Think Python

How to think like a computer scientist

Second Edition

by Allen B. Downey

Resumo por Bruno de M. Ruas

22 de agosto de 2021

Conteúdo

1 O Caminho do Programa **1**
- 1.1 O que é um Programa? . . . 1
- 1.2 Executando o Python . . . 2
- 1.3 Operadores Aritméticos . . . 2
- 1.4 Linguagens Formais e Naturais . . . 2
  - 1.4.1 Regras de Sintaxe . . . 2
- 1.5 Debugging . . . 3
- 1.6 Glossário . . . 3

2 Variáveis, Expressões e Statements **5**
- 2.1 Statements de atribuição . . . 5
- 2.2 Nome de Variáveis . . . 5
  - 2.2.1 Keywords do Python 3 . . . 5
- 2.3 Expressões e Statements . . . 6
- 2.4 Ordem de Operações . . . 6
- 2.5 Operadores de String . . . 6
- 2.6 Comentários . . . 7
- 2.7 Debugging . . . 7
- 2.8 Glossário . . . 7

3 Funções **9**
- 3.1 Chamadas de Funções . . . 9
- 3.2 Funções Matemáticas . . . 9

4 Case Study: Interface Design **11**

5 conditionals and Recursion **13**

6 Fruitful Funtions **15**

7 Iteration **17**

8 Strings **19**

9 Case Study: Word Play **21**

10 Lists **23**

11 Dictionaries **25**

12 Tuples **27**

13 Case Study: Data Structure Selection **29**

14 Files **31**

15 Classes and Objects **33**

16 Classes and Functions **35**

17 Classes and Methods **37**

18 Inheritance **39**

19 The Goodies **41**

20 Debugging **43**

21 Analysis of Algorithms **45**

1

O Caminho do Programa

> *"The single most important skill for a computer scientistis **problem solving**. That is the ability to formulate problems, think creatively about solutions, and express a solution clearly and accurately"*
>
> – página 1

1.1 O que é um Programa?

Um **programa** é uma sequencia de instruções que especifica como fazer um determinado computation. Um **computation**[1] é qualquer tipo de cálculo que inclua passos artiméticos (computação matemática) e não artiméticos (computação simbólica) e que segue uma determinada ordem.

Algumas características comuns em qualquer linguagem de programação:

- input → dados de teclado, arquivo, rede, etc.
- output → mostrar dados na tela, salvar em um arquivo, enviar na rede, etc.
- math → operações matemáticas
- conditional execution → só executar um pedaço do código após certas condições
- repetition → reexecutar um pedaço do código com alguma variação

[1] https://en.wikipedia.org/wiki/Computation

1.2 Executando o Python

O **interpretador** do Python é um programa que lê e executa códigos escritos em Python. Existem duas maneiras[2] de se executar códigos em Python a primeira é no modo **interativo** onde podemos enviar as ordens direto pro interpretador e a segunda em em modo **script** onde o interpretador analisa o código inteiro antes de começar o processamento.

1.3 Operadores Aritméticos

```
# Soma
40 + 1

# Subtracao
10 - 1

# Multiplicacao
20 * 10

# Divisao
11 / 1

# Exponenciacao
6 ** 2
```

Listing 1.1: Operadores Básicos

1.4 Linguagens Formais e Naturais

As **linguagens naturais** são as linguagens que as pessoas usam para se comunicar no dia a dia. São criadas e se modificam organicamente com o passar do tempo.

Por outro lado, as **linguagens formais** são criadas por pessoas e possuem uma aplicação específica. As linguagens de computação são linguagens formais desenvolvidas para expressar computações.

As linguagens formais possuem regras estritas que chamamos de **regras de sintaxe**.

1.4.1 Regras de Sintaxe

As regras de sintaxe possuem dois sentidos gerais:

- tokens → são os elementos da linguagem

[2] Essa parte eu to adiantando da página 13

- estrutura → é a regra de combinação dos tokens

Quando o interpretador lê um código, ele faz um processo chamado **parsing** que é justamente entender a estrutura de combinação entre os diferentes tokens presentes no script. No caso do Python a identação faz um papel importante na estrutura.

1.5 Debugging

Os erros em um programa são chamados de **bugs** e o processo de resolução desses erros é chamado de **debugging**.

1.6 Glossário

- problem solving → processo de formular um problema, achar uma solução e expressá-la
- high-level language → lingaugem desenhada para ser fácil para um humano ler e escrever
- low-level language → lingaugem desenhada para ser fácil para um computador ler. AKA "machine language"ou "assembly language"
- portability → capacidade de um programa rodar em mais de um tipo de computador
- interpreter → um programa que lê outro programa e o executa
- prompt → caracteres no prompt que indicam que ele está pronto para receber input direto do usuário
- program → um conjunto de instruções que especifica uma computation
- print statement → uma instrução que diz para o interpretador mostrar caracteres na tela
- operator → um caracter especial que representa uma computação simples (+, -, *, /, **)
- value → uma das unidades básica de dados
- type → a categoria em que um valor pertence
- interger → numero inteiros
- floating-point → números fracionados
- string → sequencia de caracteres

- natural language → linguagem humana
- formal language → linguagem com objetivo e regras específicos
- token → um dos elementos básicos de uma linguagem formal, análogo à palavra na linguagem natural
- syntax → as regras de estrutura de um programa
- parse → processo de examinar um programa e analisar sua estrutura
- bug → um erro no programa
- debugg → o processo de resolver um erro

2

Variáveis, Expressões e Statements

> *"One of the most powerful features of a programming language is the ability to manipulate* ***variables****. A variable is a name that refers to a value"*
>
> – página 11

2.1 Statements de atribuição

```
# string
message = 'acesse www.economiamainstream.com.br'

# interger
n = 12

# floating point
pi = 3.1415
```

Listing 2.1: Atribuições para diferentes classes

2.2 Nome de Variáveis

Os nomes das variáveis podem ser o que você quiser. Contudo, existem algumas regras a serem seguidas: 1) Não podem começar com número; 2) Não pode conter caracteres ilegais (@ , # , etc) e 3) Não podem ser uma **keyword**.

2.2.1 Keywords do Python 3

Essas são as palavras que o interpretador já reservou como tendo significado e que não podem ser atribuídas como nome para nenhuma variável.

False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

2.3 Expressões e Statements

Uma **expressão** é uma combinação de valores, variáveis e operadores. Um **statement** é uma unidade de código que possui algum efeito.

2.4 Ordem de Operações

Python segue a convenção matemática **PEMDAS**:

- **P**arênteses
- **E**xponenciação
- **M**ultiplicação
- **D**ivisão
- **A**dição
- **S**ubtração

2.5 Operadores de String

Quando se trata de strings, Python resignifica os operadores matemáticos para operações " similares " ao contexto original dos números.

```
1
2 # operador de soma em strings
3 'economia' + ' ' + 'mainstream'
4 >>> 'economia mainstream'
5
6 # operador de multiplicacao em strings
7 'economia' * 3
8 >>> 'economiaeconomiaeconomia'
```

Listing 2.2: Operadores com strings

2.6 Comentários

Para comentar alguma parte do seu código use o caracter #. É recomendado que se use comentários apenas para partes não óbvias do código. Se esforce para que o código seja o mais alto-explicativo possível (o nome das varáveis pode ajudar bastante)

2.7 Debugging

Existem 3 tipos de erros que podem acontecer em um programa: 1) Erros de Sintaxe, 2) Erros de Runtime e 3) Erros Semântico.

- Erro de Sintaxe → Relacionados à estrutura da linguagem formal e as regras dessa estrutura e dos tokens. Esse tipo de erro não permite a execução do código.
- Erro de Runtime → Um erro que não acontece quando o programa inicia mas, em algum momento durante a execução, aparecem posteriormente (geralmente relacionados às exceções não previstas).
- Erro de Semântica → O código roda sem erros aparentes, mas o resultado (output) é diferente do esperado. Esse tipo de erro é mais difícil de identificar e de se solucionar.

2.8 Glossário

- variable → um nome referente a um valor
- assignment → um statement que atribui uma variável a um valor
- state diagram → representação gráfica de assignment
- keyword → palavras (tokens) reservados pela lingagem para ações previamente definidas
- operand → um valor que será usado em uma operação
- expression → combinação de variáveis, operadores e valores que representa um único resultado
- evaluate → processo de simplificação de uma expressão usando os operadores na ordem correta até chegar em um único resultado
- statement → uma parte do código que representa um comando ou uma ação

- execute → computar um statement para produzir o resultado por ele ordenado
- interactive mode → maneira de interagir com o interpretador do Python escrevendo o input diretamente no prompt
- script mode → maneira de interagir com o interpretador do Python através da leitura de um script.py
- order of operations → regras de resolução das expressões de acordo com os tokens contidos nos comandos
- concatenate → juntar dois ou mais strings
- comment → lançamento no script de comentários para facilitar a leitura humana do código. O interpretador não usará essas partes no processamento
- syntax error → um erro que torna o processo de parse impossível e, por isso, impede a execução do programa
- execption → erro de runtime. Geralmente acontece por uma exceção não prevista no código
- semantics → o sentido de um programa
- semantic error → um erro onde o programa é executado mas não produz os outputs esperados

3

Funções

> *"A **function** is a named sequence of statements that performs a computation. [...] You specify the name and the sequence of statements. Later, you 'call' the function by name."*
>
> – página 21

3.1 Chamadas de Funções

Uma função possui um **nome** e, normalmente, recebe **argumentos**. O resultado obtido após a chamada de uma função com os devidos argumentos é o seu **valor de retorno**. O Python já possui internamento várias funções que são muito úteis para os desenvolvedores.

3.2 Funções Matemáticas

Um **module** é um conjunto de funções e variáveis. Essas funções ficam disponíveis ao interpretador por meio de um **statement de importação**. O Python possui nativamente um module de operações matemáticas.

```
1
2 import math
```

Listing 3.1: Carregando o modelo math

4

Case Study: Interface Design

5

conditionals and Recursion

6

Fruitful Funtions

7

Iteration

8

Strings

9

Case Study: Word Play

10

Lists

11

Dictionaries

12

Tuples

13

Case Study: Data Structure Selection

14

Files

15

Classes and Objects

16

Classes and Functions

17

Classes and Methods

18

Inheritance

19

The Goodies

20

Debugging

21

Analysis of Algorithms