Aula 08 - Programação Orientada à Objetos - Parte 01

LISTA DE FIGURAS

LISTA DE CÓDIGOS-FONTE

SUMÁRIO

1. PROGRAMAÇÃO ORIENTADA À OBJETOS

1.1. Classe com construtor não padrão

Em Python, sempre temos que criar uma classe já inicializando seus atributos com construtor

```
class Pessoa:
    def __init__(self, nome, telefone, idade):
        self.nome = nome
        self.telefone = telefone
        self.idade = idade
```

Código fonte 1 - Construtor não padrão
Fonte: Elaborado pelo autor (2023)

```
pessoa1 = Pessoa('Joseffe', '013 987654321', 34)
print(pessoa1.nome)
print(pessoa1.telefone)
print(pessoa1.idade)
```

Código fonte 2 - Instância de objeto a partir de construtor não padrão
Fonte: Elaborado pelo autor (2023)

1.2. Classe com construtor padrão

Se caso não quisermos popular os atributos, podemos definir valores default (padrão) para os parâmetros do construtor. Com isso, podemos apenas instanciar objetos sem passar nenhuma informação no construtor.

```
class Pessoa:
    def __init__(self, nome=None, telefone=None, idade=0):
        self.nome = nome
        self.telefone = telefone
        self.idade = idade
```

Código fonte 3 - Classe com construtor padrão
Fonte: Elaborado pelo autor (2023)

```
pessoa1 = Pessoa()
pessoa1.nome = input('Digite o seu nome: ')
pessoa1.telefone = input('Digite o seu telefone: ')
pessoa1.idade = int(input('Digite a sua idade: '))

print(pessoa1.nome)
print(pessoa1.telefone)
```

```
print(pessoa1.idade)
```

Código fonte 4 - Instância de objeto a partir de construtor padrão
Fonte: Elaborado pelo autor (2023)

1.3. Métodos

Em todos os métodos devemos adicionar como primeiro parâmetro o self

```
class Pessoa:
    def __init__(self, nome=None, telefone='', idade=0, saldo=0.0):
        self.nome = nome
        self.telefone = telefone
        self.idade = idade
        self.saldo = saldo

    def depositar(self, valor):
        self.saldo += valor

    def sacar(self, valor):
        self.saldo -= valor

    def exibirNomeSaldo(self):
        return ("Nome: " + str(self.nome) + " - Saldo: " +
str(self.saldo))


pessoa1 = Pessoa('Joseffe')
pessoa2 = Pessoa('Natan')


pessoa1.depositar(100)
pessoa2.depositar(50)
```

Código fonte 5 - Métodos da classe
Fonte: Elaborado pelo autor (2023)

1.4. Getters / Setters

Em python, para deixarmos os atributos privados, adicionamos __ na frente deles.

```
class Pessoa:
    def __init__(self, nome=None, telefone='', idade=0, saldo=0.0):
        self.__nome = nome
        self.__telefone = telefone
        self.__idade = idade
        self.__saldo = saldo
```

```
    def get_nome(self):
        return self.__nome

    def set_nome(self, nome):
        self.__nome = nome

    def get_telefone(self):
        return self.__telefone

    def set_telefone(self, telefone):
        self.__telefone = telefone

    def get_idade(self):
        return self.__idade

    def set_idade(self, idade):
        self.__idade = idade

    def get_saldo(self):
        return self.__saldo

pessoa1 = Pessoa()
pessoa1.set_nome('Joseffe')
pessoa1.set_idade(34)

pessoa2 = Pessoa()
pessoa2.set_nome('Natan Cruz')
pessoa2.set_idade(19)

print(pessoa1.get_nome())
print(pessoa1.get_idade())

print(pessoa2.get_nome())
print(pessoa2.get_idade())
```

Código fonte 6- Getters e Setters
Fonte: Elaborado pelo autor (2023)

1.5. Property

Podemos utilizar properties ao invés de getters e setters. Dessa forma o uso de property deixa mais limpo o desenvolvimento.

```
class Pessoa:
```

```
    def __init__(self, nome=None, telefone='', idade=0, saldo=0.0):
        self.__nome = nome
        self.__telefone = telefone
        self.__idade = idade
        self.__saldo = saldo

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

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

    @property
    def telefone(self):
        return self.__telefone

    @telefone.setter
    def telefone(self, telefone):
        self.__telefone = telefone

    @property
    def idade(self):
        return self.__idade

    @idade.setter
    def idade(self, idade):
        self.__idade = idade

    @property
    def saldo(self):
        return self.__saldo

pessoa1 = Pessoa()
pessoa1.nome = "Joseffe"
pessoa1.idade = 34

pessoa2 = Pessoa()
pessoa2.nome = 'Natan'
pessoa2.idade = 19

print(pessoa1.nome)
```

```
print(pessoa1.idade)

print(pessoa2.nome)
print(pessoa2.idade)
```

Código fonte 7 - Propertys
Fonte: Elaborado pelo autor (2023)

1.6. Separando Classes em arquivos diferentes

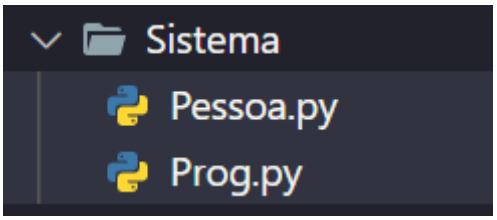

Figura 1 - Estrutura de pastas e arquivos
Fonte: Elaborado pelo autor (2023)

No arquivo Prog.py devemos adicionar o from <NomeArquivo.py> import <NomeClasse>, veja:

```
from Pessoa import Pessoa

pessoa1 = Pessoa()
pessoa1.nome = "Joseffe"
pessoa1.idade = 34

pessoa2 = Pessoa()
pessoa2.nome = 'Natan'
pessoa2.idade = 19

print(pessoa1.nome)
print(pessoa1.idade)

print(pessoa2.nome)
print(pessoa2.idade)
```

Código fonte 8 - Utilização de classe e programa em arquivos separados
Fonte: Elaborado pelo autor (2023)