



PYTHON
ACADEMY

APRENDA GIT E SEUS PRINCIPAIS COMANDOS

Nesse ebook você vai dominar o Git: uma ferramenta MUITO IMPORTANTE que todo desenvolvedor DEVE dominar!

PYTHONACADEMY.COM.BR

Gere ebooks como este com



em <https://ebookr.ai>

Crie ebooks profissionais incríveis em minutos com IA



Chega de formatar texto no Google Docs, Word ou ferramentas que só te fazem perder tempo...

E deixe que nossa IA faça o trabalho pesado!



Capas gerados por IA



Infográficos feitos por IA



Edite em Markdown em Tempo Real



Adicione Banners Promocionais

TESTE AGORA

PRIMEIRO CAPÍTULO 100% GRÁTIS

Salve salve Pythonista!

Nesse post você vai dominar uma ferramenta essencial na vida de todo desenvolvedor de software: o **Git**!

Aqui você verá:

- Quando e quem criou o **git**,
- As diferenças entre o **Git** e o **Github** (muitas pessoas confundem os dois),
- Como instalá-lo em sua máquina,
- Seus principais comandos (como configurá-lo com `git config`, o que é e como criar uma *branch* com `git branch`, e muito mais!)

Então vamos nessa!

Introdução

Git é uma **ferramenta de linha de comando** (que também possui interfaces visuais).

Foi criada em 2005 pelo nosso querido **Linus Torvalds** para auxiliar o desenvolvimento do kernel do Linux e é mantida atualmente por Junio Hamano.



Git é um software instalado localmente e não possui gerenciamento de usuários.

Git foca em resolver o problema do versionamento, compartilhamento e desenvolvimento compartilhado de códigos-fonte.

Diferenças entre Git e Github

Primeiro é importante salientar: Git e GitHub **não são a mesma coisa**.



Git é tudo isso que falamos aqui em cima 🙌

Já o GitHub é uma empresa fundada em 2008 que faz ferramentas que se **integram** ao git.

Você não precisa do GitHub para usar o git, mas não pode usar o GitHub sem usar o git.

Existem muitas outras alternativas ao GitHub, como GitLab, BitBucket e soluções gerenciadas localmente (geralmente em empresas que possuem códigos sensíveis e confidenciais).

Todos eles são referidos no “vocabulário git” como “remotos” e todos são completamente opcionais.

Você não precisa usar um controle remoto para usar o git, mas facilitará o compartilhamento de seu código com outras pessoas 😊

Nomenclaturas importantes

Antes de começarmos, é importante se acostumar com a **nomenclatura do Git**:

- **Repositório**: É onde seu código “vive”. Um repositório pode ser visto como um projeto dentro do **Git**, ou seja: cada repositório tem suas próprias dependências, suas bibliotecas, seu código, sua linguagem de programação e etc.
- **Branch**: É um conceito central do Git! Todo repositório inicia com uma *branch* (do inglês **tronco** - como se fosse um tronco de uma árvore mesmo) chamada `main` (antiga `master`). A partir desse “entroncamento” principal, outras *branches* são criadas para dar vida à outras funcionalidades e alterações no código.
- **Commit**: É uma alteração no código. Possui um **hash** - que é um identificador único, um **autor** (o criador do *commit*) e uma **mensagem** que diz o que foi feito naquele *commit*, por exemplo: `"Altera cor do botão de Login"`.
- **Servidor Remoto**: É o servidor central que vai disponibilizar o código para outros desenvolvedores. Assim como foi dito ali em cima, existem diversas plataformas, como Github, Gitlab e BitBucket.



Estou construindo o [Ebookr.ai](https://ebookr.ai), uma plataforma onde você cria ebooks profissionais com IA sobre qualquer assunto — do zero ao PDF pronto, com capas e infográficos gerados automaticamente. Dá uma olhada!


Crie Ebooks profissionais incríveis em minutos com IA



Chega de formatar texto no Google Docs, Word ou ferramentas que só te fazem perder tempo...

... e deixe que nossa IA faça o trabalho pesado!

TESTE AGORA! PRIMEIRO CAPÍTULO 100% GRÁTIS 

 Capas gerados por IA

 Adicione Banners Promocionais

 Edite em Markdown em Tempo Real

 Infográficos feitos por IA

Agora que já sabe o que é Git, suas principais diferenças para o Github e seu vocabulário, vamos **instalá-lo!**

Instalação do Git

Para fazer a instalação, visite o [site oficial do Git clicando aqui](#) e faça o download específico para seu Sistema Operacional.

Após sua instalação, vamos, finalmente, para seus **principais comandos!**

 **Quer aprender de verdade?**

Então execute cada comando aqui embaixo e vá aprendendo junto!

Principais comandos do Git

Vamos começar pelo começo de tudo, pelo comando que utilizamos para configurar algumas informações muito importantes: o `git config`!

Comando `git config`

Este comando é utilizado para configura o nome do autor e endereço de email que serão usados nos seus commits.

```
git config --global user.name "Nome"
git config --global user.email voce@email.com
```

Comando `git init`

Comando utilizado para criar novos repositórios.

Primeiro vá para a pasta raiz do projeto que será o novo repositório e execute o seguinte comando.

```
git init .
```

Com isso, será criada uma pasta `.git` na raiz do repositório que serve para gerenciamento do próprio Git.

Dica: **NÃO MEXA NELA!** 😊

Comando `git clone`

Cria uma cópia local de um repositório remoto.


```
git clone {caminho_repositorio}
```

Exemplo, a Python Academy tem um [repositório no Github com várias dicas tops de projetos](#).

Você pode clonar um projeto localmente para ver o código, executar e incrementar através do seguinte comando:

```
git clone https://github.com/PythonAcademyBrasil/Dicas.git
```

Dessa forma, você terá uma cópia do código localmente para brincar! 🥰

Comando `git add`

Adiciona um ou mais arquivos modificados localmente à área de *staging* (Index) para serem incluídos no *commit*.

```
git add {nome arquivo}  
git add *
```

Ah, esse assunto (área de *staging*, *index* e mais) será tratado no próximo Post, sobre o “**Fluxo de Trabalho no Git**”.

Comando `git commit`

Confirma as mudanças localmente (ainda não altera o servidor remoto):

```
git commit -m "Mensagem do commit"
```

Comando `git push`

Agora sim, envia as alterações feitas (*commits*) em uma *branch* específica ou em todas as branches (parâmetro `--all`) para o servidor remoto.

```
git push {nome remoto} {branch}
git push --all
```

Comando `git status`

Mostra o estado da sua *branch* atual, lista os arquivos que você já alterou e aqueles que precisam ser adicionados à um commit.

```
git status
```

Veja um exemplo:

```
[viniciusramos] (~/Development/pythonacademy/pythonacademy.github.io/assets) (master)
$ git status
On branch master
Your branch is ahead of 'origin/master' by 3 commits.
  (use "git push" to publish your local commits)

Untracked files:
  (use "git add <file>..." to include in what will be committed)

        posts/git-e-comandos/

nothing added to commit but untracked files present (use "git add" to track)
```

Comando `git branch`

Comando utilizado para gerenciamento de *branches*. Com ele é possível listar, criar ou deletar branches.

- Criar uma nova *branch*: `git branch {nome da branch}`
- Listar *branches*: `git branch`

- Deletar uma *branch*: `git branch -D {nome da branch}`

Comando `git checkout`

Comando utilizado para trocar entre as *branches* do seu projeto.

Se utilizado o parâmetro `-b` cria uma nova branch e já troca sua branch atual para esta.

```
git checkout {branch}
git checkout -b {nova branch}
```

Comando `git merge`

Combina as mudanças de uma branch em outra.

```
git merge {branch de origem}
```

Esse comando vai trazer todas as alterações de outra *branch* para sua *branch* atual.

Comando `git log`

Traz o histórico de alterações da *branch* atual.

```
git log
```

BÔNUS: Comando `git log` bombadão 💪

Traz o histórico de alterações da *branch* atual com uma visualização muito mais útil!

```
git log --decorate --oneline --all --graph
```

Veja a saída:

```
* 4205c7b (origin/master, origin/HEAD) Atualiza página da Jornada Python
* c0c1e0e Adiciona imagem PNG do post de funções
* 6442d6c Adiciona assets do post de Funções
* edd90a6 Adiciona banner do cupom
* 9ecf9b0 Alteração de datas
* fed39db Pequena correção
* 140635c Altera assets
* 413e1bd Altera módulos
* e201589 Atualiza assets
```

Cada parâmetro altera a saída do comando. Veja cada significado:

- `--decorate` : Altera a saída, decorando-a para ficar visualmente mais bem organizada.
- `--oneline` : Um commit por linha
- `--all` : Traz todos os *commits* de todas as *branches*
- `--graph` : Estiliza a saída em formato de grafo.

Para saber todos os parâmetros do comando basta usar `git [COMANDO] --help`, por exemplo: `git log --help` 😊

Conclusão

Nesse post você aprendeu como utilizar os principais comandos do Git!

Saber utilizá-lo é crucial para sua carreira de Pythonista de sucesso!

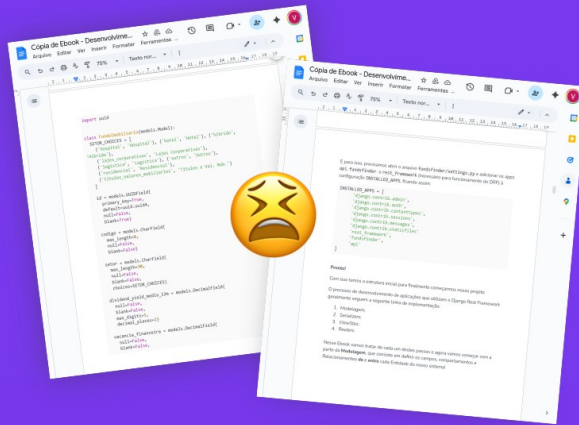
Se ficou com alguma dúvida, fique à vontade para deixar um comentário no box aqui embaixo! Será um prazer te responder! 😊

Não se esqueça de conferir!

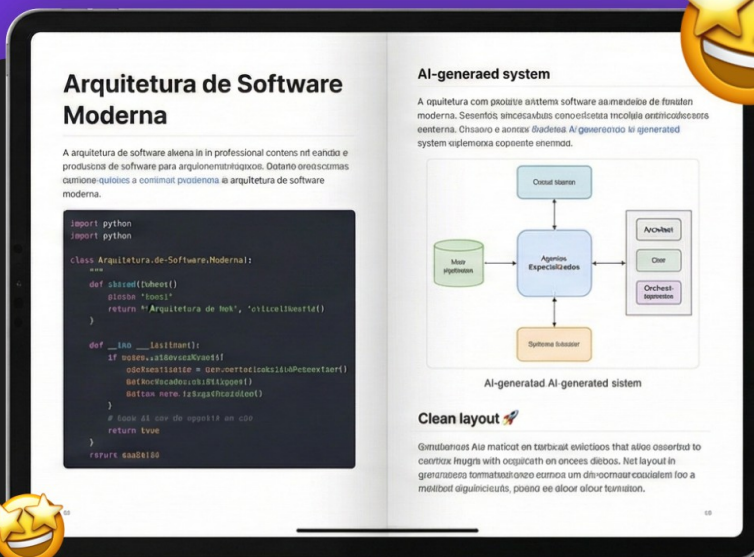


Ebookr

Crie Ebooks profissionais em minutos com IA



Chega de formatar código no Google Docs ou Word



Capas gerados por IA



Infográficos feitos por IA



Adicione Banners Promocionais

Deixe que nossa IA faça o trabalho pesado



Edite em Markdown em Tempo Real

TESTE AGORA



PRIMEIRO CAPÍTULO 100% GRÁTIS