Algoritmos e Programação I

Pedro H A Konzen

16 de maio de 2023

Licença

Prefácio

Estas notas de aula fazem uma introdução a algoritmos e programação de computadores. Como ferramenta computacional de apoio, a linguagem computacional Python é utilizada.

Agradeço a todos e todas que de modo assíduo ou esporádico contribuem com correções, sugestões e críticas. :)

Pedro H A Konzen

Conteúdo

Capítulo 1

Introdução

Vamos começar executando nossas primeiras **linhas de código** na linguagem de programação Python. Em um **terminal** Python digitamos

```
1   >>> print('Olá, mundo!')
```

Observamos que `>>>` é o símbolo do `prompt de entrada` e digitamos nossa **instrução** logo após ele. Para executarmos a instrução digitada, teclamos `<ENTER>`. Uma vez executada, o terminal apresentará as seguintes informações

```
1   >>> print('Olá, mundo!')
2   Olá, mundo!
3   >>>
```

Pronto! O fato do símbolo de `prompt de entrada` ter aparecido novamente, indica que a instrução foi completamente executada e o terminal está pronto para executar uma nova instrução.

A **linha de comando** executada acima pede ao computador para imprimir no `prompt de saída` a frase `Olá, mundo!`. O **método** `print` contém instruções para imprimir **objetos** em um dispositivo de saída, no caso, imprime a frase na tela do computador.

Bem! Talvez imprimir no `prompt de saída` uma frase que digitamos no `prompt de entrada` possa parecer um pouco redundante no momento. Vamos

considerar um outro exemplo, vamos computar a soma dos números ímpares entre 0 e 100. Podemos fazer isso como segue

```
1   >>> sum([i for i in range(100) if i%2 != 0])
2   2500
```

Oh! No momento, não se preocupe se não tenha entendido a linha de comando de entrada, ao longo dessas notas de aula isso vai ficando natural. A linha de comando de entrada usa o método **`sum`** para computar a soma dos elementos da **lista** de números ímpares desejada. A lista é construída de forma **iterada** e **indexada** pela **variável** `i`, para `i` no intervalo/faixa de 0 a 99, se o resto da divisão de `i` por 2 não for igual a 0. Ok! O resultado computado for de 2500.

De fato, a soma dos números ímpares de 0 a 100

$$(1, 3, 5, \ldots, 99) \tag{1.1}$$

é a soma dos 50 primeiros elementos da progressão aritmética $a_i = 1 + 2i$, $i = 0, 1, \ldots$, i.e.

$$\sum_{i=0}^{49} a_i = a_0 + a_1 + \cdots + a_{49} \tag{1.2}$$

$$= 1 + 3 + \cdots + 99 \tag{1.3}$$

$$= \frac{50(1 + 99)}{2} \tag{1.4}$$

$$= 2500 \tag{1.5}$$

como já esperado! Em Python, esta última conta pode ser computada como segue

```
1   >>> 50*(1+99)/2
2   2500.0
```

Capítulo 2

Linguagem de Programação

2.1 Computador

[YouTube] | [Vídeo] | [Áudio] | [Contatar]

Um computador[1] é um **sistema computacional** de elementos físicos (**hardware**) e elementos lógicos (**software**).

O **hardware** são suas partes mecânicas, elétricas e eletrônicas como: fonte de energia, teclado, mouse/painel tátil, monitor/tela, dispositivos de armazenagem de dados (HDD, *hard disk drive*; SSD, *solid-state drive*; RAM, *random-access memory*; etc.), dispositivos de processamento (CPU, *central processing unit*, GPU, *graphics processing unit*), conectores de dispositivos externos (microfone, caixa de som, fone de ouvido, USB, etc.), placa mãe, etc..

O **software** é toda a informação processada pelo computador, qualquer código executado e qualquer dado usado nas computações.

[1] Consulte Wikipédia: Computador para uma introdução sobre a história e outras questões sobre computadores.

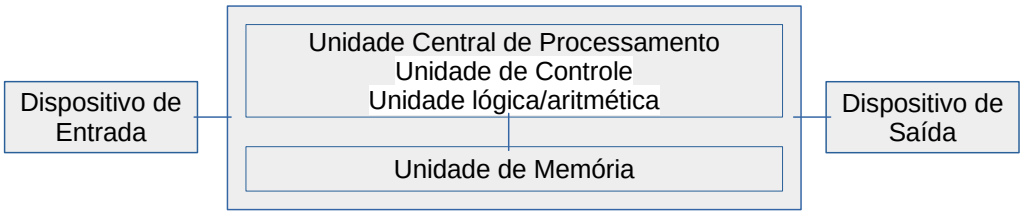


Figura 2.1: Arquitetura de computador de von Neumann.

Os computadores que comumente utilizamos seguem a arquitetura de John von Neumann[2], que consiste em dispositivo(s) de entrada de dados, unidade(s) de processamento, unidade(s) de memória e dispositivo(s) de saída de dados (Figura 2.1).

- **Dispositivos de entrada e saída**

  São elementos do computador que permitem a comunicação humana (usuária(o)) com a máquina.

  - **Dispositivos de entrada**

    São elementos que permitem o fluxo de informação da(o) usuária(o) para a máquina. Exemplos são: teclado, mouse/painel tátil, microfone, etc.

  - **Dispositivos de saída**

    São elementos que permitem o fluxo de informação da máquina para a(o) usuária(o). Exemplos são: monitor/tela, alto-falantes, luzes espia, etc.

- **Unidade central de processamento**

  A **CPU** (do inglês, *Central Processing Unit*) é o elemento de processa as informações e é composta de **unidade de controle**, **unidade lógica e aritmética** e de **memória cache**.

  - **Unidade de controle**

[2]John von Neumann, 1903 - 1957, matemático húngaro, naturalizado estadunidense. Fonte: Wikipédia.

Coordena as execuções do processador: busca e decodifica instruções, lê e escreve no *cache* e controla o fluxo de dados.

- **Unidade lógica/aritmética**

  Executa as instruções operações lógicas e aritméticas, por exemplo: executar a adição, multiplicação, testar se dois objetos são iguais, etc.

- **Memória cache**

  Memória interna da CPU muito mais rápida que as memórias RAM e dispositivos e armazenamento HDD/SSD. É um dispositivo de memória de pequena capacidade e é utilizada como memória de curto prazo e diretamente acessada.

- **Unidades de memória**

  As unidades de memória são elementos que permitem o armazenamento de dados/objetos. Como memória principal tem-se a **ROM** (do inglês, *Read Only Memory*) e a **RAM** (do inglês, *Random Access Memory*) e como memória de massa/secundária tem-se HDD, SSD, entre outras.

- **Memória ROM**

  A memória ROM é utilizada para armazenamento de dados/objetos necessários para dar início ao funcionamento do computador. Por exemplo, é onde a BIOS (dos inglês, *Basic Input/Output System*, Sistema Básico de Entrada e Saída) é armazenada. Ao ligarmos o computador este programa é iniciado e é responsável por fazer o gerenciamento inicial dos diversos dispositivos do computador e carregar o **sistema operacional** (conjunto de programas cuja função é de gerenciar os recursos do computador e controlar a execução de programas).

- **Memória RAM**

  Memória de acesso rápido utilizada para dados/objetos de uso frequente durante a execução de programas. É uma memória volátil, i.e. toda a informação guardada nela é perdida quando o computador é desligado.

- **Memória de massa/secundária**

  Memória de massa ou secundária são usadas para armazenar dados/objetos por período longo. Normalmente, são dispositivos HDD ou SSD,

os dados/objetos são guardados mesmo que o computador seja desligado e contém grande capacidade de armazenagem.

Os **software** são os elementos lógicos de um sistema computacional, são programas de computadores que contém as instruções que gerenciam o **hardware** para a execução de tarefas específicas, por exemplo, imprimir um texto, gravar áudio/vídeo, resolver um problema matemático, etc. Programar é o ato de criar programas de computadores.

2.1.1 Linguagem de programação

As informações fluem no computador codificadas como registros de *bits*[3] (sequência de zeros ou uns). Há registros de instrução e de dados. Programar diretamente por registros é uma tarefa muito difícil, o que levou ao surgimento de linguagens de programação. Uma **linguagem de programação**[4] é um método padronizado para escrever instruções para execução de tarefas no computador. As instruções escritas em uma linguagem são interpretadas e/ou compiladas por um software (interpretador ou compilador) da linguagem que decodifica as instruções em registros de instruções e dados, os quais são efetivamente executados na máquina.

Existem várias linguagens de programação disponíveis e elas são classificadas por diferentes características. Uma **linguagem de baixo nível** (por exemplo, Assembly) é aquela que se restringe às instruções executadas diretamente pelo processador, enquanto que uma **linguagem de alto nível** contém instruções mais complexas e abstratas. Estas contém sintaxe mais próxima da linguagem humana natural e permitem a manipulação de objetos mais abstratos. Exemplos de linguagens de alto nível são: Basic, Java, Javascript, MATLAB, PHP, R, C/C++, Python, etc.

Em geral, não existe uma melhor linguagem, cada uma tem suas características que podem ser mais ou menos adequadas conforme o programa que se deseja desenvolver. Por exemplo, para um site de internet, linguagens como Javascript e PHP são bastante úteis, mas não no desenvolvimento de modelagem matemática e computacional. Nestes casos, C/C++ é uma linguagem mais apropriada por conter várias estruturas de programação que facilitam a modelagem computacional de problemas científicos. Agora, R é

[3] Usualmente de tamanho 64-*bits*.

[4] Código de programação, código de máquina ou linguagem de máquina.

uma linguagem de alto nível com diversos recursos dedicados às áreas de ciências de dados e estatística. Usualmente, utiliza-se mais de uma linguagem no desenvolvimento de programas mais avançados. A ideia é de explorar o melhor de cada linguagem na criação de programas eficientes na resolução dos problemas de interesse.

Nestas notas de aula, Python é a linguagem escolhida para estudarmos algoritmos e programação. Trata-se de uma linguagem de alto nível, **interpretada**, **dinâmica** e **mutiparadigma**. Foi lançada por Guido van Rossum[5] em 1991 e, atualmente, é desenvolvida de forma comunitária, aberta e gerenciada pela ONG Python Software Foundation. A linguagem foi projetada para priorizar a legibilidade do código. Parte da filosofia da linguagem é descrita pelo poema The Zen of Python. Pode-se lê-lo pelo *easter egg* Python:

```
1   >>> import this
```

- Linguagem interpretada

  Python é uma linguagem interpretada. Isso significa que o **código-fonte** escrito em linguagem Python é interpretado por um programa (interpretador Python). Ao executar-se um código, o interpretador lê uma linha do código, decodifica-a como registros para o processador que os executa. Executada uma linha, o interpretador segue para a próxima até o código ter sido completadamente executado.

- Linguagem compilada

  Em uma linguagem compilada, como C/C++, há um programa chamado de **compilador** (em inglês, *compiler*) e outro de **ligador** (em inglês, *linker*). O primeiro, cria um programa-objeto a partir do código e o segundo gerencia sua ligação com eventuais bibliotecas computacionais que ele possa depender. O programa-objeto (também chamado de executável) pode então ser executado pela máquina.

Em geral, a execução de um programa compilado é mais rápida que a de um código interpretado. De forma simples, isso se deve ao fato de que nesse a interpretação é feita toda de uma vez e não precisa ser refeita na execução de cada linha de código, como no segundo caso. Por outro lado, a compilação de códigos-fonte grandes pode ser bastante demorada fazendo mais sentido

[5]Guido van Rossum, 1956-, matemático e programador de computadores holandês. Fonte: Wikipédia.

quando ele é compilado uma vez e o programa-objeto executado várias vezes. Além disso, linguagens interpretadas podem usar bibliotecas de programas pré-compiladas. Com isso, pode-se alcançar um bom balanceamento entre tempo de desenvolvimento e de execução do código.

O interpretador Python também pode ser usado para compilar o código para um arquivo **bytecode**, este é executado muito mais rápido do que o código-fonte em si, pois as interpretações necessárias já foram feitas. Mais adiante, vamos estudar isso de forma mais detalhada.

- Linguagem de tipagem dinâmica

  Python é uma linguagem de tipagem dinâmica. Nela, os dados não precisam ser explicitamente tipificados no código-fonte e o interpretador os tipifica com base em regras da própria linguagem. Ao executar operações com os dados, o interpretador pode alterar seus tipos de forma dinâmica.

- Linguagem de tipagem estática

  C/C++ é um exemplo de uma linguagem de tipagem estática. Em tais linguagens, os dados devem ser explicitamente tipificados no código-fonte com base nos tipos disponíveis. A retipificação pode ocorrer, mas precisa estar explicitamente definida no código.

Existem vários **paradigmas de programação** e a linguagem Python é multiparadigma, i.e. permite a utilização de mais de um no código-fonte. Exemplos de paradigmas de programação são: **estruturada**, **orientada a objetos**, **orientada a eventos**, etc.. Na maior parte destas notas de aulas, vamos estudar algoritmos para linguagens de programação estruturada. Mais ao final, vamos introduzir aspectos de linguagens orientada a objetos. Estes são paradigmas de programação fundamentais e suas estruturas são importantes na programação com demais paradigmas disponíveis em programação de computadores.

2.1.2 Instalação e execução

Python é um **software aberto**[6] e está disponível para vários sistemas operacionais (Linux, macOS, Windows, etc.) no seu site oficial

[6] Consulte a licença de uso em `https://docs.python.org/3/license.html`.

https://www.python.org/

Também, está disponível (gratuitamente) na loja de aplicativos dos sistemas operacionais mais usados. Esta costuma ser a forma mais fácil de instalá-lo na sua máquina, consulte a loja de seus sistema operacional. Ainda, há plataformas e IDEs[7] Python disponíveis, consulte, como por exemplo, Anaconda.

A execução de um código Python pode ser feita de várias formas.

- **Execução iterativa via terminal**

  Em terminal Python pode-se executar instruções/comandos de forma iterativa. Por exemplo:

```
1     >>> print('Olá, mundo!')
2     Olá, mundo!
3     >>>
```

  O símbolo `>>>` denota o **prompt de entrada**, onde uma instrução Python pode ser digitada. Após digitar, o comando é executada teclando `<ENTER>`. Caso o comando tenha alguma saída de informação, como no caso acima, esta aparecerá, por padrão, no **prompt de saída**, logo abaixo a linha de comando executada. Um novo símbolo de prompt de entrada aparece ao término da execução anterior.

- **Execução de um** *script*

  Para códigos com várias linhas de instruções é mais adequado utilizar um aquivo de *script* Python. Usando-se um editor de texto ou um IDE ditam-se as linhas de comando em um arquivo `.py`. Então, *script* pode ser executado em um terminal de seu sistema operacional utilizando-se o interpretador Python. Por exemplo, assumindo que o código for salvo do arquivo `path_to_arq/arq.py`, pode-se executá-lo em um terminal do sistema com

```
1     $ python3 path_to_arq/arq.py
```

  IDEs para Python fornecem uma ambiente integrado, contendo um campo para escrita do código e terminal Python integrado. Consulte, por exemplo, o IDE Spyder:

[7] IDE, do inglês, *Integrated Development enviroment*, ambiente de desenvolvimento integrado

https://www.spyder-ide.org/

- **Execução em um** *notebook*

  Notebooks Python são uma boa alternativa para a execução de códigos em um ambiente colaborativo/educativo. Por exemplo, Jupyter é um notebook que roda em navegadores de internet. Sua estrutura e soluções também são encontradas em notebooks online (de uso gratuito limitado) como Google Colab e Kaggle.

2.1.3 Exercícios

Exercício 2.1.1. Verifique qual a versão do sistema operacional que está utilizado em seu computador.

Exercício 2.1.2. Verifique os seguintes elementos de seu computador:

a) CPUs

b) Placa(s) gráfica(s)

c) Memória RAM

d) Armazenamento HDD/SSD.

Exercício 2.1.3. Verifique como entrar na `BIOS` de seu computador. Atenção! Não faça e salve nenhuma alteração, caso não saiba o que está fazendo. Modificações na `BIOS` podem impedir que seu computador funcione normalmente, inclusive, impedir que você inicialize seu sistema operacional.

Exercício 2.1.4. Instale Python no seu computador (caso ainda não tenha feito) e abra um terminal Python. Nele, escreva uma linha de comando que imprima no prompt de saída a frase "Olá, meu Python!".

Exercício 2.1.5. Instale o Spyder no seu computador (caso ainda não tenha feito) e use-o para escrever o seguinte *script*

```
1 import math as m
2 print(f'Número pi = {m.pi}')
3 print(f'Número de Euler e = {m.e}')
```

Também, execute o *script* diretamente em um terminal de seu sistema operacional.

Exercício 2.1.6. Use um *notebook* Python para escrever e executar o código do exercício anterior.

2.2 Algoritmos e Programação

Programar é criar um programa (um *software*) para ser executado em computador. Para isso, escreve-se um código em uma linguagem computacional (por exemplo, em Python), o qual é interpretado/compilado para gerar o programa final. Linguagens computacionais são técnicas, utilizam uma sintaxe simples, precisa e sem ambiguidades. Ou seja, para criarmos um programa com um determinado objetivo, precisamos escrever um código computacional técnico, que siga a sintaxe da linguagem escolhida e sem ambiguidades.

Um **algoritmo** pode ser definido uma sequencia ordenada e sem ambiguidade de passos para a resolução de um problema. Por exemplo, o cálculo da área de um triângulo de base e altura definida por ser feita com o seguinte algoritmo:

1. Informe o valor da base b.
2. Informe o valor da base h.
3. $a \leftarrow \dfrac{b \cdot h}{2}$.
4. Imprima o valor de a.

Algoritmos para a programação são pensados para serem facilmente transformados em códigos computacionais. Por exemplo, o algoritmo acima pode ser escrito em Python como segue:

```
1 b = float(input('Informe o valor da base.\n'))
2 h = float(input('Informe o valor da altura.\n'))
3 # cálculo da área
4 a = b*h/2
5 print(f'Área = {a}')
```

Para criar um programa para resolver um dado problema, começamos desenvolvendo um algoritmo para resolvê-lo, este algoritmo é implementado na linguagem computacional escolhida, a qual gera o programa final. Aqui, o passo mais difícil costuma ser o desenvolvimento do algoritmo. Precisamos pensar em como podemos resolver o problema de interesse em uma sequência de passos ordenada e sem ambiguidades para que possamos implementá-los em computador.

Um algoritmo deve ter as seguintes propriedades:

- Cada passo deve estar bem definido, i.e. não pode conter ambiguidades.
- Cada passo deve contribuir de forma efetiva na solução do problema.
- Deve ter número finito de passos que podem ser computados em um tempo finito.

Observação 2.2.1. A primeira pessoa a publicar um algoritmo para programação foi Augusta Ada King[8]. O algoritmo foi criado para computar os números de Bernoulli[9].

2.2.1 Fluxograma

Fluxograma é uma representação gráfica de um algoritmo. Entre outras, usam-se as seguintes formas para representar tipos de ações a serem executadas:

- **Terminal**: início ou final do algoritmo.

- **Linha de fluxo**: direciona para a próxima execução.

[8] Augusta Ada King, 1815 - 1852, matemática e escritora inglesa. Fonte: Wikipédia.
[9] Jacob Bernoulli, 1655-1705, matemático suíço. Fonte: Wikipédia.

- **Entrada**: leitura de informação/dados.

- **Processo**: ação a ser executada.

- **Decisão**: ramificação do processamento baseada em uma condição.

- **Saída**: impressão de informação/dados.

Exemplo 2.2.1. O método de Heron[10] é um algoritmo para o cálculo aproximado da raiz quadrada de um dado número x, i.e. $\sqrt{x}$. Consiste na iteração

$$s^{(0)} = \text{approx. inicial}, \tag{2.1}$$

$$s^{(i+1)} = \frac{1}{2}\left(s^{(i)} + \frac{x}{s^{(i)}}\right), \tag{2.2}$$

para $i = 0,1,2,\ldots,n$, onde n é o número de iterações calculadas.

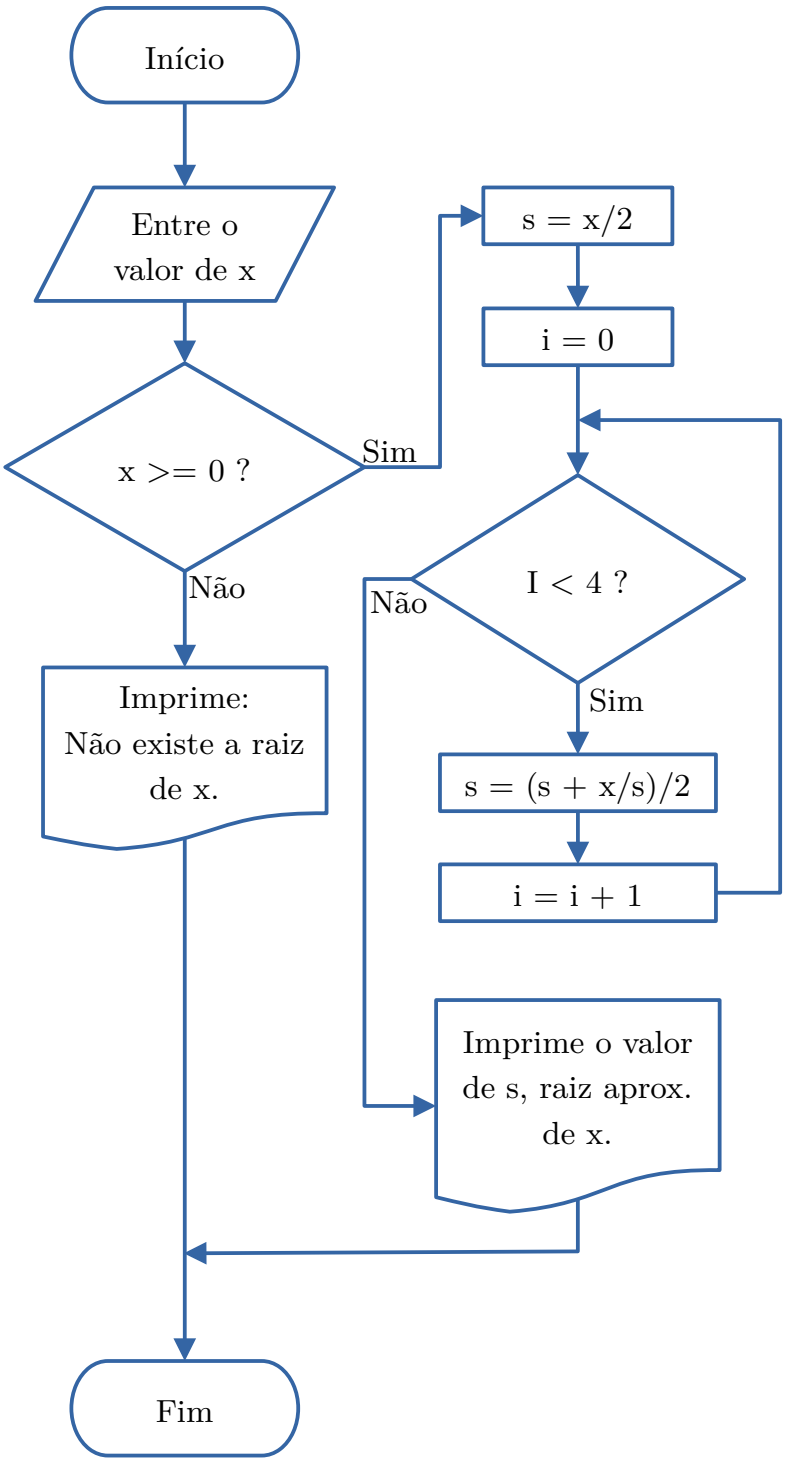
[10]Heron de Alexandria, 10 - 80, matemático e inventor grego. Fonte: Wikipédia.

Na sequência, temos um algoritmo e seus fluxograma e código Python para computar a quarta aproximação de $\sqrt{x}$, assumindo $s^{(0)} = x/2$ como aproximação inicial.

- **Algoritmo**

1. Entre o valor de x.
2. Se $x \geq 0$, faça:
   (a) $s \leftarrow x/2$
   (b) Para $i = 0,1,2,3$, faça:
      i. $s \leftarrow (s + x/s)/2$.
   (c) Imprime o valor de s.
3. Senão, faça:
   (a) Imprime mensagem "Não existe!".

- **Fluxograma**



- **Código Python**

Código 2.1: metHeron.py

```
1 x = float(input('Entre com o valor de x: '))
2 if (x >= 0.):
3     s = x/2
4     for i in range(4):
5         s = (s + x/s)/2
6     print(f'Raiz aprox. de x = {s}')
7 else:
8     print(f'Não existe!')
```

Algoritmos escritos em uma forma próxima de uma linguagem computacional são, também, chamados de **pseudocódigos**. Na prática, pseudocódigos e fluxogramas são usados para apresentar uma forma mais geral e menos detalhada de um algoritmo. Usualmente, sua forma detalhada é escrita diretamente em uma linguagem computacional escolhida.

2.2.2 Exercícios

Exercício 2.2.1. Escreva um algoritmo/pseudocódigo e um fluxograma correspondente para o calcular a média aritmética entre dois números x e y dados. Como desafio, tente escrever um código Python baseado em seu algoritmo.

Exercício 2.2.2. Escreva um algoritmo/pseudocódigo e um fluxograma correspondente para o calcular a área de um quadrado de lado l dado. Como desafio, tente escrever um código Python baseado em seu algoritmo.

Exercício 2.2.3. Escreva um algoritmo/pseudocódigo e um fluxograma correspondente para o calcular a área de um retângulo de lados a, b dados. Como desafio, tente escrever um código Python baseado em seu algoritmo.

Exercício 2.2.4. Escreva um algoritmo/pseudocódigo e um fluxograma correspondente para o calcular triângulo retângulo de hipotenusa h e um dos lados l dados. Como desafio, tente escrever um código Python baseado em seu algoritmo.

Exercício 2.2.5. Escreva um algoritmo/pseudocódigo e um fluxograma correspondente para o calcular o zero de uma função afim

$$f(x) = ax + b \tag{2.3}$$

dados, os coeficientes a e b. Como desafio, tente escrever um código Python baseado em seu algoritmo.

Exercício 2.2.6. Escreva um algoritmo/pseudocódigo e um fluxograma correspondente para o calcular as raízes reais de um polinômio quadráticos

$$p(x) = ax^2 + bx + c \tag{2.4}$$

dados, os coeficientes a, b e c. Como desafio, tente escrever um código Python baseado em seu algoritmo.

Exercício 2.2.7. A Série Harmônica é defina por

$$\sum_{k=1}^{\infty} \frac{1}{k} := \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots \tag{2.5}$$

Escreva um algoritmo/pseudocódigo e um fluxograma corresponde para calcular o valor da série harmônica truncada em $k = n$, com n dado. Ou seja, dado n, o objetivo é calcular

$$\sum_{k=1}^{n} \frac{1}{k} := \frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{n}. \tag{2.6}$$

Exercício 2.2.8. O número de Euler[11] pode ser definido pela série

$$e := \sum_{k=0}^{\infty} \frac{1}{k!} \tag{2.7}$$

$$= \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots \tag{2.8}$$

[11] Leonhard Paul Euler, 1707-1783, matemático e físico suíço. Fonte: Wikipédia.

Escreva um algoritmo/pseudocódigo e um fluxograma corresponde para calcular o valor aproximado de e dado pelo truncamento da série em $k = 4$, i.e. o objetivo é de calcular

$$e \approx \sum_{k=0}^{4} \frac{1}{k!} \tag{2.9}$$

$$= \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} \tag{2.10}$$

$$= \frac{1}{1} + \frac{1}{1} + \frac{1}{1 \cdot 2} + \frac{1}{1 \cdot 2 \cdot 3} + \frac{1}{1 \cdot 2 \cdot 3 \cdot 4}. \tag{2.11}$$

Resposta dos Exercícios

Exercício 2.1.1. Dica: Em Linux, `$ uname --all` ou `$ cat /etc/version`.

Exercício 2.1.2. Dica: Em Linux: `$ lshw`

Exercício 2.1.3. Dica: cada computador tem sua forma de acessar a `BIOS`. Verifique o manual ou busque na internet pela marca e modelo de seu computador.

Exercício 2.1.4.

```
1 >>> print('Olá, meu Python!')
2 Olá, meu Python!
3 >>>
```

Exercício 2.1.6. Dica: use um notebook online Google Colab, Kaggle ou Jupyter.

Bibliografia

[1] S. L. Banin. *Python 3 - Conceitos e Aplicações - Uma Abordagem Didática*. Saraiva, São Paulo, 2021.

[2] T. Cormen. *Algoritmos - Teoria e Prática*. Grupo GEN, São Paulo, 2012.

[3] T. Cormen. *Desmitificando Algoritmos*. Grupo GEN, São Paulo, 2021.

[4] J. Grus. *Data Science do Zero*. Alta Books, Rio de Janeiro, 2021.

[5] J. A. Ribeiro. *Introdução à Programação e aos Algoritmos*. LTC, São Paulo, 2021.

[6] R. Wazlawick. *Introdução a Algoritmos e Programação com Python - Uma Abordagem Dirigida por Testes*. Grupo GEN, São Paulo, 2021.