

Módulo 06 | Python: Programação Orientada a Objetos

Caderno de **Aula**

Desenvolvedor [Enzo Schitini](#)

▼ Tópicos

1. Um pouco de teoria;
2. Classes;
3. Objetos;
4. Herança.

▼ Aulas

▼ 0. Paradigmas de Programação

Estilos de programação.

1. Imperativa;
2. Funcional;
3. Orientada a objetos.

O Python é uma linguagem [multi-paradigma](#).



▼ 1. Um pouco de teoria

▼ 1.1. Classe

Uma representação de um elemento, real ou não. Uma receita para criar objetos (instâncias).

Exemplo: pessoa.

▼ 1.2. Objeto

Uma instancia de uma classe. Dá vida a receita (classe).

Exemplo: André é um instância da classe pessoa.

▼ 1.3. Herança

Uma especialização da classe.

Exemplo: estudante é um tipo especial de pessoa.

▼ 2. Classes

▼ 2.1. Definição

Uma classe é uma receita para criação de objetos.

```
class NomeClasse(object):

    def __init__(self, params):
        self.atributo1 = ...
        self.atributo2 = ...

    def metodo1(self, params):
        ...

    def metodo2(self, params):
        ...
```

```
class Pessoa(object):

    def __init__(self):
        pass
```

▼ 2.2. Atributos

Representam as características da classe.

```
class Pessoa(object):

    def __init__(self, nome: str, idade: int, documento: str):
        self.nome = nome
        self.idade = idade
        self.documento = documento
```

▼ 2.3. Métodos

Representam as ações da classe.

```
from time import sleep

class Pessoa(object):

    def __init__(self, nome: str, idade: int, documento: str = None):
        self.nome = nome
        self.idade = idade
        self.documento = documento

    def dormir(self, horas: int) -> None:
        for hora in range(1, horas+1):
            print(f'Dormindo por {hora} horas')
            sleep(1)

    def falar(self, texto: str) -> None:
        print(texto)

    def __str__(self) -> str:
        return f'{self.nome}, {self.idade} anos e documento numero {self.documento}'
```

▼ 3. Objetos

▼ 3.1. Definição

Uma objeto é uma instância de uma classe.

```
class NomeClasse(object):  
    ...  
  
objeto = NomeClasse()  
  
objeto.atributo  
objeto.metodo()
```

```
andre = Pessoa(nome='Andre Perez', idade=30, documento='123')  
maria = Pessoa(nome='Maria Perez', idade=56, documento='456')  
pedro = Pessoa(nome='Pedro Perez', idade=8)
```

▼ 3.2. Manipulação

- Atributos

```
print(andre.nome)  
  
Andre Perez
```

```
def maior_de_idade(idade: int) -> bool:  
    return idade >= 18  
  
if maior_de_idade(idade=andre.idade):  
    print(f'{andre.nome} é maior de idade')  
  
Andre Perez é maior de idade
```

```
score_credito = {'123': 750, '456': 812, '789': 327}  
  
score = score_credito[andre.documento]  
print(score)  
  
750
```

- Métodos

```
andre.dormir(horas=8)  
  
andre.falar(texto='Olá pessoal!')  
  
print(andre)  
  
type(andre)
```

▼ 3.3. Exemplos

- Tudo no Python é um objeto!

```
tipos = [type(10), type(1.1), type('EBAC'), type(True), type(None), type([1, 2, 3]), type({1, 2, 3}), type({'janeiro': 1}), type(lambda  
  
for tipo in tipos:  
    print(tipo)  
  
nome = 'Andre Perez'  
print(nome.upper())  
  
juros = [0.02, 0.07, 0.15]  
print(juros.pop(1))
```

- Classe Arquivo CSV

```
class ArquivoCSV(object):

    def __init__(self, arquivo: str):
        self.arquivo = arquivo
        self.conteudo = self._extrair_conteudo()
        self.colunas = self._extrair_nome_colunas()

    def _extrair_conteudo(self):
        conteudo = None
        with open(file=self.arquivo, mode='r', encoding='utf8') as arquivo:
            conteudo = arquivo.readlines()
        return conteudo

    def _extrair_nome_colunas(self):
        return self.conteudo[0].strip().split(sep=',')

    def extrair_coluna(self, indice_coluna: str):
        coluna = list()
        for linha in self.conteudo:
            conteudo_linha = linha.strip().split(sep=',')
            coluna.append(conteudo_linha[indice_coluna])
        coluna.pop(0)
        return coluna
```

▼ Função strip() com split()

```
# Exemplo 1: Removendo espaços em branco
fruta = " banana "
print(fruta.strip()) # Saída: "banana"

# Exemplo 2: Removendo caracteres específicos
fruta = ",,.,,.,rrttgg....banana....rrr"
print(fruta.strip(",.gnt")) # Saída: "banana"

# Exemplo: Removendo espaços em branco e dividindo uma string
frase = " 0 gato pulou sobre o muro "
palavras = frase.strip().split()
print(palavras) # Saída: ['0', 'gato', 'pulou', 'sobre', 'o', 'muro']


banana
banana
['0', 'gato', 'pulou', 'sobre', 'o', 'muro']
```

```
%%writefile banco.csv
age,job,marital,education,default,balance,housing,loan
30,unemployed,married,primary,no,1787,no,no
33,services,married,secondary,no,4789,yes,yes
35,management,single,tertiary,no,1350,yes,no
30,management,married,tertiary,no,1476,yes,yes
59,blue-collar,married,secondary,no,0,yes,no
35,management,single,tertiary,no,747,no,no
36,self-employed,married,tertiary,no,307,yes,no
39,technician,married,secondary,no,147,yes,no
41,entrepreneur,married,tertiary,no,221,yes,no
43,services,married,primary,no,-88,yes,yes
```

```
arquivo_banco = ArquivoCSV(arquivo='./banco.csv')
```

```
cont = arquivo_banco._extrair_conteudo()
print(cont)
```

```
[ 'age,job,marital,education,default,balance,housing,loan\n', '30,unemployed,married,primary,no,1787,no,no\n', '33,services,married,
```



```
print(arquivo_banco.colunas)
```

```
[ 'age', 'job', 'marital', 'education', 'default', 'balance', 'housing', 'loan']
```

- Extrair a coluna de job

```
job = arquivo_banco.extrair_coluna(indice_coluna=1)
print(job)
```

```
['unemployed', 'services', 'management', 'management', 'blue-collar', 'management', 'self-employed', 'technician', 'entrepreneur',
```

- Extraindo a coluna de education

```
education = arquivo_banco.extrair_coluna(indice_coluna=3)
print(education)
```

▼ 4. Herança

▼ 4.1. Definição

Todo estudante é uma pessoa mas nem toda pessoa é um estudante. Por isso usamos as Heranças para que os estudantes possam compartilhar as características das pessoas.

Uma especialização da classe.

```
class NomeClasse(object):

    def __init__(self, params):
        ...

class NomeClasseEspecializada(NomeClasse):

    def __init__(self, params):
        super().__init__(self, params)
        self.atributo3 = ...
        self.atributo4 = ...

    def metodo3(self, params):
        ...

    def metodo4(self, params):
        ...
```

Repetindo a definição da classe Pessoa:

```
from time import sleep

class Pessoa(object):

    def __init__(self, nome: str, idade: int, documento: str=None):
        self.nome = nome
        self.idade = idade
        self.documento = documento

    def dormir(self, horas: int) -> None:
        for hora in range(1,horas+1):
            print(f'Dormindo por {hora} horas')
            sleep(1)

    def falar(self, texto: str) -> None:
        print(texto)

    def __str__(self) -> str:
        return f'{self.nome}, {self.idade} anos e documento numero {self.documento}'
```

Criando a classe Universidade

```
class Universidade(object):

    def __init__(self, nome: str):
        self.nome = nome
```

Especializando a classe Pessoa na classe Estudante:

```
class Estudante(Pessoa):  
  
    def __init__(self, nome: str, idade: int, documento: str, universidade: Universidade):  
  
        super().__init__(nome=nome, idade=idade, documento=documento)  
        self.universidade = universidade
```

▼ 4.2. Manipulação

```
usp = Universidade(nome='Universidade de São Paulo')  
andre = Estudante(nome='Andre Perez', idade=30, documento='123', universidade=usp)
```

```
print(andre)
```

```
Andre Perez, 30 anos e documento numero 123
```

```
print(andre.universidade.nome)
```

```
Universidade de São Paulo
```