



escola
britânica de
artes criativas
& tecnologia

Módulo | Python: Programação Orientada a Objetos

Caderno de **Aula**

Professor [André Perez](#)

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):  
        ...
```

In [1]:

```
class Pessoa(object):  
  
    def __init__(self):  
        pass
```

2.2. Atributos

Representam as características da classe.

```
In [2]: 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.

```
In [3]: 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()
```

```
In [4]: 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

```
In [5]: print(andre.nome)
```

Andre Perez

```
In [6]: 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

```
In [7]: score_credito = {
        '123': 750,
        '456': 812,
        '789': 327
    }

    score = score_credito[andre.documento]
    print(score)
```

750

- Métodos

```
In [8]: andre.dormir(horas=8)
```

Dormindo por 1 horas
Dormindo por 2 horas
Dormindo por 3 horas
Dormindo por 4 horas
Dormindo por 5 horas
Dormindo por 6 horas
Dormindo por 7 horas
Dormindo por 8 horas

```
In [9]: andre.falar(texto='Olá pessoal!')
```

Olá pessoal!

```
In [10]: print(andre)
```

Andre Perez, 30 anos e documento numero 123

```
In [11]: type(andre)
```

```
Out[11]: __main__.Pessoa
```

3.3. Exemplos

- Tudo no Python é um objeto!

```
In [12]: 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 x: x)  
]
```

```
In [13]: for tipo in tipos:  
    print(tipo)
```

```
<class 'int'>  
<class 'float'>  
<class 'str'>  
<class 'bool'>  
<class 'NoneType'>  
<class 'list'>  
<class 'set'>  
<class 'dict'>  
<class 'function'>
```

```
In [14]: nome = 'Andre Perez'  
    print(nome.upper())
```

ANDRE PEREZ

```
In [15]: juros = [0.02, 0.07, 0.15]  
    print(juros.pop(1))
```

0.07

- Classe Arquivo CSV

In [16]:

```
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
```

In [17]:

```
%%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
```

Writing banco.csv

In [18]:

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

In [19]:

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

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

- Extraíndo a coluna de `job`

In [20]:

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

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

- Extraíndo a coluna de `education`

In [21]:

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

```
['primary', 'secondary', 'tertiary', 'tertiary', 'secondary', 'tertiary', 'tertiary', 'secondary', 'tertiary', 'primary']
```

4. Herança

4.1. Definição

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`:

In [22]:

```
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}, ' +
            f'{self.idade} anos e documento numero {self.documento}'
```

Criando a classe `Universidade`

In [23]:

```
class Universidade(object):

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

Especializando a classe `Pessoa` na classe `Estudante`:

In [24]:

```
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


```
In [25]: usp = Universidade(nome='Universidade de São Paulo')
         andre = Estudante(
             nome='Andre Perez',
             idade=30,
             documento='123',
             universidade=usp
         )
```

```
In [26]: print(andre)
```

Andre Perez, 30 anos e documento numero 123

```
In [27]: print(andre.universidade.nome)
```

Universidade de São Paulo