O texto é uma das formas mais comuns de dados com as quais seus programas lidarão. Você já sabe concatenar dois valores do tipo string usando o operador +, mas poderá fazer muito mais que isso. Você poderá extrair strings parciais a partir de valores do tipo string, adicionar ou remover espaços, fazer conversão para letras minúsculas ou maiúsculas e verificar se as strings estão formatadas corretamente. Você poderá até mesmo criar códigos Python para acessar o clipboard (área de transferência), copiar e colar textos.

Neste capítulo, aprenderemos tudo isso e muito mais. Em seguida, você trabalhará com dois projetos diferentes de programação: um gerenciador simples de senhas e um programa para automatizar a tarefa maçante que é formatar partes de um texto.

## Trabalhando com strings

Vamos dar uma olhada em algumas maneiras pelas quais o Python permite escrever, exibir e acessar strings em seu código.

## Strings literais

Digitar valores de string no código Python é bem simples: elas começam e terminam com aspas simples. Mas como é possível utilizar aspas simples dentro de uma string? Digitar não funcionará, pois o Python achará que a string termina após Alice e que o restante (s cat.') é um código Python inválido. Felizmente, há diversas maneiras de digitar strings.

### Aspas duplas

As strings podem começar e terminar com aspas duplas, assim como ocorre com as aspas simples. Uma vantagem de usar aspas duplas está no fato de a string poder conter um caractere de aspas simples. Digite o seguinte no shell

interativo:

```
>>>
```

Como a string começa com aspas duplas, o Python sabe que o caractere de aspas simples faz parte da string e não marca o seu final. Entretanto, se houver necessidade de usar tanto aspas simples quanto aspas duplas na string será preciso utilizar caracteres de escape.

## Caracteres de escape

Um permite usar caracteres que, de outra maneira, não poderiam ser incluídos em uma string. Um caractere de escape é constituído de uma barra invertida (\) seguida do caractere que você deseja incluir na string. (Apesar de ser constituído de dois caracteres, é comum referenciar esse conjunto como um caractere de escape no singular.) Por exemplo, o caractere de escape para aspas simples é `\'`. Podemos usar isso em uma string que comece e termine com aspas simples. Para ver como os caracteres de escape funcionam, digite o seguinte no shell interativo:

```
>>>
```

O Python sabe que, como o caractere de aspas simples em `Bob\s` tem uma barra invertida, ele não representa as aspas simples usadas para indicar o fim do valor da string. Os caracteres de escape `\'` e `\"` permitem inserir aspas simples e aspas duplas em suas strings, respectivamente.

A tabela 6.1 lista os caracteres de escape que podem ser usados.

| Caractere de escape | Exibido como |
|---|---|
| \' | Aspas simples |
| \" | Aspas duplas |
| \t | Tabulação |
| \n | Quebra ou mudança de linha |
| \\ | Barra invertida |

Digite o seguinte no shell interativo:

```
>>>
```

```
Hello there!
How are you?
I'm doing fine.
```

### Strings puras

Podemos inserir um `r` antes das aspas de início em uma string para transformá-la em uma string pura. Uma (raw string) ignora todos os caracteres de escape e exibe qualquer barra invertida que estiver na string. Por exemplo, digite o seguinte no shell interativo:

```
>>>
That is Carol\'s cat.
```

Pelo fato de ser uma string pura, o Python considera a barra invertida como parte da string, e não como o início de um caractere de escape. As strings puras serão úteis se você estiver digitando valores de string que contenham muitas barras invertidas, por exemplo, as strings usadas para expressões regulares descritas no próximo capítulo.

### Strings de múltiplas linhas com aspas triplas

Embora seja possível usar o caractere de escape `\n` para inserir uma quebra de linha em uma string, geralmente, é mais fácil utilizar strings de múltiplas linhas (multiline). Uma string de múltiplas linhas em Python começa e termina com três aspas simples ou três aspas duplas. Quaisquer aspas, tabulações ou quebras de linha entre as "aspas triplas" serão consideradas parte da string. As regras de indentação do Python para blocos não se aplicam a linhas dentro de uma string de múltiplas linhas.

Abra o editor de arquivo e digite o seguinte:

```
print("Dear Alice,

Eve's cat has been arrested for catnapping, catburglary, and extortion.

Sincerely,
Bob")
```

Salve esse programa como e execute-o. A saída será semelhante a:

```
Dear Alice,

Eve's cat has been arrested for catnapping, cat burglary, and extortion.

Sincerely,
Bob
```

Observe que o caractere único de aspas simples em `Eve's` não precisa ser escapado. Escapar aspas simples e duplas é opcional em strings multinhas. A chamada a `print()` a seguir exibirá um texto idêntico, porém não utiliza uma string de múltiplas linhas:

```
print('Dear Alice,\n\nEve\'s cat has been arrested for catnapping, cat burglary, and
extortion.\n\nSincerely,\nBob')
```

### Comentários de múltiplas linhas

Enquanto o caractere de sustenido (`#`) marca o início de um comentário que inclui o restante da linha, uma string de múltiplas linhas geralmente é usada para comentários que ocupem várias linhas. O código a seguir é perfeitamente válido em Python:

```
"""Este é um programa Python para testes.
Criado por Al Sweigart al@inventwithpython.com

Esse programa foi criado para Python 3, e não para Python 2.
"""


def spam():
    """Este é um comentário de múltiplas linhas para ajudar a
    explicar o que a função spam() faz."""
    print('Hello!')
```

## Indexação e slicing de strings

As strings usam índices e slices (fatias) do mesmo modo que as listas. Podemos pensar na string `'Hello world!'` como uma lista e em cada caractere da string como um item com um índice correspondente.

| ' | H | e | l | l | o | | w | o | r | l | d | ! | ' |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | | |

O espaço e o ponto de exclamação são incluídos na contagem de caracteres portanto 'Hello world!' tem 12 caracteres de tamanho, de H no índice 0 a ! no índice 11.

Digite o seguinte no shell interativo:

```
>>>
>>>
'H'
>>>
'o'
>>>
'!'
>>>
'Hello'
>>>
'Hello'
>>>
'world!'
```

Se um índice for especificado, você obterá o caractere nessa posição da string. Se um intervalo for especificado de um índice a outro, o índice inicial será incluído, mas não o índice final. É por isso que, se spam for 'Hello world!', spam[0:5] será 'Hello'. A substring obtida a partir de spam[0:5] inclui tudo de spam[0] a spam[4], deixando de fora o espaço no índice 5.

Observe que o slicing de uma string não modifica a string original. Podemos capturar um slice de uma variável em uma variável diferente. Experimente digitar o seguinte no shell interativo:

```
>>>
>>>
>>>
'Hello'
```

Ao fazer o slicing e armazenar a substring resultante em outra variável, podemos ter tanto a string completa quanto a substring à mão para termos um acesso rápido e fácil.

## Operadores in e not in com strings

Os operadores in e not in podem ser usados com strings, assim como em valores de lista. Uma expressão com duas strings unidas por meio de in ou de not in será avaliada como um booleano True ou False. Digite o seguinte no shell interativo:

```
>>>
True
>>>
True
>>>
False
>>>
True
>>>
False
```

Essas expressões testam se a primeira string (a string exata, considerando as diferenças entre letras maiúsculas e minúsculas) pode ser encontrada na segunda string.

## Métodos úteis de string

Vários métodos de string analisam strings ou criam valores transformados de strings. Esta seção descreve os métodos que utilizaremos com mais frequência.

## Métodos de string upper(), lower(), isupper() e islower()

Os métodos de string upper() e lower() retornam uma nova string em que todas as letras da string original foram convertidas para letras maiúsculas ou minúsculas, respectivamente. Os caracteres que não correspondem a letras permanecem inalterados na string. Digite o seguinte no shell interativo:

```
>>>
>>>
>>>
'HELLO WORLD!'
>>>
>>>
```

helloworld!'

Observe que esses métodos não alteram a string em si, mas retornam novos valores de string. Se quiser alterar a string original, será necessário chamar upper() ou lower() na string e, em seguida, atribuir a nova string à variável em que a original estava armazenada. É por isso que devemos usar spam = spam.upper() para alterar a string em spam no lugar de utilizar simplesmente spam.upper(). (É como se uma variável eggs contivesse o valor 10. Escrever eggs + 3 não altera o valor de eggs, porém eggs = eggs + 3 o modifica.)

Os métodos upper() e lower() serão úteis caso seja necessário fazer uma comparação sem levar em conta a diferença entre letras maiúsculas e minúsculas. As strings 'great' e 'GREat' não são iguais. No entanto, no pequeno programa a seguir, não importa se o usuário digitar Great, GREAT ou grEAT pois a string será inicialmente convertida para letras minúsculas.

```
print('How are you?')
feeling = input()
if feeling.lower() == 'great':
    print('I feel great too.')
else:
    print('I hope the rest of your day is good.')
```

Ao executar esse programa, a pergunta será exibida, e fornecer uma variação de great, por exemplo, GREat, ainda resultará na saída I feel great too. Adicionar um código ao seu programa para tratar variações ou erros nos dados de entrada do usuário, por exemplo, um uso inconsistente de letras maiúsculas, fará seus programas serem mais simples de usar e os deixará menos suscetíveis a falhas.

```
How are you?

I feel great too.
```

Os métodos isupper() e islower() retornarão um valor booleano True se a string tiver pelo menos uma letra e todas as letras forem maiúsculas ou minúsculas, respectivamente. Caso contrário, o método retornará False. Digite o seguinte no shell interativo e observe o que cada chamada de método retorna:

```
>>>
```

```
>>>
False
>>>
False
>>>
True
>>>
True
>>>
False
>>>
False
```

Como os métodos de string upper() e lower() retornam strings, também podemos chamar os métodos de string valores de string retornados. As expressões que fazem isso se parecerão com uma cadeia de chamadas de métodos. Digite o seguinte no shell interativo:

```
>>>
'HELLO'
>>>
'hello'
>>>
'HELLO'
>>>
'hello'
>>>
True
```

## Métodos de string isX

Juntamente com islower() e isupper(), há diversos métodos de string cujos nomes começam com a palavra . Esses métodos retornam um valor booleano que descreve a natureza da string. Eis alguns métodos de string is comuns:

- isalpha() retornará True se a string for constituída somente de letras e não estiver vazia.
- isalnum() retornará True se a string for constituída somente de letras e números e não estiver vazia.

- isdecimal() retornará True se a string for constituída somente de caracteres numéricos e não estiver vazia.
- isspace() retornará True se a string for constituída somente de espaços, tabulações e quebras de linha e não estiver vazia.
- istitle() retornará True se a string for constituída somente de palavras que comecem com uma letra maiúscula seguida somente de letras minúsculas.

Digite o seguinte no shell interativo:

```
>>>
True
>>>
False
>>>
True
>>>
True
>>>
True
>>>
True
>>>
True
>>>
True
>>>
False
>>>
False
```

Os métodos de string is são úteis para validar dados de entrada do usuário. Por exemplo, o programa a seguir pergunta repetidamente aos usuários a idade e pede uma senha até que dados de entrada válidos sejam fornecidos. Abra uma nova janela no editor de arquivo e insira o programa a seguir, salvando-o como :

```
while True:
    print('Enter your age:')
    age = input()
    if age.isdecimal():
```

```
        break
    print('Please enter a number for your age.')

while True:
    print('Select a new password (letters and numbers only):')
    password = input()
    if password.isalnum():
        break
    print('Passwords can only have letters and numbers.')
```

No primeiro loop `while`, perguntamos a idade ao usuário e armazenamos sua entrada em `age`. Se `age` for um valor válido (decimal), sairemos desse primeiro loop `while` e prosseguiremos para o segundo, que pede uma senha. Caso contrário, informamos o usuário que ele deve fornecer um número e perguntamos sua idade novamente. No segundo loop `while`, devemos pedir uma senha, armazenar o dado de entrada do usuário em `password` e sair do loop se a entrada for alfanumérica. Se não for, não ficaremos satisfeitos sendo assim, dizemos ao usuário que a senha deve ser alfanumérica e novamente, pedimos que uma senha seja fornecida.

Ao ser executado, a saída desse programa será semelhante a:

```
Enter your age:

Please enter a number for your age.
Enter your age:

Select a new password (letters and numbers only):

Passwords can only have letters and numbers.
Select a new password (letters and numbers only):
```

Se chamarmos `isdecimal()` e `isalnum()` em variáveis, poderemos testar se os valores armazenados nessas variáveis são decimais ou não, ou se são alfanuméricos ou não. Nesse caso, esses testes nos ajudam a rejeitar a entrada `forty two` e a aceitar `42`, além de rejeitar `secr3t!` e aceitar `secr3t`.

## Métodos de string `startswith()` e `endswith()`

Os métodos startswith() e endswith() retornarão True se o valor de string com o qual forem chamados começar ou terminar (respectivamente) com a string passada para o método; do contrário, retornarão False. Digite o seguinte no shell interativo:

```
>>>
True
>>>
True
>>>
False
>>>
False
>>>
True
>>>
True
```

Esses métodos serão alternativas convenientes ao operador == de igualdade caso seja preciso verificar se apenas a primeira ou a última parte da string, e não a string completa, é igual a outra string.

## Métodos de string join() e split()

O método join() é útil quando temos uma lista de strings que devem ser unidas em um único valor de string. O método join() é chamado em uma string, recebe uma lista de strings e retorna uma string. A string retornada corresponde à concatenação de todas as strings da lista passada para o método. Por exemplo, digite o seguinte no shell interativo:

```
>>>
'cats, rats, bats'
>>>
'My name is Simon'
>>>
'MyABCnameABCisABCSimon'
```

Observe que a string em que join() é chamada é inserida entre cada string do argumento de lista. Por exemplo, quando join(['cats', 'rats', 'bats']) é chamada na string ', ', a string retornada é 'cats, rats, bats'.

Lembre-se de que join() é chamado em um valor de string e recebe um valo de lista. (É fácil chamar acidentalmente de modo invertido.) O método split( faz o inverso: é chamado em um valor de string e retorna uma lista de strings D igite o seguinte no shell interativo:

```
>>>
[My', name', 'is', Simon']
```

Por padrão, a string My name is Simon' é separada sempre que caracteres em branco, como caracteres de espaço, tabulação ou quebra de linha, forem encontrados. Esses caracteres de espaço em branco não são incluídos nas strings da lista retornada. Podemos passar uma string delimitadora ao método split() para especificar uma string diferente em relação à qual a separação será feita. Por exemplo, digite o seguinte no shell interativo:

```
>>>
[My', name', 'is', Simon']
>>>
[My na', 'e is Si', 'on']
```

Um uso comum de split() está em dividir uma string de múltiplas linhas nos caracteres de quebra de linha. D igite o seguinte no shell interativo:

```
>>>
```

```
>>>
[Dear Alice,', How have you been? I am fine.', There is a container in the
fridge', 'that is labeled 'Milk Experiment'.', '', Please do not drink it.',
Sincerely,', Bob']
```

Passar o argumento '\n' a split() permite separar a string com múltiplas linhas armazenada em spam nas quebras de linha e retornar uma lista em que cada item corresponda a uma linha da string.

## Justificando texto com rjust(), ljust() e center()

Os métodos de string rjust() e ljust() retornam uma versão preenchida da string em que são chamados, com espaços inseridos para justificar o texto. O primeiro argumento de ambos os métodos é um inteiro referente ao tamanho da string justificada. Digite o seguinte no shell interativo:

```
>>>
'  Hello'
>>>
'          Hello'
>>>
'     HelloWorld'
>>>
Hello    '
```

'Hello'.rjust(10) diz que queremos justificar 'Hello' à direita em uma string de tamanho total igual a 10. 'Hello' tem cinco caracteres, portanto cinco espaços serão acrescentados à sua esquerda, resultando em uma string de dez caracteres, com 'Hello' justificado à direita.

Um segundo argumento opcional de rjust() e ljust() especifica um caractere de preenchimento que não seja um caractere de espaço. Digite o seguinte no shell interativo:

```
>>>
'**********Hello'
>>>
'Hello----------'
```

O método de string center() funciona como ljust() e rjust(), porém centraliza o texto em vez de justificá-lo à esquerda ou à direita. Digite o seguinte no shell interativo:

```
>>>
'    Hello      '
>>>
'======Hello========'
```

Esses métodos serão especialmente úteis quando for necessário exibir dados tabulares que tiverem o espaçamento correto. Abra uma nova janela no editor de arquivo e insira o código a seguir, salvando-o como :

```
def printPicnic(itemsDict, leftWidth, rightWidth):
    print('PICNIC ITEMS'.center(leftWidth + rightWidth, '-'))
    for k, v in itemsDict.items():
        print(k.ljust(leftWidth, '.') + str(v).rjust(rightWidth))

picnicItems = {'sandwiches': 4, 'apples': 12, 'cups': 4, 'cookies': 8000}
printPicnic(picnicItems, 12, 5)
printPicnic(picnicItems, 20, 6)
```

Nesse programa, definimos um método `printPicnic()` que recebe um dicionário contendo informações e usa `center()`, `ljust()` e `rjust()` para exibir essas informações em um formato organizado e alinhado de tabela.

O dicionário que passaremos a `printPicnic()` é `picnicItems`. Em `picnicItems` temos 4 sanduíches (`sandwiches`), 12 maçãs (`apples`), 4 copos (`cups`) e 8.000 biscoitos (`cookies`). Queremos organizar essas informações em duas colunas com o nome do item à esquerda e a quantidade à direita.

Para isso, devemos decidir qual será a largura que queremos que as colunas à esquerda e à direita tenham. Juntamente com o nosso dicionário passaremos esses valores a `printPicnic()`.

`printPicnic()` recebe um dicionário, um `leftWidth` para a coluna esquerda de uma tabela e um `rightWidth` para a coluna direita. A função exibe um título `PICNIC ITEMS` centralizado na parte superior da tabela. Em seguida, um loop percorre o dicionário exibindo cada par chave-valor em uma linha, com a chave justificada à esquerda e preenchida com pontos e o valor justificado à direita, preenchido com espaços.

Após definir `printPicnic()`, criamos o dicionário `picnicItems` e chamamos `printPicnic()` duas vezes, passando larguras diferentes para as colunas da esquerda e da direita da tabela.

Ao executar esse programa, os itens do piquenique serão exibidos duas vezes. Na primeira vez, a coluna da esquerda terá 12 caracteres de largura, e a coluna da direita, 5 caracteres de largura. Na segunda vez, as colunas terão 20 e 6 caracteres de largura, respectivamente.

```
---PICNIC ITEMS--
sandwiches..  4
apples......  12
```

```
cups.....  4
cookies.....8000
-------PICNIC ITEMS-------
sandwiches.....  4
apples.....  12
cups.....  4
cookies..... 8000
```

Usar rjust(), ljust() e center() permite garantir que as strings estejam elegantemente alinhadas, mesmo que você não saiba ao certo quantos caracteres têm suas strings.

## Removendo espaços em branco com strip(), rstrip() e lstrip()

Às vezes, você pode querer remover caracteres de espaços em branco (espaço, tabulação e quebra de linha) do lado esquerdo, do lado direito ou de ambos os lados de uma string. O método de string strip() retornará uma nova string sem caracteres de espaços em branco no início ou no fim. Os métodos lstrip() e rstrip() removerão caracteres de espaços em branco das extremidades esquerda e direita, respectivamente. Digite o seguinte no shell interativo:

```
>>>
>>>
'HelloWorld'
>>>
'HelloWorld   '
>>>
'  HelloWorld'
```

Opcionalmente, um argumento do tipo string especificará quais caracteres deverão ser removidos das extremidades. Digite o seguinte no shell interativo:

```
>>>
>>>
'BaconSpamEggs'
```

Passar o argumento 'ampS' a strip() lhe dirá para remover as ocorrências de a m, p e da letra S maiúscula das extremidades da string armazenada em spam. A

ordem dos caracteres na string passada para strip() não importa: strip('ampS' fará o mesmo que strip('mapS') ou strip('Spam').

## Copiando e colando strings com o módulo pyperclip

O módulo pyperclip tem funções copy() e paste() capazes de enviar e recebe texto do clipboard (área de transferência) de seu computador. Enviar a saída de seu programa para o clipboard facilitará colá-lo em um email, um processador de texto ou em outro software.

O pyperclip não vem com o Python. Para instalá-lo, siga as instruções para instalação de módulos de terceiros no apêndice A. Após instalar o módulo pyperclip, digite o seguinte no shell interativo:

```
>>>
>>>
>>>
'Hello world!'
```

É claro que, se algo fora de seu programa alterar o conteúdo do clipboard, a função paste() retornará essa informação. Por exemplo, se eu copiar esta frase para o clipboard e, em seguida, chamar paste(), o resultado terá a seguinte aparência:

```
>>>
Por exemplo, se eu copiar esta frase para o clipboard e, em seguida, chamar paste(), o
resultado terá a seguinte aparência:'
```

Até agora, executamos os scripts Python usando o shell interativo e o editor de arquivo no IDLE. Entretanto você não vai querer ter o inconveniente de abrir o IDLE e o script Python sempre que quiser executar um script. Felizmente, há atalhos que podemos configurar para facilitar a execução dos scripts Python. Os passos são um pouco diferentes para Windows, OS X e Linux, porém cada um deles está descrito no apêndice B. Consulte o apêndice B para saber como executar seus scripts Python de forma conveniente e poder passar argumentos de linha de comando a eles. (Você não poderá passar argumentos de linha de comando a seus programas usando o IDLE.)

## Projeto: Repositório de senhas

É provável que você tenha contas em vários sites diferentes. Usar a mesma senha em todos eles é um péssimo hábito porque, se um desses sites tiver uma falha de segurança, os hackers saberão a senha de todas as suas demais contas. É melhor usar um software gerenciador de senhas em seu computador que utilize uma senha principal para desbloquear esse gerenciador de senhas. Então você poderá copiar qualquer senha de conta para o clipboard e colá-la no campo de senha do site.

O programa gerenciador de senhas que criaremos nesse exemplo não é seguro, porém oferece uma demonstração básica de como esses programas funcionam.

> Esse é o primeiro "projeto do capítulo" deste livro. A partir de agora, cada capítulo terá projetos que demonstrarão conceitos discutidos nesse capítulo. Os projetos são criados de modo a levarem você de uma janela em branco do editor de arquivo até um programa completo e funcional. Assim como nos exemplos com o shell interativo, não leia simplesmente as seções de projeto – execute-os em seu computador!

## Passo 1: Design do programa e estruturas de dados

Queremos executar esse programa com um argumento de linha de comando correspondente ao nome da conta – por exemplo, ou . A senha dessa conta será copiada para o clipboard para que o usuário possa colá-la em um campo de Senha. Dessa maneira, o usuário poderá ter senhas longas e complexas sem a necessidade de memorizá-las.

Abra uma nova janela no editor de arquivo e salve o programa como O programa deve começar com uma linha contendo #! ( ) – veja o apêndice B –, e você também deve escrever um comentário que descreva brevemente o programa. Como queremos associar o nome de cada conta à sua senha, podemos armazenar essas informações como strings em um dicionário. O dicionário será a estrutura de dados que organizará os dados de suas contas e senhas. Faça seu programa ter o seguinte aspecto:

```
#! python3
# pw.py – Um programa para repositório de senhas que não é seguro.

PASSWORDS = {'email': 'F7minlBDDuvMJuxESSKHFhTxFtjVB6',
             'blog': 'VmALvQyKAxiVH5G8v01if1MLZF3sdt',
             'luggage': '12345'}
```

## Passo 2: Tratar argumentos da linha de comando

Os argumentos da linha de comando serão armazenados na variável sys.argv (Veja o apêndice B para obter mais informações sobre como usar argumentos de linha de comando em seus programas.) O primeiro item da lista sys.argv sempre será uma string contendo o nome do arquivo do programa ('pw.py'), e o segundo item deverá ser o primeiro argumento da linha de comando. Nesse programa, esse argumento será o nome da conta cuja senha você deseja obter. Como o argumento de linha de comando é obrigatório, você deve exibir uma mensagem de uso ao usuário caso ele se esqueça de adicioná-lo (ou seja, se a lista sys.argv contiver menos de dois valores). Faça seu programa ter o seguinte aspecto:

```
#! python3
# pw.py – Um programa para repositório de senha que não é seguro.

PASSWORDS = {'email': 'F7minlBDDuvMJuxESSKHFhTxFtjVB6',
             'blog': 'VmALvQyKAxiVH5G8v01if1MLZF3sdt',
             'luggage': '12345'}
```

o primeiro argumento da linha de comando é o nome da conta

## Passo 3: Copiar a senha correta

Agora que o nome da conta está armazenado como uma string na variáve account, devemos verificar se ele existe no dicionário PASSWORDS como uma chave. Em caso afirmativo, devemos copiar o valor da chave para o clipboard

usando pyperclip.copy(). (Como o módulo pyperclip está sendo usado, é necessário importá-lo.) Observe que não realmente da variáve account; poderíamos simplesmente usar sys.argv[1] em todos os lugares em que account é usado nesse programa. Porém uma variável chamada account é muito mais legível do que algo enigmático como sys.argv[1].

Faça seu programa ter o seguinte aspecto:

```
#! python3
# pw.py – Um programa para repositório de senhas que não é seguro.

PASSWORDS = {'email': 'F7minlBDDuvMJuxESSKHFhTxFtjVB6',
             'blog': 'VmALvQyKAxiVH5G8v01if1MLZF3sdt',
             'luggage': '12345'}

import sys,
if len(sys.argv) < 2:
    print('Usage: py pw.py [account] - copy account password')
    sys.exit()

account = sys.argv[1]  # o primeiro argumento da linha de comando é o nome da conta
```

Esse novo código procura o nome da conta no dicionário PASSWORDS. Se o nome da conta for uma chave no dicionário, leremos o valor correspondente a essa chave, copiaremos esse valor para o clipboard e exibiremos uma mensagem informando que o valor foi copiado. Caso contrário, exibiremos uma mensagem informando que não há nenhuma conta com esse nome.

Esse é o script completo. Ao usar as instruções do apêndice B para iniciar programas com linha de comando facilmente, você terá uma maneira rápida de copiar as senhas de suas contas para o clipboard. Será necessário modificar o valor do dicionário PASSWORDS no código-fonte sempre que você quiser atualizar o programa com uma nova senha.

É claro que, provavelmente, você não vai querer manter todas as suas

senhas em um local em que qualquer pessoa poderia copiá-las facilmente. No entanto esse programa poderá ser modificado e usado para copiar textos normais rapidamente para o clipboard. Suponha que você esteja enviando diversos emails que têm muitos dos mesmos parágrafos em comum. Você poderia colocar cada parágrafo como um valor no dicionário PASSWORDS (é provável que você queira renomear o dicionário a essa altura); desse modo você terá uma maneira de selecionar e copiar rapidamente uma das várias partes de texto padrão para o clipboard.

No Windows, um arquivo batch poderá ser criado para executar esse programa com a janela Run de WIN-R. (Para saber mais sobre arquivos batch consulte o apêndice B.) Digite o seguinte no editor de arquivo e salve com o na pasta :

```
@py.exe C:\Python34\pw.py %*
@pause
```

Com esse arquivo batch criado, executar o programa de senhas protegidas no Windows é somente uma questão de pressionar WIN-R e digitar pw < >.

## Projeto: Adicionando marcadores na marcação da Wiki

Ao editar um artigo na Wikipedia, podemos criar uma lista com marcações (bullets) ao inserir cada item da lista em sua própria linha e inserindo um asterisco na frente. Porém suponha que você tenha uma lista realmente extensa em que você queira acrescentar marcadores. Você poderia simplesmente digitar esses asteriscos no início de cada linha, uma a uma, ou poderia automatizar essa tarefa com um pequeno script Python.

O script obterá o texto do clipboard, adicionará um asterisco e um espaço no início de cada linha e, em seguida, colará esse novo texto no clipboard. Por exemplo, se eu copiar o texto a seguir [do artigo "Lists of Lists of Lists" (Listas de listas de listas) da Wikipedia] para o clipboard:

```
Lists of animals
Lists of aquarium life
Lists of biologists by author abbreviation
Lists of cultivars
```

e depois executar o programa , o clipboard conterá o seguinte:

```
* Lists of animals
* Lists of aquarium life
* Lists of biologists by author abbreviation
* Lists of cultivars
```

Esse texto contendo um asterisco como prefixo está pronto para ser colado em um artigo da Wikipedia como uma lista com marcadores.

## Passo 1: Copiar e colar no clipboard

Queremos que o programa faça o seguinte:

1. Obtenha o texto do clipboard.
2. Faça algo com ele.
3. Copie o novo texto para o clipboard.

O segundo passo é um pouco complicado, porém os passos 1 e 3 são bem simples: eles envolvem somente as funções pyperclip.copy() e pyperclip.paste(). Por enquanto, vamos apenas criar a parte do programa que trata os passos 1 e 3. Digite o seguinte, salvando o programa como :

```
#! python3
# bulletPointAdder.py – Acrescenta marcadores da Wikipedia no início
# de cada linha de texto do clipboard.

import pyperclip
text = pyperclip.paste()

# TODO: Separa as linhas e acrescenta asteriscos.
pyperclip.copy(text)
```

O comentário TODO é um lembrete de que você deve completar essa parte do programa em algum momento. O próximo passo consiste em realmente implementar essa parte do programa.

## Passo 2: Separar as linhas de texto e acrescentar o asterisco

A chamada a pyperclip.paste() retorna todo o texto que está no clipboard na forma de uma string extensa. Se usarmos o exemplo de "List of Lists o Lists", a string armazenada em text terá o seguinte aspecto:

```
'Lists of animals\nLists of aquarium life\nLists of biologists by author
abbreviation\nLists of cultivars'
```

Os caracteres \n de quebra de linha nessa string fazem com que ela seja apresentada em várias linhas quando for exibida ou copiada do clipboard. Há várias "linhas" nesse único valor de string. Você deve acrescentar um asterisco no início de cada uma dessas linhas.

Poderíamos criar um código que procurasse todos os caracteres \n de quebra de linha na string e, em seguida, acrescentasse o asterisco imediatamente depois deles. Todavia será mais fácil usar o método split() para retornar uma lista de strings, uma para cada linha da string original, e então acrescentar o asterisco na frente de cada string da lista.

Faça seu programa ter o seguinte aspecto:

```
#! python3
# bulletPointAdder.py – Acrescenta marcadores da Wikipedia no início
# de cada linha de texto do clipboard.

import pyperclip
text = pyperclip.paste()

# Separa as linhas e acrescenta os asteriscos

# percorre todos os índices da lista "lines" em um loop
# acrescenta um asterisco em cada string da lista "lines"

pyperclip.copy(text)
```

Separamos o texto nas quebras de linha para obter uma lista em que cada item corresponda a uma linha de texto. Armazenamos a lista em lines e percorremos seus itens usando um loop. Para cada linha, acrescentamos um asterisco e um espaço no início dessa linha. Agora cada string em lines começa com um asterisco.

## Passo 3: Juntar as linhas modificadas

A lista `lines` agora contém as linhas modificadas iniciadas com asteriscos. Porém `pyperclip.copy()` está esperando um único valor de string, e não uma lista de valores de string. Para compor esse valor único de string, passe `lines` ao método `join()` a fim de obter uma única string resultante da junção das strings da lista. Faça seu programa ter o seguinte aspecto:

```
#! python3
# bulletPointAdder.py – Acrescenta marcadores da Wikipedia no início
# de cada linha de texto do clipboard.

import pyperclip
text = pyperclip.paste()

# Separa as linhas e acrescenta os asteriscos.
lines = text.split('\n')
for i in range(len(lines)):  # percorre todos os índices da lista "lines" em um loop
    lines[i] = '* ' + lines[i] # acrescenta um asterisco em cada string da lista "lines"

pyperclip.copy(text)
```

Quando é executado, esse programa substitui o texto do clipboard por um texto que contém asteriscos no início de cada linha. Agora o programa está completo e você pode tentar executá-lo com um texto copiado no clipboard.

Mesmo que não precise automatizar essa tarefa específica, talvez você queira automatizar outro tipo de manipulação de textos, por exemplo remover espaços dos finais das linhas ou converter textos para letras maiúsculas ou minúsculas. Independentemente do que você precisar, o clipboard poderá ser usado para entrada e saída de dados.

## Resumo

O texto é uma forma comum de dados e o Python oferece diversos métodos úteis de string para processar um texto armazenado em valores de string. Você utilizará indexação, slicing e métodos de string em quase todos os programas Python que criar.

Os programas que você está criando agora não parecem ser muito

sofisticados – eles não têm interfaces gráficas de usuário com imagens e textos coloridos. Até agora, exibimos texto com `print()` e permitimos que o usuário fornecesse texto com `input()`. No entanto o usuário pode fornecer grandes quantidades de texto rapidamente por meio do clipboard. Essa capacidade oferece uma opção conveniente para escrever programas que manipulem grandes volumes de texto. Esses programas baseados em texto podem não ter janelas nem imagens sofisticadas, porém podem realizar diversas tarefas convenientes rapidamente.

Outra maneira de manipular grandes quantidades de texto consiste em ler e escrever em arquivos diretamente no disco rígido. Aprenderemos a fazer isso com o Python no próximo capítulo.

## Exercícios práticos

1. O que são caracteres de escape?
2. O que os caracteres de escape `\n` e `\t` representam?
3. Como podemos inserir um caractere `\` de barra invertida em uma string?
4. O valor de string `"How's Moving Castle"` é uma string válida. Por que não há problema no fato de o caractere único de aspas simples na palavra `How's` não estar escapado?
5. Se não quiser colocar `\n` em sua string, como você poderá escrever uma string contendo quebras de linha?
6. Para que valores as expressões a seguir são avaliadas?
   - `'Hello world!'[1]`
   - `'Hello world!'[0:5]`
   - `'Hello world!'[:5]`
   - `'Hello world!'[3:]`
7. Para que valores as expressões a seguir são avaliadas?
   - `'Hello'.upper()`
   - `'Hello'.upper().isupper()`
   - `'Hello'.upper().lower()`
8. Para que valores as expressões a seguir são avaliadas?

- 'Remember, remember, the fifth of November.'.split()
- '-'.join('There can be only one.'.split())

9. Quais métodos de string podem ser usados para justificar uma string à direita, à esquerda e para centralizá-la?
10. Como podemos remover caracteres de espaços em branco no início e no fim de uma string?

## Projeto prático

Para exercitar, escreva um programa que execute a seguinte tarefa.

## Exibição de tabela

Crie uma função chamada printTable() que receba uma lista de listas de strings e a exiba em uma tabela bem organizada, com cada coluna justificada à direita. Suponha que todas as listas internas contenham o mesmo número de strings. Por exemplo, o valor poderá ter o seguinte aspecto:

```
tableData = [['apples', 'oranges', 'cherries', 'banana'],
             ['Alice', 'Bob', 'Carol', 'David'],
             ['dogs', 'cats', 'moose', 'goose']]
```

Sua função printTable() exibirá o seguinte:

```
  apples Alice  dogs
 oranges   Bob  cats
cherries Carol moose
  banana David goose
```

Dica: seu código inicialmente deverá localizar a string mais longa em cada uma das listas internas para que a coluna toda tenha largura suficiente para que todas as strings possam ser inseridas. Você pode armazenar a largura máxima de cada coluna como uma lista de inteiros. A função printTable() pode começar com colWidths = [0] * len(tableData), que criará uma lista contendo o mesmo número de valores 0 que o número de listas internas em tableData. Dessa maneira, colWidths[0] poderá armazenar a largura da string mais longa de tableData[0], colWidths[1] poderá armazenar a largura da string mais longa de tableData[1] e assim por diante. Você poderá então identificar o maior valor na

lista colWidths e descobrir a largura na forma de um inteiro a ser passada para
o método de string rjust().