DESENVOLVIMENTO FULL STACK
NÍVEL 4: CONHECENDO NOVOS PARADIGMAS

Microatividade-1: Classe em Python

```python
# Microatividade 1 - Descrever a criação de classes em Python
class Pessoa:
    def __init__(self, nome, dataNascimento, cpf, rg):
        self.nome = nome
        self.dataNascimento = dataNascimento
        self.cpf = cpf
        self.rg = rg
```

Microatividade 2: Descrever a instanciação e utilização de objetos em Python

```python
from Pessoa import Pessoa

pessoa = Pessoa ("João", "2000-01-01", "000.111.222-33", "15975388-1" )

attrs = vars(pessoa)

print('Instancia da classe Pessoa: ')

print(', '.join("%s: %s" % item for item in attrs.items()))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS
PS C:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4> & C:/Users/PC/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/Users/PC/OneDrive/Do
os/Códigos Python/Atividades_nivel_4/main_pessoa.py"
Instancia da classe Pessoa:
nome: João, dataNascimento: 2000-01-01, cpf: 000.111.222-33, rg: 15975388-1
PS C:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4>
```

Microatividade 3: Descrever a declaração de métodos em classes no Python

1.

```python
class Pessoa:
    def __init__(self, nome, dataNascimento, cpf, rg, status):
        self.nome = nome
        self.dataNascimento = dataNascimento
        self.cpf = cpf
        self.rg = rg
        self.status = status

    def alterarStatus(self, status):
        self.status = status

```

2.

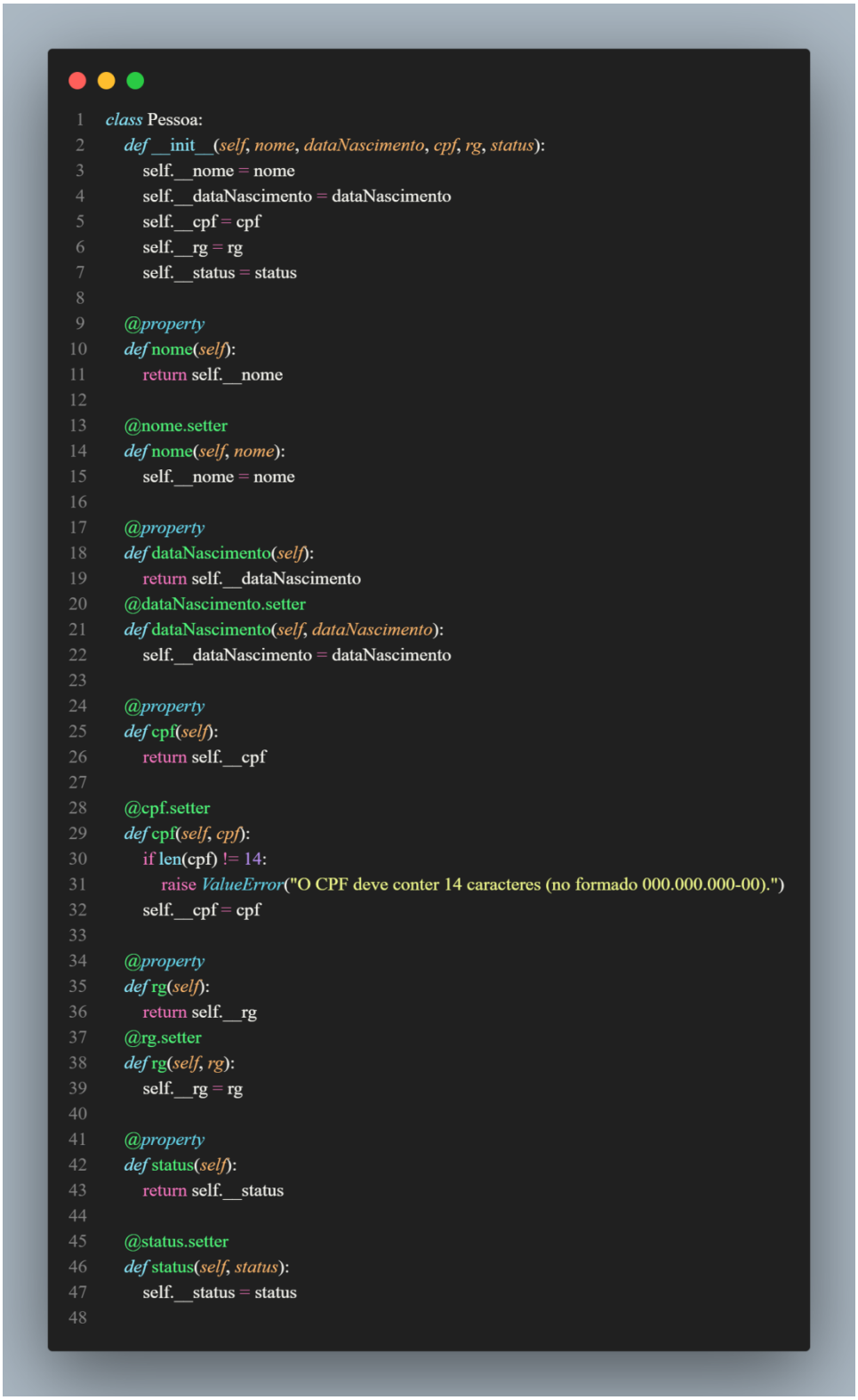
```python
from Pessoa import Pessoa

pessoa = Pessoa ("João", "2000-01-01", "000.111.222-33", "15975388-1", "False")

pessoa.alterarStatus(False)

attrs = vars(pessoa)

print('Instancia da classe Pessoa: ')

print(', '.join("%s: %s" % item for item in attrs.items()))
```

3.

```
Instancia da classe Pessoa:
nome: João, dataNascimento: 2000-01-01, cpf: 000.111.222-33, rg: 15975388-1, status: True
PS C:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4> & C:/Users/PC/AppData/Local/
Documentos/Códigos Python/Atividades_nivel_4/main_pessoa.py"
Instancia da classe Pessoa:
nome: João, dataNascimento: 2000-01-01, cpf: 000.111.222-33, rg: 15975388-1, status: False
PS C:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4>
```

Microatividade 4: Descrever a utilização de encapsulamento em Python

1.



```python
class Pessoa:
    def __init__(self, nome, dataNascimento, cpf, rg, status):
        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 (no formado 000.000.000-00).")
        self.__cpf = cpf

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

    @property
    def status(self):
        return self.__status

    @status.setter
    def status(self, status):
        self.__status = status

```

2.

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

@cpf.setter
def cpf(self, cpf):
    cpf = cpf.zfill(14)
if not cpf.isdigit():
    raise ValueError("O CPF deve ser composto apenas por números.")
    self.__cpf = cpf
```

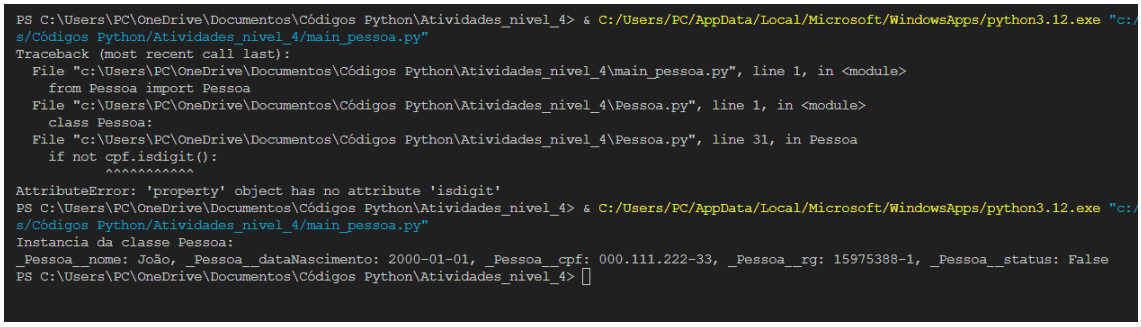
3.

```
PS C:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4> & C:/Users/PC/AppData/Local/Microsoft/WindowsApps/p
s/Códigos Python/Atividades_nivel_4/main_pessoa.py"
Traceback (most recent call last):
  File "c:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4\main_pessoa.py", line 1, in <module>
    from Pessoa import Pessoa
  File "c:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4\Pessoa.py", line 1, in <module>
    class Pessoa:
  File "c:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4\Pessoa.py", line 31, in Pessoa
    if not cpf.isdigit():
           ^^^^^^^^^^^
AttributeError: 'property' object has no attribute 'isdigit'
PS C:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4>
```

4.

```python
@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 (no formado 000.000.000-00).")
    self.__cpf = cpf
```

5.


PS C:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4> & C:/Users/PC/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/
s/Códigos Python/Atividades_nivel_4/main_pessoa.py"
Traceback (most recent call last):
File "c:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4\main_pessoa.py", line 1, in <module>
from Pessoa import Pessoa
File "c:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4\Pessoa.py", line 1, in <module>
class Pessoa:
File "c:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4\Pessoa.py", line 31, in Pessoa
if not cpf.isdigit():
^^^^^^^^^^
AttributeError: 'property' object has no attribute 'isdigit'
PS C:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4> & C:/Users/PC/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/
s/Códigos Python/Atividades_nivel_4/main_pessoa.py"
Instancia da classe Pessoa:
_Pessoa__nome: João, _Pessoa__dataNascimento: 2000-01-01, _Pessoa__cpf: 000.111.222-33, _Pessoa__rg: 15975388-1, _Pessoa__status: False
PS C:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4>

Microatividade 5: Descrever a reutilização de código através de herança no Python

1. Pessoa.py

```python
class Pessoa:
    def __init__(self, nome, numeroConta, dataAberturaConta, status):
        self.__nome = nome
        self.__numeroConta = numeroConta
        self.__dataAberturaConta = dataAberturaConta
        self.__status = status

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

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

    @property
    def numeroConta(self):
        return self.__numeroConta
    @numeroConta.setter
    def numeroConta(self, numeroConta):
        self.__numeroConta = numeroConta

    @property
    def dataAberturaConta(self):
        return self.__dataAberturaConta

    @dataAberturaConta.setter
    def dataAberturaConta(self, dataAberturaConta):
        self.__dataAberturaConta = dataAberturaConta

    @property
    def status(self):
        return self.__status

    @status.setter
    def status(self, status):
        self.__status = status


```

2.Main_pessoa.py

```python
from Pessoa import Pessoa
from PessoaFisica import PessoaFisica
from PessoaJuridica import PessoaJuridica

pessoa = Pessoa ("João", "12345678-9","2000-01-01", False)
pessoa_fisica = PessoaFisica("25-02-1991", "000.111.222-33", "15975388-1 ")
pessoa_juridica = PessoaJuridica("01-01-2019", "12.333.444/0001-22")


print("Instância da classe Pessoa:")
print(f"Nome: {pessoa.nome}")
print(f"Número da conta: {pessoa.numeroConta}")
print(f"Data de abertura da conta: {pessoa.dataAberturaConta}")
print(f"Status: {pessoa.status}")

print("Instância da classe PessoaFisica:")
print(f"Data de nascimento: {pessoa_fisica.dataNascimento}")
print(f"CPF: {pessoa_fisica.cpf}")
print(f"RG: {pessoa_fisica.rg}")

print("Instância da classe PessoaJuridica:")
print(f"Data de abertura da empresa: {pessoa_juridica.dataAberturaEmpresa}")
print(f"CNPJ: {pessoa_juridica.cnpj}")

```

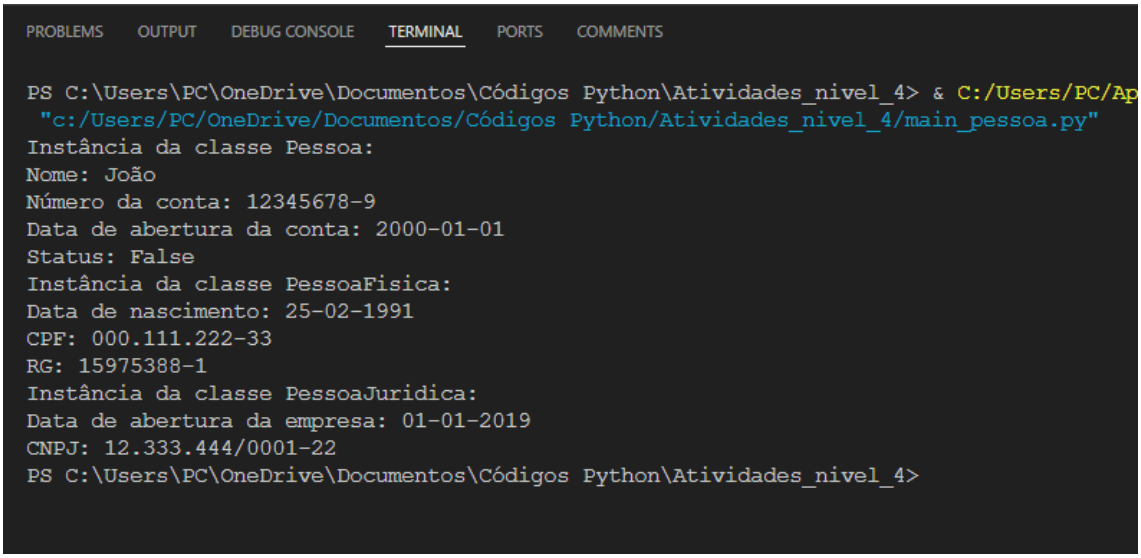
3.PessoaFisica.py

```python
from Pessoa import Pessoa

class PessoaFisica(Pessoa):
    def __init__(self, dataNascimento, cpf, rg):
        self.__dataNascimento = dataNascimento
        self.__cpf = cpf
        self.__rg = rg


    @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("CPF deve ter 14 caracteres(no formato XXX.XXX.XXX-XX)")
        self.__cpf = cpf

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

4.PessoaJuridica.py

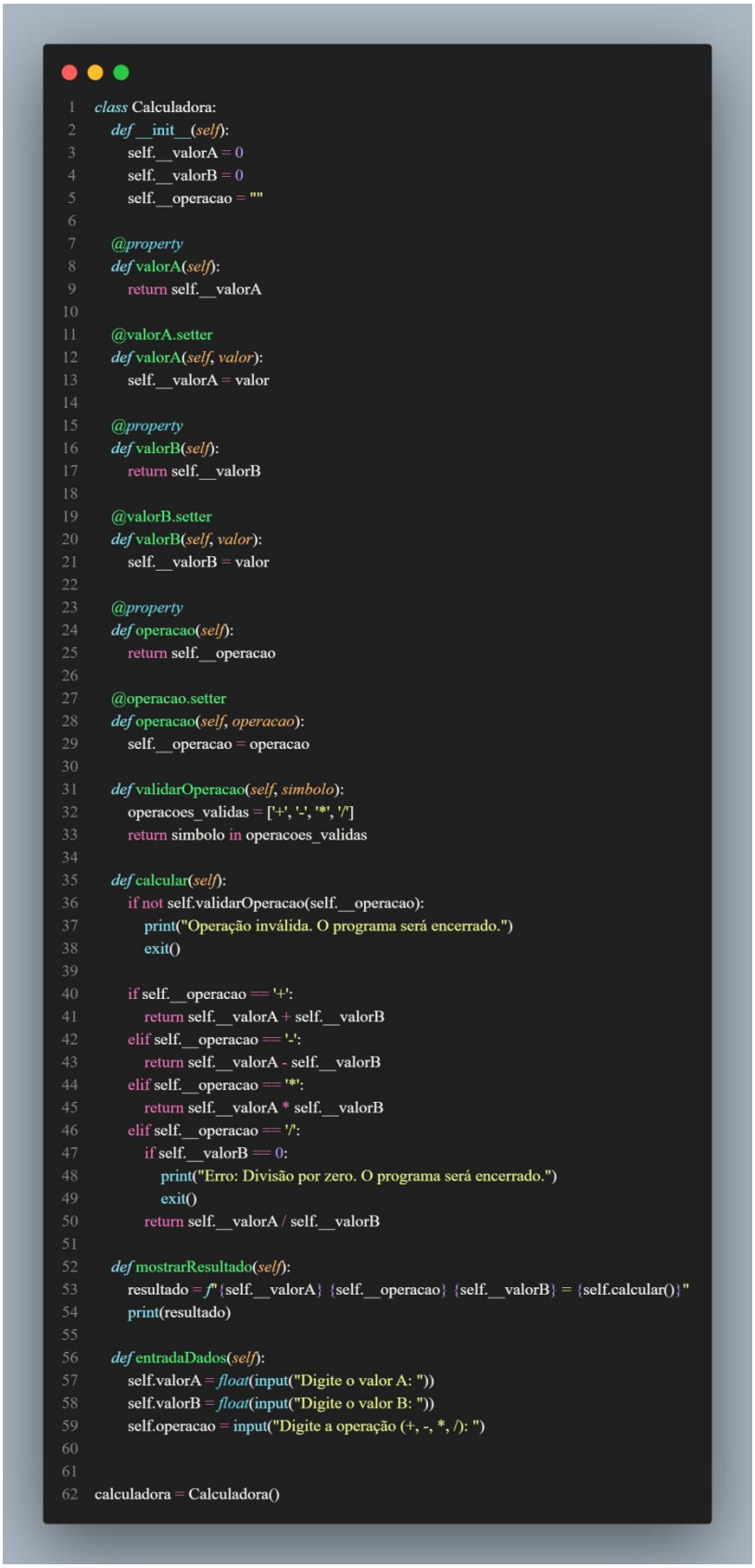
```python
from Pessoa import Pessoa

class PessoaJuridica(Pessoa):
    def __init__(self, dataAberturaEmpresa, cnpj):
        self.__dataAberturaEmpresa = dataAberturaEmpresa
        self.__cnpj = cnpj

    @property
    def dataAberturaEmpresa(self):
        return self.__dataAberturaEmpresa

    @dataAberturaEmpresa.setter
    def dataAberturaEmpresa(self, dataAberturaEmpresa):
        self.__dataAberturaEmpresa = dataAberturaEmpresa

    @property
    def cnpj(self):
        return self.__cnpj

    @cnpj.setter
    def cnpj(self, cnpj):
        if len(cnpj)!=18:
            raise ValueError("CNPJ deve ter 18 caracteres(no formato 00.000.000/0001-00)")
        self.__cnpj = cnpj
```

5. Resultado


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS
PS C:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4> & C:/Users/PC/Ap
"c:/Users/PC/OneDrive/Documentos/Códigos Python/Atividades_nivel_4/main_pessoa.py"
Instância da classe Pessoa:
Nome: João
Número da conta: 12345678-9
Data de abertura da conta: 2000-01-01
Status: False
Instância da classe PessoaFisica:
Data de nascimento: 25-02-1991
CPF: 000.111.222-33
RG: 15975388-1
Instância da classe PessoaJuridica:
Data de abertura da empresa: 01-01-2019
CNPJ: 12.333.444/0001-22
PS C:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4>

Missão Prática | Conhecendo novos paradigmas

1.Calculadora



```python
class Calculadora:
    def __init__(self):
        self.__valorA = 0
        self.__valorB = 0
        self.__operacao = ""

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

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

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

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

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

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

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

    def calcular(self):
        if not self.validarOperacao(self.__operacao):
            print("Operação inválida. O programa será encerrado.")
            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("Erro: Divisão por zero. O programa será encerrado.")
                exit()
            return self.__valorA / self.__valorB

    def mostrarResultado(self):
        resultado = f"{self.__valorA} {self.__operacao} {self.__valorB} = {self.calcular()}"
        print(resultado)

    def entradaDados(self):
        self.valorA = float(input("Digite o valor A: "))
        self.valorB = float(input("Digite o valor B: "))
        self.operacao = input("Digite a operação (+, -, *, /): ")


calculadora = Calculadora()
```

2.Main_calculadora

```python
# main_calculadora.py

from calculadora import Calculadora

def main():
    calc = Calculadora()
    calc.entradaDados()  # Permite a entrada de dados pelo usuário
    calc.mostrarResultado()

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

Resultado

```
os Python/Estacio-exemplos/Threading/main_calculadora.py"
Digite o valor A: 10
Digite o valor B: 20
Digite a operação (+, -, *, /): +
10.0 + 20.0 = 30.0
PS C:\Users\PC\OneDrive\Documentos\Códigos Python\Estacio-exemplos> & C:/Users/P
stacio-exemplos/Threading/main_calculadora.py"
Digite o valor A: 10
Digite o valor B: 20
Digite a operação (+, -, *, /): -
10.0 - 20.0 = -10.0
PS C:\Users\PC\OneDrive\Documentos\Códigos Python\Estacio-exemplos> & C:/Users/P
stacio-exemplos/Threading/main_calculadora.py"
Digite o valor A: 10
Digite o valor B: 2
Digite a operação (+, -, *, /): /
10.0 / 2.0 = 5.0
PS C:\Users\PC\OneDrive\Documentos\Códigos Python\Estacio-exemplos> & C:/Users/P
stacio-exemplos/Threading/main_calculadora.py"
Digite o valor A: 10
Digite o valor B: 3
Digite a operação (+, -, *, /): *
10.0 * 3.0 = 30.0
PS C:\Users\PC\OneDrive\Documentos\Códigos Python\Estacio-exemplos>
```