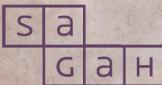


# RACIOCÍNIO ALGORÍTMICO

Juliano Vieira Martins



SOLUÇÕES  
EDUCACIONAIS  
INTEGRADAS

# Ambiente integrado de desenvolvimento em Python

## Objetivos de aprendizagem

Ao final deste texto, você deve apresentar os seguintes aprendizados:

- Identificar as principais funcionalidades de um IDE.
- Construir algoritmos utilizando um IDE.
- Usar funcionalidades de suporte ao desenvolvimento de *software*.

## Introdução

Um ambiente de desenvolvimento integrado (do inglês *Integrated Development Environment* — IDE) é uma plataforma de *software* que fornece aos programadores um conjunto abrangente de ferramentas, todas reunidas em um único produto, para o desenvolvimento de *softwares*. Esses ambientes foram desenvolvidos para trabalhar com plataformas de aplicativos específicas e ajudar na remoção das barreiras envolvidas no ciclo de vida do desenvolvimento de um *software*. Além disso, IDEs são usados pelas equipes de desenvolvimento para criar novos *softwares*, aplicativos, páginas da Web, serviços, entre outras várias aplicações.

Um IDE ajuda no trabalho do desenvolvedor ao fornecer uma ferramenta com todos os recursos, pois remove a necessidade de o desenvolvedor lidar com a integração entre várias ferramentas distintas. Os IDEs destinam-se a programar código para uma plataforma ou linguagem de programação específica e possuem recursos integrados que sabem como a linguagem de programação funciona e como usar os recursos por meio da compilação de código, da depuração de código ou do autocomplemento de código.

Neste capítulo, você vai conhecer as principais funcionalidades de um IDE. Além disso, você vai ver como usar essas funcionalidades para construir algoritmos, com foco nas funcionalidades que dão suporte ao desenvolvimento de *softwares*.

## 1 IDE: o que é, benefícios e funcionalidades

Em um nível mais básico, os IDEs fornecem interfaces para os usuários escreverem código, organizarem grupos de texto e automatizarem redundâncias de programação. Mas, em vez de um editor de código básico, os IDEs combinam as funcionalidades de vários processos de programação em apenas um (ROUSE, 2018). Alguns IDEs concentram-se em uma linguagem de programação específica, como Python ou Java, mas muitos têm recursos de diferentes idiomas. Em termos de recursos de edição de texto, os IDEs geralmente possuem ou permitem a inserção de estruturas e bibliotecas de elementos com base no código.

Para que uma ferramenta se qualifique na categoria dos IDEs, o produto deve fornecer recursos de programação por meio de um editor de texto ou por uma interface gráfica do usuário (*Graphical User Interface — GUI*). Além disso, deve integrar-se a pelo menos uma plataforma ou linguagem de programação sem a necessidade de um *plug-in* separado. Por fim, a ferramenta deve disponibilizar a interface de programação de aplicativos (do inglês *Application Programming Interface — API*) e permitir a compilação, a depuração, o controle de versão, as sugestões de código específicas da linguagem e a implantação de código.

Durante o processo de desenvolvimento de *software*, um ou vários usuários podem criar hierarquias no IDE e atribuir grupos de códigos à sua área designada. A partir disso, as partes de código podem ser agrupadas, compiladas e, assim, o *software* é construído. A maioria dos IDEs vem com depuradores internos, que são ativados na compilação do código. Os depuradores visuais são um benefício substancial de muitos IDEs. Se for detectado algum defeito, falha ou erro no código do programa (*bug*) que pode provocar mau funcionamento, isso será mostrado aos desenvolvedores, apontando em quais partes do código os problemas devem ser resolvidos. Além dos que já foram mencionados, é possível elencar os seguintes benefícios de um IDE:

- serve como um ambiente único para a maioria (senão para todas) das necessidades de um desenvolvedor, como sistemas de controle de versão, ferramentas de depuração e plataforma como serviço;
- tem recursos de autocomplemento e conclusão de código que aprimoram o fluxo de trabalho de programação;
- verifica automaticamente os erros para garantir um código de alta qualidade;

- tem recursos de refatoração que permitem que os desenvolvedores façam alterações de renomeação abrangentes e sem erros;
- mantém um ciclo de desenvolvimento suave;
- aumenta a eficiência e a satisfação do desenvolvedor;
- auxilia na entrega de um *software* de alta qualidade dentro do prazo.

Há vários motivos para usar um IDE, a maioria dos quais gira em torno do desenvolvimento de *software*. A plataforma centraliza três ferramentas críticas usadas pela maioria dos desenvolvedores, que são: editores de código-fonte, depuradores e compiladores. Isso permite que os desenvolvedores de *software* escrevam, aperfeiçoem e processem código em um único ambiente. A centralização dessas ferramentas também facilita a navegação no código-fonte em questão. Além disso, muitos IDEs incluem recursos adicionais para testar, organizar e refatorar (organizar e limpar) o código. Outros recursos, como o autocomplemento, juntamente aos recursos de criação e implantação, expandem significativamente as habilidades de um desenvolvedor e melhoram sua velocidade de desenvolvimento.

O gerenciamento aprimorado da organização do código-fonte pode reduzir erros e também o tempo de desenvolvimento. Os usuários de um IDE podem ajustar aplicativos após a conclusão das compilações. Geralmente, eles podem salvar versões de um projeto, o que é útil caso precisem reverter um programa para uma versão anterior. Eventualmente, as equipes usam o sistema de controle de versão integrado para fazer *check-in* (enviar código para um repositório, por exemplo o GitHub) e *check-out* (baixar o código do repositório) de componentes do programa no repositório de códigos. Por fim, depois que todos os programas foram ajustados e o aplicativo executou as funções desejadas, o projeto pode ser empacotado e implantado a partir do IDE, por meio de uma ferramenta integrada.

Além dos IDEs destinados ao uso local, existem também IDEs desenvolvidos para trabalhar na computação em nuvem. Os IDEs fornecidos na modalidade de *software* como serviço (do inglês *Software as a Service* — SaaS), que são baseados em nuvem, oferecem vários benefícios exclusivos em comparação com os ambientes de desenvolvimento local. Por um lado, como em qualquer oferta SaaS, não há necessidade de baixar um *software* e configurar ambientes e dependências locais para que os desenvolvedores começem a contribuir rapidamente para os projetos. Isso também fornece um nível de padronização nos ambientes de cada membro da equipe, o que pode atenuar o problema comum de que determinado aplicativo funciona em uma máquina e não em outra. Além disso, como o ambiente de desenvolvimento

é gerenciado centralmente, nenhum código reside no computador de um desenvolvedor individual, o que pode ajudar com questões de propriedade intelectual e segurança.

O impacto dos processos nas máquinas locais também é diferente. Processos como a execução de um conjunto de compilação e testes geralmente exigem muitos recursos (poder) de computação. Isso significa que os desenvolvedores provavelmente não podem continuar usando as estações de trabalho enquanto um processo está em execução. Um IDE SaaS pode despachar trabalhos de longa execução sem ter de monopolizar os recursos de computação de uma máquina local. Os IDEs implementados em nuvem também são tipicamente independentes de plataforma, permitindo conexões com diferentes fornecedores desse tipo de serviço.

Veja a seguir as ferramentas integradas de um IDE:

- refatoração de código;
- integração e implantação;
- autocomplemento de um código;
- editor de código;
- *plug-ins* e extensões;
- realce de sintaxe;
- teste de código;
- versionamento de código;
- compilação de código;
- depuração de código.

## Recursos e funcionalidades de IDEs para Python

Existem muitos tipos de IDEs para a linguagem de programação Python. Há tanto aqueles desenvolvidos com código-fonte aberto, em que vários desenvolvedores podem contribuir, ajudando a aprimorar a plataforma, quanto aqueles de uso comercial, desenvolvidos e mantidos por empresas. Isso significa que existem muitas opções de IDE de código aberto e proprietárias no mercado. No entanto, todos eles possuem características comuns e mínimas para realmente serem IDEs para Python.

## Linguagens de programação suportadas

Alguns IDEs são dedicados especificamente à linguagem de programação Python. Portanto, existe uma correspondência melhor para o paradigma de programação específico para esse ecossistema. O IDE PyCharm, por exem-

pto, desenvolvido pela empresa JetBrains, é conhecido principalmente como um IDE para a linguagem de programação Python. Outros IDEs suportam uma ampla variedade de linguagens; é o caso do IDE Eclipse, que suporta as linguagens de programação Java e XML, entre outras.

## Recursos de automação

A maioria dos IDEs incluem os três principais recursos, que são o editor de texto, a automação de compilação e o depurador de código. Contudo, IDEs para Python também podem incluir suporte a recursos adicionais para a linguagem, como refatoração, pesquisa de código e ferramentas de integração (*Continuous Integration* — CI) e implantação contínua (*Continuous Deployment* — CD).

## Impacto no desempenho do sistema

Pode ser importante considerar a presença de memória utilizada por um IDE se um desenvolvedor deseja executar outros aplicativos que consomem muita memória simultaneamente. Para Python, não é diferente, pois o próprio interpretador Python, quanto ativo no sistema, já consome determinado recurso de memória e processamento.

## Plug-ins e extensões

Alguns IDEs incluem a capacidade de personalizar fluxos de trabalho para corresponder às necessidades e preferências de um desenvolvedor. Para isso, as plataformas disponibilizam formas de adicionar *plug-ins* que anexam outras funcionalidades a elas, além daquelas que vêm por padrão juntamente com o IDE. Para Python, os *plug-ins* podem ser tão simples quanto temas e conjunto de cores de aparência do IDE ou tão complicados quanto extensões de implantação contínua, ferramentas de integração e desenvolvimento de banco de dados, por exemplo.

## Editor de texto

Praticamente todo IDE para Python terá um editor de texto projetado para escrever e manipular o código-fonte. Algumas ferramentas podem ter componentes visuais para arrastar e soltar componentes de *front-end* (interface frontal), mas a maioria possui uma interface simples com realce de sintaxe de código específico para a linguagem que a plataforma suporta.

## Depurador de código

As ferramentas de depuração embutidas no IDE para Python ajudam os usuários a identificar e corrigir erros no código-fonte. Essas ferramentas geralmente simulam cenários do mundo real para testar a funcionalidade e o desempenho de aplicativos. Programadores e engenheiros de *software* podem testar os vários segmentos de código e identificar erros antes mesmo do lançamento do *software*.

## Compilador de código

Compiladores são componentes que convertem o código Python em um formato que as máquinas entendem e podem processar, como o código binário. O código da máquina é analisado para garantir sua precisão. O compilador analisa e otimiza o código com o objetivo de melhorar o desempenho.

## Conclusão ou autocomplemento de código

Os recursos de autocompletar código ajudam os programadores a identificar e inserir de forma inteligente componentes de código padrão. Esses recursos economizam tempo para os desenvolvedores Python, auxiliando-os a escrever códigos com menor probabilidade de erros de digitação.



### Fique atento

O Sistema Operacional (SO) sob o qual o desenvolvedor Python trabalha restringe os IDEs viáveis, a menos que um IDE seja baseado em nuvem. Além disso, se o aplicativo em desenvolvimento for destinado a um usuário final e a um SO específico (por exemplo, o Windows, da empresa Microsoft, ou o macOS, da empresa Apple), isso pode ser uma restrição adicional. Da mesma forma, se o novo aplicativo for destinado ao SO iOS, da Apple, Android, do Google, e também a uma página da Web, talvez seja melhor começar com um IDE que forneça suporte de plataforma cruzada para vários sistemas operacionais.

No mercado, existem muitas opções diferentes de IDEs para Python. A maioria dessas ferramentas fornece uma variedade de recursos, como escrever, compilar e até depurar códigos, tudo em um só lugar. Obviamente,

existem dezenas de fatores que você deve considerar ao escolher o melhor ou o mais adequado IDE para Python. A seguir, veja alguns IDEs de diferentes desenvolvedores, bem como suas funcionalidades básicas.

## Idle

O editor Python Idle (*Integrated Development and Learning Environment*) é uma ferramenta de interface gráfica do usuário para o desenvolvimento do Python (PYTHON SOFTWARE FOUNDATION, 2020). Esse IDE está listado em primeiro lugar pois, além de ser gratuito, é também instalado automaticamente durante a instalação da linguagem de programação Python. Ele permite editar, executar e depurar programas Python em um ambiente simples. O Idle é na verdade um programa Python que usa o *kit* de ferramentas de interface gráfica da biblioteca padrão para construir suas janelas. Ele é portátil e pode ser executado em todas as principais plataformas, como Windows, Linux e macOS.

## PyCharm Professional

É um IDE específico para Python desenvolvido pela JetBrains, criadora também dos IDEs IntelliJ IDEA para Java, WebStorm para Javascript e PhpStorm para PHP, além de outros. É um *software* proprietário que oferece recursos de ponta, como edição inteligente de código e navegação inteligente de código. O PyCharm fornece ferramentas de desenvolvimento prontas para o uso para depuração, teste, implantação e acesso ao banco de dados. Ele está disponível para Windows, macOS e Linux e pode ser expandido por meio de dezenas de *plug-ins* e integrações.

## Komodo IDE

É um IDE multilíngue desenvolvido pela empresa Active State. Além de Python, esse IDE suporta as linguagens PHP, Perl, Go, Ruby, além de oferecer desenvolvimento Web com suporte para HTML, CSS, JavaScript, etc. O IDE vem equipado com inteligência de código para facilitar o preenchimento automático e a refatoração. Ele também fornece ferramentas para depuração e teste. A plataforma suporta múltiplos formatos de controle de versão, incluindo Git, Mercurial e Subversion. As equipes podem utilizar recursos de programação colaborativa e definir fluxos de trabalho para navegação de arquivos e projetos. A funcionalidade também pode ser expandida por meio de uma ampla variedade de *plug-ins*, que possibilitam personalizar a experiência do usuário e estender a funcionalidade do recurso.

## Codenvy

É um IDE baseado na ferramenta de código aberto Eclipse Che e é desenvolvido e mantido pela gigante de *software* Red Hat. A ferramenta combina os recursos de um IDE com os recursos de gerenciamento de configuração em um ambiente baseado em navegador Web. Os espaços de trabalho são contêineres, protegendo-os contra ameaças externas. Os recursos do desenvolvedor incluem o Che IDE totalmente funcional, o autocomplemento de código, a verificação de erros e um depurador.

## KDevelop

É um IDE gratuito e de código aberto capaz de funcionar em vários sistemas operacionais. Além de Python, suporta programação em C, C++, QML, JavaScript e PHP. Além disso, o IDE suporta a integração de controle de versão do Git, Bazaar e Subversion, entre outros. Os recursos padrão incluem navegação rápida por código, destaque inteligente e conclusão da semântica de código. A interface do usuário é altamente personalizável e a plataforma suporta vários *plug-ins*, integrações de teste e documentação.

## Anjuta

É um estúdio de desenvolvimento de *software* e IDE que suporta programação em C, C++, Java, JavaScript, Python e Vala. Possui uma interface de usuário flexível e um sistema de encaixe que permite aos usuários personalizar vários componentes da interface. O produto vem equipado com recursos IDE padrão para edição de origem, controle de versão e depuração. Ele também possui recursos para oferecer suporte ao gerenciamento de projetos e de arquivos. Além disso, vem com uma ampla variedade de opções de *plug-ins* para extensibilidade.

## Wing Python IDE

Foi projetado explicitamente para o desenvolvimento do Python. O produto vem em três edições: a edição 101, a edição Personal e a edição Pro. A 101 é uma versão simplificada com um depurador minimalista, além de recursos de edição e pesquisa. A edição Personal avança para um editor completo,

além de oferecer recursos limitados de inspeção de código e gerenciamento de projetos. O Wing Pro oferece juntamente todos esses recursos, além de desenvolvimento remoto, teste de unidade, refatoração de código, suporte a estruturas, bem como outras funcionalidades.



### Fique atento

Os IDEs apresentados são produtos de *software*, de código aberto ou proprietários. Assim, o desenvolvimento de cada um deles acontece de forma dinâmica, e a qualquer momento podem ter alguma funcionalidade descontinuada ou mesmo ser extintos por completo pela equipe ou empresa que os mantêm. Dessa forma, cabe a cada programador testar os IDEs existentes e ativos no mercado. Por fim, o programador pode utilizar os IDEs que forem mais adequados às necessidades de desenvolvimento de *software* de cada caso.

## 2 Codificação de algoritmos com o Python Idle

Os IDEs profissionais são peças de *software* robustas e têm uma curva de aprendizado acentuada. Por isso, para quem está começando na jornada de programação Python, o Idle é uma ótima alternativa. Toda instalação Python vem com o ambiente integrado de desenvolvimento e aprendizado, que é abreviado, na sigla em inglês, como Idle (RAMALHO, 2012). Essa plataforma ajuda o programador a escrever códigos com mais eficiência do que ao utilizar um simples bloco de notas. Embora existam muitas opções de IDEs para se escolher no mercado, o Python Idle é muito simples, o que o torna a ferramenta perfeita para um programador iniciante. Por esse motivo, você vai ver aqui como usar essa ferramenta a nível básico para escrever algoritmos. Além disso, você vai ver como interagir diretamente com a linguagem de programação Python usando o Idle.

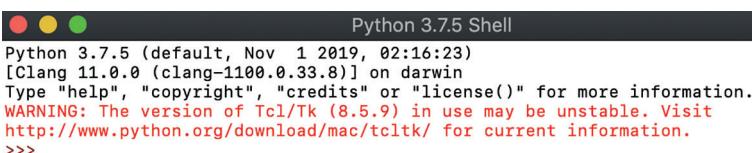
Como você já viu, o Python Idle vem por padrão incluído nas instalações do Python para os sistemas operacionais Windows e macOS. Para os usuários do SO Linux (Unix), a ferramenta pode ser encontrada pelo gerenciador de pacotes do Linux. Depois de instalado, é possível utilizar o Python Idle como intérprete “interativo” ou como editor de arquivos.

Existem dois modos de executar comandos no interpretador do Python: o modo normal e o modo interativo. O melhor modo para começar a experimentar, testar e aprender o código Python é o interpretador interativo, também conhecido como Shell. O Shell é na verdade um executor das tarefas “leia, avalie, imprima e repita” (do inglês *Read-Eval-Print-Loop* — REPL). O Shell lê uma instrução Python, avalia o resultado dessa instrução e depois imprime o resultado na tela do Shell. Em seguida, volta para ler a próxima instrução.

Quando se trabalha com Python, não é preciso importar uma biblioteca extra para ler e gravar arquivos. Essa ação é tratada de forma nativa na linguagem Python, embora de uma maneira única. O Python Idle permite criar e editar arquivos Python com facilidade por meio da interface gráfica da ferramenta. Para isso, basta navegar até a opção *File → New File*. Assim, uma nova janela se abrirá, pronta para receber código a ser adicionado. Ao fechar, a ferramenta solicita um nome e um local para gravar o arquivo.

O Shell Python do modo interativo é excelente para experimentar pequenos trechos de código. É possível acessar o ambiente Shell Python por meio do terminal ou de algum aplicativo de linha de comando que esteja instalado no computador. No entanto, é possível simplificar o fluxo de trabalho com o Python Idle, que iniciará um Shell Python imediatamente assim que a ferramenta for inicializada. Além disso, o Python Idle fornece vários recursos úteis vistos em IDEs profissionais, como destaque básico de sintaxe, conclusão de código e indentação automática.

Quando se inicializa a plataforma Idle, o Shell Python é a primeira coisa que se vê (Figura 1).



```
Python 3.7.5 (default, Nov  1 2019, 02:16:23)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
WARNING: The version of Tk/Tk (8.5.9) in use may be unstable. Visit
http://www.python.org/download/mac/tcltk/ for current information.
>>>
```

**Figura 1.** Modo interativo no ambiente Shell Python da ferramenta Idle.

Assim que é inicializada a ferramenta no modo interativo, é possível começar a interagir imediatamente com a linguagem Python. Por exemplo, pode-se testar a ferramenta com uma pequena linha de código (Figura 2).

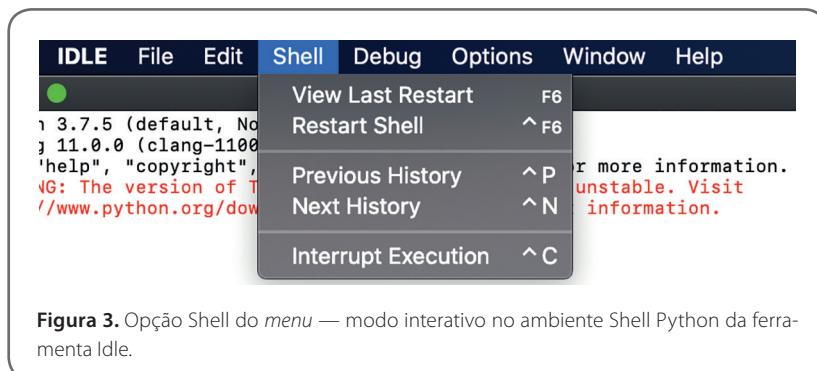


```
Python 3.7.5 (default, Nov  1 2019, 02:16:23)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
WARNING: The version of Tk (8.5.9) in use may be unstable. Visit
http://www.python.org/download/mac/tcltk/ for current information.
>>> print("Bem vindo ao Shell Python no modo interativo!")
Bem vindo ao Shell Python no modo interativo!
>>>
```

**Figura 2.** Modo interativo no ambiente Shell Python da ferramenta Idle com um pequeno trecho de código.

Na Figura 2, foi utilizada a função “*print()*” para gerar a *string* “Bem-vindo ao Shell Python no modo interativo!” para a tela da ferramenta. Essa é a maneira mais básica de interagir com o Python Idle. Dessa forma, o usuário digita comandos, um de cada vez, e o Python responde com o resultado de cada comando.

Em seguida, se o usuário der uma olhada na barra de *menus*, perceberá algumas opções para usar o Shell (Figura 3).



**Figura 3.** Opção Shell do menu — modo interativo no ambiente Shell Python da ferramenta Idle.

O usuário pode reiniciar o Shell no *menu* apresentado na Figura 3. Assim que é selecionada, essa opção limpa o estado atual do Shell. Ele funcionará como se tivesse iniciado uma nova instância do Python Idle. Dessa forma, o Shell esquecerá tudo de seu estado anterior, tais como códigos digitados e avaliações realizadas (Figura 4).



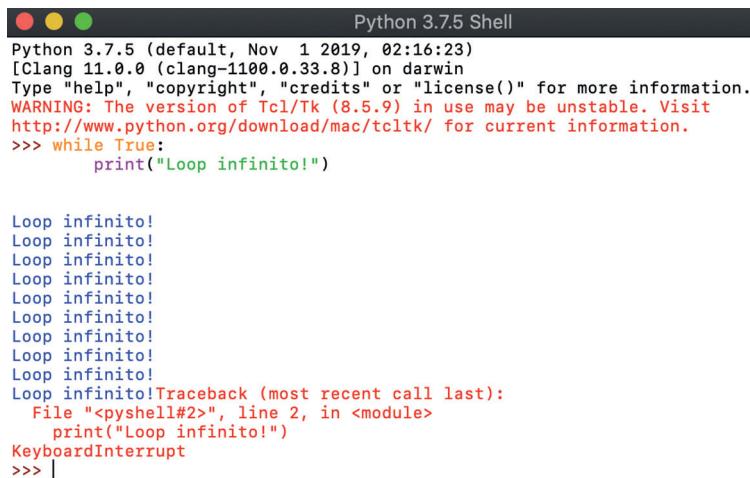
The screenshot shows the Python 3.7.5 Shell window. The title bar says "Python 3.7.5 Shell". The shell prompt is ">>>". The user has typed "x = 7" and "print(x)". The output shows "7" followed by a new line. Then, the user types "print(x)" again, which results in a "NameError: name 'x' is not defined" error. Below this, a message "RESTART: Shell" is displayed, indicating that the shell has been restarted.

```
Python 3.7.5 (default, Nov  1 2019, 02:16:23)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable. Visit
http://www.python.org/download/mac/tcltk/ for current information.
>>> x = 7
>>> print(x)
7
>>>
===== RESTART: Shell =====
>>> print(x)
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    print(x)
NameError: name 'x' is not defined
>>>
```

**Figura 4.** Modo interativo reinicializado no ambiente Shell Python da ferramenta Idle.

Na Figura 4, primeiramente foi declarada uma variável,  $x = 7$ . Assim, quando foi chamada a função “*print(x)*”, o Shell Python exibiu a saída correta, que é o número 7. No entanto, quando o Shell foi reiniciado e tentou-se chamar a função “*print(x)*” novamente, o Shell imprimiu uma mensagem de erro no retorno. Essa é uma mensagem de erro informando que a variável  $x$  não está definida. O Shell Python esqueceu tudo o que aconteceu antes de ser reiniciado.

Também é possível interromper a execução do Shell no mesmo *menu*, especificamente na opção “*Interrupt Execution*”. Isso interromperá qualquer programa ou instrução que esteja sendo executada no Shell no momento da interrupção. A seguir, na Figura 5, veja um exemplo do que acontece quando é enviado um comando de interrupção para o Shell.



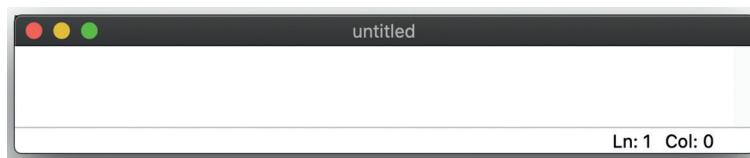
```
Python 3.7.5 (default, Nov  1 2019, 02:16:23)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable. Visit
http://www.python.org/download/mac/tcltk/ for current information.
>>> while True:
    print("Loop infinito!")

Loop infinito!
Loop infinito!Traceback (most recent call last):
  File "<pyshell#2>", line 2, in <module>
    print("Loop infinito!")
KeyboardInterrupt
>>> |
```

**Figura 5.** Repetição infinita encerrada por meio da opção “*Interrupt Execution*” — modo interativo no ambiente Shell Python da ferramenta Idle.

Como você pode notar, uma mensagem de erro “*KeyboardInterrupt*” é exibida em texto vermelho na parte inferior da janela. Ou seja, o programa recebeu a interrupção e parou de executar.

O Python Idle oferece um editor de arquivos completo, que permite escrever e executar programas Python a partir de arquivos. Além disso, o editor de arquivos embutido também inclui vários recursos, como conclusão de código e recuo (indentação) automático, que agilizarão o fluxo de trabalho no momento da codificação. Para iniciar um novo arquivo Python, é necessário selecionar a opção *File → New File* na barra de *menus*. Isso abrirá um arquivo em branco no editor (Figura 6).



**Figura 6.** Arquivo em branco (sem código) e ainda sem título — ferramenta Idle.

A partir da janela da Figura 6, é possível escrever um novo arquivo Python. Além disso, é possível abrir um arquivo Python existente selecionando a opção *File → Open* na barra de menus. Isso exibirá o navegador de arquivos do SO. Em seguida, deve-se encontrar e selecionar o arquivo Python a ser aberto.

Se houver o interesse por parte do usuário em ler o código-fonte de um módulo Python, ele poderá selecionar a opção *File → Path Browser*. Isso permitirá ver os mesmos módulos que o Python Idle pode ver. Quando se clica duas vezes em um deles, o editor de arquivos é aberto e o usuário pode lê-lo. O conteúdo dessa janela será o mesmo que os caminhos retornados quando se chama “`sys.path`” no Shell Python. Se o usuário souber o nome de um módulo específico que deseja visualizar, poderá selecionar *File → Module Browser* e digitar o nome do módulo na caixa exibida (Figura 7).



**Figura 7.** Navegador de caminhos para os módulos — ferramenta Idle.

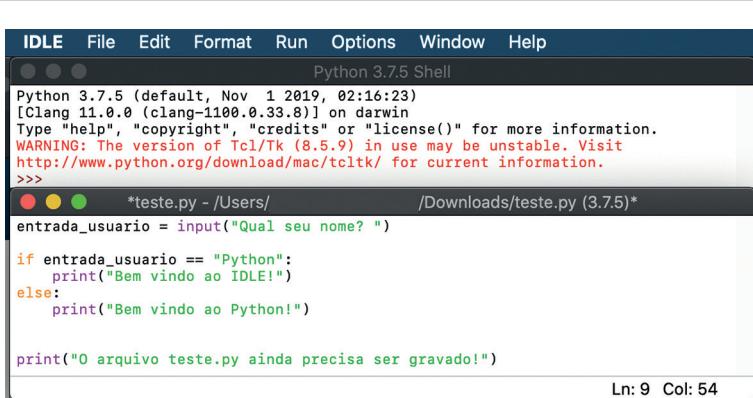
Depois de abrir um arquivo armazenado no computador no Python Idle, o usuário poderá fazer as alterações desejadas no arquivo (Figura 8). O conteúdo já existente no arquivo é exibido na janela aberta. A barra na parte superior da janela contém três informações importantes, que são:

- o nome do arquivo que se está editando;
- o caminho completo para a pasta onde está armazenado o arquivo no computador;
- a versão do interpretador Python que a plataforma Idle está usando.

Na Figura 8, está sendo editado o arquivo “`teste.py`”, localizado na pasta *Downloads*. A versão do Python é 3.7.5, que pode se ver entre parênteses. Há também dois números no canto inferior direito da janela, que são:

- **Ln** — mostra o número da linha em que o cursor está (no exemplo, linha 9);
- **Col** — mostra o número da coluna em que o cursor está (no exemplo, coluna 54).

É útil ver esses números para encontrar erros mais rapidamente. Eles também ajudam a garantir que o usuário fique dentro de determinada largura de linha. Além disso, existem algumas dicas visuais nessa janela que ajudam o usuário a lembrar de salvar (gravar no disco) o trabalho. É possível observar que o Python Idle usa asteriscos para informar que o arquivo tem alterações feitas, mas não gravadas. É possível salvar as alterações com o atalho de teclado padrão do sistema (geralmente são as combinações das teclas *Control + s*). Também é possível selecionar a opção *File → Save* na barra de menus. Quando se tratar de um arquivo novo, ainda não existente no computador, é recomendado gravar o novo arquivo com a extensão “.py”, para que o realce da sintaxe seja ativado na ferramenta Python Idle.



The screenshot shows the Python 3.7.5 Shell window. The menu bar includes IDLE, File, Edit, Format, Run, Options, Window, and Help. The title bar says "Python 3.7.5 Shell". The main area displays the following code:

```
Python 3.7.5 (default, Nov  1 2019, 02:16:23)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable. Visit
http://www.python.org/download/mac/tcltk/ for current information.
>>> *teste.py - /Users/          /Downloads/teste.py (3.7.5)*
entrad_usuario = input("Qual seu nome? ")

if entrad_usuario == "Python":
    print("Bem vindo ao IDLE!")
else:
    print("Bem vindo ao Python!")

print("O arquivo teste.py ainda precisa ser gravado!")
```

At the bottom right of the code area, it says "Ln: 9 Col: 54".

**Figura 8.** Novo código inserido no arquivo “teste.py”. O estado do arquivo é “não gravado”, demonstrado pelo asterisco ao lado da versão — ferramenta Idle.

Quando se deseja executar um arquivo criado no Idle, primeiro é necessário verificar se ele foi salvo. É possível verificar se o arquivo foi salvo corretamente procurando asteriscos em torno do nome do arquivo na parte superior da janela do editor de arquivos (Figura 9). Além disso, o Python Idle lembrará o usuário da necessidade de salvar o arquivo sempre que ele tentar executar um arquivo não salvo.

Para executar um arquivo em Idle, basta pressionar a tecla *F5* no teclado. É possível também selecionar a opção *Run → Run Module* na barra de *menus*. Qualquer uma das opções reiniciará o interpretador do Python e executará o código que foi escrito com o intérprete atualizado. O processo seria o mesmo de quando se executa o código de um arquivo Python pelo terminal. Por exemplo: “python -i teste.py”.

The screenshot shows the Python IDLE interface. The menu bar includes IDLE, File, Edit, Shell, Debug, Options, Window, and Help. A toolbar below the menu has three circular icons. The main window title bar says "teste.py - /Downloads/teste.py (3.7.5)". The code editor contains the following Python script:

```

IDLE  File  Edit  Shell  Debug  Options  Window  Help
● ● ●  teste.py - /  /Downloads/teste.py (3.7.5)
entrada_usuario = input("Qual seu nome? ")
if entrada_usuario == "Python":
    print("Bem vindo ao IDLE!")
else:
    print("Bem vindo ao Python!")

print("O arquivo teste.py foi gravado e executado!")

```

Below the code editor, status information reads "Ln: 6 Col: 0". The Python 3.7.5 Shell window shows the following output:

```

Python 3.7.5 (default, Nov  1 2019, 02:16:23)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /Users/          /Downloads/teste.py =====
Qual seu nome? Jorge
Bem vindo ao Python!
O arquivo teste.py foi gravado e executado!
>>>

```

Below the shell, status information reads "Ln: 9 Col: 4".

**Figura 9.** Código do arquivo “teste.py” já executado. O estado do arquivo é “gravado”, demonstrado pela ausência do asterisco ao lado da versão — ferramenta Idle.

Quando seu código terminar de ser executado, o intérprete saberá tudo sobre ele, incluindo quaisquer variáveis globais, funções e classes. Isso torna o Python Idle um ótimo lugar para inspecionar os dados se algo der errado. Se for necessário interromper o programa no momento da execução, basta pressionar a combinação das teclas *Control + c* no teclado para que o intérprete pare a execução do código.

### 3 Suporte ao desenvolvimento de software no Idle

Agora que você já viu como escrever, editar e executar arquivos no Python Idle, é o momento de ver como acelerar o seu fluxo de trabalho usando um

IDE. O editor Idle do Python oferece alguns recursos que existem na maioria dos IDEs profissionais para ajudar os usuários a codificar mais rapidamente. Esses recursos incluem recuo (indentação) automático, dicas de conclusão de código, chamada de código e contexto de código.

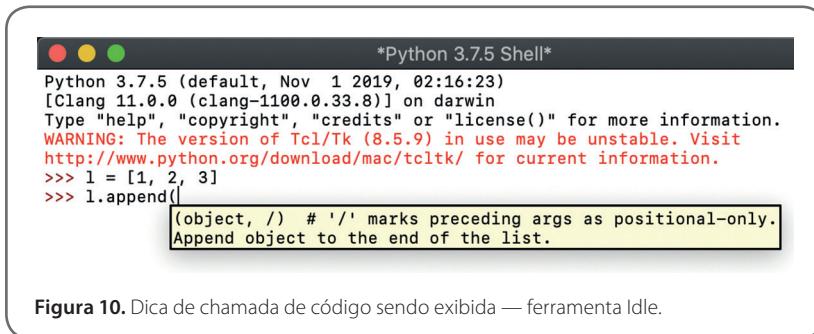
## Recuo automático de código

O Idle recuará automaticamente o código quando for necessário iniciar um novo bloco. Isso geralmente acontece depois que se digita dois-pontos (:). Assim que é pressionada a tecla *Enter* após os dois-pontos, o cursor se move automaticamente sobre certo número de espaços e inicia um novo bloco de código. É possível configurar quantos espaços o cursor moverá para o recuo nas configurações, porém o padrão são quatro espaços em branco. Os desenvolvedores do Python concordaram com um estilo padrão para que o código fosse bem escrito, e isso inclui regras sobre recuo, espaço em branco e muito mais. Esse estilo padrão foi formalizado e agora é conhecido como PEP-0008.

## Conclusão de código e dicas de chamada

Ao escrever um código para um projeto grande ou um problema complicado, você pode gastar muito tempo digitando todo o código necessário. O preenchimento de código ajuda a economizar tempo de digitação, tentando finalizar o código para o programador. O Python Idle possui a funcionalidade básica de conclusão de código, mas a ferramenta só consegue preencher automaticamente os nomes de funções e classes. Para usar o preenchimento automático no editor, basta pressionar a tecla *Tab* após uma sequência de texto.

Além do preenchimento de código, o Python Idle também fornece dicas de chamadas. Uma dica de chamada é uma dica para determinada parte do código a fim de ajudar o programador a lembrar do que dado elemento precisa. Sempre depois de digitar o parêntese do lado esquerdo para iniciar uma chamada de função, uma dica de chamada será exibida se não for digitado nada por alguns segundos. Por exemplo, se programador não consegue lembrar-se de como utilizar o método “*append()*” da classe “*list*”, para realizar a inclusão de mais um item em uma lista, pode fazer uma pausa após o parêntese de abertura para exibir a dica de chamada (Figura 10).



The screenshot shows the Python 3.7.5 Shell interface. At the top, there are three colored circles (red, yellow, green). Below them, the title bar reads "\*Python 3.7.5 Shell\*". The main window displays the following text:

```
Python 3.7.5 (default, Nov  1 2019, 02:16:23)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable. Visit
http://www.python.org/download/mac/tcltk/ for current information.
>>> l = [1, 2, 3]
>>> l.append(
    (object, /) # '/' marks preceding args as positional-only.
    Append object to the end of the list.
```

A tooltip box is overlaid on the screen, containing the text: "(object, /) # '/' marks preceding args as positional-only. Append object to the end of the list."

**Figura 10.** Dica de chamada de código sendo exibida — ferramenta Idle.

A dica de chamada será exibida como uma nota *pop-up*, lembrando o programador de como fazer o *append* em uma lista. Dicas de chamada como essas fornecem informações úteis em tempo real, enquanto o programador escreve o código.

## Contexto de código

A funcionalidade do contexto de código é um recurso interessante do editor de arquivos Python Idle. Ele mostra o escopo de uma função, classe, repetições ou outra construção. Isso é particularmente útil quando você está percorrendo um arquivo longo e precisa acompanhar onde está enquanto analisa o código no editor. Para ativá-lo, basta selecionar *Options* → *Show Code Contex* na barra de menus e será exibida uma barra cinza na parte superior da janela do editor (Figura 11).



The screenshot shows the Python Idle interface. At the top, there are three colored circles (red, yellow, green). The title bar displays "teste.py - /Users/" and "/Downloads/teste.py (3.7.5)". The main window contains the following Python code:

```
def main():
    if entrada_usuario == "Python":
        print("Bem vindo ao IDLE!")
    else:
```

In the bottom right corner, there is a status bar with the text "Ln: 1 Col: 0".

**Figura 11.** Contexto de código sendo exibido — ferramenta Idle.

À medida que o código é percorrido, o contexto que contém cada linha de código fica dentro dessa barra cinza. Isso significa que a função “`print()`” que se vê na Figura 11 faz parte da função principal “`main()`” e também está dentro do corpo da instrução “`if`”. Quando se alcança uma linha que está fora do escopo da função em questão, o código que está na barra cinza desaparece e a barra fica limpa.

## Modo de *debug* do interpretador

*Bugs* são problemas inesperados que ocorrem em um programa. Eles podem aparecer de várias formas, e alguns são mais difíceis de corrigir do que outros. Alguns *bugs* são complicados o suficiente para que o programador não possa percebê-los apenas lendo o código do programa. Felizmente, o Python Idle fornece algumas ferramentas básicas que ajudam o programador a depurar programas com facilidade. Para executar o código com o depurador interno, é necessário ativar esse recurso. Para fazer isso, você precisa estar na janela do modo interativo; então, basta selecionar a opção *Debug* → *Debugger* na barra de *menus* do Python Idle. Na janela do intérprete Python, será exibido “[DEBUG ON]” antes das marcações “>>>”, o que significa que o intérprete está ativado no modo de depuração, pronto e aguardando as instruções a serem avaliadas. Quando se executa o arquivo Python, a janela do depurador é exibida.

Antes de executar o arquivo Python com o modo de depuração ativado, é necessário marcar o código com um ponto de interrupção (*breakpoint*). Trata-se de uma linha cujo código, ao ser executado, deve levar o intérprete a fazer uma pausa. O *breakpoint* só funcionará quando o modo *debug* estiver ativado. Para definir um ponto de interrupção, o programador pode clicar com o botão direito do *mouse* na linha de código que deseja pausar. Isso destacará a linha de código em amarelo como uma indicação visual de um ponto de interrupção definido. É possível definir vários pontos de interrupção no código, tantos quantos sejam necessários. Para desfazer um ponto de interrupção, deve-se clicar com o botão direito do *mouse* na mesma linha do *breakpoint* e selecionar *Clear Breakpoint*.

Depois de definir os pontos de interrupção e ativar o modo *debug*, basta executar o código normalmente. A janela do depurador será exibida e o programador poderá começar a percorrer o código manualmente (Figura 12).

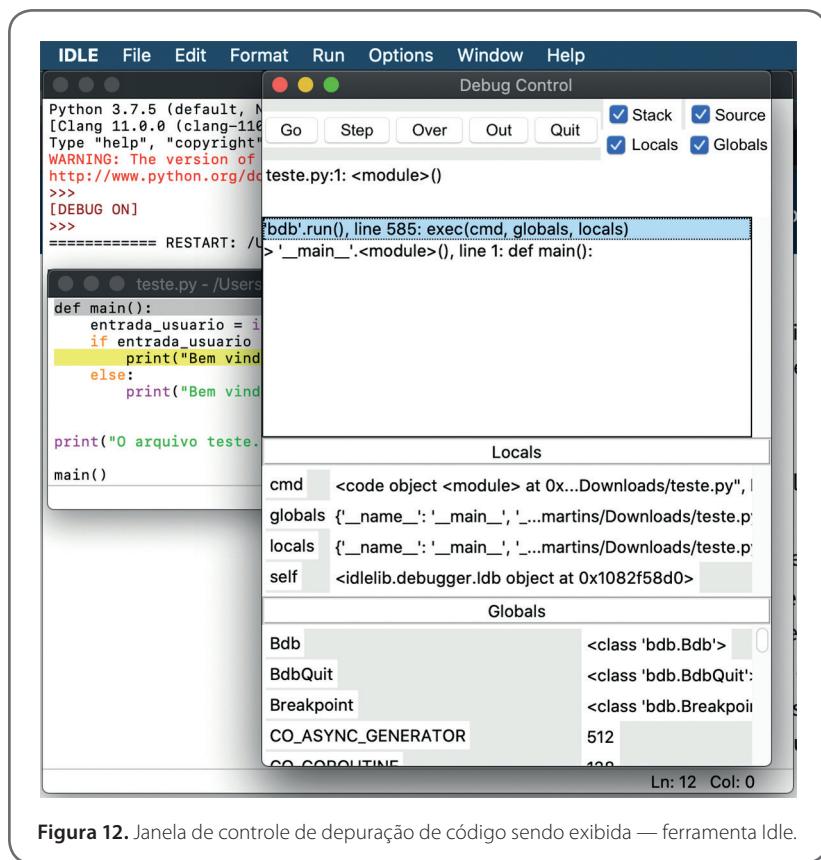


Figura 12. Janela de controle de depuração de código sendo exibida — ferramenta Idle.

Na janela do controle de depuração, é possível inspecionar os valores das variáveis locais e globais à medida que o código é executado. Isso fornece informações sobre como os dados estão sendo manipulados no algoritmo. Também estão disponíveis no controle de depuração os botões de navegação. Eles têm as funções listadas a seguir.

- **Go:** avança a execução para o próximo ponto de interrupção.
- **Step:** executa a linha atual e avança para a próxima linha.

- **Over:** se a linha de código atual contiver uma chamada de função, esse botão faz com que o *debug* passe por cima dessa função. Em outras palavras, apenas se executa essa função (internamente) e se avança para a próxima linha, sem mostrar o passo a passo pelas linhas de código da função.
- **Out:** se a linha de código atual estiver em uma função, pressiona-se esse botão para sair dessa função. Em outras palavras, continua a execução dessa função até retornar dela.
- **Quit:** encerra a avaliação pelo *debug*.

O programador deve tomar cuidado, pois não há botão para voltar na execução. É possível apenas avançar no tempo por meio da execução de cada linha de código do algoritmo.

Além dos botões de navegação da execução, também há quatro caixas de seleção na janela de depuração de código. Essas caixas de seleção exibem, quando selecionadas, as informações listadas a seguir.

- **Globals:** exibe as informações globais do programa.
- **Locals:** exibe as informações locais do programa durante a execução.
- **Stack:** exibe as funções que são executadas durante a execução.
- **Source:** exibe o arquivo que está sendo executado no editor do Python Idle.

Se você selecionar uma dessas opções, serão exibidas as informações relevantes na janela de depuração.

## Erros de exceções

Quando é exibido um erro no intérprete, o Python Idle permite que o usuário navegue diretamente para o arquivo ou linha com erro. Para isso, o usuário deve selecionar *Debug* → *Go to File/Line* na barra de menus. Isso abrirá o arquivo, se ainda não estiver aberto, e levará o usuário direto para a linha que contém o erro (Figura 13). Esse recurso funciona independentemente de o modo *debug* estar ou não ativado.

```

Python 3.7.5 (default, Nov  1 2019, 02:16:23)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
WARNING: The version of Tcl/Tk (8.5.9) in use may be unstable. Visit
http://www.python.org/download/mac/tcltk/ for current information.
>>> ===== RESTART: /Users/          /Downloads/teste.py =====
Traceback (most recent call last):
  File "/Users/          /Downloads/teste.py", line 5 in <module>
    c = a / b
ZeroDivisionError: division by zero
>>>

```

teste.py - /Users/ /Downloads/teste.py (3.7.5)

```

1 a = 2
2
3 b = 0
4
5 b = a / b
6
7 print(c)
8

```

Ln: 5 Col: 0

**Figura 13.** Janela de código do arquivo “teste.py” com a linha 5 em destaque — ferramenta Idle.



## Referências

PYTHON SOFTWARE FOUNDATION. *IDLE is Python's integrated development and learning environment*. Delaware: Python Software Foundation, 2020. Disponível em: <https://docs.python.org/pt-br/3/library/idle.html>. Acesso em: 17 jan. 2020.

RAMALHO, L. *Aprenda a programar: Python Brasil*. [S. l.: s. n.], 2012. Disponível em: <https://wiki.python.org.br/AprendaProgramar>. Acesso em: 17 jan. 2020.

ROUSE, M. Integrated development environment (IDE). In: SEARCHSOFTWAREQUALITY. [S. l.: s. n.], 2018. Disponível em: <https://searchsoftwarequality.techtarget.com/definition/integrated-development-environment>. Acesso em: 17 jan. 2020.



## Fique atento

Os *links* para sites da Web fornecidos neste capítulo foram todos testados, e seu funcionamento foi comprovado no momento da publicação do material. No entanto, a rede é extremamente dinâmica; suas páginas estão constantemente mudando de local e conteúdo. Assim, os editores declararam não ter qualquer responsabilidade sobre qualidade, precisão ou integralidade das informações referidas em tais *links*.

Encerra aqui o trecho do livro disponibilizado para esta Unidade de Aprendizagem. Na Biblioteca Virtual da Instituição, você encontra a obra na íntegra.

Conteúdo:



SOLUÇÕES  
EDUCACIONAIS  
INTEGRADAS