

Data Mining com Python

Victor Machado da Silva, MSc
victor.silva@professores.ibmec.edu.br

Índice

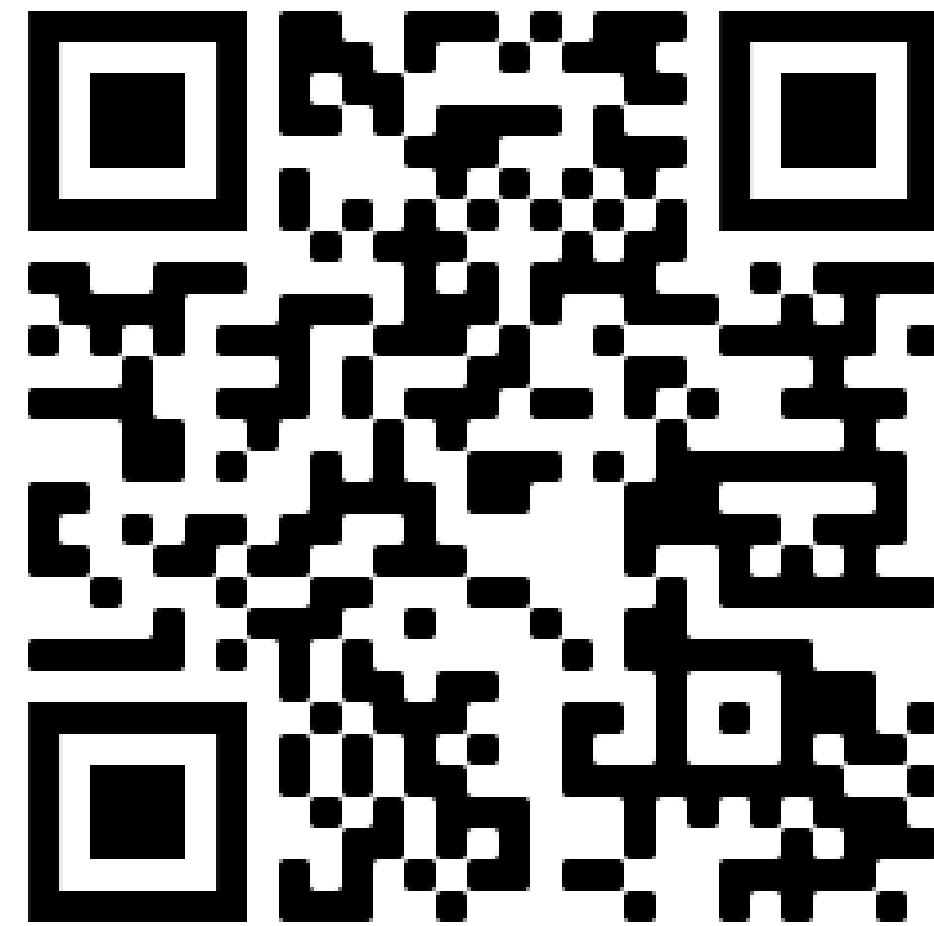
- [Apresentação do curso](#)
- [Por que Python?](#)
- [Configurando o ambiente Python](#)
- [Instalando e configurando IDEs](#)
- [Utilizando o PyCharm](#)
- [Utilizando o VSCode](#)
- [Configurando o pylint](#)
- [Instalando os pacotes via PyPI](#)
- [Ciência de Dados – Conceitos Iniciais](#)
- [Git](#)



Apresentação do curso

Apresentação do curso

- Contato: victor.silva@professores.ibmec.edu.br
- Aulas às terças-feiras, de 18:30 às 22:50
- Grupo no Whatsapp: <https://chat.whatsapp.com/JK842UrgzOHAXwUOPA9B01>
- Material no GitHub: <https://github.com/victor0machado/2021.1-datamining>



Apresentação do curso

Aula	Dia	Dia sem.	Tópico
01	23/02/2021	ter	Python: parte 1
02	02/03/2021	ter	Python: parte 2
03	09/03/2021	ter	Python: parte 3
04	16/03/2021	ter	Conceitos iniciais de mineração de dados
05	23/03/2021	ter	Extração de dados
06	30/03/2021	ter	Transformação e visualização de dados
07	06/04/2021	ter	Dúvidas / Atividade em grupo
08	13/04/2021	ter	SEM AULA (SEMANA AP1)
09	20/04/2021	ter	K-vizinhos próximos e naive Bayes
10	27/04/2021	ter	Regressão (linear simples/múltipla e logística)
11	04/05/2021	ter	Árvores de decisão e agrupamento
12	11/05/2021	ter	Processamento de linguagem natural e sistemas de recomendação
13	18/05/2021	ter	Máquinas de vetores de suporte e regressão Random Forest
14	25/05/2021	ter	Fazendo benchmarking de modelos de mineração de dados
15	01/06/2021	ter	Deploy de um modelo e uso de git
16	08/06/2021	ter	Redes Neurais
17	15/06/2021	ter	Dúvidas / Atividade em grupo
18	22/06/2021	ter	Dúvidas / Atividade em grupo
19	29/06/2021	ter	SEM AULA (SEMANA AP2)
20	06/07/2021	ter	SEM AULA (SEMANA AS)

Apresentação do curso

Avaliação

- Proporção:
 - Trabalhos: 20%
 - Projeto (P1): 40%
 - Projeto (P2): 40%
- Detalhes das entregas:
 - Trabalhos são individuais
 - Projetos de P1 e P2 em grupos de no mínimo 2 e no máximo 3 pessoas
- AS será uma prova com consulta, que substituirá a menor nota entre P1 e P2.

Sugestões de materiais para estudo

- Python é uma linguagem intuitiva para o aprendizado, porém é importante termos à mão livros, apostilas e outros materiais para auxiliar os estudos. Abaixo encontram-se algumas sugestões:
 - Documentação oficial em Python: <https://docs.python.org/pt-br/3/index.html>
 - Joel Grus - *Data Science do Zero: Primeiras regras com o Python* (Alta Books)
 - Raul S. Wazlawick - *Introdução a Algoritmos e Programação com Python* (Elsevier)
 - Sérgio Luiz Banin - *Python 3 - Conceitos e Aplicações - Uma abordagem didática* (disponível no Minha Biblioteca!)
 - Stack Overflow: <https://stackoverflow.com/questions/tagged/python>
 - Artigos no Medium.com: <https://medium.com/search?q=python>
 - Leandro A. da Silva et al – *Introdução à Mineração de Dados – Com aplicações em R* (Elsevier)

Sugestões de materiais para estudo

- Canais interessantes no Youtube sobre Python e programação:
 - Programação Dinâmica: <https://www.youtube.com/c/ProgramacaoDinamica/>
 - Curso em Vídeo: <https://www.youtube.com/c/CursoemVideo/>
 - Sentdex (em inglês): <https://www.youtube.com/c/sentdex>
 - Filipe Deschamps: <https://www.youtube.com/c/FilipeDeschamps>
 - DevMedia: <https://www.youtube.com/c/DevmediaBrasil>
 - FreeCodeCamp.org: <https://www.youtube.com/c/freeCodeCamp>

Conhecendo um pouco melhor...

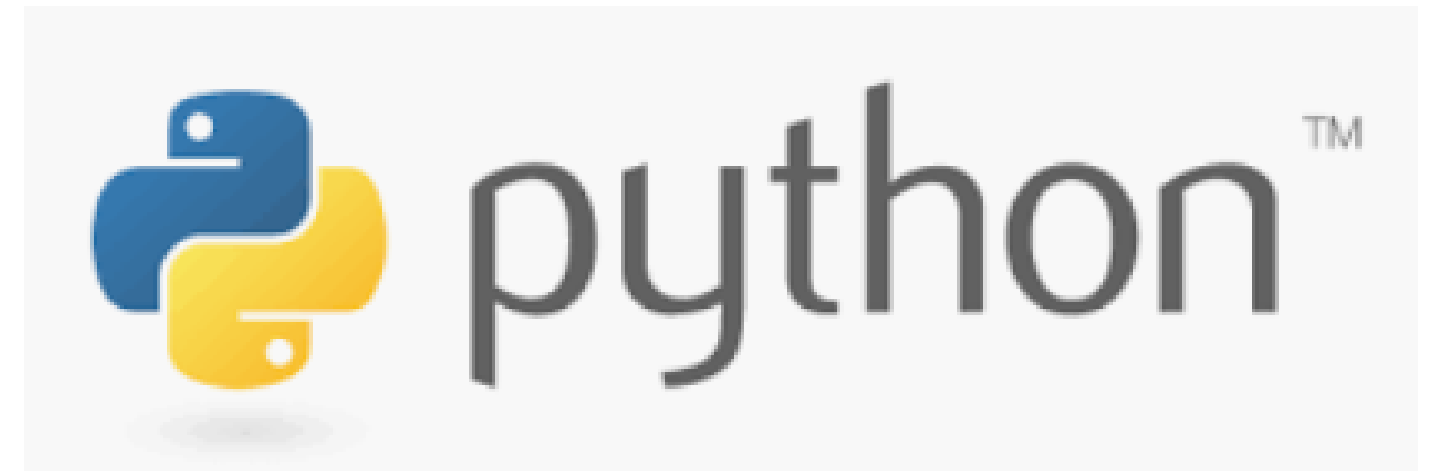




Por que Python?

Por que Python?

- Python foi concebido no final da década de 1980, por Guido van Rossum, na Holanda
- A linguagem foi batizada em homenagem ao programa de TV britânico *Monty Python's Flying Circus*, que fazia muito sucesso na época em boa parte do mundo
- A linguagem foi desenvolvida com o objetivo de ser simples de se desenvolver, e com uma estrutura semântica e sintática intuitiva e natural

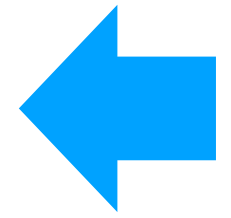


Por que Python?

- Características do Python:
 - Linguagem de propósito geral
 - Fácil e intuitiva
 - Multiplataforma
 - É possível começar a programar de forma simples, sem instalar inúmeros pacotes
 - Livre
 - Organizada
 - Orientada a objetos
 - Inúmeras bibliotecas à disposição
 - Extensa comunidade, com vasta documentação online

Por que Python?

- Principais áreas de atuação:
 - Inteligência Artificial
 - Biotecnologia
 - Computação 3D
 - Ciência de Dados
 - Internet das Coisas



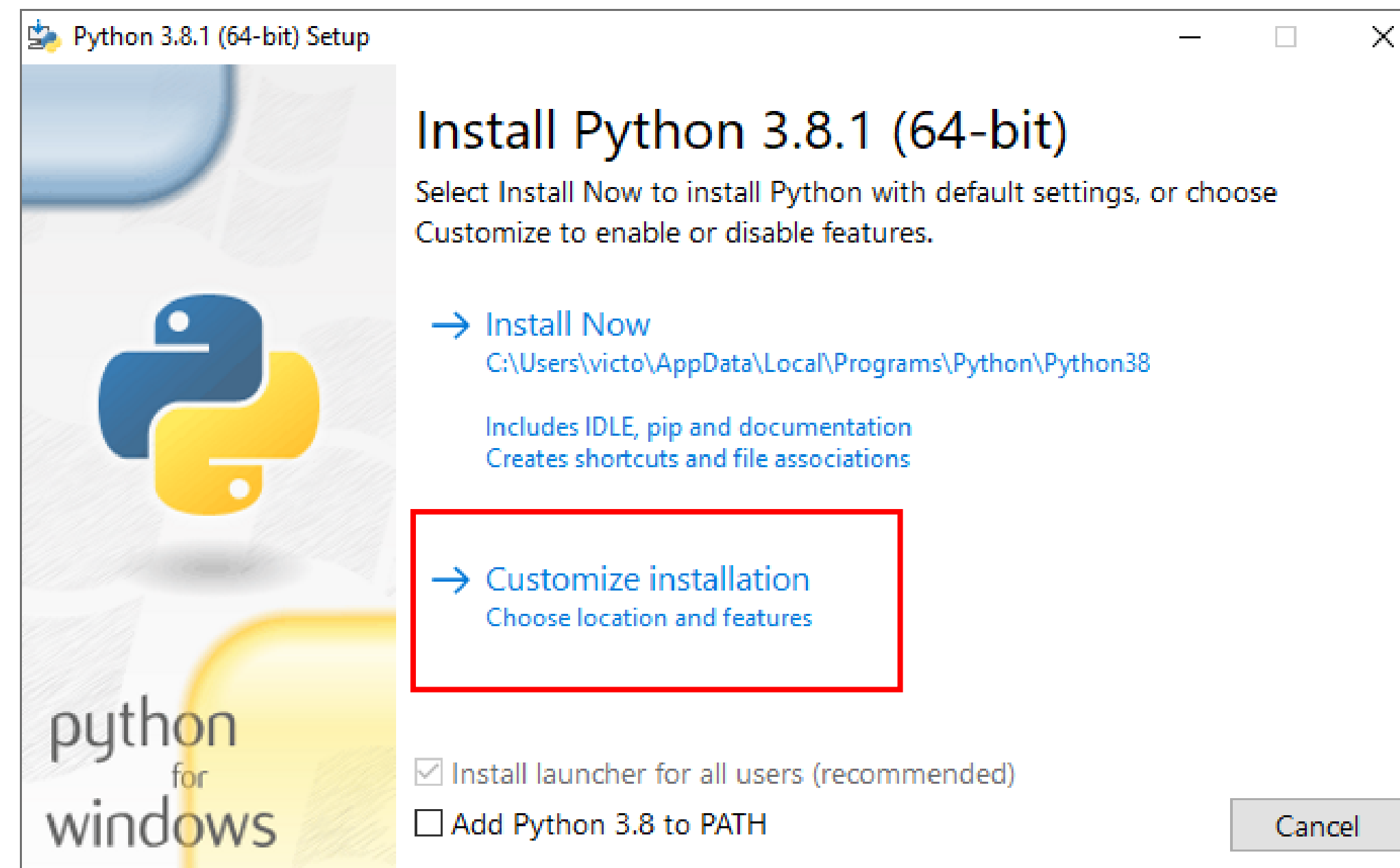
Configurando o ambiente Python

O que é Python?

- Python é uma linguagem de programação de alto nível, lançada por Guido van Rossum em 1991. Atualmente é uma das linguagens de uso mais abrangentes no mundo todo, principalmente nas áreas de Data Science e em aplicações de *back-end*, ou seja, de processamento de dados que não interagem diretamente com o usuário final.
- Diversas organizações utilizam Python atualmente:
 - Google
 - Yahoo!
 - NASA
 - AirCanada

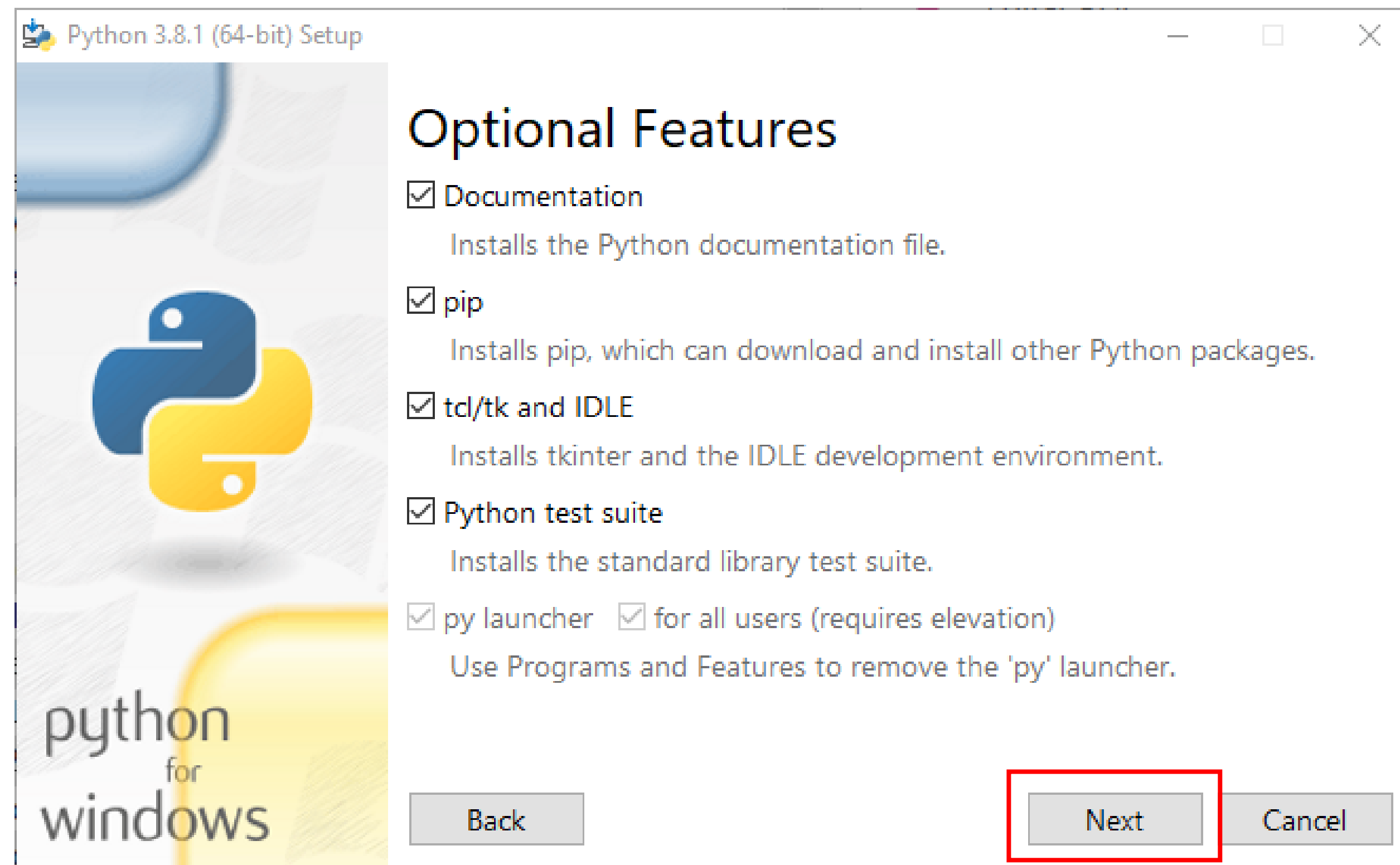
Como instalar Python?

- Neste curso podemos trabalhar com qualquer versão recente do Python, 3.7 ou superior. Para baixar o instalador para Windows, clique [neste link](#). O download do instalador para macOS se encontra [neste link](#).
- Este curso focará no uso do Python para Windows. Para o uso no macOS, veja no [site oficial da linguagem](#) informações particulares.
- Ao clicar no instalador, selecione a opção “Customize installation”.



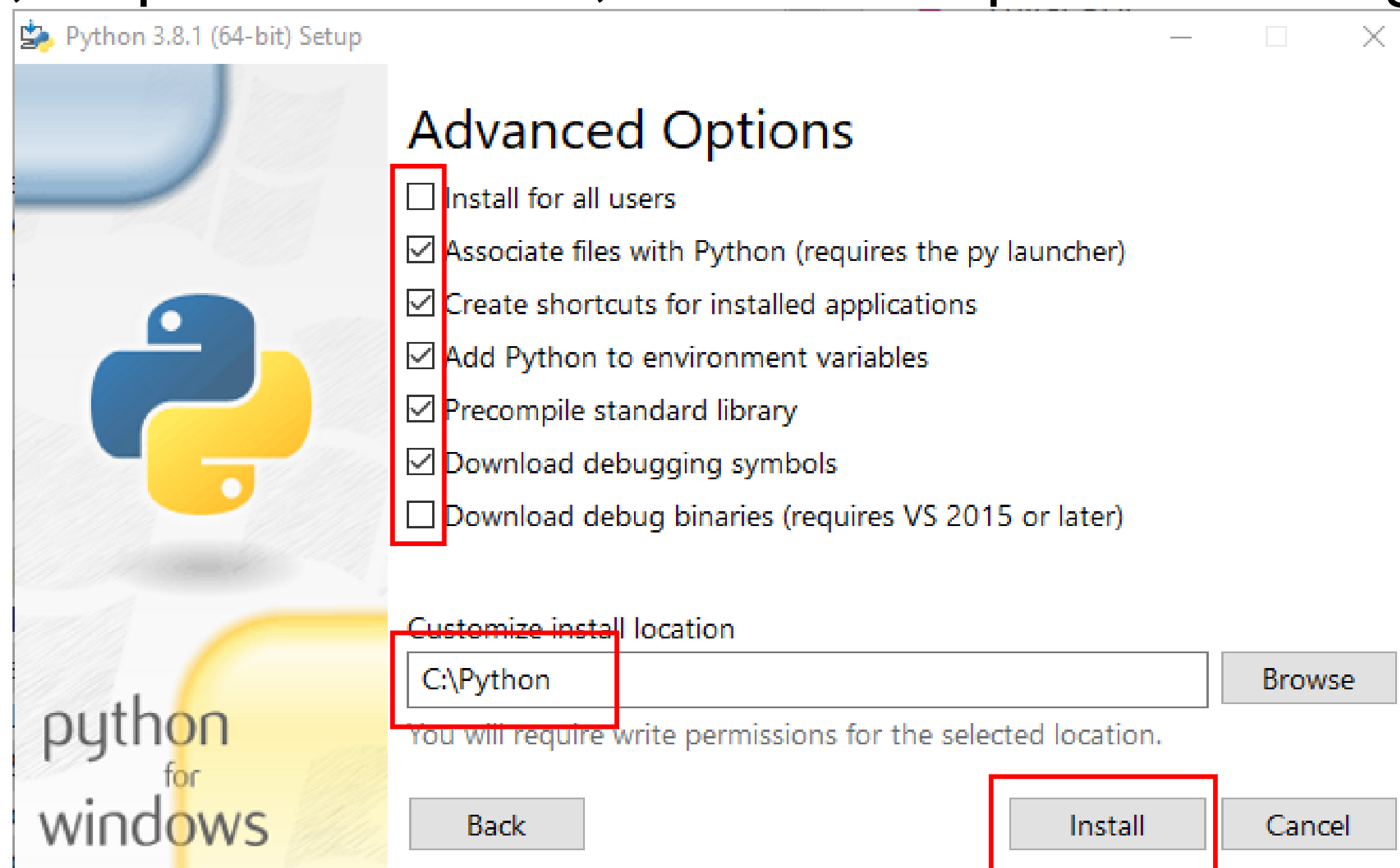
Como instalar Python?

- Na tela “Optional Features”, clique em “Next”.




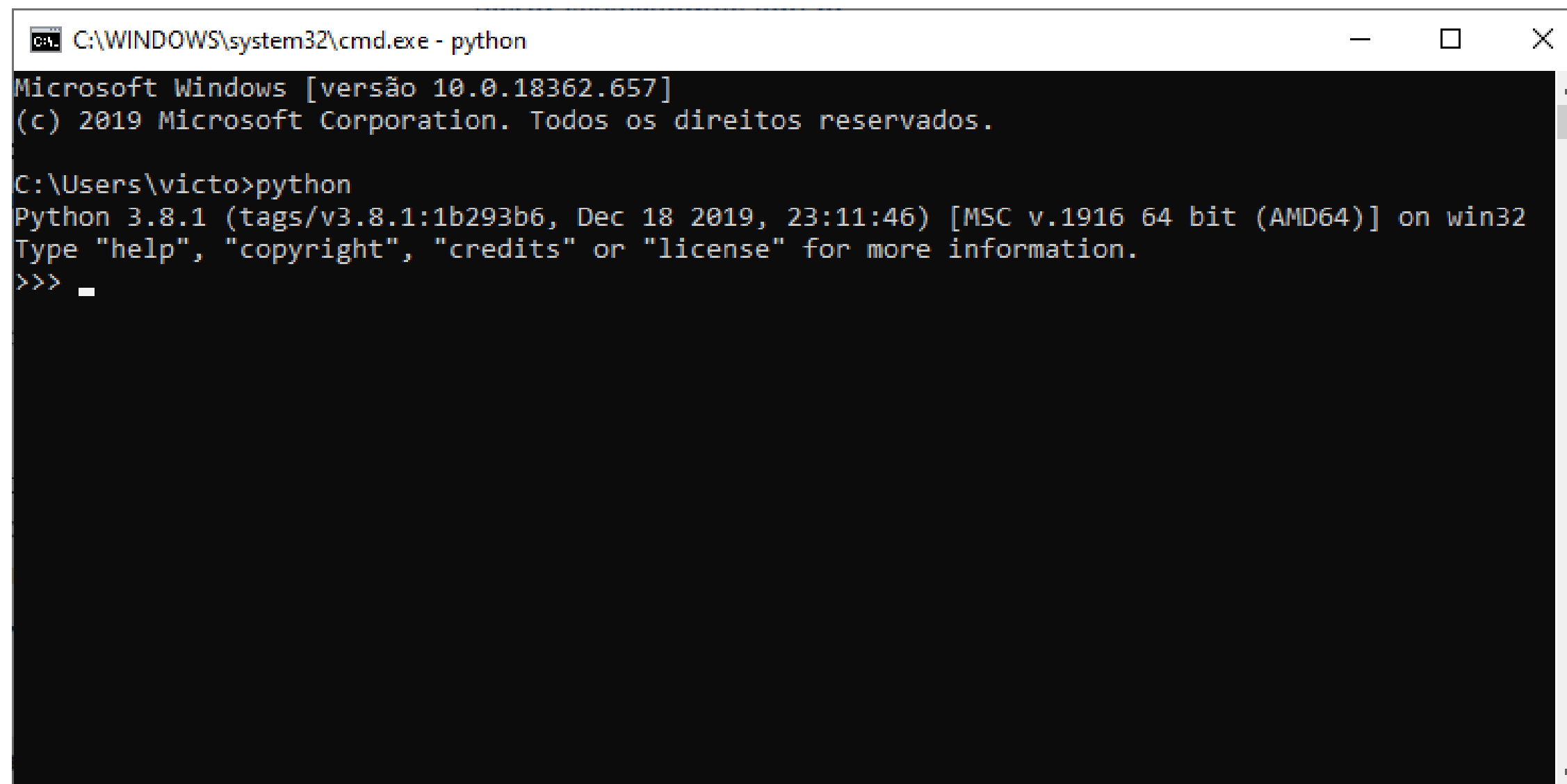
Como instalar Python?

- Na tela “Advanced Options”, deixe as caixas de opções marcadas conforme a imagem abaixo.
- No campo “Customize install location”, escolha um caminho de fácil acesso. O caminho sugerido é “C:\Python”.
- Com tudo pronto, clique em “Install”, e conclua após a mensagem de sucesso.



Como instalar Python?

- Para conferir se a instalação foi bem sucedida, faça os seguintes passos:
 - Clique no botão do Windows ;
 - Digite a caixa de pesquisa **prompt de comando**, e abra o programa;
 - Na janela que abrir, digite o comando **python** e pressione Enter;
 - O Windows deve inicializar um editor de Python na mesma janela, como mostrado abaixo.



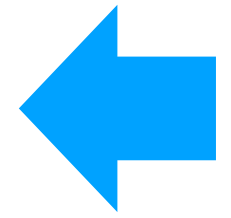
```
C:\WINDOWS\system32\cmd.exe - python
Microsoft Windows [versão 10.0.18362.657]
(c) 2019 Microsoft Corporation. Todos os direitos reservados.

C:\Users\victo>python
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```



Algumas dicas iniciais após instalar o Python

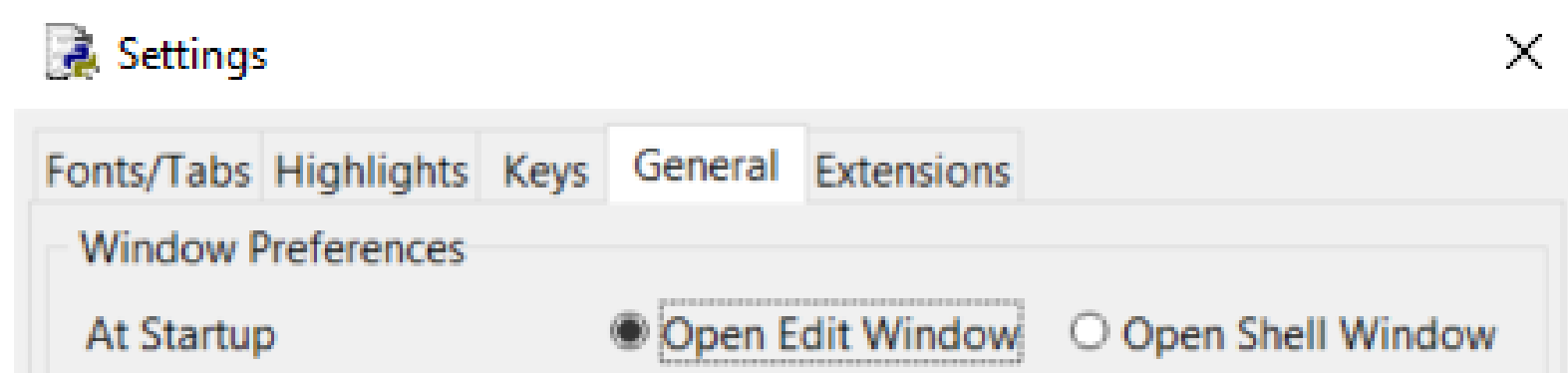
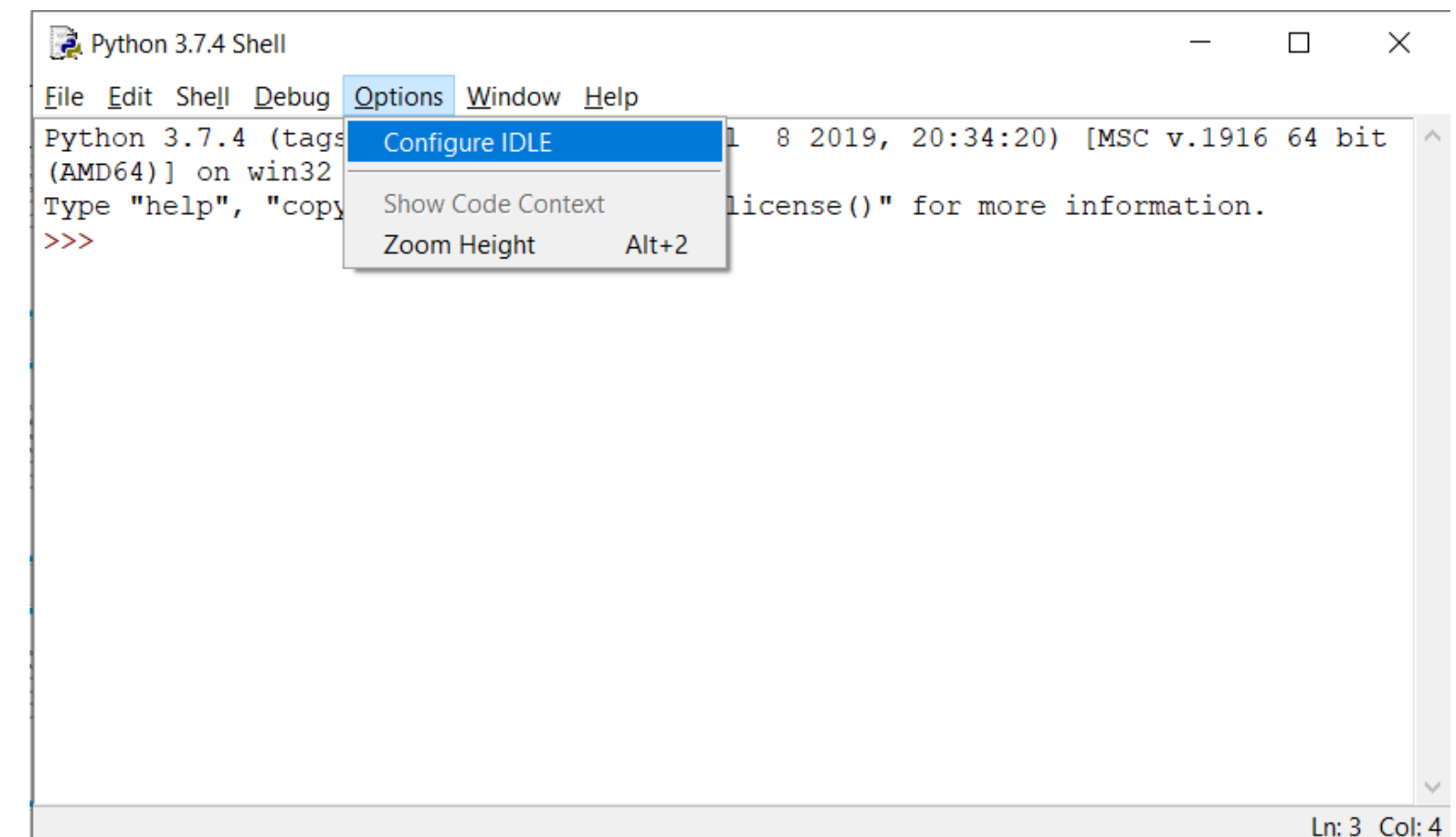
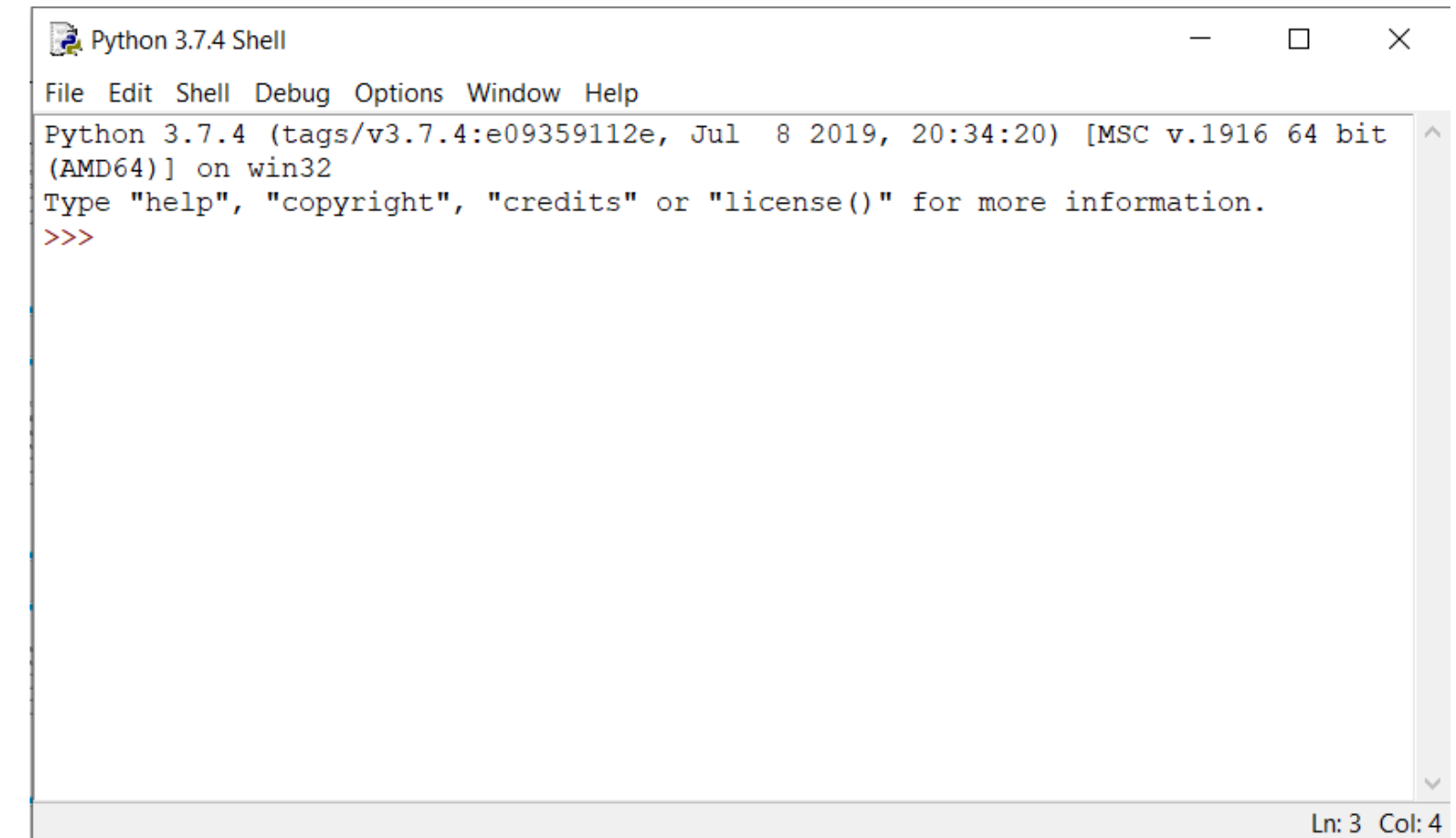
1. Antes de abrir o programa, abra o Windows Explorer, clique em **Este Computador** e, em seguida, no drive do seu computador (C: ou D:, dependendo da sua máquina);
2. Neste diretório, crie uma pasta chamada **Projetos**. Esta pasta será usada para armazenar todos os seus projetos de software;
3. Dentro da pasta de projetos, crie a pasta da disciplina (p.ex., **algoritmos**);
4. Evite utilizar caminhos muito longos (p.ex., C:\Users\12304010\Projetos\Nome-da-pessoa\Documentos\etc...) ou incluir espaços no caminhos (p.ex., C:\Victor Machado). O primeiro é muito trabalhoso para utiliza-lo recorrentemente, e o segundo pode causar alguns problemas na execução do código;
5. Sempre que criar arquivos Python, comece o nome do arquivo com uma letra (p.ex., **main.py**, **app.py**, **aula.py**). Evite usar números, espaços ou acentos nos nomes dos arquivos.



Instalando e configurando IDEs

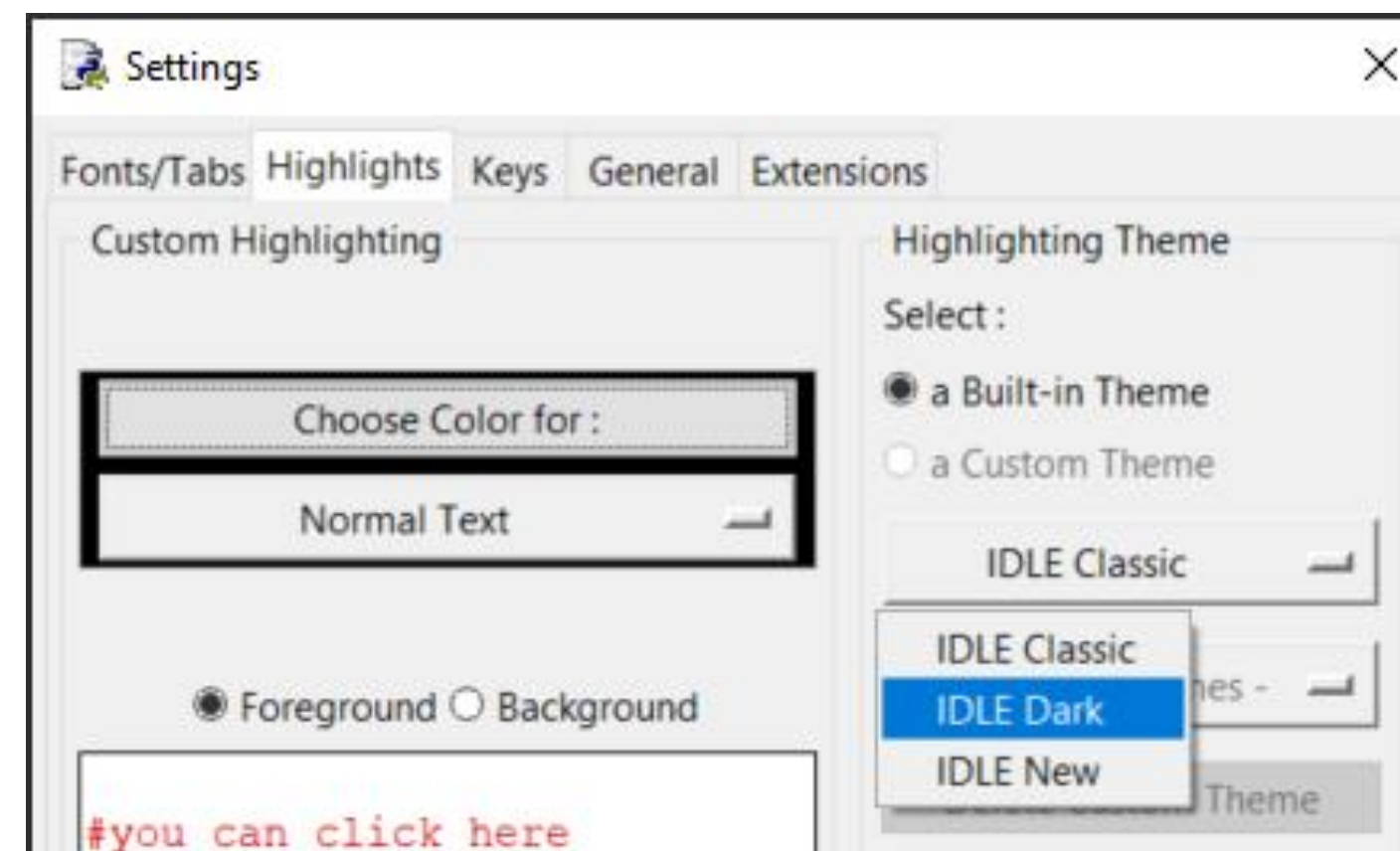
Configurando o IDLE

- Abra IDLE utilizando o ícone do Windows e procurando pelo programa
- O programa abre no modo **Shell**, que é a janela de execução do código. Usamos essa tela apenas para checar os resultados ou quando queremos executar códigos de uma linha (testar uma função, por exemplo)
- Para abrir o editor automaticamente, vá no menu **Options > Configure IDLE**
- Na aba **General**, em **Windows Preferences**, marque a opção **Open Edit Window**. Clique em Ok e reinicie o IDLE



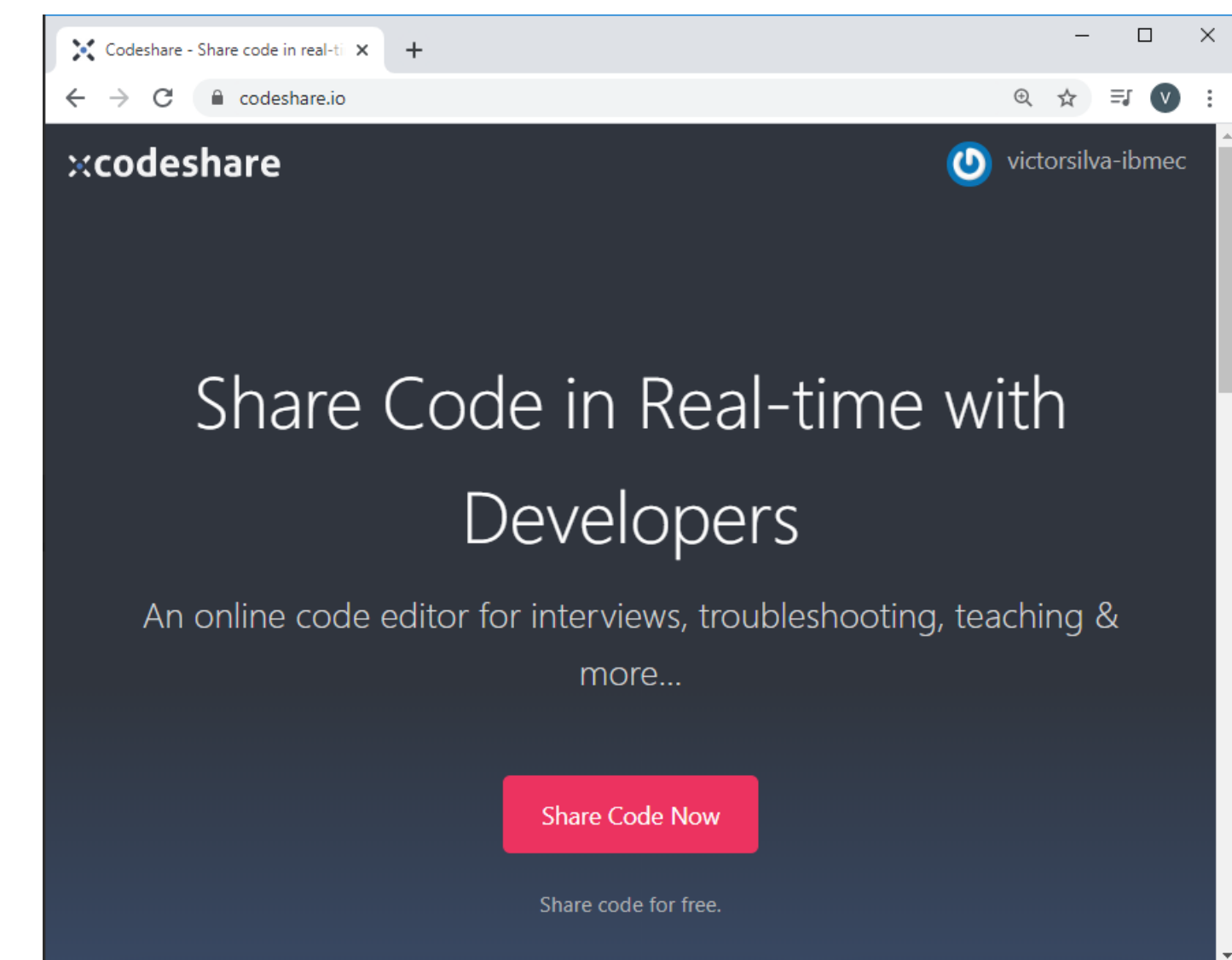
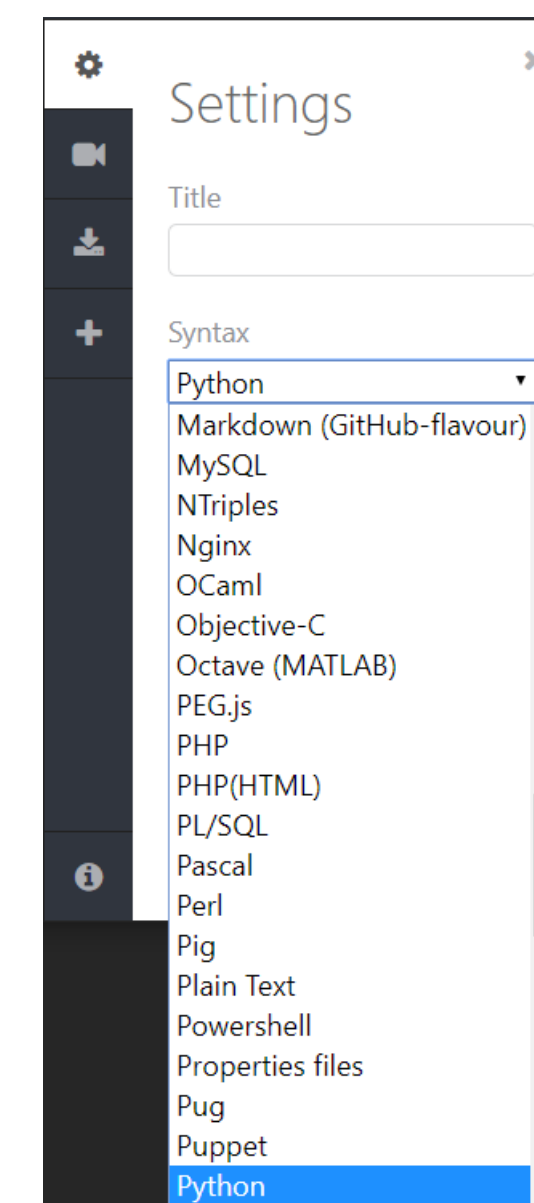
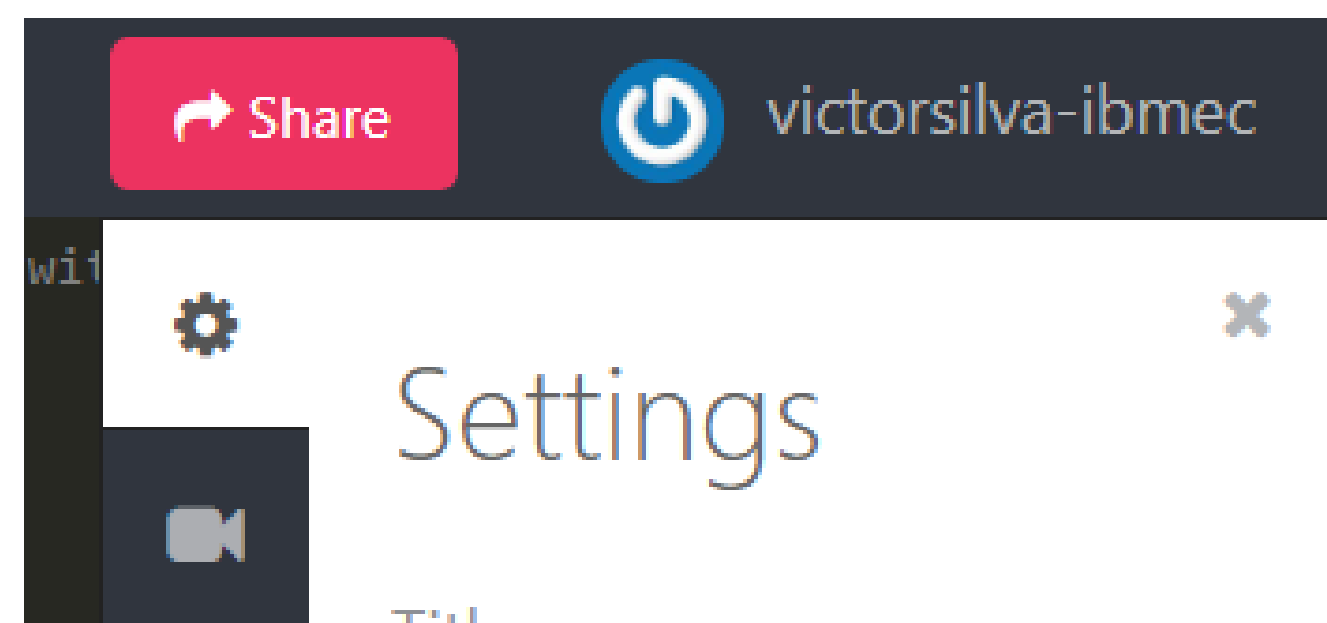
Configurando o IDLE

- Normalmente é usual programadores trabalharem com editores de texto que tenham um fundo escuro, o que prejudica menos a visão
- Na mesma tela de **Configurações**, vá para a aba **Highlights**, e na coluna da direita, em **Highlight Theme**, clique em **IDLE Classic** e selecione a opção **IDLE Dark**
- Clique em OK para mudar o tema para escuro



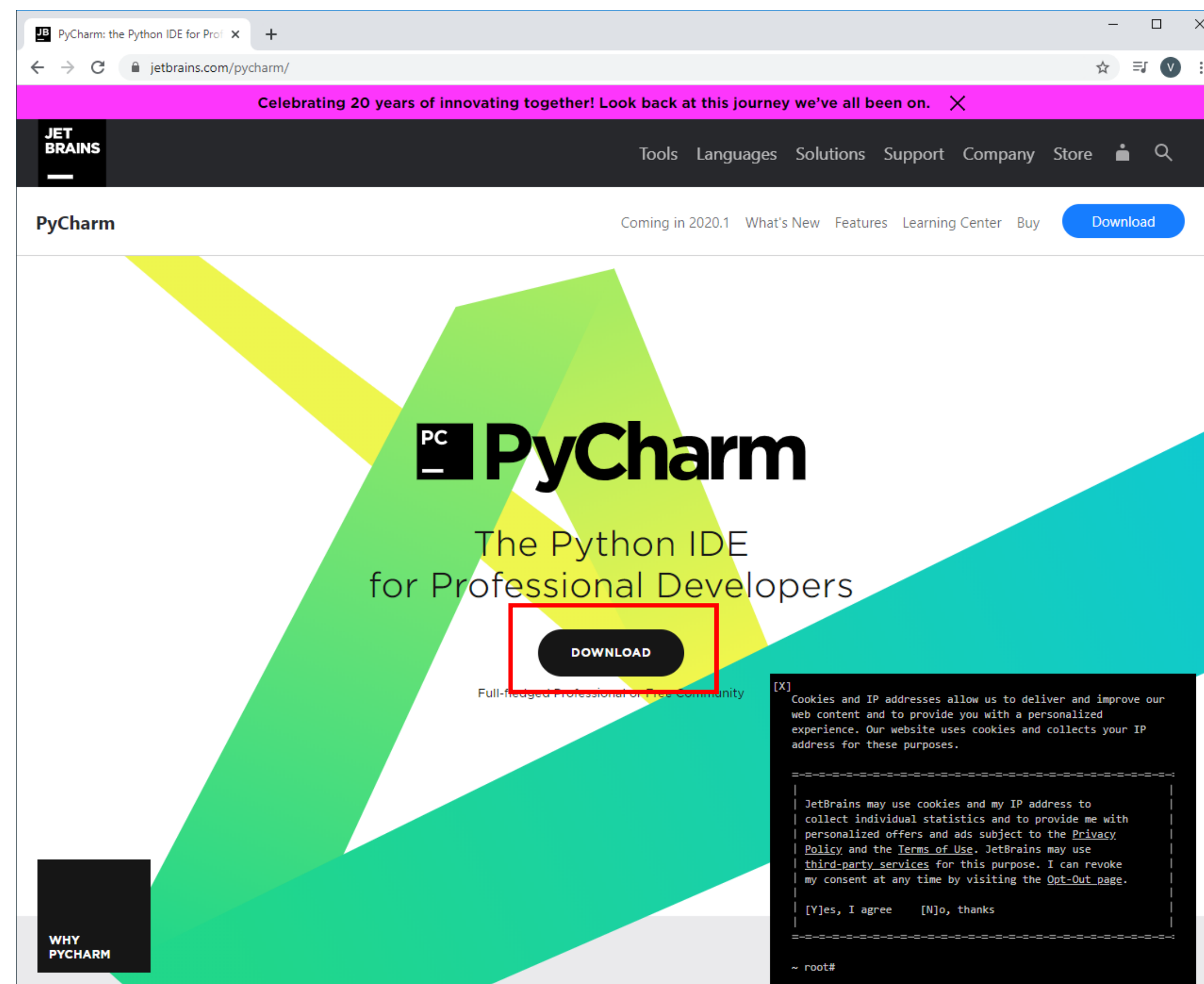
Usando o CodeShare

- Acesse <http://codeshare.io> e faça o cadastro
- Tendo cadastrado, na tela inicial clique em “Share Code Now”
- Um editor de texto vai aparecer. Na coluna da direita, clique na engrenagem, e em **Syntax** marque a opção **Python**
- Para compartilhar, clique em **Share**, no canto superior da janela
- O CodeShare vai liberar um link para ser usado por outras pessoas



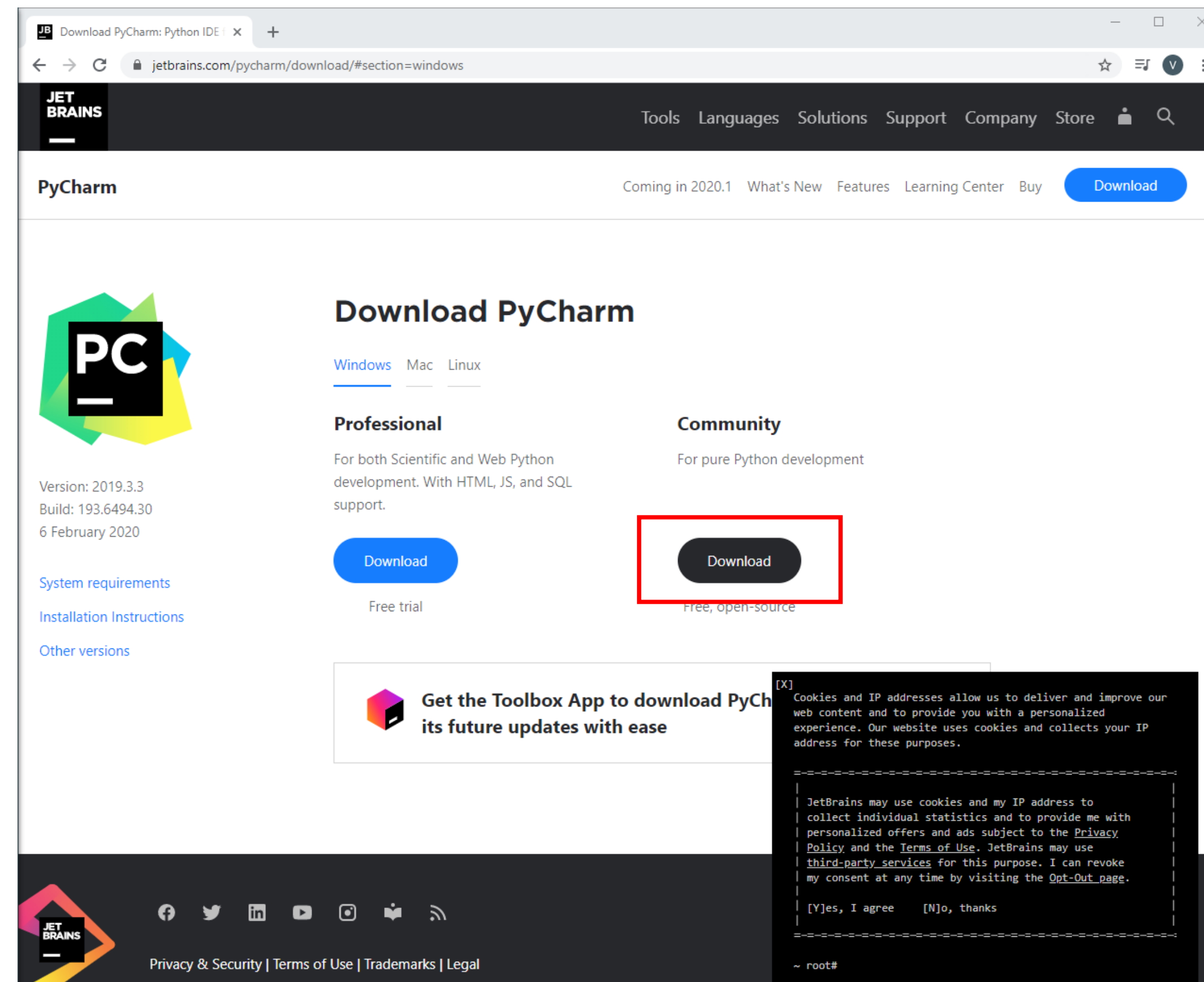
Instalando e configurando o PyCharm

- O **PyCharm** é um dos editores (ou IDEs) mais usados para a programação de aplicações em Python. Para baixar e instalar, primeiro acesse a página <https://www.jetbrains.com/pycharm/> e clique em “Download”.



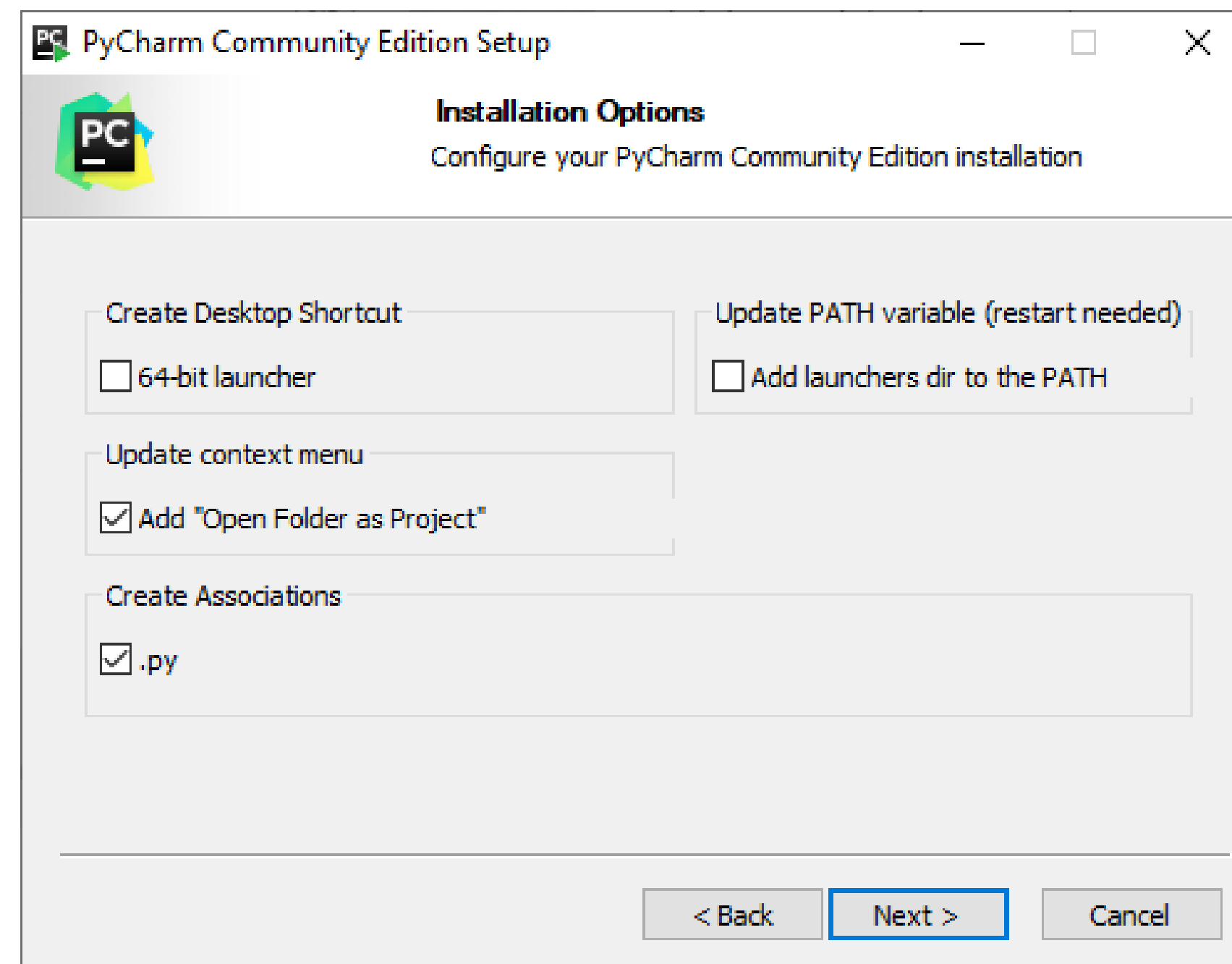
Instalando e configurando o PyCharm

- Escolha o seu sistema operacional (Windows, Mac ou Linux - vamos trabalhar com Windows) e baixe a versão “Community”. O download deve começar automaticamente.



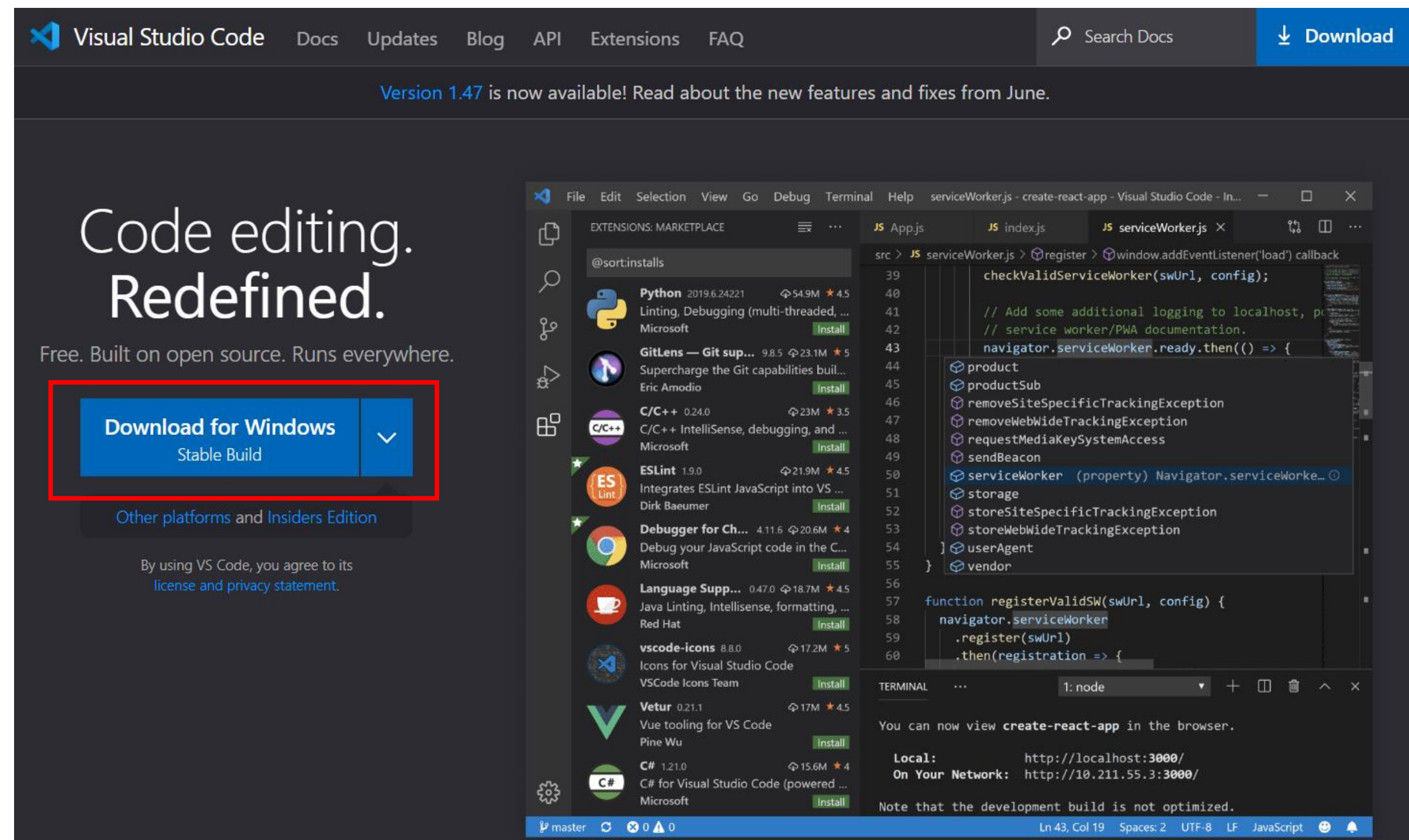
Instalando e configurando o PyCharm

- Abra o instalador do PyCharm e clique em “Next”;
- Na tela de escolha do caminho de instalação, clique em “Next”;
- Na tela seguinte, marque as opções indicadas na imagem abaixo e clique em “Next”;
- Na tela seguinte, clique em “Install” para começar a instalação.



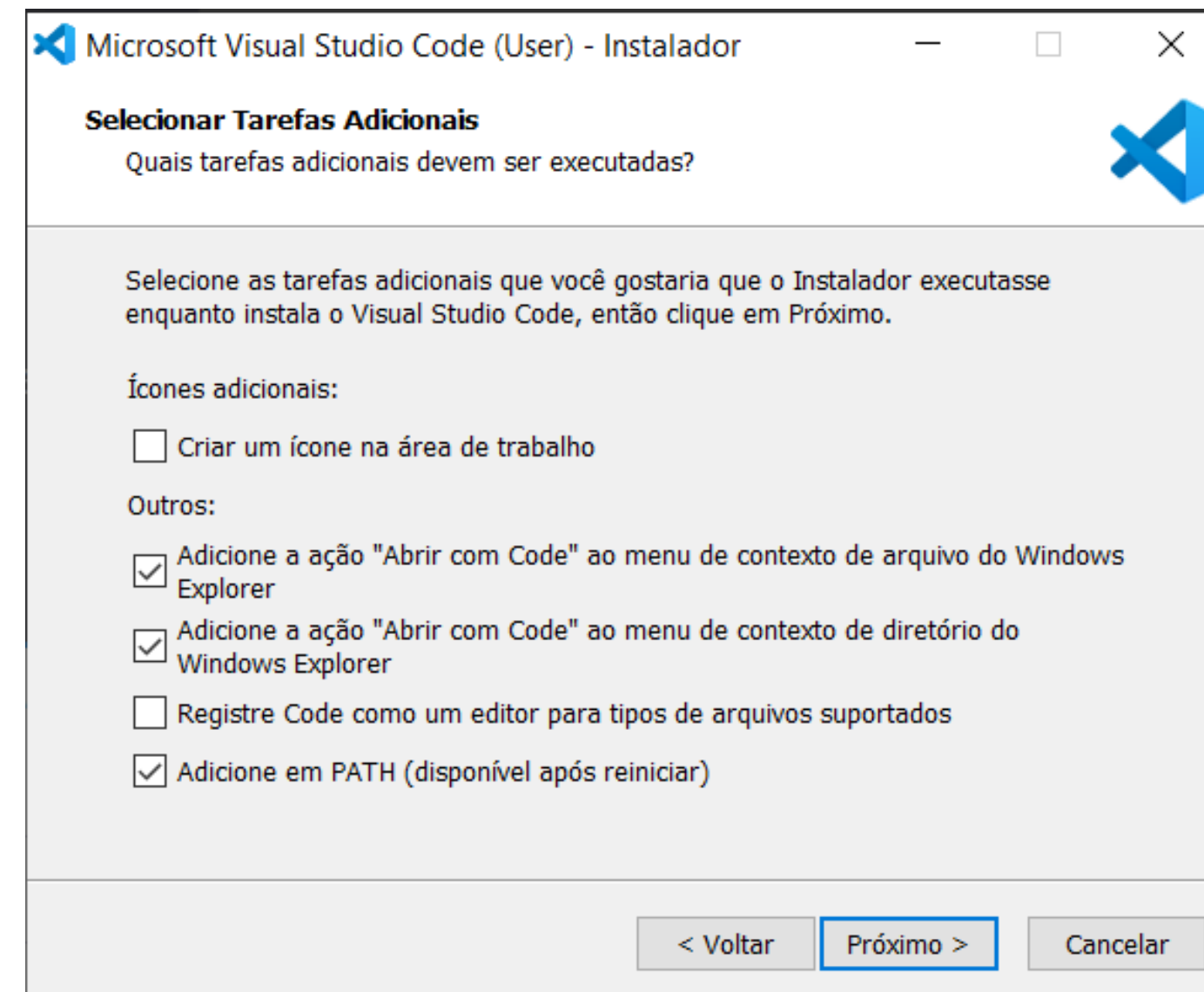
Instalando e configurando o VSCode

- O **VSCode** é um dos IDEs recentes mais famosos para basicamente qualquer linguagem de programação. Possui inúmeras extensões que facilitam e customizam o editor para a necessidade de cada programador. Para baixar e instalar, primeiro acesse a página <https://code.visualstudio.com/> e clique em “Download for Windows”. Clicando na seta à direita existem opções para MAC e Linux.



Instalando e configurando o VSCode

- Abra o instalador, e após aceitar o acordo de licença e clicar em “Próximo”, clique em “Próximo” novamente até chegar na tela “Selecionar Tarefas Adicionais”
- Marque as opções abaixo e clique em “Próximo” e depois em “Instalar”

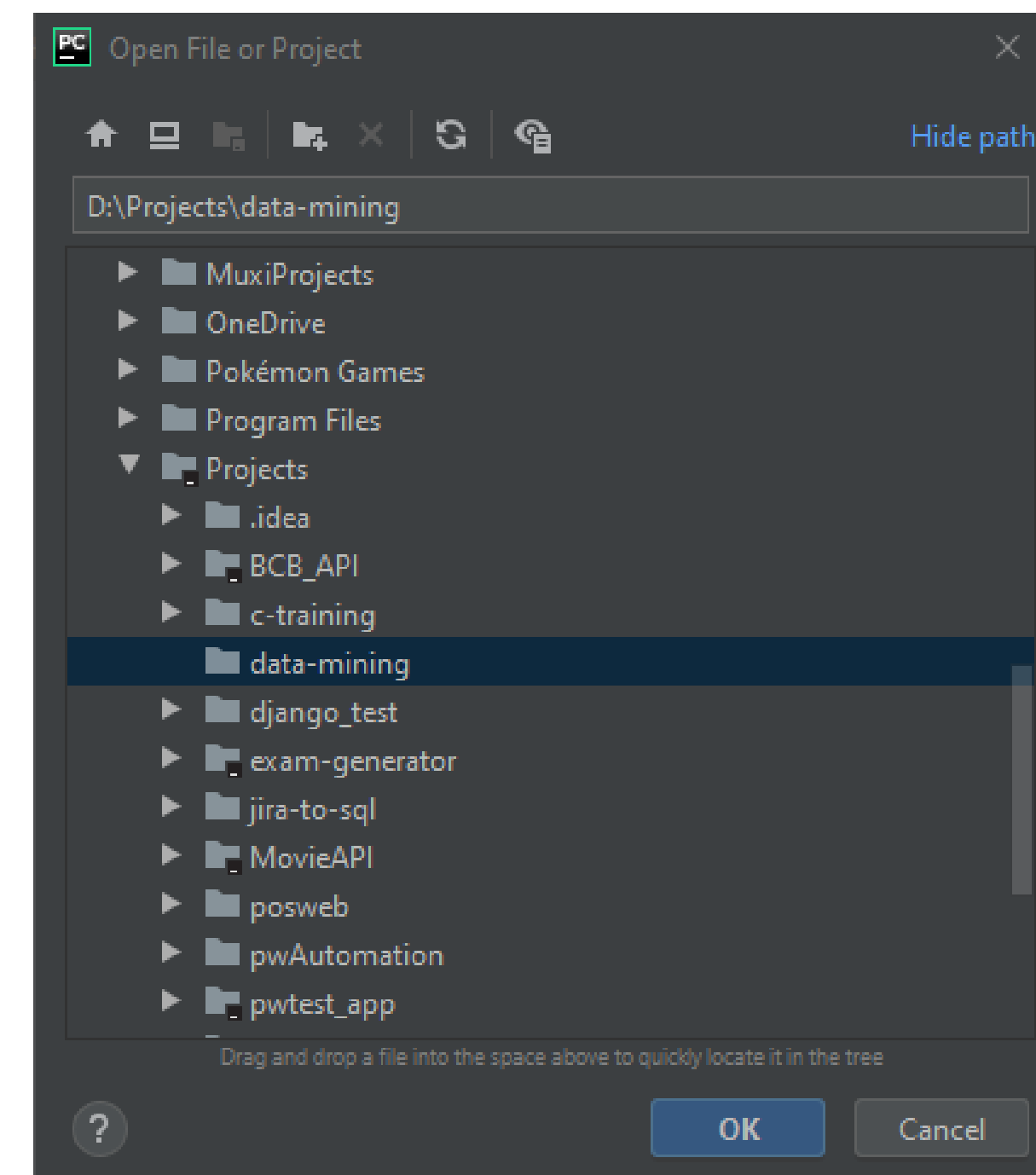
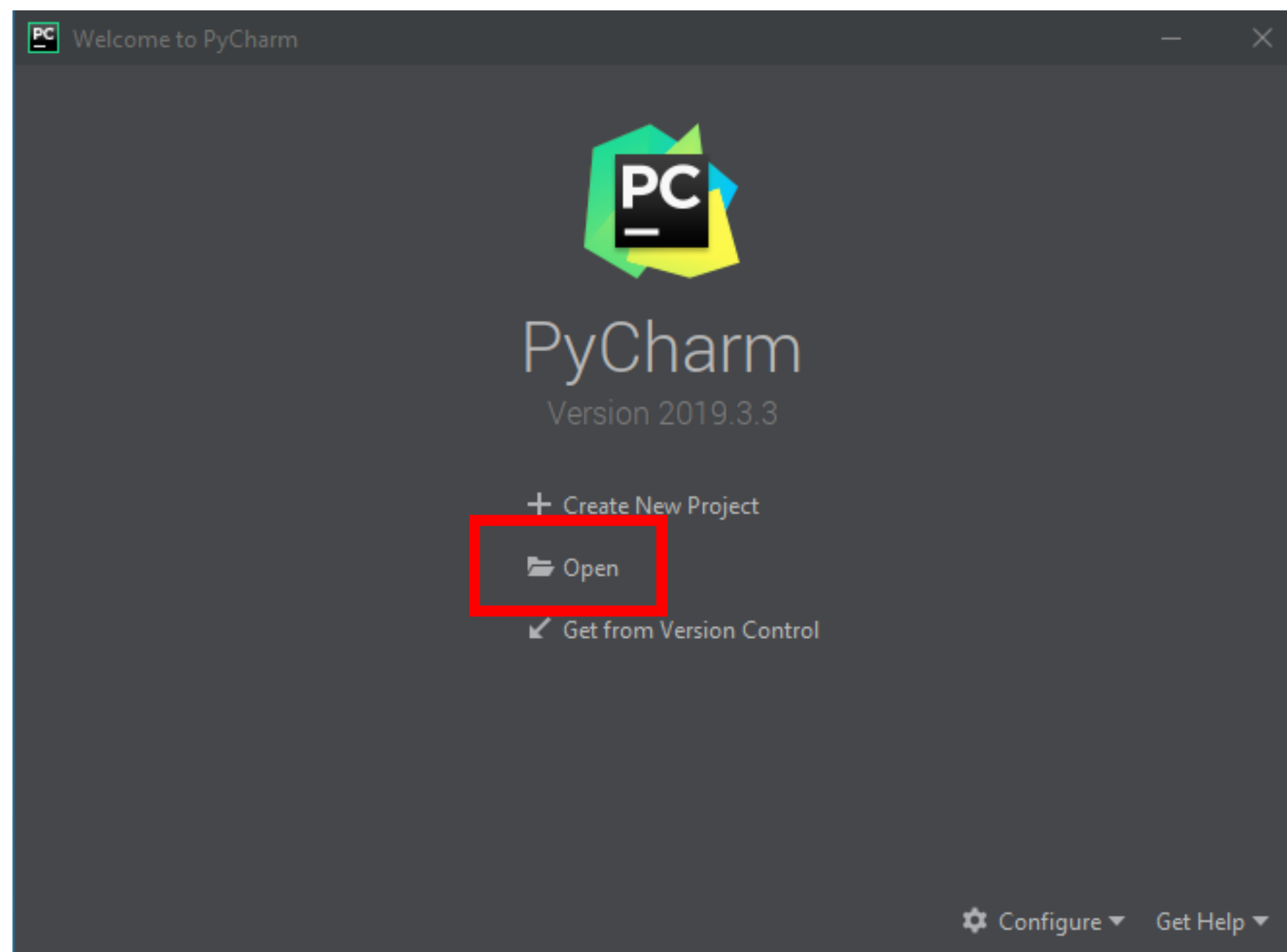




Utilizando o PyCharm

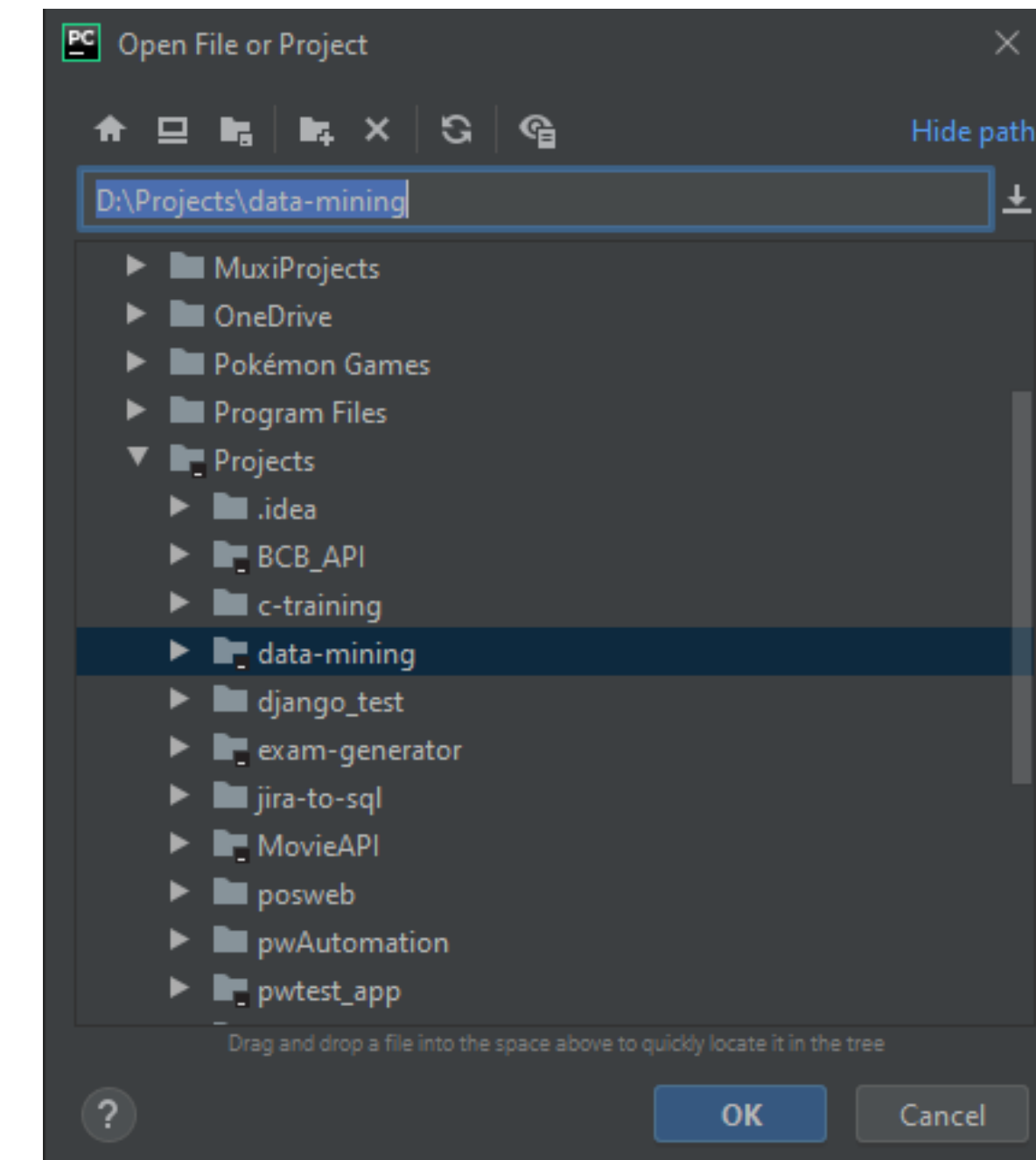
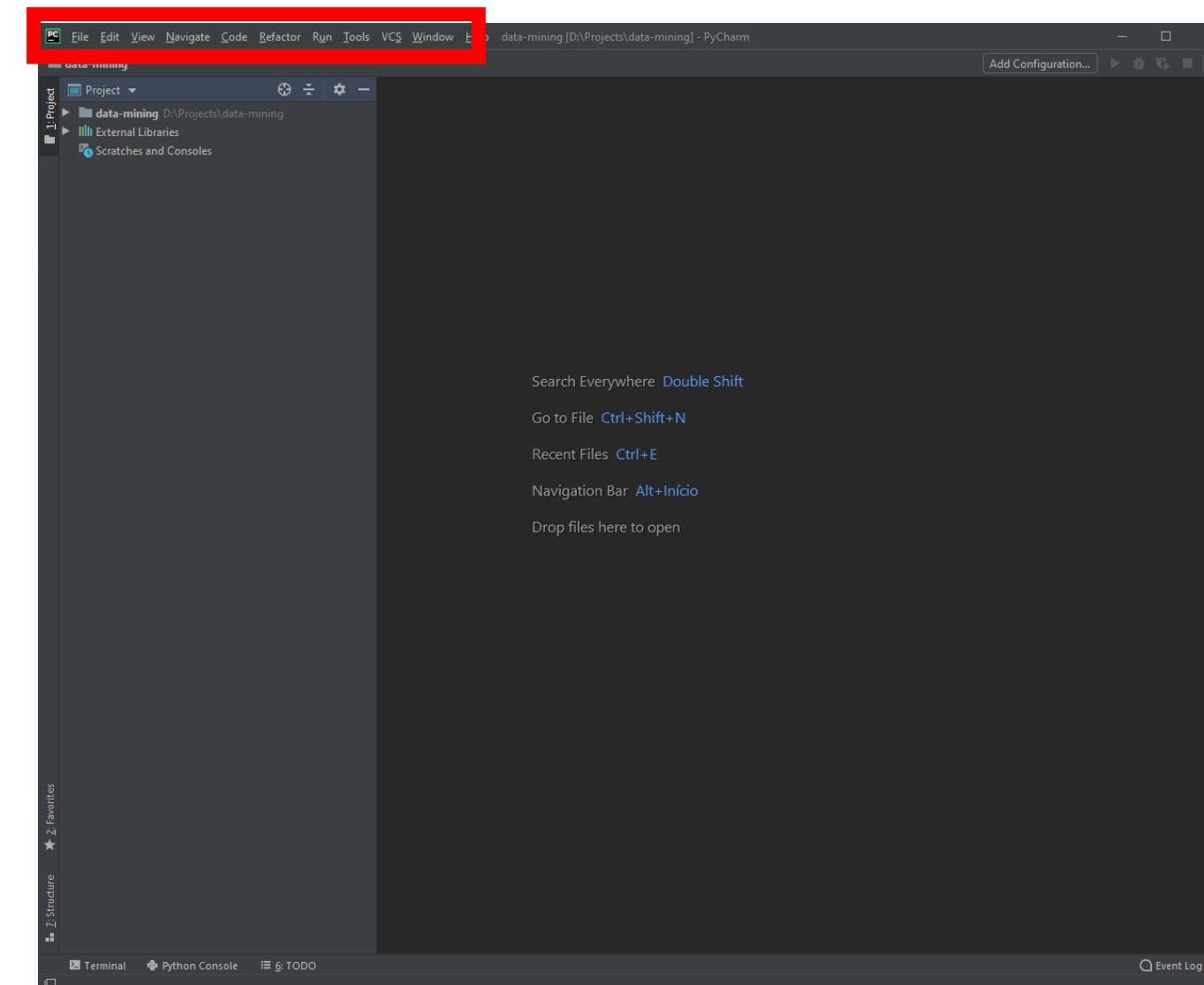
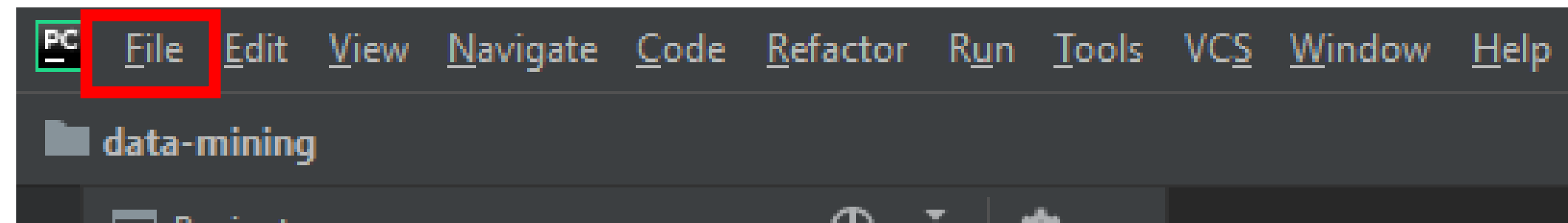
Iniciando o PyCharm

- **Se essa é a primeira vez que abre o PyCharm:**
 - Garanta que você possui uma pasta com o projeto desejado (p.ex., **D:\Projetos\data-mining**);
 - Abra o PyCharm, e na tela inicial clique em **Open**. Selecione a pasta que você criou e clique em **Ok**;



Iniciando o PyCharm

- **Se você já abriu o PyCharm antes:**
 - Garanta que você possui uma pasta com o projeto desejado (p.ex., **C:\Projetos\data-mining**);
 - Abra o PyCharm, e na tela que abrir clique em **File > Open**. Selecione a pasta que você criou e clique em **Ok**;

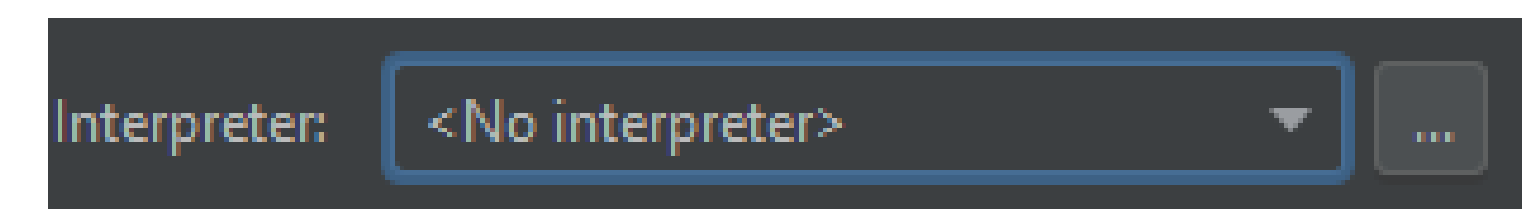
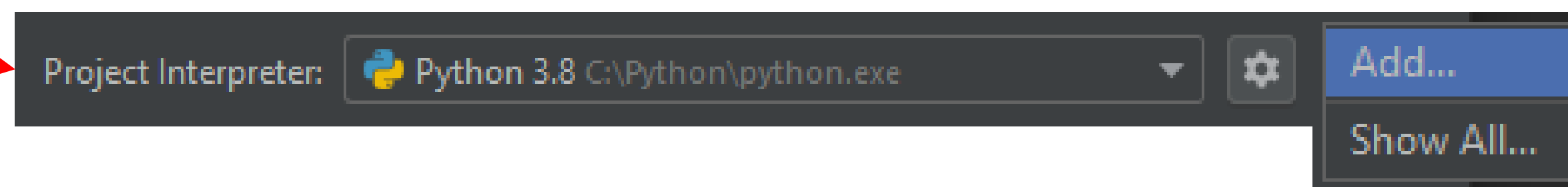


Algumas configurações do PyCharm

- **Mudar o tema para escuro:**
 - Clique em **File > Settings**;
 - Na janela que aparecer, procure por **Appearance & Behavior > Appearance**. No campo **Theme**, selecione o tema desejado (minha sugestão é o **Darcula**);
 - Clique em **Ok** para fechar a tela de configurações.

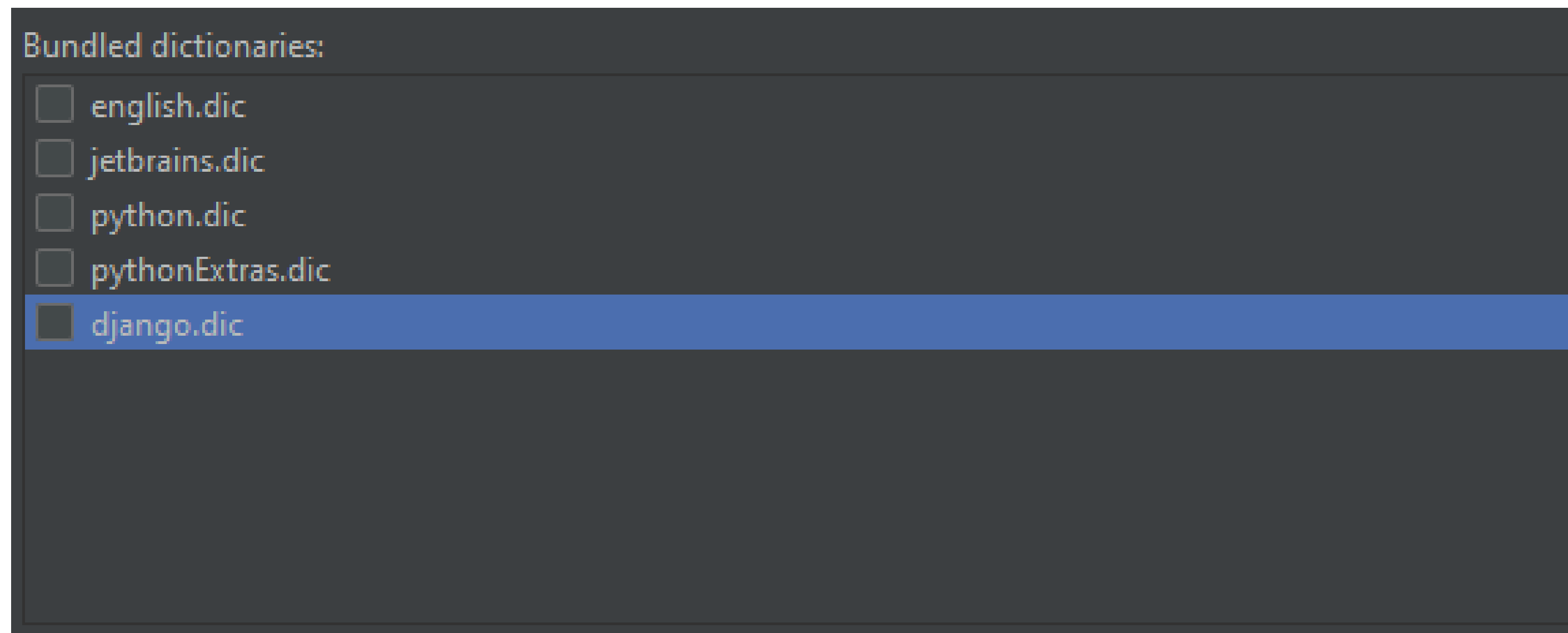
Algumas configurações do PyCharm

- **Configurar o interpretador de Python:**
 - Clique em **File > Settings**;
 - Na janela que aparecer, procure por **Project: <nome-do-projeto> > Project Interpreter**. No campo **Project Interpreter**, verifique se o Python informado é o instalado (no meu caso, **C:\Python\python.exe**). Caso não seja, clique na engrenagem e em **Add...**;
 - Na nova janela, clique em **System Interpreter**, e no campo **Interpreter** clique nos três pontos à direita para indicar o local que o seu Python está instalado. Aperte **Ok** até voltar à tela de **Settings**;
 - Novamente no campo **Project Interpreter**, altere o Python para refletir o que está instalado. Em seguida clique em **Ok**.



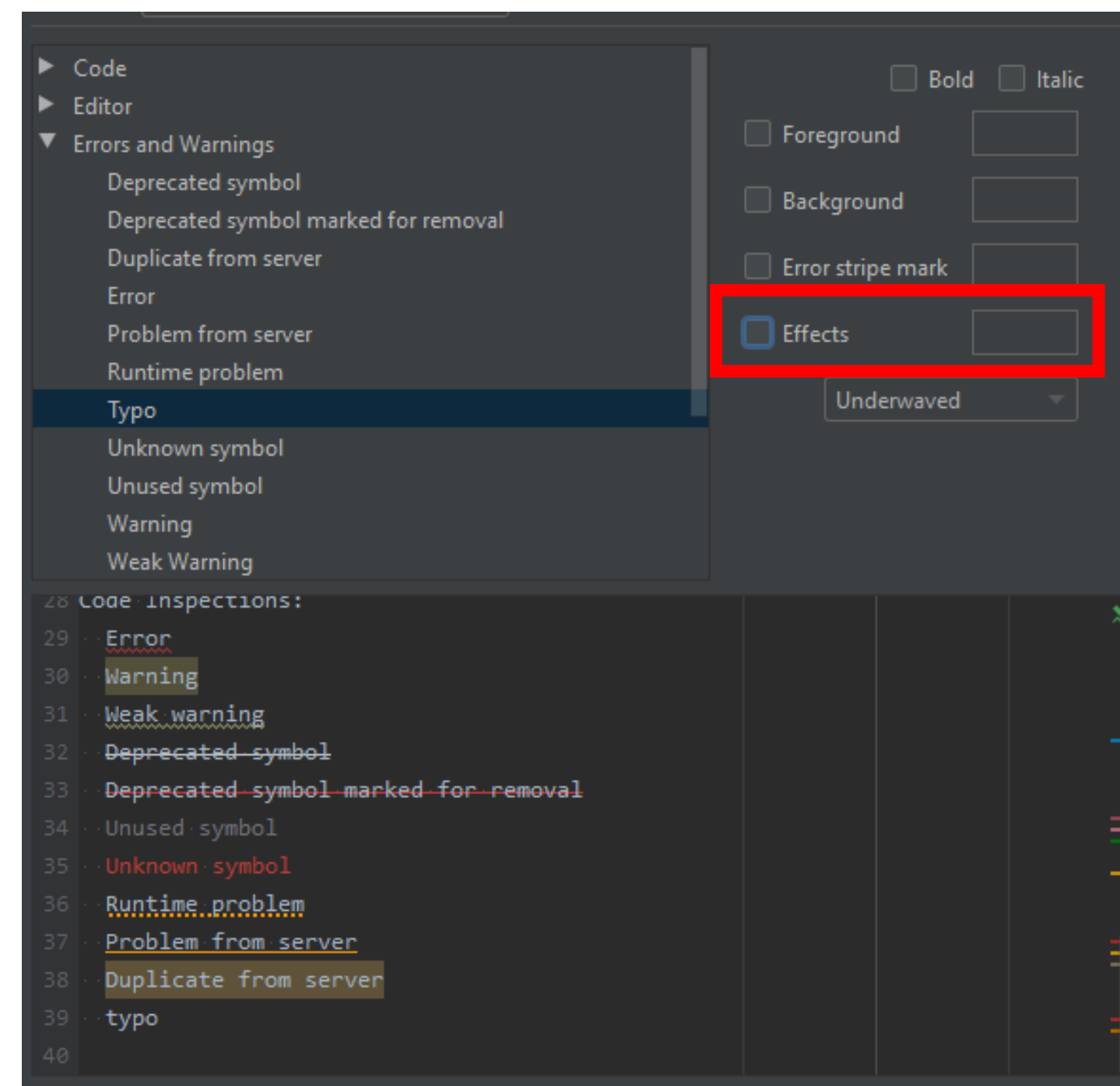
Algumas configurações do PyCharm

- **Desativar inspeção ortográfica:**
 - Clique em **File > Settings**;
 - Na janela que aparecer, procure por **Editor > Spelling**. No campo **Bundled dictionaries**, desmarque todas as opções;
 - Clique em **Ok** para sair das configurações.



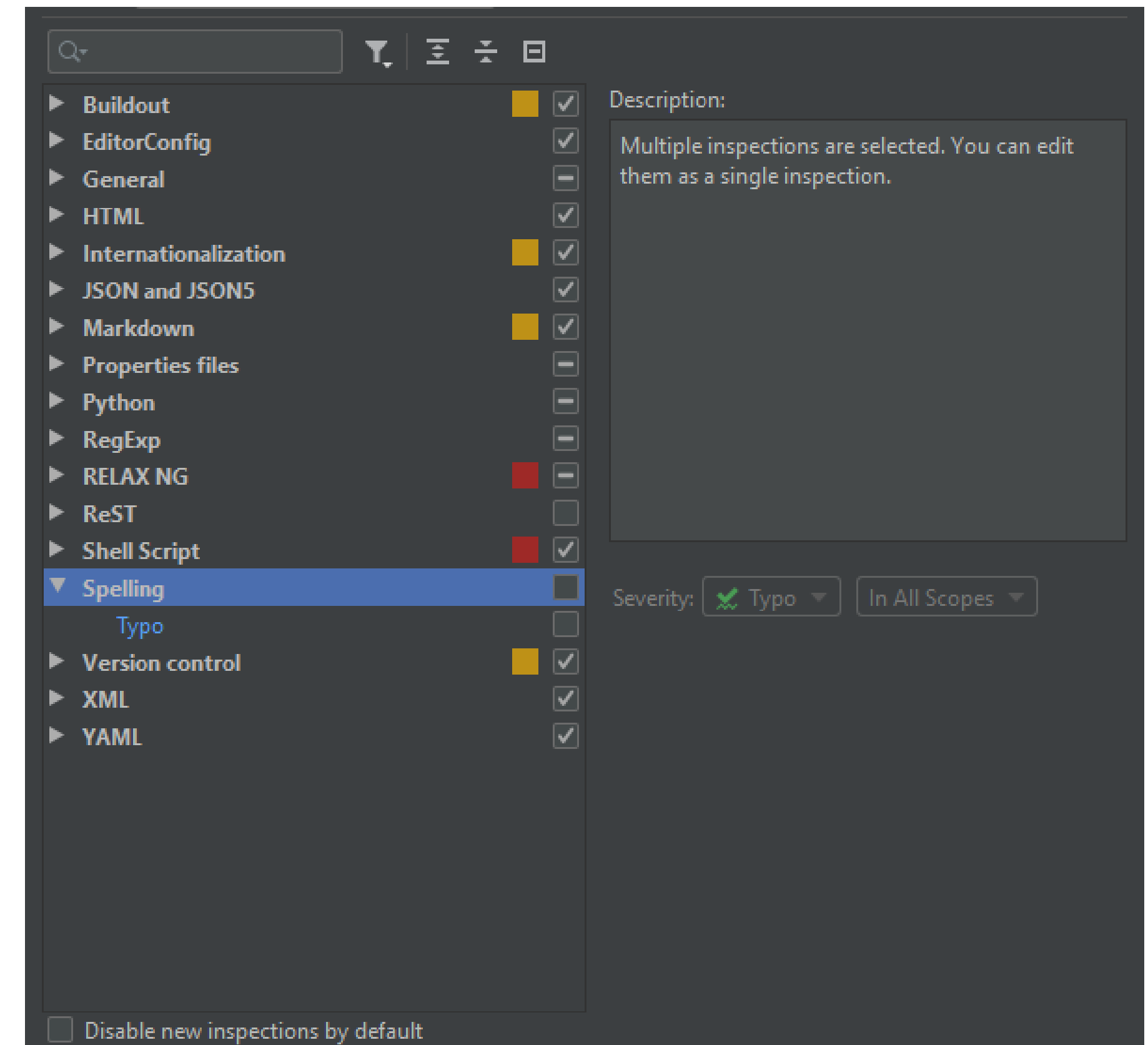
Algumas configurações do PyCharm

- Desmarcar esquema de cores para *tipos*:
 - Clique em **File > Settings**;
 - Na janela que aparecer, procure por **Editor > Color Scheme > General**. No campo **Errors and Warnings > Typo**, desmarque a opção **Effects**;
 - Clique em **Ok** para sair das configurações.



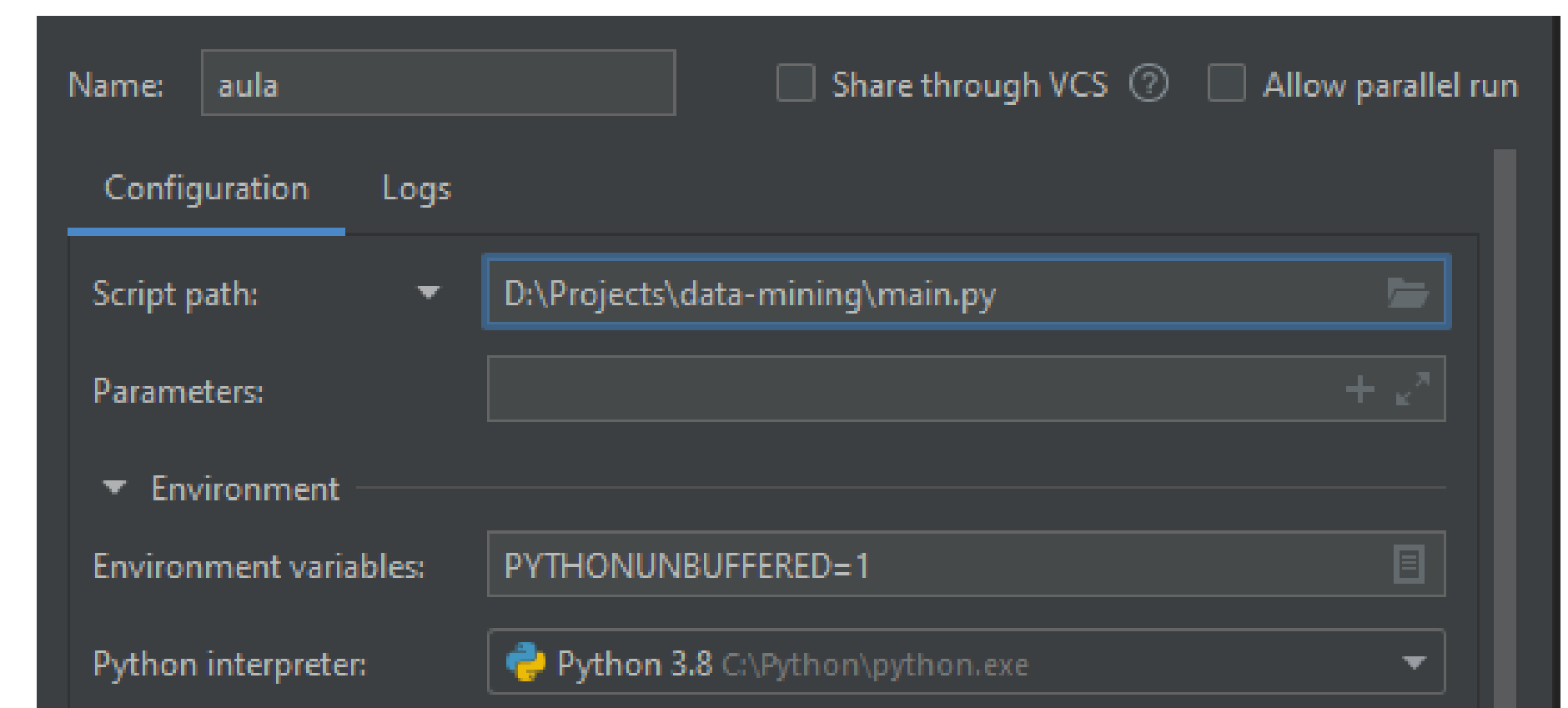
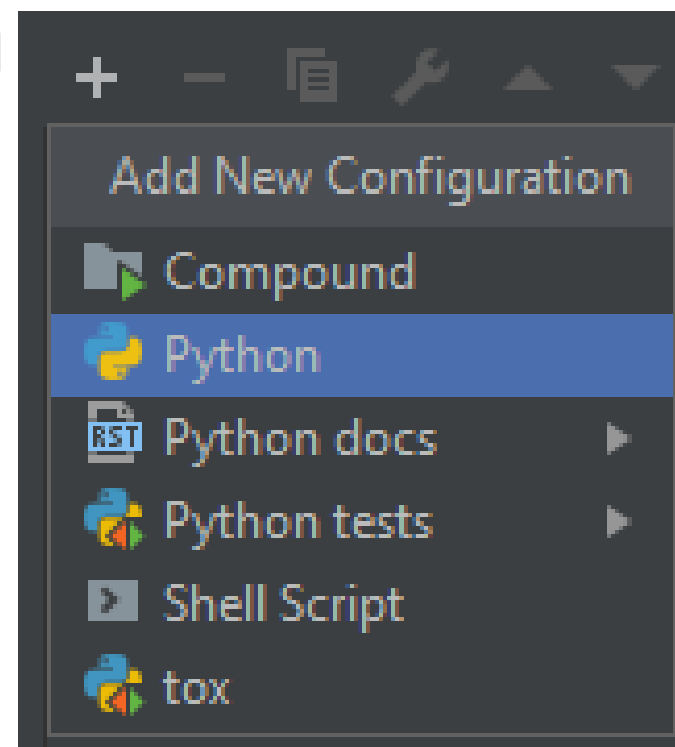
Algumas configurações do PyCharm

- **Ajustar os destaques de inspeções:**
 - Clique em **File > Settings**;
 - Na janela que aparecer, procure por **Editor > Inspections**. Desmarque o campo **Spelling**;
 - Na seção **Python**, recomendo *desmarcar* algumas opções para simplificar o aprendizado:
 - *Boolean variable check can be simplified*;
 - *Chained comparisons can be simplified*;
 - *Comparison with None performed with equality operators*.
 - Clique em **Ok** para sair das configurações.



Algumas configurações do PyCharm

- **Configurar uma determinada execução de código:**
 - Garanta que as configurações do slide **Configurar o interpretador de Python** foram executadas;
 - Aperte **Alt + Shift + F10 > Edit Configurations**, ou na barra de tarefas clique em **Run > Edit Configurations**;
 - Na janela que aparecer, no canto superior esquerdo clique no botão de **+ > Python**;
 - Dê um nome para a execução (p.ex., **aula**) e em **Script path**, informe o caminho do arquivo que você deseja executar;
 - Certifique-se que o **Python interpreter** é o configurado anteriormente, clique em **Apply** e em seguida em **Close**;
 - Para rodar o arquivo, é só usar o atalho **Shift + F10**.

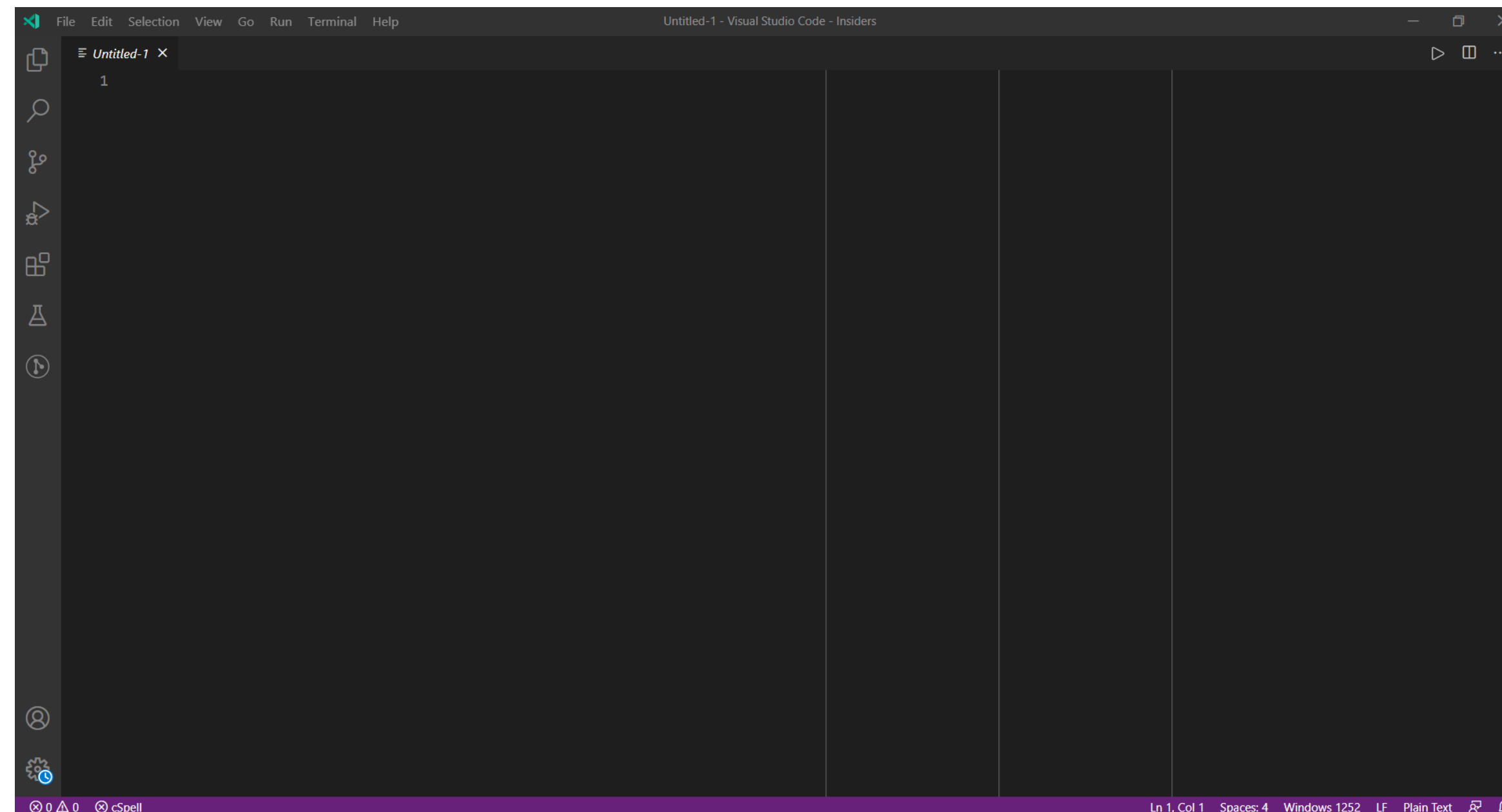




Utilizando o VSCode

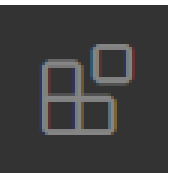
Iniciando o VSCode

- **Se essa é a primeira vez que abre o VSCode:**
 - Garanta que você possui uma pasta com o projeto desejado (p.ex., **D:\Projetos\algoritmos**);
 - Abra o VSCode, e na tela inicial clique em **File > Open Folder**. Selecione a pasta que você criou e clique em **Selecionar Pasta**.



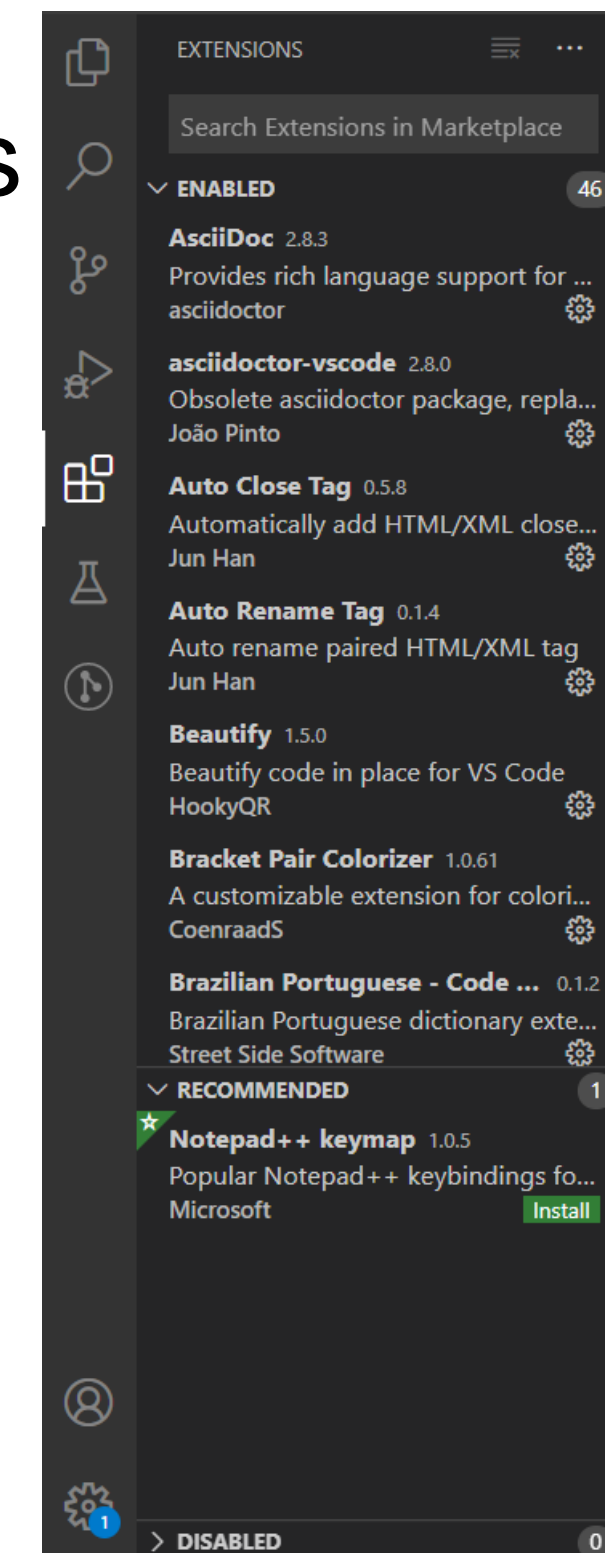
Iniciando o VSCode

- **Instalando extensões:**

- Boa parte do atrativo no VSCode é a possibilidade de instalar extensões e customizar a experiência de uso;
- Para instalar extensões, vá na coluna à esquerda e clique no ícone  ;
- Uma barra de extensões vai aparecer, e você poderá buscar as extensões desejadas e instalá-las;

- Abaixo seguem algumas sugestões de extensões:

- **Beautify**
- **Bracket Pair Colorizer**
- Brazilian Portuguese - Code Spell Checker
- C/C++
- GitLens - Git supercharged
- Git History Diff
- Git History
- Gitconfig Syntax
- Markdownlint
- Partial Diff
- **Python**
- **Visual Studio IntelliCode**
- vscode-python-docstring
- AsciiDoc



Iniciando o VSCode

- **Atalhos interessantes no VSCode:**
 - Ctrl + K, Ctrl + O: Abre uma pasta
 - Ctrl + D: Quando uma palavra estiver selecionada, seleciona todas as palavras no arquivo
 - Ctrl + F: Procura por uma palavra ou sentença no arquivo
 - Ctrl + Shift + F: Procura por uma palavra ou sentença em todos os arquivos da pasta
 - Ctrl + H: Substitui uma palavra ou sentença por outra em todo o arquivo
 - Alt + ↓ ou Alt + ↑: Move a linha inteira para baixo ou para cima
 - Shift + Alt + ↓ ou Shift + Alt + ↑: Copia a linha inteira para baixo ou para cima
 - Ctrl + Alt + ↓ ou Ctrl + Alt + ↑: Inclui um ou mais cursores nas linhas abaixo ou acima

Iniciando o VSCode

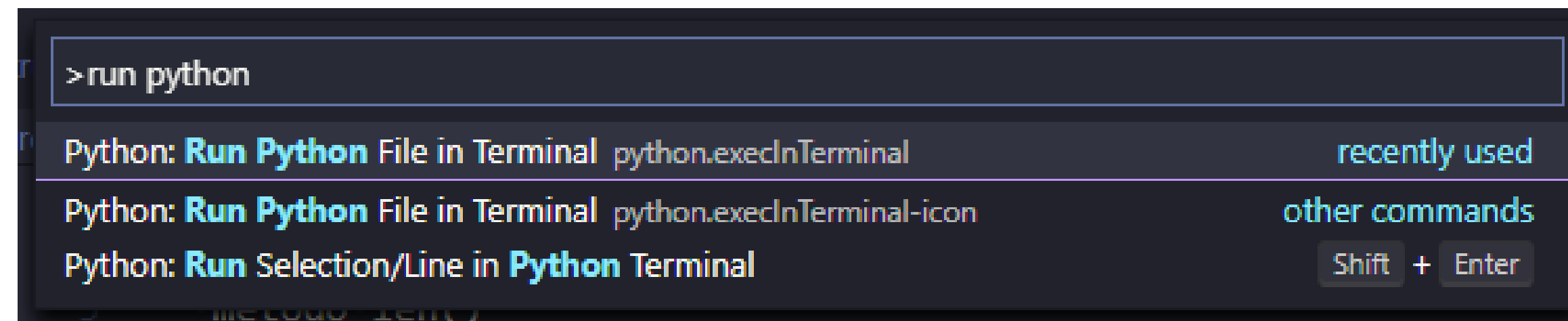
- **Atalhos interessantes no VSCode:**
 - F12: Vai para a declaração de uma variável ou função
 - F12 F12: Mostra todos os usos de uma variável ou função
 - Alt + → ou Alt + ←: Retrocede ou avança na última posição do cursor
 - Ctrl + S: Salva um arquivo
 - Ctrl + P: Abre um arquivo diretamente
 - Ctrl + ,: Abre o menu de configurações
 - Ctrl + Shift + P: Abre uma dropdown com opções de configuração
 - Ctrl + `: Abre a janela do terminal

Iniciando o VSCode

- **Atalhos interessantes no VSCode:**
 - Ctrl + G: Vai para uma linha específica do arquivo
 - Ctrl +]: Divide a tela em duas
 - Ctrl + 1 (ou 2, 3, 4): Seleciona um painel específico
 - Alt + Shift + 0: Alterna entre divisão na horizontal ou na vertical
 - Ctrl + F4 ou Ctrl + W: Fecha o arquivo selecionado (se for um painel selecionado e ele estiver vazio, fecha o painel)
 - Ctrl + ;: Comenta a(s) linha(s) selecionada(s)

Iniciando o VSCode

- **Executando um código Python no VSCode:**
 - Aperte as teclas **Ctrl + Shift + P**;
 - Na caixa de busca que aparece, digite “Run Python File in Terminal”, selecione a opção adequada e aperte **Enter**;
 - O VSCode deve rodar o código no terminal do próprio IDE.





Configurando o pylint

Configurando o pylint

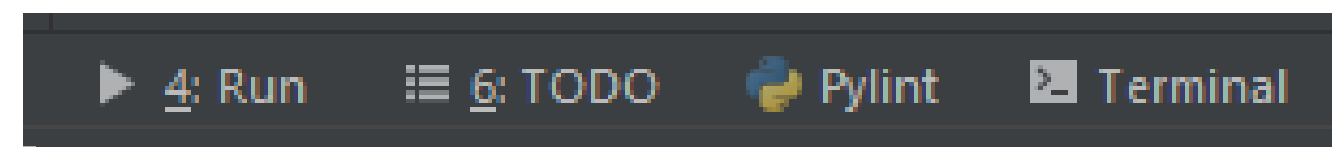
- O pylint é um pacote do Python que auxilia na adequação do código aos padrões de programação da comunidade Python
- Para instalar o pylint...
 - ...para MacOS veja [aqui](#)
 - ...para Windows veja [aqui](#)
- Para executar o pylint basta entrar com o comando **pylint <caminho>**, com o caminho do arquivo que deseja rodar o linter.

```
C:\Users\vmachado>pylint D:\Victor\Pessoal\IBMEC\2020.2\ALG\Aulas\teste.py
***** Module teste
D:\Victor\Pessoal\IBMEC\2020.2\ALG\Aulas\teste.py:1:0: C0114: Missing module docstring (missing-module-docstring)

-----
Your code has been rated at 0.00/10
```

Configurando o pylint

- O pylint ainda pode ser rodado direto no terminal do VSCode ou do PyCharm, basta abrir o terminal e seguir os mesmos passos mencionados anteriormente. No entanto, os ambos os IDEs fornecem meios de utilizar o pylint diretamente durante a implementação do código.
- **Usando o pylint no PyCharm:**
 - O PyCharm já possui a inspeção automática do editor, após configurar o interpretador. Para identificar as inspeções vá em **File > Settings** e na janela clique em **Editor** e depois em **Inspections**;
 - O PyCharm também possui um plugin. Em **File > Settings**, clique em **Plugins** e busque por **Pylint**. Instale e reinicie o IDE. Sempre que abrir um arquivo Python o IDE vai incluir a opção “Pylint” no canto inferior esquerdo. Para executar basta clicar na opção e depois em “Run”.



Configurando o pylint

- **Usando o pylint no VSCode:**

- Com a extensão “Python” instalada no VSCode e o pacote pylint instalado, abra o menu de configurações (use o atalho **Ctrl + ,** ou vá em **File > Preferences > Settings**) e procure por python linting;
- Marque as seguintes opções:
 - Python > Linting: Enabled
 - Python > Linting: Lint On Save
 - Python > Linting: Pylint Enabled
- Para rodar o pylint sem ser pelo terminal, use o atalho **Ctrl + Shift + P** e em seguida digite “Run Linting” e aperte Enter. O IDE vai marcar no código os problemas.

Python > Linting: Enabled

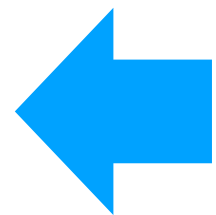
☒ Whether to lint Python files.

Python > Linting: Lint On Save

☒ Whether to lint Python files when saved.

Python > Linting: Pylint Enabled

☒ Whether to lint Python files using pylint.




Instalando pacotes pelo PyPI

Instalando pacotes pelo PyPI

- Uma das grandes vantagens ao se programar em Python é ter à disposição uma gama de pacotes e bibliotecas disponíveis pela própria comunidade, que desenvolve novas funcionalidades e distribui online, na maioria das vezes de forma gratuita.
- Um dos locais mais confiáveis e mais simples de se obter um novo pacote é através do PyPI, um repositório oficial de pacotes da linguagem, mantido pela própria organização que mantém o Python.
- O PyPI é acessado utilizando uma ferramenta chamada **pip**, que é instalada automaticamente ao se instalar o interpretador de Python. Portanto, a instalação de novos pacotes é muito fácil de se realizar, com apenas uma linha de comando.

Instalando pacotes pelo PyPI

- Instalando pacotes via pip no Windows:
 - Clique no botão do Windows ;
 - Digite a caixa de pesquisa **prompt de comando**, e abra o programa;
 - No terminal entre com o seguinte comando:

```
C:\Users\vmachado>C:\Python\python.exe -m pip install pylint
```
- Lembre-se de substituir o caminho do Python para o caminho instalado no seu computador, e altere **pylint** para o pacote escolhido.

Instalando pacotes pelo PyPI

- A instalação do Python no MacOS não costuma vir com o pip. Caso não tenha vindo, tente fazer os passos abaixo primeiro:

- Abra o terminal (pasta **Applications > Utilities > Terminal**);
- Entre com os seguintes comandos:

```
curl https://bootstrap.pypa.io/get-pip.py > get-pip.py  
sudo python get-pip.py
```

- Para instalar um pacote via pip no MacOS:
 - Usando o mesmo terminal usado para instalar o pip, entre com o seguinte comando:

```
sudo pip install <nome_do_pacote>
```

- Veja o vídeo abaixo para mais detalhes:
 - <https://www.youtube.com/watch?v=yBdZZGPpYxg>



Ciência de Dados – Conceitos Iniciais

A Onipresença das Oportunidades de Dados

- Atualmente, a informação está amplamente disponível em eventos externos, como tendências de mercado, notícias industriais e os movimentos dos concorrentes.
- Ampla disponibilidade de dados → aumento no interesse de métodos para extrair informações úteis e conhecimento a partir de dados.
- Hoje, empresas em quase todos os setores estão focadas em explorar esses dados para obter vantagem competitiva. Adicionalmente, as empresas precisam transformar os departamentos de análises de dados, que antes focavam em equipes de estatísticos e modeladores para a execução de cálculos e análises manuais.

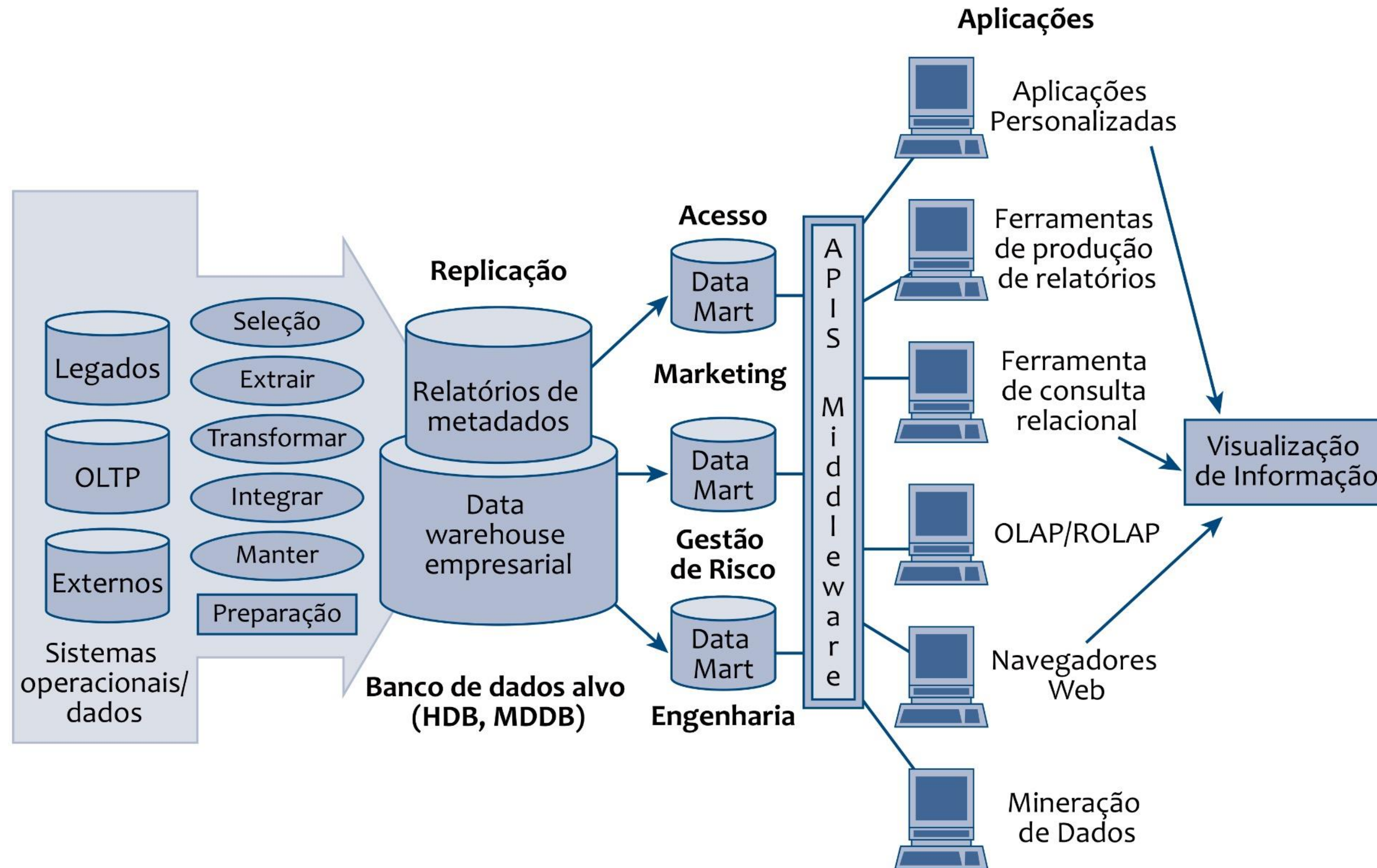
A Onipresença das Oportunidades de Dados

- Possíveis aplicações de técnicas de mineração de dados:
 - Marketing direcionado
 - Publicidade online
 - Recomendações de venda cruzada
 - Gestão de relacionamento com o cliente
 - Classificação e negociação de crédito
 - Detecção de fraude
 - Otimização de processos logísticos
 - Compra e venda de ações

A Onipresença das Oportunidades de Dados

- Exemplo:
 - Série **A Era dos Dados**, Netflix, 2020
 - Temporada 1, Episódio 1: Monitoramento
 - 2m20s a 10m00s

De dado para conhecimento



Fonte: Turban et al (2009), p. 61.

Ciência de Dados e Tomada de Decisão Orientada em Dados

- Ciência de Dados envolve princípios, processos e técnicas para compreender fenômenos por meio da análise (automatizada) de dados.
- Tomada de decisão orientada por dados (DOD) refere-se à prática de basear as decisões na análise dos dados, em vez de apenas na intuição.
- Por exemplo, ao invés de selecionar anúncios baseados puramente na intuição do dono do negócio, a escolha pode ser feita a partir da análise dos dados sobre a forma como os consumidores reagem a diferentes anúncios.
- São dois tipos de decisões mais relevantes para a área:
 - Decisões para as quais “descobertas” precisam ser feitas nos dados;
 - Decisões que se repetem, principalmente em grande escala.

Ciência de Dados e Tomada de Decisão Orientada em Dados



- Caso Target:
 - Os tomadores de decisão sabiam que a chegada de um bebê na família é um momento em que as pessoas mudam significativamente os hábitos de compras.
 - “Assim que percebemos que estão comprando nossas fraldas, eles comprarão todo o resto também.”
 - Não satisfeita, a Target decidiu identificar se seria possível **prever** se as pessoas estavam **esperando** um bebê.
 - Analisando os dados de mulheres que tiveram bebês, observaram que mulheres grávidas tendem a mudar o guarda-roupa e a dieta, além de comprarem vitaminas específicas.
 - A partir daí, foi possível começar a prever quando um bebê estaria chegando na família, permitindo a antecipação de vendas.

Processamento de Dados e “Big Data”

- Atualmente, muitas habilidades, sistemas e tecnologias de processamento de dados são, muitas vezes, erroneamente lançados como data science.
- A ciência de dados precisa ter acesso aos dados e, muitas vezes, beneficia-se da sofisticada engenharia de dados que as tecnologias de processamento podem facilitar, mas essas não são tecnologias de data science propriamente ditas.
- Recentemente, tecnologias “Big Data” têm recebido considerável atenção da mídia. Essencialmente, o termo *big data* significa conjunto de dados que são grandes demais para os sistemas tradicionais de processamento e, portanto, exigem novas tecnologias para processá-los.
- Ocasionalmente, tecnologias de big data são, na verdade, utilizadas para *implementar* as técnicas de mineração de dados. Essas tecnologias também podem ser utilizadas para apoio às técnicas de mineração de dados e outras atividades de ciência de dados.

De problemas de negócios a tarefas de mineração de dados

- Nos negócios, cada problema de tomada de decisão orientada em dados é exclusivo, composto por sua própria combinação de metas, desejos, limitações e até mesmo personalidades.
- Contudo, há conjuntos de tarefas comuns que permeiam os problemas de negócios, que são decompostos em subtarefas. As soluções para as subtarefas podem, então, ser compostas para resolver o problema geral.
- Apesar do grande número de algoritmos específicos de mineração de dados desenvolvidos ao longo dos anos, há apenas um punhado de tipos de tarefas fundamentalmente diferentes tratadas por esses algoritmos.
- Veremos alguns desses tipos de tarefas ao longo do curso.

De problemas de negócios a tarefas de mineração de dados

- *Classificação e estimativa de probabilidade* de classe tentam prever, para cada indivíduo de uma população, a que (pequeno) conjunto de classes este indivíduo pertence. Ex.: “Entre todos os clientes da empresa X, quais são suscetíveis de responder a determinada oferta?”
- *Regressão* (“estimativa de valor”) tenta estimar ou prever, para cada indivíduo, o valor numérico de alguma variável. Ex.: “Quanto um determinado cliente usará do serviço?”
- *Combinação por similaridade* tenta *identificar* indivíduos semelhantes com base nos dados conhecidos sobre eles. Ex.: “Quais empresas semelhantes aos nossos melhores clientes comerciais são potenciais novos clientes?”
- *Agrupamento* tenta *reunir* indivíduos de uma população por meio de sua similaridade, mas não é motivado por nenhum propósito específico. Ex.: “Nossos clientes formam grupos naturais ou específicos?”

De problemas de negócios a tarefas de mineração de dados

- *Agrupamento de coocorrência* (também conhecido como mineração de conjunto de itens frequentes, descoberta da regra de associação e análise de portfólio de ações) tenta encontrar *associações* entre entidades com base em transações que as envolvem. Ex.: “Quais itens são comumente comprados juntos?”
- *Perfilamento* (também conhecido como descrição de comportamento) tenta caracterizar o comportamento típico de um indivíduo, grupo ou população. Ex.: “Qual é o uso típico de celular nesse segmento de cliente?”
- *Previsão de vínculo* tenta prever ligações entre itens de dados, geralmente sugerindo que um vínculo deveria existir e, possivelmente, também estimando a força do vínculo. Ex.: “Como você e Karen compartilham 10 amigos, talvez você gostaria de ser amigo de Karen?”

De problemas de negócios a tarefas de mineração de dados

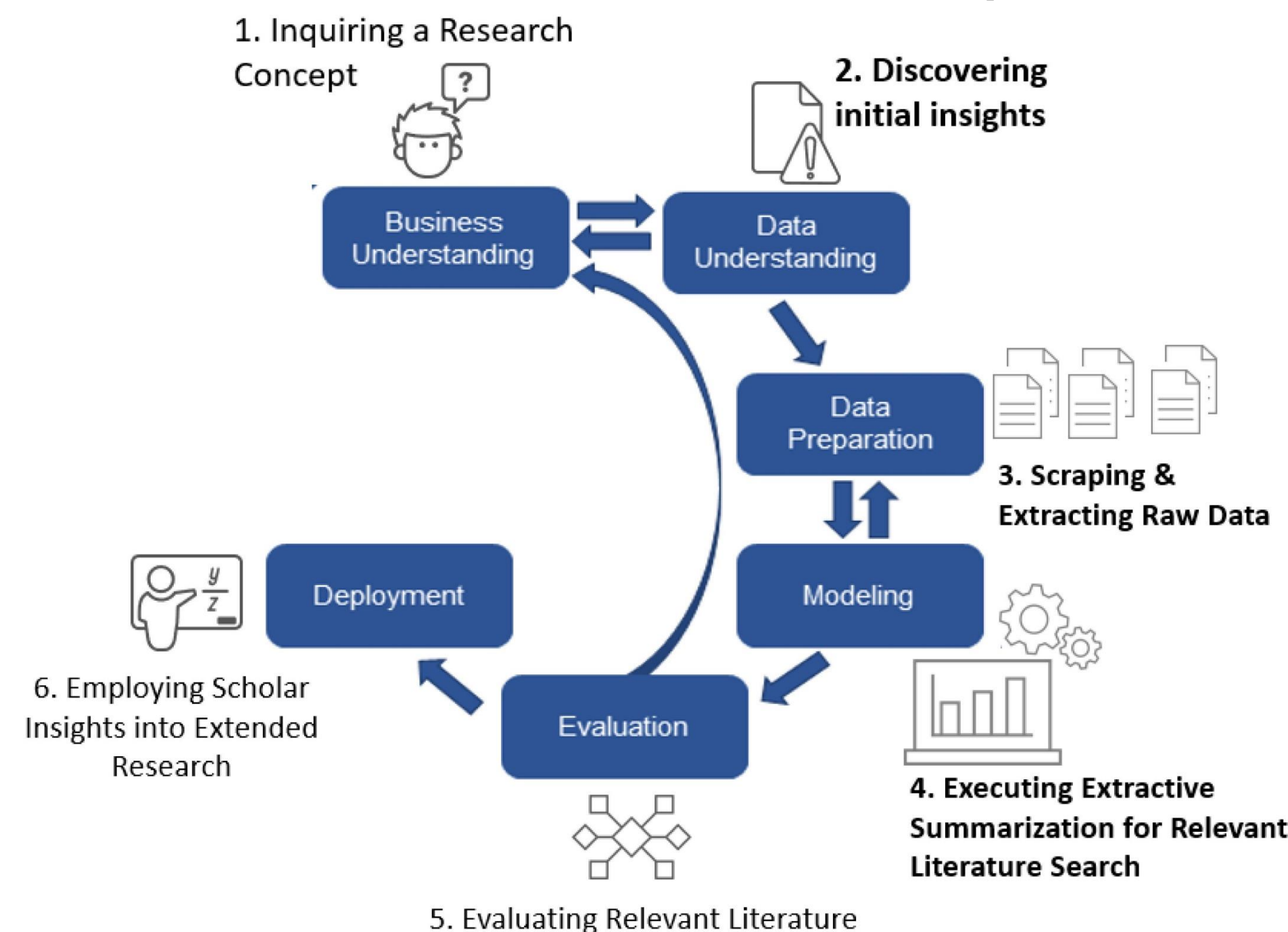
- *Redução de dados* tenta pegar um grande conjunto de dados e substituí-lo por um conjunto menor que contém grande parte das informações importantes do conjunto maior. Ex.: Um enorme conjunto de dados sobre preferências de filmes dos consumidores pode ser reduzido a um conjunto de dados muito menor revelando os gostos do consumidor mais evidentes na visualização de dados.
- *Modelagem causal* tenta nos ajudar a compreender que acontecimentos ou ações realmente influenciam outras pessoas. Ex.: “Os anúncios que fizemos influenciaram os consumidores a comprar?”

Métodos supervisionados vs não supervisionados

- Definição de **alvo**: objetivo específico a se estudar, com base em uma massa de dados.
- Métodos supervisionados:
 - Contemplam um objetivo específico (alvo) para o agrupamento (previsão)
 - Dispõem de dados sobre o alvo – normalmente obtidos de uma observação histórica
 - Ex.: “Podemos encontrar grupos de clientes com maior probabilidade de cancelarem seus serviços logo após o vencimento dos contratos?”
- Métodos não supervisionados
 - Não possui um objetivo específico
 - Ex.: “Nossos clientes naturalmente se encaixam em grupos diferentes?”

O processo de mineração de dados

- A mineração de dados é uma arte. Ela envolve a aplicação de uma quantidade substancial de ciência e tecnologia, mas a aplicação adequada ainda envolve arte também.
- Mas, como acontece com muitas artes maduras, existe um processo bem compreendido que coloca uma estrutura no problema, permitindo consistência, repetitividade e objetividade razoáveis. Um exemplo é o apresentado abaixo.



O processo de mineração de dados

- Compreensão do negócio
 - Inicialmente, é vital compreender o problema a ser resolvido. Isso pode parecer óbvio, mas projetos de negócios raramente vêm pré-moldados como problemas claros e inequívocos de mineração de dados.
 - Muitas vezes, reformular o problema e projetar uma solução é um processo repetitivo de descoberta. O processo apresentado representa isso como ciclos dentro de um ciclo, em vez de um simples processo linear.
 - A fase de compreensão do negócio representa uma parte da arte onde a análise da criatividade desempenha um grande papel.
 - Nesta primeira etapa, a equipe de projeto deve pensar cuidadosamente sobre o cenário de uso e o problema a ser resolvido.

O processo de mineração de dados

- Compreensão dos dados
 - Se a solução do problema de negócios é o objetivo, os dados compreendem a matéria-prima disponível a partir da qual a solução será construída.
 - É importante entender os pontos fortes e as limitações dos dados porque raramente há uma correspondência exata com o problema. Os dados históricos, muitas vezes, são recolhidos para fins não relacionados com o problema de negócio atual ou para nenhum propósito explícito.
 - Também é comum os *custos* dos dados variarem. Alguns dados estarão disponíveis praticamente de graça, enquanto outros exigirão um esforço para serem obtidos. Alguns dados podem ser comprados. Ainda outros simplesmente não existem e exigirão projetos auxiliares para organizar sua coleção.

O processo de mineração de dados

- Preparação dos dados
 - As tecnologias analíticas que podemos utilizar são poderosas, mas impõem determinados requisitos sobre os dados que usam. Com frequência, elas exigem que os dados estejam em uma forma diferente de como são fornecidos naturalmente, e alguma conversão será necessária.
 - Exemplos típicos de preparação de dados são sua conversão para o formato tabular, removendo ou inferindo valores ausentes, e convertendo dados para diferentes tipos.
 - Algumas técnicas de mineração de dados são projetadas para dados simbólicos e categóricos, enquanto outras lidam apenas com valores numéricos. Além disso, valores numéricos devem, muitas vezes, ser normalizados ou dimensionados de forma que sejam comparáveis.

O processo de mineração de dados

- Modelagem
 - Modelagem é o tema de algumas das nossas aulas, e o resultado da modelagem é algum tipo de modelo ou padrão que captura regularidades nos dados.
 - A etapa de modelagem é o principal local onde as técnicas de mineração de dados são aplicadas aos dados. É importante ter alguma compreensão das ideias fundamentais de mineração de dados, incluindo os tipos de técnicas e algoritmos existentes, porque esta é a parte da arte em que a maioria da ciência e da tecnologia podem ser exercidas.

O processo de mineração de dados

- Avaliação
 - O objetivo da fase de avaliação é estimar os resultados de mineração de dados de forma rigorosa e obter a confiança de que são válidos e confiáveis antes de avançar.
 - Se prestarmos bastante atenção em qualquer conjunto de dados encontraremos padrões, mas eles podem não sobreviver a um exame minucioso. Gostaríamos de ter a confiança de que os modelos e padrões extraídos dos dados são regularidades verdadeiras e não apenas anomalias de amostra.
 - Avaliar os resultados de mineração de dados inclui avaliações quantitativas e qualitativas. Vários investidores se preocupam com o processo de tomada de decisão nos negócios que será realizada ou apoiada pelos modelos resultantes. Em muitos casos, esses investidores precisam “aprovar” a implantação dos modelos e, para isso, precisam estar satisfeitos com a qualidade das decisões de tais modelos.

O processo de mineração de dados

- Implantação
 - Na implantação, os resultados da mineração de dados – e cada vez mais nas próprias técnicas de mineração de dados – são colocados em uso real, a fim de se constatar algum retorno sobre o investimento.
 - Os casos mais claros de implantação envolvem a implementação de um modelo preditivo em algum sistema de informação ou processo de negócios.
 - Cada vez mais, as próprias técnicas de mineração de dados são implantadas. Por exemplo, para direcionamento de anúncios online, os sistemas são implantados de forma a construir (e testar) automaticamente modelos em produção quando uma nova campanha publicitária é apresentada.
 - A implantação de um modelo no sistema de produção normalmente requer que o modelo seja recodificado para o ambiente de produção, geralmente para maior velocidade ou compatibilidade com um sistema existente.

Modelagem

- Um **modelo** é a especificação de uma relação matemática (ou probabilística) existente entre variáveis diferentes.
- Por exemplo, se você está tentando levantar dinheiro para o seu site de rede social, talvez você precise de um *modelo de negócios* (possivelmente em uma planilha) que receba entradas como “número de usuários”, “rendimento de propaganda por usuário” e “número de funcionários” e exiba como saída seu lucro anual pelos próximos anos.
- O modelo de negócios é, provavelmente, baseado em relações simples de matemática: lucro é o rendimento menos as despesas, o rendimento é a soma das unidades vendidas vezes o preço médio e assim por diante.

O que é Mineração de Dados?

- Todo mundo possui sua própria definição, mas usaremos *mineração de dados* para nos referir à criação e ao uso de modelos que são aprendidos *a partir dos dados*.
- Em outros contextos isso pode ser chamado de *modelo preditivo* ou *aprendizado de máquina*, mas vamos manter a mineração de dados.
- Normalmente, nosso objetivo será usar os dados existentes para desenvolver modelos que possamos usar para **prever** possíveis saídas para os dados novos, como:
 - Prever se uma mensagem de e-mail é spam ou não;
 - Prever se uma transação do cartão de crédito é fraudulenta;
 - Prever qual a probabilidade de um comprador clicar em uma propaganda;
 - Prever qual time de futebol ganhará o campeonato brasileiro.

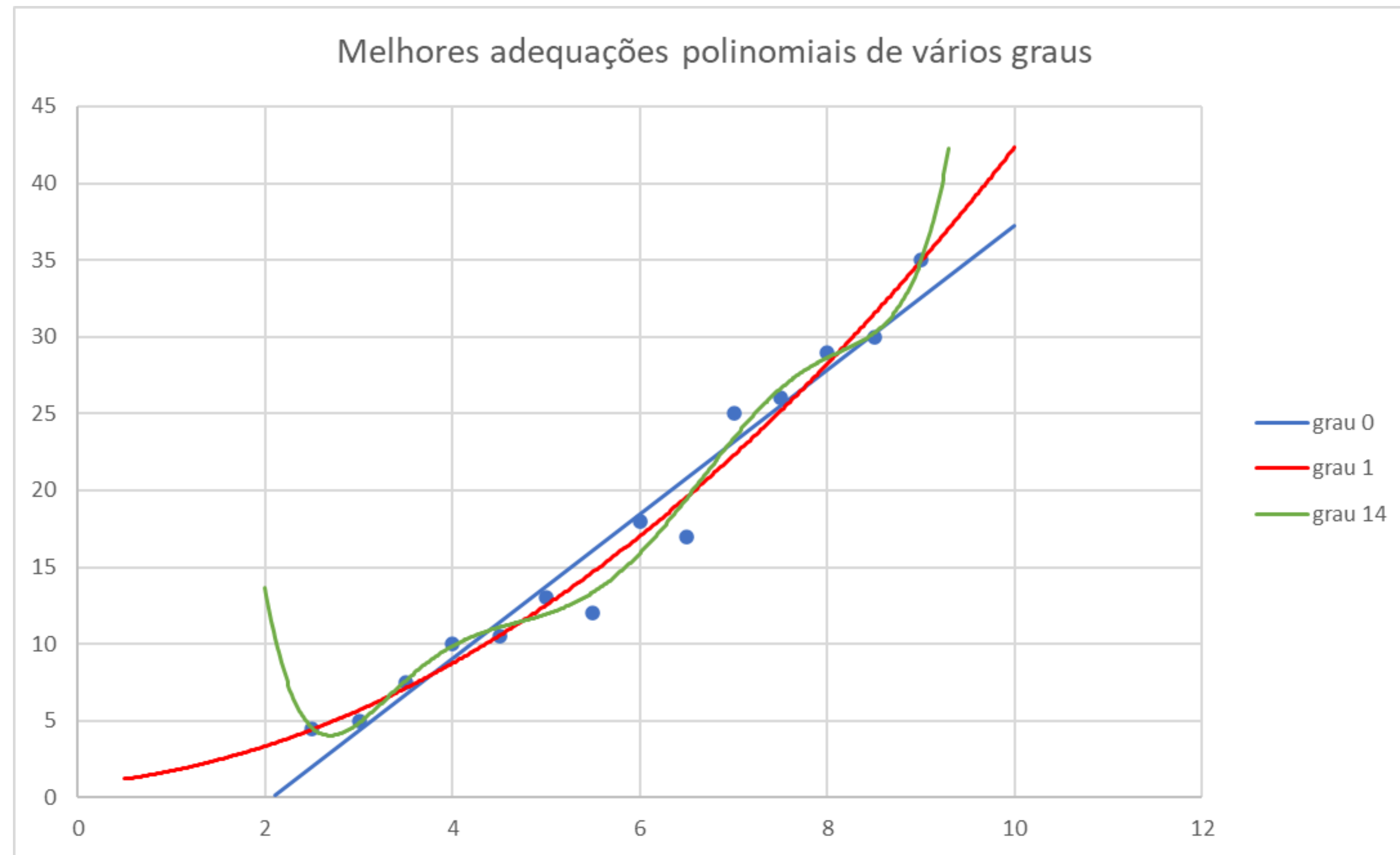
O que é Mineração de Dados?

- Consideraremos os modelos *supervisionados* (nos quais existe um conjunto de dados etiquetados com a resposta correta para aprendizagem) e modelos *não supervisionados* (nos quais apenas alguns dados são etiquetados).
- Agora, até mesmo na situação mais simples existe um universo inteiro de modelos que podem descrever a relação na qual estamos interessados. Na maioria dos casos, nós mesmos escolheremos uma família *parametrizada* de modelos e então usaremos os dados para aprender parâmetros que são, de certa forma, ótimos.
- Por exemplo, podemos presumir que a altura de uma pessoa é (mais ou menos) uma função linear do seu peso e então usar os dados para descobrir qual função linear é essa. Ou, podemos presumir que uma árvore de decisão é uma boa maneira de identificar quais doenças nossos pacientes possuem e então usar os dados para descobrir a ótima árvore de decisão.

Sobreaajuste e Subajuste

- Um perigo comum em mineração de dados é o *sobreaajuste* – produzir um modelo de bom desempenho com os dados que você treina, mas que não lide muito bem com novos dados.
- Isso pode implicar o aprendizado com base no ruído dos dados. Ou, pode implicar em aprender a identificar entradas específicas em vez de qualquer fator que sejam de fato preditivos da saída desejada.
- O outro lado é o *subajuste*, produzindo um modelo que não desempenha bem nem com os dados usados no treino, apesar de que, quando acontece isso, normalmente se decide que o modelo não é bom o suficiente e o processo volta para procurar melhores.

Sobreajuste e Subajuste



Sobreajuste e Subajuste

- Os modelos que são muito complexo tendem ao sobreajuste e não lidam bem com dados além daqueles com os quais foram treinados.
- O método mais fundamental para se ter certeza de que os nossos modelos não são muito complexos envolve o uso de dados diferentes para treinar e testar o modelo.
- A maneira mais fácil de fazer isso é dividir o conjunto de dados, a fim de que, por exemplo, dois terços dele sejam usados para treinar o modelo e depois medir o desempenho do modelo com a parte restante.
- Com frequência, teremos uma matriz x de variáveis de entrada e um vetor y de variáveis de saída. Nesse caso, precisamos nos certificar de colocar os valores correspondentes tanto nos dados em treinamento como nos dados de teste.

Sobreajuste e Subajuste

- Se o modelo foi sobreajustado para os dados em treinamento, então ele deve desempenhar mal sobre os dados de teste (completamente separados). De outra maneira, se ele desempenha bem sobre os dados de teste, então você pode ficar mais confiantes que ele está *ajustado* em vez de *sobreajustado*.
- Caso faça parte do processo a determinação de qual *modelo* melhor se ajusta aos dados, recomenda-se dividir os dados em três partes: um conjunto de *treinamento* para construir modelos, um conjunto de *validação* para escolher entre os modelos treinados e um conjunto de *teste* para avaliar o modelo final.

Precisão

- Vamos analisar a precisão do seguinte modelo de previsão que afirma o seguinte, para dados coletados nos EUA:

Um bebê recém-nascido desenvolverá leucemia no futuro se e somente se o seu nome for **Luke**.

- Esse modelo é usado para fazer uma avaliação *binária*. Dado um conjunto de dados etiquetados e um modelo preditivo, cada ponto de dados se estabelece em quatro categorias:
 - Positivo verdadeiro: “teve leucemia e previmos que ela teria corretamente”;
 - Positivo falso (erro tipo 1): “não teve leucemia, e previmos que ela teria”;
 - Negativo falso (erro tipo 2): “teve leucemia, mas previmos que ela não teria”;
 - Negativo verdadeiro: “não teve leucemia, e previmos que ela não teria”.

Precisão

- Representamos essa contagem em uma **matriz de confusão**:

	Teve leucemia	Não teve leucemia
Premissa “teve leucemia”	Positivo Verdadeiro	Positivo Falso
Premissa “não teve leucemia”	Negativo Falso	Negativo Verdadeiro

- Vamos aos dados:
 - Aproximadamente 3,8 bebês em cada 1000 se chamam Luke (<https://www.babycenter.com/baby-names-luke-2918.htm>);
 - A incidência de leucemia é de aproximadamente 1,4%, ou 14 de cada 1000 pessoas (<https://seer.cancer.gov/statfacts/html/leuks.html>).

Precisão

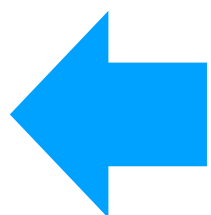
- Se acreditarmos que esses dois fatores são independentes e aplicar o teste “Luke para leucemia” em um milhão de pessoas, esperaríamos ver uma matriz de confusão como essa:

	Teve leucemia	Não teve leucemia	Total
Luke	53	3.747	3.800
Não Luke	13.947	975.853	996.200
Total	14.000	979.600	1.000.000

- Podemos usar isso para computar diversas estatísticas sobre o desempenho do modelo:
 - Acurácia**, ou fração de premissas corretas: 0,9759;
 - Exatidão**, ou precisão sobre as previsões positivas: 0,0139;
 - Sensibilidade**, ou fração dos positivos identificados pelo modelo: 0,0038;
 - Média harmônica** entre exatidão e sensibilidade: 0,0060.

Precisão

- Geralmente, a escolha de um modelo implica em um compromisso entre acurácia e sensibilidade. Um modelo que prevê “sim” quando se está um pouco confiante provavelmente terá uma sensibilidade alta mas uma acurácia baixa; um modelo que prevê “sim” somente quando está extremamente confiante provavelmente terá uma sensibilidade baixa e uma acurácia alta.
- Em outras palavras, dizer “sim” com muita frequência trará muitos positivos falsos; dizer “não” com muita frequência trará muitos negativos falsos.



Git

Introdução

- Talvez você reconheça esse nome por causa de sites como *github* ou *gitlab*. **Git** é um sistema *open-source* de controle de versionamento.
- Versionar projetos é uma prática essencial no mundo profissional e, em particular, na área de tecnologia, para manter um histórico de modificações deles, ou poder reverter alguma modificação que possa ter comprometido o projeto inteiro, dentre outras funcionalidades que serão aprendidas na prática.
- Os projetos são normalmente armazenados em **repositórios**.
- Vamos aqui falar de alguns conceitos principais sobre como funciona o sistema, independente da plataforma que vamos utilizar (p.ex., *github*). Em seguida vamos aplicar alguns desses conceitos na prática.

Para mais informações...

- O tempo que temos em aula não é suficiente para conseguirmos discutir todos os detalhes por trás dessa tecnologia. Portanto, seguem abaixo algumas sugestões de conteúdos extras para estudarem:
 - <http://git-scm.com/book/en/Getting-Started-About-Version-Control>
 - <http://git-scm.com/book/en/Getting-Started-Git-Basics>
 - <http://learngitbranching.js.org/>
 - <http://try.github.io/levels/1/challenges/1>

O que é Controle de Versões?

- Controle de Versões é um sistema de grava mudanças aplicadas em um arquivo ou um conjunto de arquivos ao longo do tempo, para que o usuário possa lembrar ou recuperar depois.
- Na área de tecnologia o controle de versões já é algo consolidado, uma vez que inúmeras alterações são aplicadas em um software durante a sua implementação e manutenção. No entanto, adotar um sistema de controle de versões (VCS, da sigla em inglês) é algo recomendado para qualquer área.
- Quando se quer adotar um VCS, normalmente o primeiro passo é trabalhar com um controle local, fazendo cópias dos arquivos e os renomeando com algum padrão (p.ex., incluindo a data ao final do nome do arquivo).
- O Git veio para otimizar esse controle de versões, permitindo inúmeras alterações que seriam muito complicadas se fossem feitas manualmente.

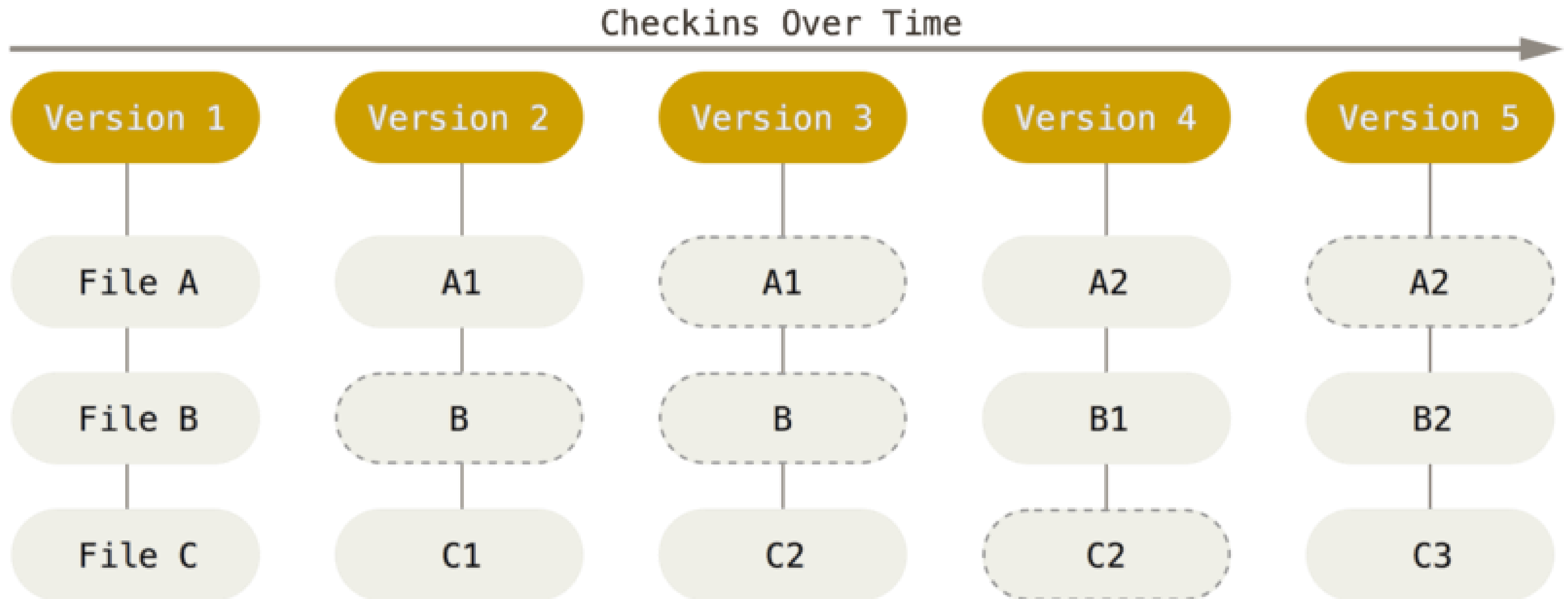
Uma breve história do Git

- A criação do Git está atrelada à criação do Linux, quando o time responsável utilizou uma solução de controle de versão que, eventualmente, tornou-se paga.
- A comunidade Linux, portanto, começou a planejar um sistema que aproveitasse algumas das características da solução anterior, porém evoluindo diversos aspectos.
- A comunidade tinha os seguintes objetivos para o novo sistema:
 - Velocidade
 - Design simples
 - Suporte forte para desenvolvimento não-linear, com milhares de atividades sendo realizadas em paralelo
 - Completamente distribuído, permitindo o acesso de qualquer lugar
 - Eficiente no suporte a grandes projetos

O que é Git?

- **Git** funciona pensando que os dados de um repositório compõem uma série de “fotografias” de um sistema de arquivos em miniatura.
- No Git, a cada vez que você aplica uma alteração (ou *commit*), ou salva o estado do seu projeto, o Git basicamente tira uma “foto” de como os arquivos do repositório estão naquele momento e então ele armazena uma referência a essa foto.
- Para ser eficiente, se os arquivos não foram alterados, o Git não altera os arquivos novamente, apenas um link para a última versão do arquivo armazenada.
- Sendo assim, o Git trabalha os dados como um **fluxo de fotografias**.

O que é Git?



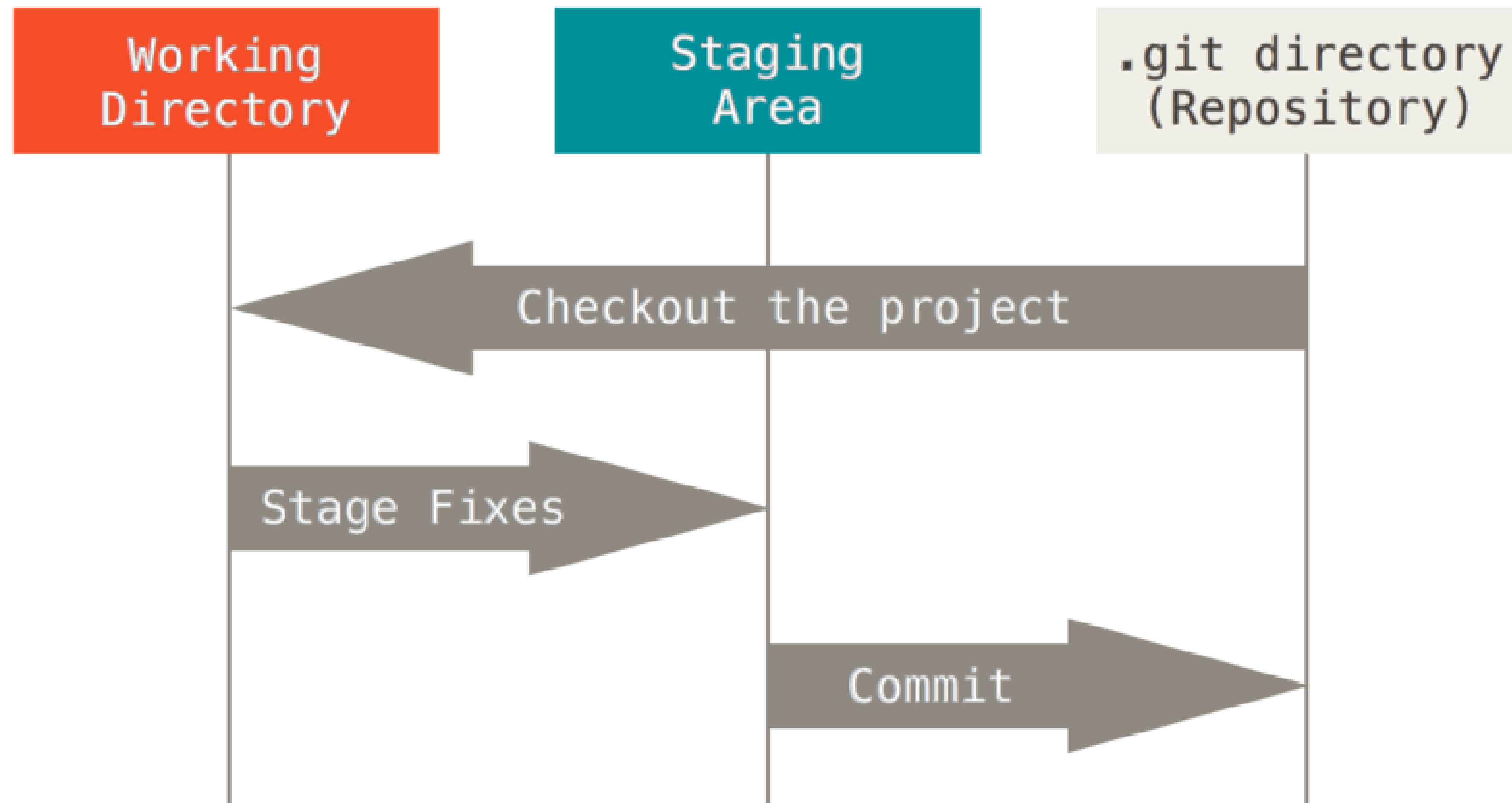
O que é Git?

- A maior parte das operações no Git precisa apenas de arquivos e recursos locais para operarem, e normalmente nenhuma informação é necessária de outro computador na rede. Isso fornece uma velocidade de operação que outros VCS não possuem. Como cada usuário possui todo o histórico do projeto no computador, a maioria das operações aparenta ser quase instantânea.
- Para todos os efeitos, na prática, o Git normalmente só acrescenta informações ao seu banco de dados, nunca removendo informações. É muito difícil, e não recomendado, gerar operações que removam informações, já que essas operações podem afetar o histórico do seu projeto.

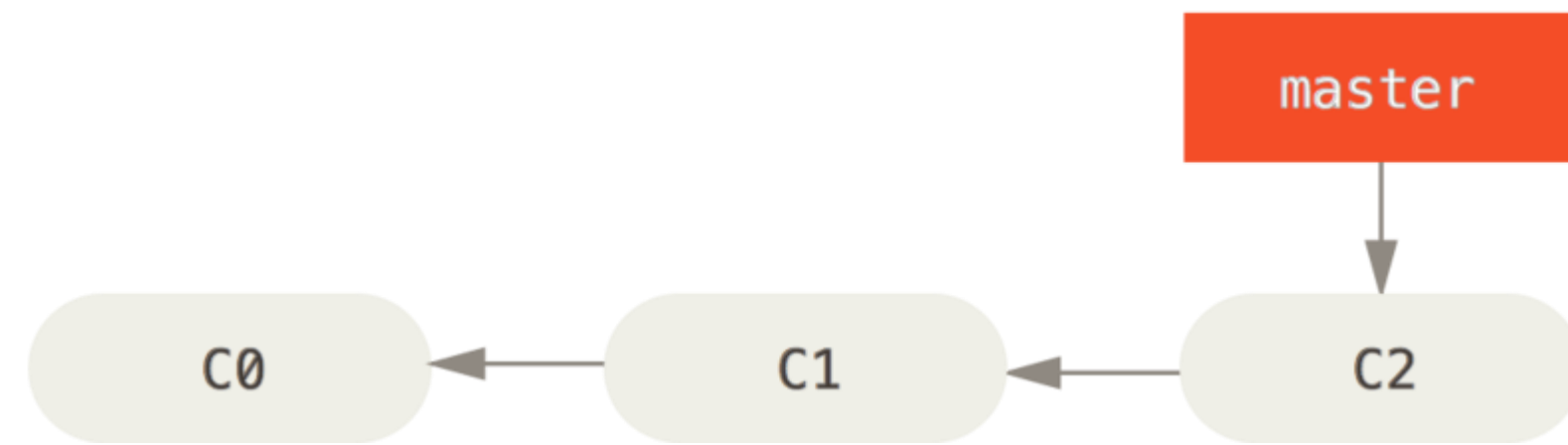
Os três estados

- O Git tem três estados principais nos quais os arquivos de um repositório podem se encontrar: **modificado**, **preparado** e “**commitado**”:
 - **Commitado** significa que os dados estão armazenados de forma segura em seu banco de dados local;
 - **Modificado** significa que você alterou o arquivo, mas ainda não fez o commit no banco de dados;
 - **Preparado** significa que você marcou a versão atual de um arquivo modificado para fazer parte do seu próximo commit.
- Isso leva a três seções principais de um projeto Git: o diretório Git, o diretório de trabalho e área de preparo.

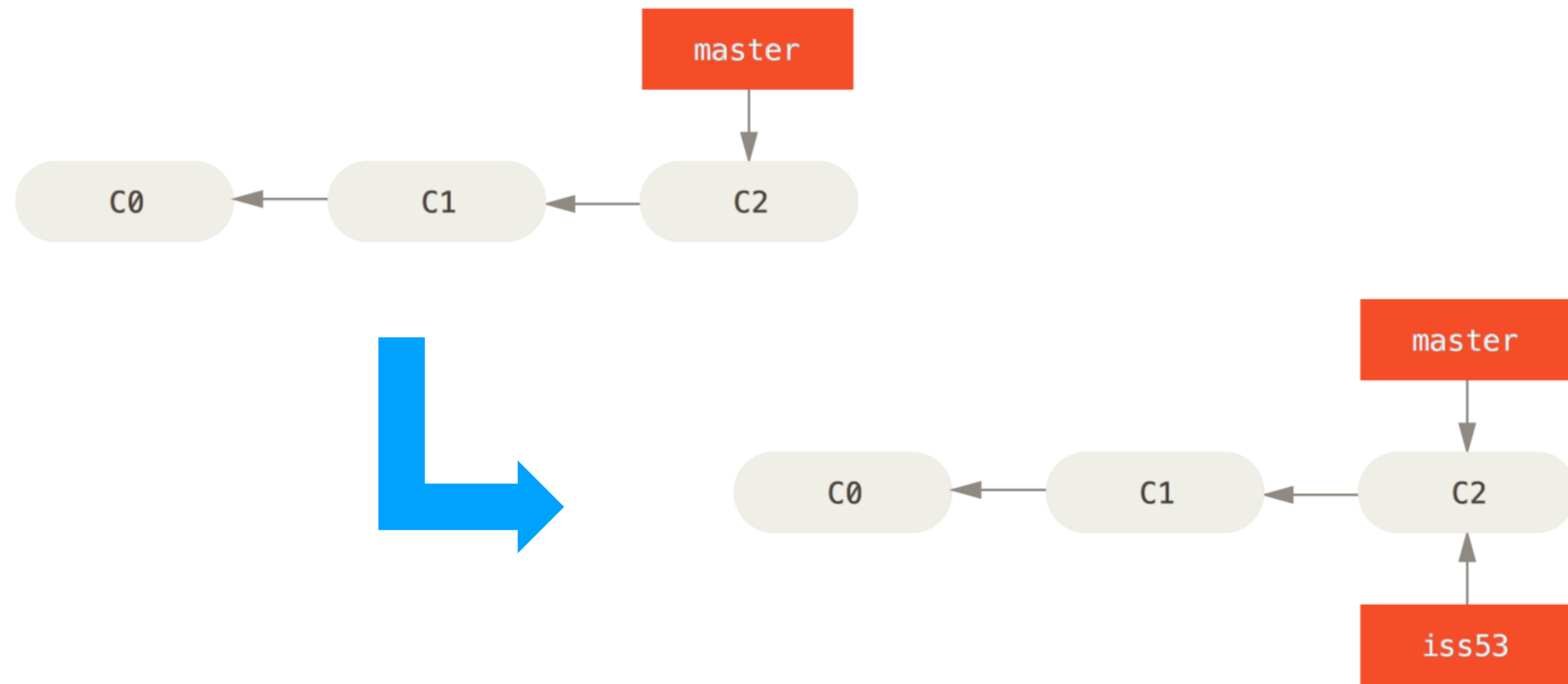
Os três estados



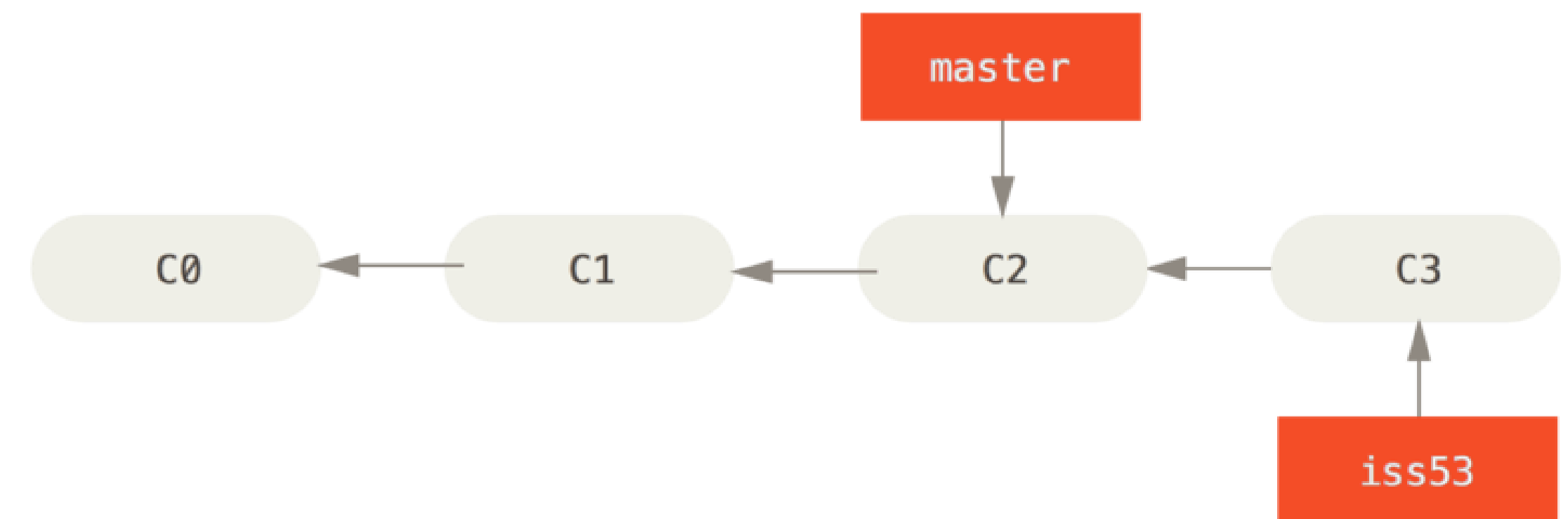
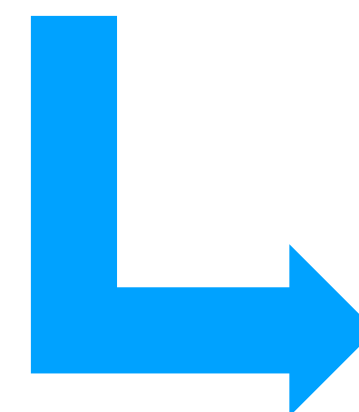
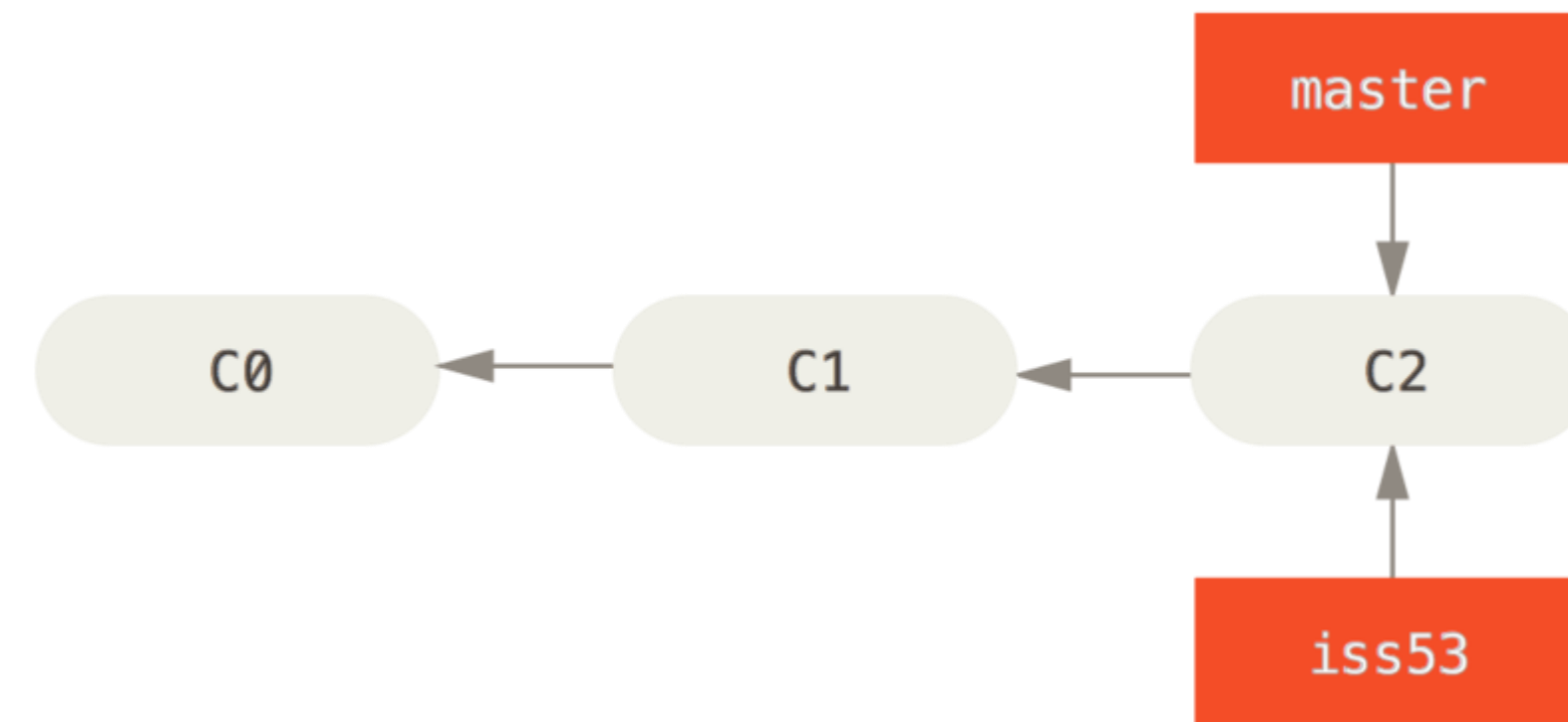
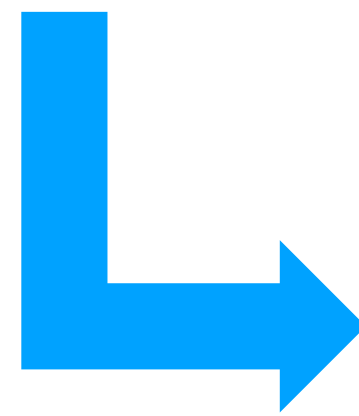
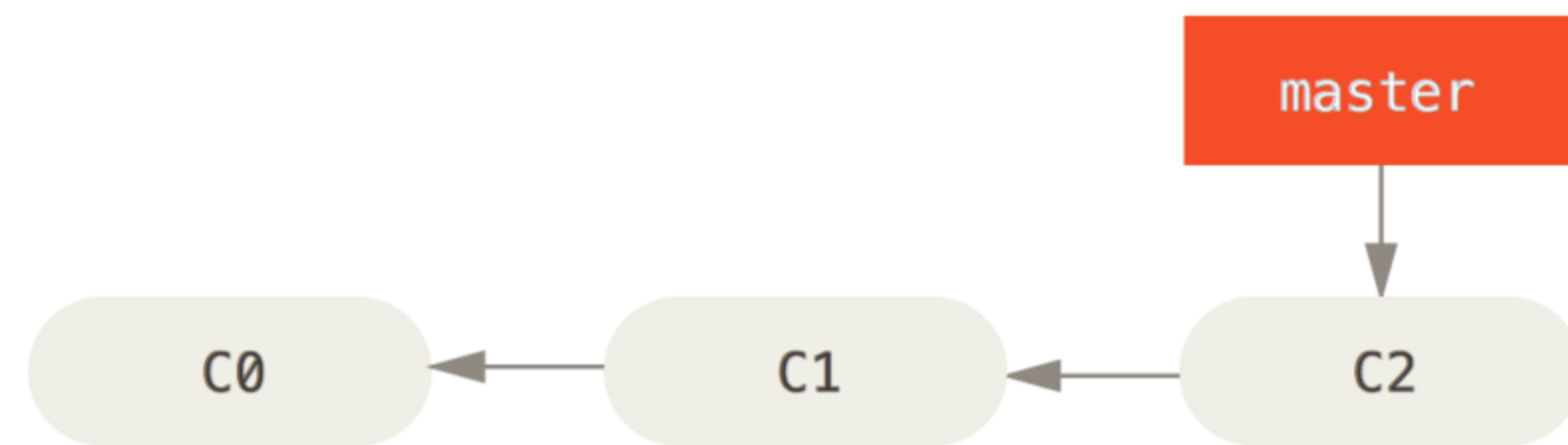
Branches no Git



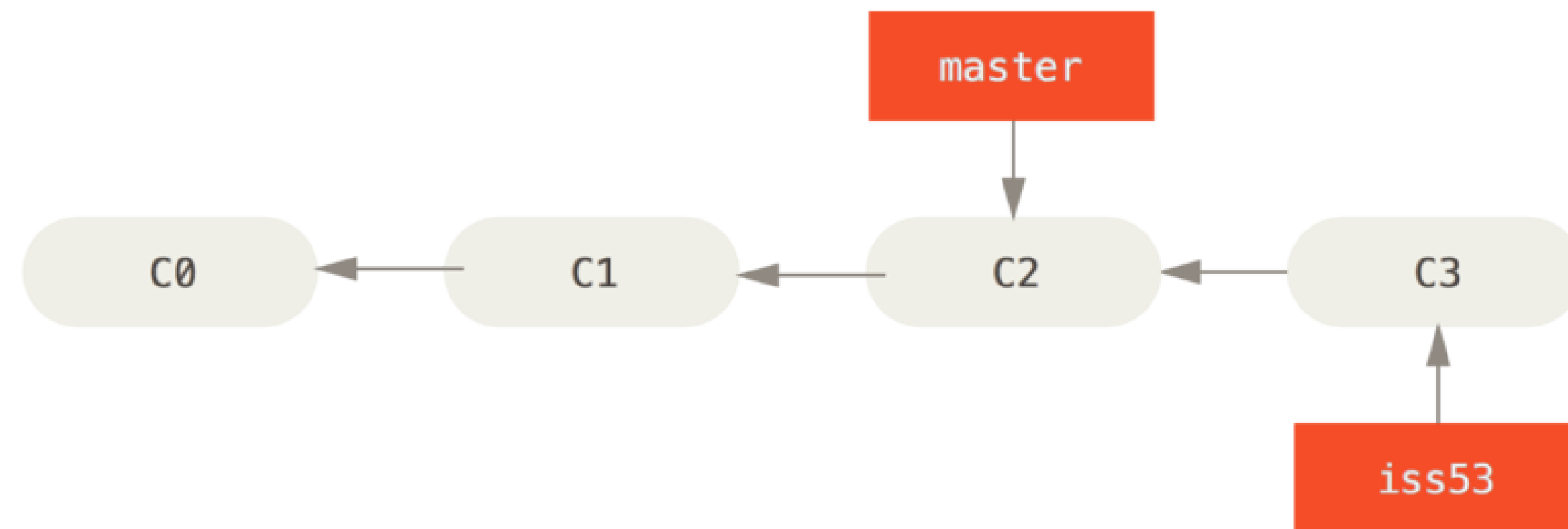
Branches no Git



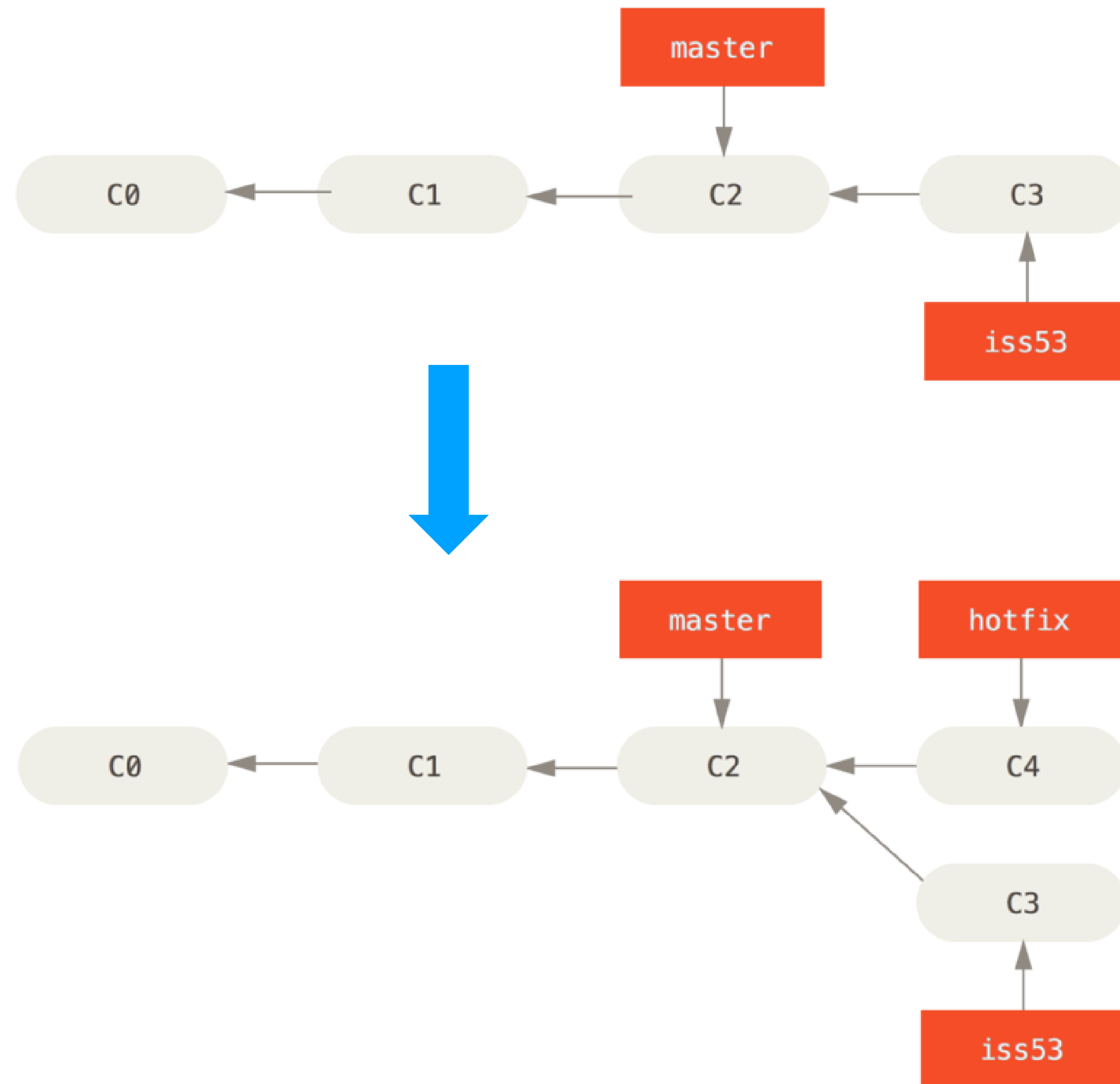
Branches no Git



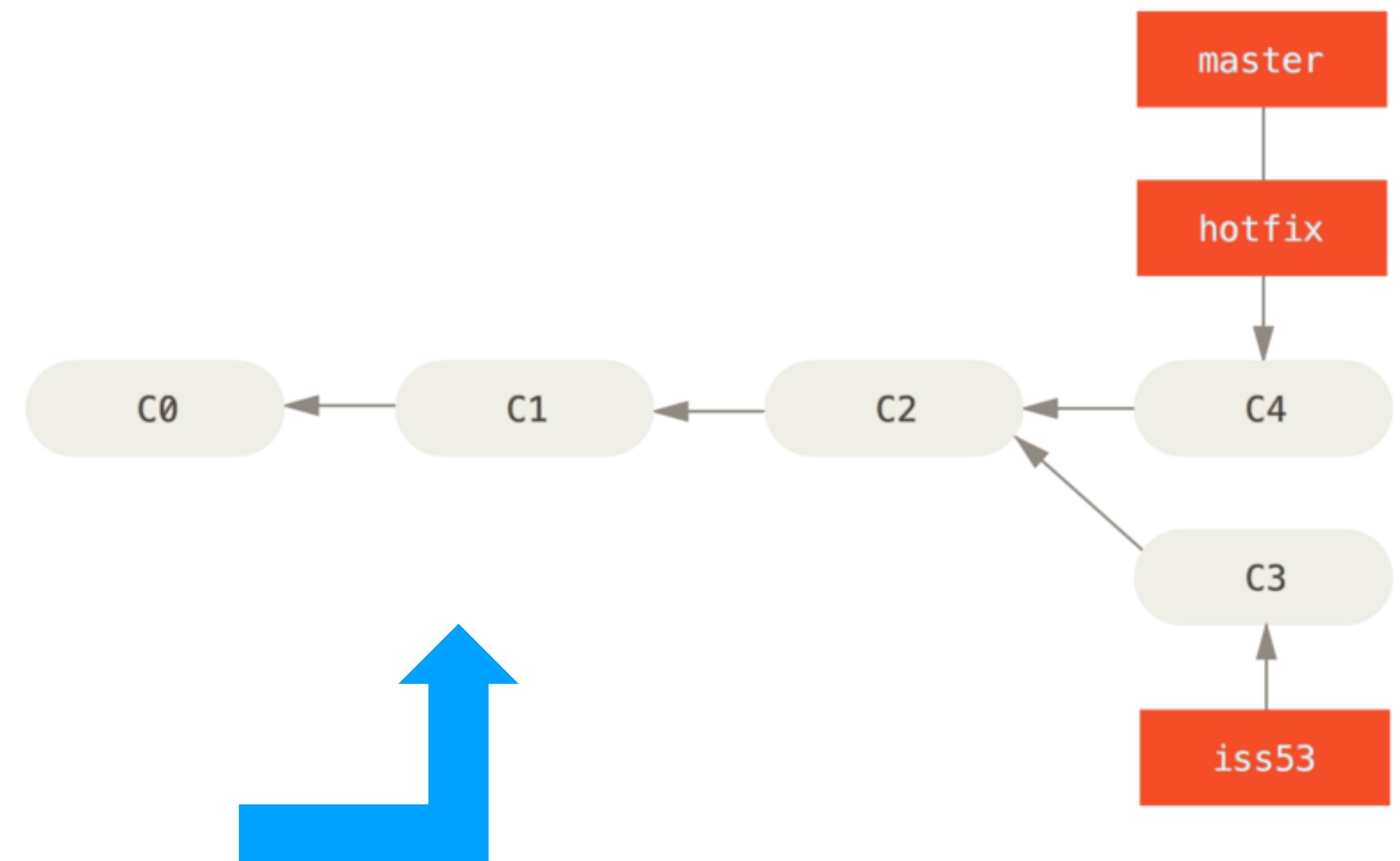
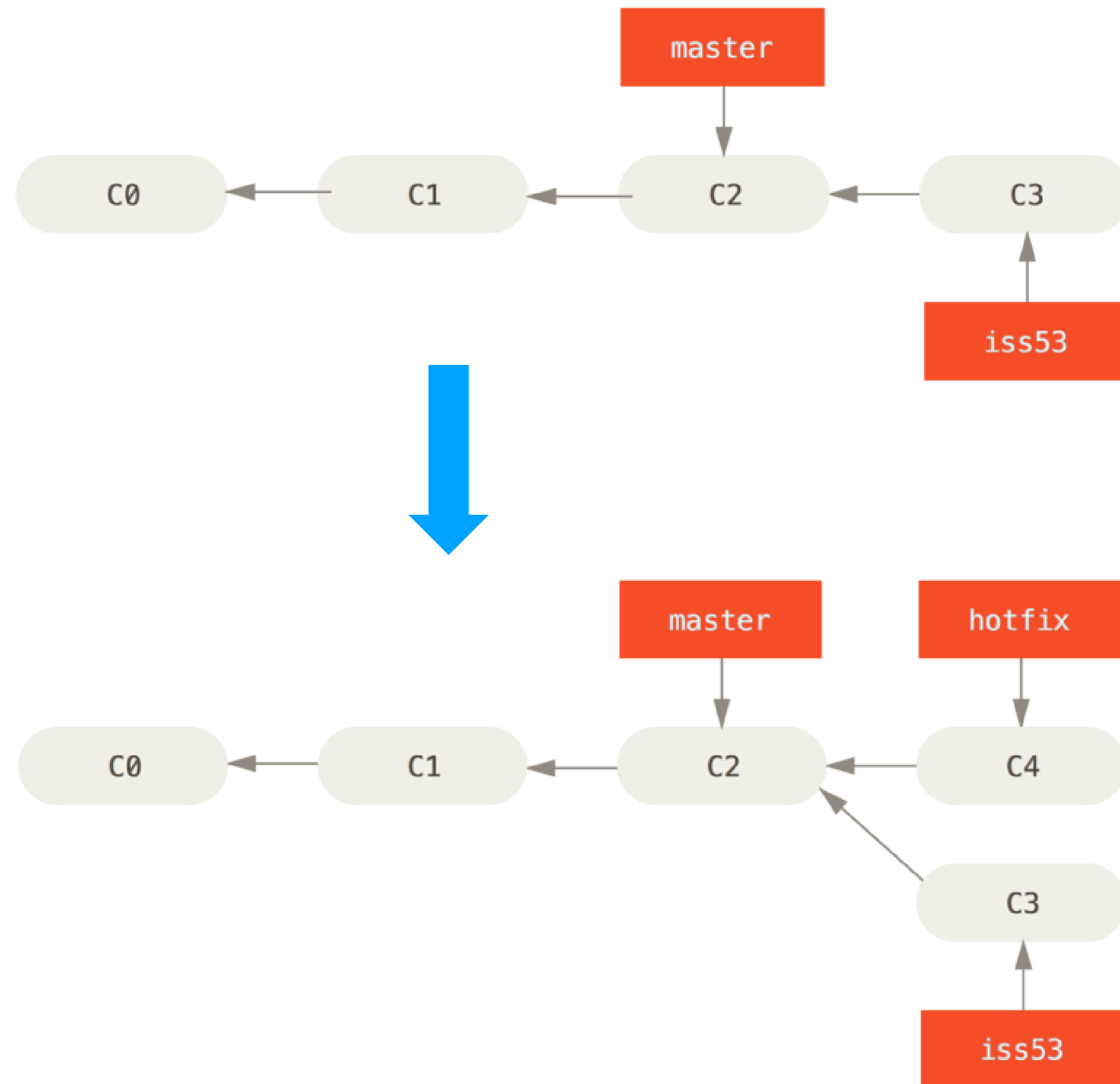
Branches no Git



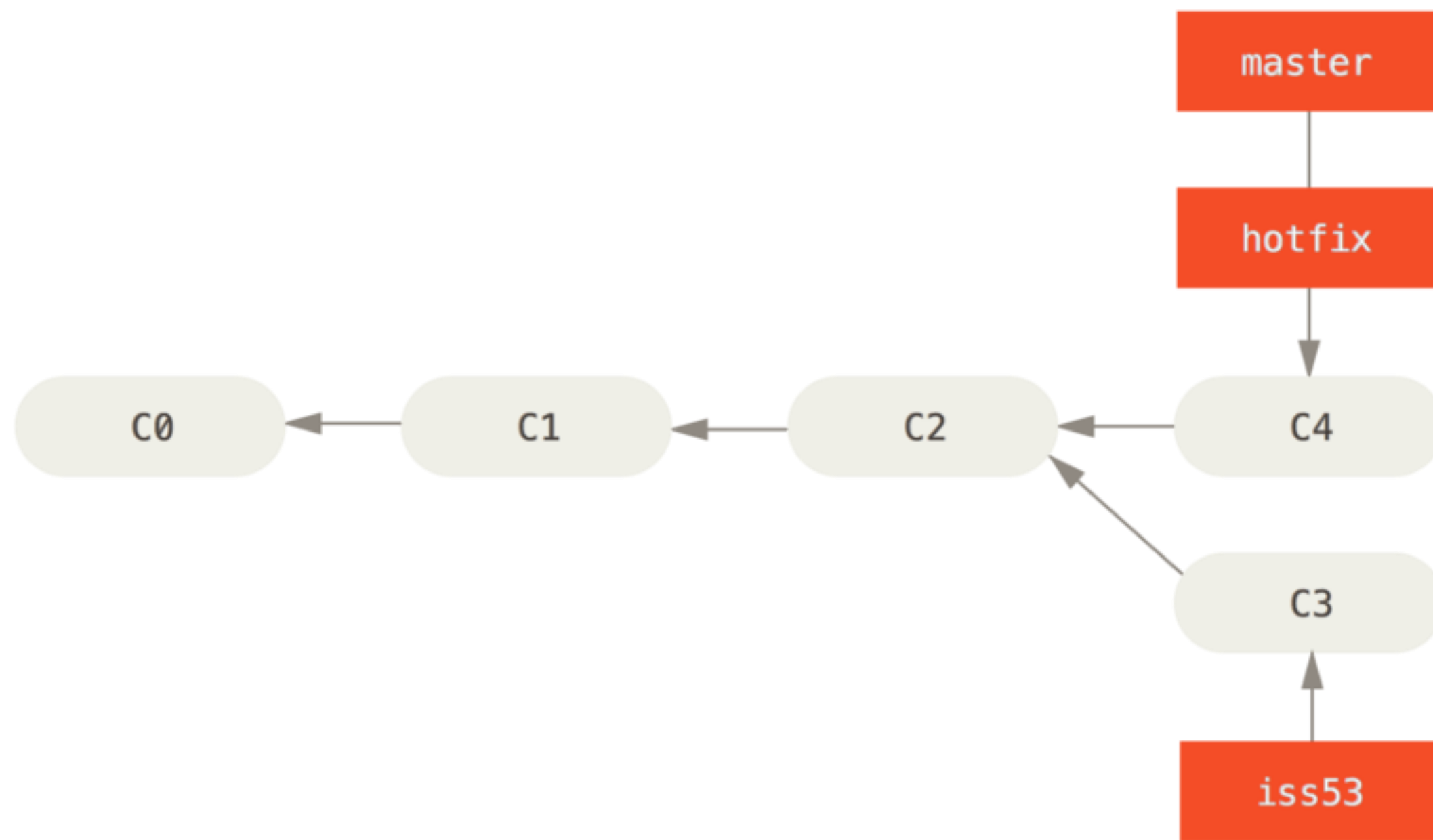
Branches no Git



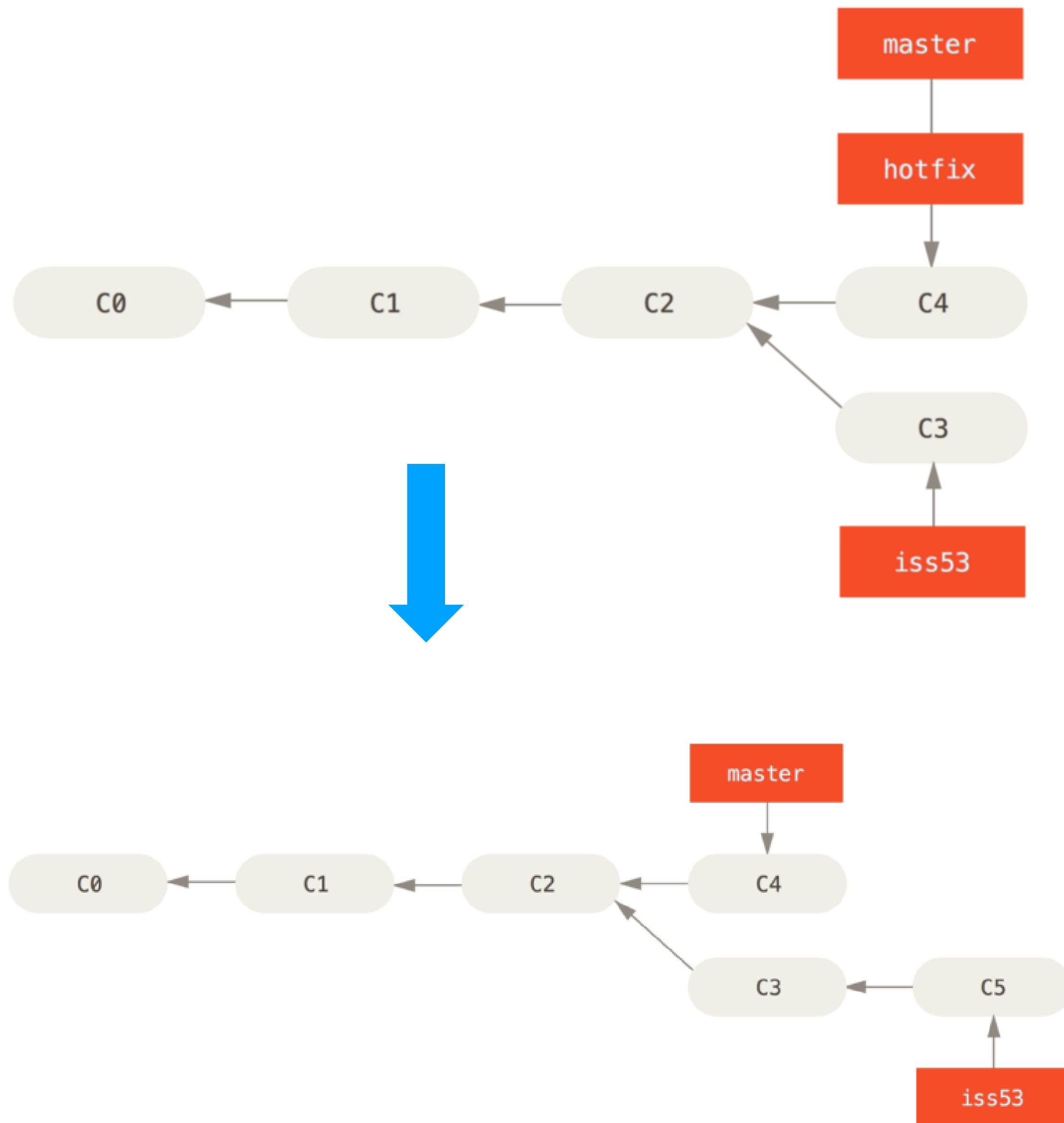
Branches no Git



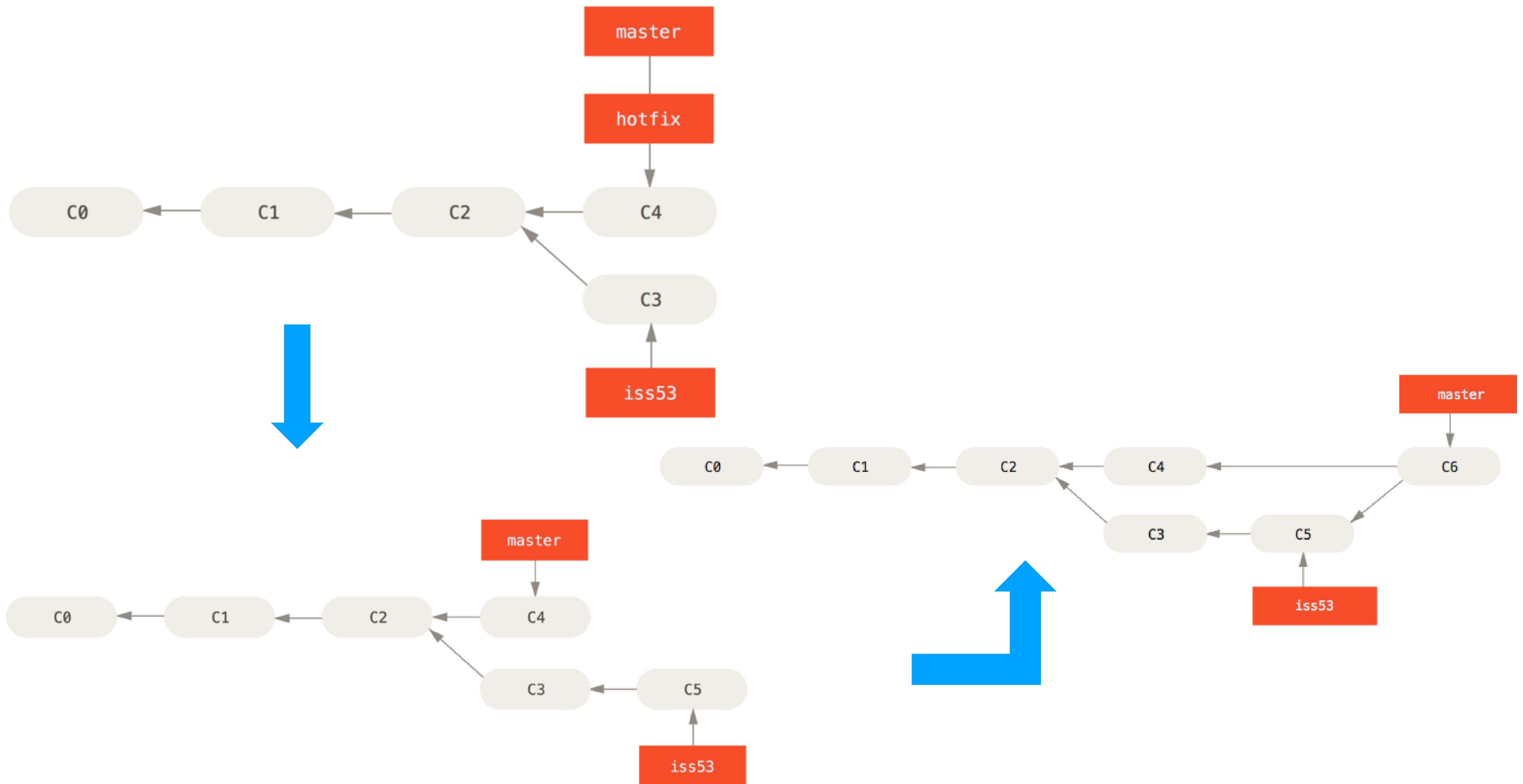
Branches no Git



Branches no Git



Branches no Git



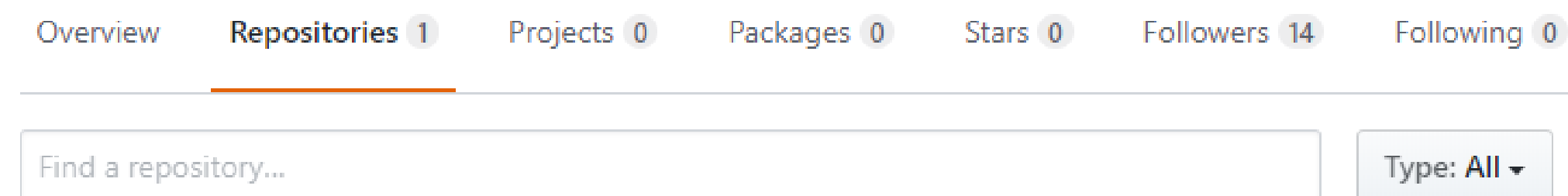
Iniciando o uso de Git com GitHub

- Antes de mais nada, precisamos instalar o sistema Git na nossa máquina.
- <https://git-scm.com/downloads>
- O GitHub (www.github.com) é uma plataforma de versionamento que utiliza Git como base.
- Não é o nosso objetivo ensinar a plataforma a fundo, mas vamos utilizar algumas funcionalidades. Caso tenha interesse em se aprofundar no assunto, recomendo algumas páginas:
 - <https://github.com/culturagovbr/primeiros-passos>;
 - <https://help.github.com/en>;
 - https://rogerdudler.github.io/git-guide/index.pt_BR.html.
- Antes de começar a configurar, crie uma conta no site.

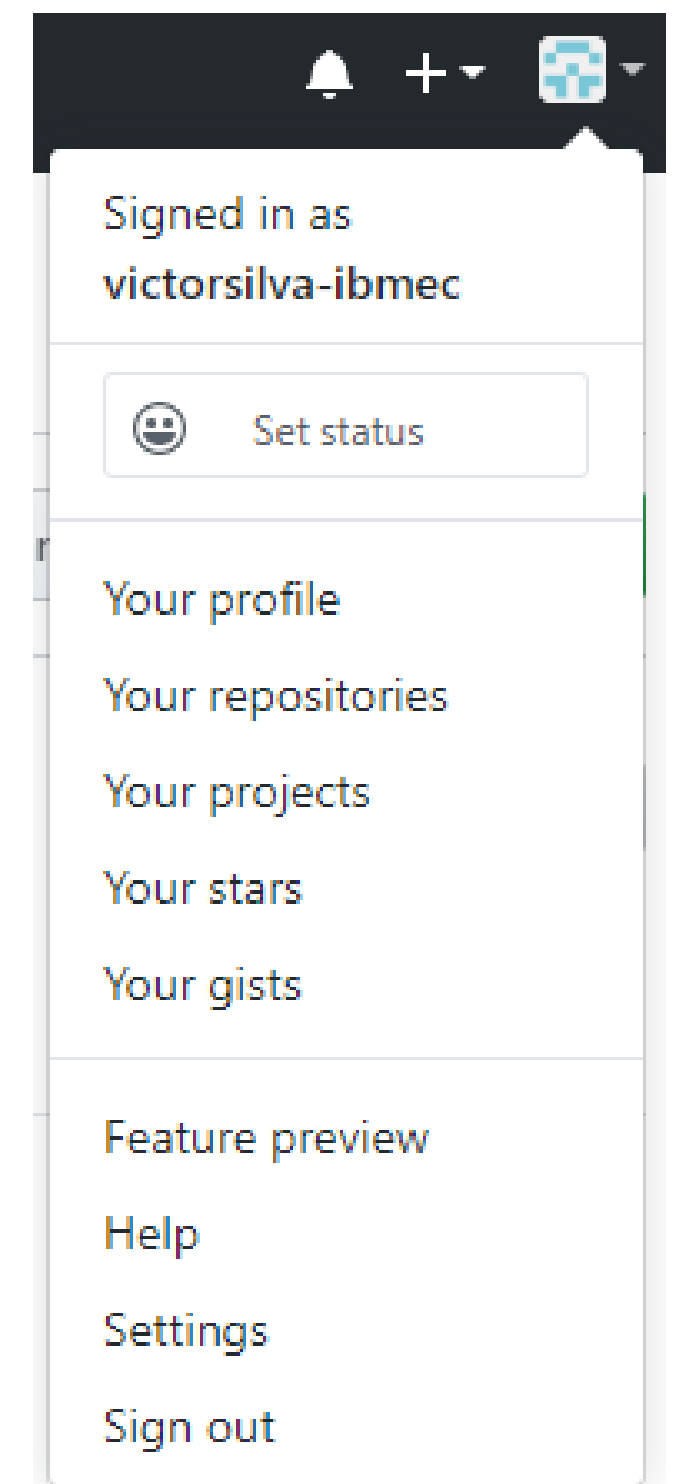
Algumas tarefas no GitHub

- **Criando um novo repositório:**

- No canto superior direito, clique no ícone do seu usuário e, em seguida, em **Your Profile**;
- Na nova janela, clique em **Repositories** e em **New**;

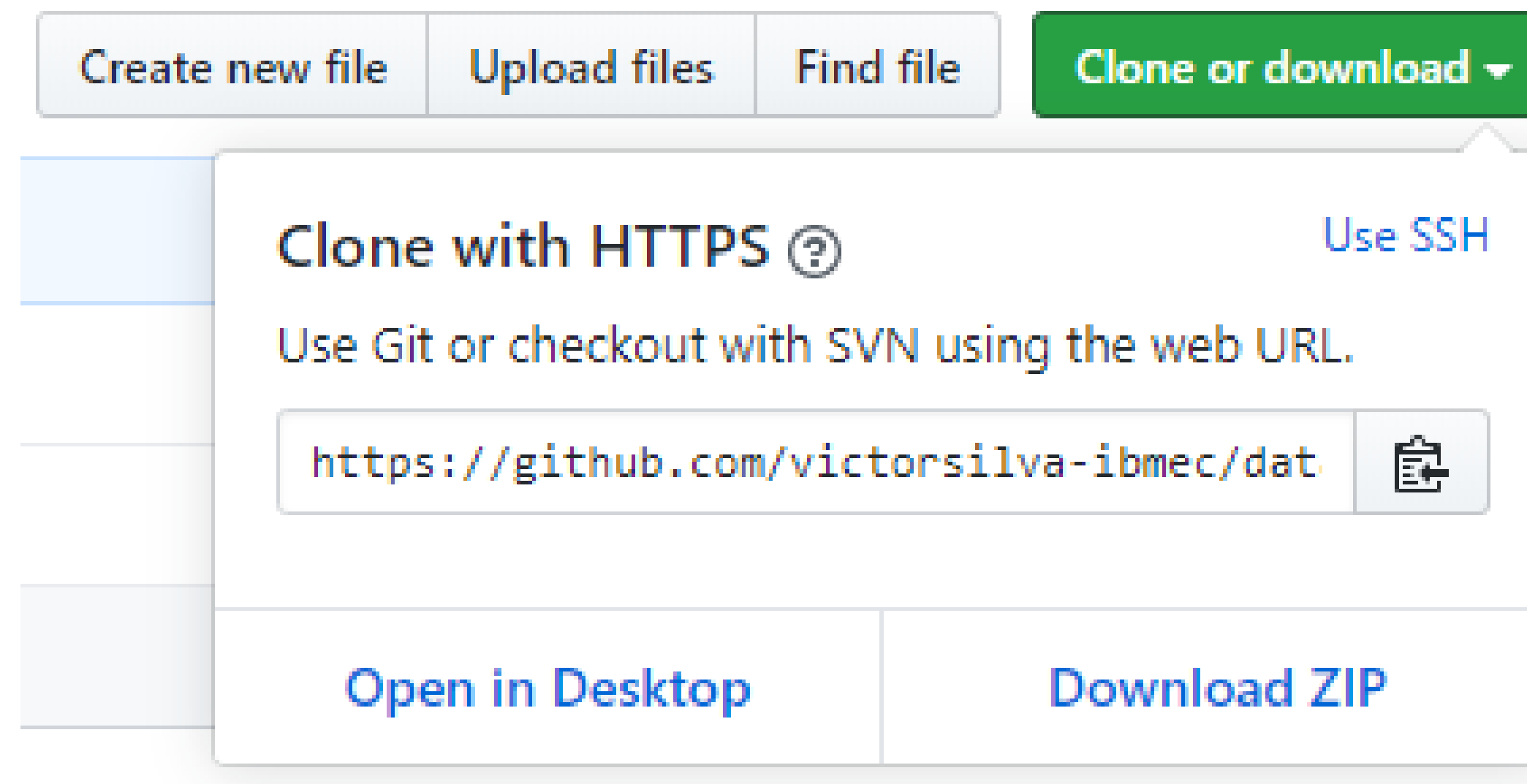


- Defina um nome (evite espaços e números), insira uma descrição se desejar e marque a opção **Public**, para que ele possa ser compartilhado. Por fim, marque a caixa **Initialize this repository with a README**;
- Clique em **Create repository**.

A screenshot of the GitHub 'Create new repository' form. The 'Owner' is 'victorsilva-ibmec' and the 'Repository name' is 'data-mining' with a green checkmark. A hint says 'Great repository names are short and memorable. Need inspiration? How about upgraded-octo-bassoon?'. The 'Description (optional)' is 'Repositório para curso de Data Mining com Python'. The 'Public' option is selected, with a note 'Anyone can see this repository. You choose who can commit.' The 'Private' option is also visible. A note says 'Skip this step if you're importing an existing repository.' The 'Initialize this repository with a README' checkbox is checked, with a note 'This will let you immediately clone the repository to your computer.' There are dropdowns for 'Add .gitignore: None' and 'Add a license: None'. A green 'Create repository' button is at the bottom.

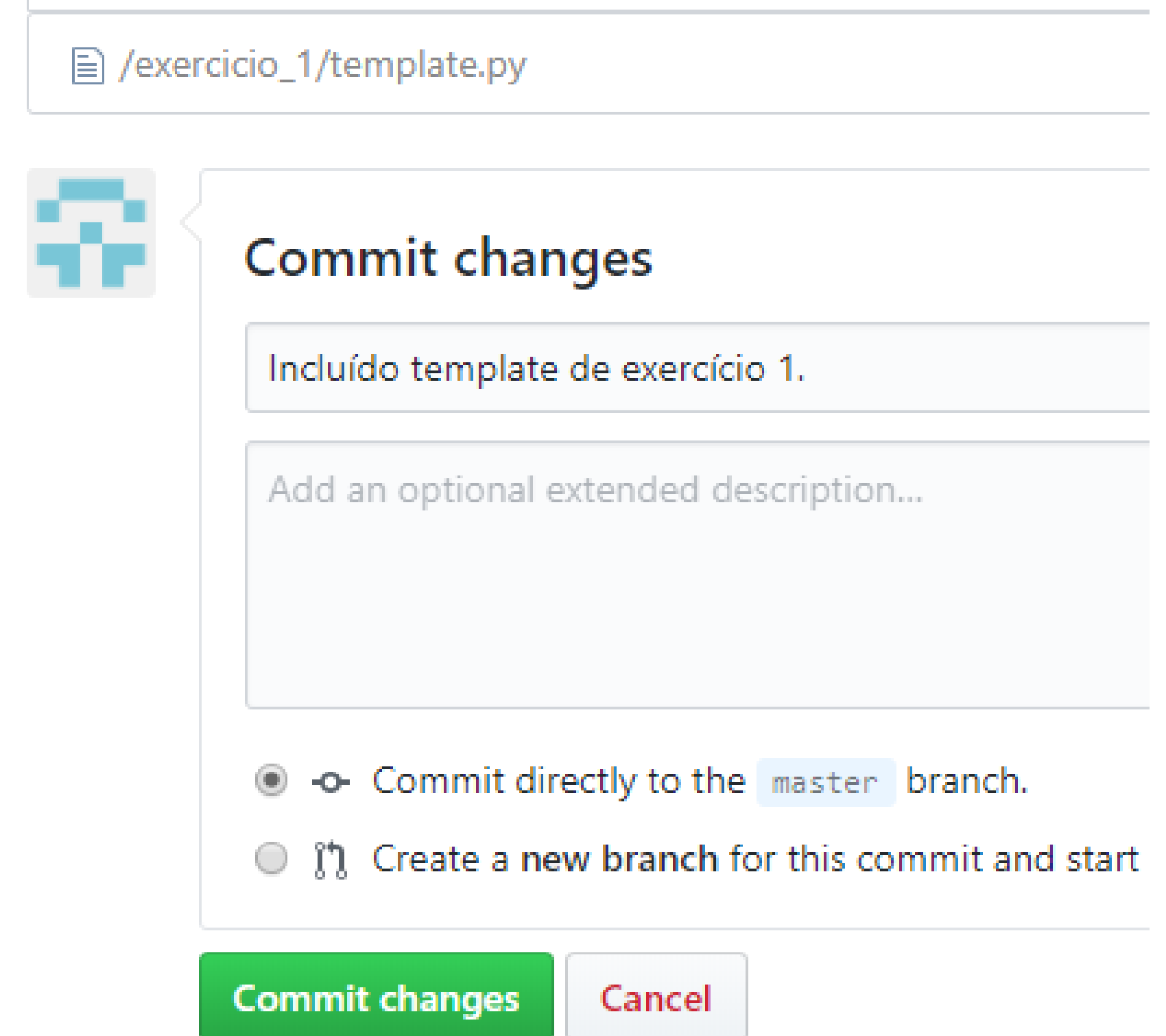
Algumas tarefas no GitHub

- **Baixando manualmente um repositório para a máquina:**
 - Na tela do seu repositório, clique em **Clone or download**;
 - Em seguida, clique em **Download ZIP**;
 - Com o arquivo .zip baixado, descompacte-o na sua pasta de projeto, substituindo arquivos antigos se necessário.



Algumas tarefas no GitHub

- **Fazendo um novo commit no seu repositório:**
 - Commits são alterações feitas no repositório. Podem conter um único arquivo ou vários. Caso um arquivo commitado já exista, o GitHub vai fazer um controle de versão, comparando as alterações entre a versão anterior e a que foi commitada;
 - Na tela do seu repositório, clique em **Upload files**;
 - Arraste para a tela os arquivos que deseja incluir;
 - Insira uma breve descrição do que está sendo commitado;
 - Deixe marcada a opção **Commit directly to the master branch**;
 - Clique em **Commit changes**.



The screenshot shows the GitHub 'Commit changes' interface. At the top, a file path `/exercicio_1/template.py` is displayed. Below it is a blue icon representing a commit. The main section is titled 'Commit changes' and contains a text box with the message 'Incluído template de exercício 1.' Below this is a larger text box with the placeholder 'Add an optional extended description...'. At the bottom, there are two radio button options: 'Commit directly to the master branch.' (which is selected) and 'Create a new branch for this commit and start'. At the very bottom are two buttons: 'Commit changes' (green) and 'Cancel' (grey).

Algumas tarefas no GitHub

- **Submetendo um trabalho para revisão:**
 - Um **pull request** é o ato de submeter para aprovação as alterações ou inserções de código de um ou mais arquivos. A pessoa que abre um **pull request** sinaliza que gostaria de uma aprovação do conteúdo antes de ele ser, de fato, incorporado ao repositório;
 - No repositório desejado, clique na pasta em que você deseja incluir ou atualizar os arquivos;
 - Clique em **Upload files**;
 - Arraste para a tela o(s) arquivo(s) com a sua atualização, e na descrição explique o que está sendo feito. Em seguida, clique em **Commit changes**;
 - Na nova janela, insira um comentário e depois clique em **Create pull request**.

Alguns comandos do Git

- Apesar do Github fornecer recursos para operar com Git direto pelo navegador, esses recursos são limitados. Por exemplo, fazer checkout de um novo branch apenas pelo navegador pode ser bem complicado.
- Uma forma mais usual de se usar o Git é através de programas específicos para o computador.
- É bem comum utilizar os comandos do Git por linha de comando, porém existem bons programas para uso do Git através de uma interface gráfica, como o [Sourcetree](#).
- No slide a seguir serão apresentados alguns comandos comuns para o uso do Git pela linha de comando. Eles podem ser executados pelo Git Bash (programa instalado junto com o Git), ou pelo terminal.

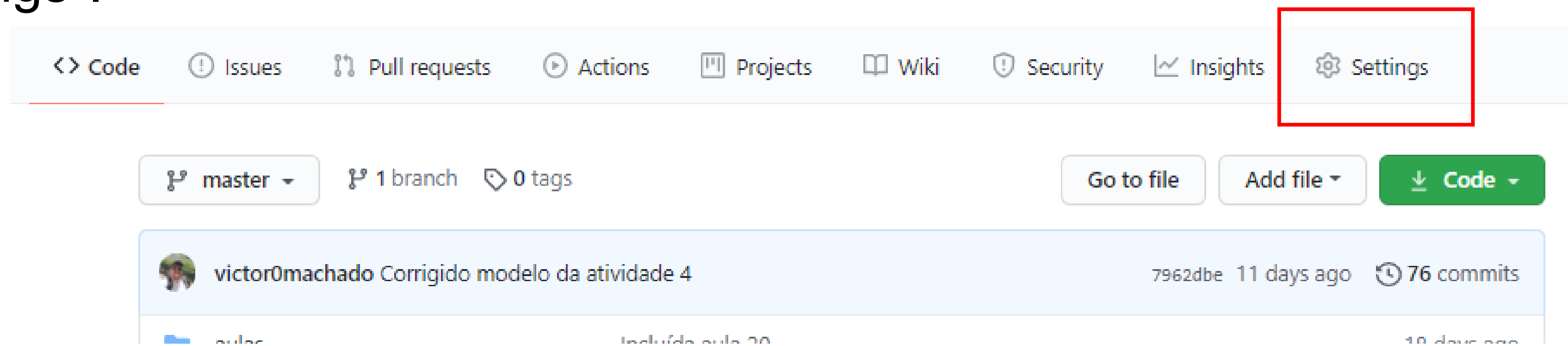
Alguns comandos do Git

- Uma observação, para facilitar as operações, é trabalhar sempre na raiz do repositório.
- Com o terminal na raiz do repositório, insira os seguintes comandos para obter os efeitos apresentados:

Comando	Efeito
git init	Inicia um repositório Git no diretório em questão
git status	Indica os status de arquivos modificados, adicionados ou removidos, além de arquivos preparados (staged)
git add <arquivo>	Prepara o arquivo mencionado
git add -u	Prepara todos os arquivos modificados (porém não faz nada com arquivos novos)
git add .	Prepara todos os arquivos (incluindo arquivos novos)
git commit -m "Mensagem"	Faz um commit dos arquivos preparados, incluindo a mensagem de commit definida
git restore <arquivo>	Desfaz modificações do arquivo que não foi preparado
git restore --staged <arquivo>	Desfaz a preparação do arquivo (porém mantém modificações)
git pull	No branch escolhido, atualiza as informações com o repositório remoto
git branch	Lista todos os branches armazenados localmente
git branch --show-current	Lista o branch atual
git branch -m <novo_nome>	Renomeia o branch atual (cuidado ao fazer isso para branches que já estão no repositório remoto!)
git checkout <branch>	Dá checkout no branch mencionado
git checkout -b <branch>	Cria um novo branch, com o nome mencionado, e dá checkout nele
git push	Envia os commits realizados localmente para o branch remoto

GitHub pages

- Uma boa forma de manter o seu portfólio sempre atualizado é com uma página pessoal, na qual você inclui seu currículo, projetos realizados, trabalhos, interesses pessoais e profissionais, e outras informações que achar pertinente.
- O Github possui uma forma muito simples de se criar um repositório que também serve como página pessoal.
- Para isso, crie um repositório normal, vá na página desse repositório e clique em “Settings”.



GitHub pages

- Nas configurações, desça a página até encontrar a seção “GitHub Pages”.

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Source
GitHub Pages is currently disabled. Select a source below to enable GitHub Pages for this repository. [Learn more.](#)

None ▾ Save

Theme Chooser
Select a theme to publish your site with a Jekyll theme using the gh-pages branch. [Learn more.](#)

Choose a theme

- No campo “Source”, indique o branch que você quer que seja a sua página principal (usualmente é o branch master). Se quiser, escolha um tema da lista de temas gratuitos disponíveis e, em seguida, clique em “Save”.

GitHub pages

- A página terá uma atualização que mostrará a URL do site, além de incluir um campo no qual você pode inserir um domínio customizado, caso o tenha.

GitHub Pages

GitHub Pages is designed to host your personal, organization, or project pages from a GitHub repository.

Your site is ready to be published at <https://victor0machado.github.io/2020.2-logprog/>.

Source

Your GitHub Pages site is currently being built from the master branch. [Learn more.](#)

🔑 Branch: master ▾

📁 / (root) ▾

Save

Theme Chooser

Select a theme to publish your site with a Jekyll theme. [Learn more.](#)

Choose a theme

Custom domain

Custom domains allow you to serve your site from a domain other than victor0machado.github.io. [Learn more.](#)

Save

GitHub pages

- O site funciona como um repositório normal. Todas as páginas devem ser escritas em Markdown, que já vimos ao longo do curso.

```
# Boas-vindas

Vou atualizar essa página com os materiais das disciplinas que leciono no
IBMEC/RJ.

## Disciplinas

* [Algoritmos e Programação de Computadores](/courses/algprog.md)
* [Lógica e Programação de Computadores](/courses/logprog.md)
* [Data Mining com Python](/courses/datamining.md)

## Meus contatos

* E-mail: <victor.silva@professores.ibmec.edu.br>
* [Linkedin](https://www.linkedin.com/in/victormachadodasilva/)
* [Lattes](http://lattes.cnpq.br/1584907276781609)
```

Repositório público do Prof. Victor Machado

Material usado nas minhas disciplinas do
IBMEC/RJ

[View My GitHub Profile](#)

Boas-vindas

Vou atualizar essa página com os materiais das disciplinas que leciono no
IBMEC/RJ.

Disciplinas

- [Algoritmos e Programação de Computadores](#)
- [Lógica e Programação de Computadores](#)
- [Data Mining com Python](#)

Meus contatos

- E-mail: victor.silva@professores.ibmec.edu.br
- [Linkedin](#)
- [Lattes](#)

- O único arquivo exigido para o site é o **index.md**, que deve ficar na raiz do repositório. Novos arquivos e pastas podem ser criados se necessário, e a navegação é sempre relativa à raiz do repositório.

OBRIGADO!



www.ibmec.br

 /ibmec

 ibmec

 @ibmec_oficial

 ibmec

