Curso Python e Django

Francisco André
fandrefh@gmail.com
Senac/PI


GO AHEAD!

Paradigmas de programação

O que é esse tal paradigma?

Podemos conceituar um paradigma como sendo uma visão, ou um ponto de vista, do mundo real (da realidade em que vivemos) e a forma de atuação sobre tal concepção.

Resumindo, é a forma como o analista e o programador lidam com um determinado problema na busca de uma solução em forma de sistema de software.

Classificação dos paradigmas

- ✓ Imperativo;
- ✓ Estruturado;
- ✓ Funcional;
- ✓ Lógico;
- ✓ **Orientado a objetos.**

Orientação a objetos

É um modelo de análise, projeto e programação de sistemas computacionais baseado na composição e na interação entre diversas unidades do software chamadas de *objetos.*

Conceitos essenciais - POO

Classe: representa um conjunto de objetos com características afins. Uma classe define o comportamento dos objetos através de seus métodos, e quais estados ele é capaz de manter através de seus atributos.

Objeto/Instância: um objeto é capaz de armazenar estados através de seus atributos e reagir a mensagens enviadas a ele, assim como se relacionar e enviar mensagens a outros objetos.

Atributo: são características de um objeto. Basicamente a estrutura de dados que vai representar a classe.

Método: definem as habilidades dos objetos.

Mensagem: é uma chamada a um objecto para invocar um de seus métodos, ativando um comportamento descrito por sua classe.

Conceitos essenciais - POO

Herança: (ou generalização) é o mecanismo pelo qual uma classe (sub-classe) pode estender outra classe (super-classe), aproveitando seus comportamentos (métodos) e variáveis (atributos).

Encapsulamento: consiste na separação de aspectos internos e externos de um objeto. Este mecanismo é utilizado amplamente para impedir o acesso direto ao estado de um objeto (seus atributos), disponibilizando externamente os métodos que acessam (getters) e alteram (setters) estes estados.

Orientação a objetos com Python

Para definir a estrutura de uma classe em *Python* usa-se a palavra reservada *class* precedendo o nome da classe.

Python suporta duas formas de criação de classes nomeadas de Old Style e New Style

Orientação a objetos com Python

```
#Old-Style

>>> class Pessoa:
        pass

>>> p = Pessoa()
>>> p
<__main__.Pessoa object at 0x031201F0>
```

Orientação a objetos com Python

```
#New-Style

>>> class Pessoa(object):
        pass

>>> p = Pessoa()
>>> p
<__main__.Pessoa object at 0x031109F0>
```

Orientação a objetos com Python

```
#Atributos em Python

>>> class Pessoa(object):
        nome = ""
        idade = 0

>>> p = Pessoa( )
>>> p.nome = "Francisco André"
>>> p.idade = 34
>>> print(p.nome)
Francisco André
>>> print(p.idade)
34
```

Orientação a objetos com Python

Métodos são funções criadas dentro de uma classe que determinam o comportamento do objeto.

Orientação a objetos com Python

```
#Métodos em Python

class Pessoa(object):
    nome = ""
    idade = 0

    def andar(self):
        print("A pessoa está andando...")

p = Pessoa( )
p.nome = "Francisco André"
p.idade = 34
print(p.nome)
print(p.idade)
p.andar( )
```

Self, Whats????

Todo método do escrito em Python recebe como primeiro parâmetro uma instância do próprio objeto, *self*, significa a si mesmo.

Orientação a objetos com Python

O método especial __*init*__ serve para inicializar variáveis de instância criadas na criação do objeto.

Em outras palavras, o __*init*__ é o método construtor.

Orientação a objetos com Python

```
class Pessoa(object):
    nome = ""
    idade = 0

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

    def andar(self):
        print("A pessoa está andando...")

p = Pessoa("Francisco André")
p.idade = 34
print(p.nome)
print(p.idade)
p.andar()
```

Desafios - POO

1 – Crie um programa que calcule valores reais em dolares;

2 – Converta em pés a altura de uma pessoa;

3 – Calcule a capacidade máxima de pessoas dentro de uma sala;