

# CAPÍTULO 12

## Próximos passos

A programação é um longo caminho que você começou a percorrer. Neste livro, mostramos as técnicas básicas para que você continue seus estudos. A programação de computadores é uma área de conhecimento muito grande e rica. Este livro apresentou uma introdução à programação utilizando a linguagem Python. Os próximos passos dependem de sua área de interesse. Você não precisa estudar todos os tópicos ou mesmo seguir a ordem em que são apresentados neste capítulo. Vamos listar alguns tópicos.

### 12.1 Programação funcional

A programação funcional é um paradigma de programação diferente do que aprendemos aqui com Python. Embora Python tenha recursos de uma linguagem funcional, permitindo misturar os dois paradigmas em um só programa. Para enriquecer sua experiência em programação, estudar uma linguagem puramente funcional, como LISP, Scheme ou F#, é realmente interessante, sendo uma área que deve popularizar-se cada vez mais. Estudar uma nova forma de organizar seus programas ajudará a entender melhor vários conceitos que você já aprendeu e a modificar a forma que você encara a programação. Atualmente, diversas linguagens são chamadas de híbridas ou multiparadigmas, pois misturam ou possibilitam a utilização de recursos de programação funcional e imperativa. Python é uma delas. Além disso, algumas propriedades da programação funcional são excelentes para resolver ou abordar problemas de concorrência, cada vez mais comuns hoje em dia.

O livro SICP – Structure and Interpretation of Computer Programs, de Harold Abelson e Gerald Jay Sussman, pode ser lido on-line no site <http://mitpress.mit.edu/sicp/full-text/book/book.html>. O livro é usado em diversos cursos de computação e é considerado um dos melhores disponíveis.

### 12.2 Algoritmos

Até agora utilizamos o básico da programação para resolver nossos problemas. À medida que você for ganhando experiência, terá que estudar novos algoritmos e entender algoritmos que já utiliza hoje, sem perceber. Um exemplo é a compreensão de técnicas de espalhamento (*hashes*), que utilizamos em Python com dicionários, ou como as listas funcionam internamente. Quanto maior seus programas, mais importante entender como cada algoritmo funciona, conhecer suas aplicações e limitações.

A linguagem C também é recomendada para o estudo de algoritmos e estruturas de dados. Ao contrário de Python, em C você está sozinho e ela vem quase sem baterias. A vantagem de estudar e aprender a programar em C é conhecer os detalhes de cada implementação e controlar cada passo de seus programas.

Um livro recomendado para estudos aprofundados é *Algoritmos: teoria e prática*, de Thomas H. Cormen. Embora a leitura não seja das mais fáceis, seu conteúdo é esclarecedor. Não leia esse livro antes de programar em pelo menos duas linguagens de programação. Sua leitura é recomendada para pessoas que desejem realmente se aprofundar no assunto, normalmente cursando ciência da computação.

### 12.3 Jogos

A programação de jogos é uma das mais áreas mais recompensadoras e inspiradoras. Além disso, o que você aprende programando jogos serve para solucionar diversos outros problemas do dia a dia. Python apresenta diversas bibliotecas externas prontas para a criação de jogos em 2D ou 3D. Essas bibliotecas incluem a manipulação de imagens, sons, controle do tempo e mesmo de arquivos.

Pygame (<http://www.pygame.org>): biblioteca multimídia fácil de aprender. Permite a manipulação de superfícies de desenho e a criação de jogos com gráficos em 2D e som. Se você tiver curiosidade de saber como criar jogos simples, visite o projeto Invasores (<http://www.sourceforge.net/projects/invasores>). O projeto foi inteiramente desenvolvido em Python e é open source. Você pode baixar e estudar o código-fonte, fazer modificações e testar. Tudo escrito em Python.

Panda 3D (<http://www.panda3d.org/>): biblioteca gráfica, desenvolvida pelos estúdios Disney, open source e supercompleta. Com Panda 3D você pode criar jogos profissionais, e o melhor de tudo: usando Python. A biblioteca é poderosa e complexa, tendo sido utilizada em jogos comerciais. Não se esqueça de revisar álgebra linear e estudar gráficos tridimensionais antes de se aventurar na documentação (em inglês).

PyOgre (<http://www.ogre3d.org/tikiwiki/PyOgre>): outra biblioteca ou kit de desenvolvimento de jogos 3D. A PyOgre é também poderosa e complexa.

Se você começar a estudar essas bibliotecas, encare essa tarefa como um projeto de pelo menos um ano. Essas bibliotecas são grandes e permitem a criação de aplicativos multimídia, além de jogos, claro. O ideal é realizar esse estudo em grupo: visite o site <http://www.unidev.com.br>.

## 12.4 Orientação a objetos

No capítulo 10, apenas introduzimos a programação orientada a objetos. Como já dito, esse é um assunto realmente extenso que vale a pena ser estudado. Python é uma linguagem que implementa quase todos os conceitos de orientação a objeto, mas você deve aprender a dominá-los. Estude poliformismo, interfaces e padrões de projeto (*design patterns*).

Um livro que ajuda a decifrar a orientação a objetos é *Head First Design Patterns*, de Eric T. Freeman, Elisabeth Robson, Bert Bates e Kathy Sierra. O livro apresenta os conceitos básicos e os principais padrões de projeto de forma simples e diversificada, com diversas histórias e gráficos.

Um livro clássico sobre padrões de projeto é *Design Patterns: Elements of Reusable Object-Oriented Software*, de Erich Gamma, Richard Helm, Ralph Johnson e John Vlissides. O livro apresenta um conteúdo denso em formato de referência. Não é indicado para uma introdução a padrões de projeto, mas você deve lê-lo um dia.

## 12.5 Banco de dados

Embora tenhamos abordado banco de dados no capítulo 11, o assunto é extenso.

Depois de ler a documentação do módulo `sqlite3`, você pode aventurar-se com bibliotecas mais modernas que isolam seus programas do banco de dados em si, tornando a manipulação de seus dados quase transparente. Uma dica é a biblioteca SQLAlchemy (<http://www.sqlalchemy.org/>).

## 12.6 Sistemas web

Um dos assuntos mais quentes hoje em dia é a programação de sistemas Web. Esses sistemas também podem ser escritos em Python. No entanto, a comunicação de um programa com o servidor de páginas web é repleta de detalhes e torna seus

programas repetitivos e trabalhosos de escrever. Se você deseja aprender a escrever, ou se interessa por sistemas web, estude o Django (<http://www.djangoproject.com/>). O Django estrutura seus sistemas para que se tornem fáceis de escrever e modificar. Além disso, as bibliotecas do Django já realizam as tarefas repetitivas de comunicação com o servidor web e mesmo com o banco de dados.

## 12.7 Outras bibliotecas Python

Além de Python vir com “baterias incluídas”, você encontra na Internet diversas bibliotecas ou mais baterias para fazer praticamente tudo. Um bom site para procurar um módulo é o Python Package Index - PyPI (<http://pypi.python.org/pypi>).

Para aplicações científicas ou de cálculo intenso, não se esqueça de olhar os módulos NumPy (<http://numpy.scipy.org/>) e Scipy (<http://www.scipy.org/>).

Se você precisar trabalhar com gráficos (coluna, barra, torta, superfície e muitos outros), procure o módulo matplotlib (<http://matplotlib.sourceforge.net/>).

Outro projeto que não pode deixar de ser visitado é o Pycairo (<http://cairographics.org/pycairo/>), que permite o desenho de gráficos com recursos avançados, como suavização de curvas e transparências. Para trabalhar com imagens (PNG, JPG etc), o módulo PIL, ou Python Image Library (<http://www.pythonware.com/products/pil/>), é referência.

## 12.8 Listas de discussão

Uma forma de não ficar isolado é participar de listas de discussão. Para se aprofundar e continuar a estudar Python, inscreva-se na lista python-brasil (<http://www.python.org.br/wiki/EnvolvaSe>). Essa lista é muito ativa e conta com uma comunidade participativa que acolhe dúvidas básicas e avançadas. Antes de perguntar, leia os guias disponíveis no site e as regras de como fazer perguntas. O histórico da lista também traz milhares de perguntas e respostas que você pode pesquisar. A wiki é rica em exemplos de problemas e soluções escritas em Python e disponíveis em português.

Já para quem estiver estudando Django, a lista de discussão é django-brasil (<http://groups.google.com/group/django-brasil/?pli=1>).

# APÊNDICE A

## Mensagens de erro

Erros sempre podem ocorrer. Ora digitamos algo errado, ora nos esquecemos de digitar algo importante.

O interpretador Python informa erros por meio de mensagens que indicam o tipo do erro, assim como onde ele ocorreu (arquivo e linha). As seções deste apêndice mostram a listagem de um programa e a mensagem de erro respectiva. Lembre-se de que o interpretador segue regras como qualquer programa e que, às vezes, você pode ter mensagens causadas por erros em outras linhas de seu programa. Utilize as mensagens de erro como um bom palpite do que ocorreu. O que elas realmente indicam é onde, em seu programa, o interpretador foi interrompido e a causa dessa interrupção. É investigando esse palpite que você encontrará o verdadeiro erro.

### A.1 SyntaxError

Um dos tipos mais comuns. Um erro de sintaxe acontece quando o interpretador não consegue ler o que você escreveu. Em outras palavras, seu programa está mal formado, normalmente com erros de digitação ou símbolos esquecidos.

```
a="5
File "erros/syntax.py", line 1
    a="5
    ^
SyntaxError: EOL while scanning string literal
```

No exemplo anterior, a linha foi terminada sem fechar aspas.

```
for e in [1,2,3]
    print e
File "erros/syntax2.py", line 1
```

### Apêndice A ■ Mensagens de erro

```
for e in [1,2,3]
```

```
^
```

```
SyntaxError: invalid syntax
```

Todas as linhas com `for`, `while`, `if` e `else` devem ser encerradas com o símbolo de `:` para indicar o início de um novo bloco.

```
a=10
print a
File "erros/syntax4.py", line 2
    print a
    ^
SyntaxError: invalid syntax
```

Aqui a função `print` foi utilizada, mas observe que o parâmetro `a` não foi escrito entre parênteses.

Observe que um circunflexo é exibido na coluna em que o interpretador considera que o erro pode ter acontecido. Essa indicação é apenas uma dica, devendo ser investigada caso a caso. Alguns erros podem fazer com que o interpretador se perca, indicando o lugar errado. Sempre leia a linha indicada na mensagem de erro, mas não se esqueça de também olhar as linhas anteriores, caso não ache o erro. Sempre que ocorrer um erro de sintaxe, verifique:

1. A linha onde ocorreu um erro (`line`).
2. Se você fechou todas as aspas que abriu, o mesmo valendo para parênteses.
3. Os dois pontos após o `while`, `for`, `if`, `else`, definições de função, métodos e classes.
4. Se você não trocou letras minúsculas por maiúsculas e vice-versa.
5. Se você digitou corretamente todos os nomes.

### A.2 IndentationError

```
x=0
while x<10:
    print(x)
    x=x+1
File "erros/syntax5.py", line 4
```

```
x=x+1
^
IndentationError: unindent does not match any outer indentation level
```

Nesse caso, observe que a linha `x=x+1` não está alinhada nem com o bloco do `print` nem com o `while`. Todas as linhas de um mesmo bloco devem ser alinhadas na mesma coluna.

Uma variação muito difícil de perceber desse erro é quando misturamos espaços em branco com tabulações (tabs). Configure seu editor de textos para substituir tabs por espaços em branco ou vice-versa. Jamais misture tabulações e espaços em branco em seus programas em Python.

### A.3 KeyError

```
>>> mensalidades = {"carro":500, "casa":1500}
>>> print(mensalidades["seguro"])

Traceback (most recent call last):
  File "<pyshell#54>", line 1, in <module>
    print(mensalidades["seguro"])
  KeyError: 'seguro'
```

A exceção `KeyError` ocorre quando acessamos ou tentamos acessar um dicionário usando uma chave que não existe. No exemplo anterior, o dicionário `mensalidades` contém as chaves `carro` e `casa`. Na linha da função `print`, tentamos acessar `mensalidades["seguro"]`. Nesse caso, `"seguro"` é a chave que causa a mensagem de erro, uma vez que ela não pertence ao dicionário. Atenção ao trabalhar com chaves string, pois Python diferencia letras minúsculas de maiúsculas.

### A.4 NameError

```
while x<0:
    print(x)
    x=x+1

Traceback (most recent call last):
  File "erros/sintax3.py", line 1, in <module>
    while x<0:
NameError: name 'x' is not defined
```

### Apêndice A ■ Mensagens de erro

Aqui a variável `x` é utilizada em `while`, mesmo antes de ser iniciada. Toda variável em Python precisa ser inicializada antes de ser utilizada. Lembre-se de que para inicializarmos uma variável devemos atribuir um valor inicial. No caso, poderíamos escrever `x=0` antes da linha de `while` para inicializar a variável `x` com zero, resolvendo o erro.

Ao receber esse erro, verifique também se escreveu corretamente o nome da variável. Lembre-se de que o nome de uma variável, como qualquer identificador em Python, leva em consideração variações como letras minúsculas e maiúsculas. Por exemplo `X=0` não resolveria o erro, pois `x` e `X` são variáveis diferentes. Não se esqueça também de digitar corretamente os acentos, pois nomes acentuados são também diferentes de nomes sem acentos. Assim, `posicao` é diferente de `posição`.

### A.5 ValueError

A exceção `ValueError` pode acontecer por diversas causas. Uma delas é a impossibilidade de converter um valor com as funções `int` ou `float`:

```
>>> int("abc")
Traceback (most recent call last):
  File "<pyshell#67>", line 1, in <module>
    int("abc")
ValueError: invalid literal for int() with base 10: 'abc'
>>> float("maria")
Traceback (most recent call last):
  File "<pyshell#68>", line 1, in <module>
    float("maria")
ValueError: could not convert string to float: maria
```

Ela pode ocorrer se, por exemplo, o valor retornado pela função `input` for inválido.

Essa exceção também ocorre quando procuramos uma string que não existe, como mostrado abaixo.

```
>>> s="Alô mundo"
>>> s.index("rei")
Traceback (most recent call last):
  File "<pyshell#64>", line 1, in <module>
    s.index("rei")
ValueError: substring not found
```

## A.6 TypeError

Essa exceção acontece se tentamos chamar uma função usando mais parâmetros do que ela recebe. No exemplo abaixo, a função `float` foi chamada com dois parâmetros: 35 e 4. Lembre-se de que números em Python devem ser escritos no formato americano, separando a parte inteira da parte decimal de um número com ponto e não com vírgula. Esse também é um exemplo de que um erro pode ser apresentado como outro erro, exigindo que você sempre interprete a mensagem de forma a saber o que realmente aconteceu.

```
>>> float(35,4)
Traceback (most recent call last):
  File "<pyshell#69>", line 1, in <module>
    float(35,4)
TypeError: float() takes at most 1 argument (2 given)
```

`TypeError` também ocorre quando trocamos o tipo de um índice. No exemplo abaixo, tentamos utilizar a string “marrom” como índice de uma lista. Lembre-se de que listas só podem ser indexadas por números inteiros. Dicionários, por sua vez, permitem índices do tipo string.

```
>>> s["amarelo", "vermelho", "verde"]
>>> s["marrom"]
Traceback (most recent call last):
  File "<pyshell#71>", line 1, in <module>
    s["marrom"]
TypeError: string indices must be integers
```

## A.7 IndexError

`IndexError` indica que um valor inválido de índice foi utilizado. No exemplo a seguir, a string `s` contém apenas cinco elementos, podendo ter índices de 0 a 4.

```
>>> s="ABCDE"
>>> s[10]
Traceback (most recent call last):
  File "<pyshell#66>", line 1, in <module>
    s[10]
IndexError: string index out of range
```

## Referências

Python Foundation. Python 3.1.2 Tutorial. 2010 Disponível em <http://docs.python.org/py3k/tutorial>.

Summerfield, M. *Programming in Python 3 – A Complete Introduction to the Python Language*. 2 ed. Addison-Wesley, 2010.

Pilgrim, M. *Dive into Python 3*. 2 ed. Apress, 2009.

Cormen, Thomas H; Leiserson, Charles E; Rivest, Ronald L; Stein, Clifford. *Algoritmos: teoria e prática*. Rio de Janeiro: Elsevier, 2002

**Símbolos**

`_eq__`, 235  
`_ge__`, 236  
`_getitem__`, 233  
`_gt__`, 235  
`_init__`, 217, 218, 219, 223, 228  
`_iter__`, 232  
`_le__`, 236  
`_len__`, 232  
`_lt__`, 235  
`_neq__`, 235  
`@nome.setter`, 240  
`@property`, 240  
`_repr__`, 235  
`@staticmethod`, 237, 238  
`_str__`, 233  
`@total_ordering`, 237

**A**

`abs`, 172  
`abspath`, 214  
`acumuladores`, 91, 92  
`abstração`, 221  
`adição`, 42  
`alinhandando`. Consulte `string`, `ljust`, `center`, `rjust`  
`alter table`, 280  
`and`, 55, 56  
`apagando elementos`. Consulte `listas`, `del`; Consulte `dicionários`, `del`  
`append`. Consulte `listas`, `append`  
`argv`. Consulte `sys`, `argv`  
`arquivos`, 188  
`abrindo`, 188  
`abrindo utf-8`, 202  
`apagando`. Consulte `remove`  
`escrevendo`, 189  
`fechando`, 190  
`gravando`, 189  
`isfile`, 209  
`lendo linhas`, 190  
`listando`. Consulte `listdir`  
`modos de abertura`, 188  
`renomeando`. Consulte `rename`  
`tamanho`. Consulte `getsize`  
`tempo de criação`. Consulte `getctime`  
`tempo de modificação`. Consulte `getmtime`  
`tempo do último acesso`. Consulte `getatime`  
`verificando se existe`. Consulte `exists`  
`verificando se um caminho é`. Consulte `isdir`; Consulte `isfile`  
`atribuição`, 44  
`avg`, 281

**B**

`banco de dados`, 260  
`conexão`, 264  
`consulta`, 266  
`cursor`, 264  
`transação`, 265  
`basename`, 214  
`bloco`, 76, 77, 86  
`booleano`, 53  
`break`, 93, 114, 163  
`Bubble Sort`. Consulte `listas`, `ordenação`

**C**

`caminho`. Consulte `path`  
`Campos`, 262  
`caracteres`. Consulte `string`  
`center`. Consulte `string`, `center`  
`centralizando`. Consulte `string`, `center`  
`chave de grupo`, 281  
`chave primária`, 278, 290  
`chdir`, 206  
`class`, 217  
`classes`, 216, 217  
`atributos`, 218  
`construtor`, 217, 220  
`encapsulamento`, 220  
`herança`, 227, 228, 229  
`métodos`, 218  
`superclasse`, 228, 229  
`close`, 189  
`condições`, 75  
`conjunção`. Consulte `and`  
`contadores`, 87  
`continue`, 159  
`count`, 281. Consulte `string`, `count`  
`ctime`. Consulte `time`, `ctime`

**D**

`datas`. Consulte `tempo` Consulte `time`  
`def`, 161  
`del`. Consulte `listas`, `del`; Consulte `dicionários`, `del`  
`dias`. Consulte `tempo` Consulte `time`  
`dicionários`, 127  
`com listas`, 131  
`del`, 131  
`keys`, 130  
`values`, 130  
`diretório corrente`. Consulte `.getcwd`  
`diretórios`  
`apagando`. Consulte `rmdir`  
`atual`. Consulte `.getcwd`

`corrente`. Consulte `.getcwd`  
`criando`. Consulte `mkdir`  
`listando o conteúdo`. Consulte `listdir`  
`trocando`. Consulte `chdir`  
`verificando se um caminho é`, 209  
`dirname`, 214  
`disjunção`. Consulte `or`  
`divisão`, 42

**E**

`elevar um número`. Consulte `exponenciação`  
`elif`, 83  
`else`, 79, 80, 81, 82, 115, 158  
`endswith`. Consulte `string`, `endswith`  
`entrada de dados`. Consulte `input`  
`enumerate`, 117, 132  
`e, operador`. Consulte `and`  
`estruturas de repetição`. Consulte `repetições`  
`exists`, 209  
`exponenciação`, 43  
`expressões lógicas`, 58  
`extend`. Consulte `listas`, `extend`

**F**

`False`. Consulte `falso`  
`falso`, 53  
`fatorial`, 166, 167, 170  
`Fibonacci`, 171  
`FIFO`, 109  
`filas`. Consulte `listas`, `fila`  
`find`. Consulte `string`, `find`  
`float`, 71, 73, 74  
`for`, 114, 159  
`força`, 156  
`from`, 182  
`funções`  
`como parâmetro`, 178  
`criando`. Consulte `def`  
`desempacotamento de parâmetros`, 180  
`lambda`, 181  
`nomeando parâmetros`, 176  
`parâmetros opcionais`, 174, 180  
`recursivas`, 170  
`retornado valores`. Consulte `return`

**G**

`getatime`, 210  
`getctime`, 210  
`getcwd`, 205  
`getmtime`, 210

`getsize`, 210  
`global`, 170  
`gmtime`, 210  
`group by`, 281

**H**

`horas` ;Consulte `tempo` Consulte `time`  
`html`  
`body`, 202  
`h1`, 204  
`h2`, 204  
`h3`, 204  
`h4`, 204  
`h5`, 204  
`h6`, 204  
`head`, 201  
`html`, 201  
`li`, 205  
`meta`, 201  
`p`, 203  
`title`, 201  
`ul`, 205  
`HTML`, 200

**I**

`IndentationError`, 319  
`if`, 75, 80  
`import`, 181, 182, 223, 229  
`imprimindo`. Consulte `print`  
`in`, 129, 139, 140  
`index`. Consulte `string`, `index`  
`IndexError`, 322  
`init`, 217  
`input`, 70, 73  
`instância`, 216  
`int`, 71  
`isalnum`. Consulte `string`, `isalnum`  
`isalpha`. Consulte `string`, `isalpha`  
`isdigit`. Consulte `string`, `isdigit`  
`isdir`, 209  
`isfile`, 209  
`islower`. Consulte `string`, `islower`  
`isnumeric`. Consulte `string`, `isnumeric`  
`isprintable`. Consulte `string`, `isprintable`  
`isspace`. Consulte `string`, `isspace`  
`isupper`. Consulte `string`, `isupper`  
`iter`, 232

**J**

`jogo da forca`, 156  
`join`. Consulte `os.path`, `join` Consulte `string`, `join`

juntando caminhos. Consulte `os.path.join`  
juntando strings. Consulte `string.join`

**K**  
`KeyError`, 129, 320

**L**  
`lastrowid`, 299  
`len`, 61, 104, 105  
`lendo`  
de arquivos. Consulte `read` Consulte `readlines`  
do teclado. Consulte `input`

`LIFO`, 110  
`list`, 130, 137  
listas, 98  
alterando, 99  
`append`, 105, 107  
com strings, 120  
cópia, 103  
dentro de listas, 120  
desempacotamento, 179  
elementos, 98  
empacotamento, 179  
`extend`, 106, 107  
fatiá, 103  
fila, 109  
índices, 98, 101  
índices negativos, 103  
ordenação, 123  
pesquisa, 112  
pilha, 110  
`pop`, 109, 111  
referências, 102  
tamanho, 104  
vazia, 98

`listdir`, 208  
`ljust`. Consulte `string.ljust`  
`localtime`. Consulte `time.localtime`  
`lower`, 139  
`lstrip`. Consulte `string.lstrip`

**M**  
maior divisor comum, 172  
`makedirs`, 207  
`max`, 281  
média, 92  
média aritmética, 92  
menor múltiplo comum, 172  
meses. Consulte `tempo` Consulte `time`  
método, 221  
`min`, 281

`mkdir`, 206  
modelo, 221  
módulos, 181  
mostrando na tela. Consulte `print`  
multiplicação, 42

**N**  
`NameError`, 320  
não, operador. Consulte `not`  
negação. Consulte `not`  
`not`, 55, 56  
`not in`, 139  
números  
aleatórios, 183  
faixas. Consulte `range`  
inteiros, 52  
ponto flutuante, 52  
sem sinal. Consulte `abs`

**O**  
objetos, 216  
`open`, 188  
operações matemáticas  
adição, 42  
divisão, 42  
exponenciação, 43  
multiplicação, 42  
parênteses, 43  
prioridade, 43  
resto da divisão, 43  
subtração, 42  
operadores lógicos, 58  
operadores relacionais, 53  
`or`, 55, 56, 57  
ordenando. Consulte `listas`, `ordenação`  
`order by`, 279  
`os`  
`chdir`, 206  
`getcwd`, 205  
`listdir`, 208  
`makedirs`, 207  
`mkdir`, 206  
`rename`, 208  
`rmdir`, 208  
`walk`, 215  
`os.path`, 209  
`abspath`, 214  
`basename`, 214  
`dirname`, 214  
`exists`, 209  
`getatime`, 210  
`getctime`, 210

`getmtime`, 210  
`getsize`, 210  
`isdir`, 209  
`isfile`, 209  
`join`, 214  
`splitdrive`, 214  
ou, operador. Consulte `or`

**P**  
parâmetro, 42  
`path`, 206  
`pickle`, 259  
pilhas. Consulte `listas`, `pilha`  
`pop`. Consulte `listas`, `pop`  
`print`, 26, 41, 45  
printend, 116  
procurando. Consulte `string`, `count`, `find`, `index`, `rindex`, `rfind`; Consulte `listas`, `pesquisa`

**Python**  
editando arquivos, 36  
executando, 39, 40  
instalação, 27  
instalação Linux, 34  
instalação Mac OS X, 34  
instalação Windows, 28  
interpretador, 34  
salvando arquivos, 38

**R**  
`randint`, 183  
`random`, 184  
`range`, 115, 116  
rastreamento, 68  
`read`, 189  
`readlines`, 190  
`registro`, 262  
`rename`, 208  
repetições, 85  
for. Consulte `for`  
interrompendo. Consulte `break`  
while. Consulte `while`  
`replace`. Consulte `string.replace`  
resto da divisão, 43  
`return`, 162, 163, 164  
`rfind`. Consulte `string.rfind`  
`rindex`. Consulte `string.rindex`  
`rjust`. Consulte `string.rjust`  
`rmdir`, 208  
`rstrip`. Consulte `string.rstrip`

**S**  
`sample`, 184  
se. Consulte `if`  
segundos. ;Consulte `tempo` Consulte `time`  
`select`, 290  
`self`, 217, 221, 228  
senão. Consulte `else` Consulte `elif`  
sequência de caracteres. Consulte `string`  
`shuffle`, 184  
`SyntaxError`, 318  
sistema binário, 51  
`split`. Consulte `string.split`  
`splitdrive`, 214  
`splines`. Consulte `string.splines`  
`SQL`, 262  
`alter table`, 280  
`create table`, 264  
`delete`, 276  
`group by`, 281  
`having`, 285  
`insert into`, 264  
`parâmetros`, 265  
`select`, 266  
`update`, 273  
`where`, 270  
`SQLite`, 263  
`startswith`. Consulte `string.startswith`  
`strftime`. Consulte `time.strftime`  
`string`, 60  
alinhando. Consulte `string.ljust`, `center`, `rjust`  
`center`, 144  
começando em. Consulte `string.startswith`  
composição, 63  
concatenação, 62  
convertendo em maiúsculas. Consulte `string.upper`  
convertendo em minúsculas. Consulte `string.lower`  
`count`, 140  
`endswith`, 138  
fatiamento, 65  
`find`, 140, 141  
formatação, 150  
`index`, 142  
`isalnum`, 147  
`isalpha`, 147  
`isdigit`, 147  
`islower`, 148  
`isnumeric`, 148  
`isprintable`, 149  
`isspace`, 149  
`isupper`, 148

join, 137  
justificando. Consulte string, ljust, center, rjust  
ljust, 144  
lower, 138, 158  
lstrip, 146  
quebrando. Consulte string, split, splitlines  
replace, 145  
retirando espaços. Consulte string, strip,  
    lstrip, rstrip  
rfind, 141  
rindex, 142  
rjust, 144  
rstrip, 146  
se é alfanumérica. Consulte string, isalnum  
se é minúscula. Consulte string, islower  
se é numérica. Consulte string, isnumeric  
separando. Consulte string, split, splitlines  
se pode ser impressa. Consulte string, isprint  
se tem apenas dígitos. Consulte string, isdigit  
se tem apenas espaços. Consulte string, isspace  
split, 145  
splitlines, 145  
startswith, 138  
strip, 146, 158  
substituindo. Consulte string, replace  
trocando. Consulte string, replace  
upper, 138  
strip. Consulte string, strip  
Structured Query Language. ver SQL  
subclasse, 228  
subtração, 42  
sum, 281  
superclasse, 228  
superclasses, 229  
sys, 191  
    argv, 191

**T**  
tabelas, 262  
tags, 200  
tags html. Consulte html  
tempo

    formatando como string. Consulte time,  
        strftime  
    hora local. Consulte time, localtime  
time, 210  
    ctime, 210  
    gmtime, 210  
    localtime, 211  
    strftime, 212

tipo  
    lógico. Consulte variáveis, tipo lógico  
    numérico. Consulte variáveis, numéricas  
    string. Consulte string  
verificando. Consulte type  
trocando de diretório. Consulte chdir  
True. Consulte verdadeiro  
tupla, 117, 133, 210, 265  
type, 185, 187  
TypeError, 322

**U**  
uniform, 184  
upper, 139

**V**  
validação, 173  
valor absoluto. Consulte abs  
ValueError, 73, 142, 321  
variáveis, 44, 67  
    arquivos. Consulte arquivos  
    dicionários. Consulte dicionários  
    escopo, 168  
    globais, 168  
    locais, 168  
    numéricas, 50  
    regras de nomenclatura, 49  
    tipo file, 189  
    tipo lista. Consulte Listas  
    tipo lógico, 53  
    tipo string, 60  
    troca de valores, 124  
verdadeiro, 53

**W**  
walk, 215  
while, 86, 88, 93, 95, 114, 159  
write, 189

**Y**  
yield, 299

# INTRODUÇÃO À PROGRAMAÇÃO COM PYTHON

Este livro é orientado ao iniciante em programação.

Os conceitos básicos de programação, como expressões, variáveis, repetições, decisões, listas, funções, arquivos e banco de dados com SQLite 3 são apresentados um a um com exemplos e exercícios. A obra visa a explorar a programação de computadores como ferramenta do dia a dia. Ela pode ser lida durante um curso de introdução à programação de computadores e usada como guia de estudo para autodidatas. Para aproveitamento pleno do conteúdo, conhecimentos básicos de informática, como digitar textos, abrir e salvar arquivos, são suficientes. Todo software utilizado no livro pode ser baixado gratuitamente, sendo executado em Windows, Linux e Mac OS X.

Embora a linguagem Python (versão 3.x) seja muito poderosa e repleta de recursos modernos de programação, este livro não pretende ensinar a linguagem em si, mas ensinar a programar. Alguns recursos da linguagem não foram utilizados para privilegiar os exercícios de lógica de programação e oferecer uma preparação mais ampla ao leitor para outras linguagens. Essa escolha não impedi a apresentação de recursos poderosos da linguagem, embora o livro não seja fundamentalmente uma obra de referência.

## sobre o autor

**Nilo Ney Coutinho Menezes** é pesquisador na área de redes e sistemas distribuídos. Coordenou equipes de desenvolvimento de software para indústrias em Manaus. Trabalha hoje como pesquisador na área de simulação distribuída de sistemas ferroviários, onde termina seu doutorado pela Universidade de Mons, na Bélgica.

É mestre em informática e bacharel em processamento de dados pela Universidade Federal do Amazonas.

O livro foi elaborado com base em sua experiência como professor de lógica de programação desde 1994.

### Fique conectado:



[twitter.com/novateceditora](http://twitter.com/novateceditora)



[facebook.com/novatec](http://facebook.com/novatec)



[www.novatec.com.br](http://www.novatec.com.br)

ISBN 978-85-7522-408-3



9 788575 224083