



Paradigmas de linguagens de programação em python

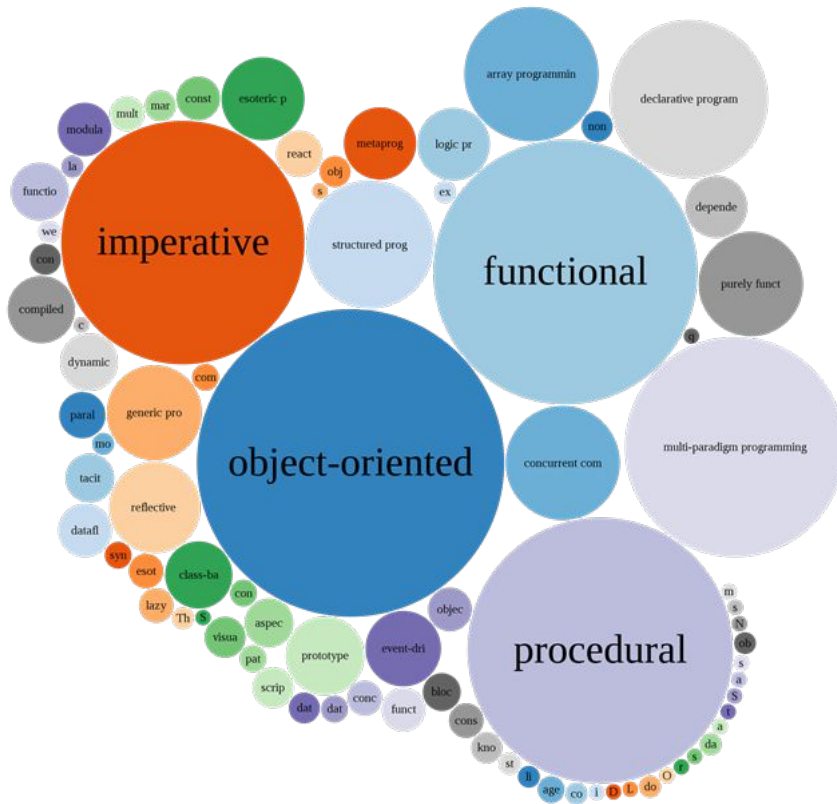
Professores:

Sebastião Rogério feat. Kayo Monteiro

The background is a dark navy blue. It features several horizontal bars of different colors: a light purple bar at the top left, a bright cyan bar at the top center, a dark blue bar below it, a pink bar below that, and a white bar at the top right. On the left side, there is a white bar and a cyan bar. A large, solid pink rectangle is positioned in the center, containing the word 'Resumo' in white. At the bottom, there is a cyan bar on the left and a pink bar on the right.

Resumo

Principaux Paradigmas



Por que aprender sobre paradigmas de programação?

- É um estilo de programação, caracterizado por uma seleção de conceitos
- Fornece a determina a visão que o programador possui sobre a estruturação e execução do programa
- Diferenciam-se pelas técnicas de programação que permitem ou proíbem
- Estão divididos em dois grupos/categorias:
 - Programação Imperativa
 - Programação Declarativa

Programação imperativa

Nesse tipo de construção, as instruções devem ser passadas ao computador na **sequência** em que devem ser executadas. Vários tipos de linguagem de programação suportam esse tipo de paradigma, como **Cobol**, **Fortran** e **Pascal**. O programador instrui a máquina sobre como devem ser computados os processos e como resolver um problema



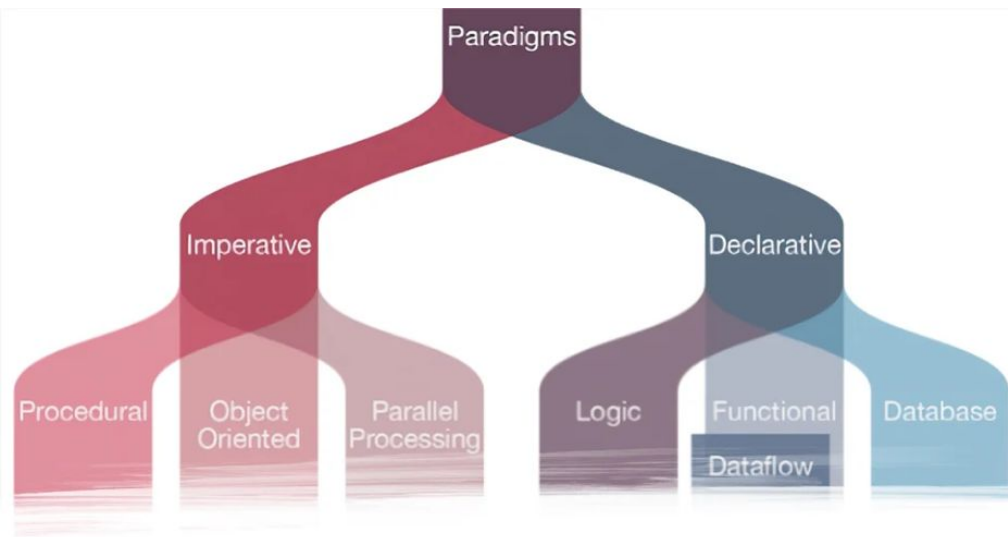
Programação imperativa

- Pontos importantes sobre essa programação:
 - Descreve o passo-a-passo dos procedimentos que a máquina deverá executar (daí o nome procedural);
 - A solução do problema é muito dependente da experiência e criatividade de quem trabalha com a programação;
 - O foco da resolução estará em “como” deve ser feito;
 - Ideal para projetos com poucas mudanças ao longo do tempo;
 - Eficiente e permite uma modelagem semelhante ao mundo real;
 - Bem estabelecido e flexível.

Programação declarativa

Nesse paradigma, há declarações iniciais de **verdades lógicas que são imutáveis**. Assim, depois de realizadas algumas interações entre elas, o resultado encontrado será sempre igual (para as declarações outrora feitas). Isso entra em contraste com a programação imperativa, na qual um mesmo trecho de código pode retornar resultados diferentes.

- Pontos importantes sobre essa programação:
 - Definem o que o programa faz e não como seus processos funcionam;
 - Ênfase nos resultados;




Source: <https://www.watelectronics.com/types-of-programming-languages-with-differences/>



Paradigma Procedural

Nesse tipo de programação, a pessoa passará uma espécie de passo-a-passo dos procedimentos que a máquina deverá executar (daí o nome procedural).

Nesse caso, a solução do problema será muito dependente da experiência e criatividade de quem trabalha com a programação. O foco da resolução estará em “como” deve ser feito. Características: **variáveis, comandos e procedimentos.**



Paradigma Orientado a Objetos

Esse paradigma é bastante conhecido. Foi popularizado na década de 90 com a linguagem de programação **Java**, ao permitir uma programação multiplataforma. Antes disso, não era possível realizar tal tipo de trabalho.

Características: **objetos, métodos e classes.**



Paradigma Funcional

No paradigma de programação funcional, o uso de funções é destaque. O problema é dividido em blocos e, para sua resolução, são implementadas funções que definem variáveis em seu escopo e retornam algum resultado. São exemplos de linguagens suportadas por esse paradigma o **LISP**, o **Scheme** e o **Haskell**.

Paradigma funcional


- Pontos importantes sobre esse paradigma:
 - É bastante indicado quando a solução requerida **é fortemente dependente de uma base matemática**;
 - Subdivide-se o problema proposto e as funções implementadas farão os cálculos matemáticos;
 - Possui alocação de memória automática;
 - Características: **valores, expressões e funções**;



Paradigma lógico

O paradigma lógico é um tanto distinto dos demais paradigmas e deriva do declarativo. Fundamentalmente, utiliza formas de lógica simbólica como padrões de entrada e saída. A partir daí, realiza inferências para produzir os resultados.

Características: **asserções e relações.**



Paradigma orientado a eventos

O paradigma de orientação a eventos é usado por toda linguagem de programação que tem uso de recursos gráficos, como jogos e formulários. Dessa forma, a execução do programa se dá a medida que determinados eventos são disparados pelo usuário. Portanto, quem usa é responsável pelo momento em que o programa é executado.



Variáveis

```
>>> a = 10
>>> a
10
```

```
>>> b = 1.2
>>> b
1.2
```

```
>>> c = "Olá Mundo"
>>> c
'Olá Mundo'
```

Variáveis

- São pequenos espaços de memória, utilizados para armazenar e manipular dados;
- Em Python, os tipos de dados básicos são: tipo inteiro, float e tipo string;
- Cada variável pode armazenar apenas um tipo de dado a cada instante;
- Em Python, diferentemente de outras linguagens de programação, não é preciso declarar de que tipo será cada variável no início do programa.

Variáveis

- A atribuição de valor para uma variável pode ser feita utilizando o comando `input()`, que solicita ao usuário o valor a ser atribuído à variável.



```
>>> nome = input("Entre com o seu nome: ")
Entre com o seu nome: Fulano da Silva
>>> nome
'Fulano da Silva'
```

Variáveis

O comando `input()`, sempre vai retornar uma `string`. Nesse caso, para retornar dados do tipo inteiro ou `float`, é preciso converter o tipo do valor lido. Para isso, utiliza-se o `int (string)` para converter para o tipo inteiro, ou `float (string)` para converter para o tipo `float`.



```
>>> num = int(input("Entre com um numero? :"))  
Entre com um numero? :100  
>>> num  
100
```

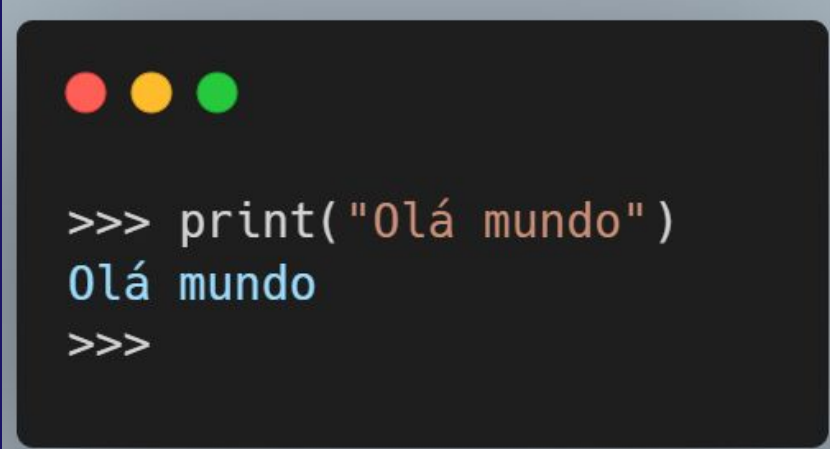
Variáveis

- Em Python, os nomes das variáveis devem ser iniciados com uma letra, mas podem possuir outros tipos de caracteres, como números e símbolos;
- O símbolo sublinha (`_`) também é aceito no início de nomes de variáveis;

Exemplo de nomes Variáveis

Nome	Válido	Comentários
a3	Sim	Embora contenha um número, o nome a3 inicia com letra.
velocidade	Sim	Nome formado com letras.
velocidade90	Sim	Nome formado por letras e números, mas inicia com letras.
salario_médio	Sim	O símbolo (_) é permitido e facilita a leitura de nomes grandes.
salario médio	Não	Nomes de variáveis não podem conter espaços em branco.
_salário	Sim	O sublinha (_) é aceito em nomes de variáveis, mesmo no início.
5A	Não	Nomes de variáveis não podem começar com números

Strings


A terminal window with a dark background and three colored window control buttons (red, yellow, green) at the top left. It displays a Python command and its output.

```
>>> print("Olá mundo")  
Olá mundo  
>>>
```

- Uma string é uma sequência de caracteres simples. Na linguagem Python, as strings são utilizadas com aspas simples ('... ') ou aspas duplas ("...");
- Para exibir uma string, utiliza-se o comando print();

Strings

- Para concatenar strings, utiliza-se o operador +



```
>>> print("Apostila"+"Python")
ApostilaPython
>>> a='Programação'
>>> b='Python'
>>> c=a+b
>>> print(c)
ProgramaçãoPython
```

Strings

Em Python, existem várias funções (métodos) para manipular strings. Na tabela a seguir são apresentados os principais métodos para a manipulação as strings.

Exemplo de manipulação de strings

Método	Descrição	Exemplo
len()	Retorna o tamanho da string.	teste = "Apostila de Python" len(teste) 18
capitalize()	Retorna a string com a primeira letra maiúscula	a = "python" a.capitalize() 'Python'
count()	Informa quantas vezes um caractere (ou uma sequência de caracteres) aparece na string.	b = "Linguagem Python" b.count("n") 2
startswith()	Verifica se uma string inicia com uma determinada sequência.	c = "Python" c.startswith("Py") True

Números

- Os quatro tipos numéricos simples, utilizados em Python, são números inteiros (**int**), números longos (**long**), números decimais (**float**) e números complexos (**complex**);
- A linguagem Python também possui operadores aritméticos, lógicos, de comparação e de bit;

Estrutura de decisão



O que são estruturas de decisão?

Em Python, assim como na maioria das linguagens de programação, o programa deve ser capaz de **tomar decisões** com base em valores e resultados gerados durante sua execução, ou seja, deve ser capaz de decidir se determinada instrução deve ou não ser executada de acordo com uma condição

O que são estruturas de decisão?

Para atender a esse tipo de situação, podemos utilizar instruções especiais denominadas estruturas condicionais ou estruturas de decisão.

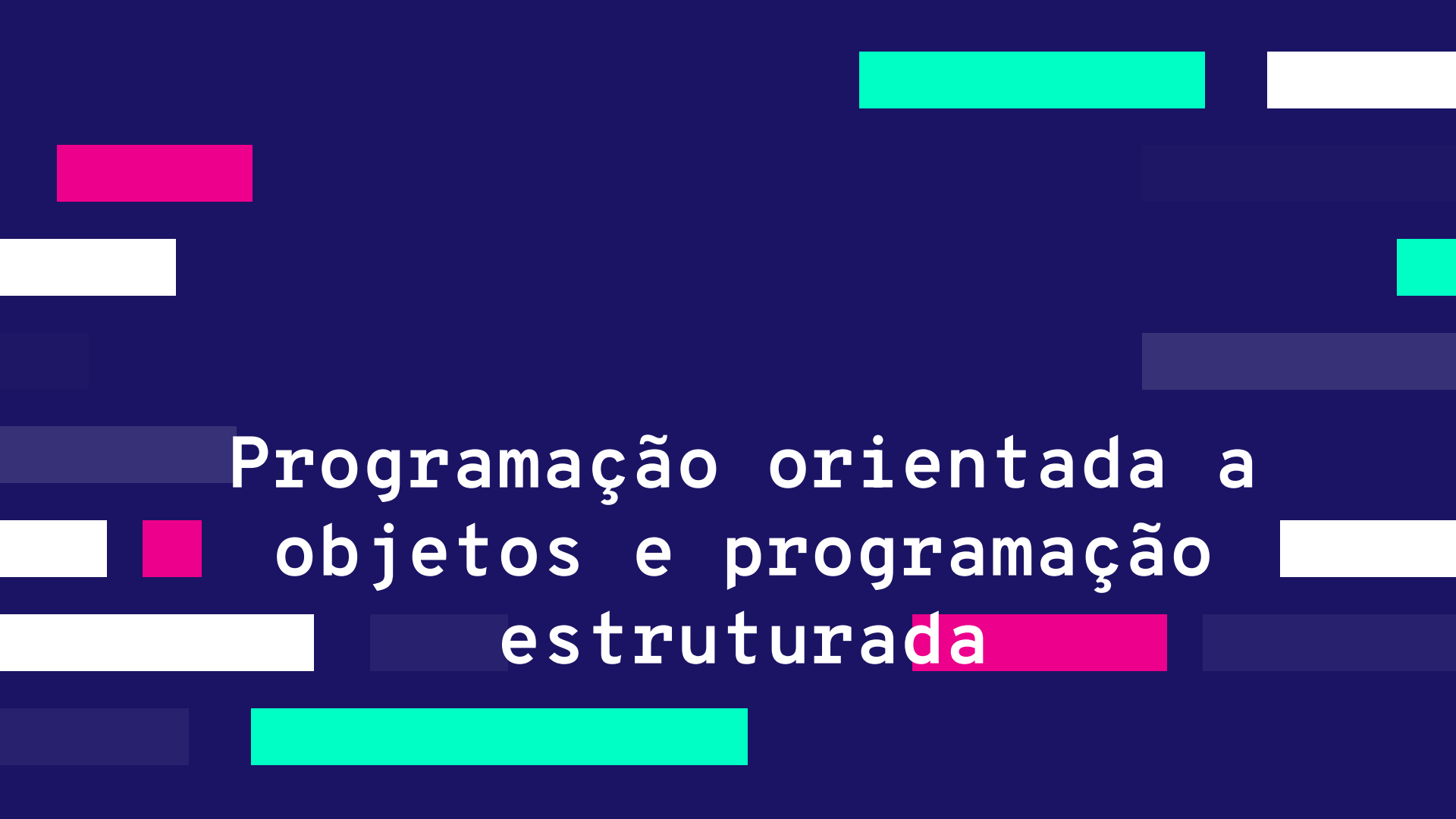
Resumo – Coleção de Dados

Lista: inicializada com `[]`.

Tupla: inicializada com `()`.

Dicionário: inicializada com `{}`.

Possuem uma série de operações equivalentes para acesso, checagem de tamanho, etc



Programação orientada a
objetos e programação
estruturada

Programação orientada a objetos e programação estruturada

Quando começamos a utilizar linguagens como Java, C#, Python e outras que possibilitam o paradigma orientado a objetos, é comum errarmos e aplicarmos a programação estruturada achando que estamos usando recursos da orientação a objetos.



Programação orientada a objetos e programação estruturada

Na programação estruturada, um programa é composto por três tipos básicos de estruturas:

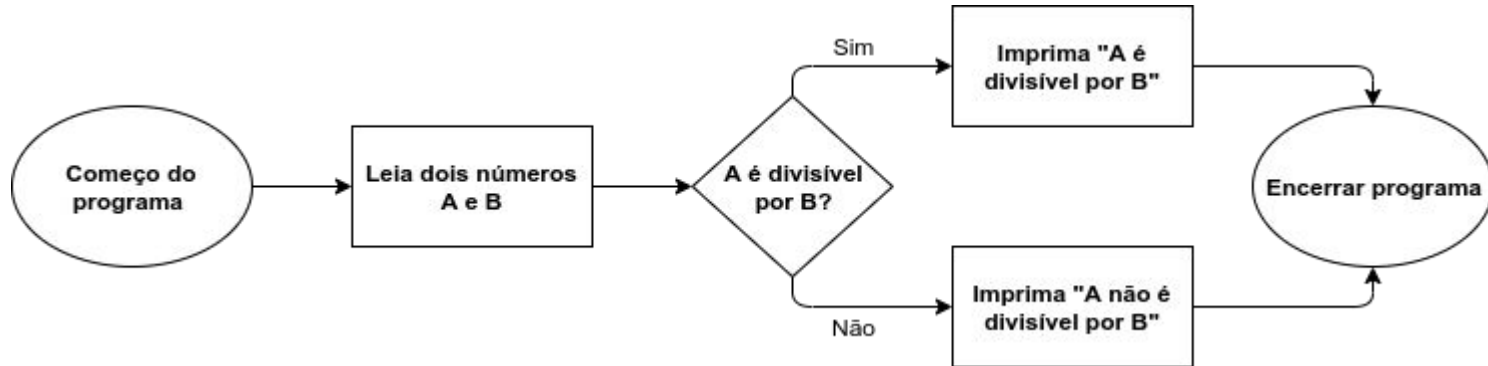
- **sequências:** são os comandos a serem executados
- **condições:** sequências que só devem ser executadas se uma condição for satisfeita (exemplos: if-else, switch e comandos parecidos)
- **repetições:** sequências que devem ser executadas repetidamente até uma condição for satisfeita (for, while, do-while etc)

Programação orientada a objetos e programação estruturada

- Essas estruturas são usadas para processar a entrada do programa, alterando os dados até que a saída esperada seja gerada.
- A diferença principal é que na programação estruturada, um programa é tipicamente escrito em uma única rotina (ou função) podendo, é claro, ser quebrado em subrotinas.

Programação orientada a objetos e programação estruturada

Mas o fluxo do programa continua o mesmo, como se pudéssemos copiar e colar o código das subrotinas diretamente nas rotinas que as chamam, de tal forma que, no final, só haja uma grande rotina que execute todo o programa.



Programação orientada a objetos e programação estruturada

- O acesso às variáveis não possui muitas restrições na programação estruturada.
- Em linguagens fortemente baseadas nesse paradigma, restringir o acesso à uma variável se limita a dizer se ela é visível ou não dentro de uma função (ou módulo, como no uso da palavra-chave static, na linguagem C), mas não se consegue dizer nativamente que uma variável pode ser acessada por apenas algumas rotinas do programa.

Programação orientada a objetos e programação estruturada

- O contorno para situações como essas envolve práticas de programação danosas ao desenvolvimento do sistema, como o uso excessivo de variáveis globais. Vale lembrar que variáveis globais são usadas tipicamente para manter estados no programa, marcando em qual parte dele a execução se encontra.

Programação orientada a objetos

- Surgiu como uma alternativa a essas características da programação estruturada.
- O intuito da sua criação também foi o de aproximar o manuseio das estruturas de um programa ao manuseio das coisas do mundo real, daí o nome "objeto" como uma algo genérico, que pode representar qualquer coisa tangível.

Programação orientada a objetos

- Esse novo paradigma se baseia principalmente em dois conceitos chaves: **classes** e **objetos**.
- Todos os outros conceitos, igualmente importantes, são construídos em cima desses dois.
-

Programação orientada a objetos

A Programação Orientada a Objetos (POO) diz respeito a um padrão de desenvolvimento que é seguido por muitas linguagens. Esse padrão se baseia em quatro pilares que são a base da construção desse Paradigma, são elas:

- Classes e Objetos;
- Encapsulamento.
- Herança.
- Polimorfismo

Referências

- https://www.alura.com.br/artigos/poo-programacao-orientada-a-objetos?gclid=Cj0KCQiwi7CZBhDHARIsAPPWv3dI9zVtxqGDNRmG5KQK3T77_qXpidIWCKbkMxv-IFcqchAdwAn3WdUaAlctEALw_wcB
- <https://www.youtube.com/watch?v=jpu11qrluoU>
- <https://www.devmedia.com.br/como-criar-minha-primeira-classe-em-python/38912>

To be...



CREDITS: This presentation template was created by Slidesgo, including icons by Flaticon, and infographics & images by Freepik.