return self.valorA * self.__valorB
        elif self.__operacao == '/':
            if self.__valorB == 0:
                print("Não é possível dividir o valor por 0.")
                exit()
            return self.__valorA / self.__valorB

    def mostrarResultado(self):
        resultado = self.calcular()
        print(f"{self.__valorA} {self.__operacao} {self.__valorB} = {resultado}")
    def entradaDados(self):
        self.__valorA = float(input("Digite o valor A: "))
        self.__operacao = input("Digite a operação matemática: ")
        self.__valorB = float(input("Digite o valor B: "))
        if not self.validarOperacao():
            print(f"Operação inválida: {self.__operacao}. As operações válidas são +, -, *, e /.")
            exit()
```

OBS: com as especificações em relação ao trabalho achei melhor deixar dois tipos de códigos. um mostrando os resultados de todas as operações de uma vês e outro com a entrada de dados.

main.calculadora.py

```
from Calculadora import Calculadora

calc = Calculadora()
calc.valorA = 40
calc.valorB = 20
calc.operacao = '+'

calc.mostrarResultado()
resultado = calc.calcular()
print(f"Resultado: {resultado}")

calc.operacao = '-'
calc.mostrarResultado()
resultado = calc.calcular()
print(f"resultado: {resultado}")

calc.operacao = '*'
calc.mostrarResultado()
resultado = calc.calcular()
print(f"resultado: {resultado}")

calc.operacao = '/'
calc.mostrarResultado()
resultado = calc.calcular()
print(f"resultado: {resultado}")


calc.valorB = 0
try:
    resultado = calc.calcular()
except SystemExit as e:
    print("tentativa de divisão por 0 detectada")
```

---

main _calculadora2.py

```
from Calculadora import Calculadora

calc = Calculadora()
calc.entradaDados()

calc.mostrarResultado()
```
