Pedro Maciel Xavier

COMPUTAÇÃO I

O GUIA DE ESTUDOS

PRIMEIRA EDIÇÃO

UFRJ

2021

PREFÁCIO

A intenção deste livro é reunir exercícios interessantes para um curso de introdução à computação na forma de pequenos projetos. Conta com enunciados elaborados e desenvovimentos longos, a fim de explorar alguns dos conceitos básicos de programação.

Este texto é fruto das minhas atividades de monitoria nos cursos de Computação I (MAB125) na Universidade Federal do Rio de Janeiro (UFRJ) entre os anos de 2018 e 2019. Entre as turmas que trabalhei estão os cursos de Engenharia Naval, Engenharia de Produção e o Bacharelado em Ciências Matemáticas e da Terra (BCMT). Por conta disso, você deve encontrar referências a temas bastante diversos, dadas as inúmeras aplicações distintas que foram exercitadas por mim e pelos professores nestes diferentes contextos.

O curso é pensado para alunos que acabaram de ingressar na Graduação e que possivelmente não tiveram contato com computação anteriormente.

Vamos utilizar como referência a linguagem Python. No entanto, este material não pretende ser um curso ou referência da linguagem. Ela nos dará suporte para enunciar os fundamentos de programação, assim como noções de algoritmos e estruturas de dados. De maneira geral, as especificidades da linguagem são discutidas com maior profundidade nos tópicos finais de cada capítulo.

A tipografia foi elaborada no sistema LaTeXe fortemente inspirada no *layout* da terceira edição do livro *Linear Algebra and its Applications*[1], de Gilbert Strang, um dos meus livros favoritos na jornada acadêmica.

Petrópolis, março de 2021

NOTAÇÃO

Como referência, a notação matemática segue o padrão encontrado na maioria dos livros e artigos do assunto. No entanto, principalmente na hora de falar de código utilizaremos alguns recursos tipográficos para auxiliar na comunicação.

PYTHON ■

Quando falando sobre os elementos da linguagem, estes aparecerão em fonte monoespaçada, tal como `x`, `y` e `math`. As palavras reservadas (comandos) são destacadas como em `if`, `return` e `del`. Funções, tipos e exceções da biblioteca padrão são mostrados como `list`, `range`, `open`, etc. Por fim, temos as *strings*, `'Oi Mundão'`, `"Adeus, Mundinho"`; e os comentários, `# Voltei Mundão`. Trechos de código serão formatados como a seguir:

```
1 import math
2
3 def f(n):
4     if n == 0 or n == 1:
5         return 1
6     else:
7         return f(n - 1) + f(n - 2)
8
9 print("f(10) =", f(10))
```

Sessões do console interativo aparecem de maneira bastante similar:

```
1 >>> L = [2, 3, 5, 7, 11]
2 >>> L[1]
3 3
4 >>> L.append(13)
5 >>> L
6 [2, 3, 5, 7, 11, 13]
```

Pedro Maciel Xavier

COMPUTAÇÃO I

O GUIA DE ESTUDOS

PRIMEIRA EDIÇÃO

UFRJ

2021

Contents

INTRODUÇÃO

1

TIPOS, VARIÁVEIS, OPERADORES E FUNÇÕES

INTRODUÇÃO

É interessante, de fato, iniciar o estudo da computação através das funções. As funções trazem consigo os conceitos de tipos e variáveis de maneira muito natural. De maneira simples, uma função pode ser compreendida como a transformação dos dados de entrada nos dados de saída, cada qual com seus respectivos tipos.

A noção de tipo está também relacionada à ideia de conjunto. Ser de um determinado tipo é pertencer a um conjunto, mais especificamente ao conjunto daquele tipo. Isto fica bem claro quando se pensa nos números, embora seja um conceito adequado a uma miríade de situações.

Em textos de matemática a nível superior, é comum que as funções recebam outros nomes a depender do contexto. Uma nomeclatura um tanto elucidativa é chamar funções de *aplicações*, em inglês, *mappings*. É de bom gosto observar que as funções são *mapas* que levam de um ponto em um conjunto noutro ponto de outro conjunto.

Dito isto, é importante pensar por um tempo no significado de uma função, ter em mente que qualquer processo computacional, por mais complicado que seja, pode ser representado pela aplicação de diferentes funções em sequência. Formalmente, uma transformação será descrita pela composição de diversas transformações mais simples. Isso tudo, porém, no mundo das ideias.

Os *operadores*, por outro lado, são a materialização da computação que desejamos realizar. Operações tratam da maneira explícita como uma tarefa lógica ou aritmética deve ser encadeada. Pode-se dizer, portanto, que as operações estão entre as funções mais simples e mais próximas da realidade. O que acabo de dizer aparece de maneira clara no projeto dos processadores digitais. A arquitetura x86 moderna possui algumas centenas de instruções, dentre operações

aritméticas, de acesso à memória e de controle. Um processador RISC, por sua vez, chega a operar com apenas 34 intruções[2]. O importante é ter em mente que as mais complexas atividades realizadas por um computador são compostas a partir de um conjunto reduzido de operações básicas.

TIPOS

A teoria por trás dos tipos é bastante extensa e é tratada com detalhe muito maior no volume de Computação II[3]. No decorrer dos próximos capítulos vamos tratar das especificidades e casos de uso dos tipos básicos. Por enquanto, vamos nos ater à sua apresentação e ao uso prático inicial.

Vamos começar pelos tipos numéricos. Estão disponíveis os tipos `int`, `float` e `complex`, construídos para representar os números inteiros ($\mathbb{Z}$), reais ($\mathbb{R}$) e complexos ($\mathbb{C}$), respectivamente. A primeira vista, podemos distingui-los pela sintaxe: os números representados em ponto-flutuante[1]podem ser escritos adicionando a parte decimal após o ponto, como em `0.5` e `-1.2` ou através de notação científica, como `6.02e+23` ($6,02 \times 10^{23}$). Os números complexos são simplesmente uma extensão do tipo `float`, onde a parte imaginária é indicada acrescentando um `j` ao final do número. A unidade imaginária `i`, por exemplo, escreve-se `1j`. Tendo em mãos estes três tipos, podemos realizar as operações aritméticas básicas de adição `+`, subtração `-`, multiplicação `*` e divisão `/`; além da exponenciação `**` e da divisão inteira, realizada através dos operadores de quociente `//` e resto `%`.

Tome um tempo para assimilar o comportamento destes operadores, procurando explorar tanto os casos simples quanto os mais problemáticos como a divisão por zero ou operações entre números grandes demais. Atente, em cada operação para os tipos dos operandos assim como para o tipo do resultado. A própria sintaxe deve deixar isso claro mas você pode sempre contar com a função `type`, como no exemplo a seguir.

```
1 >>> 1 + 1
2 2
3 >>> 1 + 1.0
4 2.0
5 >>> type(1 + 1)
6 <class 'int'>
7 >>> type(1 + 1.0)
8 <class 'float'>
```

[1]Uma discussão mais aprofundada sobre ponto-flutuante e o padrão IEE 754 se encontra no capítulo 3.

Se tem uma coisa fundamental a ser percebida neste primeiro exemplo é a hierarquia existente entre os tipos `int`, `float` e `complex`, relação que vamos relembrar através da adaptação de uma velha figura. Pensando na correspondência

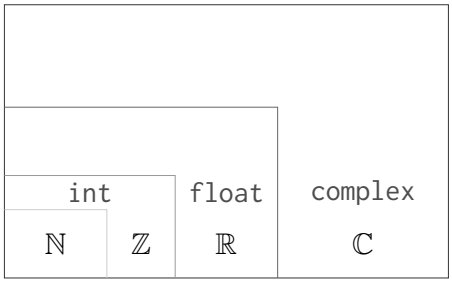


Figure 1.1: Conjuntos numéricos

entre os tipos e os conjuntos que representam, podemos afirmar que em uma operação entre números de tipos distintos o resultado será do tipo mais geral, ou seja, do conjunto mais externo.

VARIÁVEIS E ATRIBUIÇÃO ■

```
1 >>> x = 0
2 >>> y = 1
3 >>> x + y
4 1
```

FUNÇÕES ■

Para definir uma função vamos utilizar o comando `def`, seguido do nome da função, dos seus parâmetros e, por fim, o código que deve executar.

```
1 >>> def f(x, y):
2 ...     return x + y
3 ...
4 >>> z = f(2, 3)
5 >>> z
6 5
```

1.1 Cálculo I - Limites

O limite ℓ de uma função $f(x)$ no ponto a é o valor que a função assume conforme x se aproxima de a. Escrevemos:

$$\ell = \lim_{x \to a} f(x)$$

Quando a função não apresentar nenhuma descontinuidade ao redor do ponto a, podemos afirmar com tranquilidade que $\ell = f(a)$. Uma maneira simples de aproximar os limites laterais no computador é calcular

$$\lim_{x \to a^-} f(x) \approx f(a - \epsilon) \text{ e } \lim_{x \to a^+} f(x) \approx f(a + \epsilon)$$

escolhendo um valor suficientemente pequeno para ϵ.

```
1 def lim(f, a, e=0.01):
2     """Limite da função f no ponto a."""
3     e = 1E-4 # 0.0001
4     return (f(a - e) + f(a + e)) / 2.0
```

Proposta
Faça uma função `lim(f, a)` para calcular o limite da função `f` no ponto `a`. Leve em consideração a existência do limite.

1.2 Música I - As notas e os sons

Cada nota musical corresponde a uma frequência distinta (em Hz). Tomando o lá central (A4) como referência, em 440Hz, podemos calcular a frequência das outras notas com base na distância relativa a essa nota.

$$f(n) = 440 \times 2^{(n/12)}$$

Na tabela abaixo, vemos as notas musicais, seus símbolos, e a distância em semitons[2] para o lá central (A4).

Símbolo		F3	G3	A4	B4	C4	D4	E4	F4	G4	
Nome	...	fá	sol	lá	si	dó	ré	mi	fá	sol	...
Semitons		-4	-2	0	2	3	5	7	8	10	

[2]Dois semitons equivalem a um tom. No violão, cada casa de uma mesma corda está a um semitom da casa adjacente. No piano, quando há uma tecla preta entre as brancas, há uma distância de um tom entre elas. Quando a tecla preta não está, a distância é de meio tom, ou um semitom.

Você pode notar que a cada 12 semitons, a nota se repete com o dobro da frequência. Chamamos este intervalo entre notas de oitava. Na notação acima, a cada letra indica uma nota diferente, enquanto o número diz a oitava em que ela se encontra.

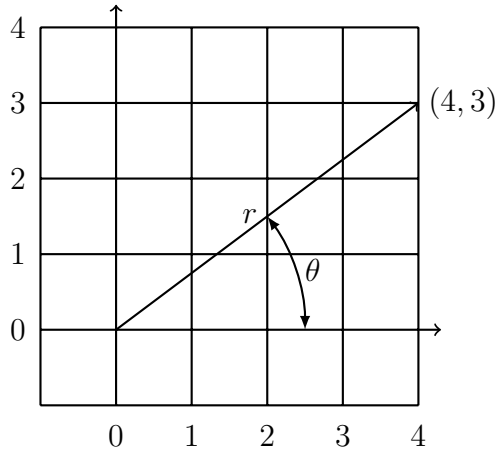
Proposta

Faça uma função `f(n)` que retorne a frequência em Hertz de uma nota que se encontra a n semitons de distância do lá central. Arredonde o resultado para o número inteiro mais próximo usando a função `round(numero, digitos)`.

```
>>> f(0), f(2), f(3)
(440, 494, 523)
>>> f(12) # frequência dobra ...
880
```

1.3 Coordenadas polares

Estamos acostumados a pensar em coordenadas cartesianas na hora de descrever a geometria de um determinado objeto. No entanto, o sistema de coordenadas deve ser escolhido conforme o cenário em que se está trabalhando.



O ponto $(4, 3)$, quando escrito em coordenadas polares, nos dá:

$$r = \sqrt{4^2 + 3^2} = \sqrt{16 + 9} = \sqrt{25} = 5$$

$$\theta = \arctan \frac{3}{4} = 0.6435 \text{ rad} \approx 36.87^\circ$$

Proposta

Construa duas funções: `polar(x, y)` levará um ponto em coordenadas cartesianas (x, y) para a forma polar (r, θ) e `cart(r, theta)`, que fará o caminho contrário.

```
>>> import math
>>> polar(-1, 0)
(1.0, 3.141592653589793)
>>> cart(2, math.pi)
(-2.0, 0.0)
```

CONDICIONAIS

Uma das ferramentas mais simples e poderosas dos computadores é a sua capacidade de tomar diferentes atitudes mediante uma condição.

1.1 Meia-entrada

A Lei Federal nº 12933/2013, também conhecida como Lei da Meia-Entrada, garante o benefício do pagamento de Meia-Entrada para estudantes, pessoas com deficiência e jovens, de baixa renda, com idade entre 15 e 29 anos. Já a Lei Federal nº 10741/2003, mais conhecida como Estatuto do Idoso, as pessoas com idade igual ou superior a 60 anos tem direito à Meia-Entrada para eventos artísticos e de lazer. Aqui no estado do Rio de Janeiro, contamos ainda com a Lei Estadual RJ n° 3.364/00, que garante o benefício a todos os menores de 21 anos.

Proposta
Escreva a função `meia\_entrada` que receba os parâmetros:

1. `idade` (`int`)
2. `estudante` (`bool`)
3. `deficiencia` (`bool`)
4. `baixa\_renda` (`bool`)

informando com `True` ou `False` se a pessoa tem direito ao desconto. normal,

```
>>> meia_entrada(60, False, False, False)
True
>>> meia_entrada(30, True, False False)
False
>>> meia_entrada(20, False, False, False)
True
```

REPETIÇÃO ■

1.1 Sequência de *Collatz*

A sequência de *Collatz* é obtida aplicando sucessivamente a função

$$f(n) = \begin{cases} 3\,n + 1, & \text{se } n \text{ for ímpar} \\ n \div 2, & \text{se } n \text{ for par} \end{cases}$$

Por exemplo, começamos com $n = 26$. Após sucessivas aplicações temos:

$$26 \to 13 \to 40 \to 20 \to 10 \to 5 \to 16 \to 8 \to 4 \to 2 \to 1$$

Isso nos dá uma sequência com 11 números. 40 é maior que 26, mas sua sequência só teria 9 números.

Ainda não se sabe se todos os números induzem uma sequência que termina em 1. No entanto, até agora não foi encontrado um número sequer em que isso não tenha acontecido!

Proposta
Faça uma função que calcule o comprimento da sequência gerada a partir de um número natural n qualquer.

```
>>> collatz(26)
11
>>> collatz(40)
9
>>> collatz(1)
1
```

1.2 *Chatbot* I

Existe uma história corrente de que o grande motor da inteligência artificial em escala industrial é o uso indiscriminado do comando `if` e seus derivados. Não é difícil pensar que através de um número suficientemente grande de condições seja possível abarcar uma grande variedade de comportamentos.

Proposta
Construa uma função chamada `chatbot`

RECURSIVIDADE ■

PARÂMETROS ■

DECORADORES ■

2

ESTRUTURAS, SEQUÊNCIAS E CONDICIONAIS

ENCADEAMENTO DE IDEIAS ■

3

NÚMEROS

- **INTEIROS**
- **REAIS**
- **COMPLEXOS**
- **VETORES**

1

SÉRIES DE EXERCÍCIOS

CÁLCULO ■

I Limites

II Derivadas

III Derivadas parciais

IV Integrais

MÚSICA ■

2

GLOSSÁRIO

BIBLIOGRAPHY

[1] Gilbert Strang. *Linear Algebra and its Applications*. Harcourt Brace, Massachusetts Institute of Technology, 1988.

[2] Advanced RISC Machines Ltd. *ARM-7TDMI Data Sheet*, 1995.

[3] Pedro M. Xavier. *Computação II*. ?, Universidade Federal do Rio de Janeiro, 2021.