



Pycharm for Anaconda

INTRODUÇÃO

1. Informações Iniciais

- ▶ **PyCharm**

- ▶ *IDE cross-platform dedicada para Python (03/02/2010, 10yo)*
- ▶ *Desenvolvida em Java e Python*

- ▶ **Desenvolvedor:** JetBrains *(empresa tcheca, 14/08/2000, 20yo)*

- ▶ **Principais produtos:**

- ▶ **IntelliJ IDEA:** a mais popular IDE Java paga
 - ▶ **Android Studio:** principal IDE para desenvolvimento Android

	PyCharm Professional Edition	PyCharm Community Edition
Intelligent Python editor	✓	✓
Graphical debugger and test runner	✓	✓
Navigation and Refactorings	✓	✓
Code inspections	✓	✓
VCS support	✓	✓
Scientific tools	✓	
Web development	✓	
Python web frameworks	✓	
Python Profiler	✓	
Remote development capabilities	✓	
Database & SQL support	✓	



Free trial

Free, open-source

2. Versões

► <https://www.jetbrains.com/pycharm/>

Yearly billing Save 2 months

Monthly billing

PyCharm

Python IDE for professional developers



first year

US \$89.00

second year
third year onwards

US \$71.00
US \$53.00

[Get quote](#)

[Buy](#)

BEST OFFER

All Products Pack

Get all JetBrains desktop tools including 10 IDEs,
2 profilers, and 3 extensions.

► Includes 15 tools

first year

US \$249.00

second year
third year onwards

US \$199.00
US \$149.00

[Learn more](#) [Get quote](#)

[Buy](#)

3. Preços

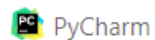
► <https://www.jetbrains.com/pycharm/buy/#personal?billing=yearly>

Yearly billing Save 2 months

Monthly billing

PyCharm

Python IDE for professional developers



per user, first year

US \$199.00

second year
third year onwards

US \$159.00
US \$119.00

[Get quote](#)

[Buy](#)

BEST OFFER

All Products Pack

Get all JetBrains desktop tools including 10 IDEs,
2 profilers, and 3 extensions.

► Includes 15 tools

per user, first year

US \$649.00

second year
third year onwards

US \$519.00
US \$389.00

[Learn more](#) [Get quote](#)

[Buy](#)

3. Preços

► <https://www.jetbrains.com/pycharm/buy/#commercial?billing=yearly>

4. Qual instalar?

Other Versions

Version 2020.2

2020.2.1 ▼

PyCharm Professional Edition

[2020.2.1 - Linux \(tar.gz\)](#)

[2020.2.1 - Linux with Anaconda plugin \(tar.gz\)](#)

[2020.2.1 - Windows \(exe\)](#)

[2020.2.1 - Windows with Anaconda plugin \(exe\)](#)

[2020.2.1 - macOS \(dmg\)](#)

[2020.2.1 - macOS with Anaconda plugin \(dmg\)](#)

PyCharm Community Edition

[2020.2.1 - Linux \(tar.gz\)](#)

[2020.2.1 - Linux with Anaconda plugin \(tar.gz\)](#)

[2020.2.1 - Windows \(exe\)](#)

[2020.2.1 - Windows with Anaconda plugin \(exe\)](#)

[2020.2.1 - macOS \(dmg\)](#)

[2020.2.1 - macOS with Anaconda plugin \(dmg\)](#)

Version: 2020.2.1 ([Release notes](#))

Build: 202.6948.78

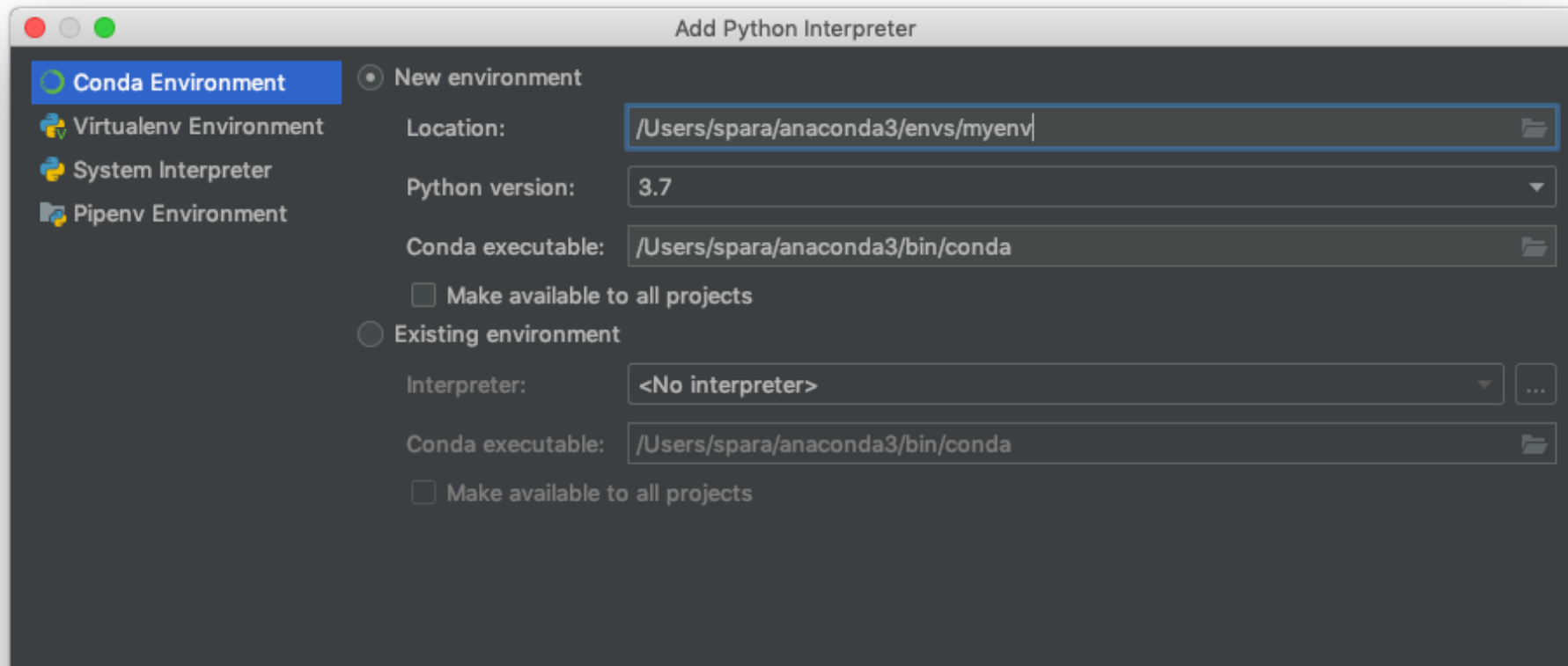
Released: 25 August 2020

[PyCharm Professional Edition third-party software](#)

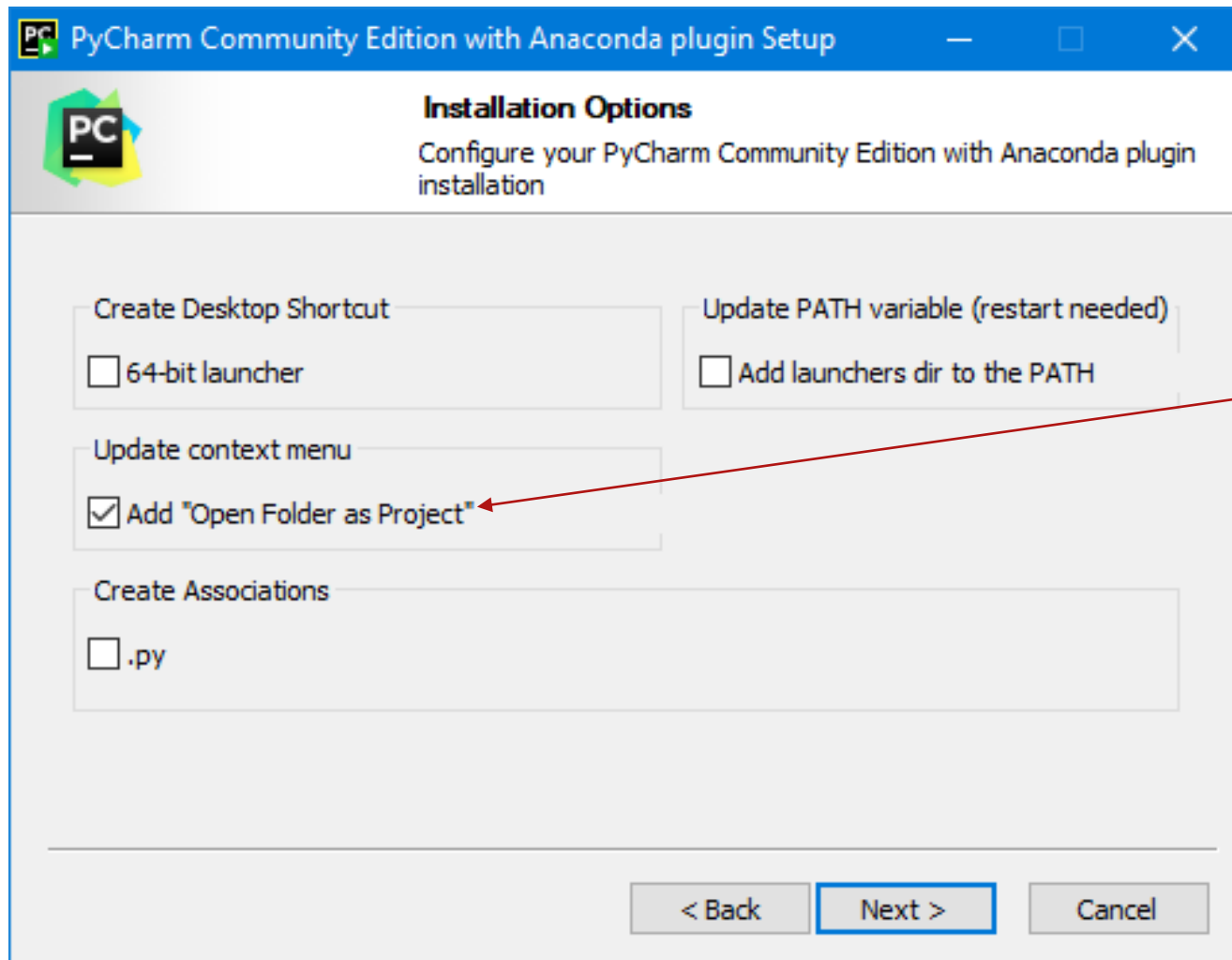
[PyCharm Community Edition third-party software](#)

► <https://www.jetbrains.com/pycharm/download/other.html>

4. O Plugin Anaconda



<https://docs.anaconda.com/anaconda/user-guide/tasks/pycharm/>



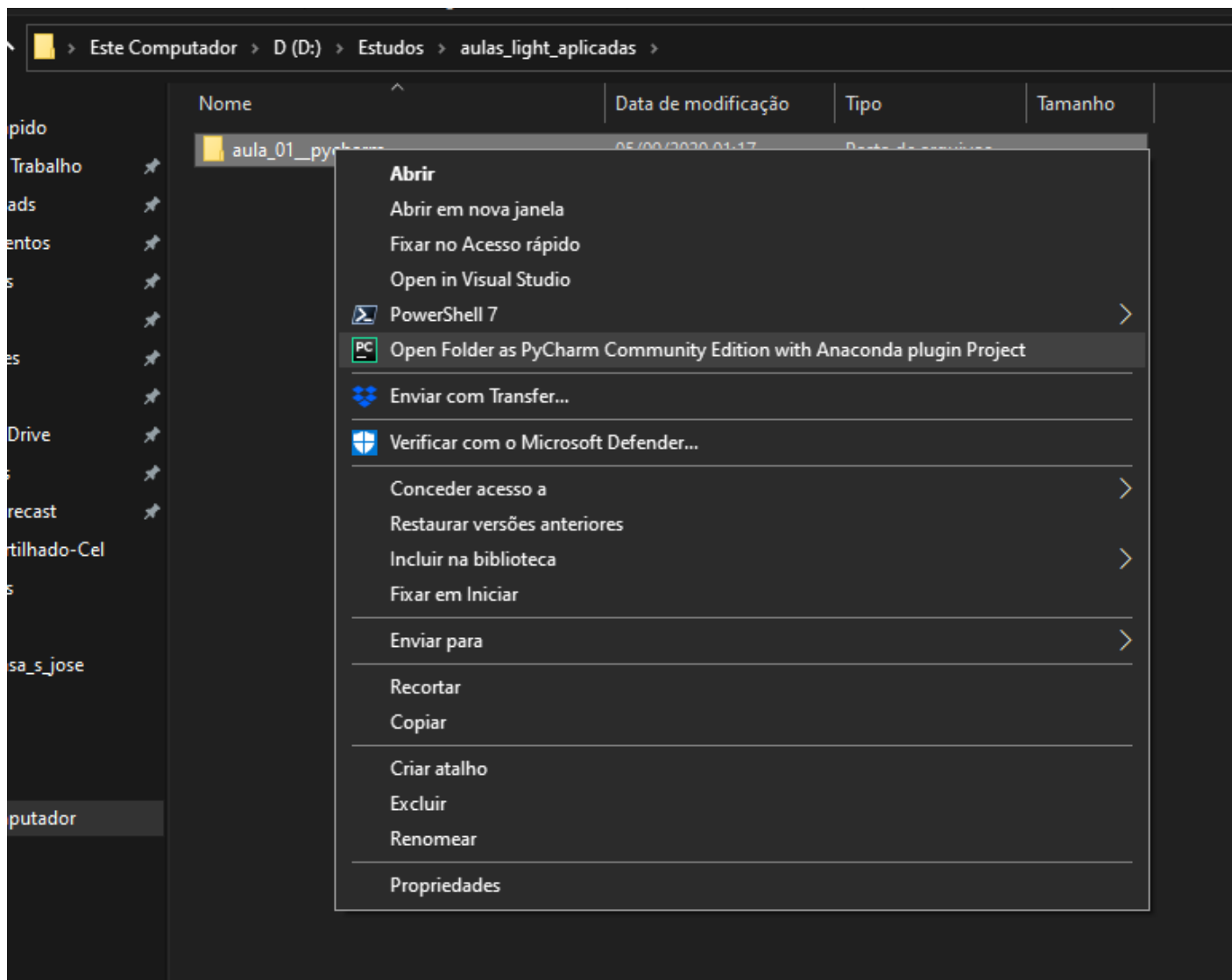
Para abrir/criar
projetos
rapidamente

5. Dica de instalação

(Windows)

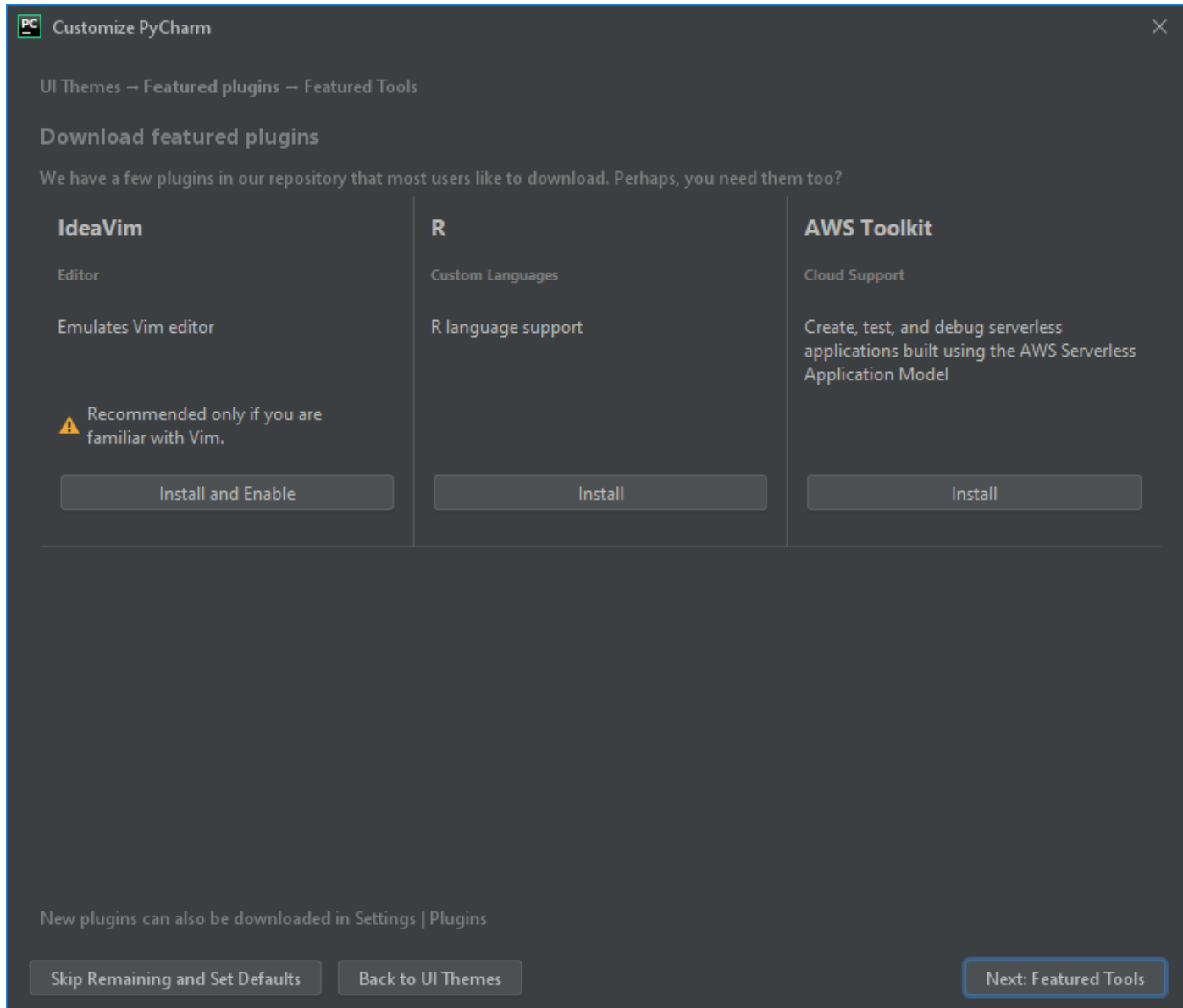
Marcar:

Add "Open Folder as Project"



5. Dica de instalação

Para criar ou abrir um projeto rapidamente



6. Plugins

Plugin para emular Vim

7. Personalizações

ESQUEMA DE CORES: MONOKAI

FONTE: JETBRAINS MONO 16

COMENTÁRIOS: ITÁLICO

```
def mostrar_debugger_jump():  
    # Use a breakpoint in the code  
    hex_2_ascii = dict()  
    for i in range(32, 126):  
        hex_2_ascii[hex(i)] = chr(i)  
  
    dec_2_ascii = dict()  
    for i in range(32, 145):  
        dec_2_ascii[i] = chr(i)  
  
    return hex_2_ascii, dec_2_ascii
```

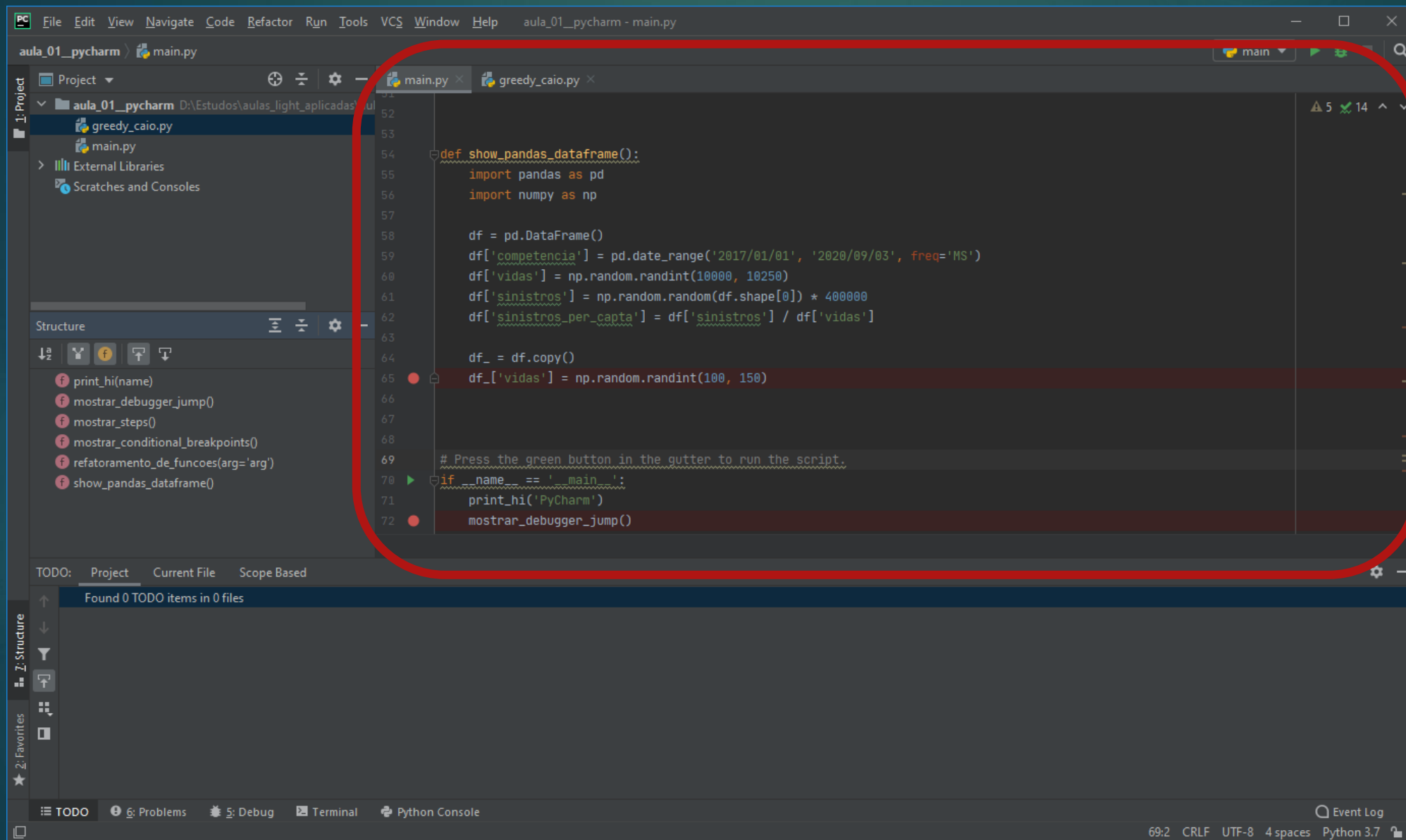
```
def mostrar_steps():  
  
    hex_2_ascii, dec_2_ascii = most  
  
    print(f'Tabela Hex-ASCII: {hex_  
  
    print(f'Tabela Dec-ASCII: {dec_2
```

```
if name == 'main':
```

Separador de funções e métodos

SETTINGS -> GENERAL -> EDITOR -> APPEARANCE -> SHOW METHOD SEPARATORS

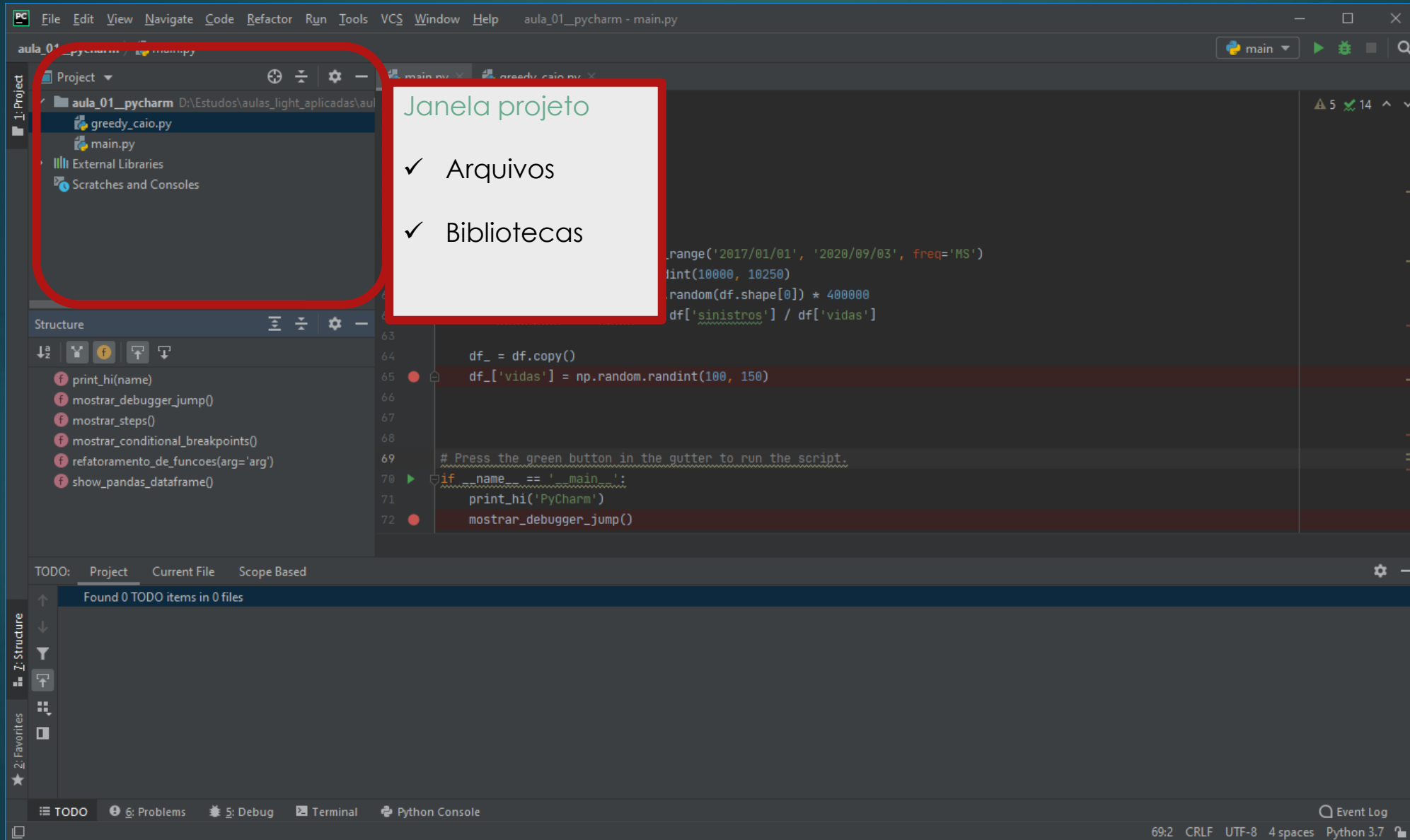
7. Janela: editor



Janela de edição
de código

- ✓ Arquivos organizados em abas
- ✓ Breakpoints
- ✓ Execução da função ou trecho

7. Janela: projeto



7. Janela: estrutura do arquivo atual

The screenshot shows the PyCharm IDE interface. The main editor displays the code for `main.py` and `greedy_caio.py`. The `Structure` window is open on the left, showing the following items:

- numero
- print_hi(name)
- mostrar_debugger_jump()
- mostrar_steps()
- mostrar_conditional_breakpoints()
- refatoramento_de_funcoes(arg='arg')
- show_pandas_dataframe()

The `TODO` list at the bottom shows two items found in the current file:

- (45, 11) # TODO: mostrar breakpoint condicional
- (50, 7) # TODO: adicionar novos argumentos

A red box highlights the `Structure` window and the `TODO` list. A red arrow points from the `Structure` window to the `TODO` list.

Janela estrutura do arquivo

- ✓ Funções
- ✓ Variáveis
- ✓ Classes

7. Janela: TODO

The screenshot displays the PyCharm IDE interface. The main editor window shows a Python file named `main.py` with the following code:

```
43 hex_2_ascii = dict()
44 for i in range(32, 126):
45     # TODO: mostrar breakpoint condicional
46     hex_2_ascii[hex(i)] = chr(i)
47
48
49 def refatoramento_de_funcoes(arg='arg'):
50     # TODO: adicionar novos argumentos
51     if arg == 'arg':
52         print('Nada mudou')
53
54     print(f'arg: {arg}')
55
56
57
58 def show_pandas_dataframe():
59     import pandas as pd
60     import numpy as np
61
62     df = pd.DataFrame()
63     df['competencia'] = pd.date_range('2017/01/01', '2020/09/03', freq='MS')
64     df['vidas'] = np.random.randint(10000, 10250)
```

The left sidebar shows the Project view with the file structure:

- aula_01_pycharm
 - greedy_caio.py
 - main.py
- External Libraries
- Scratches and Consoles

The Structure view shows the following elements:

- numero
- print_hi(name)
- mostrar_debugger_jump()
- mostrar_steps()
- mostrar_conditional_breakpoints()
- refatoramento_de_funcoes(arg='arg')
- show_pandas_dataframe()

The bottom panel shows the TODO window, which is highlighted with a red box. It displays the following TODO items:

- Found 2 TODO items in 1 file
 - aula_01_pycharm 2 items
 - main.py 2 items
 - (45, 11) # TODO: mostrar breakpoint condicional
 - (50, 7) # TODO: adicionar novos argumentos

A red line points from the TODO window to a callout box on the right.

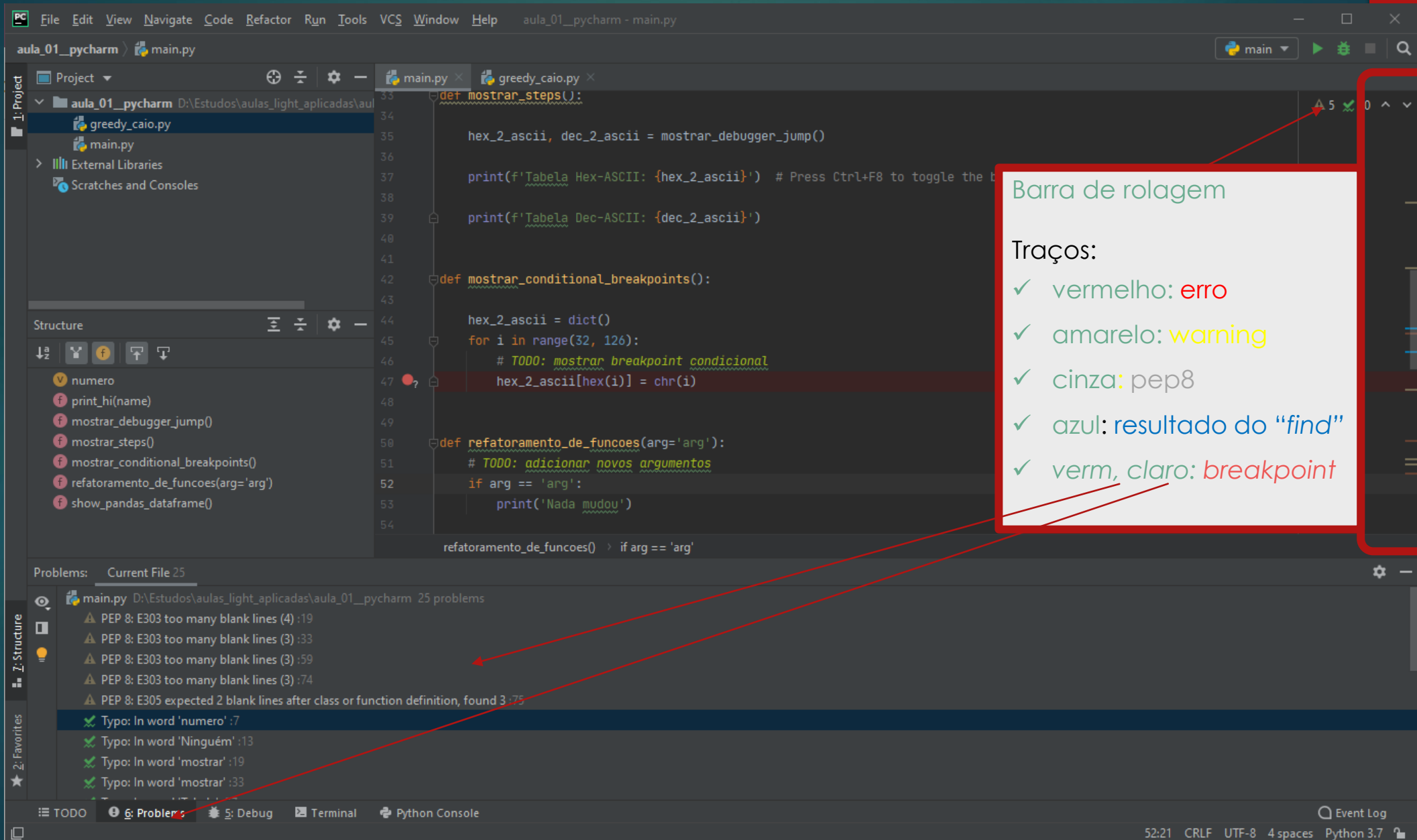
Gestor de tarefas

- ✓ Mostra todos textos acompanhados da palavra "TODO"

The bottom status bar shows the following information:

- 50:39 CRLF UTF-8 4 spaces Python 3.7

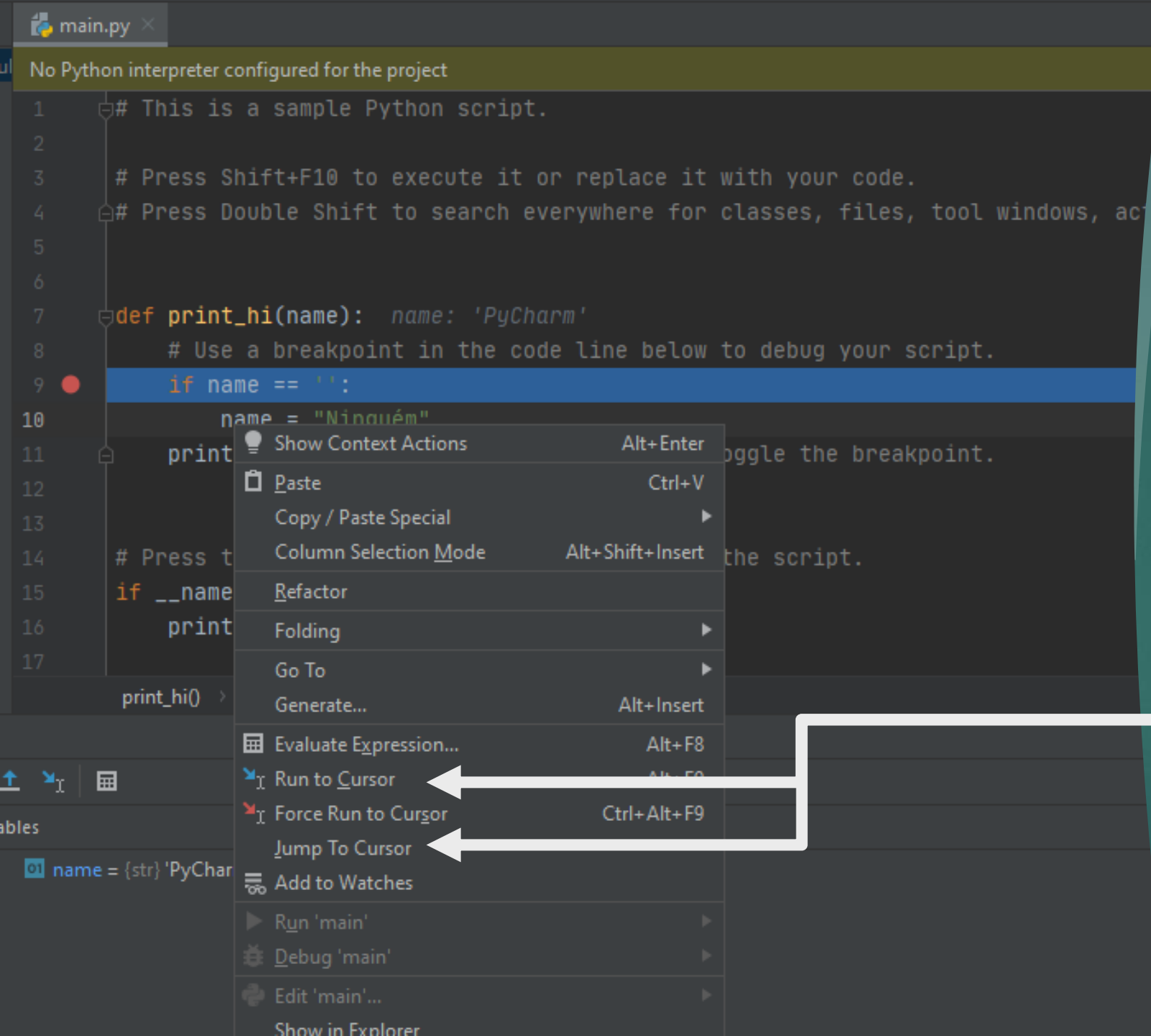
7. Janela: a barra de rolagem



7. Janela: debug

The image shows the PyCharm IDE interface with the following components and annotations:

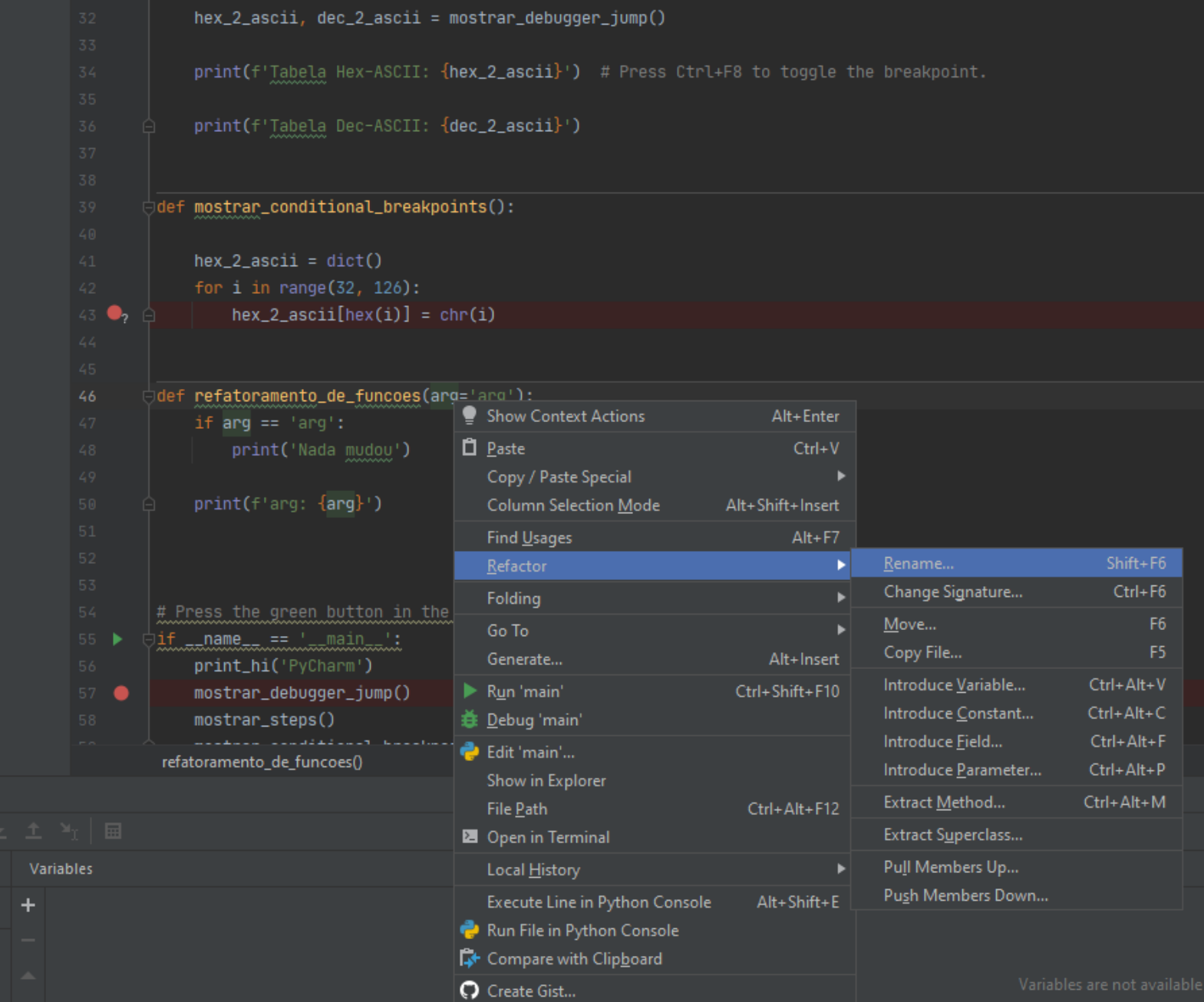
- Top Bar:** Contains the 'main' dropdown and a green play button. An annotation '1. Arquivo em que iniciará o debug' points to the 'main' dropdown.
- Run/Debug Buttons:** A green play button and a green bug icon. An annotation '2. Iniciar debug' points to the bug icon.
- Code Editor:** Displays the code for 'main.py'. A context menu is open over line 75, showing options: 'Run \'main\'' (Ctrl+Shift+F10), 'Debug \'main\'', and 'Edit \'main\'...'. An annotation 'Executa a partir do ponto selecionado' points to the 'Debug \'main\'' option.
- Structure View:** Located on the left, showing the project structure and a list of functions: 'numero', 'print_hi(name)', 'mostrar_debugger_jump()', 'mostrar_steps()', 'mostrar_conditional_breakpoints()', 'refatoramento_de_funcoes(arg='arg')', and 'show_pandas_dataframe()'. An annotation 'Mostrar painel de Debug' points to the 'Debug' tab in the bottom bar.
- Debug Panel:** Located at the bottom, showing the 'main' debug session. It includes a 'Debugger' tab, a 'Console' tab, and a 'Frames' section. An annotation 'Reexecuta último comando de debug' points to the 'Debugger' tab.
- Bottom Bar:** Contains tabs for 'TODO', 'Problems', 'Debug', 'Terminal', and 'Python Console'. The 'Debug' tab is active.
- Annotations:** A red box titled 'Painel de debug' contains a list of items: '✓ Console' and '✓ Debugger'.



8. Por que escolhi o PyCharm?

Motivo 1:

O DEBUGGER!



8. Por que escolhi o PyCharm?

Motivo 2:

REFATORAÇÃO:

renomear

```

def mostrar_steps():

    hex_2_ascii, dec_2_ascii = mostrar_debugger_jump()

    print(f'Tabela Hex-ASCII: {hex_2_ascii}') # Press Ctrl+F8 to toggle the breakpoint.

    print(f'Tabela Dec-ASCII: {dec_2_ascii}')

def mostrar_conditional_breakpoints():

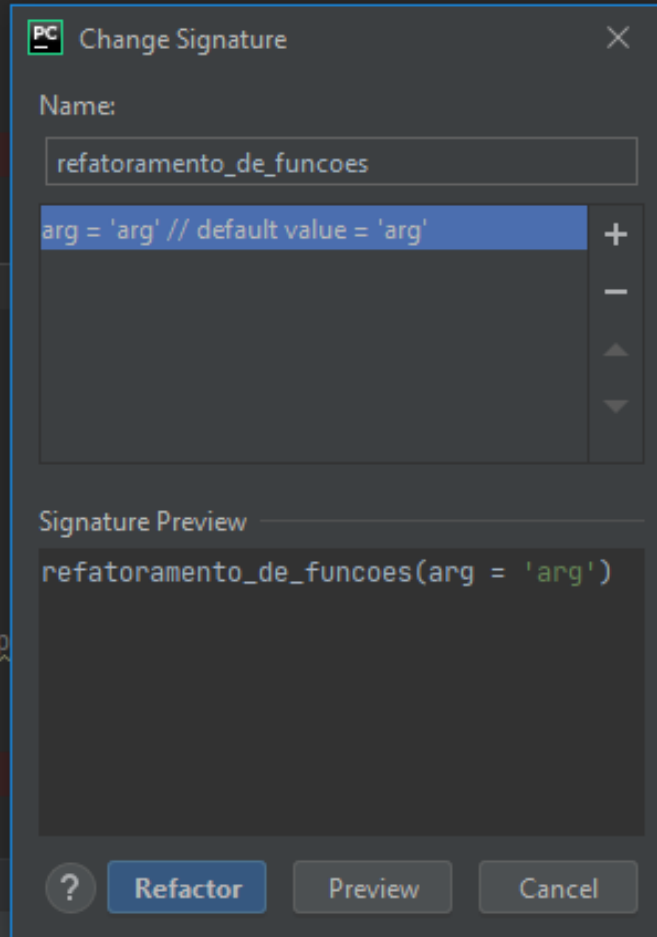
    hex_2_ascii = dict()
    for i in range(32, 126):
        hex_2_ascii[hex(i)] = chr(i)

def refatoramento_de_funcoes(arg='arg'):
    if arg == 'arg':
        print('Nada mudou')

    print(f'arg: {arg}')

# Press the green button in the gutter to run the script
if __name__ == '__main__':
    print_hi('PyCharm')
    mostrar_debugger_jump()
    mostrar_steps()
    mostrar_conditional_breakpoints()
    refatoramento_de_funcoes()

```

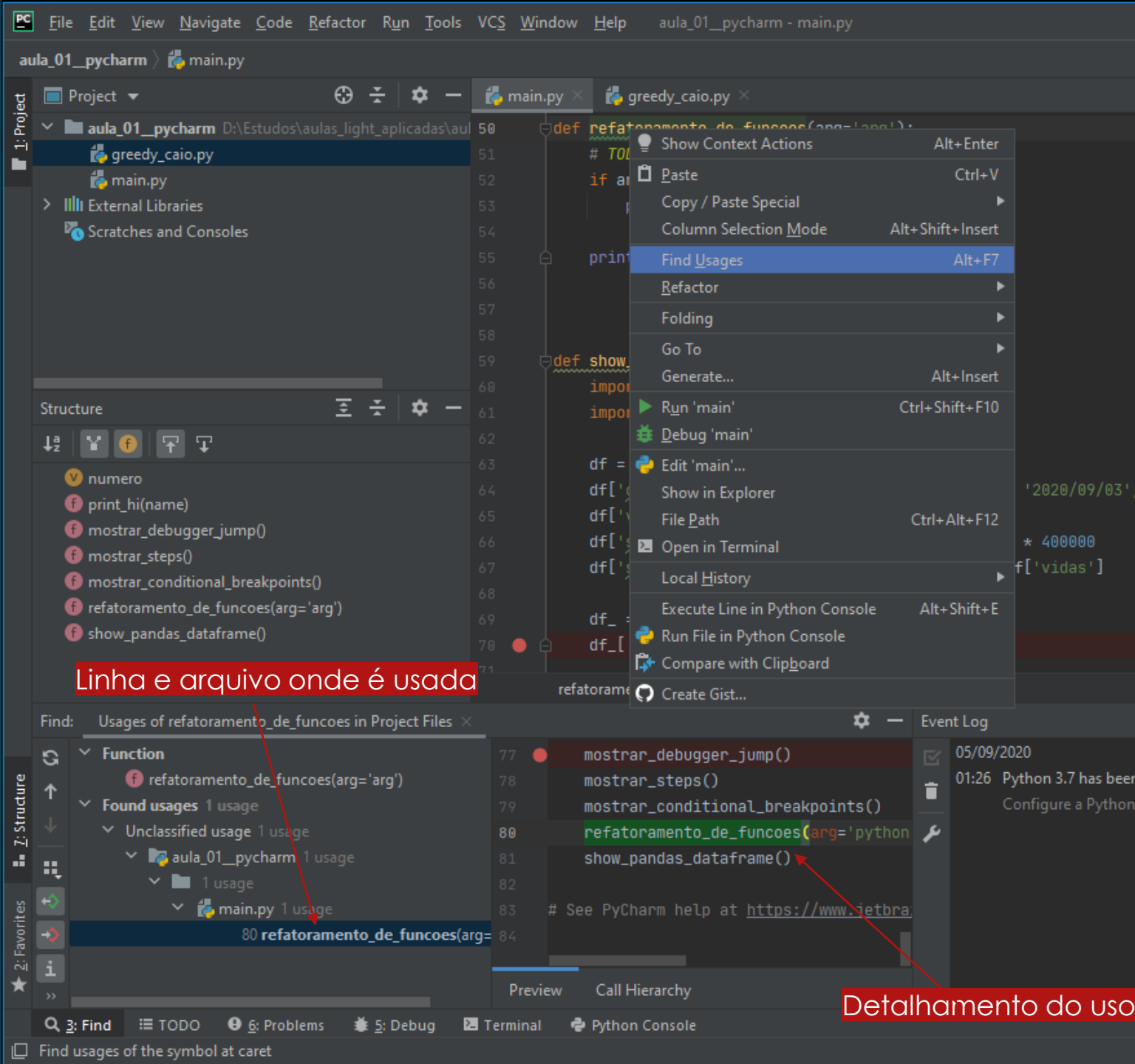


8. Por que escolhi o PyCharm?

Motivo 3:

REFATORAÇÃO:

mudança da
assinatura da
função/método



Linha e arquivo onde é usada

Detalhamento do uso

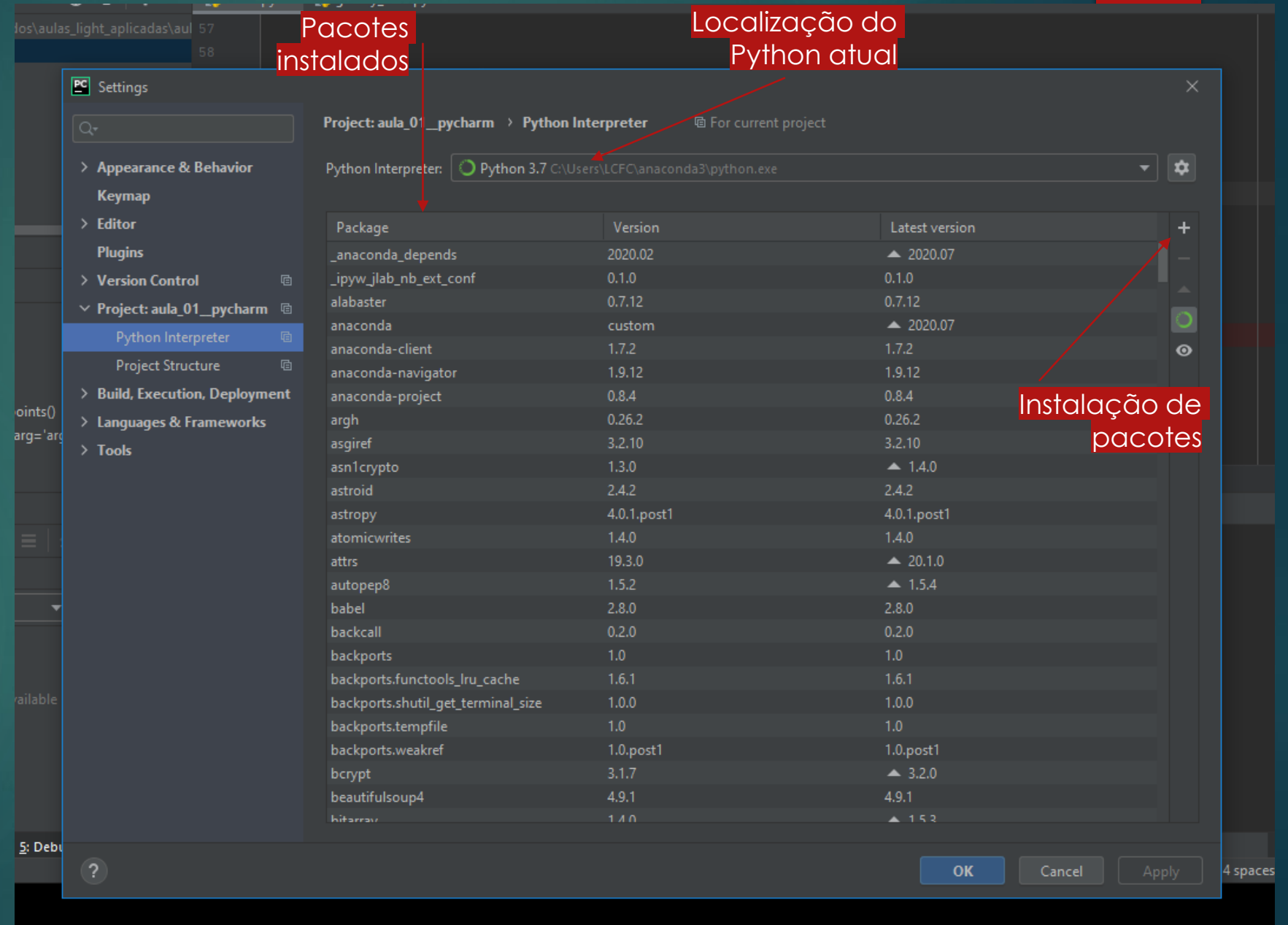
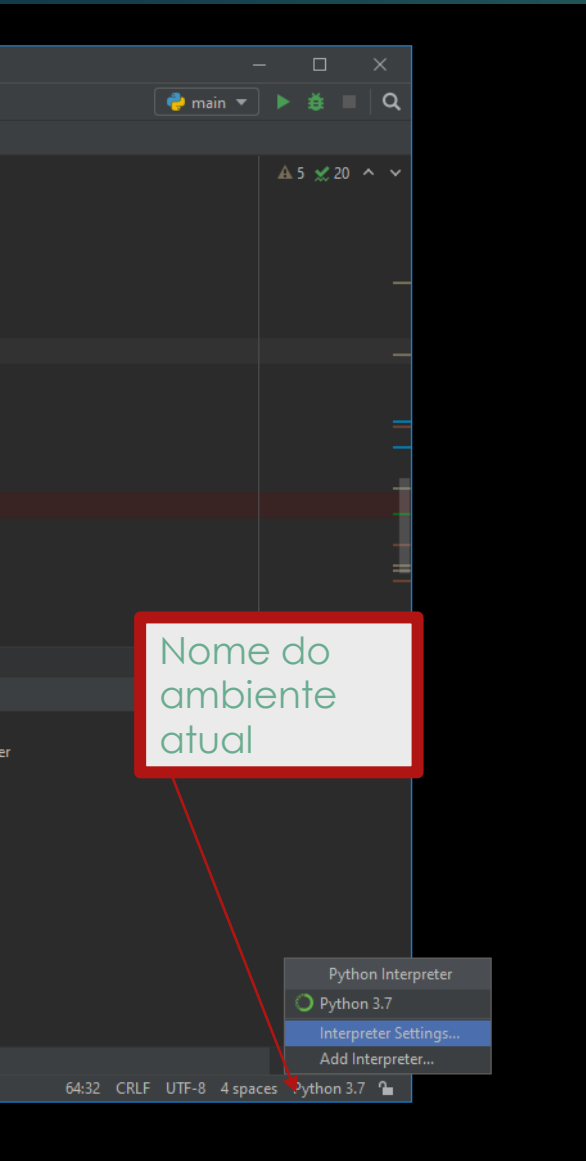
8. Por que escolhi o PyCharm?

Motivo 4:

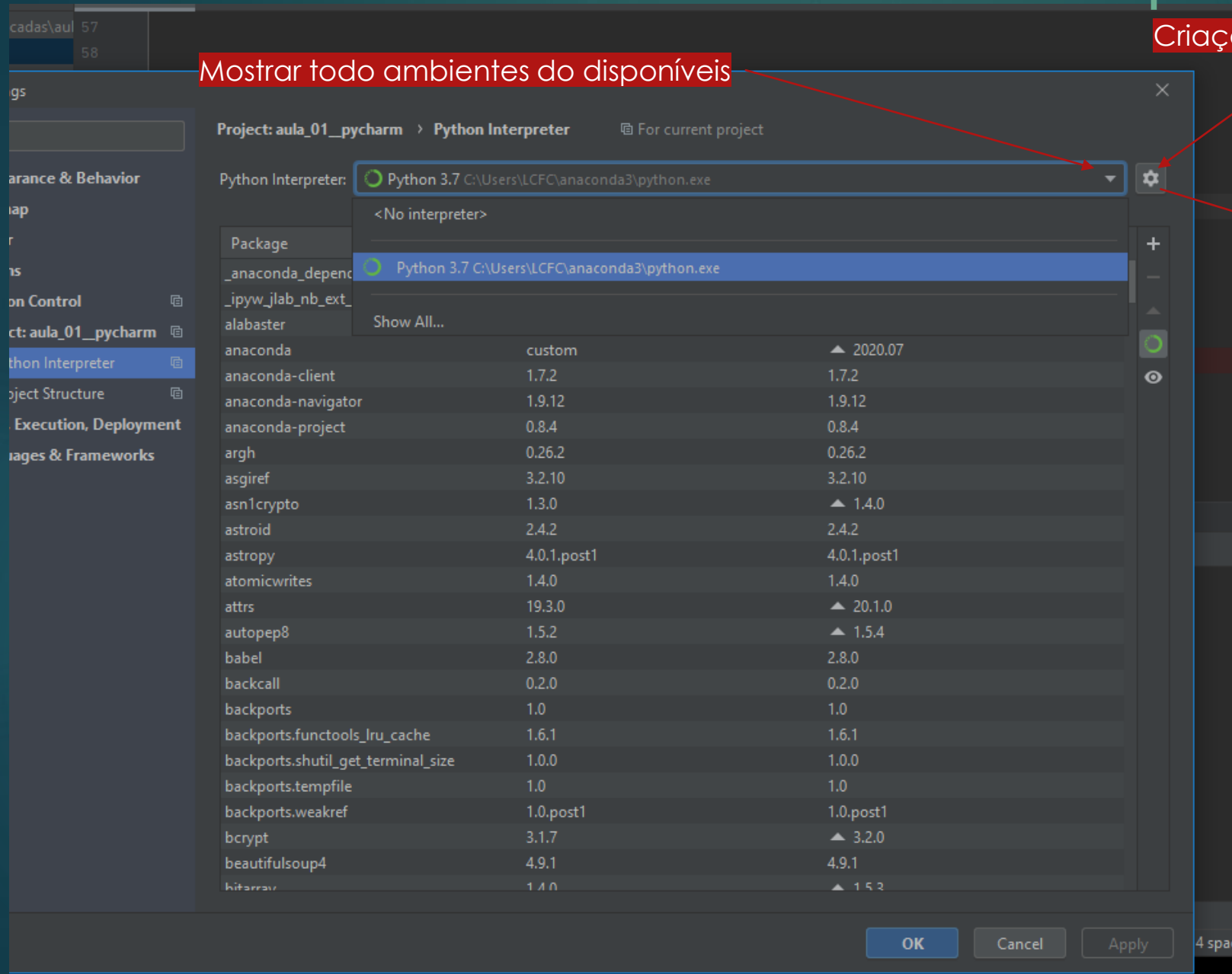
BUSCA:

Verificar arquivos , locais e contextos que uma função/classe é usada

7. Ambientes: pacotes Python

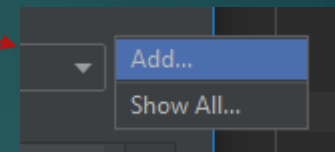


9. Ambientes: mostrar disponíveis



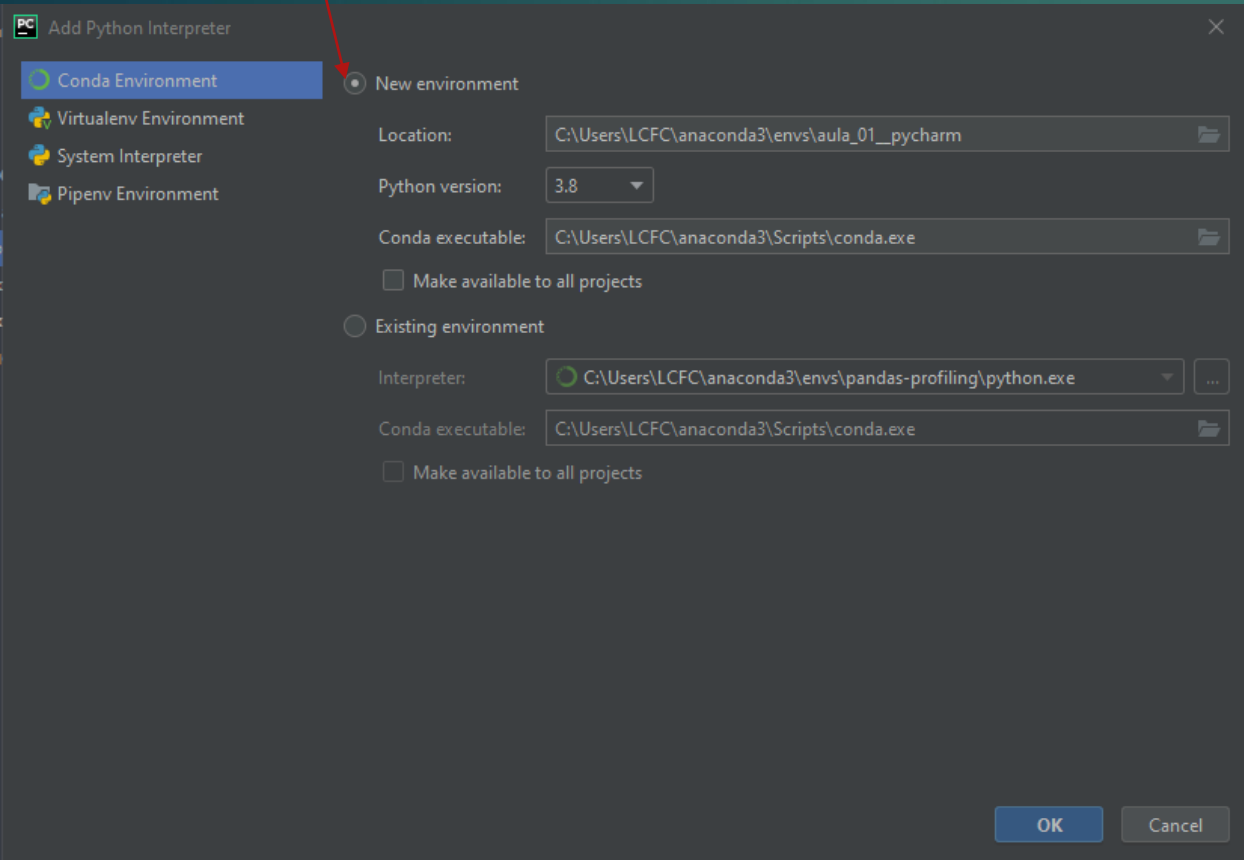
Criação de outro ambiente

Mostrar todos os ambientes disponíveis

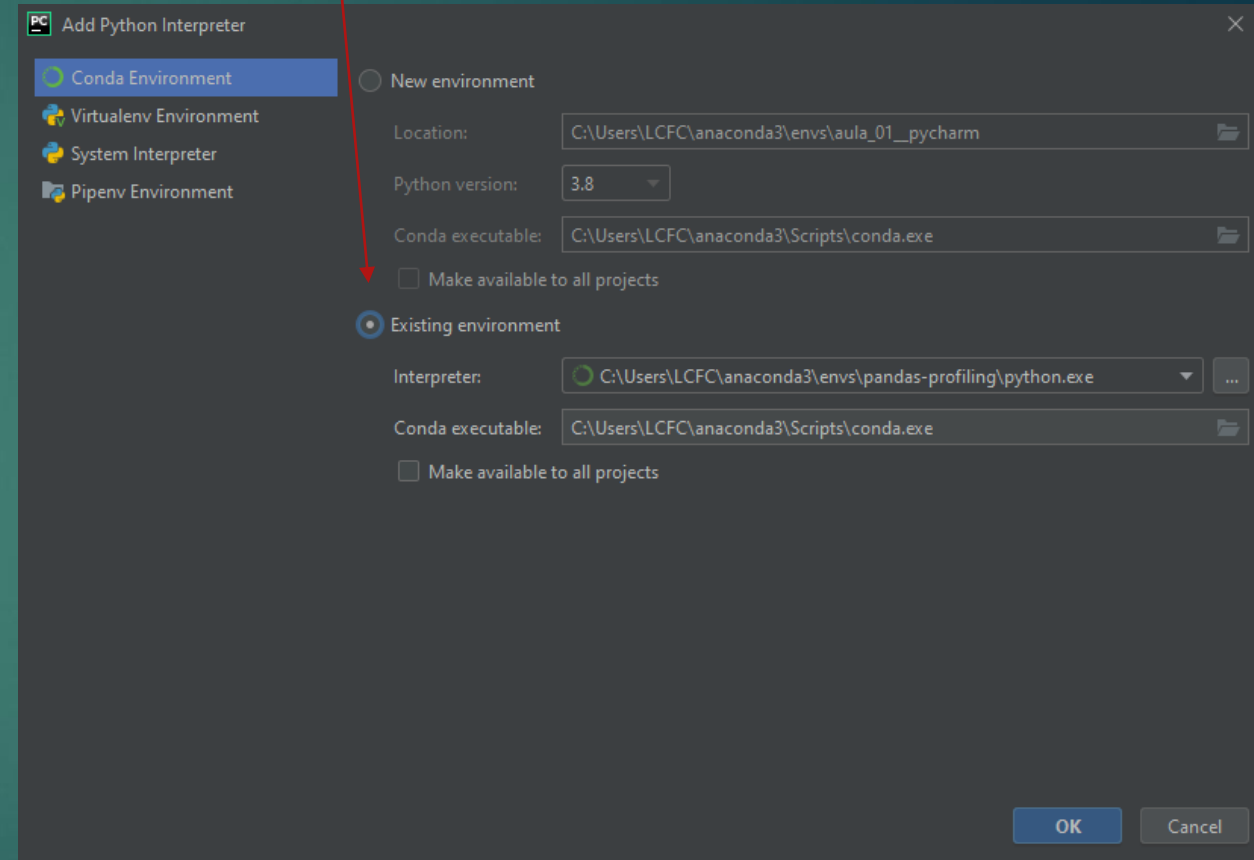


9. Ambientes: criação (conda, venv)

Criar novo ambiente



Procurar e trocar para outro ambiente já existente



10. Git: criação do repositório local

The screenshot shows the PyCharm IDE interface with several annotations explaining the steps to create a local Git repository:

- 1. Repositório git criado através do Terminal**: Points to the terminal output showing the successful execution of `git init` and `git flow init`.
- 2. Branch atual**: Points to the "develop" branch name in the bottom status bar.
- 3. Nova aba**: Points to the "Commit" button in the "Commit Message" dialog.
- 4. Selecionar arquivos que serão adicionado**: Points to the "Unversioned Files" list in the "Default Changelist" panel.
- 5. Mensagem de commit**: Points to the "Commit Message" input field.
- 6. Mensagem de commit**: Points to the "Commit" button in the "Commit Message" dialog.

Additional annotations include:

- Obs.: Nomes em vermelho são os arquivos não adicionados ao repositório**: Points to the red filenames in the "Unversioned Files" list.

The terminal output shows the following commands and their results:

```
(base) D:\Estudos\aulas_light_aplicadas\aula_01_pycharm>git init
Initialized empty Git repository in D:/Estudos/aulas_light_aplicadas/aula_01_pycharm/.git/
(base) D:\Estudos\aulas_light_aplicadas\aula_01_pycharm>git flow init
No branches exist yet. Base branches must be created now.
Branch name for production releases: [master]
Branch name for "next release" development: [develop]
How to name your supporting branch prefixes?
```

10. Git: visualização do log

The screenshot shows the PyCharm IDE interface with the Git Log window open. The main editor displays the code for `main.py` and `greedy_caio.py`. The Git Log window shows a list of commits, including 'Iniciando o projeto' and 'Initial commit'. A red box highlights the 'Initial commit' entry, and a red arrow points to it with the text '2. Git log'. Another red box highlights the 'Initial commit' entry, and a red arrow points to it with the text '1. Visualização do histórico de commits'.

```
def show_pandas_dataframe():
    import pandas as pd
    import numpy as np

    df = pd.DataFrame()
    df['competencia'] = pd.date_range('2017/01/01', '2020/09/03', freq='MS')
    df['vidas'] = np.random.randint(10000, 10250)
    df['sinistros'] = np.random.random(df.shape[0]) * 400000
    df['sinistros_per_capta'] = df['sinistros'] / df['vidas']

    df_ = df.copy()
    df_['vidas'] = np.random.randint(100, 150)
    df_['sinistros'] = 0

    # Press the green button in the gutter to run the script.
    if __name__ == '__main__':
        print_hi('PyCharm')
    mostrar_debugger_jump()
```

Git Log window showing the commit history:

- Local branches: develop, master
- Commits: Iniciando o projeto, Initial commit
- Branch: All, User: All, Date: All, Paths: All
- Commit details: Select commit to view changes

Bottom status bar: 8 files committed: Iniciando o projeto (4 minutes ago)

10. Git: visualização da alteração do arquivo

The screenshot displays the PyCharm IDE interface with several annotations indicating Git workflow steps:

- 1. Alteração no arquivo**: Points to a code change in `main.py` at line 71, where `df['sinistros'] = 0` is added.
- 2. Commit**: Points to the `Commit` button in the bottom-left toolbar.
- 3. Alteração do git log**: Points to the `Criacao da nova_funcao` entry in the `Git Log` panel.
- 4. Visualização da alteração**: Points to the `Show Diff` context menu option for the selected log entry.
- 5. pull**: Points to the `pull` button in the top-right toolbar.
- 6. push**: Points to the `push` button in the top-right toolbar.

The code editor shows the following Python code in `main.py`:

```
69 df_ = df.copy()
70 df_['vidas'] = np.random.randint(100, 150)
71 df_['sinistros'] = 0
72
73
74 def nova_funcao_apos_comitar():
75     pass
76
77
78 # Press the green button in the gutter to run the script.
79 if __name__ == '__main__':
80     print_hi('PyCharm')
81     mostrar_debugger_jump()
82     mostrar_steps()
83     mostrar_conditional_breakpoints()
84     refatoramento_de_funcoes(arg='python')
85     show_pandas_dataframe()
86
87 # See PyCharm help at https://www.jetbrains.com/help/pycharm/
88
```

The `Git Log` panel shows the following commit history:

- `Criacao da nova_funcao` (develop, Luiz Favaro, Today 13:26)
- `Iniciando o projeto` (Luiz Favaro, Today 13:18)
- `Initial commit` (master, Luiz Favaro, Today 13:05)

The `Show Diff` context menu is open, showing options like `Show Diff` (Ctrl+D), `Compare with Local`, `Compare Before with Local`, `Edit Source` (F4), `Open Repository Version`, `Revert Selected Changes`, `Cherry-Pick Selected Changes`, `History Up to Here`, and `Show Changes to Parents`.

10. Git: visualização da alteração do arquivo

The image shows the PyCharm interface with a diff view of a file named `main.py`. The top bar indicates the file path: `D:\Estudos\aulas_light_aplicadas\aula_01_pycharm`. The diff view compares two versions of the file, identified by their hashes: `151f77e2f06d97ad6e6fe550b33fa123647bb3d7` (left) and `fc32cf97a237abe6a7d7ac6660e9b3db52a98589` (right). The left pane shows the code before the commit, and the right pane shows the code after the commit. A third pane at the bottom visualizes the changes, with a green highlight on the new function `nova_funcao_apos_commitar()` and a red arrow pointing to it.

1. Arquivo antes do commit

```
df = pd.DataFrame()
df['competencia'] = pd.date_range('2017/01/01', '2020/09/03', freq='M')
df['vidas'] = np.random.randint(10000, 10250)
df['sinistros'] = np.random.random(df.shape[0]) * 400000
df['sinistros_per_capta'] = df['sinistros'] / df['vidas']

df_ = df.copy()
df_['vidas'] = np.random.randint(100, 150)
df_['sinistros'] = 0
```

2. Arquivo após o commit

```
df = pd.DataFrame()
df['competencia'] = pd.date_range('2017/01/01', '2020/09/03', freq='MS')
df['vidas'] = np.random.randint(10000, 10250)
df['sinistros'] = np.random.random(df.shape[0]) * 400000
df['sinistros_per_capta'] = df['sinistros'] / df['vidas']

df_ = df.copy()
df_['vidas'] = np.random.randint(100, 150)
df_['sinistros'] = 0
```

3. Visualização da alteração

```
def nova_funcao_apos_commitar():
    pass
```

Press the green button in the gutter to run the script.

```
if __name__ == '__main__':
    print_hi('PyCharm')
    mostrar_debugger_jump()
    mostrar_steps()
    mostrar_conditional_breakpoints()
    refatoramento_de_funcoes(arg='python')
    show_pandas_dataframe()

# See PyCharm help at https://www.jetbrains.com/help/pycharm/
```

11. Produtividade: Lista de atalhos de teclado

Mostrar automaticamente informações sobre os argumentos da função

PyCharm Default Keymap



Editing

Ctrl + Space	Basic code completion (the name of any class, method or variable)
Ctrl + Alt + Space	Class name completion (the name of any project class independently of current imports)
Ctrl + Shift + Enter	Complete statement
Ctrl + P	Parameter info (within method call arguments)
Ctrl + Q	Quick documentation lookup
Shift + F1	External Doc
Ctrl + mouse over code	Brief Info
Ctrl + F1	Show descriptions of error or warning at caret
Alt + Insert	Generate code...
Ctrl + O	Override methods
Ctrl + Alt + T	Surround with...
Ctrl + /	Comment/uncomment with line comment
Ctrl + Shift + /	Comment/uncomment with block comment
Ctrl + W	Select successively increasing code blocks
Ctrl + Shift + W	Decrease current selection to previous state
Ctrl + Shift + J/L	Select till code block end/start
Alt + Enter	Show intention actions and quick-fixes
Ctrl + Alt + L	Reformat code
Ctrl + Alt + O	Optimize imports
Ctrl + Alt + I	Auto-indent line(s)
Tab / Shift + Tab	Indent/unindent selected lines
Ctrl + X or Shift + Delete	Cut current line or selected block to clipboard
Ctrl + C or Ctrl + Insert	Copy current line or selected block to clipboard
Ctrl + V or Shift + Insert	Paste from clipboard
Ctrl + Shift + V	Paste from recent buffers...
Ctrl + D	Duplicate current line or selected block
Ctrl + Y	Delete line at caret
Ctrl + Shift + J	Smart line join
Ctrl + Enter	Smart line split
Shift + Enter	Start new line
Ctrl + Shift + U	Toggle case for word at caret or selected block
Ctrl + Delete	Delete to word end
Ctrl + Backspace	Delete to word start

PyCharm Default Keymap



Running

Alt + Shift + F10	Select configuration and run
Alt + Shift + F9	Select configuration and debug
Shift + F10	Run
Shift + F9	Debug
Ctrl + Shift + F10	Run context configuration from editor
Ctrl+Alt+R	Run manage.py task

Debugging

F8	Step over
F7	Step into
Shift + F8	Step out
Alt + F9	Run to cursor
Alt + F8	Evaluate expression
Ctrl + Alt + F8	Quick evaluate expression
F9	Resume program
Ctrl + F8	Toggle breakpoint
Ctrl + Shift + F8	View breakpoints

Navigation

Ctrl + N	Go to class
Ctrl + Shift + N	Go to file
Ctrl + Alt + Shift + N	Go to symbol
Alt + Right/Left	Go to next/previous editor tab
F12	Go back to previous tool window
Esc	Go to editor (from tool window)
Shift + Esc	Hide active or last active window
Ctrl + Shift + F4	Close active run/messages/find/... tab
Ctrl + G	Go to line
Ctrl + E	Recent files popup
Ctrl + Alt + Left/Right	Navigate back/forward
Ctrl + Shift + Backspace	Navigate to last edit location
Alt + F1	Select current file or symbol in any view
Ctrl + B or Ctrl + Click	Go to declaration
Ctrl + Alt + B	Go to implementation(s)
Ctrl + Shift + I	Open quick definition lookup

12. Outras Features

[HTTPS://WWW.JETBRAINS.COM/PYCHARM/FEATURES/](https://www.jetbrains.com/pycharm/features/)