



# Python

## Lógica de Programação

Ferramentas PyCharm

### Conteúdo:

- Introdução
- Criando seu projeto
- Ambiente PyCharm
- Palavra Bug
- Breakpoints
- Benefício Debugger
- Como usar
- Principais funções
- Atalhos
- Desafio

## 01 Introdução

- Neste *cheat sheet*, apresentaremos a ferramenta PyCharm, usada para desenvolvimento com Python, provida pela JetBrains
  - <https://www.jetbrains.com/pt-br/pycharm/download/>
- As ferramentas de desenvolvimento, chamamos de *Integrated Development Environment (IDE)*.
- O PyCharm tem 2 versões: uma chamada *Professional*, que é paga e uma intitulada *Community*, que é gratuita e a utilizada neste curso.

### Baixar PyCharm

Windows macOS Linux

#### Professional

Para desenvolvimento Web com Python e desenvolvimento científico. Com suporte para HTML, JS e SQL.

Baixar

Avaliação gratuita por 30 dias disponível

#### Community

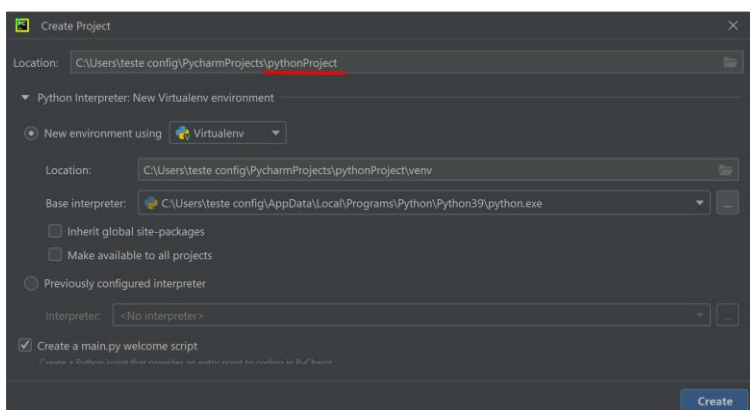
Para o autêntico desenvolvimento Python

Baixar

Gratuito, com base em open source

## 02 Criando o seu Projeto

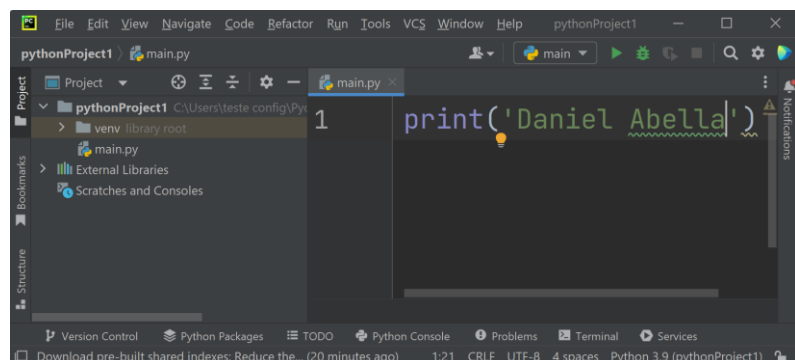
- Pycharm solicita que, sempre que for codificar algo, seja criado um projeto. Ao iniciar o Pycharm pela primeira vez, a janela a seguir é apresentada. Em **vermelho** deve ser inserido o nome do projeto.



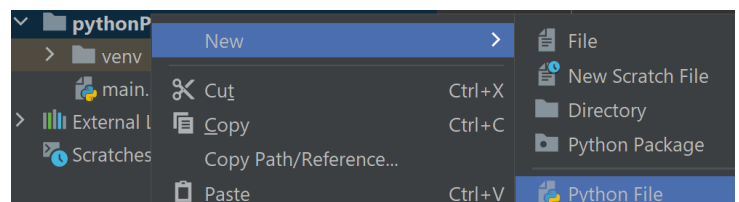
- Caso você já tenha criado algum projeto antes, o último projeto aberto é carregado. Neste caso, para criar um novo projeto, temos que ir em **File** e depois em **New Project...**

## 03 Ambiente PyCharm

- Uma vez o projeto foi criado, a tela a seguir é apresentada. No lado esquerdo, temos o nome do projeto **pythonProject1** e temos o arquivo onde vamos iniciar o código **main.py**.
- Caso você clique em **main.py** no lado esquerdo, o arquivo é aberto no lado direito, inclusive eu coloquei um `print('Daniel Abella')`.




- Para executar o projeto, clicamos no ícone ou **Shift + F10**.
- Para interromper a execução do projeto, clicamos no ícone
  - Ou **Ctrl + F12**
- Para criar novos arquivos (módulos), clique com o botão direito no projeto, depois em **New** e logo depois em **Python File**, conforme imagem a seguir.

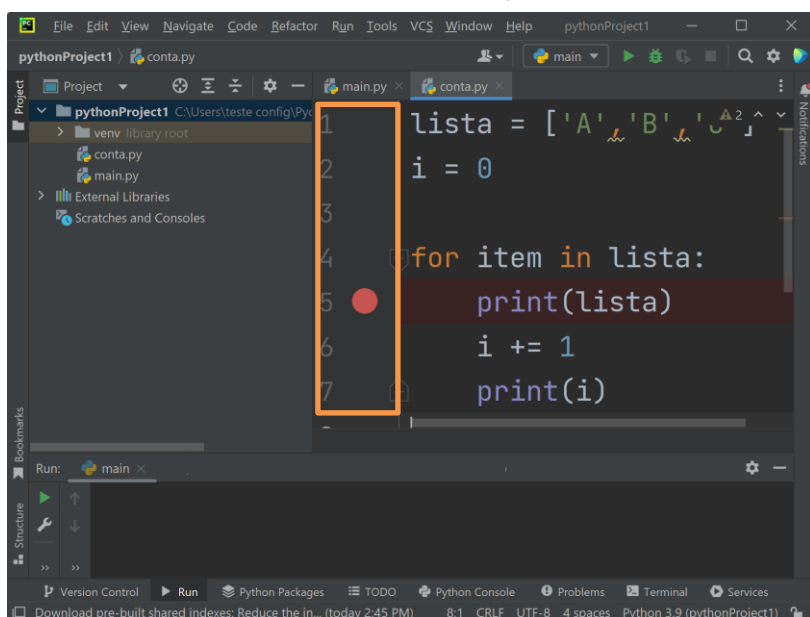


## 04 Origem da Palavra Bug

- Bug é uma palavra em inglês que significa inseto e foi usada para definir problemas por Grace Hopper, programadora da marinha dos EUA em 1947. Ela usou este termo (Bug) porque um inseto ficou preso nos contatos de um relê, causando mau funcionamento do computador Mark II na época. E até hoje o termo é usado.

## 05 Debugger e Breakpoints


- **Debugger** (Depurador, em Português) é a ferramenta ideal para você entender como o seu código está funcionando e identificar possíveis bugs.
- Como funciona? Basicamente você vai definir um ou mais pontos onde você gostaria que a execução desse uma pausa e a partir daí, você acompanha.
  - Ou seja, o código roda normalmente e assim que chegar nesse ponto, ele pausa imediatamente a execução e você controla a execução, isto é, vendo o que acontece linha a linha
  - E, durante isso, você fica sabendo os valores das variáveis
  - Esses pontos de interrupção, chamamos de **breakpoints**
- Para criar um **breakpoint**, clique na linha desejada na área em **laranja** indicada na imagem a seguir.
  - Uma vez selecionada a linha, clique de modo que o ícone  seja apresentado. Na imagem a seguir, colocamos o breakpoint na linha 5, isto é, quando a linha 5 for executada, a execução é pausada.

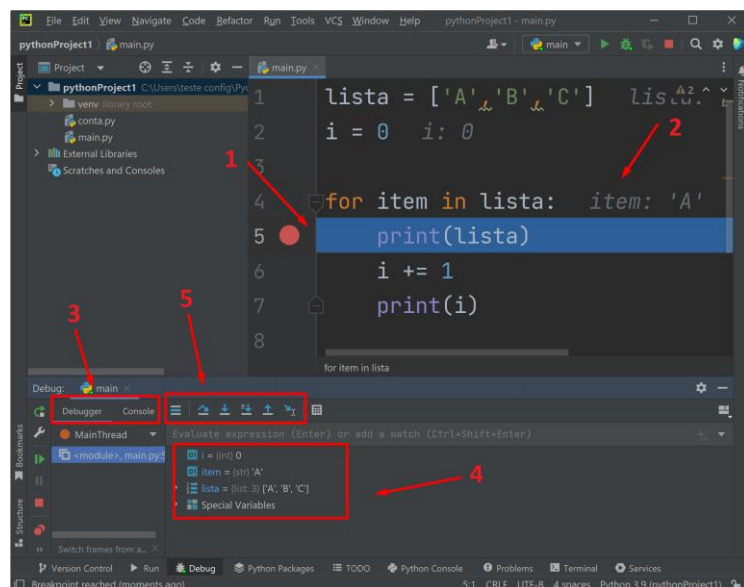


## 06 Benefícios do Debugger

- Um dos principais benefícios, como dito anteriormente, é a possibilidade de definir **breakpoints** e acompanhar a execução a partir deste ponto
- Entretanto, outros benefícios podem ser listados:
  - Verificar os valores que foram atribuídos as variáveis
  - Acompanhar a execução de métodos
  - E uma coisa muito legal que é, alterar o valor de variáveis enquanto ainda estiver executando.
    - Ou seja, permite alterar o valor em memória de uma variável

## 07 Como depurar?

- Uma vez que você colocou os **breakpoints**, deve clicar no ícone  para iniciar a execução em modo **debug**.
  - Alternativamente, você pode usar **⇧ Shift + F9**
  - Só assim os **breakpoints** serão considerados.



- Na imagem anterior, iniciamos o debug. Destaco 5 pontos (indicados por setas):
  - **Seta Número 1:** A linha 5, indica que, a execução foi interrompida devido ao breakpoint da linha 5
  - **Seta Número 2:** Para ajudar no entendimento do código, o PyCharm coloca em uma fonte com cor bem clara o valor variável. Neste exemplo, a variável `item` está com o valor 'A'
  - **Seta Número 3:** Nesta seção, indicamos o que gostaríamos de visualizar no PyCharm, que pode ser o **Debugger** (que permite a visualização do valor das variáveis, indicado na Seta 4) ou **Console**, que apresenta a saída do programa (geralmente, oriunda de prints)
  - **Seta Número 4:** Se na área indicada na seta de número 3, você selecionou **Debugger**, você vai poder visualizar os valores das variáveis do sistema, como elas estão na memória
  - **Seta Número 5:** Principais funções do depurador para poder acompanhar o código (descritas na seção a seguir).

## 08 Principais Funções para Acompanhar o Código

- Ao iniciar o Debug, na parte inferior, destacada na Seta 5 da página anterior, temos alguns ícones para as funções de acompanhar o código com Debug.

- Na imagem a seguir, apresentamos as principais funções que vocês precisam conhecer.



- Iniciar o Debug**
  - Opção 1: Clicar em
  - Opção 2: +
- Step Over (Executa a Linha de Código onde o Cursor Está)**
  - Opção 1: Clicar em
  - Opção 2:
- Step Into (Entra no Método e Chama o Método)**
  - Opção 1: Clicar em
  - Opção 2:
- Step Out (Termina o Debug da Função e o Cursor Volta ao Local Anterior)**
  - Opção 1: Clicar em
  - Opção 2: +

## 09 Principais Atalhos do PyCharm

- Abaixo relacionamos os principais atalhos do PyCharm. Entretanto, nos links a seguir você pode verificar mais atalhos:
  - Verificar todos os disponíveis no próprio PyCharm
    - Acessar o menu depois
    - Clicamos em na janela apresentada
  - <https://www.shortcutfoo.com/app/dojos/pycharm-win/cheatsheet>

Atalho	Função
+	Completa o código
+  +	Comenta ou descomenta com '''
+	Comenta ou descomenta com #
+	Exclui uma linha
+	Seleciona todas (All) linhas do arquivo
+  +  e  +  +	Pula de um método pra outro
+	Mostra os arquivos abertos recentemente
+	ctrl + C junto com Ctrl + V do código selecionado

Atalho	Função
+	Busca (find)
+  +  e  +  +	Move as linhas selecionadas
+	Importa os pacotes ausentes
+  +	Otimiza seus pacotes importados, remove os não utilizados
+	Formata (Identa) uma linha de código

## CHALLENGE

### 10 Desafio

- Acompanhe o código abaixo usando o Debug do PyCharm.

```

1  num_list = [500, 600, 700]
2  alpha_list = ['x', 'y', 'z']
3
4
5  def nested_loop():
6      for number in num_list:
7          print(number)
8          for letter in alpha_list:
9              print(letter)
10
11  nested_loop()
  
```

### 11 Quer Saber Mais?

- <http://excript.com/python/depuracao-pycharm-python.html>
- <https://medium.com/analytics-vidhya/useful-pycharm-shortcuts-65343a72e6f2>
- <https://www.shortcutfoo.com/app/dojos/pycharm-win/cheatsheet>
- <https://github.com/nareddyt/cs3600-pycharm-debugging/blob/master/pycharm-setup-and-debugging.md>
- <https://realpython.com/pycharm-guide/>