

A listagem do programa 1.1 tem apenas uma linha de código. A palavra `print` é uma função utilizada para enviar dados para a tela do computador. Ao escrevermos `print ("Olá")`, ordenamos ao computador que exiba o texto “Olá!” na tela. Veja o que seria exibido na tela ao se executar esse programa no computador:

```
Olá!
```

Observe que as aspas (") não aparecem na tela. Esse é um dos detalhes da programação: precisamos marcar ou limitar o início e o fim de nossas mensagens com um símbolo, nesse caso, aspas. Como podemos exibir praticamente qualquer texto na tela, as primeiras aspas indicam o início da mensagem, e as seguintes, o fim. Ao programar, não podemos esquecer as limitações do computador. Um computador não interpreta textos como seres humanos. A máquina não consegue diferenciar o que é programa ou mensagem. Se não utilizarmos as aspas, o computador interpretará nossa mensagem como um comando da linguagem Python, gerando um erro.

O interpretador Python é uma grande ferramenta para o aprendizado da linguagem. O interpretador é o programa que permite digitar e testar comandos escritos em Python e verificar os resultados instantaneamente. Veremos como utilizar o interpretador na seção 2.2.

A linguagem Python foi escolhida para este livro por simplificar o trabalho de aprendizado e fornecer grande poder de programação. Como é um software livre, disponível praticamente para qualquer tipo de computador, sua utilização não envolve a aquisição de licenças de uso, muitas vezes a um custo proibitivo.

Bem-vindo ao mundo da programação!

CAPÍTULO 2

Preparando o ambiente

Antes de começarmos, precisamos instalar o interpretador da linguagem Python. O interpretador é um programa que aceita comandos escritos em Python e os executa, linha a linha. É ele quem vai traduzir nossos programas em um formato que pode ser executado pelo computador. Sem o interpretador Python, nossos programas não podem ser executados, sendo considerados apenas como texto. O interpretador também é responsável por verificar se escrevemos corretamente nossos programas, mostrando mensagens de erro, caso encontre algum problema.

O interpretador Python não vem instalado com o Microsoft Windows: você deverá instalá-lo fazendo um download da internet. Se você utiliza Mac OS X ou Linux, provavelmente isso já foi feito. Veja, a seguir, como verificar se você tem a versão correta.

2.1 Instalação do Python

Neste livro, usamos o Python versão 3.4. A linguagem sofreu diversas modificações entre as versões 2 e 3. A versão 3.4 foi escolhida por permitir a correta utilização dos novos recursos, além do fato de que é interessante aprendermos a versão mais nova. A versão 2.7 é também muito interessante, mas permite misturar a sintaxe (forma de escrever) do Python 2 com a do Python 3. Para evitar complicações desnecessárias, apresentaremos apenas as novas formas de escrever, e deixaremos o Python 3.4 verificar se fizemos tudo corretamente. O Python 3.4 ainda não estava disponível para todas as distribuições Linux na época de lançamento desta edição. Você pode utilizar também o Python 3.3 com todos os exemplos do livro.

2.1.1 Windows

Python está disponível em todas as versões de Windows. Como é um software livre, podemos baixar o interpretador Python gratuitamente no site <http://www.python.org>. O site deve parecer com o da figura 2.1. Veja que, no centro da página, temos a opção **Downloads**. Passe o mouse sobre **Downloads** e espere o menu abrir.

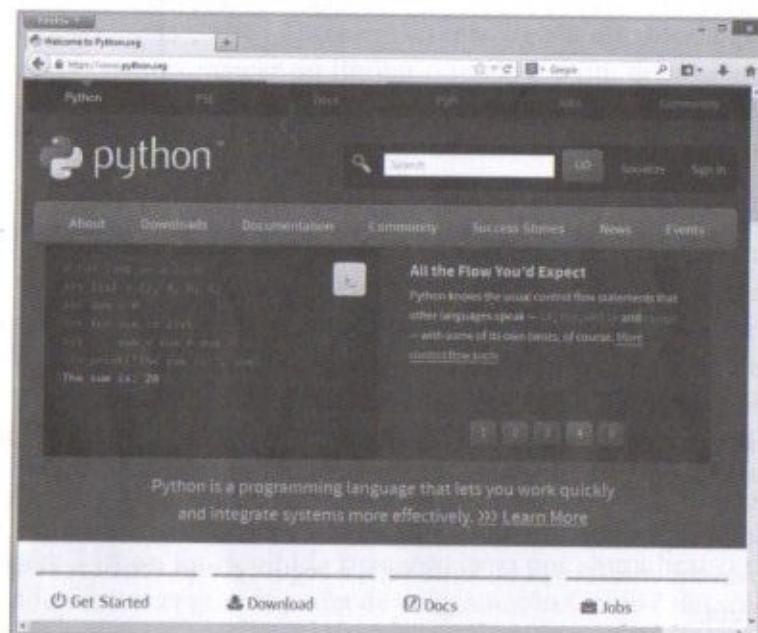


Figura 2.1 – Site da Python Foundation.

A figura 2.2 mostra o menu da página de download aberto. Procure o texto Python 3.4.0. Se uma versão mais nova da série 3 estiver disponível, você deve escolhê-la. Quando este livro foi escrito, a versão mais nova era a 3.4.0. Clique no botão para iniciar o download do arquivo.

Se você não estiver utilizando o Firefox, o menu de download pode não aparecer. Nesse caso, clique em Windows e siga as instruções nas páginas que irão se abrir. Se sua versão de Windows é Windows Vista, 7, 8 ou 8.1 32 bits, escolha Python 3.4.0 Windows x86 MSI. Caso utilize o Windows Vista, 7, 8 ou 8.1 64 bits, escolha Python 3.4.0 Windows x86-64 MSI Installer. Se você não conseguir encontrar esses arquivos, tente digitar o endereço completo da página de download: <https://www.python.org/downloads/release/python-340/>.

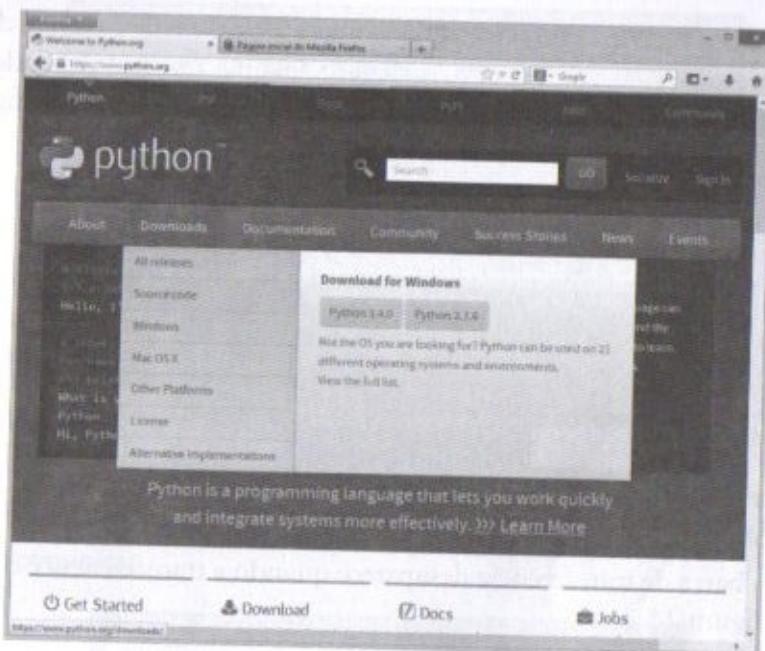


Figura 2.2 – Página de download.

Nos exemplos a seguir, faremos o download da versão 3.4.0 para Windows 8.1. A versão do Python pode mudar, e o nome do arquivo, também. Se você escolheu a versão de 32 ou 64 bits, o nome do arquivo será um pouco diferente, mas não se preocupe: se você escolher a versão errada, ela simplesmente não funcionará. Você pode então tentar outra versão.

A figura 2.3 mostra a janela de download do Firefox. Essa janela também pode variar de acordo com a versão de seu navegador de internet. Clique no botão **download** para iniciar a transferência do arquivo. Aqui, o Firefox foi utilizado como exemplo, mas você pode utilizar qualquer outro navegador de internet, como Internet Explorer, Google Chrome, Opera ou Safari.

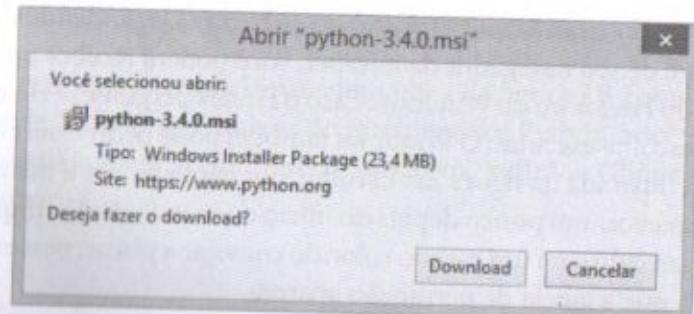


Figura 2.3 – Janela de confirmação de download do Firefox.

O interpretador Python ocupa aproximadamente 24 MB em disco (antes da instalação), e a transferência do arquivo deve demorar alguns minutos, dependendo da velocidade de conexão à internet. O Firefox exibe o download como na figura 2.4.

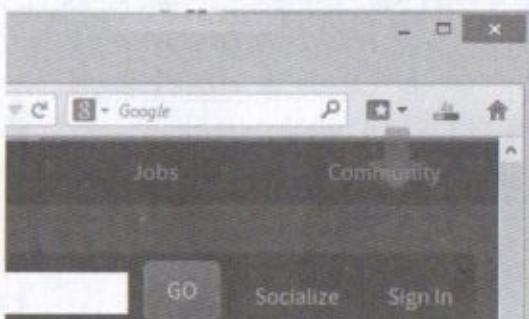


Figura 2.4 – Janela de download com arquivo python-3.4.0.msi sendo baixado.

Veja que a barra de transferência desaparece quando a transferência é concluída, como na figura 2.5.

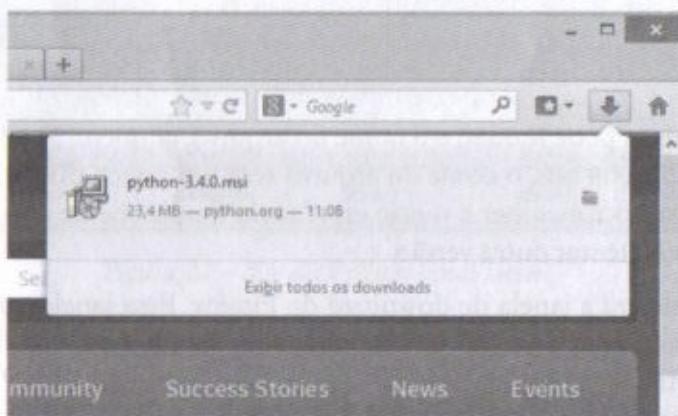


Figura 2.5 – Janela de download com a transferência do arquivo finalizada.

Clique no nome do arquivo que você acabou de baixar. Dependendo da versão do seu Windows e de seu navegador de internet, você poderá receber um pedido de confirmação do Firefox ou do Windows. Caso o Firefox pergunte se deseja executar o programa, escolha executar. O Windows também pode pedir uma confirmação de instalação, ilustrada na figura 2.6. Escolha Sim para aprovar a instalação. Essa janela pode aparecer um pouco depois do início da instalação. Verifique também se um ícone parecido com um escudo colorido começar a piscar; nesse caso, clique no ícone para que a janela de permissão apareça.

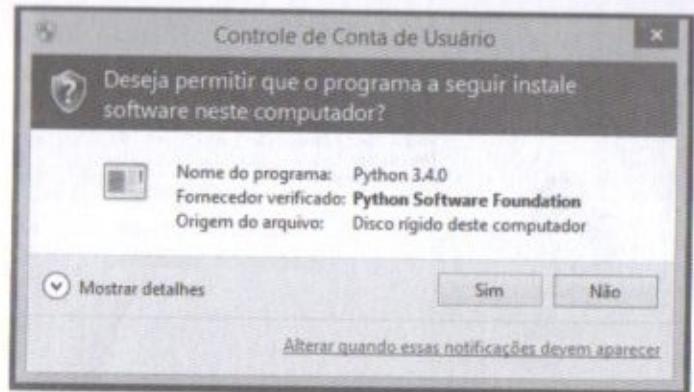


Figura 2.6 – Janela de confirmação da execução do instalador Python.

A figura 2.7 mostra a tela inicial do instalador. Até agora, não instalamos nada. É esse programa que vai instalar o Python 3 em seu computador. Clique no botão **Next** para continuar.



Figura 2.7 – Janela do instalador do Python.

Você deve estar com uma janela parecida com a da figura 2.8. Essa janela permite escolher onde instalar os arquivos do interpretador Python. Você pode mudar a pasta ou simplesmente aceitar o local proposto (padrão). Clique no botão **Next** para continuar.

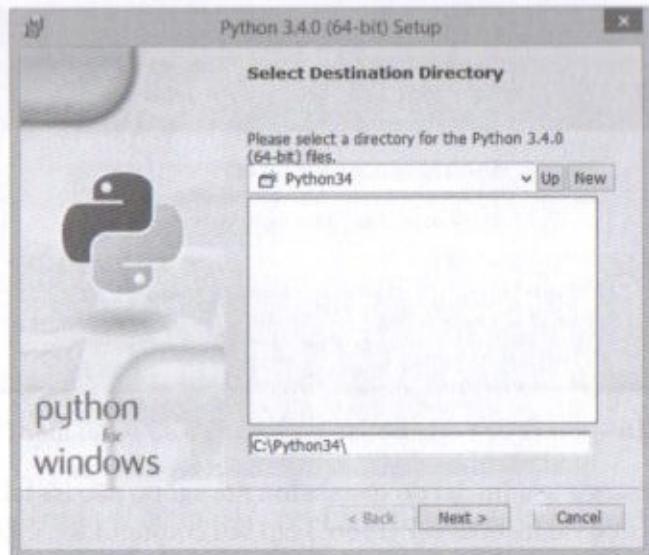


Figura 2.8 – Janela do instalador pedindo confirmação da pasta (diretório) de instalação do Python.

A janela da figura 2.9 permite selecionar que partes do pacote do Python queremos instalar. Essa janela é útil para utilizadores mais avançados da linguagem, mas vamos simplesmente aceitar o padrão, clicando no botão **Next** para continuar.

Finalmente começamos a instalar os arquivos que precisamos. Uma barra de progresso como a da figura 2.10 será exibida. Aguarde a finalização do processo.

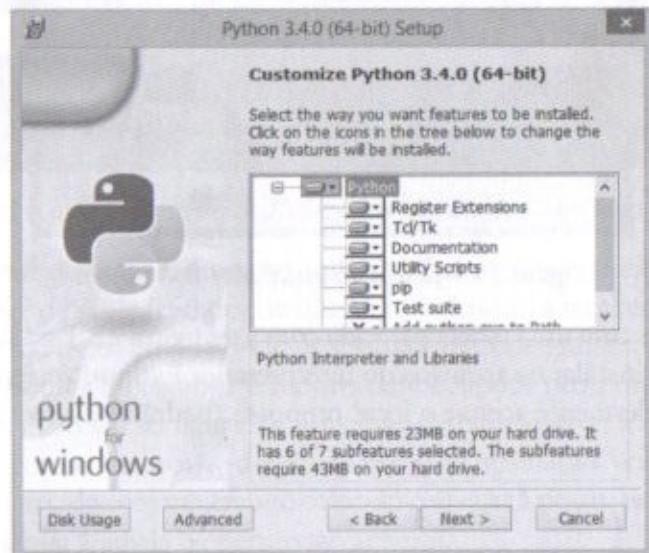


Figura 2.9 – Janela de seleção de pacotes do instalador Python.

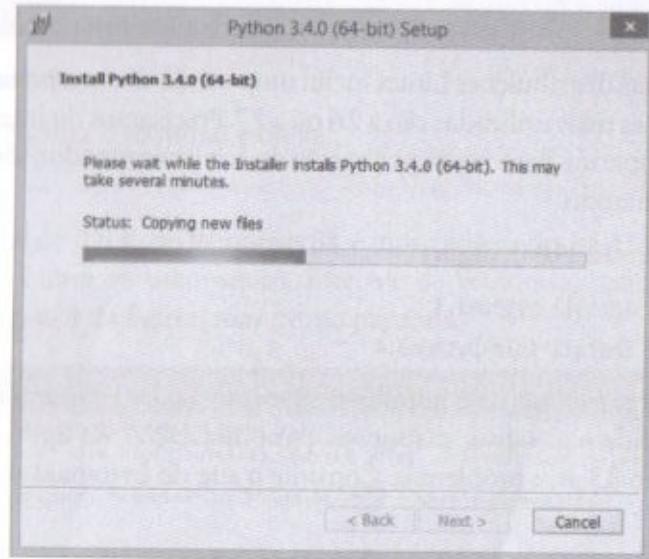


Figura 2.10 – Janela do instalador exibindo o progresso da instalação.

A instalação foi concluída. Clique no botão **Finish**, mostrado na figura 2.11, para terminar o instalador.



Figura 2.11 – Janela de finalização do instalador.

2.1.2 Linux

Grande parte das distribuições Linux inclui uma versão do interpretador Python; porém, as versões mais utilizadas são a 2.6 ou a 2.7. Precisamos do interpretador na versão 3.3 ou superior. Para verificar a versão de seu interpretador, digite `python -V` na linha de comando.

No Ubuntu (versão 14.04), digite:

```
sudo apt-get install python3.4
sudo apt-get install idle-python3.4
```

Você precisa de permissões de administrador (root) para efetuar a instalação. Se a versão 3.4 ainda não estiver disponível para instalação via apt-get, você pode utilizar a versão 3.3 sem problemas. Consulte o site do livro para mais detalhes.

2.1.3 Mac OS X

Os Mac OS X Leopard, Snow Leopard, Lion, Mountain Lion, Maverick e Yosemite vêm com uma versão do interpretador Python da Apple. No entanto, essa versão não é a 3.4. Para contornar o problema, instale o MacPorts (<http://www.macports.org/>), fazendo o download do arquivo dmg, e, depois, instale o Python 3.4 com:

```
sudo port install python34
```

Para executar o interpretador recém-instalado, digite `python3.4` na linha de comando. Você também pode utilizar outros gerenciadores de pacote disponíveis para Mac OS X, como Fink (<http://www.finkproject.org/>) e Homebrew (<http://brew.sh/>). Uma versão instalável, em formato dmg, também está disponível no site da <http://www.python.org>.

2.2 Usando o interpretador

Com o Python instalado, vamos começar a trabalhar.

O IDLE é uma interface gráfica para o interpretador Python, permitindo também a edição e execução de nossos programas. No Windows Vista e no Windows 7, você deve ter uma pasta no menu **Iniciar > Programas > Python 3.4**. Escolha **IDLE**. No Windows 8 e 8.1, procure por Python 3.4 na lista de aplicativos e, então, por IDLE (Python GUI).

No Linux, abra o terminal e digite:

```
idle-python3.4 &
```

No Mac OS X, abra o terminal e digite:

```
IDLE3.4 &
```

A janela inicial do IDLE, no Windows 8.1, é mostrada na figura 2.12. Se você utiliza o Mac OS X, Linux ou uma versão diferente de Windows, essa janela não será exatamente igual à da figura, mas muito parecida.

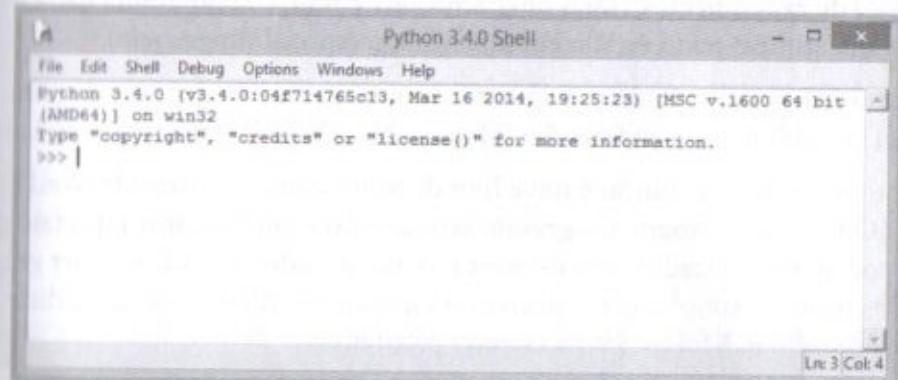


Figura 2.12 – Janela inicial do IDLE.

Observe que o cursor está posicionado dentro da janela Python Shell, e que a linha de comandos é iniciada pela sequência `>>>`.

Digite:

```
print("Olá")
```

Pressione a tecla **ENTER**. Você deve obter uma tela parecida com a da figura 2.13.

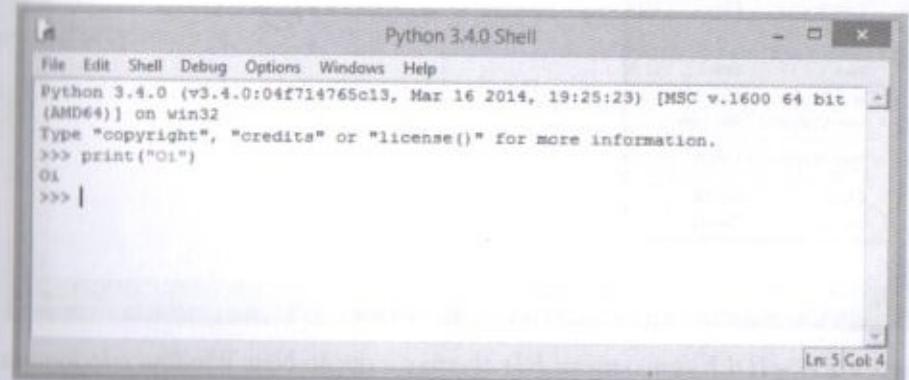


Figura 2.13 – IDLE mostrando o resultado de `print("Olá")`.

Uma das grandes vantagens do Python é contar com o interpretador de comandos. Você pode abrir uma janela como essa sempre que tiver alguma dúvida em Python. O interpretador permite que você verifique o resultado de um comando instantaneamente. Veremos mais adiante que esse tipo de verificação é muito importante e facilita o aprendizado da linguagem.

2.3 Editando arquivos

Nem só de experimentos vive o programador Python. Um programa nada mais é que um arquivo-texto, escrito em um formato especial (linguagem).

No caso da linguagem Python, podemos usar qualquer editor de textos disponível: PSPad ou Sublime, no Windows; TextMate, no Mac OS X; Vim ou Emacs, no Linux.

O que você não deve utilizar é um editor de textos como o Microsoft Word ou o OpenOffice. Esses programas gravam seus arquivos em formatos especiais que não podem ser utilizados para escrever programas, salvo se você escolher gravar apenas texto ou simplesmente gravar no formato txt. Além disso, um editor de textos comum não foi feito para escrever programas.

Você pode também utilizar o editor de textos incluído na instalação do interpretador Python. Com o interpretador aberto, clique no menu **File** e depois selecione a opção **New Window**, como mostra a figura 2.14.

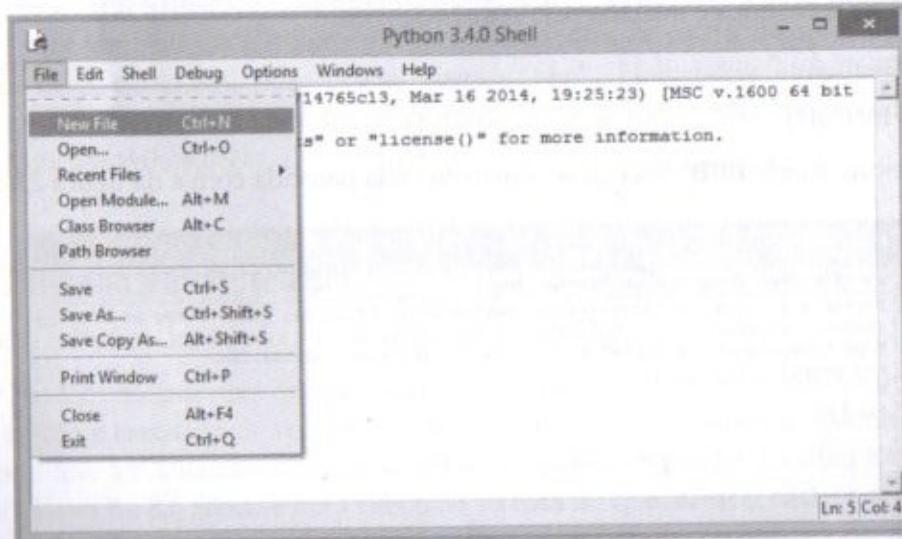


Figura 2.14 – IDLE com o menu File aberto e a opção New Window selecionada.

Como utilizamos a versão 3.4 do interpretador, é muito importante que você use um editor de textos correto. Isso é, um editor que suporte a edição de textos em UTF-8. Se você não sabe o que é UTF-8, ou se o seu editor de textos suporta esse formato de codificação, utilize o IDLE como editor de textos. Além de suportar a edição no formato UTF-8, que permitirá a utilização de acentos em nossos programas, ele é preparado para os colorir em Python, tornando a leitura mais fácil.

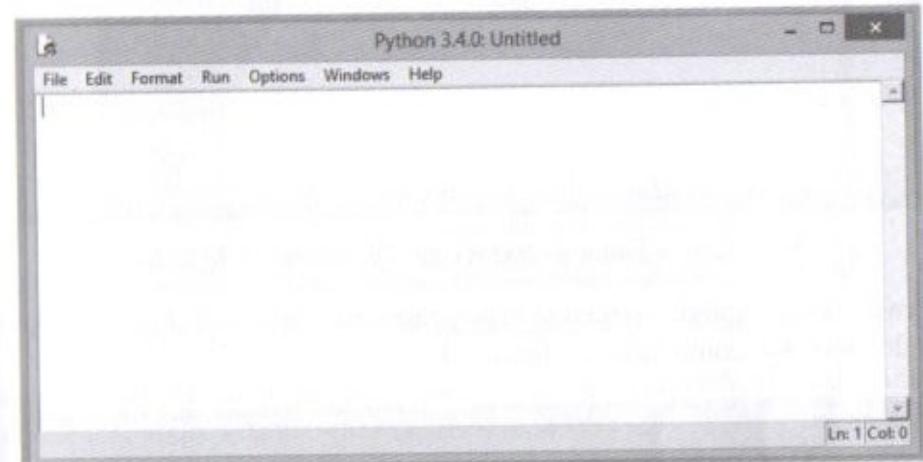


Figura 2.15 – Janela do editor de textos do IDLE.

Uma nova janela, como a da figura 2.15, deverá aparecer. É nessa janela que escreveremos nossos programas. Observe que, embora parecida com a janela principal do IDLE, a janela tem opções de menu diferentes da outra. Para esclarecer essa separação, chamaremos a primeira de janela do interpretador; e a segunda, de janela do editor de textos. Se você ainda estiver em dúvida, a janela do editor de textos é a que apresenta a opção **Run** no menu.

Experimente um pouco, escrevendo:

```
print("Olá")
```

Sua janela do editor de textos deverá ser semelhante à mostrada na figura 2.16.

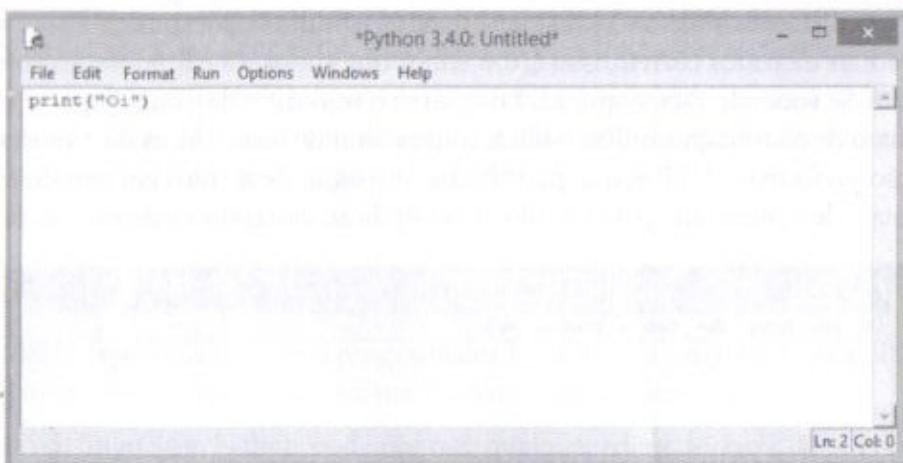


Figura 2.16 – Editor de textos com o programa já digitado.

Agora que escrevemos nosso pequeno programa, vamos salvá-lo. Escolha a opção **Save** do menu **File**, como mostra a figura 2.17.

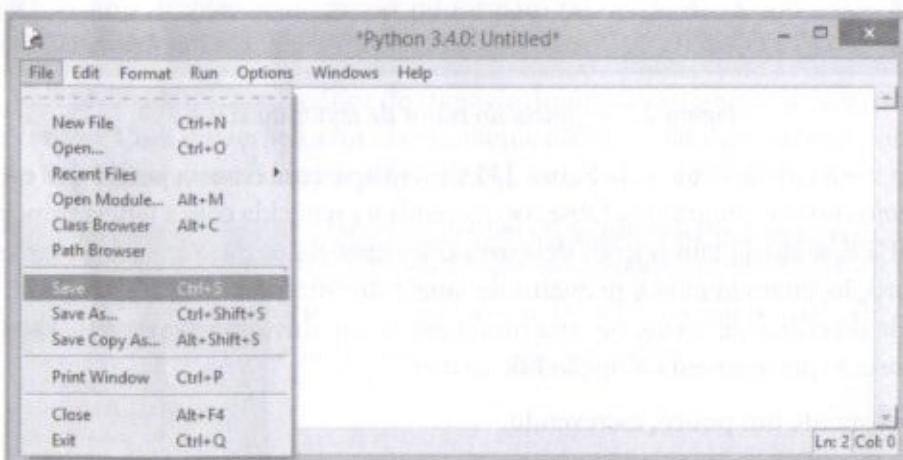


Figura 2.17 – Janela do editor de textos com o menu **File** aberto e a opção **Save** selecionada.

Uma janela-padrão de gravação de arquivos será exibida. A janela muda com a versão do sistema operacional, mas se você utiliza Windows 8.1, ela parecerá com a da figura 2.18. Escreva *teste.py* no nome do arquivo e clique no botão para salvar. Atenção: a extensão .py não é adicionada automaticamente pelo IDLE. Lembre-se de sempre gravar seus programas escritos em Python com a extensão .py.

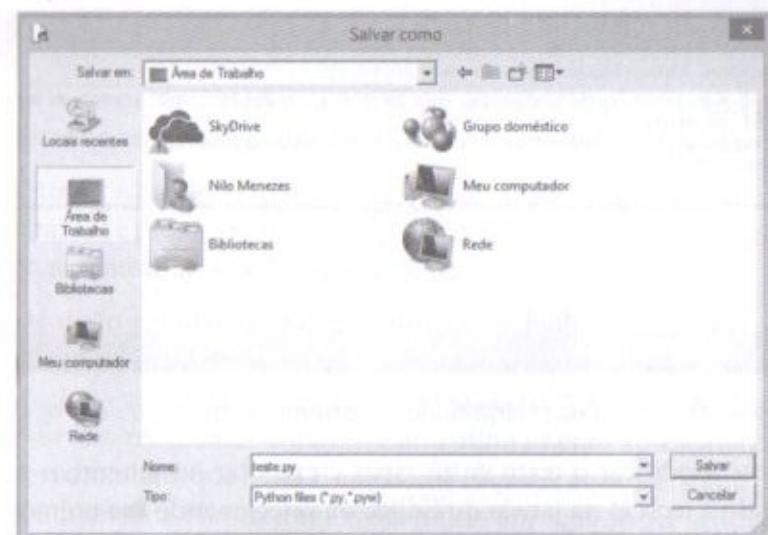


Figura 2.18 – Janela de gravação de arquivos do Windows 8.1.

Agora clique no menu **Run** para executar seu programa. Você pode também pressionar a tecla F5 com o mesmo efeito. Essa operação pode ser visualizada na figura 2.19.

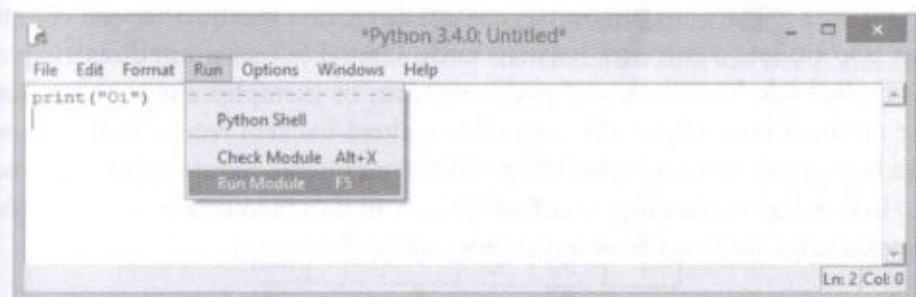


Figura 2.19 – Janela do interpretador mostrando a opção **Run Module** selecionada.

Seu programa será executado na outra janela, a do interpretador. Veja que uma linha com a palavra RESTART apareceu, como na figura 2.20. Observe que obtivemos o mesmo resultado de nosso primeiro teste.

```

Python 3.4.0 (v3.4.0:04f714765c13, Mar 16 2014, 19:25:23) [MSC v.1600 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.

>>> print("Olá")
Oi
>>> RESTART
>>>
>>> Olá
>>>

```

Figura 2.20 – Janela do interpretador mostrando a execução do programa.

Experimente modificar o texto entre aspas e executar novamente o programa, pressionando a tecla **F5** na janela do Editor ou selecionando **Run** no menu.

Agora estamos com tudo instalado para continuar aprendendo a programar. Durante todo o livro, programas de exemplo serão apresentados. Você deve digitá-los na janela do editor de textos, gravá-los e executá-los. Você pode voltar a ler as seções 2.2 e 2.3 sempre que precisar. Com o tempo, essas tarefas serão memorizadas e realizadas sem esforço.

Experimente gravar seus programas em um diretório específico em seu computador. Isso ajudará a encontrá-los mais tarde. Uma dica é criar um diretório para cada capítulo do livro, mas você pode organizar os exemplos como quiser. Você pode também fazer cópias dos exemplos e alterá-los sem receio. Embora você possa baixar todos os exemplos já digitados, é altamente recomendado que você leia o livro e digite cada programa. Essa prática ajuda a memorizar as construções da linguagem e facilita a leitura dos programas.

2.4 Cuidados ao digitar seus programas

Ao digitar um programa no editor de textos, verifique se você o copiou exatamente como apresentado. Em Python, você deve tomar cuidado com os seguintes itens:

- Letras maiúsculas e minúsculas são diferentes. Assim, `print` e `Print` são completamente diferentes, causando um erro caso você digite o P maiúsculo. No início, é comum termos o hábito de escrever a letra inicial de cada linha em letra maiúscula, causando erros em nossos programas. Em caso de erro, leia atentamente o que você digitou e compare com a listagem apresentada no livro.

- Aspas são muito importantes e não devem ser esquecidas. Toda vez que abrir aspas, não se esqueça de fechá-las. Se você se esquecer, seu programa não funcionará. Observe que o IDLE muda a cor do texto entre aspas, facilitando essa verificação.
- Parênteses não são opcionais em Python. Não remova os parênteses dos programas e preste a mesma atenção dada às aspas para abri-los e fechá-los. Todo parêntese aberto deve ser fechado.
- Espaços são muito importantes. A linguagem Python se baseia na quantidade de espaço em branco antes do início de cada linha para realizar diversas operações, explicadas posteriormente no livro. Não se esqueça de digitar o texto dos programas com o mesmo alinhamento apresentado no livro. Observe também que o IDLE ajuda nesses casos, avisando sobre problemas de alinhamento. Nunca junte duas linhas em uma só até sentir-se seguro sobre como escrever corretamente em Python.

2.5 Os primeiros programas

Vamos analisar nosso primeiro programa. A figura 2.21 mostra a separação entre o nome da função, os parênteses, a mensagem e as aspas. É muito importante saber o nome de cada uma dessas partes para o correto entendimento dos conceitos e programas apresentados.

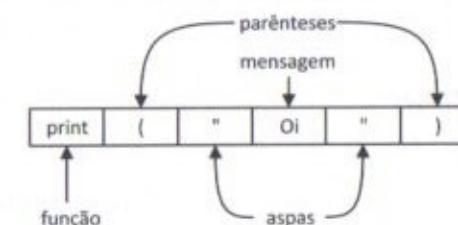


Figura 2.21 – Nomes dos elementos do primeiro programa.

A função `print` informa que vamos exibir algo na tela. Pode-se dizer que a função exibe uma mensagem na tela do computador. Sempre que quisermos mostrar algo para o usuário do computador, como uma mensagem, uma pergunta ou o resultado de uma operação de cálculo, utilizaremos a função `print`.

Para separar os programas do texto explicativo do livro, todo trecho de programa, funções, variáveis e demais itens em Python serão apresentados com uma fonte de texto diferente (monoespacada), como a utilizada para explicar a função `print`.

Voltando à figura 2.21, temos as aspas como elementos importantes. Elas são utilizadas para separar textos destinados ao usuário do computador do resto do programa. Utilizamos aspas para indicar o início e o fim do texto de nossa mensagem. A mensagem é nosso texto em si e será exibida exatamente como digitada. Os parênteses são utilizados para separar os parâmetros de uma função, no caso, os de `print`. Um parâmetro é um valor passado para uma função: no caso da função `print`, a mensagem a imprimir.

O interpretador Python também pode ser utilizado como calculadora. Experimente digitar `2+3` no interpretador, como mostra a listagem 2.1.

► Listagem 2.1 – Usando o interpretador como calculadora

```
>>> 2+3
5
```

Não se esqueça de pressionar a tecla **Enter** para que o interpretador saiba que você terminou a digitação. O resultado deve ser apresentado na linha seguinte. Podemos também subtrair valores, como na listagem 2.2.

► Listagem 2.2 – Subtração

```
>>> 5-3
2
```

Podemos combinar adição e subtração na mesma linha, como mostra a listagem 2.3.

► Listagem 2.3 – Adição e subtração

```
>>> 10-4+2
8
```

A multiplicação é representada por um asterisco (*); e a divisão, pela barra (/), como mostra a listagem 2.4.

► Listagem 2.4 – Multiplicação e divisão

```
>>> 2*10
20
>>> 20/4
5.0
```

Para elevar um número a um expoente, utilizaremos dois asteriscos (**) para representar a operação de exponenciação. Observe que não há qualquer espaço entre os dois asteriscos. Assim, para calcularmos 2^3 , escreveremos de forma semelhante ao conteúdo da listagem 2.5.

► Listagem 2.5 – Exponenciação

```
>>> 2**3
8
```

Podemos também obter o resto da divisão de dois números usando o símbolo %. Assim, para calcularmos o resto da divisão entre 10 e 3, digitariammos como mostrado na listagem 2.6.

► Listagem 2.6 – Resto da divisão inteira

```
>>> 10 % 3
1
>>> 16 % 7
2
>>> 63 % 8
7
```

Os parênteses são utilizados em Python da mesma forma que em expressões matemáticas, ou seja, para alterar a ordem de execução de uma operação. Para lembrar a ordem de precedência das operações, temos as seguintes prioridades:

1. Exponenciação ou potenciação (**).
2. Multiplicação (*) e divisão (/ e %).
3. Adição (+) e subtração (-).

A expressão $1500 + (1500 * 5 / 100)$ é equivalente a:

$$1500 + \left(\frac{1500 \times 5}{100} \right)$$

Não se esqueça de que, tanto em Python como na matemática, as operações de mesma prioridade são realizadas da esquerda para direita. Utilize parênteses sempre que precisar alterar a ordem de execução das operações e também para aumentar a clareza da fórmula.

Exercício 2.1 Converta as seguintes expressões matemáticas para que possam ser calculadas usando o interpretador Python.

$$10 + 20 \times 30$$

$$4^2 + 30$$

$$(9^4 + 2) \times 6 - 1$$

Exercício 2.2 Digite a seguinte expressão no interpretador:

$$10 \% 3 * 10 ** 2 + 1 - 10 * 4 / 2$$

Tente resolver o mesmo cálculo, usando apenas lápis e papel. Observe como a prioridade das operações é importante.

2.6 Conceitos de variáveis e atribuição

Além de operações simples de cálculo, o interpretador também pode ser usado para realizar operações mais complexas e mesmo executar programas completos. Antes de continuarmos, é importante observar o conceito de variáveis e como podemos usá-las em um programa. Em matemática, aprendemos o conceito de variável para representar incógnitas em equações do tipo $x + 1 = 2$, onde devemos determinar o valor de x , resolvendo a equação. Em programação, variáveis são utilizadas para armazenar valores e para dar nome a uma área de memória do computador onde armazenamos dados. Variáveis serão mais bem estudadas no capítulo 3. Por enquanto, podemos imaginar a memória do computador como uma grande estante, onde cada compartimento tem um nome. Para armazenar algo nesses compartimentos, usaremos o símbolo de igualdade ($=$) entre o nome do compartimento e o valor que queremos armazenar. Chamaremos essa operação de atribuição, onde um valor é atribuído a uma variável. Quando lermos nosso programa, as operações de atribuição serão chamadas de “recebe”, ou seja, uma variável recebe um valor.

A fim de simplificar as explicações de como um programa funciona, utilizaremos bolas pretas ① com números para relacionar uma determinada linha a um texto explicativo. Esses símbolos não fazem parte do programa e não devem ser digitados nem no interpretador nem no editor de textos.

Como quase tudo na vida, aprende-se a programar programando. Vamos escrever outro programa. Observe a listagem 2.7:

► Listagem 2.7 – O primeiro programa com variáveis

```
a = 2 ①
```

```
b = 3 ②
```

```
print (a + b) ③
```

Vejamos o que cada linha significa. Em ①, temos `a = 2`. Leia “a recebe 2”. Essa linha diz que uma variável chamada `a` receberá o valor 2. Variáveis em programação têm o mesmo significado que em matemática. Você pode entender uma variável como uma forma de guardar valores na memória do computador. Toda variável precisa ter um nome para que seu valor possa ser utilizado posteriormente. Esse conceito ficará mais claro um pouco mais adiante.

Em ②, temos `b = 3`. Leia “b recebe 3”. Essa linha realiza um trabalho muito parecido com o da linha anterior, mas a variável se chama `b`, e o valor é o número 3. Para entender o que faz essa linha, imagine que criamos um espaço na memória do computador para guardar outro valor, no caso, 3. Para podermos usar esse valor mais tarde, chamamos esse espaço de “b”.

A linha ③ solicita que o resultado da soma do conteúdo da variável `a` com o conteúdo da variável `b` seja exibido na tela. A função `print` realiza a impressão, mas, antes, o resultado de `a + b` é calculado. Veja que nessa linha estamos ordenando ao programa que calcule `a + b` e que exiba o resultado na tela. Como em matemática, passamos parâmetros ou valores para uma função usando parênteses. Esses parênteses são requeridos pelo interpretador Python. Vale se lembrar de `f(x)`; onde `f` é o nome da função, e `x` um parâmetro. No exemplo anterior, `print` é o nome da função; e o resultado de `a + b`, o valor passado como parâmetro. No decorrer deste livro, veremos diversas funções disponíveis no Python para realizar operações com o computador, como ler valores do teclado ou gravar dados em um arquivo.

Você pode experimentar o programa da listagem 2.7 na janela do interpretador Python, como mostra a seção 2.2. O resultado desse programa pode ser visto na listagem 2.8.

► Listagem 2.8 – Exemplo mostrado no interpretador

```
>>> a = 2
```

```
>>> b = 3
```

```
>>> print (a + b)
```

```
5
```

As duas primeiras linhas não enviam nada para a tela; por isso, apenas o resultado da terceira linha é mostrado.

Você pode estar-se perguntando por que criamos duas variáveis, `a` e `b`, para somar dois números? Poderíamos ter obtido o mesmo resultado de diversas formas, como na listagem 2.9.

► Listagem 2.9 – Outra forma de resolver o problema

```
print (2 + 3)
```

Ou mesmo como mostra a listagem 2.10.

► Listagem 2.10 – Outra forma de resolver o problema

```
print (5)
```

Então, por que escolhemos resolver o problema usando variáveis? Primeiramente, para podermos falar de variáveis, mas também para exemplificar uma grande diferença entre resolver um problema no papel e por meio de um computador. Quando estamos resolvendo um problema de matemática no papel, como somar dois números, realizamos diversos cálculos mentalmente e escrevemos parte desse processo no papel, quando necessário. Depois de escrito no papel, mudar os valores não é tão simples. Ao programarmos um computador, estamos transferindo esse cálculo para o computador. Como programar é descrever os passos para a solução do problema, é aconselhável escrevermos programas o mais claramente possível, de forma que possamos alterá-los caso precisemos e, mais importante, que possamos entendê-los mais tarde.

Na listagem 2.9, o problema foi representado como sendo a soma de 2 e 3. Se precisarmos mudar as parcelas dessa soma, teremos de escrever outro programa. Isso também é válido para o primeiro programa, mas observe que, ao utilizarmos variáveis, estamos dando nome aos valores de entrada de nosso problema, aumentando, assim, a facilidade de entendermos o que o programa faz.

Já a solução apresentada na listagem 2.10 não descreve o problema em si. Estamos apenas ordenando ao computador que imprima o número 5 na tela. Não fizemos qualquer registro do que estávamos fazendo, ou de que nosso problema era somar dois números. Isso ficará mais claro no exemplo a seguir. Veja a listagem 2.11.

► Listagem 2.11 – Cálculo de aumento de salário

```
salário = 1500 ①
```

```
aumento = 5 ②
```

```
print (salário + (salário * aumento / 100)) ③
```

Em ① temos uma variável que é chamada `salário`, recebendo o valor 1500. Em ②, outra variável, `aumento`, recebe o valor 5. Finalmente, em ③ descrevemos a fórmula que calculará o valor do novo salário depois de receber um aumento. Teríamos, então, um resultado como o da listagem 2.12.

► Listagem 2.12 – Resultado do aumento de salário no interpretador

```
>>> salário = 1500
>>> aumento = 5
>>> print (salário + (salário * aumento / 100 ))
1575.0
```

O programa da listagem 2.11 pode ser escrito de forma mais direta, utilizando outra fórmula sem variáveis. Veja a alternativa da listagem 2.13.

► Listagem 2.13 – Alternativa para o cálculo de aumento de salário

```
print (1500 + (1500 * 5 / 100))
```

O objetivo desse exemplo é apresentar a diferença entre descrevermos o problema de forma genérica, separando os valores de entrada do cálculo. O resultado é idêntico: a diferença está na clareza da representação de nosso problema. Se mudarmos o valor do salário, na primeira linha da listagem 2.11, obteremos o resultado correto na saída do programa, sem precisar nos preocuparmos novamente com a fórmula do cálculo. Observe também que, se fizermos a mesma coisa no programa da listagem 2.13, teremos que mudar o valor de salário em duas posições diferentes da fórmula, aumentando nossas chances de nos esquecermos de uma delas e, consequentemente, de recebermos um resultado incorreto.

Ao utilizarmos variáveis, podemos referenciar o mesmo valor várias vezes, sem nos esquecer de que podemos utilizar nomes mais significativos que simples `x` ou `y` para aumentar a clareza do programa. Por exemplo, na listagem 2.11, registramos a fórmula para o cálculo do aumento especificando o nome de cada variável, facilitando a leitura e o entendimento.

Se você já utilizou uma planilha eletrônica, como Microsoft Excel ou OpenOffice Calc, o conceito de variável pode ser entendido como as células de uma planilha eletrônica. Você pode escrever as fórmulas de sua planilha sem utilizar outras células, mas teria de reescrevê-las toda vez que os valores mudassem. Assim como as células de uma planilha eletrônica, as variáveis de um programa podem ser utilizadas diversas vezes e em lugares diferentes.

Exercício 2.3 Faça um programa que exiba seu nome na tela.

Exercício 2.4 Escreva um programa que exiba o resultado de $2a \times 3b$, onde a vale 3 e b vale 5.

Exercício 2.5 Modifique o primeiro programa, listagem 2.7, de forma a calcular a soma de três variáveis.

Exercício 2.6 Modifique o programa da listagem 2.11, de forma que ele calcule um aumento de 15% para um salário de R\$ 750.

CAPÍTULO 3

Variáveis e entrada de dados

O capítulo anterior apresentou o conceito de variáveis, mas há mais por descobrir. Já sabemos que variáveis têm nomes que permitem acessar os valores dessas variáveis em outras partes do programa. Neste capítulo, vamos ampliar nosso conhecimento sobre variáveis, estudando novas operações e novos tipos de dados.

3.1 Nomes de variáveis

Em Python, nomes de variáveis devem iniciar obrigatoriamente com uma letra, mas podem conter números e o símbolo sublinha (_). Vejamos exemplos de nomes válidos e inválidos em Python na tabela 3.1.

Tabela 3.1 – Exemplo de nomes válidos e inválidos para variáveis

Nome	Válido	Comentários
a1	Sim	Embora contenha um número, o nome a1 inicia com letra.
velocidade	Sim	Nome formado por letras.
velocidade90	Sim	Nome formado por letras e números, mas iniciado por letra.
salário_médio	Sim	O símbolo sublinha (<u>_</u>) é permitido e facilita a leitura de nomes grandes.
salário médio	Não	Nomes de variáveis não podem conter espaços em branco.
<u>_b</u>	Sim	O sublinha (<u>_</u>) é aceito em nomes de variáveis, mesmo no início.
1a	Não	Nomes de variáveis não podem começar com números.

A versão 3 da linguagem Python permite a utilização de acentos em nomes de variáveis, pois, por padrão, os programas são interpretados utilizando-se um conjunto de caracteres chamado UTF-8 (<http://pt.wikipedia.org/wiki/Utf-8>), capaz de representar praticamente todas as letras dos alfabetos conhecidos.