

# DESENVOLVIMENTO FULL STACK

## NÍVEL 4: CONHECENDO NOVOS PARADIGMAS

---

### Microatividade-1: Classe em Python

```
1 # Microatividade 1 - Descrever a criação de classes em Python
2 class Pessoa:
3     def __init__(self, nome, dataNascimento, cpf, rg):
4         self.nome = nome
5         self.dataNascimento = dataNascimento
6         self.cpf = cpf
7         self.rg = rg
```

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

```
1 from Pessoa import Pessoa
2
3 pessoa = Pessoa("João", "2000-01-01", "000.111.222-33", "15975388-1")
4
5 attrs = vars(pessoa)
6
7 print('Instancia da classe Pessoa: ')
8
9 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/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
PS C:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4>
```

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

1.

```
1 class Pessoa:
2     def __init__(self, nome, dataNascimento, cpf, rg, status):
3         self.nome = nome
4         self.dataNascimento = dataNascimento
5         self.cpf = cpf
6         self.rg = rg
7         self.status = status
8
9     def alterarStatus(self, status):
10         self.status = status
11
```

2.

```
1 from Pessoa import Pessoa
2
3 pessoa = Pessoa("João", "2000-01-01", "000.111.222-33", "15975388-1", "False")
4
5 pessoa.alterarStatus(False)
6
7 attrs = vars(pessoa)
8
9 print('Instancia da classe Pessoa: ')
10
11 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.

```
1  class Pessoa:
2      def __init__(self, nome, dataNascimento, cpf, rg, status):
3          self.__nome = nome
4          self.__dataNascimento = dataNascimento
5          self.__cpf = cpf
6          self.__rg = rg
7          self.__status = status
8
9      @property
10     def nome(self):
11         return self.__nome
12
13     @nome.setter
14     def nome(self, nome):
15         self.__nome = nome
16
17     @property
18     def dataNascimento(self):
19         return self.__dataNascimento
20     @dataNascimento.setter
21     def dataNascimento(self, dataNascimento):
22         self.__dataNascimento = dataNascimento
23
24     @property
25     def cpf(self):
26         return self.__cpf
27
28     @cpf.setter
29     def cpf(self, cpf):
30         if len(cpf) != 14:
31             raise ValueError("O CPF deve conter 14 caracteres (no formado 000.000.000-00).")
32         self.__cpf = cpf
33
34     @property
35     def rg(self):
36         return self.__rg
37     @rg.setter
38     def rg(self, rg):
39         self.__rg = rg
40
41     @property
42     def status(self):
43         return self.__status
44
45     @status.setter
46     def status(self, status):
47         self.__status = status
48
```

2.

```
1  @property
2  def cpf(self):
3      return self.__cpf
4
5  @cpf.setter
6  def cpf(self, cpf):
7      cpf = cpf.zfill(14)
8      if not cpf.isdigit():
9          raise ValueError("O CPF deve ser composto apenas por números.")
10     self.__cpf = cpf
```

3.

```
PS C:\Users\PC\OneDrive\Documentos\Códigos Python\Atividades_nivel_4> & C:/Users/PC/AppData/Local/Microsoft/WindowsApps/powershell.exe C:\Users\PC\OneDrive\Documentos\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.

```
1  @property
2  def cpf(self):
3      return self.__cpf
4
5  @cpf.setter
6  def cpf(self, cpf):
7      if len(cpf) != 14:
8          raise ValueError("O CPF deve conter 14 caracteres (no formato 000.000.000-00).")
9      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

```
1 class Pessoa:
2     def __init__(self, nome, numeroConta, dataAberturaConta, status):
3         self.__nome = nome
4         self.__numeroConta = numeroConta
5         self.__dataAberturaConta = dataAberturaConta
6         self.__status = status
7
8     @property
9     def nome(self):
10         return self.__nome
11
12     @nome.setter
13     def nome(self, nome):
14         self.__nome = nome
15
16     @property
17     def numeroConta(self):
18         return self.__numeroConta
19     @numeroConta.setter
20     def numeroConta(self, numeroConta):
21         self.__numeroConta = numeroConta
22
23     @property
24     def dataAberturaConta(self):
25         return self.__dataAberturaConta
26
27     @dataAberturaConta.setter
28     def dataAberturaConta(self, dataAberturaConta):
29         self.__dataAberturaConta = dataAberturaConta
30
31     @property
32     def status(self):
33         return self.__status
34
35     @status.setter
36     def status(self, status):
37         self.__status = status
38
39
```

## 2.Main\_pessoa.py

```
1  from Pessoa import Pessoa
2  from PessoaFisica import PessoaFisica
3  from PessoaJuridica import PessoaJuridica
4
5  pessoa = Pessoa("João", "12345678-9", "2000-01-01", False)
6  pessoa_fisica = PessoaFisica("25-02-1991", "000.111.222-33", "15975388-1 ")
7  pessoa_juridica = PessoaJuridica("01-01-2019", "12.333.444/0001-22")
8
9
10 print("Instância da classe Pessoa:")
11 print(f"Nome: {pessoa.nome}")
12 print(f"Número da conta: {pessoa.numeroConta}")
13 print(f>Data de abertura da conta: {pessoa.dataAberturaConta}")
14 print(f>Status: {pessoa.status}")
15
16 print("Instância da classe PessoaFisica:")
17 print(f>Data de nascimento: {pessoa_fisica.dataNascimento}")
18 print(f>CPF: {pessoa_fisica.cpf}")
19 print(f>RG: {pessoa_fisica.rg}")
20
21 print("Instância da classe PessoaJuridica:")
22 print(f>Data de abertura da empresa: {pessoa_juridica.dataAberturaEmpresa}")
23 print(f>CNPJ: {pessoa_juridica.cnpj}")
24
```

### 3.PessoaFisica.py

```
1  from Pessoa import Pessoa
2
3  class PessoaFisica(Pessoa):
4      def __init__(self, dataNascimento, cpf, rg):
5          self.__dataNascimento = dataNascimento
6          self.__cpf = cpf
7          self.__rg = rg
8
9
10     @property
11     def dataNascimento(self):
12         return self.__dataNascimento
13     @dataNascimento.setter
14     def dataNascimento(self, dataNascimento):
15         self.__dataNascimento = dataNascimento
16
17     @property
18     def cpf(self):
19         return self.__cpf
20     @cpf.setter
21     def cpf(self, cpf):
22         if len(cpf) != 14:
23             raise ValueError("CPF deve ter 14 caracteres(no formato XXX.XXX.XXX-XX)")
24         self.__cpf = cpf
25
26     @property
27     def rg(self):
28         return self.__rg
29     @rg.setter
30     def rg(self, rg):
31         self.__rg = rg
```



## 4.PessoaJuridica.py

```
1  from Pessoa import Pessoa
2
3  class PessoaJuridica(Pessoa):
4      def __init__(self, dataAberturaEmpresa, cnpj):
5          self.__dataAberturaEmpresa = dataAberturaEmpresa
6          self.__cnpj = cnpj
7
8      @property
9      def dataAberturaEmpresa(self):
10         return self.__dataAberturaEmpresa
11
12     @dataAberturaEmpresa.setter
13     def dataAberturaEmpresa(self, dataAberturaEmpresa):
14         self.__dataAberturaEmpresa = dataAberturaEmpresa
15
16     @property
17     def cnpj(self):
18         return self.__cnpj
19
20     @cnpj.setter
21     def cnpj(self, cnpj):
22         if len(cnpj)!=18:
23             raise ValueError("CNPJ deve ter 18 caracteres(no formato 00.000.000/0001-00)")
24         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/AppData/Local/Programs/Python/Python39-6/Scripts/python.exe "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

```
1 class Calculadora:
2     def __init__(self):
3         self.__valorA = 0
4         self.__valorB = 0
5         self.__operacao = ""
6
7     @property
8     def valorA(self):
9         return self.__valorA
10
11    @valorA.setter
12    def valorA(self, valor):
13        self.__valorA = valor
14
15    @property
16    def valorB(self):
17        return self.__valorB
18
19    @valorB.setter
20    def valorB(self, valor):
21        self.__valorB = valor
22
23    @property
24    def operacao(self):
25        return self.__operacao
26
27    @operacao.setter
28    def operacao(self, operacao):
29        self.__operacao = operacao
30
31    def validarOperacao(self, simbolo):
32        operacoes_validas = ['+', '-', '*', '/']
33        return simbolo in operacoes_validas
34
35    def calcular(self):
36        if not self.validarOperacao(self.__operacao):
37            print("Operação inválida. O programa será encerrado.")
38            exit()
39
40        if self.__operacao == '+':
41            return self.__valorA + self.__valorB
42        elif self.__operacao == '-':
43            return self.__valorA - self.__valorB
44        elif self.__operacao == '*':
45            return self.__valorA * self.__valorB
46        elif self.__operacao == '/':
47            if self.__valorB == 0:
48                print("Erro: Divisão por zero. O programa será encerrado.")
49                exit()
50            return self.__valorA / self.__valorB
51
52    def mostrarResultado(self):
53        resultado = f"{self.__valorA} {self.__operacao} {self.__valorB} = {self.calcular()}"
54        print(resultado)
55
56    def entradaDados(self):
57        self.valorA = float(input("Digite o valor A: "))
58        self.valorB = float(input("Digite o valor B: "))
59        self.operacao = input("Digite a operação (+, -, *, /): ")
60
61
62 calculadora = Calculadora()
```

## 2.Main\_calculadora

```
1 # main_calculadora.py
2
3 from calculadora import Calculadora
4
5 def main():
6     calc = Calculadora()
7     calc.entradaDados() # Permite a entrada de dados pelo usuário
8     calc.mostrarResultado()
9
10 if __name__ == "__main__":
11     main()
```

## Resultado

```
PS 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/PC/OneDrive/Documentos/Códigos Python/Estacio-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/PC/OneDrive/Documentos/Códigos Python/Estacio-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/PC/OneDrive/Documentos/Códigos Python/Estacio-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> 
```

