

Python para Ciência de Dados

Fundamentos



6 de 2022

Fundamentos

Por quê Python?

- Python é inequivocamente uma das três linguagens de programação mais usadas no mundo.
- Motivos principais:
 - É uma linguagem de **propósito geral** e **multi-paradigma**
 - Comunidade grande e engajada em projetos Open Source
 - Apoio institucional do Google, que fomenta vários projetos importantes

Python e você

Algumas características importantes pra se ter em mente enquanto estudamos:

Python é uma linguagem que foi criada para ser utilizada em **qualquer contexto**, de computação científica a desenvolvimento web. Por isso, muitas vezes as opções disponíveis para resolver um problema são do jeito que são pois representam as necessidades de várias pessoas.

Embora seja uma linguagem que comporte muitos paradigmas diferentes de programação, o paradigma predominante é a Programação Orientada a Objetos ou *Object Oriented Programming* (OOB).

O que é um paradigma de programação?

Um programa (ou *algoritmo*) é uma sequência de passos lógicos executáveis por um computador que transformam um input em um output.

Programação é o ato de escrever (ou desenvolver) programas.

Um paradigma de programação é um **jeito de pensar** que orienta o desenvolvimento de um programa.

Boas ideias que orientam o desenvolvimento de vários programas:

- Se fosse possível um programa não deveria ter partes repetidas
- O código deveria ser fácil de ler. Ele não deveria estar inteiro na mesma linha por exemplo

Algumas ideias são bem específicas!

Exemplo: Programando um robô

Ultimamente tem chovido muito no Brasil, obstruindo várias estradas com árvores caídas, carros, lama etc.

Imagine que nós somos capazes de escrever comandos para um robô gigante que pode desobstruir uma ponte.

O fabricante do robô disponibiliza as funcionalidades da máquina em dois formatos, pra termos várias opções na hora de escrever.



Exemplo: Programando um robô (ideia 1)

No sistema do robô tem várias funções simples com ordens que podemos dar ao robô. O código de uma desobstrução poderia ser assim:

```
local_bloqueado = ler_endereco(  
    'endereco_do_bloqueio_da_estrada.csv'  
)  
  
destruir_arvores_caidas(local = local_bloqueado,  
    raio = '1km')  
  
aspirar_lama(local = local_bloqueado,  
    raio = '1km')  
  
secar_estrada(local = local_bloqueado,  
    raio = '1km')
```

O fabricante do robô aqui nos deu comandos que combinam com o paradigma da Programação Funcional: algumas **funções** (`destruir_arvores_caidas`, `aspirar_lama`, `secar_estrada`) fazem o que a gente precisa que seja feito. Elas recebem como input algumas **variáveis**, o endereço até onde o robô precisa ir e o raio em que ele deve atuar.

Exemplo: Programando um robô (ideia 2)

O robô vem equipado com uma arma de raios que o fabricante considerou parecido com os poderes de um Pikachu. Por isso, o fabricante permite que a gente escreva o código que controla o robô de uma outra maneira, fazendo alusão aos poderes quem um Pikachu:

```
import ModoPikachu
# precisamos carregar o 'módulo do pikachu'
# que o fabricante deixou pra nós

RoboPikachu = ModoPikachu.iniciar()
# aqui o robô sai do seu estado natural
# e passa a atuar no 'modo pikachu'

local_bloqueado = ler_endereco(
    'endereco_do_bloqueio_da_estrada.csv'
)

RoboPikachu.choque_do_trovaio(local_bloqueado, raio = '1km')
RoboPikachu.engolir_lama(local_bloqueado, raio = '1km')
```


Exemplo: Programando um robô (ideia 2)

Aqui usamos OOB: tudo que escrevemos fez menção a um **objeto** real, o Robô Pikachu cujo modo foi iniciado no começo do código. Em OOB nós sempre fazemos menção às coisas que um objeto pode fazer, que são chamadas de **métodos**. Nesse paradigma nós interpretamos o nosso código como instruções para um **objeto**, como se o computador fosse ou controlasse uma entidade (pessoa, pokemon etc) com certos poderes específicos.

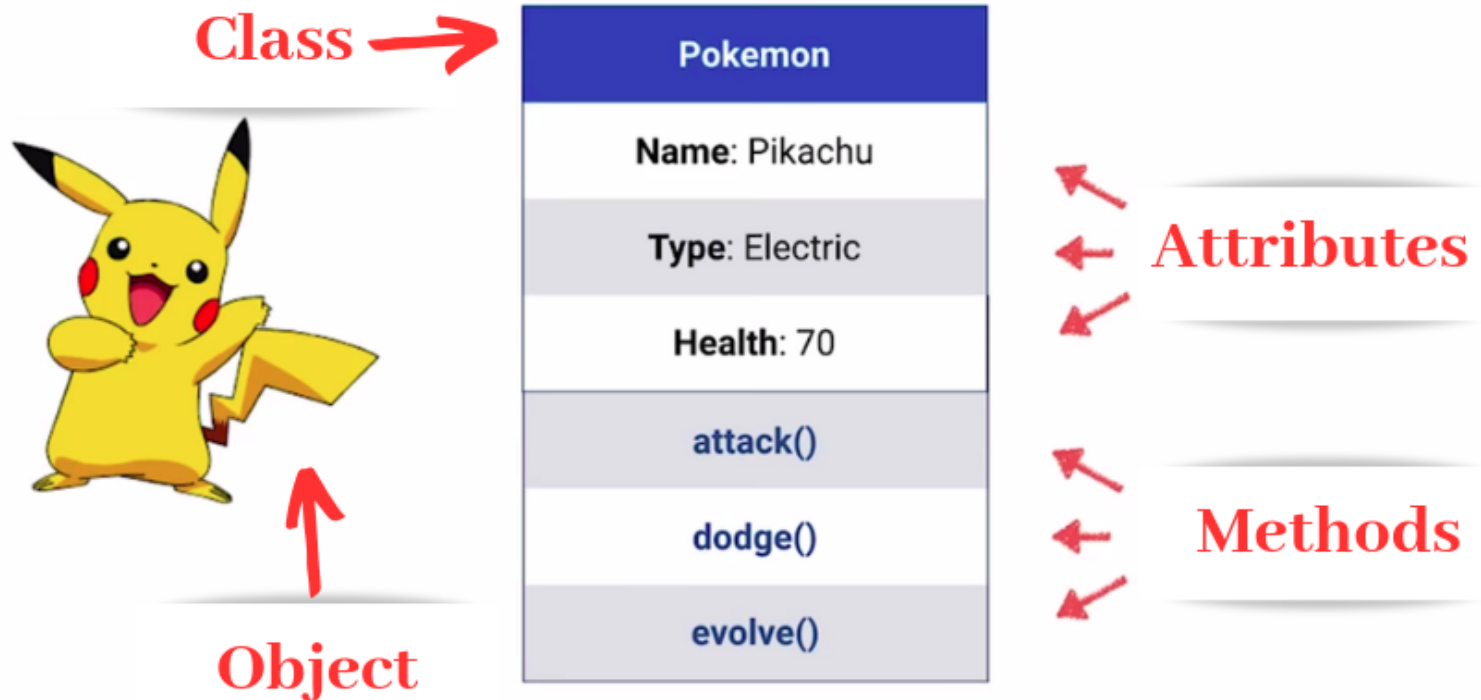
Pode parece uma loucura às vezes, mas para programar coisas dinâmicas como jogos, comunicação entre usuários etc a orientação a objetos é uma ferramenta interessante para expressar comandos para um computador.

Programação Orientada a Objetos

Nesse segundo exemplo surgiram algumas ideias abstratas, mas que serão importantes para darmos nossos passos no aprendizado de Python

- Em Python praticamente tudo que manipulamos são **objetos** que tem certos poderes específicos, que chamamos de **métodos**. Os **métodos** de um objeto são acessados usando o `.`
- Também existem **funções**, que são parecidos com objetos, mas normalmente não podem ser interpretados como "poderes" de um objeto específico, são mais parecidas com "ordens que damos ao computador".
- Além de **métodos**, os objetos também carregam valores próprios, que são chamados de **atributos**

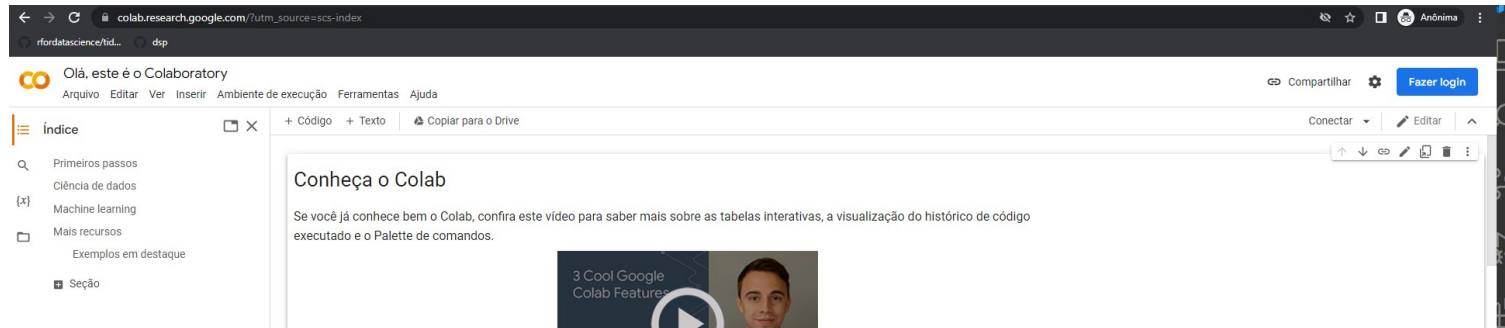
OOB e Pokemons



Mais justificativas: porque pensar em OOB ajuda?

- Aderindo ao paradigma **OOB** com bastante fidelidade, podemos pensar no geral que:
 - Várias coisas que vamos querer fazer com um objeto está nos seus métodos, isso é, como um "poder" que pode ser usado, dentre uma lista de possibilidades.
 - Alterações em um objeto podem ser feitas com relação a várias coisas, os seus atributos. Quão mais complexo for um objeto mais informações (ou atributos) ele carrega em si.
- Vamos ao Python!

Notebook e Google Collab



Se logue com a sua conta do Google

Notebook e Google Collab

Exemplos



Recente

Google Drive

GitHub

Upload

Filtrar notebooks

Título	Aberto pela última vez ▲	Primeiro acesso ▼	
 Olá, este é o Colaboratory	18:27	18:27	

[Novo notebook](#) [Cancelar](#)

Clique em "Novo notebook"