

# **A linguagem de programação Python**

Módulo II e III



# Introdução

- Criada por Guido Van Rossum em 1991
- Linguagem de altíssimo nível
- Orientada a objeto
- Tipagem dinâmica e forte
- Linguagem interpretada e interativa

“Python é uma linguagem de programação poderosa e fácil de aprender. Ela possui estruturas de dados de alto nível e uma simples mas eficiente abordagem da programação orientada a objetos. Sua elegante sintaxe e tipagem dinâmica juntamente com seu interpretador nativo fazem dela a linguagem ideal para scripting e desenvolvimento rápido de aplicações em diversas áreas sob várias plataformas.”

– Guido Van Rossum

# Por que Python?

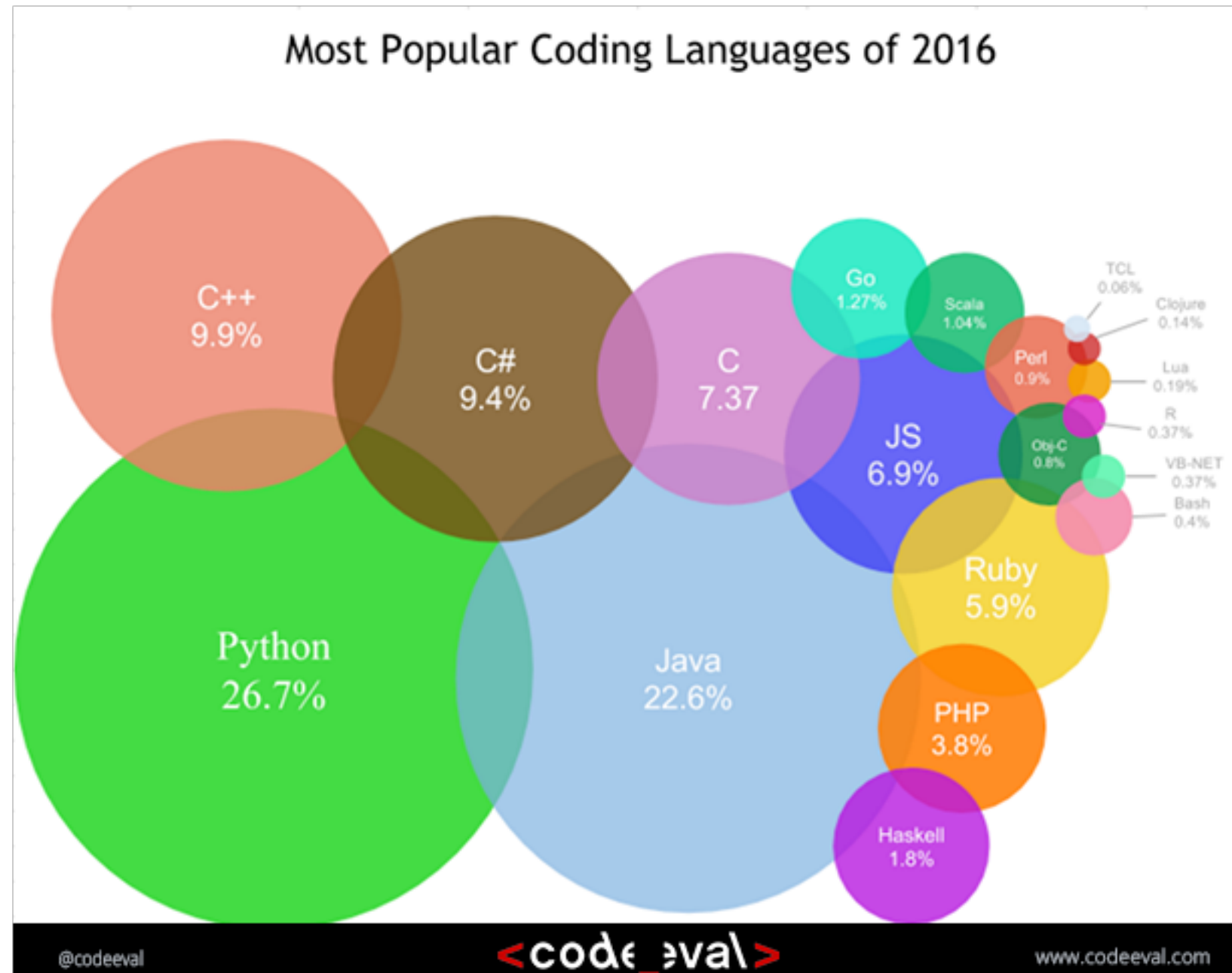
- Linguagem produtiva
- Sem ; e {}
- Organização e legibilidade



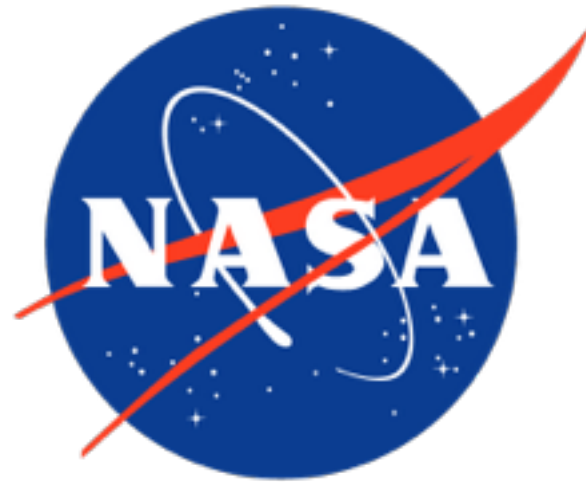
# Cuidado com as cantadas...



# Por que Python?



# Quem usa Python?





# Zen do Python

## **O Zen do Python, por Tim Peters**

Bonito é melhor que feio.

Explícito é melhor que implícito.

Simples é melhor que complexo.

Complexo é melhor que complicado.

Linear é melhor do que aninhado.

Esparso é melhor que denso.

Legibilidade conta.

# Zen do Python

Casos especiais não são especiais o bastante para quebrar as regras.

Ainda que praticidade vença a pureza.

Erros nunca devem passar silenciosamente.

A menos que sejam explicitamente silenciados.

Diante da ambigüidade, recuse a tentação de adivinhar.

Deveria haver um — e preferencialmente só um — modo óbvio para fazer algo.

Embora esse modo possa não ser óbvio a princípio a menos que você seja holandês.

Agora é melhor que nunca.

# Zen do Python

Embora nunca freqüentemente seja melhor que \*já\*.

Se a implementação é difícil de explicar, é uma má idéia.

Se a implementação é fácil de explicar, pode ser uma boa idéia.

Namespaces são uma grande idéia — vamos ter mais dessas!

# Interpretador Interativo

- Vá até o terminal e digite “python”, seja bem-vindo ao modo interativo!

# Tipagem dinâmica e forte

- Você não precisa declarar o tipo da variável

```
>>> variavel = "Estamos no mini curso vendo exemplos de tipagem dinamica"
>>> type(variavel)
<class 'str'>
>>> variavel_int = 123456
>>> type(variavel_int)
<class 'int'>
>>> variavel_double = 1.2
>>> type(variavel_double)
<class 'float'>
>>> variavel_bool = True
>>> type(variavel_bool)
<class 'bool'>
>>> variavel = 1
>>> type(variavel)
<class 'int'>
```

# Tipagem dinâmica e forte

- Você não pode somar inteiros com strings

```
>>> nome = "Leo"
>>> int = 20
>>> nome - int
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unsupported operand type(s) for -: 'str' and 'int'
```



# Indentação

```
>>> if (20 < 40):  
...     print("Caramba, 20 é menor que 40 mesmo.")  
...  
Caramba, 20 é menor que 40 mesmo.
```

```
>>> if (20 < 40):  
...     print("Caramba, 20 é menor que 40 mesmo.")  
File "<input>", line 3  
    print("Caramba, 20 é menor que 40 mesmo.")  
    ^
```

**IndentationError:** expected an indented block

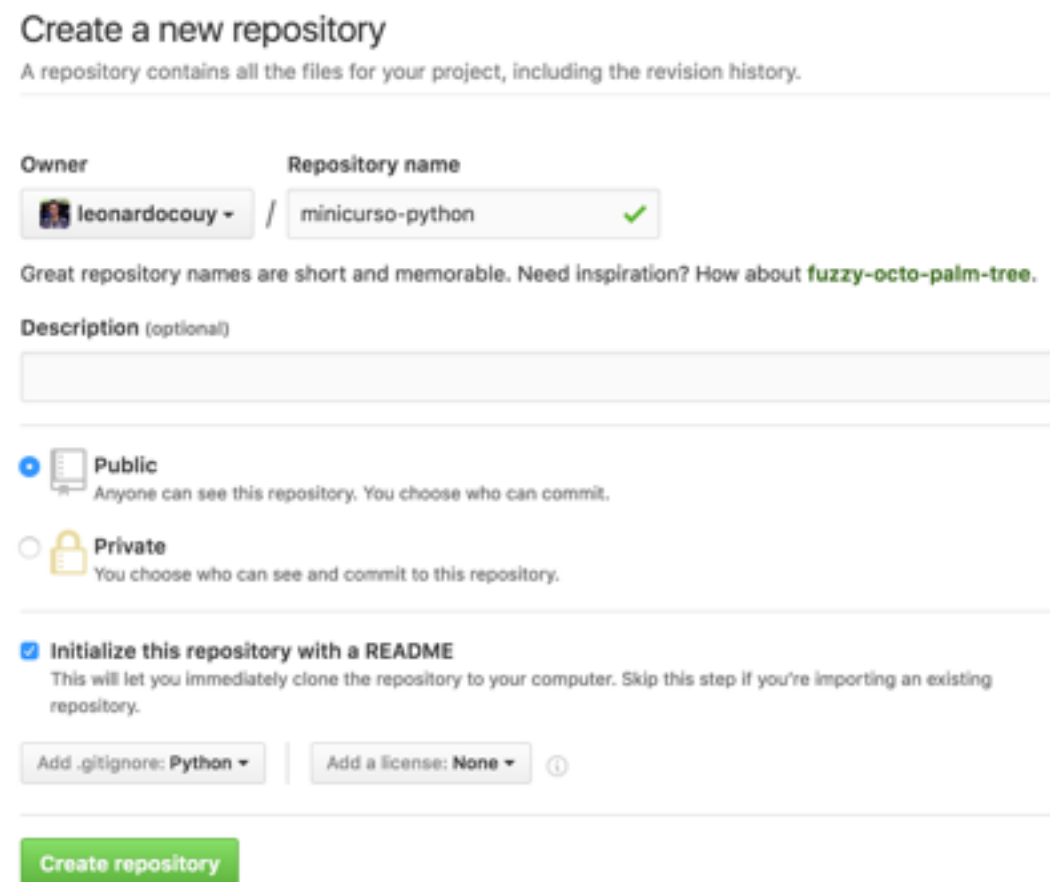
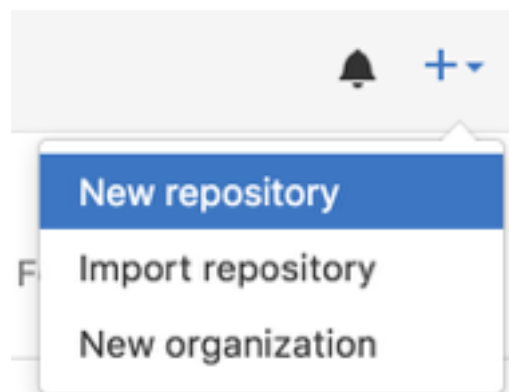
# Conhecendo o ambiente de desenvolvimento Python



PyCharm

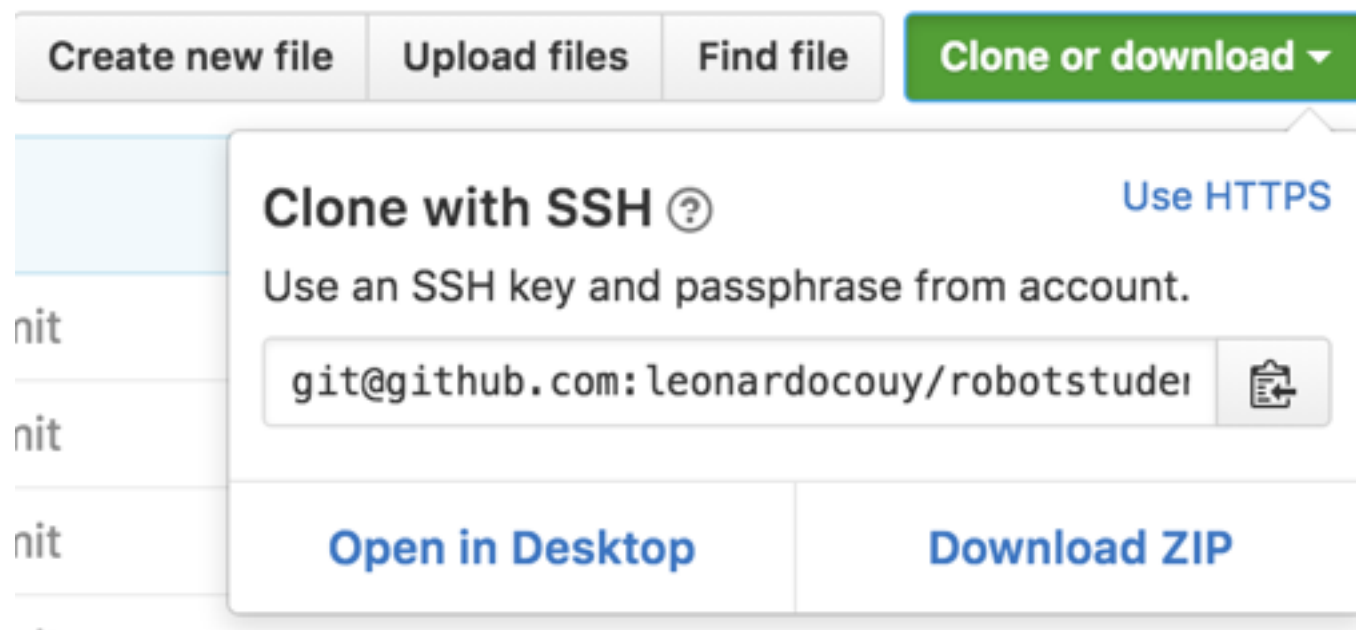
# Criando nosso projeto de exercício

- Acesse o Github e inicie um novo projeto

A screenshot of the 'Create a new repository' form on GitHub. The form has the title 'Create a new repository' and a subtitle 'A repository contains all the files for your project, including the revision history.' Below this, there are two input fields: 'Owner' with the value 'leonardocouy' and 'Repository name' with the value 'minicurso-python', which has a green checkmark. A note says 'Great repository names are short and memorable. Need inspiration? How about fuzzy-octo-palm-tree.' There is a 'Description (optional)' text area. Below that, there are two radio button options: 'Public' (selected) and 'Private'. The 'Public' option has a description: 'Anyone can see this repository. You choose who can commit.' The 'Private' option has a description: 'You choose who can see and commit to this repository.' There is a checked checkbox for 'Initialize this repository with a README' and a description: 'This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.' At the bottom, there are two dropdown menus: 'Add .gitignore: Python' and 'Add a license: None'. A green 'Create repository' button is at the bottom.

# Criando nosso projeto de exercício

- Clone o repositório



# Criando nosso projeto de exercício

- Digite no terminal:
  - `git clone LINK`

# Criando nosso projeto de exercício

- Selecione o projeto no PyCharm
- Certifique-se que o Interpreter esteja correto
- Arrumar o .gitignore
- Arrumar o .readme
- Enviar para o Git e para o Github



# Como aproveitar melhor nossas aulas?

- Entender
- Praticar junto na aula (Assim que eu explicar)
- Testar
- Enviar pro Github
- Praticar em casa (Imagine projetos reais)
- Manda pro github tbm!