



# AceleraDev Loadsmart Women Edition

## Módulo 2

**Camila Maia**

**Esta apresentação está disponível em:**

<https://github.com/camilamaia/acelera-dev-loadsmart-women/>



# Módulos

## Python: Noções básicas

Primeiro programa, teste, estrutura de dados, condicionais, repetições, operações, funções, classes, objetos...

1

2

3

4

5

**Django I:** instalação, iniciando um projeto, arquitetura, modelos, views, django admin, URLs

**REST APIs:** O protocolo HTTP, Rest APIs e Django REST Framework

## Um pouco mais de Python

Exceções, decorators, list e dict comprehensions, map, reduce, filter...

**Django II:** Templates, HTML e CSS básico, Forms, Autorização e Autenticação.



## Top 3 Oscar Movies Exercise

- python standard lib
- pip install
- Requirements.txt



# Modules and projects management

- A module is a file containing Python definitions and statements. The file name is the module name with the suffix `.py` appended.



# Top 3 Oscar Movies Exercise

-

## With modules

### Pros

- Uma implementação apenas para cada ação

### Cons

- Muitos parâmetros repetidos
- Se precisar mudar algo em um parâmetro, tem que mudar em vários locais



# Top 3 Oscar Movies Exercise

-

## With modules By movie

### Pros

- Métodos sem precisar passar parâmetros

### Cons

- Muitos arquivos
- Difícil de fazer alterações - muitos arquivos diferentes para alterar



# Paradigmas de Progração

- É o modo como o programador enxerga o mundo e traduz os problemas reais para código.
- É um **padrão** de raciocínio para resolução de problemas.
- A maioria das linguagens podem suportar mais de um paradigma de programação.
- Cabe ao programador escolher qual paradigma se encaixa melhor com o problema a ser resolvido
- Cada problema tem um paradigma que é o melhor para descrevê-lo.





# Paradigmas de Progração

- Programação Procedural
  - O programador vê o mundo como um conjunto de instruções a serem seguidas
- Programação Funcional
  - O programador vê o mundo como funções matemáticas, com entradas e saídas



# Orientação à Objetos

- É um paradigma de programação
- *Object-oriented programming is a software programming model constructed around **objects**. This model compartmentalizes data into objects (data fields) and describes object contents and behavior through the declaration of classes (**methods**).*



# Classes e Objetos

- Uma classe é uma estrutura que abstrai um conjunto de objetos com características similares.
- Uma classe define o comportamento de seus objetos - através de métodos - e os estados possíveis destes objetos - através de atributos.
- Camelcase



# Top 3 Oscar Movies Exercise - With Class

- Facilidade em reutilizar código
- A classe armazenada dados - menos repetição, mais fácil de fazer alterações



# **Instância X Classe: métodos e variáveis**

- Variáveis de Classe
- Variáveis de Instância
- Métodos de Classe
- Métodos de Instância



# Herança

- Como herdar de uma classe
- Método Super
- Sobrescrever métodos



# Exceções

- This type of error occurs whenever syntactically correct Python code results in an error.

## Erro Sintático

Python

```
>>> print( 0 / 0 )  
File "<stdin>", line 1  
    print( 0 / 0 )  
           ^  
SyntaxError: invalid syntax
```

## Exceção

Python

```
>>> print( 0 / 0 )  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
ZeroDivisionError: integer division or modulo by zero
```



# Programação Funcional

- É um paradigma de programação.
- É baseado em funções matemáticas e presa por imutabilidade de dados e de estados.
- Programação funcional **não** é o oposto de orientação a objetos.
- A execução de uma função deve retornar sempre o mesmo valor : imutabilidade.





# Dict and Dict Comprehensions

## Dicionários

- Conjunto de chave e valor
- Imagine a dictionary in the real world... when you need to look up the meaning of a word, you try to find the meaning using the word itself and not the possible index of the word
- Get Keys and Get values

## Dictionary Comprehension



# Lambda Functions

- Função anônima
- Sintaxe
  - **lambda** arguments: expression



# Map, Filter e Reduce

- These are three functions which facilitate a functional approach to programming.
- Map
  - Map applies a function to all the items in an input\_list
  - List => List
  - `map(function_to_apply, list_of_inputs)`
- Filter
  - Creates a list of elements for which a function returns true.
  - List => List (de tamanho igual ou menor que a primeira lista)
  - `filter(function, iterable)`
- Reduce
  - Perform some computation on a list and return a result



# Map, Filter e Reduce



Steven Luscher  
@steveluscher



Map/filter/reduce in a tweet:

```
map([🌽, 🐮, 🐔], cook)
```

```
=> [🍿, 🍔, 🍳]
```

```
filter([🍿, 🍔, 🍳], isVegetarian)
```

```
=> [🍿, 🍳]
```

```
reduce([🍿, 🍳], eat)
```

```
=> 💩
```



## Desafio Extra

- Fazer um **fork** do repositório [accelera-dev-loadsmart-women](https://github.com/accelera-dev/loadsmart-women)
- Criar um **branch** chamado “mod2\_tests”
- Implementar a parte faltante, marcada com o comentário #TODO da pasta 11\_with\_tests
- Quando terminado, criar um **Pull Request** (PR) para o seu fork do seu novo branch.
- Me adicionar como reviewer no **PR**





**Dúvidas?**

# Conhecimento Passado

Legal, mas  
não  
aprendi  
nada  
novo, não.



Aprendi  
muita  
coisa  
nova!

Conhecimento: 0-10



# Velocidade

5: Velocidade  
ideal.

ZzzZzzz,  
pode  
acelerar  
isso aí.



Muito  
rápido, tô  
assimilando  
o primeiro  
slide ainda.

Velocidade: 0-10





# Conteúdos

- [Programming Paradigms for Dummies: What Every Programmer Should Know](#)
- [Paradigmas de Programação - Imperativo, Orientado a Objetos e Funcional](#)
- [Object-Oriented Programming \(OOP\) in Python 3](#)
- [How to Think Like a Computer Scientist](#)
- <https://docs.python.org/3/tutorial/modules.html>
- [https://www.tutorialspoint.com/python3/python\\_modules.htm](https://www.tutorialspoint.com/python3/python_modules.htm)
- [Python Exceptions: An Introduction](#)
- [Começando com Programação Funcional](#)
- [Functional Programming HOWTO](#)



# Conteúdos

- [Python Dictionary Comprehension Tutorial](#)
- [Python Anonymous/Lambda Function](#)
- [Lambda, filter, reduce and map](#)
- [Python 3 Generators](#)
- [Python Wiki - Generators](#)
- [Python3 Decorators](#)
- [Python with Context Managers](#)
- [Python Errors and Exceptions - Official Doc](#)
- <https://docs.python.org/3/howto/functional.html>





**MUITO OBRIGADA!**