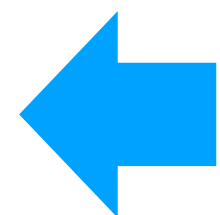


# Algoritmos e Programação de Computadores

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

# Índice

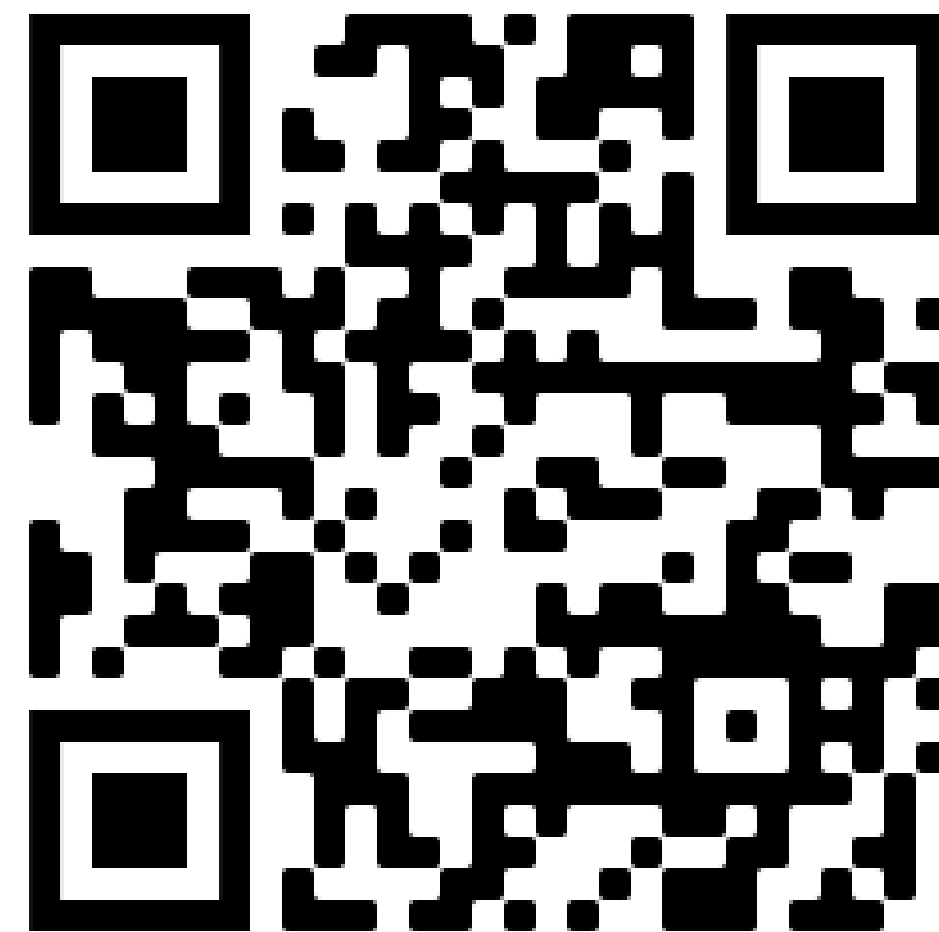
- [Apresentação do curso](#)
- [Por que aprender a programar?](#)
- [Algoritmos: definição e representação](#)
- [Configurando o ambiente Python](#)
- [Instalando e configurando IDEs](#)
- [Utilizando o PyCharm](#)
- [Utilizando o VSCode](#)
- [Configurando o pylint](#)
- [Instalando pacotes pelo PyPI](#)



# Apresentação do curso

# Apresentação do curso

- Contato: [victor.silva@professores.ibmec.edu.br](mailto:victor.silva@professores.ibmec.edu.br)
- Aulas às segundas e quartas-feiras, de 7:30 às 9:21
- Grupo no Whatsapp: <https://chat.whatsapp.com/IFDRlYeZewh72bcJOFGVj2>
- Material no GitHub: <https://github.com/victor0machado/2021.1-algprog>



# Apresentação do curso

Aula	Dia	Dia sem.	Tópico
01	22/02/2021	seg	Introdução à disciplina
02	24/02/2021	qua	Introdução ao Python
03	01/03/2021	seg	Variáveis
04	03/03/2021	qua	Estruturas sequenciais
05	08/03/2021	seg	Estruturas condicionais: parte 1
06	10/03/2021	qua	Estruturas condicionais: parte 2
07	15/03/2021	seg	Estruturas de repetição: parte 1
08	17/03/2021	qua	Estruturas de repetição: parte 2
09	22/03/2021	seg	Estruturas de repetição: parte 3
10	24/03/2021	qua	Estruturas de dados primitivas: parte 1
11	29/03/2021	seg	Estruturas de dados primitivas: parte 2
12	31/03/2021	qua	Estruturas de dados primitivas: parte 3
13	05/04/2021	seg	Dúvidas
14	07/04/2021	qua	SEM AULA (SEMANA AP1)
15	12/04/2021	seg	SEM AULA (SEMANA AP1)
16	14/04/2021	qua	SEM AULA (SEMANA AP1)
17	19/04/2021	seg	Estruturas de dados derivadas: parte 1
18	21/04/2021	qua	SEM AULA (TIRADENTES)
19	26/04/2021	seg	Estruturas de dados derivadas: parte 2
20	28/04/2021	qua	Estruturas de dados derivadas: parte 3

# Apresentação do curso

Aula	Dia	Dia sem.	Tópico
21	03/05/2021	seg	Manipulação de strings: parte 1
22	05/05/2021	qua	Manipulação de strings: parte 2
23	10/05/2021	seg	Uso e importação de módulos: parte 1
24	12/05/2021	qua	Uso e importação de módulos: parte 2
25	17/05/2021	seg	Manipulação de arquivos
26	19/05/2021	qua	Manipulação de elementos do sistema operacional
27	24/05/2021	seg	Desenvolvendo uma aplicação com python: parte 1
28	26/05/2021	qua	Desenvolvendo uma aplicação com python: parte 2
29	31/05/2021	seg	Aplicação de python com o módulo pygame: parte 1
30	02/06/2021	qua	Aplicação de python com o módulo pygame: parte 2
31	07/06/2021	seg	Aplicação de python com o módulo pygame: parte 3
32	09/06/2021	qua	Desenvolvendo uma aplicação com GUI: parte 1
33	14/06/2021	seg	Desenvolvendo uma aplicação com GUI: parte 2
34	16/06/2021	qua	Desenvolvendo uma aplicação com GUI: parte 3
35	21/06/2021	seg	Dúvidas
36	23/06/2021	qua	SEM AULA (SEMANA AP2)
37	28/06/2021	seg	SEM AULA (SEMANA AP2)
38	30/06/2021	qua	SEM AULA (SEMANA AP2)
39	05/07/2021	seg	SEM AULA (SEMANA AS)
40	07/07/2021	qua	SEM AULA (SEMANA AS)

# Apresentação do curso

## Avaliação

- Proporção:
  - Exercícios periódicos (AC): 20%
  - Projeto (AP1): 40%
  - Projeto (AP2): 40%
- Detalhes das entregas:
  - Exercícios da AC devem ser individuais
  - Projetos de AP1 e AP2 em grupos de no mínimo 3 e no máximo 4 pessoas
- AS será uma prova com consulta, que substituirá a menor nota entre AP1 e AP2.



# 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>
  - 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!)
  - Luciano Ramanho - *Python Fluente: Programação Clara, Concisa e Eficaz* (Novatec)
  - Paul Barry - *Use a Cabeça! Python - 2ª Edição* (Alta Books)
  - Stack Overflow: <https://stackoverflow.com/questions/tagged/python>
  - Artigos no Medium.com: <https://medium.com/search?q=python>

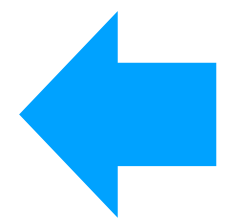


# Sugestões de materiais para estudo

- 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>

# Conhecendo melhor a turma...





# Por que aprender a programar?

# O que significa “programar”?

- Programar um computador significa fornecer à máquina um conjunto de instruções que indica o que você quer que ele faça.
  - Quando você pressiona o botão “enviar” no seu aplicativo de e-mail, está comandando a máquina para que ela produza um determinado resultado. Ela reage a esse comando executando um conjunto de instruções previamente programadas pelos criadores do aplicativo, as quais realizam a tarefa de enviar um e-mail.
- A programação passa então por acessar essas características internas da máquina e ser capaz de usá-las
- Programar significa escrever instruções que sejam executadas diretamente pela máquina, sem utilizar aplicações intermediárias preconcebidas como planilhas ou browsers de internet.

# O que significa “programar”?

- A programação se resume a uns poucos conceitos fundamentais:
  - *Entrada de dados:* define de que forma a máquina obtém informações, que podem ser recebidas via teclado, arquivos, rede, mouse ou outros dispositivos. Essas informações tanto podem estar sendo produzidas por usuários humanos quanto por outros dispositivos.
  - *Saída de dados:* define como a máquina apresenta respostas ao mundo externo, que podem ser exibidas através de um monitor, impressora ou outros dispositivos. Da mesma forma que as entradas, as saídas podem ser enviadas usuários humanos ou a outros dispositivos.

# O que significa “programar”?

- A programação se resume a uns poucos conceitos fundamentais:
  - *Processamento*: define como os dados de entrada são transformados e combinados com outras informações eventualmente já armazenadas pela máquina, ou obtidas por ela, para produzir as saídas. O processamento segue uma lógica de execução que é conhecida como “algoritmo”. Um algoritmo é basicamente uma lista finita de instruções que resolve um problema específico e se estrutura a partir de:
    - *Comandos*: instruções individuais que o programador dá à máquina e que ela é capaz de executar;
    - *Seleção*: estruturas que permitem à máquina decidir quando executar uma determinada lista de comandos ou outra;
    - *Repetição*: estruturas que permitem à máquina repetir uma lista de comandos um número fixo de vezes ou até que uma determinada condição seja obtida.



# Linguagens de programação?

- A forma como o programador transmite as instruções definidas no algoritmo para a máquina é dada através de um código, escrito utilizando uma linguagem de programação
- Existem basicamente dois tipos de linguagem de programação:
  - As de *baixo nível* são aquelas cujos comandos podem ser executados diretamente pelo hardware do computador, ou seja, pelo microprocessador da máquina. Esse tipo de linguagem usualmente é utilizado por programadores altamente especializados em situações muito particulares ligadas ao controle das funções mais básicas da máquina;
  - A maioria dos programas hoje em dia são escritos em linguagens de *alto nível*, ou seja, linguagens que possuem comandos muito mais fáceis de serem entendidos por um ser humano. Essas linguagens usualmente produzem programas muito mais fáceis de ler.
- [Evolução das linguagens de programação](#)

# Linguagens de programação?

- O problema das linguagens de alto nível é que o hardware não consegue executar diretamente esses comandos. Assim, um programa especial deve ser usado para traduzir os comandos de alto nível em comandos de baixo nível executáveis pela máquina.
- Existem basicamente duas formas principais de fazer essa tradução: através de programas compiladores e programas interpretadores:
  - Um *compilador* é um programa que traduzirá a totalidade dos comandos escritos em linguagem de alto nível gerando um programa completo traduzido para a linguagem de baixo nível. Esse novo programa traduzido pode então ser diretamente executado pela máquina. Exemplos: C, C++, Pascal, Fortran.
  - Os *interpretadores* normalmente tomam um comando de vez na linguagem de alto nível e interpretam o seu significado, executando diretamente os comandos de alto nível em uma máquina virtual, que simula um computador de alto nível. Exemplos: Python, PHP, Ruby, Lua.

# Linguagens de programação?

- Existe uma diferença fundamental entre a linguagem humana (ou natural) e a linguagem de computadores. Os seres humanos são capazes de compreender expressões mesmo que elas não estejam escritas corretamente.

**Olá! Td bem c vc?**

- Um tópico fundamental em programação é:

*Programas são escritos em uma linguagem formal. Cada letra e cada sinal matemático ou de pontuação têm um significado muito preciso. Mudar qualquer um destes sinais pode, muitas vezes, mudar o sentido do programa, ou até fazê-lo parar de funcionar.*

```
1  for cont in range(10):  
2  |  print(str(cont))  
3  print(str(cont))
```

```
1  for cont in range(10):  
2  |  print(str(cont))  
3  |  print(str(cont))
```

# O que é um programa?

- Um *programa* consiste em um texto escrito em uma determinada linguagem de programação, seja de alto ou baixo nível, que por sua vez é processada de forma a comunicar ao computador comandos que devem ser executados.
- Um programa pode conter um ou mais algoritmos.
- O texto escrito do programa (ou código) pode ser armazenado no computador com extensões que variam conforme a linguagem de programação que estiver sendo utilizada.
- Em particular, a linguagem Python aceita diversas extensões, sendo a mais comum (e utilizada nesta disciplina) a extensão **.py**.



# 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



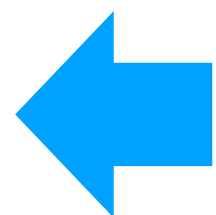
# Por que Python?

- 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?

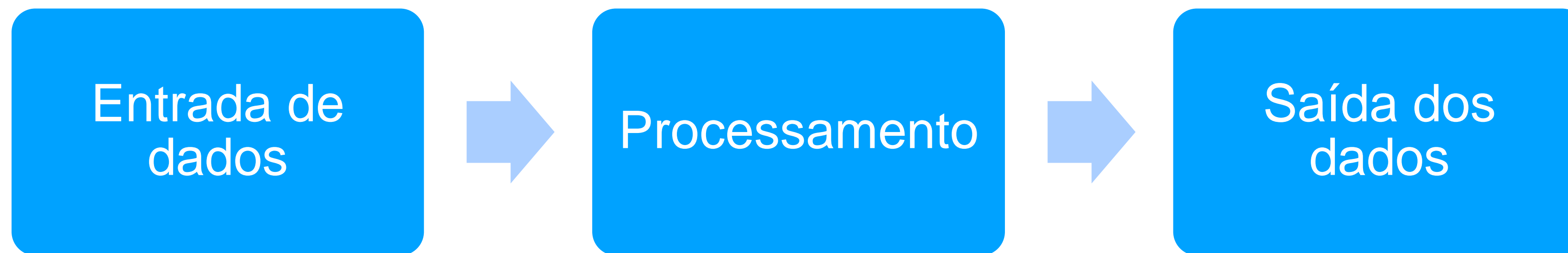
- Principais áreas de atuação:
  - Inteligência Artificial
  - Biotecnologia
  - Computação 3D
  - Ciência de Dados
  - Internet das Coisas



# Algoritmos: definição e representação

# Dados e instruções

- O computador executa **instruções** que operam sobre **dados**
- Essas instruções executadas pelo computador efetuam o **processamento** dos dados, retornando informações para o usuário, outros usuários ou outras máquinas, que por sua vez também processarão dados



# Dados e instruções

- Os tipos de dados de entrada podem ser:
  - Números (inteiros, reais, etc.), como p.ex.: 1, 2.5, 3.78, ...
    - **Importante! Sempre devemos utilizar como separador decimal o ponto (.), ao invés da vírgula (,)!**
  - Caracteres e cadeias de caracteres, como p.ex.: 'a', 'c', 'x1b2', '123', ...
    - As cadeias de caracteres (ou **strings**) são indicadas por aspas (em Python, são aceitas aspas simples ou duplas).
- Lógico ou booleano.
  - Transmitimos para o computador apenas a informação de um dado verdadeiro ou falso. Normalmente esse é o menor tipo de dado que podemos passar para um computador, já que o “falso” é representado pelo número 0, e o “verdadeiro” pelo número 1.

# Programa

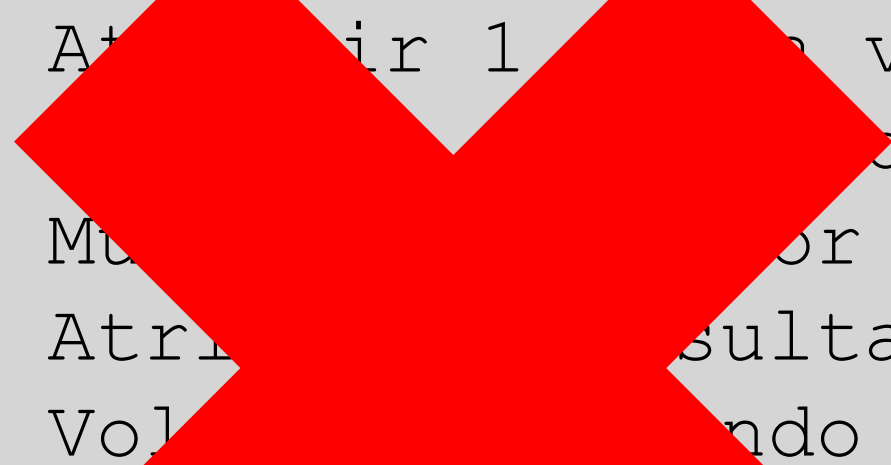
- Normalmente refere-se ao conjunto de instruções escrito em uma **linguagem de programação** que possam ser **entendidas** pela **máquina** que o executa.
- Toda linguagem pressupõe o uso de **regras de sintaxe** e de **semântica**.
  - Regras de sintaxe: especificam as cadeias de texto normalmente reconhecidas por uma linguagem. É a forma como as instruções da linguagem são escritas, sem levar em consideração o seu significado.
    - Exemplo: toda função em Python deve ser declarada utilizando-se a palavra reservada **def**
  - Regras de semântica: complementar à sintaxe, ela corresponde ao significado das instruções válidas de uma linguagem, e de como isso se traduz para a máquina para a máquina.
    - Exemplo: sempre que uma instrução **def** for declarada, deve entender que toda vez que uma determinada função for chamada no código, deve-se executar o bloco definido pela instrução **def**

# Algoritmos

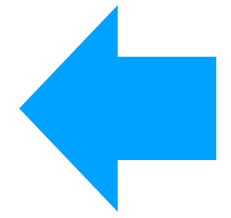
- Conjunto de instruções ou comandos, operando sobre dados que, obedecidos, resultam uma sucessão **finita** de ações.
- Exemplos:

```
Comando 1: Atribuir 2 a uma variável X  
Comando 2: Atribuir 3 a uma variável Y  
Comando 3: Somar X e Y  
Comando 4: Escrever o resultado
```

```
Comando 1: Atribuir 1 a uma variável X  
Comando 2: Multiplicar o valor de X  
Comando 3: Multiplicar o valor 2  
Comando 4: Atribuir o resultado a X  
Comando 5: Voltar ao comando 2
```







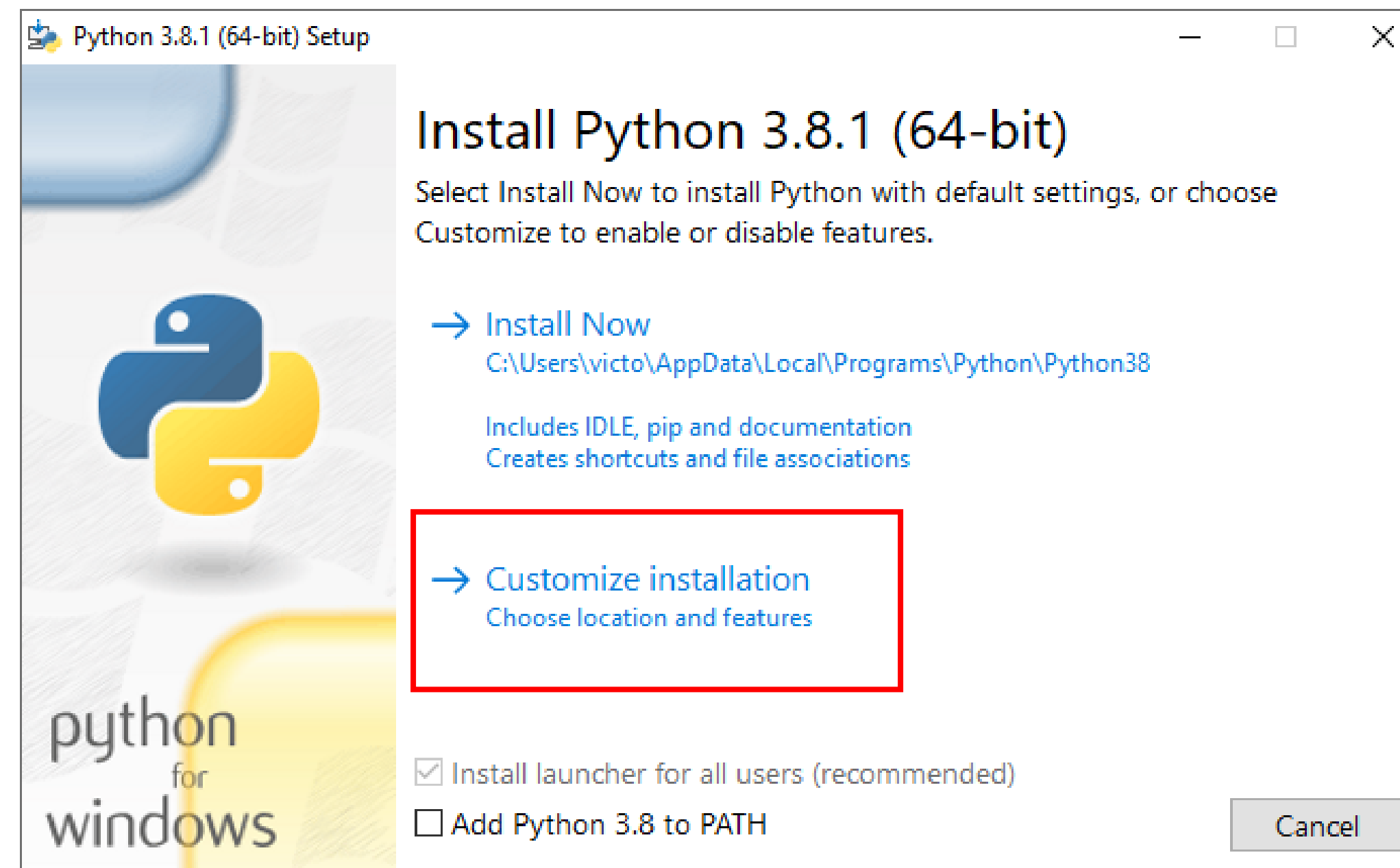
# Configurando o ambiente Python

# 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

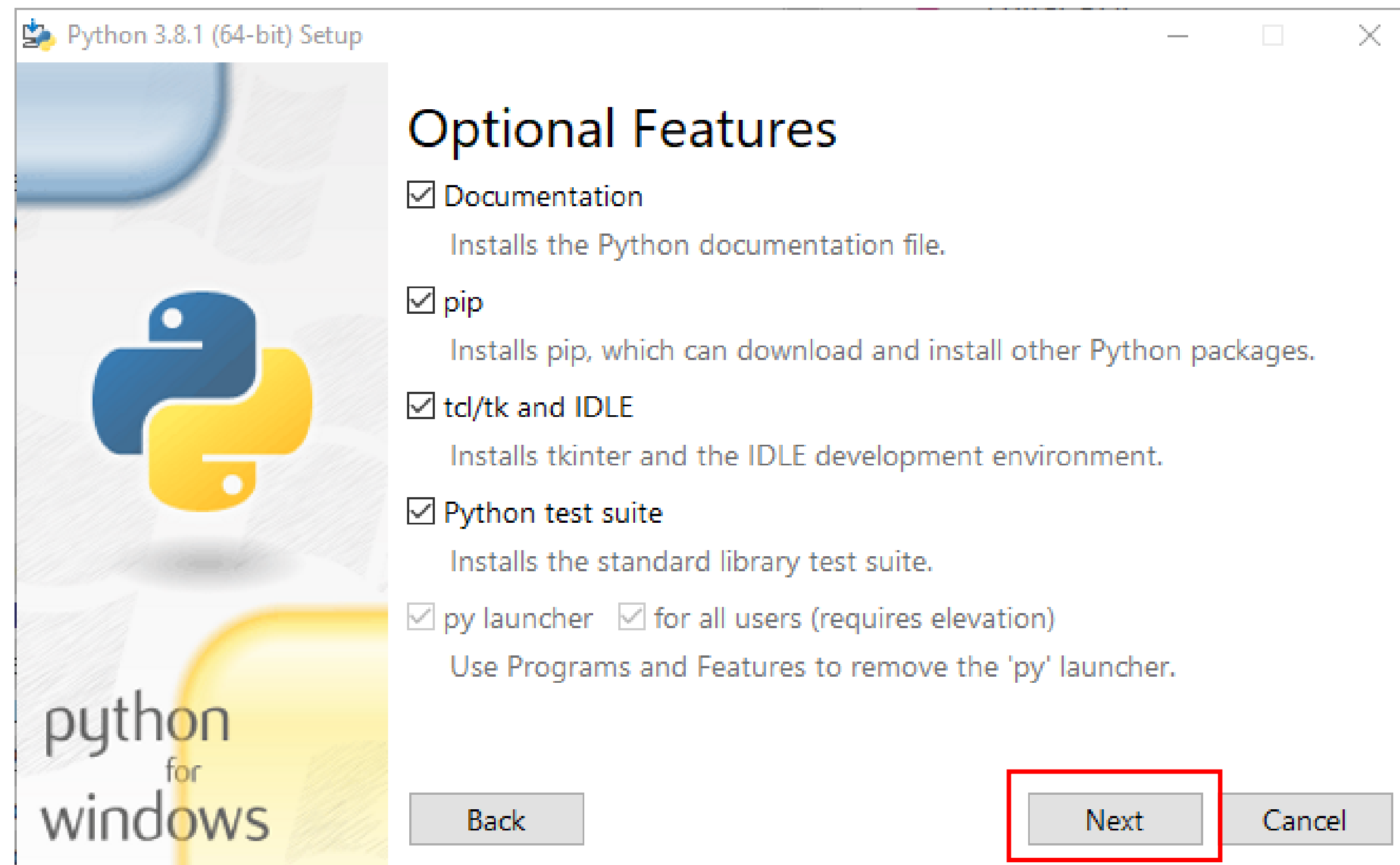
# Como instalar Python?

- 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”.



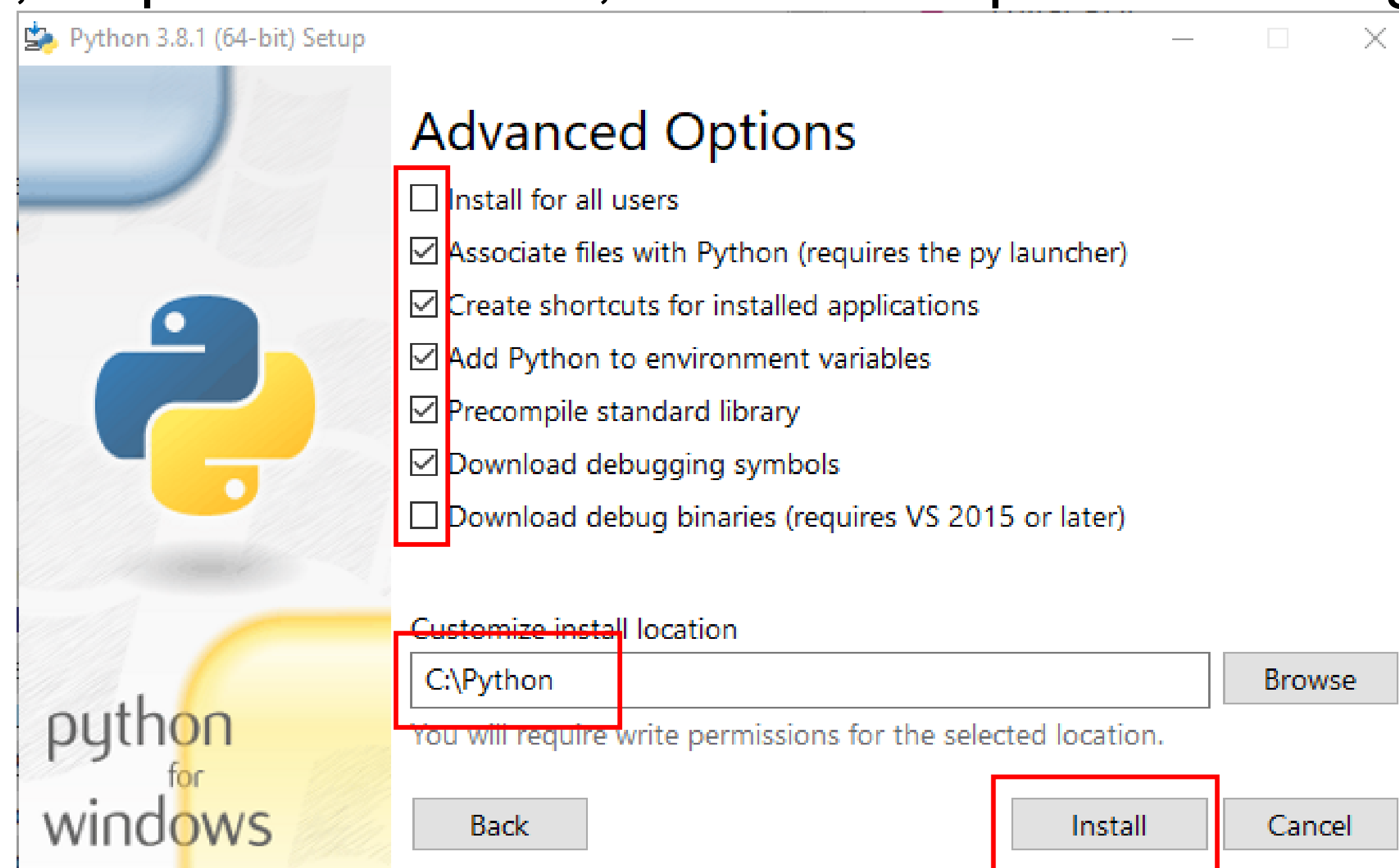
# Como instalar Python?

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




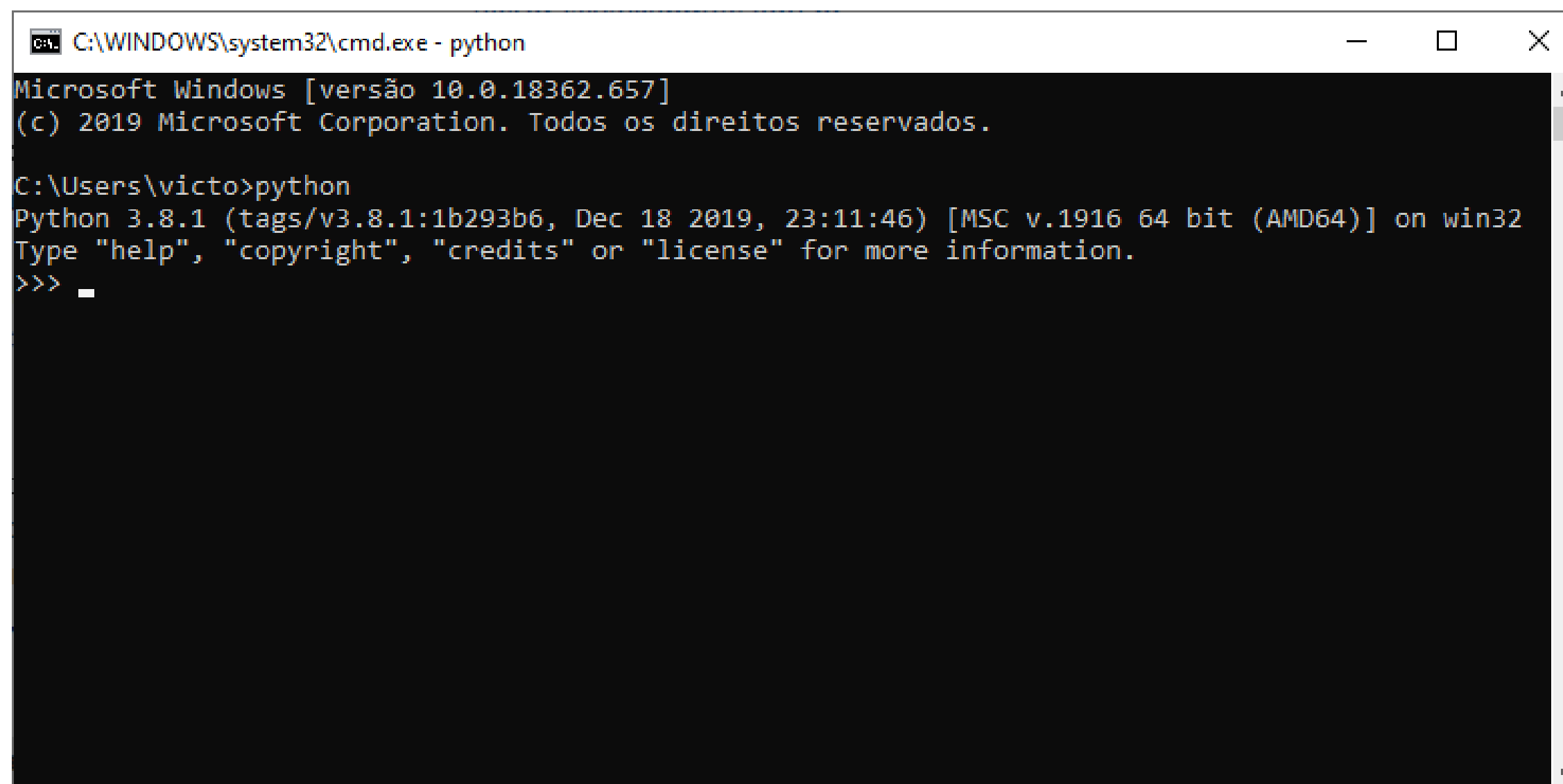
# Como instalar Python?

- 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.



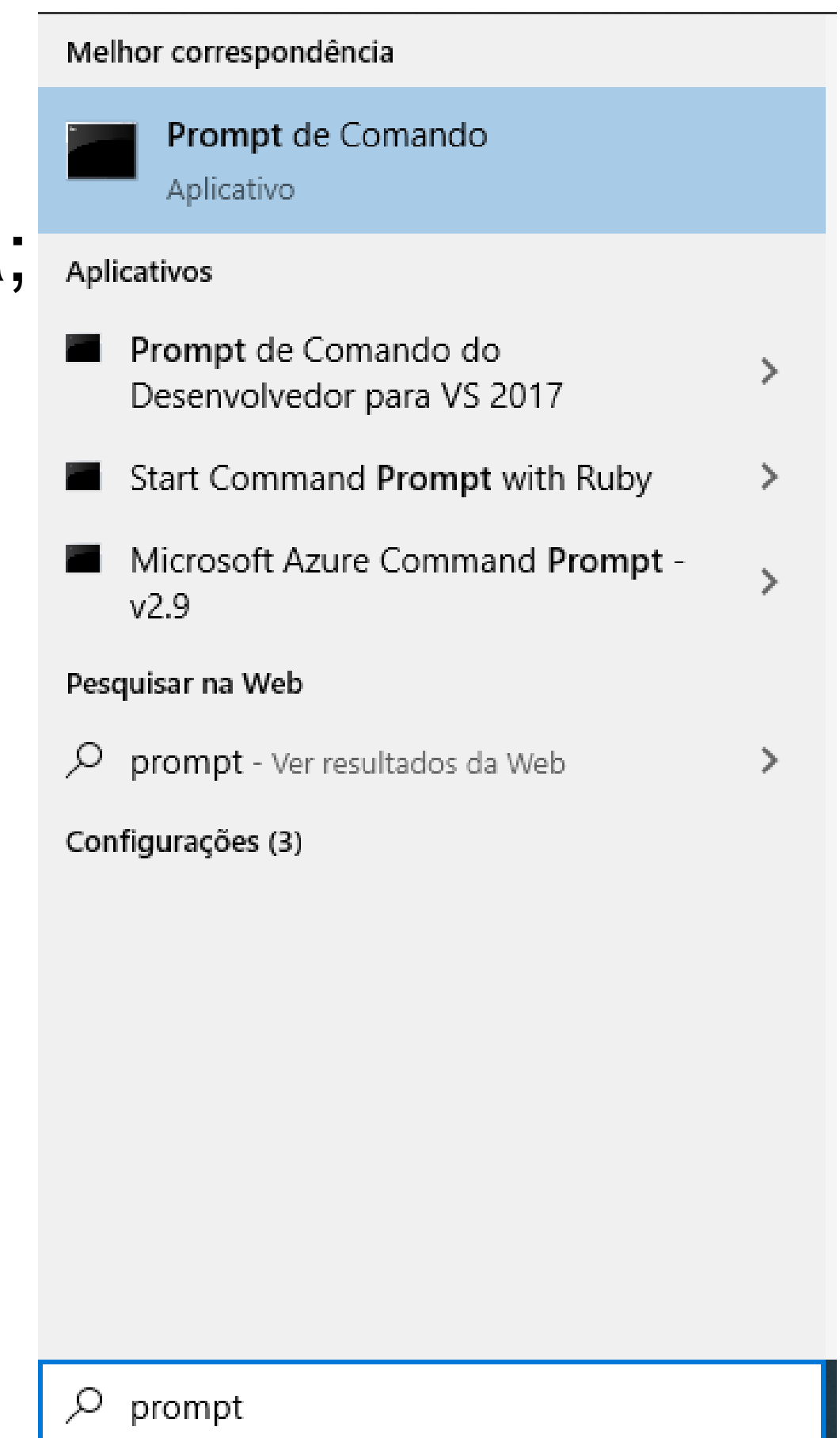
# Como instalar Python?

- 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.

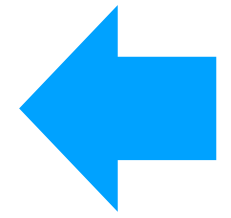
C:\Users\victo>python
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> _
```





# Algumas dicas iniciais após instalar o Python

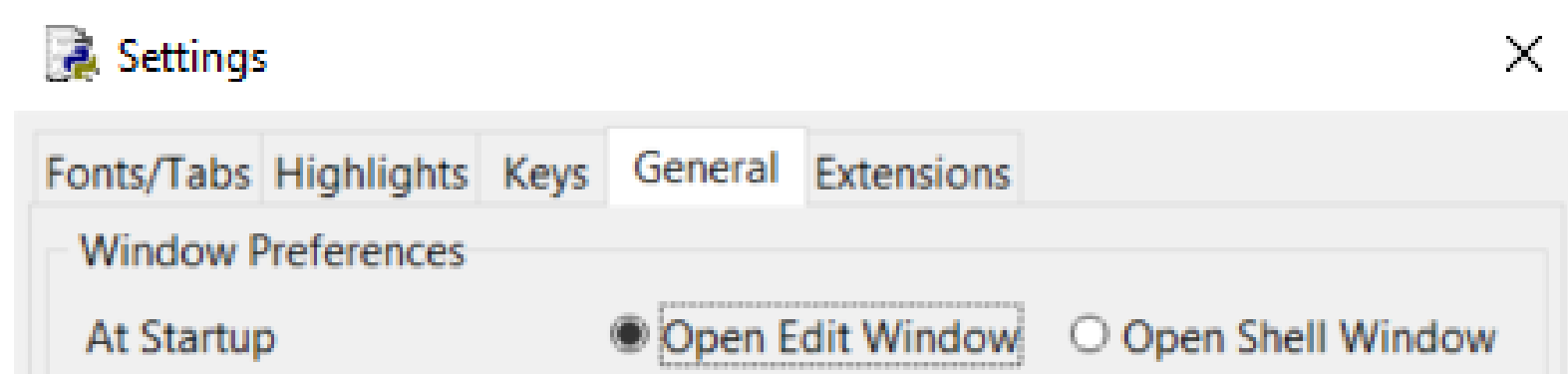
1. Antes de abrir o programa, abra o Windows Explorer, clique em **Este Computador** e, em seguida, no drive do seu computador (C: ou D:, dependendo da sua máquina);
2. Neste diretório, crie uma pasta chamada **Projetos**. Esta pasta será usada para armazenar todos os seus projetos de software;
3. Dentro 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



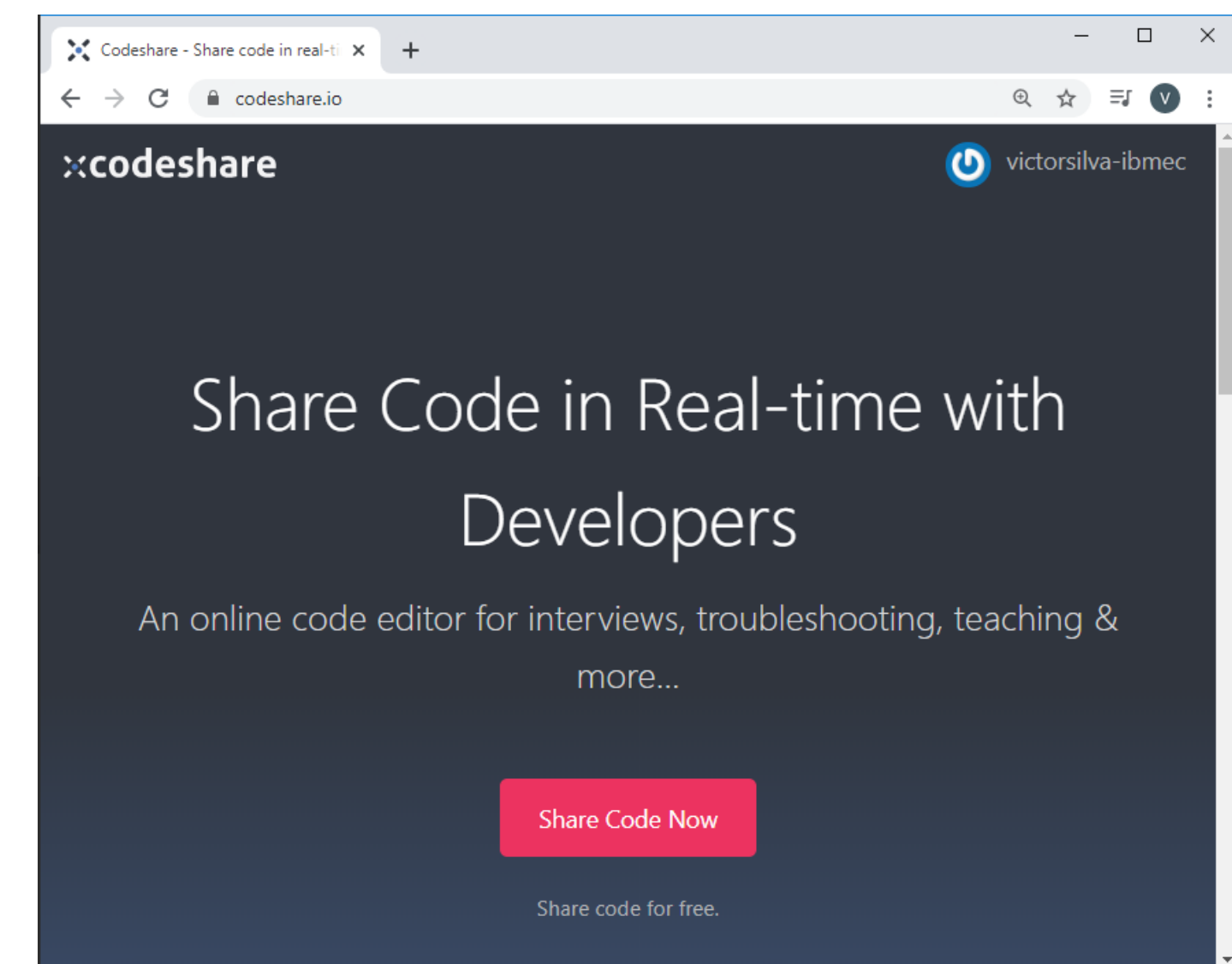
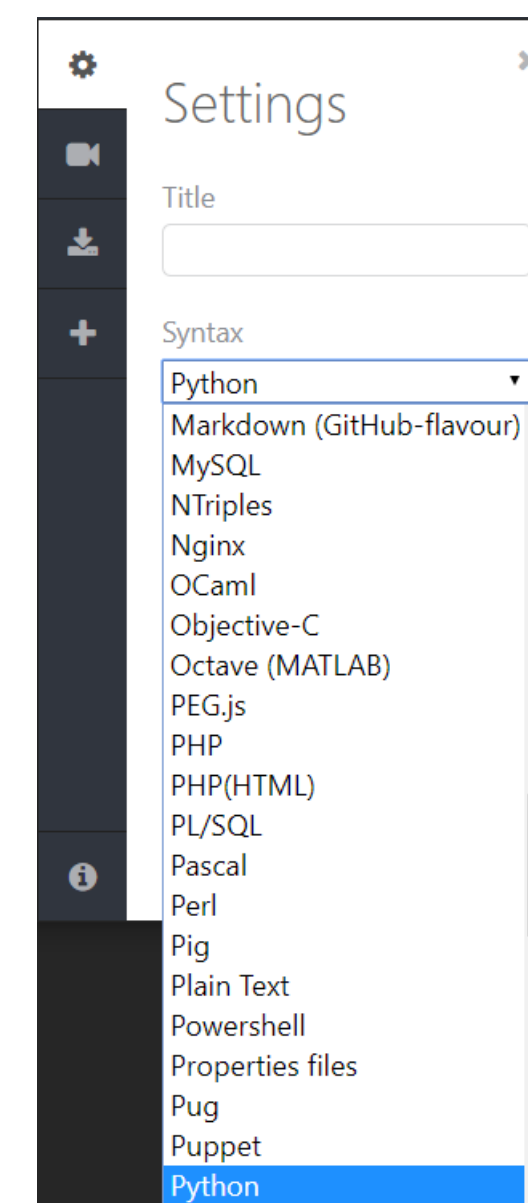
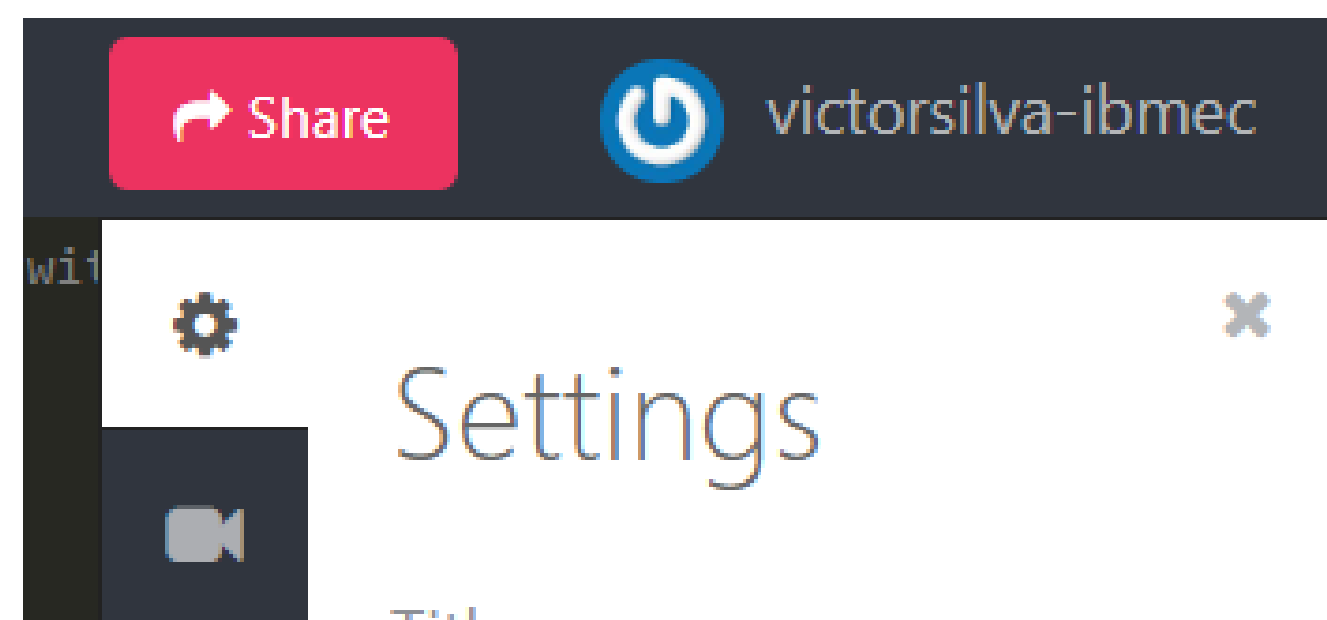
# Configurando o IDLE

- 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

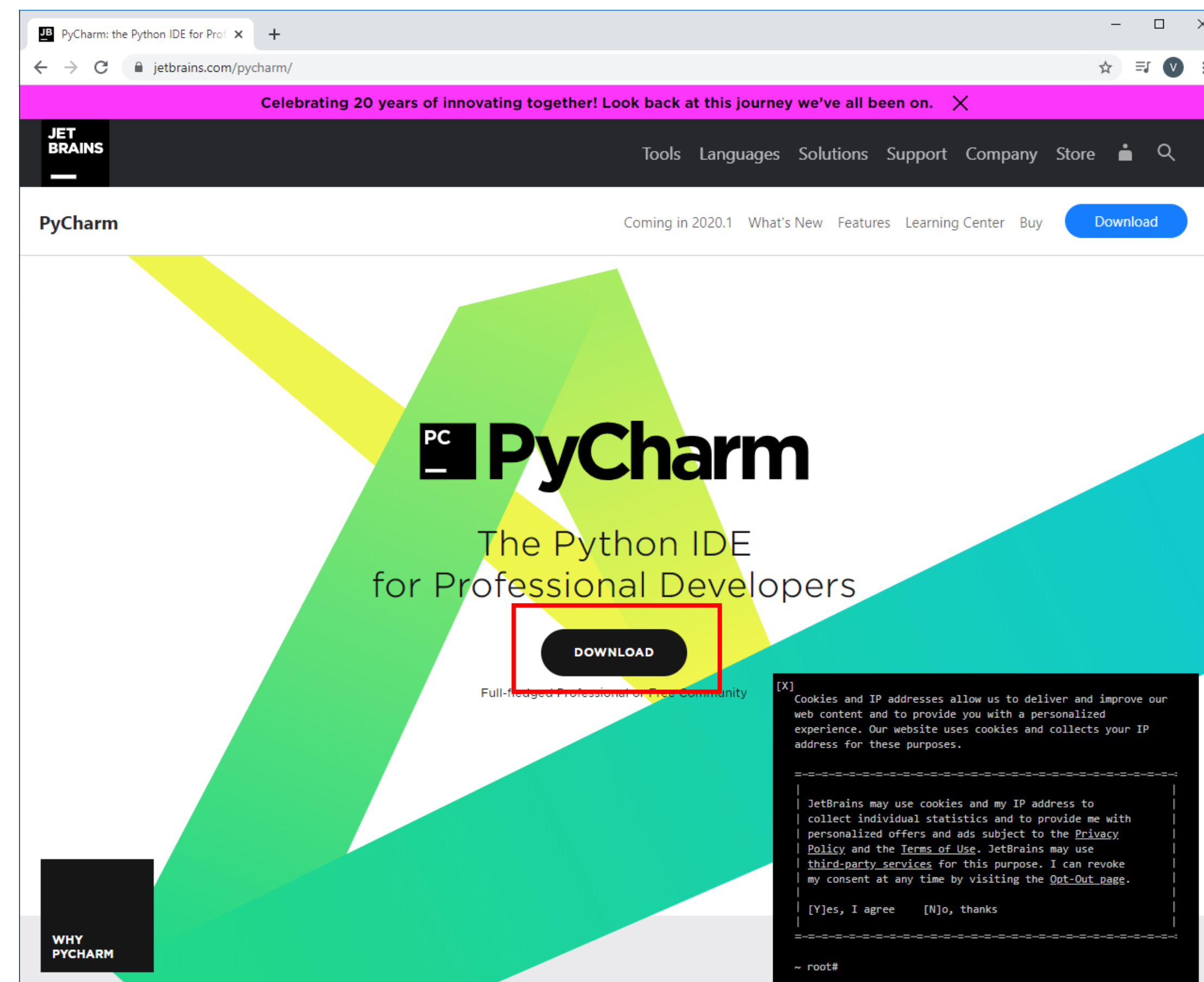
- Acesse <http://codeshare.io> e faça o cadastro
- Tendo cadastrado, na tela inicial clique em “Share Code Now”
- Um editor de texto vai aparecer. Na coluna da direita, clique na engrenagem, e em **Syntax** marque a opção **Python**
- Para compartilhar, clique em **Share**, no canto superior da janela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





# Instalando e configurando o PyCharm

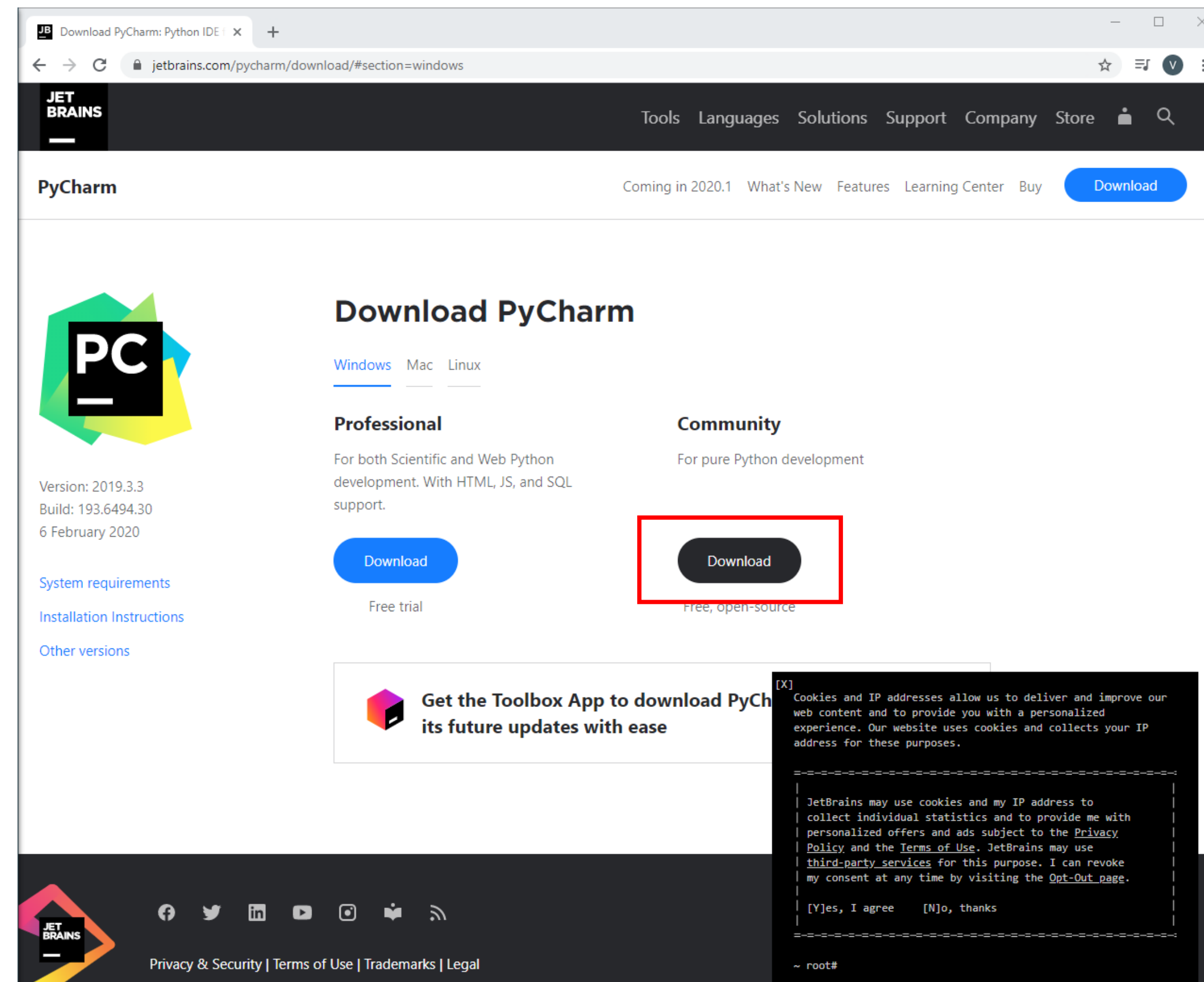
- O **PyCharm** é um dos editores (ou IDEs) mais usados para a programação de aplicações em Python. Para baixar e instalar, primeiro acesse a página <https://www.jetbrains.com/pycharm/> e clique em “Download”.





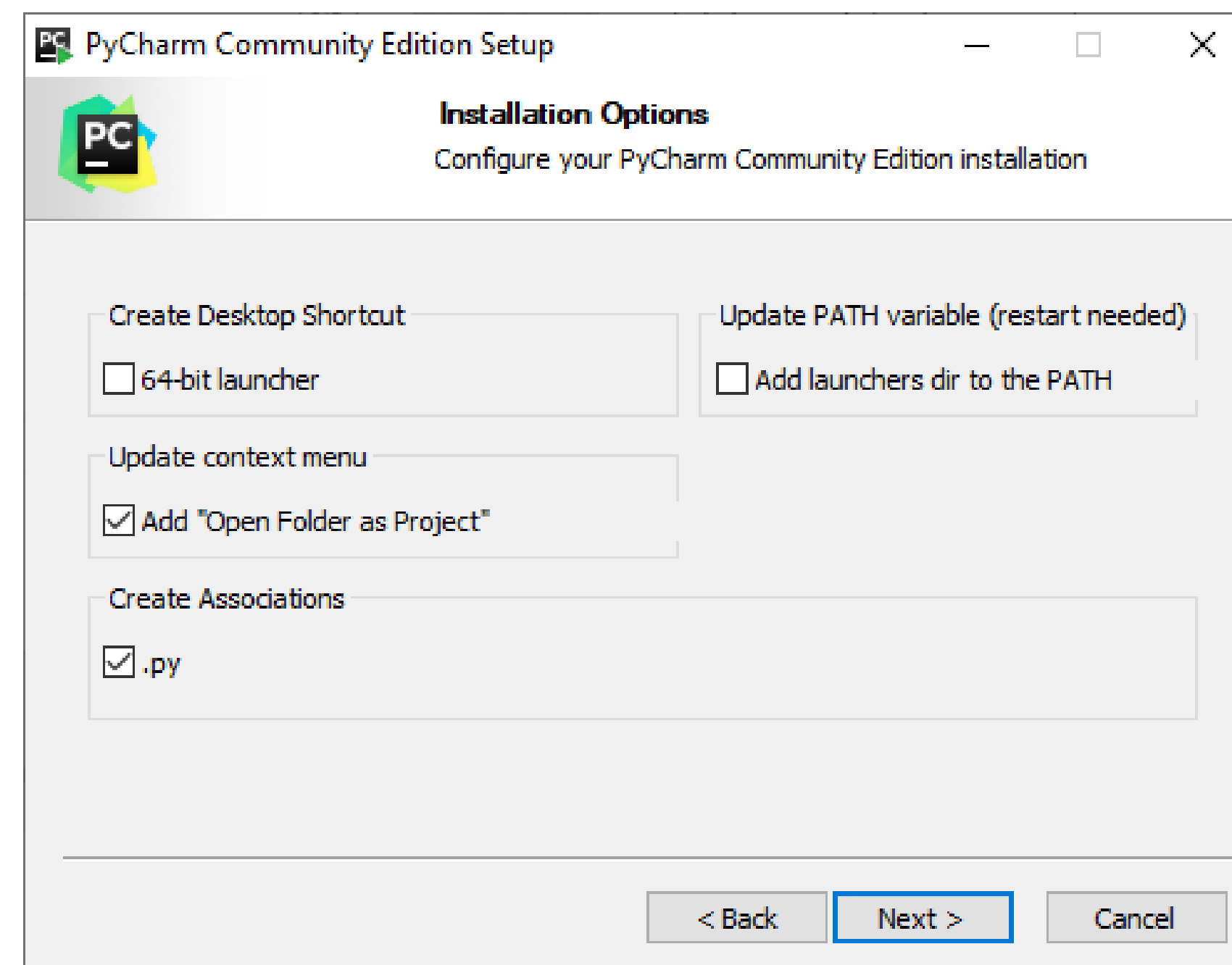
# Instalando e configurando o PyCharm

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



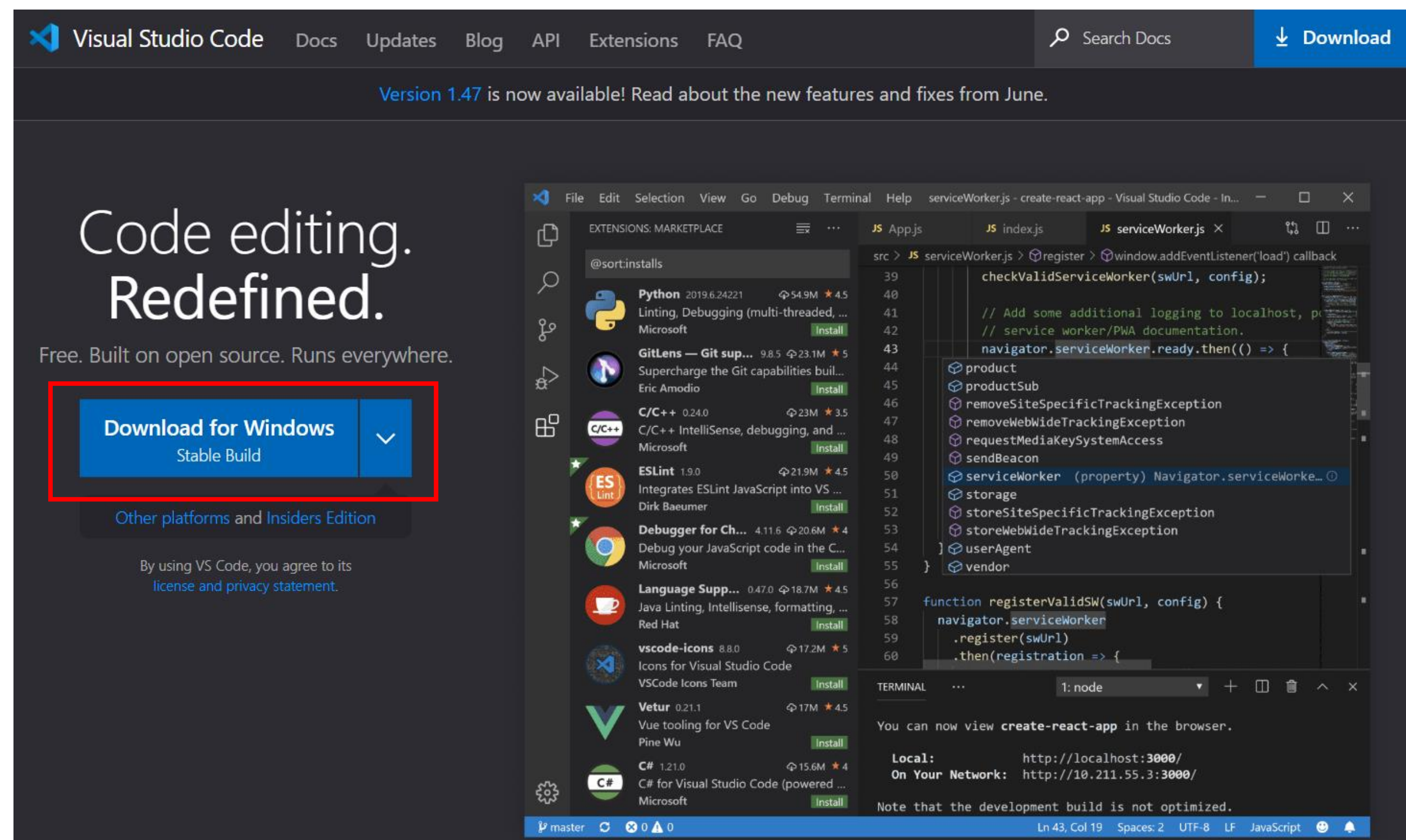
# Instalando e configurando o PyCharm

- 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

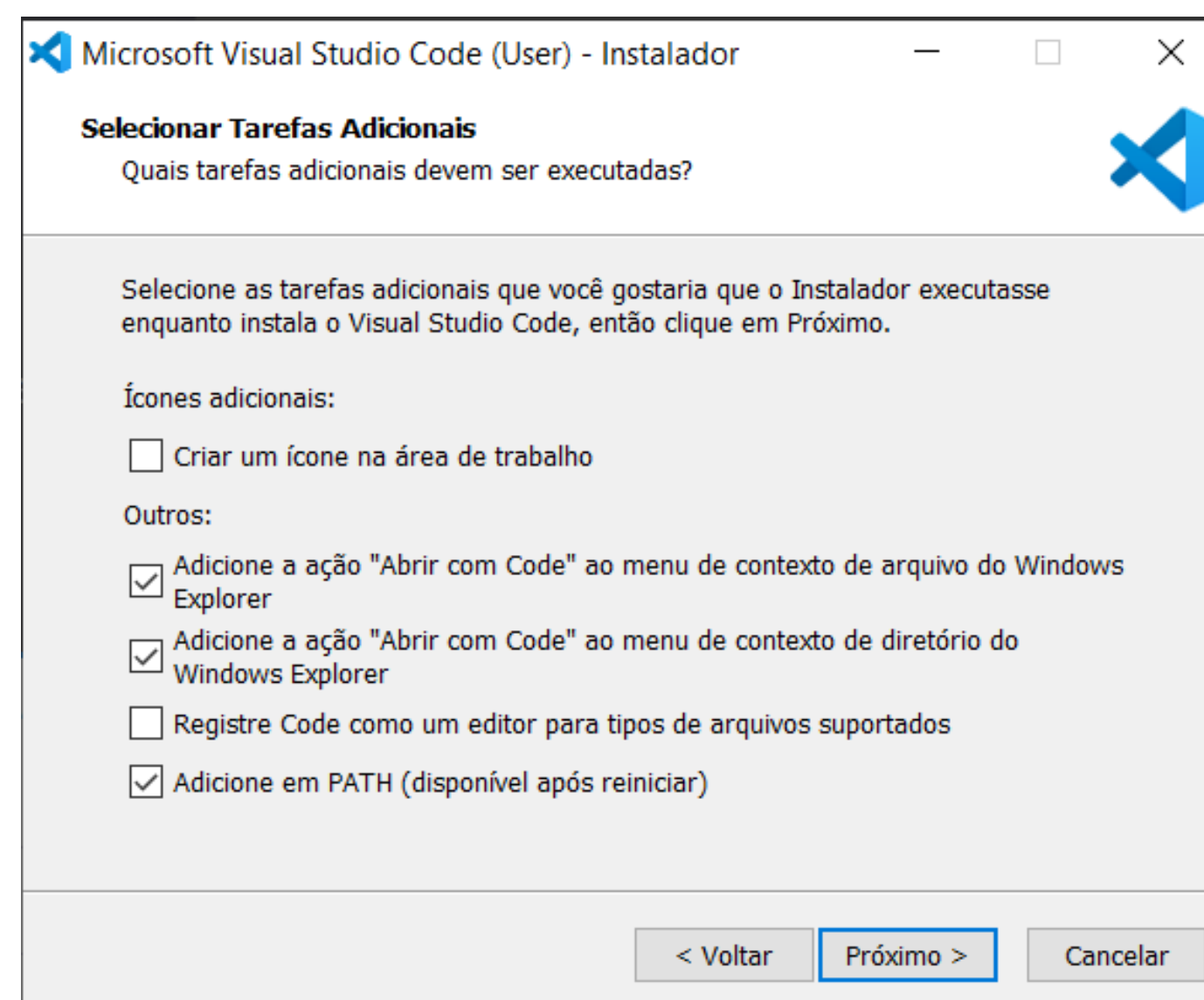
- O **VSCode** é um dos IDEs recentes mais famosos para basicamente qualquer linguagem de programação. Possui inúmeras extensões que facilitam e customizam o editor para a necessidade de cada programador. Para baixar e instalar, primeiro acesse a página <https://code.visualstudio.com/> e clique em “Download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

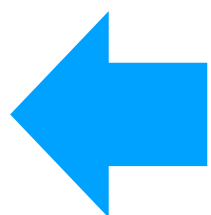




# Instalando e configurando o VSCode

- 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”

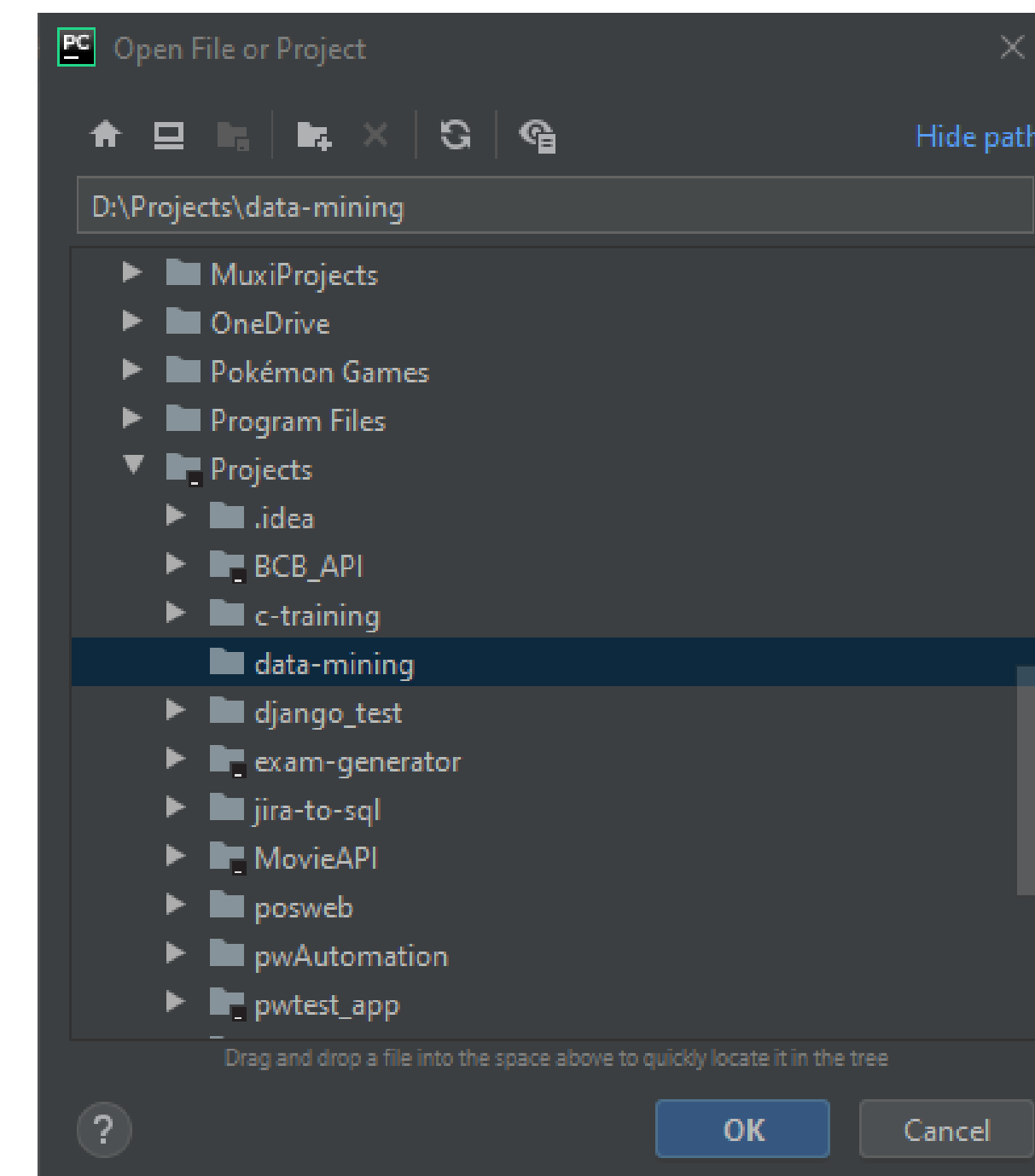
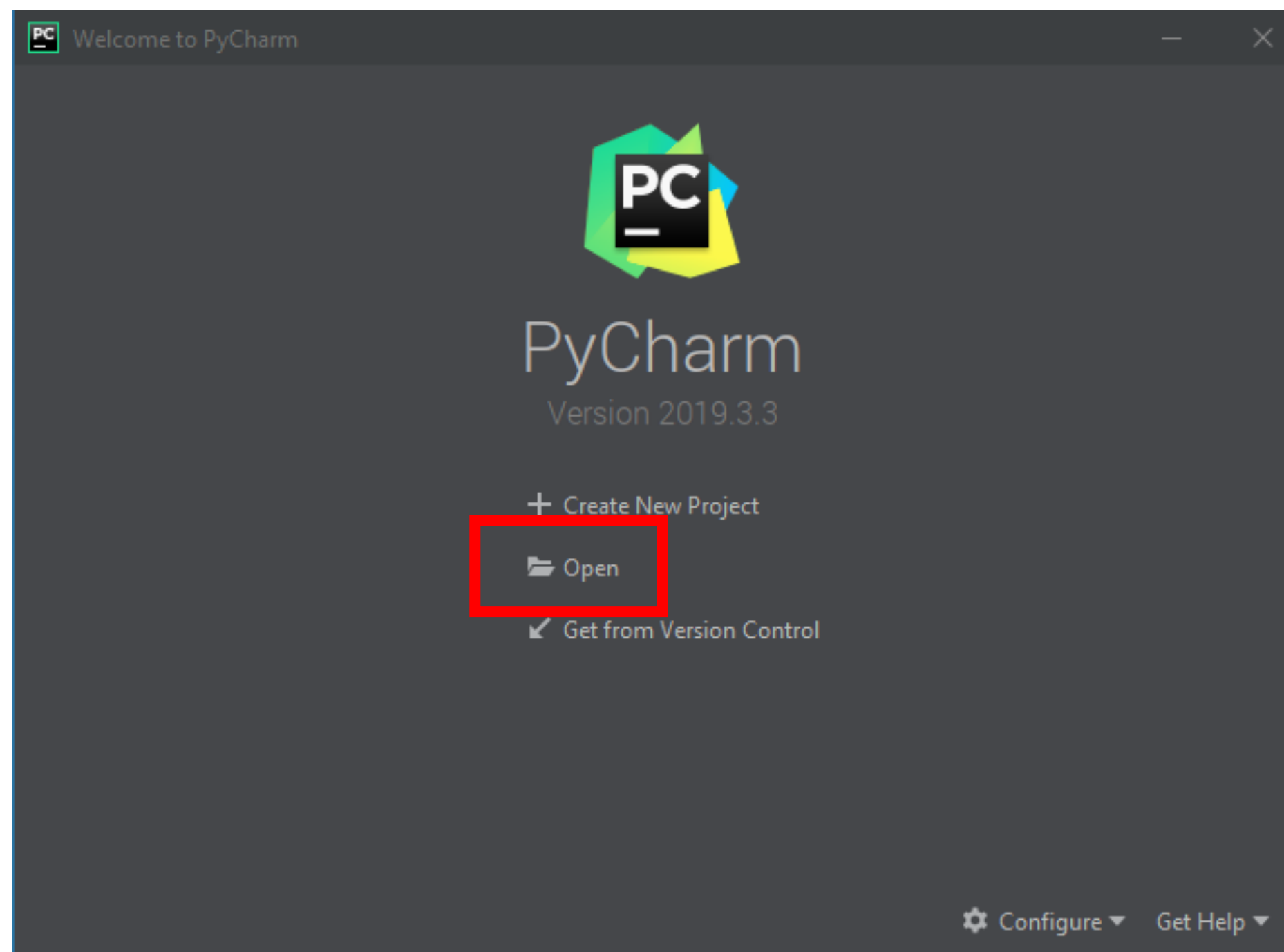




# Utilizando o PyCharm

# Iniciando o PyCharm

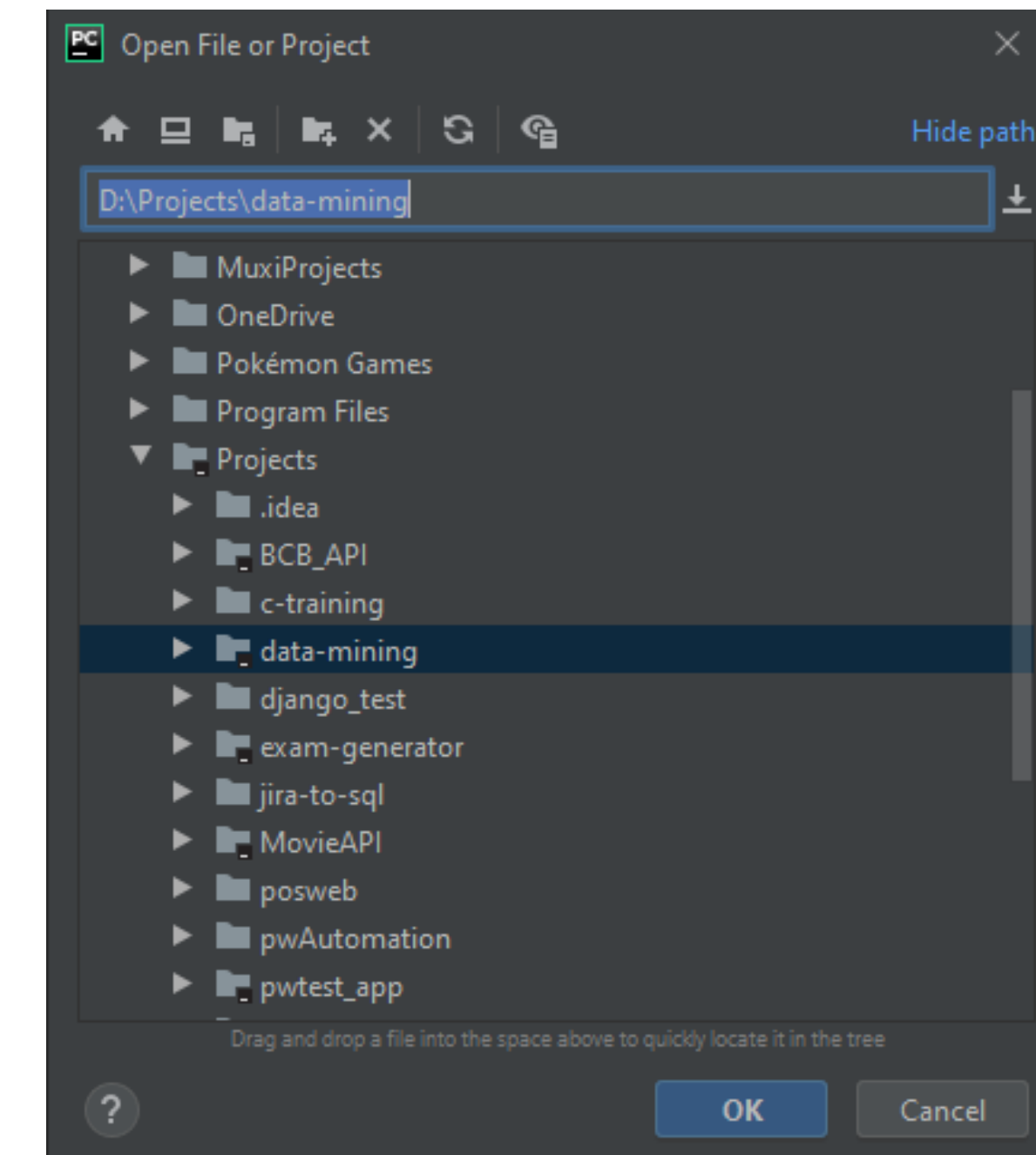
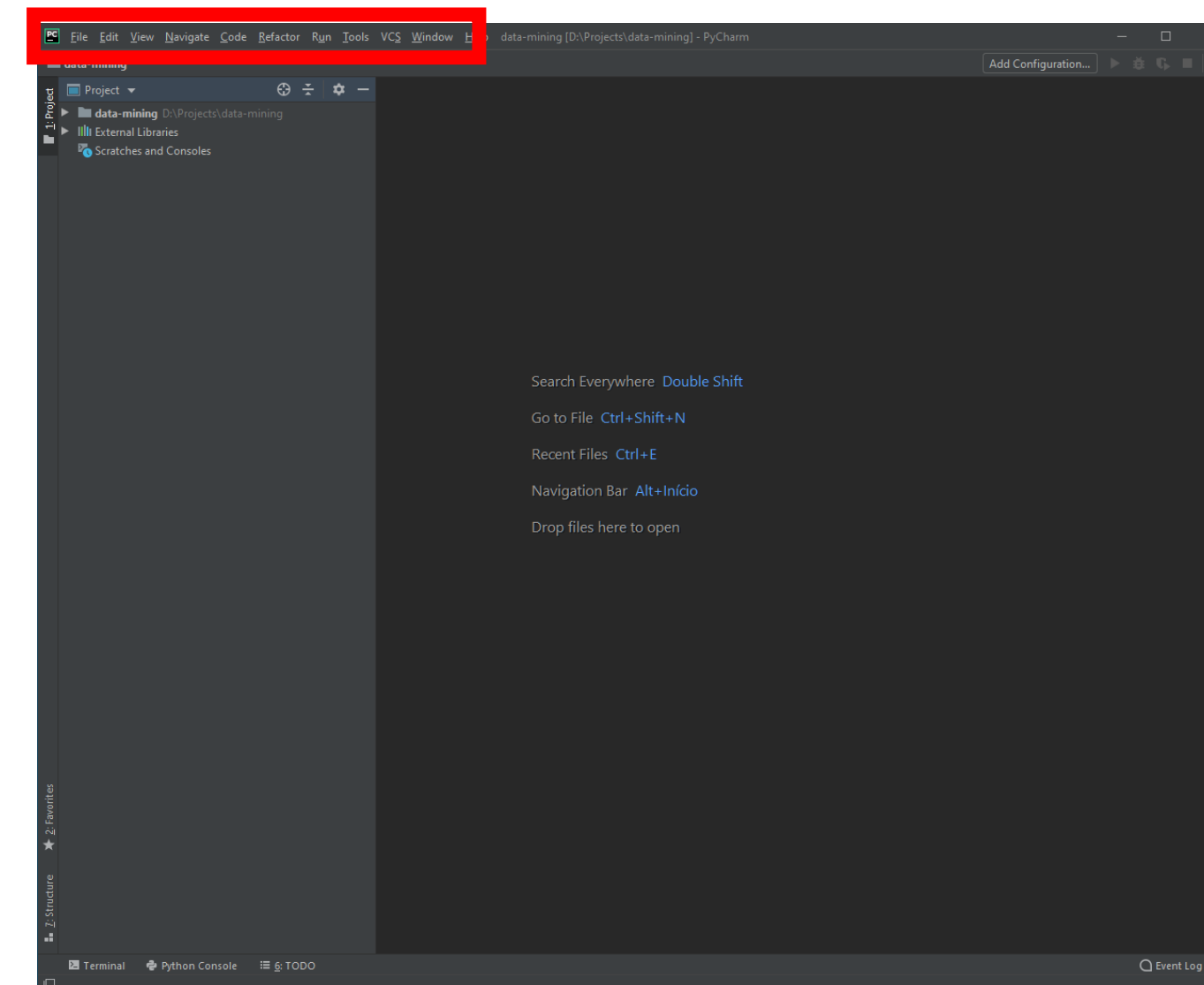
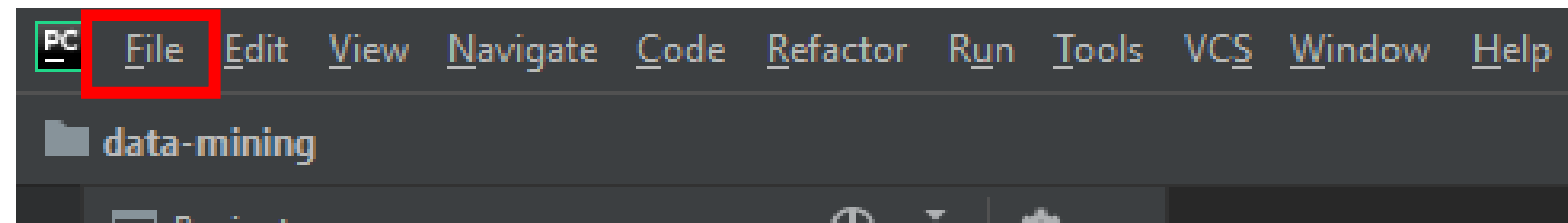
- **Se essa é a primeira vez que abre o PyCharm:**
  - Garanta que você possui uma pasta com o projeto desejado (p.ex., **D:\Projetos\data-mining**);
  - Abra o PyCharm, e na tela inicial clique em **Open**. Selecione a pasta que você criou e clique em **Ok**;





# 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**;

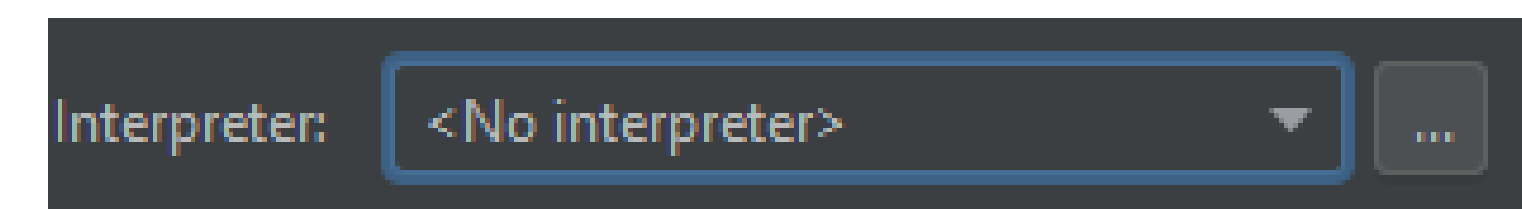
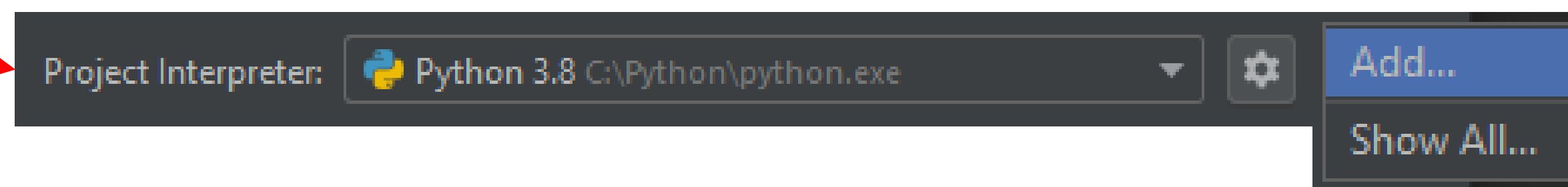


# Algumas configurações do PyCharm

- **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.

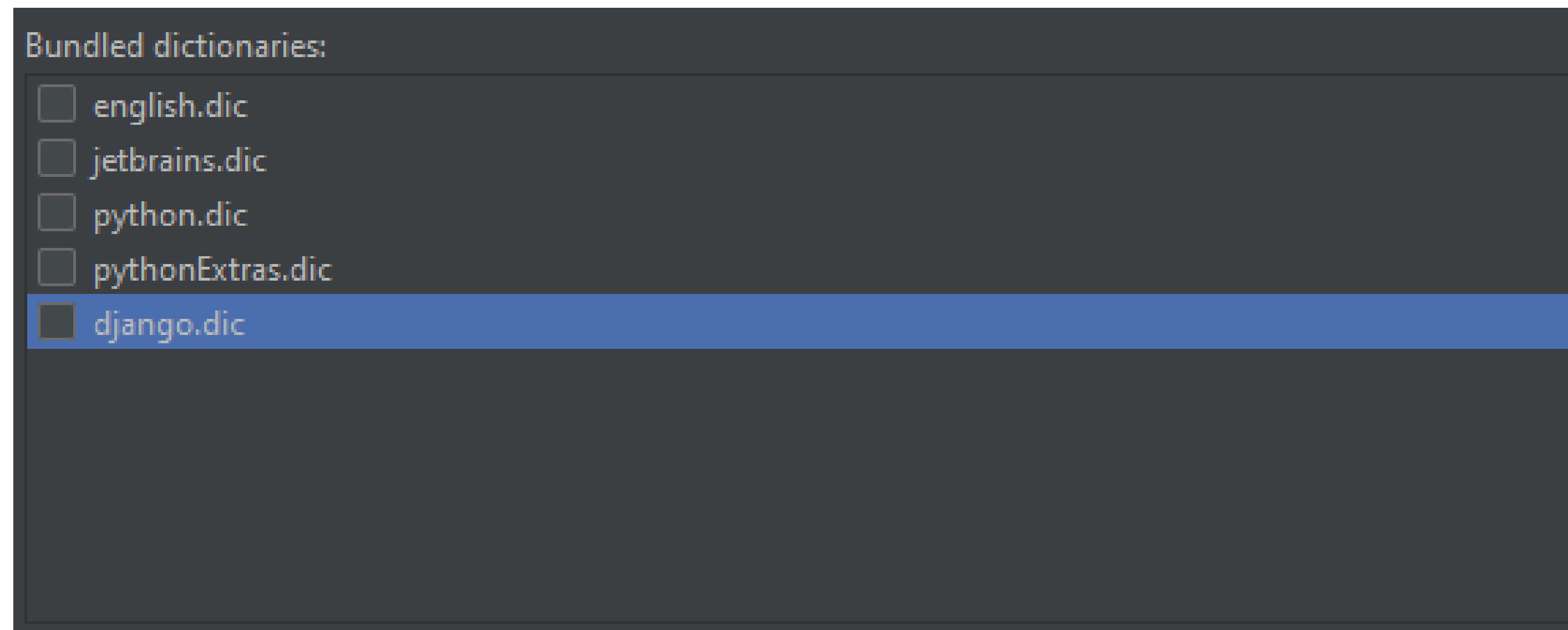
# Algumas configurações do PyCharm

- **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**.



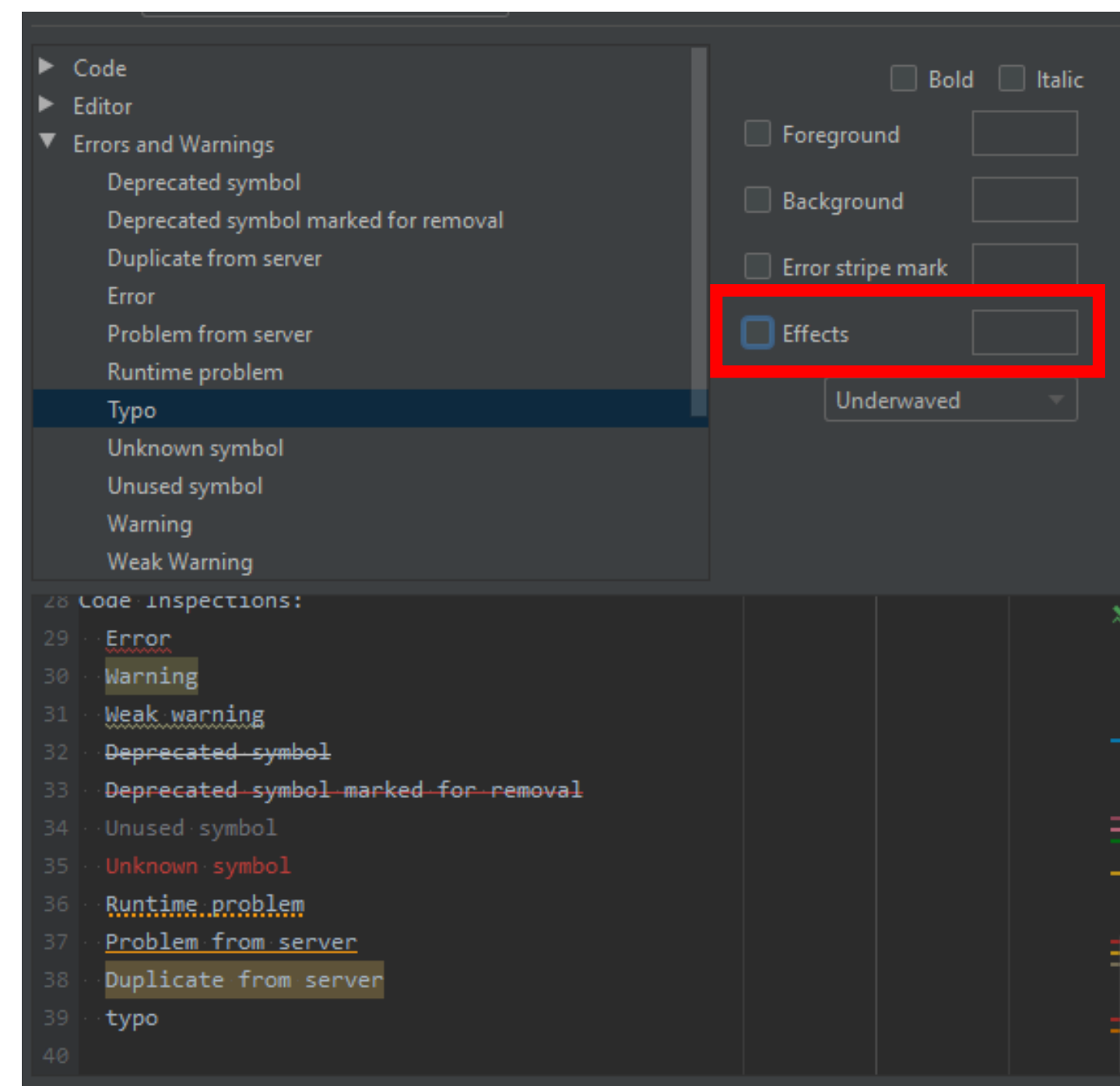
# Algumas configurações do PyCharm

- **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.



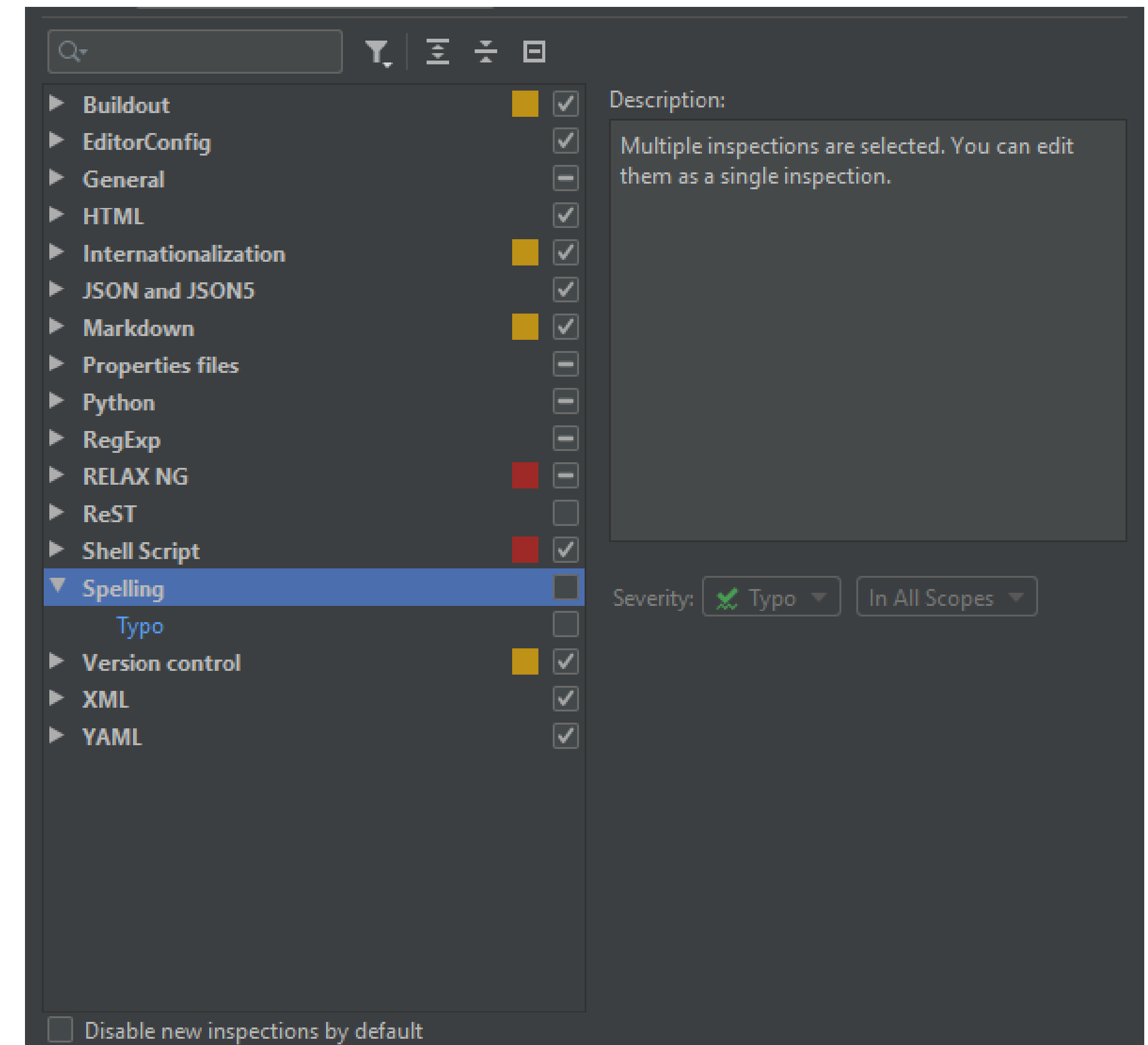
# Algumas configurações do PyCharm

- 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

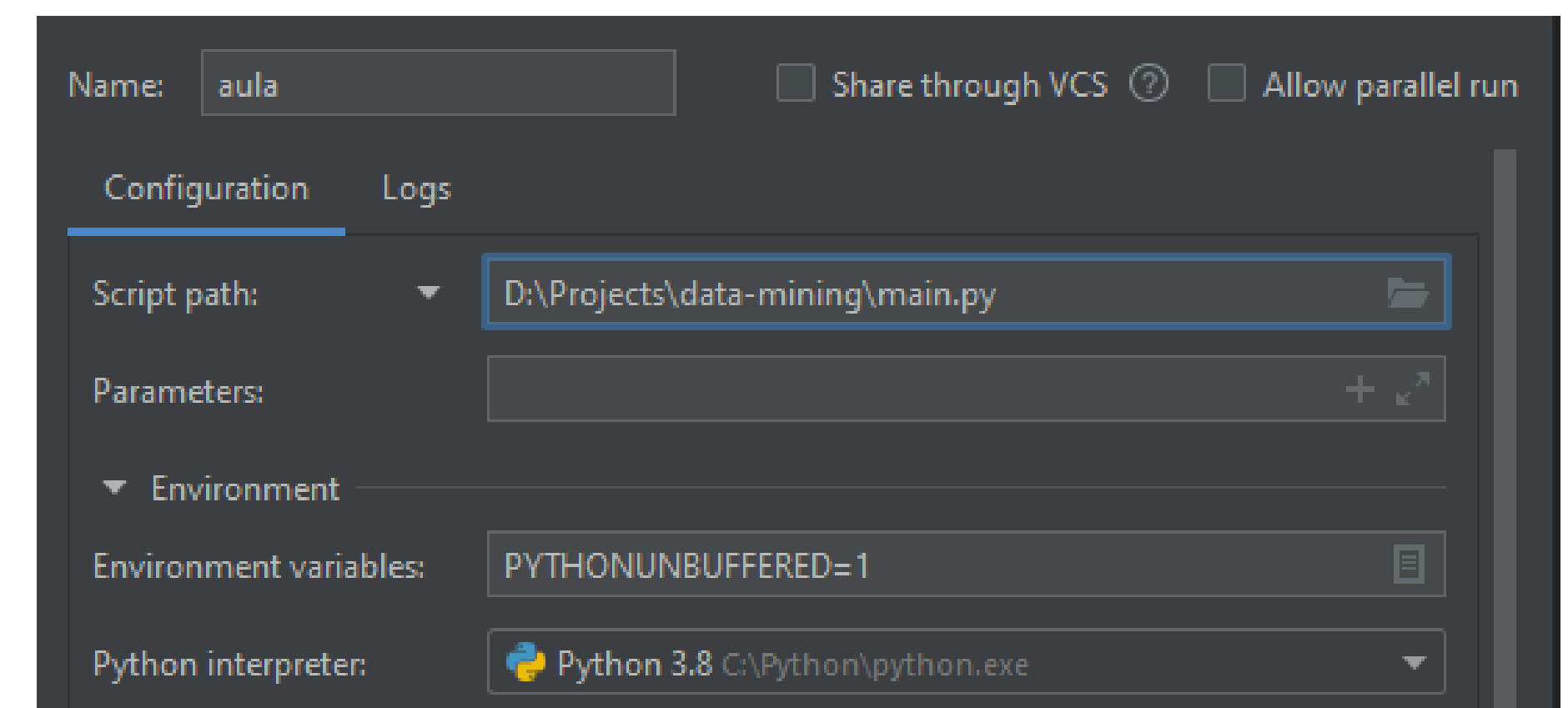
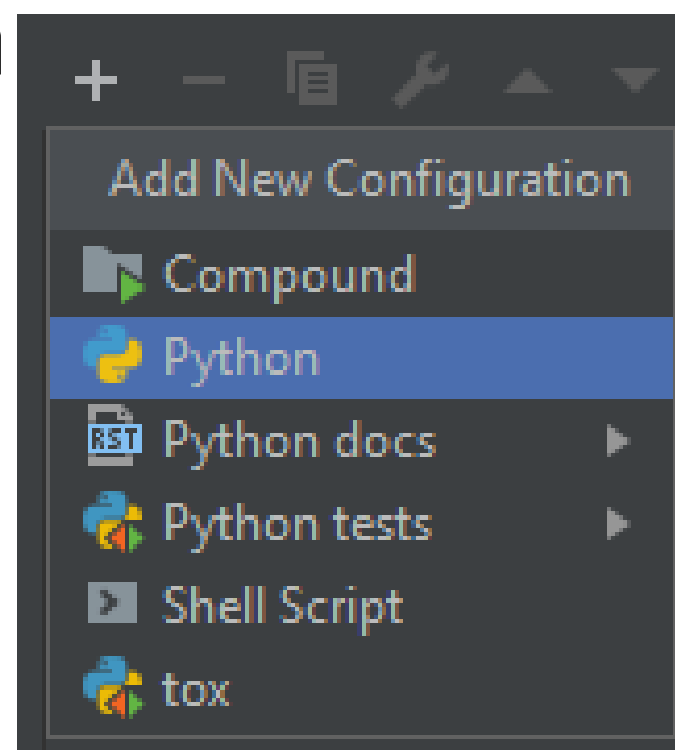
- **Ajustar os destaques de inspeções:**
  - Clique em **File > Settings**;
  - Na janela que aparecer, procure por **Editor > Inspections**. Desmarque o campo **Spelling**;
  - Na seção **Python**, recomendo *desmarcar* algumas opções para simplificar o aprendizado:
    - *Boolean variable check can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

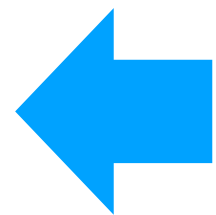




# Algumas configurações do PyCharm

- **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**.

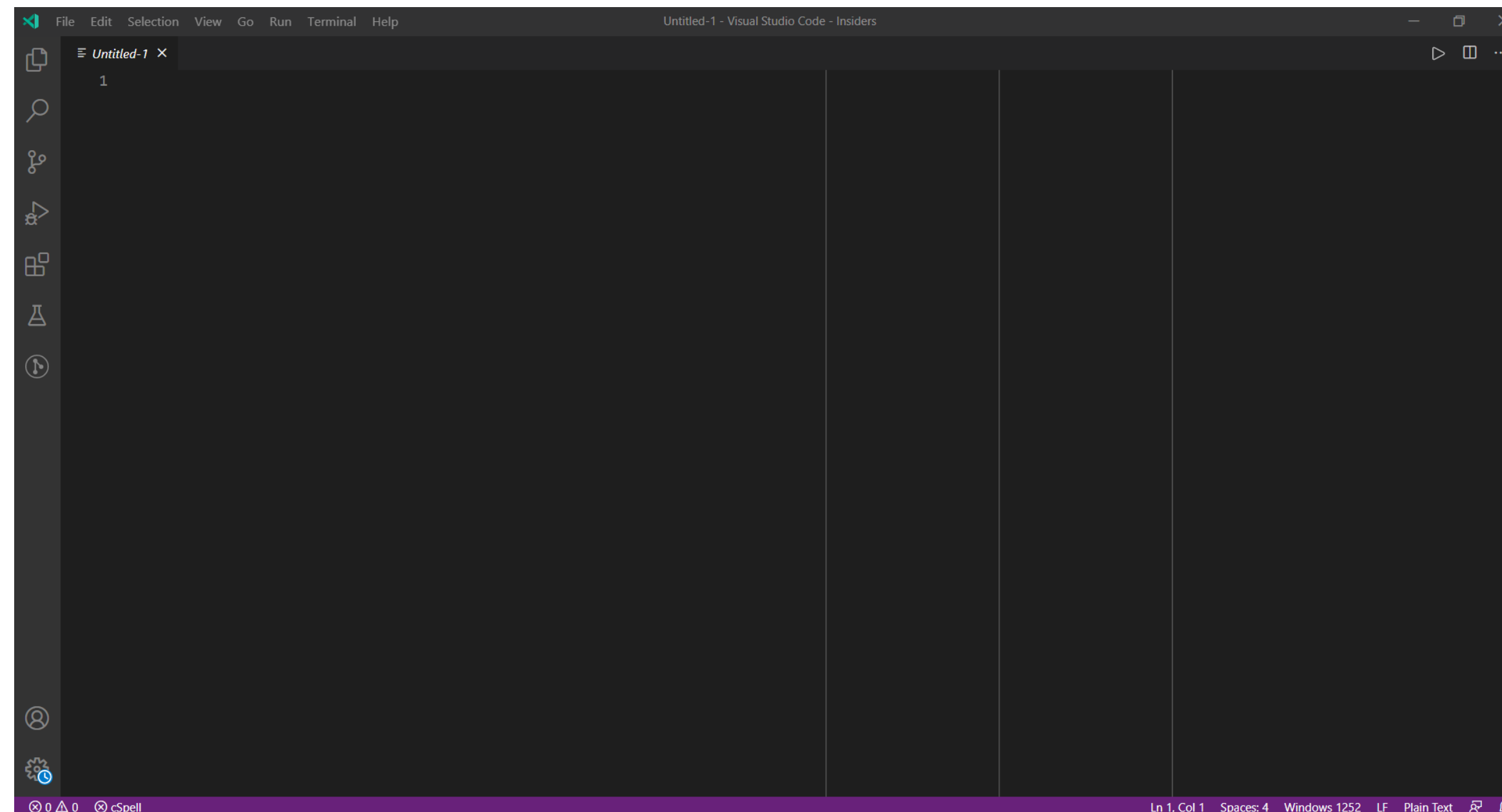




# Utilizando o VSCode

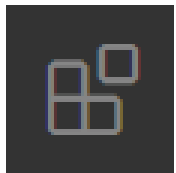
# Iniciando o VSCode

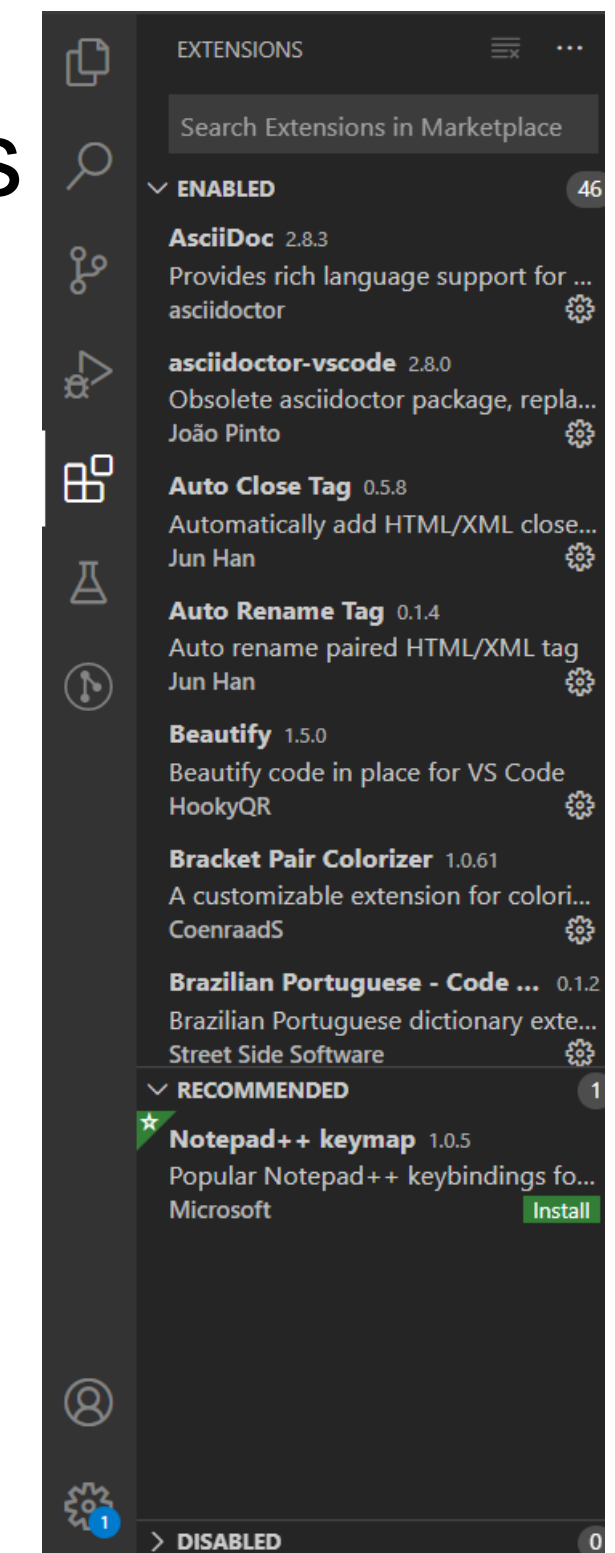
- **Se essa é a primeira vez que abre o VSCode:**
  - Garanta que você possui uma pasta com o projeto desejado (p.ex., **D:\Projetos\algoritmos**);
  - Abra o VSCode, e na tela inicial clique em **File > Open Folder**. Selecione a pasta que você criou e clique em **Selecionar Pasta**.



# 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;
- Para o curso, vamos precisar das extensões Python, Beautify, Bracket Pair Colorizer e Visual Studio IntelliCode.



# 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

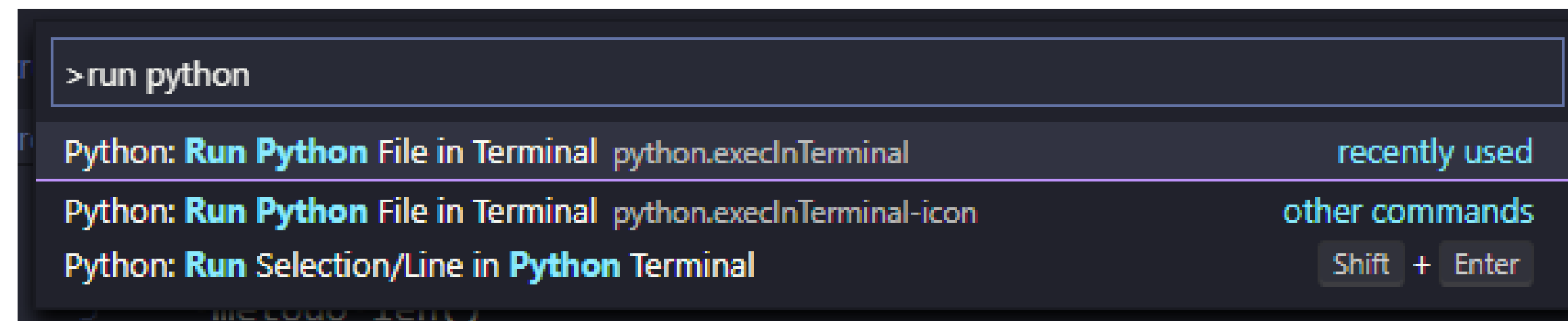


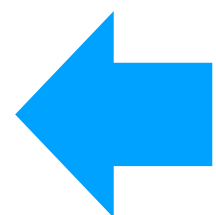
# Iniciando o VSCode

- **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

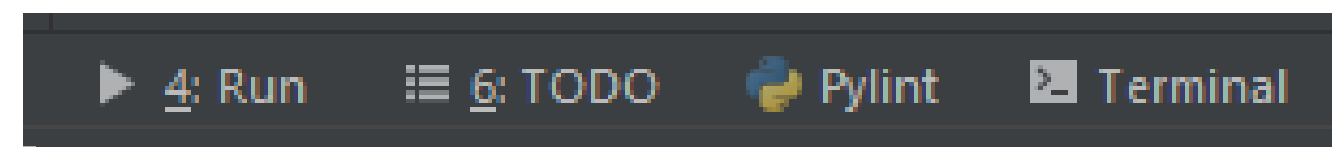
- O pylint é um pacote do Python que auxilia na adequação do código aos padrões de programação da comunidade Python
- Para instalar o pylint...
  - ...para MacOS veja [aqui](#)
  - ...para Windows veja [aqui](#)
- Para executar o pylint basta entrar com o comando **pylint <caminho>**, com o caminho do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

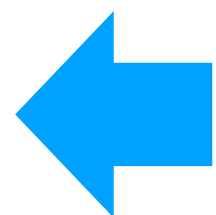
☒ Whether to lint Python files.

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 via 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>



OBRIGADO!



[www.ibmec.br](http://www.ibmec.br)

 /ibmec

 ibmec

 @ibmec\_oficial

 ibmec

