

---

**Programação em Python**  
*Fundamentos e Resolução de Problemas*

Ernesto Costa

---

Versão 1.0

**ENUNCIADOS**

© 2015, Ernesto Costa.  
Departamento de Engenharia Informática, Universidade de Coimbra  
Polo II - Pinhal de Marrocos, 3030-290 Coimbra  
<http://ernesto.dei.uc.pt>

# Introdução

If I had an hour to solve a problem I'd spend 55 minutes thinking about the problem and 5 minutes thinking about solutions.

---

Albert Einstein

Apresentam-se neste texto os enunciados dos problemas do livro *Programação em Python: fundamentos e resolução de problemas*, publicado pela FCA.

Muitos destes exercícios não são originais, encontrando-se várias soluções para eles, seja em Python ou noutra linguagem, em diferentes textos. A sua utilidade é, em muitos casos, apenas didática. Muitos deles foram sendo propostos aos alunos da disciplina de **Introdução à Programação e Resolução de Problemas (IPRP)** do curso de Engenharia Informática da Universidade de Coimbra ao longo dos anos. Aos meus alunos e a todos os docentes que ao longo do ano me têm auxiliado nesta disciplina o meu agradecimento. Um agradecimento especial é devido àqueles que, mais recentemente e de modo mais continuado, me têm ajudado: Filipe Araújo, Tiago Baptista, Rui Lopes, Nuno Lourenço, Bernardete Ribeiro e João Vilela.

Existe um blogue associado à disciplina de IPRP, acessível através da ligação <http://programacaocompython.blogspot.com>. Aí o leitor encontrará muitos mais exemplos resolvidos, alguns conceitos explicados e outros elementos relevantes para a programação com Python.

Terei todo o gosto em receber a opinião dos leitores, seja na forma de soluções para os problemas seja na forma de correcções de eventuais erros. Posso ser contactado por correio electrónico para: [ernesto@dei.uc.pt](mailto:ernesto@dei.uc.pt). Boas soluções!

Ernesto Costa



## Parte I

# Programação Procedimental



# Capítulo 1

## Introdução

### Exercício 1.1 MF

Determine como é que na sua plataforma se pode por o interpretador Python a correr.

### Exercício 1.2 MF

Python pode ser usado como uma simples calculadora. Verifique o resultado das seguintes computações.

1.  $2 + 4$
2.  $40 * 300$
3.  $1/2$
4.  $1.0/2$
5.  $1.0 // 2$
6.  $20e30 * 4$
7.  $20e50 * 20e50$
8.  $7 \% 5$
9.  $(5 + 2j) + (3 + 4j)$
10.  $(5 + 2j) * (3 + 4j)$
11.  $(5 + 2j) / (3 + 4j)$

**Exercício 1.3 MF** DONE

A galáxia Andrómena está a 2,9 milhões de anos-luz da Terra. Um ano luz equivale  $9.459 \times 10^{12}$  quilómetros. A quantos quilómetros se encontra a galáxia da Terra?

**Exercício 1.4 MF** DONE

Calcule o número de segundos que existe num ano *normal*.

**Exercício 1.5 MF**

Suponha que tem uma sala rectangular de dimensão  $8 \times 6$ . Admitindo que quer cobrir o chão com tijoleira de  $2 \times 2$ , calcule o número de unidades de que vai precisar.

**Exercício 1.6 MF**

Escolha objectos numéricos de tipos diferentes e inspeccione os seus três atributos.

**Exercício 1.7 MF** DONE

A área de um triângulo é igual a metade do produto do comprimento de um dos lados pela distância ao vértice oposto medida perpendicularmente. Diga como podia usar Python para calcular o valor concreto da área de um triângulo conhecidos aqueles valores.

**Exercício 1.8 F** DONE

Você não gosta de ser enganado e é muito meticoloso. Quando foi comer à sua hamburgeria preferida foi confrontado com uma nova forma: agora o hambúrguer é um quadrado de lado  $7.62\text{ cm}$  (eu avisei que você era meticoloso!). Para saber se devia protestar (sim porque você adora uma boa luta . . . ), procurou comparar com o formato antigo, bem redondo como uma circunferência de diâmetro  $8.89\text{ cm}$  (precisa que eu insista em como é meticoloso?). Eu que eu quero saber é se você, consumidor compulsivo de carne picada, tem ou não razões para protestar devido ao design do novo hambúrguer.

**Exercício 1.9 F** DONE

O Índice de Massa Corporal é dado pela fórmula:

$$IMC = \frac{\text{peso}(kg)}{\text{altura}^2(m^2)}$$

Refaça os cálculos da secção 1.5 adaptando o exemplo do peso para o caso do IMC.

### Exercício 1.10 F DONE

Escreva um programa que lhe permita converter uma temperatura na escala Celsius ( $T_c$ ) na escala Fahrenheit ( $T_f$ ), baseando-se na fórmula:

$$T_f = \frac{9}{5} \cdot T_c + 32$$

### Exercício 1.11 F DONE

Escreva um programa que lhe permita calcular o volume de um cone, conhecidos o raio da base  $r$  e a altura  $h$ . O volume pode ser calculado pela fórmula:

$$V = \frac{\pi \cdot r^2 \cdot h}{3}$$

### Exercício 1.12 F DONE

Suponha que tem o seguinte polinómio:  $x^4 + x^3 + 2x^2 - x$ . Socorrendo-se da linguagem Python calcule o valor do polinómio nos seguintes pontos:

1.  $x = 1.1$
2.  $x = 5$
3.  $x = \frac{2}{3}$

### Exercício 1.13 F

Importe o módulo `math` e experimente as várias funções fornecidas. O que acontece se tentar calcular a raiz quadrada de um inteiro negativo.

### Exercício 1.14 F DONE

Suponha que quer trocar uma certa quantidade de euros por dólares americanos, conhecida a taxa de câmbio. Diga como pode resolver o problema socorrendo-se de Python para um caso concreto. Se por acaso quer uma solução genérica, em que medida a sua solução anterior lhe resolve a questão?

### Exercício 1.15 F DONE

Suponha que tem uma certa quantidade de garrafas vazias de capacidade 5, 1.5, 0.5 e 0.25 litros. Admita que tem um número ilimitado de garrafas

de cada tipo. Dado um certa quantidade de água que pretende guardar em garrafas, como resolveria o problema minimizando o **número** de garrafas a usar. Como poderia usar o computador para lhe calcular o número de garrafas de cada tipo necessárias?

### Exercício 1.16 M

Volte ao exemplo do jogo da adivinha do número e procure definir novas variantes para o jogo, de modo a torná-lo mais interessante e realista.

### Exercício 1.17 Módulo math M DONE

Um ponto no plano pode ser identificado pelas suas coordenadas cartesianas (par  $(x,y)$ ) ou pelas coordenadas polares (par  $((r,\theta))$ ). Escreva um programa que converte das coordenadas cartesianas para as polares. A relação entre os dois tipos de representação é dada pelas fórmulas que se apresentam na figura 1.1.

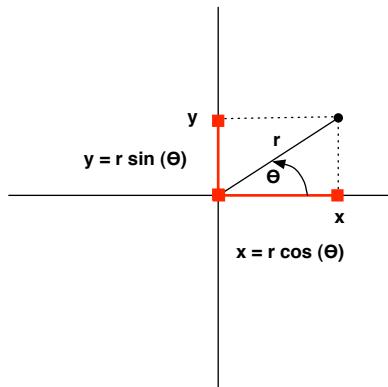


Figura 1.1: De cartesianas a polares

### Exercício 1.18 Módulo math F DONE

Johannes Kepler<sup>1</sup> foi um cientista que no início do século XVII formulou três leis relativas ao movimento dos planetas em torno do Sol, com base nas observações do astrónomo Tycho Brahe. A primeira lei, estipula que as órbitas são elipsoidais, com o Sol num dos focos; a segunda lei, estabelece que a linha que une o planeta ao Sol varre áreas iguais durante intervalos de tempo iguais; a terceira lei, diz que o quadrado do período orbital de um planeta é directamente proporcional ao cubo do semi-eixo maior da sua órbita, ou seja que  $p^2 = a^3$ . Sendo a órbita uma elipse a distância de um planeta ao Sol

<sup>1</sup>[http://en.wikipedia.org/wiki/Johannes\\_Kepler](http://en.wikipedia.org/wiki/Johannes_Kepler)

varia. É usual usar como medida dessa distância o valor médio das distâncias máxima e mínima. Esse valor é dado em **Unidades Astronómicas**, AU. 1 AU é igual ao valor do semi-eixo maior da órbita da Terra em volta do sol e vale  $149.597890 \times 10^6 \text{ km}$ . Por exemplo, Mercúrio está a 0.387 AU, enquanto Neptuno está a 30.06 AU.

Escreva um programa baseado na terceira lei de Kepler, que permita calcular o período, em anos, da órbita de um planeta, conhecida a sua distância ao Sol em AUs. Para o auxiliar a verificar a correcção do seu programa deve consultar [http://en.wikipedia.org/wiki/Attributes\\_of\\_the\\_largest\\_solar\\_system\\_bodies](http://en.wikipedia.org/wiki/Attributes_of_the_largest_solar_system_bodies), onde encontrará os valores de teste de que necessita.

### Exercício 1.19 M DONE

No seguimento de Johannes Kepler, Isaac Newton, publicou em 1687 o seu livro *Principia Mathematica*, onde formulou a sua teoria sobre a gravidade. No contexto da teoria, Newton generaliza a terceira lei de Kepler, que deixa de estar limitada ao sistema solar:

$$p^2 = \frac{4\pi^2}{G(M_1 + M_2)} a^3$$

sendo que  $G = 6.67 \times 10^{-11} \text{ N m}^2/\text{Kg}^2$  é a constante gravitacional, e  $M_1$  e  $M_2$  a massa de dois objectos no espaço. Usando esta formulação, escreva um programa para resolver o problema de determinar o período orbital de qualquer corpo que orbita em volta de qualquer estrela. Para facilitar a nossa vida vamos escolher uma estrela concreta: **Gliese 581** (ver dados em [http://pt.wikipedia.org/wiki/Gliese\\_581](http://pt.wikipedia.org/wiki/Gliese_581)).



# Capítulo 2

## Visões I

### Exercício 2.1 F DONE

Neste capítulo desenvolvemos um programa para desenhar polígonos regulares. No centro do programa está um ciclo que repete um certo número de vezes duas operações: avançar e rodar. Retome o programa e faça variar estes três elementos por forma a obter outro tipo de figuras. A figura 2.1 ilustra a situação de repetir 5 vezes o avanço de 100 unidades e uma rotação de 144 graus.



Figura 2.1: Uma estrela

### Exercício 2.2 F DONE

Adapte a ideia do exercício anterior para desenhar uma figura semelhante a 2.2. Procure implementar uma solução que possibilite variantes da figura apresentada.

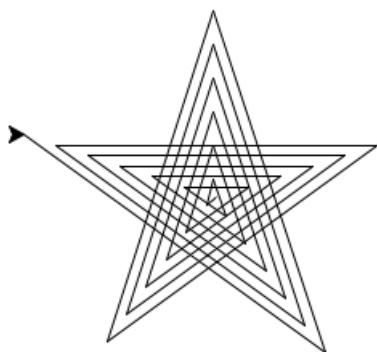


Figura 2.2: Espiral de Estrelas

**Exercício 2.3 F DONE**

Faça um programa que lhe permita simular um passeio aleatório (*random walk*). A figura 2.3 ilustra o que se pretende. Procure alterar a cor de cada segmento de modo aleatório.

**Exercício 2.4 F DONE**

No texto mostrámos como se podia desenhar uma circunferência usando a função **poligono\_regular**. Mas o desenho é sempre o mesmo, isto é, não podemos controlar o **raio** da circunferência. Diga como pode usar na mesma a função **poligono\_regular** mas tendo em conta o valor do raio.

**Exercício 2.5 F**

Usando o método descrito no capítulo construa diferentes formas para a tartaruga e use-as.

**Exercício 2.6 M**

O módulo **turtle** tem um método **circle** pré-definido que lhe permite desenhar circunferências. O método tem um parâmetro obrigatório que é o raio da circunferência. Adicionalmente podemos indicar qual a extensão que vamos desenhar, tudo (por defeito) ou apenas um arco. Como a circunferência é aproximada através a um polígono regular inscrito, existe um terceiro parâmetro que torna possível o uso de **circle** para desenhar polígonos regulares. Explore essa facilidade para desenhar diversos polígonos

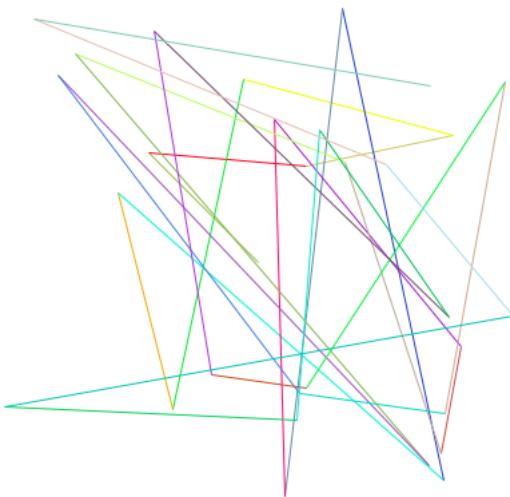


Figura 2.3: Sem rei nem roque

regulares.

**Exercício 2.7 M** DONE

Utilize o método `circle` para desenhar um smiley do tipo do da figura 2.4.

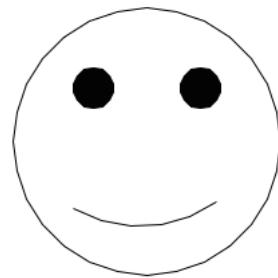


Figura 2.4: Cara sorridente

**Exercício 2.8 M**

O método `forward` permite movimentar uma tartaruga no sentido da sua

orientação actual. Queremos alterar o seu funcionamento por forma que a tartaruga nunca se afaste mais do que um certo valor, definido pelo utilizador, do seu ponto de partida. Caso chegue ao limite do espaço que pode visitar a tartaruga pode, em alternativa:

1. alterar a sua orientação ficando apontada para a posição de partida e continuar a movimentar-se;
2. como no caso anterior mas permitindo uma pequena variação aleatória na nova orientação da tartaruga.

Socorra-se dos métodos `distance`, `towards` e `setheading` do módulo `turtle` para implementar dois programas que resolvam as duas situações

### **Exercício 2.9** M DONE

Existe um método no módulo `turtle` que permite à tartaruga deixar um rastro dos lugares por onde andou. Esse comando chama-se `stamp`. Veja no manual da linguagem como funciona e escreva um programa que lhe permita desenhar o que a figura 2.5 ilustra. Procure que a sua solução seja genérica.

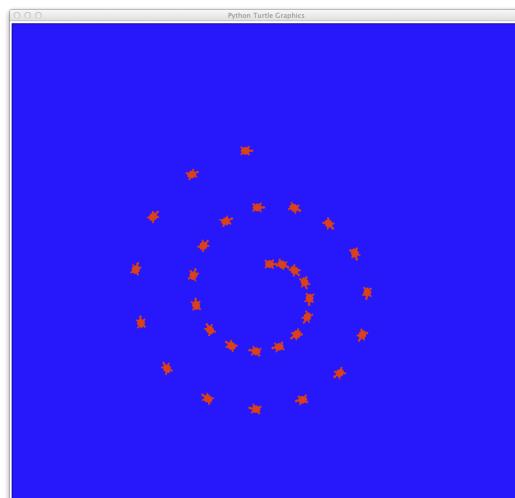


Figura 2.5: Deixar rasto

### **Exercício 2.10** M DONE

Pretende-se escrever um programa que construa o desenho da figura 2.6.

### **Exercício 2.11** M DONE

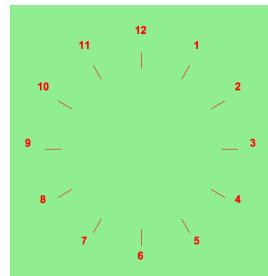


Figura 2.6: Um relógio sem ponteiros...

Desenvolva um programa que lhe permita desenhar a figura 2.7. Deve basear-se apenas na possibilidade de desenhar triângulos.

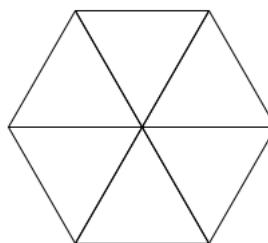


Figura 2.7: Um hexágono fatiado

### Exercício 2.12 M DONE

Escreva um programa que lhe permita desenhar um figura como a ilustrada em 2.8. Procure que a sua solução seja modular e potencia a reutilização de código.

### Exercício 2.13 M DONE

Desenvolva um programa que lhe permita desenhar a figura 2.9.

### Exercício 2.14 D DONE

Desenvolva um programa que lhe permita desenhar quadrados concêntricos como, por exemplo, indicado na figura 2.10.

### Exercício 2.15 D DONE

Desenvolva um programa que lhe permita desenhar a figura 2.11. Se reparar bem a figura é formada por quadrados cujos lados têm dimensão

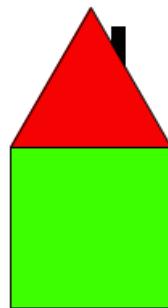


Figura 2.8: Uma casa portuguesa...



Figura 2.9: Um esse deitado

diferentes e os ângulos iniciais têm orientações diferentes.

### Exercício 2.16 M DONE

Todos conhecemos o símbolo dos Jogos Olímpicos. A figura 2.12 mostra um exemplo.

Socorrendo-se do módulo **turtle** implemente um programa que permita desenhar o símbolo. Procure que a sua solução seja modular por forma a poder eventualmente aproveitar partes do seu programa para resolver outros problemas.

### Exercício 2.17 D DONE

Existem vários sinais universais, alguns que nos ajudam a evitar lugares indesejáveis. É o caso do sinal de presença de radioactividade que a figura 2.13 ilustra.

Socorrendo-se do módulo **turtle** implemente um programa que permita desenhar o símbolo. Procure que a sua solução seja modular por forma a poder eventualmente aproveitar partes do seu programa para resolver outros

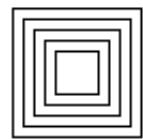


Figura 2.10: Tanto quadrado...

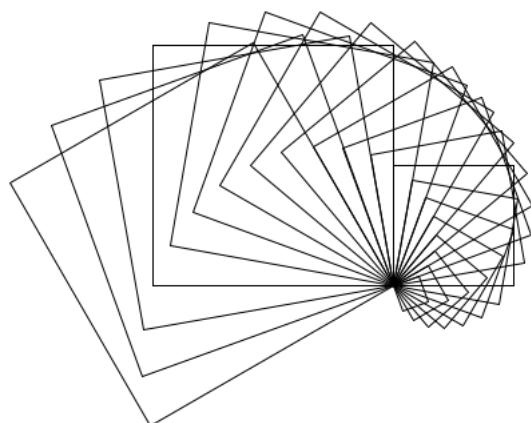


Figura 2.11: Parece um nautilus...

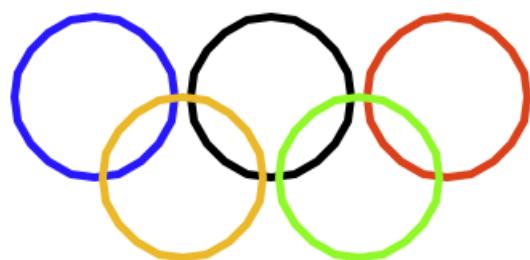


Figura 2.12: O símbolo dos Jogos Olímpicos

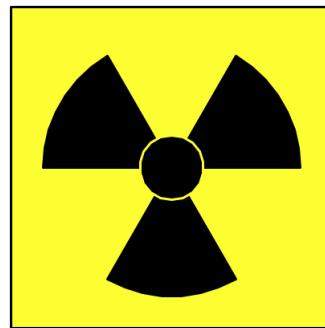


Figura 2.13: Radioactividade

problemas.

**Exercício 2.18 D** DONE

Desenvolva um programa que lhe permita desenhar a figura 2.14. Em que medida a sua solução é modular permitindo a reutilização de partes do código para resolver outros problema ou variantes desta figura?



Figura 2.14: Calma...

**Exercício 2.19 D**

Conhece seguramente o Jogo do Galo: dois jogadores colocam alternadamente uma marca num tabuleiro 3 X 3, ganhando aquele que primeiro colocar três marcas suas em linha. Vamos usar o módulo `turtle` para implementar uma versão muito simples do jogo. Usaremos uma grelha de pontos inicialmente todos pretos. Cada vez que um jogador joga, e isso faz-se indicando uma posição na grelha em resposta a um pedido do programa, é colocada a

sua marca na posição indicada, sendo que os jogadores se distinguem pela cor da marca. Como ainda não aprendemos muito vamos ter que confiar nos jogadores: eles indicam sempre umas posição da grelha ainda não ocupadas. Por outro lado, não é responsabilidade do programa determinar se há um vencedor. A figura 2.15 mostra um momento do jogo.

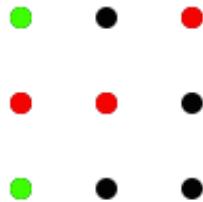


Figura 2.15: Jogo do Galo



# Capítulo 3

## Objectos (I)

### Exercício 3.1 MF

Arranque o interpretador **Python** e associe um nome com um objecto do tipo inteiro. Recorra ao comando `help` para saber mais coisas sobre o objecto. Observe o resultado e tire conclusões.

### Exercício 3.2 F DONE

Escreva um programa que calcula a área de um triângulo por recurso à fórmula de Heron. Neste caso, se o triângulo tiver como lados  $a$ ,  $b$  e  $c$  a área é dada por:

$$\text{área} = \sqrt{s \times (s - a) \times (s - b) \times (s - c)}$$

com:

$$s = \frac{a + b + c}{2}$$

### Exercício 3.3 F DONE

Suponha que quer colocar uma escada encostada a uma parede de sua casa por forma que ela alcance uma dada altura  $alt$ . Por razões de segurança a escada deve fazer um dado ângulo  $ang$  com o solo. Escreva um programa que determine o comprimento  $comp$  da escada. A relação entre as três variáveis é dada por:

$$comp = \frac{alt}{\operatorname{seno}(ang)}$$

O cálculo do seno é feito em radianos mas admita que o ângulo é dado pelo utilizador em graus. A relação é dada por:

$$\text{radianos} = \frac{\pi}{180} \times \text{graus}$$

### Exercício 3.4 F DONE

O valor do batimento cardíaco máximo tem sido objecto de vários estudos, existindo várias fórmulas que dão o seu valor médio. Uma delas é:

$$163 + 1.16 * \text{idade} - 0.018 * \text{idade}^2$$

Desenvolva um programa que dada a idade calcule o valor médio do batimento cardíaco máximo.

### Exercício 3.5 F DONE

Admitamos que colocamos uma certa quantidade de dinheiro a render. A fórmula que nos permite calcular o valor ao fim de vários anos, conhecida a taxa de juro fixa é:

$$v * (1 + t)^a$$

Escreva um programa que conhecido o valor inicial ( $v$ ), a taxa de juro ( $t$ ), e os anos decorridos ( $a$ ), calcula o valor ao fim desses anos. Use este programa para saber ao fim de quanto tempo consegue duplicar o seu dinheiro.

### Exercício 3.6 F

Problema semelhante a 3.5 só que agora a taxa pode ser composta várias vezes ao ano. Neste caso a fórmula passa a ser:

$$v * (1 + \frac{t}{n})^{n*a}$$

Faça alguns testes para o caso em que a composição é mensal ( $n = 12$ ) e compare com os resultados obtidos no caso da acumulação ser anual ( $n = 1$ ).

Capitalizar com vários períodos é melhor.

### Exercício 3.7 M DONE

Suponha um conjunto com  $n$  objectos de um tipo e  $m$  objectos de outro tipo. A proporção dos objectos de cada tipo dá-nos uma medida da desordem do conjunto. Essa desordem pode ser medida pela fórmula:

$$H(p_1, p_2) = - \sum_{i=1}^2 p_i \log_2(p_i)$$

em  $p_i$  dá a proporção de elementos do tipo  $i$ . Escreva um programa que conhecendo o numero de objectos de cada tipo calcula a desordem do conjunto.

### Exercício 3.8 M

Podemos calcular as raízes de um polinómio  $f(x) = a_nx^n + a_{n-1}x^{n-1} + \dots + a_1x + a_0$  por um método simples conhecido pelo **método da bissecção**. No pressuposto de que a função é contínua num intervalo  $[k, p]$  e que  $f(k) * f(p) < 0$  o método converge para a solução. Funciona calculando o ponto médio,  $x$ , entre  $k$  e  $p$ . Se nesse ponto o valor do polinómio for 0 estamos perante a solução. Caso contrário procuramos num novo intervalo cujos extremos garantam a propriedade do valor do polinómio nesses pontos tiver sinal contrário. Esse novo intervalo será ou  $[x, p]$  ou  $[k, x]$ , dependendo de qual dos dois verifica a condição. Implemente o respectivo programa e teste-o com o polinómio  $f(x) = x^3 - x - 1$ . Para o ajudar na escolha dos valores de  $k$  e de  $p$  apresentamos na figura 3.1 uma visão parcial do polinómio.

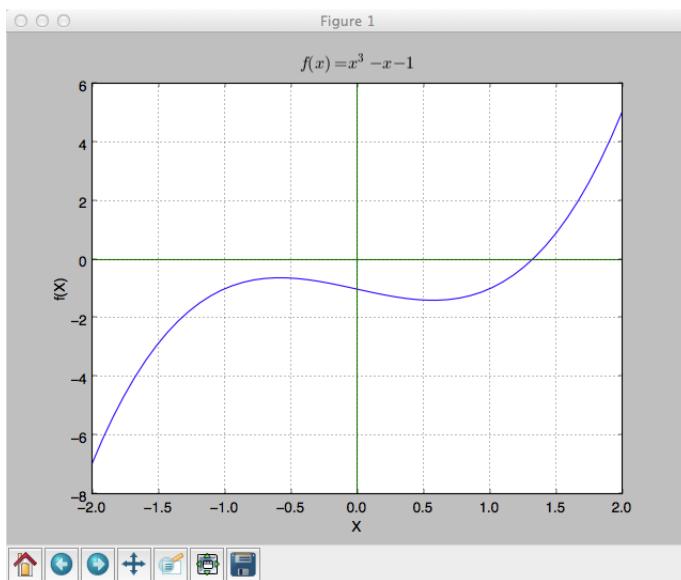


Figura 3.1: Um polinómio do terceiro grau

Note que existem pelo menos duas abordagens ao problema. Uma, que repete a divisão do intervalo um número fixo de vezes, a outra, que procura obter um resultado com um erro inferior a um certo valor.

### Exercício 3.9 F

Escreva um programa que permita descodificar um texto que foi codificado com o método da separação entre caracteres nas posições pares e caracteres nas posições ímpares.

### Exercício 3.10 F DONE

Escreva um programa que lhe permita descodificar um texto codificado pelo método da chave de substituição.

### Exercício 3.11 F DONE

Apresentámos um programa que nos permite calcular a cadeia complementar de uma dada cadeia de ADN. Apresente uma solução alternativa ao problema. **Sugestão:** Pense em usar a instrução condicional **if**.

### Exercício 3.12 F DONE

Desenvolva um programa que lhe permitir gerar uma cadeia de ADN. O tamanho deve ser um parâmetro do seu programa. **Sugestão:** Pense em usar o padrão ciclo - acumulador.

### Exercício 3.13 F DONE

Desenvolva um programa que substitua as ocorrências de vogais numa cadeia de caracteres por espaços em branco.

### Exercício 3.14 F DONE

Escreva um programa que dada uma cadeia de caracteres e um número inteiro positivo, imprima todas as suas **sub-cadeias** de comprimento igual ao número. Por exemplo, para a cadeia de caracteres 'Monty Python' e comprimento 3, o resultado será o apresentado na listagem.

```
Mon
ont
nty
ty
y P
Py
Pyt
yth
tho
hon
```

### Exercício 3.15 M DONE

Escreva um programa que dada uma cadeia de caracteres nos permite obter todos os **prefixos** da cadeia. O exemplo abaixo mostra a saída gerada no caso em que a cadeia é a frase 'Monty Python'.

```
M
Mo
Mon
Mont
Monty
Monty
Monty P
Monty Py
Monty Pyt
Monty Pyth
Monty Pytho
Monty Python
```

### Exercício 3.16 M DONE

Semelhante ao problema 3.15, só que agora pretende-se obter na saída todos os **sufixos**. Faça também uma versão genérica que funcione para qualquer cadeia de caracteres.

```
n
on
hon
thon
ython
Python
 Python
y Python
ty Python
nty Python
onty Python
Monty Python
```

### Exercício 3.17 Módulo turtle F DONE

À semelhança dos humanos a nossa amiga tartaruga **tarta** tem um **código genético** baseado num alfabeto de quatro letras:  $\{f', t', e', d'\}$ . Ainda como no caso dos humanos, aquilo que ela é (faz) resulta da *expressão* do seu

ADN. Para tal, cada letra está ligada a uma acção simples : 'f', move-se para a frente, 't', move-se para trás, 'd' roda à direita e, 'e' roda à esquerda. Por exemplo, se o seu ADN for 'feftd' **tarta** passa o tempo a executar as acções: para a frente ('f'), roda à esquerda ('e'), para a frente ('f'), para trás ('t') e roda à direita ('d').

Escreva um **simulador** que permita mostrar o percurso da nossa amiga, conhecido o seu ADN. Admita que os movimentos são de valor constante, o mesmo sucedendo com as rotações.

### Exercício 3.18 Módulo random Módulo turtle F DONE

Pretendemos um simulador semelhante ao problema 3.17, mas em que, agora, os movimentos e as rotações tenham valores aleatórios. Escolha no entanto os valores possíveis dentro de uma gama razoável!

### Exercício 3.19 Módulo random Módulo turtle F DONE

Nos problemas 3.17 e 3.18 o ADN da tartaruga é determinado à partida. Escreva um simulador em que o ADN é definido primeiro de modo aleatório e depois executado. Também neste caso suponha que os valores das deslocações e rotações podem variar aleatoriamente. O único argumento da função deve ser o **comprimento** do ADN.

### Exercício 3.20 M DONE

Um método para codificar/descodificar um texto baseia-se na ideia de substituir um carácter pelo carácter que está a uma certa **distância** dele. Por exemplo, se a distância escolhida for 2, então o **c** substitui o **a**, o **d** substitui o **b** e assim sucessivamente. Escreva um programa para codificar e outro para descodificar recorrendo a este método. A distância deve ser um parâmetro do problema e pode ser positiva ou negativa. Pense como vai resolver o caso dos caracteres nas extremidades do alfabeto.

## Instruções destrutivas

### Exercício 4.1 MF

Recorrendo ao interpretador, identifique quais dos seguintes nomes são válidos para nomes de objectos:

- abc
- 5peso
- \_valor
- Ernesto Costa
- ABC
- with
- peso\$
- minha\_altura
- class
- nome\_ALUNO
- a(b)
- \_\_\_\_1
- \_\_x\_\_
- import\_from

- area-rect

Justifique os casos em que os nomes não são válidos.

### Exercício 4.2 MF

Escreva um programa que lhe permita imprimir os caracteres gregos  $\alpha, \beta, \gamma$ . **Sugestão:** obtenha os códigos **unicode** de cada caracter.

### Exercício 4.3 MF

Experimente fazer o seguinte:

```
>>> x = 5
>>> y = 5
>>>
```

Inspeccione os objectos de nome  $x$  e  $y$ , isto é, determine a sua identidade, valor e tipo. Que conclusões pode tirar?

### Exercício 4.4 MF

Experimente fazer o seguinte:

```
>>> a = 10
>>> b = a
>>>
```

Inspeccione os objectos e tire conclusões.

### Exercício 4.5 MF

Simule a seguinte sessão no interpretador:

```
>>> x = 5
>>> x = x + 1
>>>
```

Inspeccione o objecto  $x$  após cada passo. Que conclusões pode tirar?

### Exercício 4.6 MF

Considere a seguinte sessão no interpretador:

```
>>> a = 10
>>> b = a
>>> a = 11
```

Como explica os resultados da inspecção?

### Exercício 4.7 M

Desenhar diagramas do ambiente depois de um conjunto de atribuições.

### Exercício 4.8 M

Explique o que acontece de modo claro, sintético e **rigoroso** quando executa o comando:

```
>>> cad = 'a' * 3
```

A sua explicação deve incluir a visualização do **espaço de nomes** e do **espaço de objectos** depois de executado o comando indicado.

### Exercício 4.9 F

Considere a seguinte definição:

```
def add2me(x):
    return x + x
```

Indique, **justificando**, quais os resultados esperados ao executar os comandos:

```
>>> add2me(23.4)
???
>>> add2me('toto')
???
```

### Exercício 4.10 M

Explique de modo claro, sintético e **rigoroso** o que aconteceu na sessão seguinte:

```
>>> def prod(x,y):
...     return x * y
...
>>> a = 5
>>> print(prod(a,3))
15
>>> a
5
>>> x
Traceback (most recent call last):
  File "<string>", line 1, in <fragment>
NameError: name 'x' is not defined
```

&gt;&gt;&gt;

**Exercício 4.11 F**

Usando a instrução `print` e o método `format` diga como podia obter o efeito da listagem seguinte:

```
Bem vindo a IPRP
Bem vindo a IPRP
    Bem vindo a IPRP e ao DEIUC
```

**Exercício 4.12 M DONE**

Desenvolva um programa que lhe permita imprimir a seguinte tabela.

Número	Quadrado
1	1
2	4
3	9
4	16
5	25

Caso pretenda que a tabela possa ter um número variável de linhas em que medida precisa, ou não, de alterar a sua solução? Se a resposta for afirmativa apresente o respectivo programa.

**Exercício 4.13 F DONE**

Escreva um programa que lhe permita apresentar uma tabela de conversão entre milhas e quilómetros. Uma milha é igual a 1,609 quilómetros. A tabela deve conter todas as equivalências entre dois valores de referência. A tabela deve ter um aspecto semelhante ao da listagem seguinte, que ilustra o caso entre os números 10 e 20.

Milhas	Quilómetros
10.00	16.09
11.00	17.70
12.00	19.31
13.00	20.92
14.00	22.53
15.00	24.13
16.00	25.74
17.00	27.35
18.00	28.96

19.00	30.57
20.00	32.18

**Exercício 4.14 M DONE**

Desenvolva um programa que dado um número inteiro menor ou igual a dez imprime a tabela da respectiva tabuada. A listagem abaixo ilustra para o caso do número 7.

Tabuada do número 7

```
-----
7 x 1 = 7
7 x 2 = 14
7 x 3 = 21
7 x 4 = 28
7 x 5 = 35
7 x 6 = 42
7 x 7 = 49
7 x 8 = 56
7 x 9 = 63
7 x 10 = 70
```

**Exercício 4.15 M DONE**

Desenvolva um programa que dado um nome por extenso constrói o respetivo acrónimo. A listagem abaixo ilustra o pretendido.

```
>>> print(acronimo('Random Access Memory'))
RAM
```

**Exercício 4.16 F DONE**

Na descolagem de um avião a relação entre a aceleração,  $a$ , a velocidade,  $v$ , determina o comprimento mínimo da pista,  $c$ , para tudo correr bem, de acordo com a fórmula:

$$c = \frac{v^2}{2 \times a}$$

Escreva um programa que pede os dados ao utilizador e imprime uma mensagem com o resultado. Uma hipótese de interacção é dada na listagem seguinte:

```
Velocidade de descolagem (m/s): 50
Aceleração para descolagem (m/s?): 4.5
```

Para a velocidade 50.00 e aceleração 4.50 o comprimento mínimo da pista é: 277.78.

### Exercício 4.17 F DONE

A energia necessária para elevar a temperatura de uma dada massa de água de uma temperatura inicial até uma temperatura final é dada pela fórmula:

$$E = m \times (t_i - t_f) \times 4184$$

$E$  é a energy (em Joules),  $t_i$  e  $t_f$  as temperaturas inicial e final (em graus Celsius) e  $m$  a massa (em quilogramas). Escreva um programa que pede os dados ao utilizador e imprime uma mensagem com o resultado. Uma hipótese de interacção é dada na listagem seguinte:

```
Temperatura inicial (Celsius): 10
Temperatura final (Celsius): 30
Quantidade de água (Quilogramas): 25
Para a massa de água 25.00, temperatura inicial 10.00 e
temperatura final 30.00 a energia necessária é: 2092000.00
Joules.
```

### Exercício 4.18 F DONE

A temperatura exterior depende de vários factores. Uma das fórmulas de cálculo faz intervir a velocidade do vento:

$$t_v = 35.4 + 0.6215 \times t - 35.75 \times v^{0.16} + 0.4275 \times t \times v^{0.16}$$

A temperatura é medida em graus Fahrenheit e a velocidade do vento em milhas por hora. Escreva um programa que pede os dados ao utilizador e imprime uma mensagem com o resultado. Uma hipótese de interacção é dada na listagem seguinte:

```
Velocidade do vento (milhas/hora): 5
Temperatura (Fahrenheit [-58, 41]): 10
Para a velocidade do vento 5.00 e temperatura exterior 10.00 a
temperatura é sentida como: 1.24.
```

### Exercício 4.19 M DONE

Desenvolva um programa que lhe permita imprimir os elementos de uma matriz antecedidos pela sua posição na matriz. A listagem abaixo exemplifica para a matriz `[[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]]`.

```
(0,0): 1  (0,1): 2  (0,2): 3  
(1,0): 4  (1,1): 5  (1,2): 6  
(2,0): 7  (2,1): 8  (2,2): 9  
(3,0): 10 (3,1): 11 (3,2): 12
```

### Exercício 4.20 M DONE

Pretende-se desenvolver um programa que pergunta ao utilizador a frequência de nascimentos, de mortes e de emigrantes em minutos. Depois, conhecida a população inicial deve calcular a nova população no final do ano. Assuma que o ano tem 365 dias. A listagem mostra uma interação possível.

Frequência de nascimentos (minutos): 20

Frequência de falecimentos (minutos): 15

Frequência de emigração (minutos): 10

Resumo dos dados:

-----  
Frequência de nascimentos: 20

Frequência de mortes: 15

Frequência de emigrantes: 10

População Inicial: 10000000

Estimativa:

-----  
A população ao fim de um ano: 9938680

Se quiser estimar o resultado ao fim de vários anos como modificaria a sua solução?



# Capítulo 5

## Instruções de controlo

### Exercício 5.1 F

Escreva um programa que apresenta por ordem crescente três números inteiros positivos dados como entrada. Em que medida a sua solução minimiza o número de comparações necessárias? **Nota:** Não pode usar nenhuma função/método de ordenamento pré-definido de Python .

### Exercício 5.2 F

Para realizar a viagem entre o Porto e Coimbra (120 km de distância) existem várias estradas como alternativa. A tabela 5.1 ilustra as diferentes alternativas em termos de trajecto considerando o custo de combustível por km e o custo das portagens. Escreva um programa que dada a designação da estrada retorne o custo total da viagem para essa alternativa.

Estradas	Custo combustível / Km	Custo Portagens
A1	0.15	6.52
A20	0.12	15.2
A21	0.10	5.75

Tabela 5.1: O que escolher? Preços em euros.

### Exercício 5.3 F

O vencimento bruto de um trabalhador está sujeito a descontos: 25% para o IRS, 5% para a Segurança social e 10% para a Caixa Nacional de Aposentações. O vencimento líquido é o que resulta da subtracção destes descontos

ao vencimento bruto. Desenvolva um programa que dado o vencimento bruto devolve o correspondente vencimento líquido.

### Exercício 5.4 F

A avaliação nesta cadeira resulta de 5 provas: 4 testes e um exame. Cada teste vale 7.5% da nota final, enquanto que o exame vale 70%. Isto significa que a nota é dada pela expressão:

$$\text{nota} = 0.075 * (t_1 + t_2 + t_3 + t_4) + 0.7 * e$$

Escreva um programa que dadas as 5 notas parciais calcula a nota final e, como resultado devolve a cadeia de caracteres "Aprovado", se a média for maior ou igual a 14, "Reprovado", se a média for inferior a 7, e "Oral", se a média for maior ou igual a 7 e inferior a 14. Admita que as notas são números reais entre 0 e 20.

### Exercício 5.5 F

Considere o seguinte pedaço de código Python .

```
1 i = 20
2 while (i >= 0):
3     print( "i= ",i)
4     i = i - 2
```

Listagem 5.1: Ciclo while

Escreva um pedaço de código equivalente em que o ciclo **while** é substituído por um ciclo **for**.

### Exercício 5.6 M

```
>>> for i in range(3):
...     for j in range(1,3):
...         print(i/j)
...
??
>>>
```

Diga, **justificando**, o que vai aparecer no lugar do **ponto de interrogação** quando o código é executado no interpretador.

### Exercício 5.7 M DONE

Duas palavras de igual comprimento dizem-se **amigas** se o número de posições em que os respectivos caracteres **difere** for inferior 10%. Escreva

um programa que dadas duas palavras indica se são ou não amigas.

### Exercício 5.8 M DONE

Escreva um programa que calcula o divisor mais pequeno de um número inteiro maior do que 1. Use esse programa como auxiliar na determinação de um dado inteiro maior do que um é um número primo.

### Exercício 5.9 M random DONE

Suponha um dado em que cada uma das faces está numerada com os inteiros de 1 a 6. Desenvolva um programa que simula o lançamento repetido do dado um certo número de vezes, e calcula a **percentagem** de vezes em que saiu um número par.

### Exercício 5.10 M DONE

Considere a figura 5.1. Suponha que o quadrado externo tem uma dimensão 2 por 2 e os internos 1 por 1.

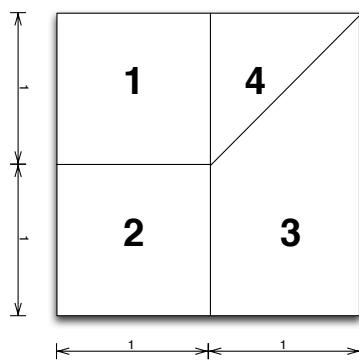


Figura 5.1: Calcular probabilidades

Use o Método de Monte Carlo para calcular a probabilidade de ao atirar um dardo ele cair numa região de número ímpar.

### Exercício 5.11 F DONE

O factorial de um número é dado por:

$$\text{fact}(n) = n! = n \times (n - 1) \times (n - 2) \times \dots \times 1$$

Implemente um programa que lhe permita calcular o factorial de um inteiro positivo.

### Exercício 5.12 M

O valor do seno de um ângulo que pode ser computado a partir da fórmula:

$$\text{seno}(x) = \sum_{i=0}^{\infty} \frac{(-1)^i \times x^{(2 \times i + 1)}}{(2 \times i + 1)!}$$

Implemente o respectivo programa usando o número de parcelas como parâmetro. Altere o programa por forma a poder usar a precisão como critério de paragem do seu programa.

### Exercício 5.13 Módulo matplotlib F

O número harmónico  $H_n$  define-se pela fórmula:

$$H_n = \sum_{k=1}^n \frac{1}{k}$$

Escreva um programa que permita calcular o valor de  $H_n$ , usando  $n$  como parâmetro. Use esse programa para poder encontrar os sucessivos valores dos números harmónicos até um dado limite. Recorra ao módulo **matplotlib** para visualizar o resultado. Que comentários se lhe oferecem fazer face ao gráfico?

### Exercício 5.14 F

Os números harmónicos podem ser calculados de modo aproximado pela fórmula:

$$H_n \approx \ln(n) + \gamma$$

com  $\ln(n)$  o logaritmo natural (base e) e  $\gamma = 0.5772156649$  a constante de Euler. Escreva um programa que lhe permita calcular o valor de  $H_n$  de modo aproximado para sucessivos valores de  $n$ . Visualize o resultado e compare com o obtido no exercício 5.13. Que pode dizer sobre a qualidade da aproximação?

### Exercício 5.15 F DONE

O logarithm natural é definido pela fórmula:

$$e = \sum_{i=0}^{\infty} \frac{1}{i!}$$

Escreva um programa que lhe permita calcular o valor aproximado de  $e$  com uma dada precisão.

### Exercício 5.16 M DONE

Um número diz-se perfeito se for igual à soma dos seus divisores, excluindo ele próprio. Por exemplo, 6 e 28 são perfeitos. Escreva um programa que determine quais os números perfeitos que existem num dado intervalo.

**Exercício 5.17** M Considere os padrões de números seguintes. **DONE**

# Padrão A

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

#Padrão B

```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

# Padrão C

```
      1
     2 1
    3 2 1
   4 3 2 1
  5 4 3 2 1
```

Escreva três programas, que lhe permitam imprimir cada um dos padrões. Em que medida a sua solução depende da dimensão do número máximo  $n$ ?

**Exercício 5.18** Módulo turtle M **DONE**

Suponha que quer desenhar uma grelha como a da figura 5.2, em que a dimensão da grelha e o tamanho de cada célula são parâmetros do problema. Use o módulo **turtle** para o fazer.

**Exercício 5.19** Módulo turtle M **DONE**

Um passeio aleatório é um conceito que permite modelizar vários processos que ocorrem na natureza. Admita que tem um agente que se movimenta de modo aleatório num mundo 2D. Suponha que a cada momento o agente decide deslocar-se ou para norte, ou para este, ou para sul ou para oeste, sendo que essa decisão é aleatória. Usando o módulo **turtle** simule um passeio aleatório do nosso agente, admitindo que o seu mundo 2D tem a forma



Figura 5.2: Desenho de uma grelha

de uma grelha como a que obteve no exercício 5.18. O mundo é suposto ser **finito**. A figura 5.3 ilustra o que se pretende ( o ponto marca o início do passeio e a seta o final).

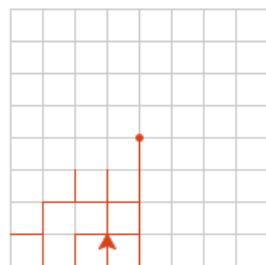


Figura 5.3: Passeio aleatório

### Exercício 5.20 D DONE

A sequência de Fibonacci define-se indutivamente do modo seguinte: os seus dois primeiros números são iguais a um e a partir daí cada elemento da sequência é igual à soma dos dois elementos imediatamente anteriores. Eis os primeiros números da sequência:

**1, 1, 2, 3, 5, 8, 13, 21, ...**

Escreva um programa que dado um número **verifica** se ele pertence ou não à sequência de Fibonacci.

### Exercício 5.21 F DONE

Sabemos que existem dois métodos sobre cadeias de caracteres que nos indicam a posição, o índice, do início de uma ocorrência de uma sub-cadeia

numa outra cadeia. Mas são diferentes: `find` indica o índice caso ocorra ou -1 no caso de não ocorrer; `index` indica o índice caso ocorra ou levanta uma exceção caso não se encontre presente. Escreva um programa que recorra explicitamente às exceções e ao método `index` por forma a que o comportamento final seja igual ao do método `find`.



## Objectos (II)

### Exercício 6.1 F

Dada uma lista com as idades dos alunos de uma turma, desenvolva um programa para cada um dos seguintes problemas, usando apenas as operações acima referidas.

1. Mostre o número de idades;
2. Exiba todos as idades da lista;
3. Exiba todas as idades na ordem inversa à da lista fornecida;
4. Exiba todas as idades excepto a primeira e a última da lista;
5. Mostre a idade menor e a maior;
6. Calcule e mostre a soma dos valores na lista;
7. Calcule e mostre o número de elementos de valor abaixo de um outro, dado como referência.
8. Verifique se existe algum aluno com 17 anos;

### Exercício 6.2 F DONE

Desenvolva um programa que dada uma lista de números devolva a **soma** dos seus números pares e a **soma** dos seus números ímpares. A listagem 6.2 ilustra o que se pretende.

```

1 >>> lst = [1,4,7,9,3,2,8,5,6]
2 >>> # ---> Aqui o seu programa de nome pares_impares.
3 >>> pares_impares(lst)
4 (20, 25)
5 >>>

```

Listagem 6.1: Pares e Ímpares

**Exercício 6.3 F DONE**

Desenvolva um programa que dadas duas listas devolve uma terceira formada pelos elementos das primeiras dispostos de modo alternado. Começa com a primeira lista.. A listagem 6.2 ilustra o que se pretende.

```

1 >>> l1 = [1,2,3]
2 >>> l2 = ['a','b','c']
3 >>> # ---> Aqui o seu programa de nome alterna.
4 >>> alterna(l1, l2)
5 [1,'a',2,'b',3,'c']
6 >>>

```

Listagem 6.2: alterna

**Exercício 6.4 F DONE**

Desenvolva um programa que, dados um elemento numérico e uma lista de números, determina **quantos** elementos da lista são **menores** do que o número. A listagem 6.3 ilustra o que se pretende.

```

1 >>> # ---> Aqui o seu programa de nome conta_menores.
2 >>> conta_menores(5,[2,8,6,5,3,2])
3 3
4 >>>

```

Listagem 6.3: contar menores

**Exercício 6.5 F Módulo random DONE**

Suponha que tem dois dados numerados de 1 a 6. Vai lançá-los sucessivas vezes e guardar os resultados (a soma). Escreva um programa que mostre os resultados dos sucessivos lançamentos e determine a percentagem de vezes em que saiu uma soma par.

**Exercício 6.6 M DONE**

Desenvolva um programa que receba uma lista de números e calcule a soma cumulativa, i.e., o programa deve retornar uma nova lista em que o

elemento de ordem **i** é a soma dos primeiros **i+1** elementos da lista original.  
Exemplo: para [1,2,3] deve devolver [1,3,6].

### Exercício 6.7 M DONE

Uma imagem a preto e branco pode ser guardada como uma lista de listas. Cada elemento representa uma linha da imagem. O preto é representado por 1 e o branco por zero. Por exemplo, `[[0,1,0],[1,1,1],[0,1,0]]` representa uma cruz. Escreva um programa que, dada uma imagem produz o seu **negativo**, isto é uma nova imagem em que o branco passa a preto e o preto a branco.

### Exercício 6.8 MD DONE

Uma imagem a preto e branco pode ser guardada como uma lista de listas. Cada elemento representa uma linha da imagem. O preto é representado por 1 e o branco por zero. Por exemplo, `[[0,1,0],[1,1,1],[0,1,0]]` representa uma cruz. Escreva um programa que, dada uma imagem a roda 90° no sentido dos ponteiros do relógio.

### Exercício 6.9 M DONE

Suponha que está perdido no meio de uma cidade e não tem GPS para se orientar. Pergunta a um transeunte como pode chegar ao seu destino. Como a cidade é geométrica a resposta é fácil. Recebemos uma sequência de indicações do tipo **vira à esquerda (E)**, **depois avança (A)**, **depois roda à direita (D)**, **depois avança (A)**, **depois avança de novo (A)**, **depois recua (R)**, .... Usando o módulo **turtle** desenvolva um programa, que quando executado, **simule** com a tartaruga os seus movimentos quando esta executa os comandos recebidos. A imagem 6.1 mostra o que acontece quando manda correr o programa geral **main\_tarta()**. Por conveniência de visualização marcámos o inicio e o fim do percurso com pontos, verde e vermelho, respectivamente.

O seu programa chama-se, no código abaixo, **navega**.

```
def main_tarta(n):
    tartaruga = turtle.Turtle()
    comandos = gera_comandos(n)
    navega(comandos, tartaruga)
    turtle.exitonclick()
```

Vai ter que implementar, para além de navega, a função **gera\_comandos(n)**

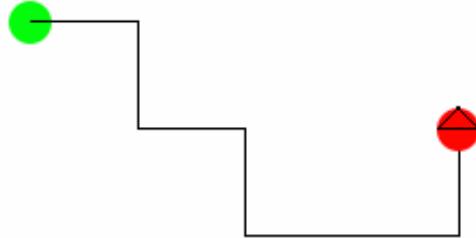


Figura 6.1: Passeando na cidade

que gera uma sequência válida de comandos de tamanho **n**.

### Exercício 6.10 F

Crie o seguinte dicionário de linguagens de programação e respectivos autores:

```
autor = {"php":"Rasmus Lerdorf", "perl":"Larry Wall", "tcl":"John Ousterhout",
"awk":"Brian Kernighan", "java": "James Gosling", "parrot": "Simon Cozens",
"python": "Guido van Rossum", "xpto": "zxcv"}.
```

- a) Acrescente um elemento ao dicionário **DONE**
- b) Altere o autor do python para "Guido van Rossum". **DONE**
- c) Remova o elemento com chave "xpto". **DONE**
- d) Quantos elementos tem o dicionário? **DONE**
- e) Existe uma entrada para "c++"? **DONE**

### Exercício 6.11 F **DONE**

Suponha que tem uma pequena loja de vender fruta, e que construiu uma pequena base de dados para fazer a gestão do stock da fruta. Para cada tipo de fruta tem a indicação da quantidade, em quilos, que tem para venda. Admita que inicialmente tem esses elementos em duas listas. Uma com o nome das frutas e outra com as quantidades. A partir desses dados crie o respectivo dicionário.

### Exercício 6.12 M **DONE**

Suponha que quer tornar a sua gestão da loja de fruta mais eficiente. Para isso, para cada tipo de fruta associa a informação da quantidade de fruta que **você** comprou, do preço de compra por quilo, da quantidade que tem em stock e do preço de venda por quilo. Como guardaria esta informação? Escreva programas para cada uma destas questões:

- a) Qual o lucro já obtido?
- b) Qual a fruta mais cara?

### **Exercício 6.13** M DONE

Vamos querer implementar um conversor de datas. Para tal vamos supor que temos guardado num dicionário a relação entre números e dias da semana, `dias_semana={1:'Domingo', 2:'Segunda-Feira', 3:'Terça-feira', ..., 7:'Sábado'}`, e outro para os meses do ano `meses_ano = {1: 'Janeiro', 2:'Fevereiro', ..., 12: 'Dezembro'}`. O formato DS/DM/M/A é um dos que é possível utilizar para representar uma data. Neste formato *DS* corresponde ao valor inteiro do dia da semana (0 a 7), *DM* corresponde valor inteiro do mês e *A* corresponde ao ano. Faça uma função que receba os dois dicionários criados anteriormente e uma cadeia de caracteres com a data no formato DS/DM/M/A, e apresente essa data por extenso.

Exemplo:

Para a data: "4/5/2006"

Quarta-feira, 5 de Junho de 2006

### **Exercício 6.14** M DONE

Escreva uma função que recebe um dicionário, em que cada elemento é formado pela chave, o número do BI de uma pessoa, e o valor contém informação sobre o sexo, idade, altura e peso, e devolve um novo dicionário com os rácios de metabolismo basal dessas pessoas. Tenha em conta que o rácio de metabolismo basal é dado por:  $66 + (6.3 * \text{peso}) + (12.9 * \text{altura}) - (6.8 * \text{idade})$  no caso de ser homem, e  $65.5 + (4.3 * \text{peso}) + (4.7 * \text{altura}) - (4.7 * \text{idade})$  no caso de ser mulher.

### **Exercício 6.15** M DONE

Escreva uma função que receba um dicionário, em que cada elemento associa o número do Bilhete de Identidade de uma pessoa (chave), com informação sobre a sua altura e peso, e devolva o mesmo dicionário onde foi acrescentado o índice de massa corporal de cada pessoa. O índice de massa

corporal de uma pessoa é calculado dividindo o seu peso pelo quadrado da sua altura.

### Exercício 6.16 M DONE

Desenvolva um programa que dado um texto, isto é uma cadeia de caracteres, construa um **dicionário**, com as **posições** em que ocorrem as vogais. Note que os caracteres podem ser maiúsculos ou minúsculos. A listagem 6.4 ilustra o pretendido.

```
>>> print(posicoes('agora e que vao ser elas, Ai, Ai!'))
{'a': [0, 4, 13, 22], 'A': [26, 30], 'e': [6, 10, 17, 20], 'i':
 : [27, 31], 'o': [2, 14], 'u': [9]}
>>>
```

Listagem 6.4: Índices das ocorrências

### Exercício 6.17 D DONE

Faça um programa que inverta um dicionário, i.e., que coloque os valores como chaves e as chaves como valores. Deverá ter em atenção que chaves diferentes podem ter o mesmo valor.

Exemplo:

Input:{'joao':10,'pedro':18, 'tiago':13,'luis':18}  
Output:{18: ['luis', 'pedro'], 10: ['joao'], 13: ['tiago']}.

### Exercício 6.18 F DONE

Dada uma árvore genealógica, organizada como um dicionário, escreva um programa que determine se duas pessoas são **irmãos/irmãs**.

### Exercício 6.19 M DONE

Dada uma árvore genealógica, organizada como um dicionário, escreva um programa que determine os **netos** de uma pessoa, caso existam.

### Exercício 6.20 M DONE

Dada uma árvore genealógica, organizada como um dicionário, escreva um programa que determine o **avô/avó** de uma pessoa, caso exista.

### Exercício 6.21 F DONE

Sabemos que podemos testar de modo imediato se dois conjuntos são iguais. Imagine que esse teste não podia ser feito com conjuntos. Implemente um programa que lhe resolva a questão.

### Exercício 6.22 M

Suponha que tem um conjunto e quer criar um novo, formado por todos os elementos que satisfazem um dado predicado. Diga como podia implementar o respectivo programa.

**Exercício 6.23 M DONE**

O produto cartesiano de dois conjuntos, cujo programa apresentámos no texto, define uma relação binária sobre os dois conjuntos. Admita que os dois conjuntos são iguais. As relações têm propriedades. Por exemplo, uma relação sobre um conjunto diz-se reflexiva se para todo o elemento  $x$  do conjunto o par  $(x, x)$  fizer parte da relação. Escreva um programa que verifique se uma relação binária é, ou não, reflexiva.

**Exercício 6.24 M DONE**

O produto cartesiano de dois conjuntos, cujo programa apresentámos no texto, define uma relação binária sobre os dois conjuntos. Admita que os dois conjuntos são iguais. As relações têm propriedades. Por exemplo, uma relação sobre um conjunto diz-se simétrica se para todo o par  $(x, y)$  da relação, o par  $(y, x)$  também faz parte da relação. Escreva um programa que verifique se uma relação binária é, ou não, simétrica.



# Capítulo 7

## Ficheiros

### Exercício 7.1 F DONE

Desenvolva um programa que crie um ficheiro **primeiro.txt** com o seguinte conteúdo “Acabei de criar o meu primeiro ficheiro em Python.”. Depois de criado, use um editor para ver o que foi guardado. Pode escolher se, e como, divide o texto em linhas.

### Exercício 7.2 F DONE

Desenvolva um programa que mostre uma sequência de caracteres, em número pré-definido, presentes no ficheiro **primeiro.txt** a partir de uma dada posição de referência.

### Exercício 7.3 F DONE

Desenvolva um programa que adicione uma nova linha ao ficheiro **primeiro.txt** contendo a data de hoje, mas sem criar um novo ficheiro.

### Exercício 7.4 M DONE

Desenvolva um programa que permita identificar se um ficheiro contém números. O resultado do programa deve ser uma lista dos números existentes no ficheiro. Pode testar com um ficheiro de texto criado por si com um editor.

### Exercício 7.5 Módulo matplotlib M

Desenvolva um programa que analise as temperaturas médias de várias cidades portuguesas, guardadas no ficheiro **temperaturas.txt**<sup>1</sup>, determine os valores máximo e mínimo, e mostre o gráfico do resultado. Cada linha

<sup>1</sup>Disponível em <http://www.fca.pt>.

representa os dados referentes a uma cidade.

### Exercício 7.6 M DONE

Escreva um programa que crie uma cópia de um ficheiro. O nome dos ficheiros origem e destino devem ser pedidos ao utilizador e, de seguida, deve ser chamada uma função que faça a cópia. Os nomes dos ficheiros deverão ser argumentos da função. Pode testar com um ficheiro de texto criado por si com um editor.

### Exercício 7.7 Módulo random Módulo turtle M DONE

Desenvolva um programa que coloca, num ficheiro a criar, pares de números. Esses pares devem estar um por linha e ser gerados aleatoriamente. Admita que esses valores correspondem aos números (entre 1 e 6) de dois dados. Leia depois o ficheiro, interprete cada par como coordenadas num espaço 2D, e use o módulo **turtle** para desenhar a figura que resulta de unir esses pontos respeitando a ordem em que aparecem no ficheiro.

### Exercício 7.8 Módulo matplotlib Módulo turtle M

Usando o ficheiro de dados do exercício 7.7, desenvolva um programa que permita analisar a frequência dos valores que existem no ficheiro. A informação deve ser representada visualmente, e deve mostrar o número de vezes que cada um dos valores saiu. Utilize o módulo **matplotlib** ou o módulo **turtle** para fazer um diagrama de barras que represente o número de vezes que um valor saiu.

### Exercício 7.9 Módulo matplotlib Módulo turtle M

Usando o ficheiro obtido para os problemas 7.7 e 7.8, desenvolva um programa que permita analisar a frequência da **soma** de dois valores. A informação deve ser representada visualmente, e deve ser mostrada em termos de percentagens. Utilize o módulo **matplotlib** ou o módulo **turtle**, para fazer um diagrama de barras das **percentagens** referentes a soma dos dois valores.

### Exercício 7.10 Módulo matplotlib Módulo turtle M

Desenvolva um programa que permita analisar a frequência dos caracteres que existem num ficheiro. A informação deve ser mostrada visualmente. Pode socorrer-se do módulo **matplotlib** ou do módulo **turtle**. Pode testar com um ficheiro de texto criado por si com um editor.

### Exercício 7.11 M DONE

Escreva um programa que permita gerir vendas a dinheiro. Num ficheiro

de texto tem informação sobre vendas já efectuadas. Cada linha do ficheiro tem informação sobre uma transação, na forma: número da transação, nome da empresa, número de contribuinte, data e valor. O seu programa deve obter os elementos de uma nova transacção e actualizar o ficheiro. Deve também imprimir um documento onde constem os elementos referidos e ainda o nome do funcionário que fez a venda. A listagem ilustra o pretendido para a impressão.

#### Venda a Dinheiro No 100

---

**Empresa:** Vendas&Vendas  
**N.C.:** 987654321  
**Data:** 6/Out/2008  
**Valor:** 100.20 Euros  
**Vendedor:** Manuel Antunes

#### Exercício 7.12 M DONE

Admita que tem um ficheiro onde estão guardados números **inteiros**. Não sabe quantos existem por linha, nem quantos existem no total. Agora o que sabe é que podem existir números **repetidos** espalhados pelo ficheiro. Pretende-se um programa que leia o conteúdo do ficheiro e crie um **dicionário** em que as chaves são os números e os valores o número de vezes em que cada um aparece no ficheiro.

#### Exercício 7.13 M DONE

Suponha que tem um ficheiro com informação sobre dados pessoais. Mais concretamente em cada linha do ficheiro tem o nome, apelido, idade, código da profissão e código do estado civil. A tabela 7.1 ilustra uma situação possível.

Tabela 7.1: Ficheiro de Dados: entrada

Ernesto Costa	60	102	1
Ana Paz	44	411	2
Carlos Ferreira	20	203	2

Escreva um programa que leia este ficheiro e produza um novo no qual o nome e apelido foram substituídos pelas respectivas iniciais, a idade se manteve, e os códigos de profissão e estado civil alterados para os respectivos nomes. A tabela 7.2 mostra a saída resultante de aplicar o programa ao

ficheiro de entrada da tabela 7.1. A relação entre os códigos e os nomes correspondentes (por exemplo, 102 corresponde a professor, 1 corresponde a casado) estão guardadas em dois **dicionários**. É evidente que pode definir a correspondência do modo que entender.

Tabela 7.2: Ficheiro transformado

EC	60	Professor	Casado
AP	44	Advogado	Solteiro
CF	20	Estudante	Solteiro

### Exercício 7.14 D

Pretendemos saber se existe alguma **correlação** entre o crescimento do produto interno bruto ( **PIB**) e o **desemprego**, na zona Euro. Para isso temos dois ficheiros com essa informação. A informação cobre anos distintos nos dois ficheiros, e a informação neles presente tem periodicidade diversa.

```
# Desemprego na Zona euro
# Fonte: http://www.economagic.com
1993 01 1512·202
1993 02 4112·391
1993 03 8812·655
1993 04 4512·842
1993 05 2913·036
...

```

Listagem 7.1: 'Desemprego'

```
# Crescimento do Produto Interno Bruto : Zona EURO
# fonte http://www.economagic.com
1996 01 502·7
1996 02 262·0
1996 03 11·6
1996 04 471·4
1997 01 591·4
...

```

Listagem 7.2: 'Crescimento do PIB'

Desenvolva um programa que **dado um período de tempo**, analisa a informação, calcula o coeficiente de correlação e visualiza os dados.

### **Exercício 7.15 D**

Suponha que é dono de uma empresa e tem uma pequena base de dados com informações sobre os seus clientes. Essa informação, guarda num ficheiro, inclui, entre outras coisas, o nome, a data de nascimento, morada e número de telefone. Imagine que quer enviar aos clientes nascidos antes de um dado ano uma carta. A diferença nas cartas é apenas no cabeçalho, que deve personalizar pelo nome e pela morada. Admita que antes de enviar as cartas as guarda todas em ficheiros separados. Desenvolva a respectiva aplicação, isto é, um programa que leia o modelo da carta de um ficheiro, determine quais os clientes a quem deve enviar a carta, produza acarta para cada cliente e a guarde num ficheiro individual.

### **Exercício 7.16 D**

Admita que tem uma pequena base de dados com informação sobre a sua biblioteca de músicas. Cada música deve ter como informação: intérprete, título, tipo de música, duração e se está emprestado ou não. Desenvolva uma aplicação que lhe permita:

- introduzir um novo título
- marcar uma música como emprestada
- mostrar todas as músicas de um certo tipo

### **Exercício 7.17 D**

Admita que tem um ficheiro em que **cada linha** contém um endereço de Internet com o **URL** para a página pessoal de um utilizador, no formato habitual que o exemplo abaixo ilustra.

<http://eden.dei.uc.pt/~ernesto>

Notar que a parte final é sempre o nome do utilizador precedido pelo *til*. Suponha também que definiu um dicionário onde a cada nome de utilizador faz corresponder uma lista com o nome completo do utilizador e o seu endereço de correio electrónico (ver exemplo).

```
{'ernesto': ['Ernesto Costa', 'ernesto@dei.uc.pt'], ...}
```

Escreva um programa que, dados um dicionário e o endereço de um ficheiro, cada um com a informação acima descrita, faça a leitura deste último

e, a partir dos dados obtidos, extraia o nome dos utilizadores, usando-o em conjunção com o dicionário para devolver uma lista com os **nomes completos** dos utilizadores presentes no ficheiro, **ordenada** por ordem alfabética.

### Exercício 7.18 Módulo csv D

O ficheiro *zoo.csv*<sup>2</sup> tem informação diversa sobre animais. Em cada linha encontra a descrição de um animal específico, o nome na primeira posição e a sua classificação na última. Leia o ficheiro e construa uma estrutura do tipo dicionário em que as chaves são a classe e o valor a lista com os nomes dos animais da classe.

### Exercício 7.19 Módulo urllib D

Os ficheiros HTML têm no seu início um conjunto importante de informações. Eis um exemplo retirado do sitio <http://www.python.org>.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang=
  "en">

<head>
  <meta http-equiv="content-type" content="text/html; charset=
    utf-8" />
```

Uma delas é o tipo de codificação usada no texto, identificado através de **charset**. Escreva um programa que lhe permita ir buscar a um ficheiro HTML essa informação.

---

<sup>2</sup>Pode ser obtido no sitio da cadeira em <http://iprnet>.

# Capítulo 8

## Visões (II)

### Exercício 8.1 F

Desenvolva um programa que lhe permita mostrar os elementos que formam a matriz triangular inferior de uma da matriz à semelhança do que foi feito para o caso da matriz triangular superior. A listagem abaixo mostra o pretendido para a matriz

```
mat = [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
```

```
1  
5 6  
9 10 11
```

### Exercício 8.2 M

Desenvolva um programa que lhe permita extrair uma sub-matriz de uma matriz dada, conhecido o elemento inicial e as dimensão da sub-matriz. Por exemplo, se chamarmos o programa com:

```
print(sub_matriz([[1,2,3,4],[5,6,7,8],[9,10,11,12]],1,1,2,2))
```

o resultado obtido deverá ser:

```
[[6, 7], [10, 11]]
```

### Exercício 8.3 F

Desenvolva um programa que lhe permita gerar uma matriz de tuplos em que cada tuplo contém possíveis valores de pixeis. Os parâmetros do programa são a largura e a altura da imagem.

### Exercício 8.4 M

Desenvolva um programa que lhe permita desenhar uma linha recta numa janela, conhecidas as coordenadas de dois dos seus pontos. Relembramos que a equação de uma recta é dada por:

$$y = \frac{y_2 - y_1}{x_2 - x_1} \times (x - x_1) + y_1$$

### Exercício 8.5 M

Desenvolva um programa que lhe permita desenhar um arco de circunferência com uma dada amplitude, conhecidos também as coordenadas do centro e o raio. Deve ser genérico: no limite deve permitir desenhar uma circunferência!

### Exercício 8.6 F

Pretende-se implementar um programa que permita adicionar uma moldura a uma imagem. Deve ser possível definir o tamanho da moldura e a sua cor. A figura 8.1 ilustra o pretendido.

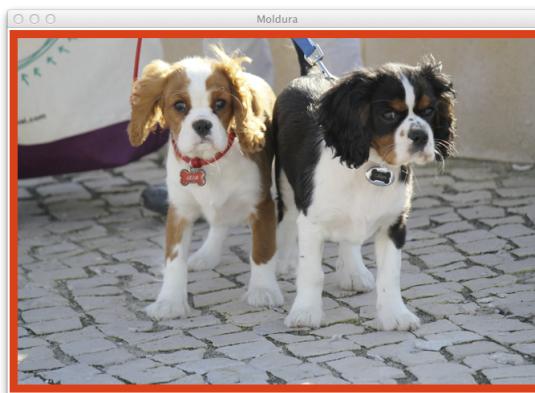


Figura 8.1: Acrescentar uma moldura

### Exercício 8.7 F

Implemente um programa que permita obter um corte de uma imagem. Para além da imagem devem ser fornecidos ao programa o ponto de início do corte e as dimensões. A figura 8.2 ilustra a ideia pretendida.

### Exercício 8.8 M

Discutimos um modelo para transformar toda uma imagem alterando um a um os seus pixeis de acordo com a mesma função de transformação. Vamos



Figura 8.2: Corte de imagem

aplicar esse modelo ao caso da transformação de uma imagem a cores numa a **preto e branco**. A figura 8.3 ilustra o pretendido. **Sugestão:** Para obter a imagem a preto e branco converta cada pixel para cinzento e depois compare com um limiar (por exemplo 128). Os maiores que o limiar são transformados em branco (ou seja  $(255,255,255)$ ) e os menores ou igual em preto (ou seja em  $(0,0,0)$ ).



Figura 8.3: A preto e branco

### Exercício 8.9 M

Desenvolva um programa que lhe permita variar a intensidade dos pixels de modo **independente**, isto é, a variação no vermelho, no verde e no azul pode ser diferente.

### Exercício 8.10 D

Vimos um exemplo de como ampliar uma imagem, eventualmente com

distorção. O problema agora é o inverso. Implementar um programa que permita reduzir uma imagem, uma vez mais com eventual distorção. A figura 8.4 ilustra o que se pretende.

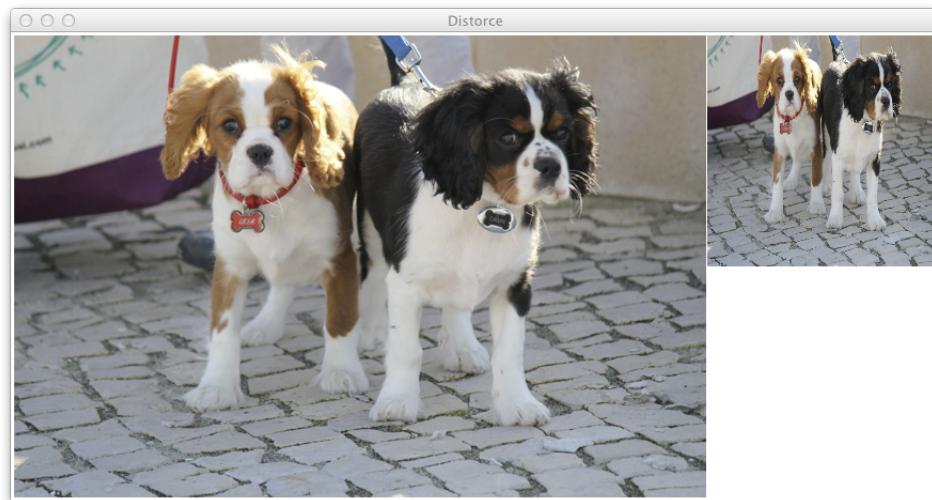


Figura 8.4: Redução de uma imagem

### Exercício 8.11 M

Vimos no texto como pegar na metade esquerda de uma imagem, criar uma cópia e juntar as duas partes como se fossem a imagem no espelho uma da outra. Pretende-se agora fazer algo semelhante, mas em que usamos a metade superior da imagem e o espelho é feito segunda uma perspectiva horizontal. A figura 8.5 mostra o efeito pretendido.

### Exercício 8.12 M

Muitas vezes acontece termos que reduzir as cores de uma imagem a um número limitado de cores. O processo de redução passa por determinar para cada pixel qual a cor mais próxima de entre as cores disponíveis. Implemente o programa que lhe permite fazer essa transformação. Por exemplo, para a paleta de cores:

```
palete_cores = [(255,0,0), (0,255,0), (0,0,255), (0,0,0),
(255,255,255), (255,255,0), (0,255,255), (255,0,255)]
```

A execução do nosso programa para a imagem com os dois cães, que temos vindo a usar ao longo do capítulo, resulta no que a figura 8.6 ilustra.



Figura 8.5: Espelho horizontal

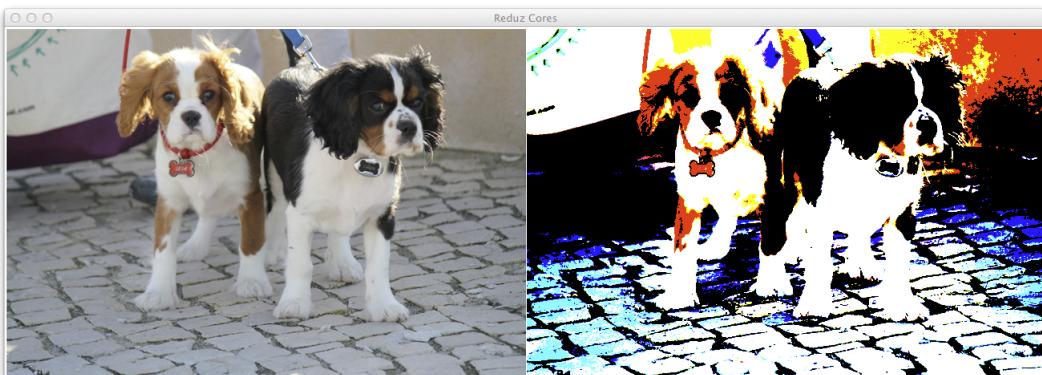


Figura 8.6: Redução de cores

### Exercício 8.13 M

Pretendemos desenvolver um programa que permita fazer uma colagem a partir de várias imagens. As imagens podem ter tamanhos diferentes, terem sido manipuladas para criar diversos efeitos, e ser colocadas em posições específicas da janela de visualização.

### Exercício 8.14 M

Estudámos o problema de minimizar o efeito da pixelização. A nossa solução evita considerar o bordo da imagem o que se traduz por uma fina moldura de cor preta. Reveja o programa e altere-o por forma que tal não aconteça. A figura 8.7 mostra o resultado pretendido. **Sugestão:** no bordo

calcule a média dos vizinhos existentes.

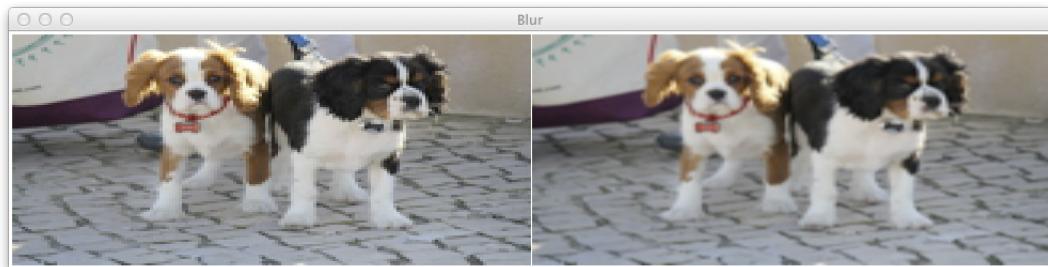


Figura 8.7: Suavizar sem moldura

### Exercício 8.15 M

Nas aulas falámos no conceito de Kernel (ou filtro, ou máscara) e mostrámos como o conceito podia ser usado para detectar os contornos de uma figura. Vamos agora explorar vários tipos de filtros. Faça uma pesquisa no **Google** pelas palavras **convolution** e **kernel** e descubra filtros diferentes dos apresentados neste capítulo. Escolha as imagens do seu agrado e aplique sobre elas o respectivo filtro. Visualize o resultado.

### Exercício 8.16 M

Um modo de eliminar o ruído de uma fotografia consiste em considerar para cada pixel e cada canal a mediana dos valores do pixel e dos seus vizinhos. O cálculo da mediana faz-se ordenando os valores e considerando o que está na posição do meio. Implemente o respectivo programa.

### Exercício 8.17 M

Pretende-se um programa que permita pegar em duas imagens e fabricar uma terceira. Desenvolva o programa de forma modular de modo a controlar o processo que usa para combinar os pixéis das imagens. A título de exemplo, na imagem 8.8 apresentamos uma combinação entre Einstein e Gandhi.

### Exercício 8.18 D

Desenvolva um programa que dada uma imagem a roda de 90 graus. O sentido é à sua escolha.

### Exercício 8.19 MD

Neste exemplo pretendemos resolver uma questão semelhante à do exemplo 8.18. Só que, agora, pretende-se que a rotação possa ser de um ângulo

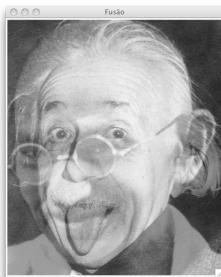


Figura 8.8: Gandstein

arbitrário. **Nota:** A resolução desta questão obriga a saber algo sobre translação de sistemas de eixos, de trigonometria e de álgebra.

### Exercício 8.20 MD

Muitas vezes queremos enviar imagens encriptadas. Para isso vamos criar um método que consiste em alterar a ordem das linhas da imagem original de modo aleatório. Quem receber a mensagem e conhecer o modo como se misturaram as linhas deve ser capaz de reconstruir a imagem original. Implemente os respectivos programas. A figura 8.9 ilustra o pretendido.

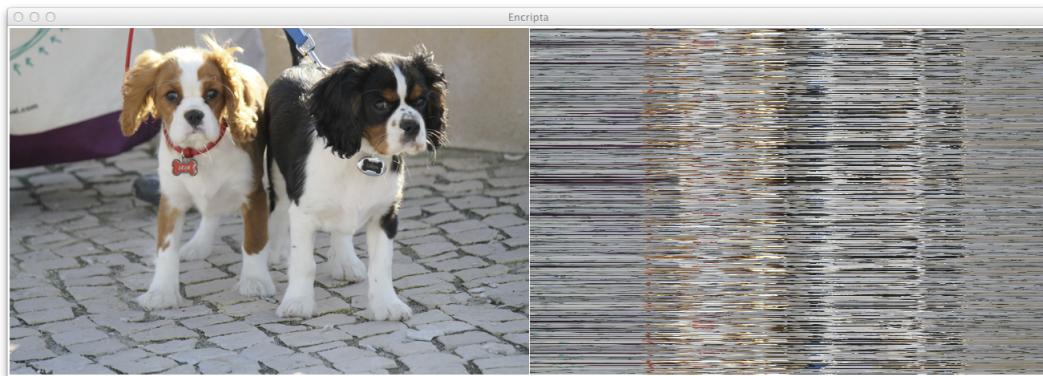


Figura 8.9: Encriptar uma imagem



# Capítulo 9

## Recursividade

### Exercícios

#### Exercício 9.1 F

Todos sabemos que as linguagens de programação têm um operador que permite calcular o resto da divisão inteira de dois números. Admitindo que os números são positivos ou nulos implemente um versão recursiva para o problema.

#### Exercício 9.2 F DONE

O produto escalar de dois vectores é dado por:

$$V \times W = \sum_{i=1}^n V_i \times W_i$$

Apresente uma solução recursiva que resolve o problema de calcular o produto escalar de dois vectores.

#### Exercício 9.3 F DONE

Existe um modo alternativo de calcular uma exponencial que se baseia na identidade:

$$x^n = \begin{cases} x^{n/2} \times x^{n/2} & \text{se } x \text{ for par} \\ x^{n/2} \times x^{n/2} \times x & \text{se } x \text{ for ímpar} \end{cases}$$

Com base nesta identidade defina uma solução recursiva para o cálculo da exponencial. Tenha em atenção problemas de eficiência computacional

evitando a duplicação de cálculos.

### Exercício 9.4 M

Escreva um programa que permite eliminar de uma cadeia de caracteres os casos de caracteres repetidos em posições consecutivas. Por exemplo:

```
>>> print removedup('aabccda')
abcda
```

### Exercício 9.5 F DONE

Escreva um programa que dados dois conjuntos determina se um está incluído no outro.

### Exercício 9.6 F DONE

Escreva um programa que dados dois conjuntos determina a sua intersecção.

### Exercício 9.7 M

Um polinómio é uma expressão com a forma:

$$p(x) = \sum_{i=0}^n a_i x^i = a_0 + a_1 x + a_2 x^2 + \dots + a_n x^n$$

Um dos problemas que se coloca, conhecido um polinómio, é saber qual é o seu valor num dado ponto. Existe um método, conhecido por **método de Horner** que permite efectuar o cálculo de acordo com a fórmula:

$$p(x) = \sum_{i=0}^n a_i x^i = a_0 + x(a_1 + x(a_2 + \dots + x(a_{n-1} + a_n x^n)))$$

Implemente uma versão recursiva do método de Horner.

### Exercício 9.8 D

O conjunto potência de um dado conjunto é o conjunto de todos os seus subconjuntos. Implemente o programa que calcula o conjunto potência para um dado conjunto.

### Exercício 9.9 D

Suponha que desenha  $n$  ovais no plano que satisfazem as seguintes condições:

- as ovais intersectam-se duas a duas em, exactamente, dois pontos

- nunca três ovais se encontram no mesmo ponto

Quantas regiões distintas no plano são criadas por essas ovais? Encontre a expressão genérica para um dado  $n$  e escreva o respectivo programa. Nota: É encontrar esse valor para pequenos valores de  $n$ :  $n = 1$  são duas,  $n = 2$  são quatro,  $n = 3$  são 8, como ilustra a figura 9.1.

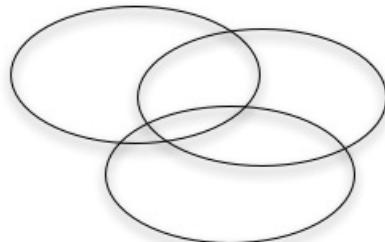


Figura 9.1: Ovais no plano. Caso de  $n = 3$

### Exercício 9.10 M

Retome o exemplo da listagem ?? e faça variar os próprios incrementos. Simule para alguns valores.

### Exercício 9.11 F

Queremos usar o módulo **turtle** para desenhar uma árvore simples como a indicada na figura 9.2.

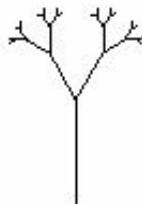


Figura 9.2: Uma árvore recursiva

Desenvolva o respectivo programa recursivo. A figura 9.3 dá uma ideia do processo gerativo.

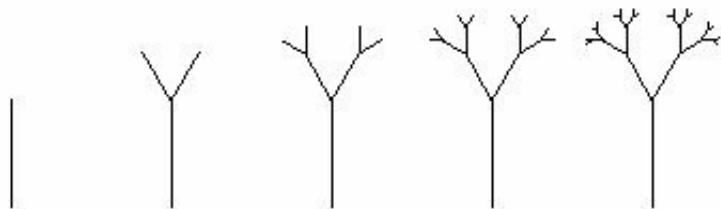


Figura 9.3: O processo

**Exercício 9.12 M**

A árvore do exercício 9.11 não é muito *natural*. Uma maneira de desenhar árvores mais interessantes consiste em desequilibrar a sub árvore direita e a esquerda fazendo, por exemplo os ramos de uma maiores do que a outra. A figura 9.4 ilustra um resultado quando o lado esquerdo é duplo do lado direito. Tente criar o programa recursivo que permite obter estes desenhos.

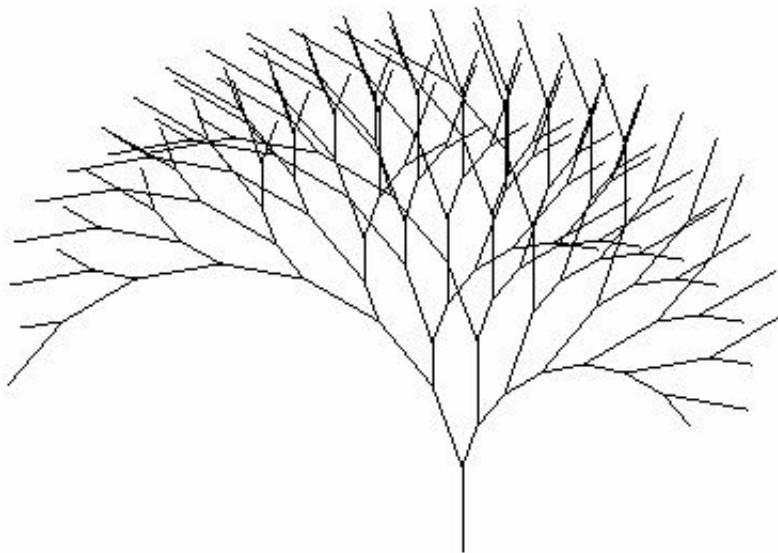


Figura 9.4: Uma árvore mais realista

**Exercício 9.13 M**

Como sabemos uma lista pode ter como elementos listas. Um problema interessante é o de transformar uma lista genérica numa lista plana, isto é, uma lista formada apenas pela sequência dos seus elementos. Exemplo:

```
>>> print aplana([[1,2],[[3]],4, [5,[6],7]])
[1, 2, 3, 4, 5, 6, 7]
```

Desenvolva o respectivo programa recursivo.

**Exercício 9.14 M**

Escreva um programa recursivo que permita determinar se um dado padrão ocorre ou não num dado texto. O que se lhe oferece dizer sobre os custos computacionais da sua solução?

O custo computacional é elevado. Medido pelo número de comparações é igual a  $\mathcal{O}(|pad| \times |text|)$ .

**Exercício 9.15 M**

Suponhamos que temos uma sequência de  $n$  vectores  $(V_1, V_2, \dots, V_n)$ , todos do mesmo cumprimento. Pretende-se um programa que forneça todos os vectores possíveis de comprimento  $n$  formados combinando ordenadamente os elementos dos vectores  $V_i$ . Exemplo:

```
>>> print prod_vectores([[1,2,3],['a','b','c']])
[[1, 'a'], [1, 'b'], [1, 'c'], [2, 'a'], [2, 'b'], [2, 'c'],
 [3, 'a'], [3, 'b'], [3, 'c']]
```

**Exercício 9.16 M**

A multiplicação de matrizes é um processo fundamental em muitas áreas da computação. O seu custo computacional, medido em termos do número de multiplicações e adições dos seus elementos, é bastante elevado. Existe um método recursivo de multiplicação de matrizes, conhecido por método de **Strassen**, bastante eficiente. Se tivermos duas matrizes  $n \times n$ ,  $X$  e  $Y$ , e quisermos calcular  $Z = X \times Y$  sabemos que o número de multiplicações necessárias é da ordem de  $\mathcal{O}(n^3)$ . O método de Strassen consiste em dividir **recursivamente** cada uma das duas matrizes em quatro matrizes de dimensão  $n/2 \times n/2$ . Quando temos matrizes  $2 \times 2$  Strassen encontrou um conjunto de formulas que apenas necessitam de 7 multiplicações e 18 somas e/ou adições. Com este algoritmo a complexidade baiuxa para  $\mathcal{O}(n^{2.81})$ . Vejamos como. Consideremos as matrizes  $X$  e  $Y$  e o seu produto  $Z$ .

$$\begin{pmatrix} z_{11} & z_{12} \\ z_{21} & z_{22} \end{pmatrix} = \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \end{pmatrix} \times \begin{pmatrix} y_{11} & y_{12} \\ y_{21} & y_{22} \end{pmatrix}$$

As fórmulas de Strassen são as seguintes:

$$\begin{aligned} p_1 &= (x_{11} + x_{22}) \times (y_{11} + y_{22}) \\ p_2 &= (x_{21} + x_{22}) \times y_{11} \\ p_3 &= x_{11} \times (y_{12} - y_{22}) \\ p_4 &= x_{22} \times (y_{21} + y_{11}) \\ p_5 &= (x_{11} + x_{12}) \times y_{22} \\ p_6 &= (x_{21} - x_{11}) \times (y_{11} + y_{12}) \\ p_7 &= (x_{12} - x_{22}) \times (y_{21} + y_{22}) \end{aligned}$$

A partir delas podemos computar os elementos da matriz  $Z$ :

$$\begin{aligned} z_{11} &= p_1 + p_4 - p_5 + p_7 \\ z_{12} &= p_3 + p_5 \\ z_{21} &= p_2 + p_4 \\ z_{22} &= p_1 + p_3 - p_2 + p_6 \end{aligned}$$

Para simplificar admita que  $n$  é uma potência de 2, isto é  $n = 2^k$  e implemente o algoritmo.

### Exercício 9.17 F

Um autómato finito é uma máquina de estados usada em várias aplicações informáticas, como por exemplo para implementar um analisador léxical de um compilador. Um **Autómato finito** pode ser definido matematicamente pelo tuplo:

$$\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$$

no qual,  $Q$  é o conjunto de estados do autómato,  $\Sigma$  é o alfabeto de entrada,  $\delta$  é a função de transição entre estados determinada pela leitura de um símbolo do alfabeto de entrada,  $q_0$  é o estado inicial do autómato e  $F$  é o conjunto de estados finais de  $(M)$ . Notar que  $F \subseteq Q$ . Quando usado como reconhecedor de sequências de caracteres do alfabeto de entrada a máquina é colocada no seu estado inicial e vai transitando entre estados à medida que consome os símbolos da sequência. Se quando consumir todos os símbolos

de entrada a máquina se encontrar num dos seus estados finais diz-se que reconheceu a sequência. O que se pretende é uma implementação de um simulador **recursivo** de um autómato finito. O simulador deve ser genérico e não depender de um autómato em particular. A figura 9.5 mostra graficamente um autómato finito que reconhece cadeias de 1 e 0 em que o número de uns é par.

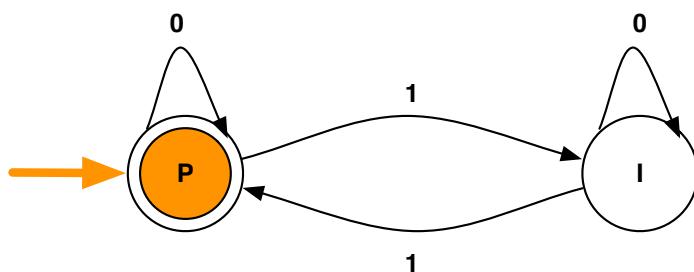


Figura 9.5: Detector de Paridade Par

Para lhe simplificar a vida na listagem 9.1 mostramos como se pode **representar** o autómato indicando explicitamente  $\delta$  (por recurso a um dicionário), o estado inicial  $q_0$  e o conjunto dos estados finais  $F$ .

```

transit={'P': {'0': 'P', '1': 'I'}, 'I': {'0': 'I', '1': 'P'}}
inicial= 'P'
final= ['P']
    
```

Listagem 9.1: 'Detector de Paridade Par'

### Exercício 9.18 Módulo turtle D

A curva conhecida por Floco de Neve<sup>1</sup> forma-se de acordo com uma regra simples. Num dado nível, cada lado é dividido em três partes iguais sendo retirada a parte do meio. De seguida, a partir das extremidades interiores formam-se dois novos segmentos, de tamanho igual a um terço do original, com uma inclinação de  $60^\circ$  em relação aos segmentos que se mantém e que se unem numa das extremidades. A figura 9.6 mostra o processo de construção desta curva para os níveis 1, 2 e 3. Usando o módulo **turtle** desenvolva um programa que permita desenhar a curvas. O valor do lado e o número de níveis a considerar são parâmetros do programa.

### Exercício 9.19 Módulo turtle MD

<sup>1</sup>Do inglês *Snowflake*.

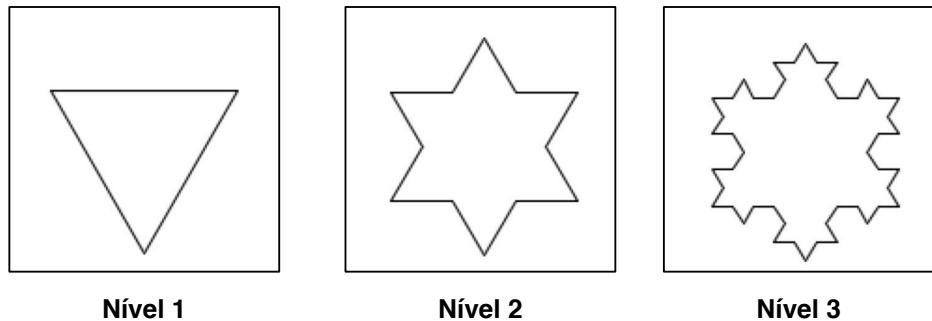


Figura 9.6: Floco de Neve

A curva de Hilbert é um exemplo de curva de preenchimento de espaço. A sua ideia baseia-se em decompor a curva do nível  $n$  em quatro curvas de nível  $n - 1$  ligadas entre si. A figura 9.7 mostra a curva desenhada caso só tenha um nível, dois níveis ou três níveis. Usando o módulo **turtle** desenvolva um programa que permita desenhar a curvas. O valor do lado e o número de níveis a considerar são parâmetros do programa.

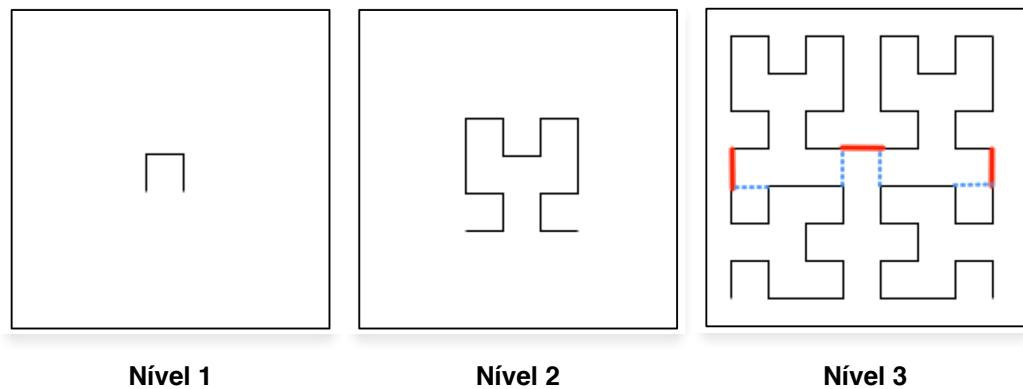


Figura 9.7: Curvas de Hilbert

# Capítulo 10

## Complementos

### Exercício 10.1 F

Considere a sessão seguinte no interpretador.

```
>>> nome = 'ernesto'  
>>> def toto():  
...     nome = 'costa'  
...     return None  
...  
>>> toto()  
>>> nome  
????  
>>>
```

Diga, justificando o que vai aparecer no lugar dos pontos de interrogação.

A primeira ocorrência de nome é distinta da segunda. No primeiro caso estamos perante um nome de nível global , enquanto no segundo caso o nome é local. Como não há comunicação entre os dois espaços o valor do primeiro não é alterado, logo o valor associado continua o mesmo, ou seja, 'ernesto'.

### Exercício 10.2 F

O que vai aparecer no lugar dos pontos de interrogação na sessão abaixo? Porquê?

```
>>> ultima_resposta = 60  
>>> def ultima_maquina():  
...     global ultima_resposta  
...     ultima_resposta = 'Nope!'  
...     return ultima_resposta  
...
```

```
>>>
>>> ultima_maquina()
???
>>> ultima_resposta
???
```

**Exercício 10.3 M**

O que vai aparecer no lugar dos pontos de interrogação na sessão abaixo? Porquê?

```
>>> def f(t=0):
...     def g(t=0):
...         def h():
...             nonlocal t
...             t += 1
...             return h, lambda:t
...         h, gt = g()
...         return h,gt,lambda:t
...
...>>> h,gt,ft = f()
>>> ft(),g()
??? # <---
>>> h()
>>>ft(),gt()
??? # <--
>>>
```

**Exercício 10.4 M**

Uma conta bancária tem como elemento fundamental o seu saldo e a sua gestão durante os movimentos de conta (depósitos e levantamentos). Pretende-se criar um programa que permita definir diversas contas e o seu saldo inicial. Esse mesmo programa é o responsável pela actualização do saldo de cada conta criada. A listagem abaixo dá uma ideia do pretendido.

```
>>> conta_1 = gere_depositos(0) # cria conta com saldo...0
>>> conta_2 = gere_depositos(42) # cria conta com saldo 42
>>> print(conta_1(-10))
Saldo insuficiente
>>> print(conta_2(-10))
32
>>> print(conta_1(25))
```

25

**Exercício 10.5 D**

Implemente um programa que lhe permita criar contadores. Esses contadores devem poder fazer duas acções: contar e ser reiniciados. Alistagem ilustra o pretendido.

```
>>> contador_1 = gere_contador(0)
>>> contador_2 = gere_contador(42)
>>> print(contador_1('conta'))
1
>>> print(contador_2('conta'))
43
>>> print(contador_2('reiniciar'))
0
>>> print(contador_1('conta'))
2
>>> print(contador_2('conta'))
1
>>> print(contador_1('oops'))
Acção desconhecida
```

**Exercício 10.6 D**

Imagine um agente que vai proferindo frases, uma atrás da outra. Use a ideia de fecho e de variáveis não locais para criar um programa que cada vez que é executado (tempo  $t+1$ ) imprime a frase anterior do agente (tempo  $t$ ).  
**Nota:** Tem que prever que na primeira vez que é executado ainda não foi proferida nenhuma frase.

**Exercício 10.7 M**

Considere o código seguinte módulo.

```
def soma_defs(f,g,x):
    return f(x) + g(x)

def func_1(y):
    return y**3

def func_2(z):
    return z + 5 # <-----
```

```
if __name__ == '__main__':
    print(soma_defs(func_1, func_2, 4))
```

Descreva o ambiente através de um diagrama no momento da execução da instrução assinalada por uma seta. Qual o resultado final após a execução.

### Exercício 10.8 MD

Diga como pode desenvolver um programa de traço de execução para algoritmos recursivos. A listagem mostra o pretendido quando aplicado às funções fibonacci e factorial.

```
('|__', 'fib', 3)
('| |__', 'fib', 2)
('| | |__', 'fib', 1)
('| | | |__', 'return', '1')
('| | | |__', 'fib', 0)
('| | | |__', 'return', '1')
('| | | |__', 'return', '2')
('| | |__', 'fib', 1)
('| | | |__', 'return', '1')
('| | | |__', 'return', '3')
3
('|__', 'fact', 4)
('| |__', 'fact', 3)
('| | |__', 'fact', 2)
('| | | |__', 'fact', 1)
('| | | | |__', 'fact', 0)
('| | | | | |__', 'return', '1')
('| | | | | |__', 'return', '1')
('| | | | | |__', 'return', '2')
('| | | | | |__', 'return', '6')
('| | | | |__', 'return', '24')
24
```

### Exercício 10.9 F

Use a ideia que apresentámos sobre geradores para definir o iterador `range`. A sua solução funciona para o caso de números reais?

### Exercício 10.10 F

Implemente uma função geradora para os números naturais pares.

### Exercício 10.11 M

Implemente uma função geradora genérica que permita fornecer um a um os elementos de  $f(x)$  com

$$f : \mathbb{N} \rightarrow \mathbb{N}$$

Os valores de  $x$  também eles são gerados de acordo com uma dada função. Deste modo a função geradora terá dois argumentos funcionais, um para a função  $f$ , o outro para a função que vai gerar os sucessivos valores sobre os quais se aplica a função  $f$ .

### Exercício 10.12 M

Implemente uma função geradora que lhe permita gerar as letras do alfabeto. Deve ser possível indicar a letra inicial. A listagem ilustra o pretendido.

```
>>> gera_1 = letras('c')
>>> next(gera_1)
c
>>>next(gera_1)
d
>>>
```

### Exercício 10.13 M

A função exponencial  $e^x$  pode ser definida pela correspondente série de Taylor:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

Desenvolva um gerador que cada vez que é chamado devolve um valor mais correcto para os valores de  $e^x$ . Use o método para obter o valor aproximado de  $e$ .

### Exercício 10.14 M

Maximizando o uso dos conceitos deste capítulo, implemente uma função que lhe permita determinar se uma lista de objectos está ou não ordenada. Pode apresentar uma solução genérica que permita escolher a função de comparação.

### Exercício 10.15 M

Usando funções de ordem superior implemente um filtro que permita retirar a uma sequência todos os elementos que não satisfazem um dado

critério. Procure que a sua solução seja o mais genérica possível.

### Exercício 10.16 F

Admita que não existe a função pré-definida `min`. Implemente-a socorrendo-se da função `reduce` e de funções anónimas.

### Exercício 10.17 F

Implemente a função factorial, não recursiva, recorrendo à função `reduce`.

### Exercício 10.18 D

Pretende-se transformar uma função com vários argumentos numa cadeia de funções com apenas um argumento. Implemente um programa que permite fazer isso, para o caso de qualquer função com dois argumentos.

```
if __name__ == '__main__':
    my_pow = cadeia_f(pow)
    print(my_pow(2)(3)) # --> 8
```

### Exercício 10.19 M

O produto escalar de dois vectores é dado por:

$$V \times W = \sum_{i=1}^n V_i \times W_i$$

Apresente uma solução baseada em funções de segunda ordem, que resolva o problema de calcular o produto escalar de dois vectores.

**Exercício 10.20 MD** Recorrendo a geradores escreva um programa que permita obter permutações de uma sequência de objectos. **Sugestão:** Tente combinar os geradores com recursividade.

# Capítulo 11

## Desenvolvimentos

### Exercício 11.1 F

Quando discutimos as questões de complexidade apresentámos um exemplo que consistia em determinar se um número aparece repetido numa lista de inteiros positivos. Uma das soluções recorre a uma lista auxiliar, mas apenas funciona se o maior número contido na lista for menor ou igual ao valor do tamanho da lista. Altere o programa para que esta restrição não se aplique.

### Exercício 11.2 F

Altere o código dado no texto por forma a ser possível comparar o custo temporal das três versões do algoritmo que detecta a existência de pelo menos um número repetido numa lista de inteiros positivos. Que conclusões pode tirar?

### Exercício 11.3 M

Apresentámos várias versões em **Python** para o problema de determinar se numa lista existe pelo menos um valor duplicado. Vejamos uma possibilidade.

```
def duplicados(lista):
    for i in range(len(lista)):
        if lista[i] in lista[i+1:]:
            return True
    return False
```

Determine a complexidade temporal do algoritmo usando a notação Grande O. Sabendo que o custo de determinar se um elemento pertence a uma sequência depende linearmente do seu tamanho, que pode dizer sobre o resultado

anteriormente encontrado. Use um profiler para calcular o custo para vectores de diferentes comprimentos. Em que medida os resultados teóricos estão de acordo com o resultado prático?

#### Exercício 11.4 D

O problema das  $n$  ovais consiste em determinar o número de regiões em que podemos dividir o plano recorrendo a  $n$  ovais que têm que satisfazer as seguintes condições:

- as ovais interceptam-se duas a duas em exactamente dois pontos
- nunca três ovais de encontram num mesmo ponto

Encontre uma definição recursiva para o problema que lhe permita implementar um programa. Procure encontrar a complexidade para o respetivo algoritmo.

#### Exercício 11.5 M

No capítulo sobre recursividade apresentámos o algoritmo de procura binária. Apresente a sua complexidade em função da dimensão do vector de entrada. Assuma que esse tamanho é uma potência de 2.

#### Exercício 11.6 D

No capítulo sobre recursividade apresentámos um problema que permite multiplicar matrizes usando uma técnica recursiva (Algoritmo de Strassen). Calcule a complexidade desse algoritmo, função da dimensão das matrizes, e expressa pela soma das multiplicações e adições necessárias para multiplicar as matrizes.

#### Exercício 11.7 F

O valor de  $e^x$  é definido por:

$$e^x = \sum_{i=1}^{\infty} \frac{x^i}{i!}$$

Vamos apresentar dois programas que permitem aproximar esse valor ao considerar a soma dos  $k$  primeiros termos.

```
def exp_e(x,k):
    pot = 1
    for i in range(k):
        pot += pow(x,i+1)/factorial(i+1)
    return pot
```

```
def exp_e_2(x,k):
    pot = 1
    fact = 1
    res = 1
    for i in range(1,k):
        pot *= x
        fact *= i
        res += pot/fact
    return res
```

Teste estes programas para diferentes valores das entradas. Efectue o *profiling* do código e analise o resultado. O que pode dizer sobre as duas versões? Em que medida os resultados obtidos são compatíveis com uma análise formal da complexidade?

### Exercício 11.8 F

Chamam-se palíndromes as cadeias de caracteres que são idênticas quando lidas da esquerda para a direita ou da direita para a esquerda. Um exemplo é a palavra 'radar'. Construa uma solução para o problema de determinar se uma dada cadeia é ou não uma palavra palíndrome. Use o módulo `doctest` para documentar/testar o seu código.

### Exercício 11.9 M

Neste capítulo apresentámos um programa para o cálculo de  $\sum_{i=1}^n i^i$ . Analise o programa do ponto de vista da sua complexidade e apresente o resultado recorrendo à notação Grande O.

### Exercício 11.10 M

Escreva um programa que dado um vector de números, inteiros e/ou em vírgula flutuante, positivos e/ou negativos, identifique o sub-vector cuja soma dos seus elementos é máxima. Por exemplo, para o vector (31,-41,59,26,-53,58,97,-93,-23,84) esse valor é 187 e corresponde ao sub-vector entre as posições 2 (valor 59) e 6 (valor 97). Analise o problema do ponto de vista da sua complexidade e da sua correcção.

### Exercício 11.11 M

Considere os dois programas seguintes que permitem inverter uma cadeia de caracteres.

```
def inverte1(cadeia):
    if cadeia == '':
```

```

    return ''
else:
    return cadeia[-1] + inverte1(cadeia[:-1])

def inverte2(cadeia):
    res = ''
    for car in cadeia:
        res = car + res
    return res

```

A primeira versão é recursiva e a segunda iterativa. Analise os dois casos do ponto de vista da complexidade e da correcção.

### Exercício 11.12 D

Considere o seguinte programa recursivo que permite calcular o mínimo e o máximo de uma lista de inteiros positivos.

```

def min_max(lista):
    comp = len(lista)
    if comp == 1:
        return (lista[0], lista[0])
    elif comp == 2:
        return (minimo_2(lista[0], lista[1]), maximo_2(lista[0], lista[1]))
    else:
        meio = comp//2
        min_1, max_1 = min_max(lista[:meio])
        min_2, max_2 = min_max(lista[meio:])
        return (minimo_2(min_1, min_2), maximo_2(max_1, max_2))

def minimo_2(x,y):
    if x < y:
        return x
    else:
        return y

def maximo_2(x,y):
    if x > y:
        return x
    else:

```

```
return y
```

Faz uso de dois sub-programas para calcular o mínimo e o máximo de dois valores. Determine a complexidade do algoritmo usando o número de comparações como critério.

### Exercício 11.13 M

Considere o seguinte programa iterativo que calcula o mínimo e o máximo de uma lista de inteiros positivos.

```
def min_max_3(lista):
    comp = len(lista)
    if comp == 1:
        return (lista[0], lista[0])
    else:
        min_ = lista[0]
        max_ = lista[0]
        for i in range(1,comp):
            if lista[i] < min_:
                min_ = lista[i]
            if lista[i] > max_:
                max_ = lista[i]
        return (min_, max_)
```

Use um *profiler* para comparar a complexidade deste algoritmo e do seu equivalente recursivo.

### Exercício 11.14 F

Desenvolva um programa que lhe permita imprimir uma tabuada para os números até um dado  $n$ . Use uma abordagem descendente e mostre as diferentes fases de construção do seu programa. Identifique a sua complexidade e apresente um argumento em favor da sua correcção.

### Exercício 11.15 M

Neste capítulo apresentámos uma solução para o problema do ordenamento de um vector de números baseado na ideia da programação por indução. Essa solução recebeu o nome de selecção directa. Vamos procurar usar a mesma abordagem para o problema de ordenamento, alternado no entanto a estratégia. Vamos supor que usamos a abordagem de um jogador de cartas que vai recolhendo cartas do baralho mantendo as que estão na sua mão ordenadas. Cada vez que recolhe uma carta do baralho, identifica a posição onde deve colocar a nova carta e insere-a na sua posição. A figura 11.1 ilustra a ideia.

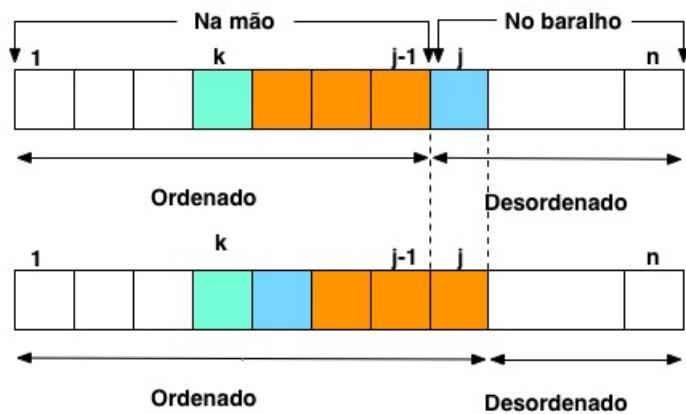


Figura 11.1: Ordenamento por inserção

Desenvolva o seu programa com base na programação por indução. No final faça uma análise da correcção e da complexidade da sua solução.

### Exercício 11.16 M

Um **quadrado mágico** é uma matriz quadrada  $n \times n$ , contendo **todos** os números inteiros positivos de 1 a  $n^2$ , de tal modo que a soma dos valores em cada coluna, linha ou diagonal é o mesmo. A esse número chamamos **número mágico**. Eis um exemplo, com  $n = 3$ , em que o número mágico é igual a 15.

$$\begin{bmatrix} 4 & 9 & 2 \\ 3 & 5 & 7 \\ 8 & 1 & 6 \end{bmatrix}$$

O nosso objectivo é escrever um programa que nos permita **verificar** se um dado quadrado é, ou não, mágico. No caso da resposta ser positiva, queremos também identificar o respectivo número mágico. Construa de modo descendente uma solução para o problema, identificando de modo claro as diferentes etapas no desenvolvimento da solução e as decisões que foram sendo tomadas.

### Exercício 11.17 M

Suponha que tem que manter actualizada uma classificação de uma corrida de ciclismo do tipo contra-relógio individual. Admita, para facilitar, que cada concorrente é caracterizado pelo nome, equipa e tempo gasto. Desen-

volve o respectivo programa.

### **Exercício 11.18 M**

As cadeias de ADN para um informático não são mais do que cadeias de caracteres formadas com símbolos de um alfabeto de quatro letras  $\Sigma = \{A, T, C, G\}$ . Com frequência coloca-se a questão de encontrar a chamada sequência de consenso de um conjunto de cadeias de ADN. Uma sequência de consenso obtém-se do seguinte modo: dada um conjunto de sequências de ADN, todas de igual tamanho, construir uma nova sequência, do mesmo tamanho, em que a base presente numa dada posição será a base que aparece com maior frequência nessa posição em todas as sequências iniciais. É essa a sequência de consenso.

Construa de modo descendente uma solução para o problema, identificando de modo claro as diferentes etapas no desenvolvimento da solução e as decisões que foram sendo tomadas.

### **Exercício 11.19 F**

Modifique a sua solução para o problema da sequência de consenso admitindo agora que existe um quinto símbolo, chamado em inglês de *gap* que pode aparecer nas sequências de ADN. Como pode resolver a questão das sequências não terem todas o mesmo tamanho?

### **Exercício 11.20 MD**

Todos conhecemos o jogo do **Mastermind**. Um jogador A dispõe de um conjunto colorido de piões. Um segundo jogador, B, dispõe igualmente de um conjunto de piões coloridos dos quais escolhe um subconjunto que dispõe de forma ordenada sem conhecimento do primeiro jogador. O objectivo é o jogador A descobrir a sequência escolhida pelo jogador B, no menor número de tentativas. Por cada tentativa o jogador A recebe uma indicação do jogador B acerca da qualidade da sua tentativa: quantas cores e posições acertou e quantas cores acertou sem acertar na posição. Queremos implementar um **simulador** para este jogo. O computador faz o papel do jogador B, enquanto que um humano assumirá o papel de jogador A. O número de cores diferentes, o comprimento da sequência escondida e o número de jogadas máximas possível são parâmetros do programa. A figura 11.2 ilustra um caso simples após duas jogadas.

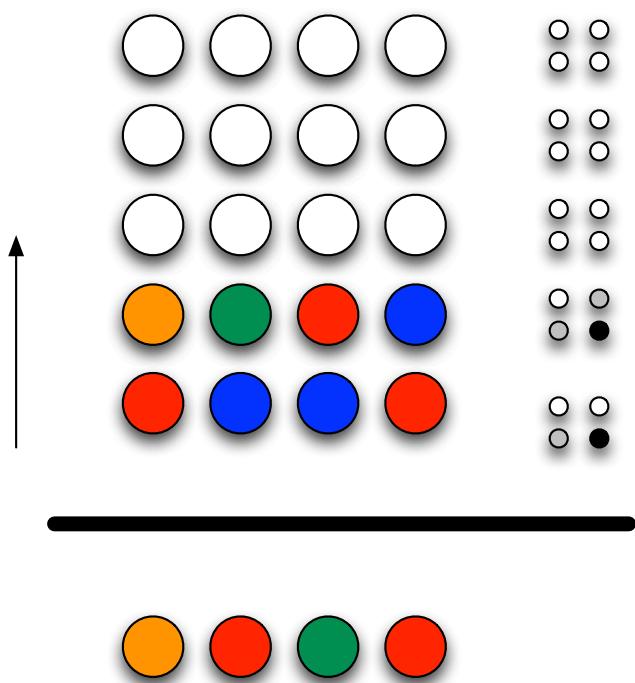


Figura 11.2: Mastermind

# Parte II

## Programação Orientada aos Objectos



# Capítulo 12

## Tipos e Classes

### Exercício 12.1 F

Altere a implementação do tipo de dados abstracto pilha, por forma a que as introduções e as eliminações sejam feitas no início da lista e não no final. Do ponto de vista do custo computacional como compara esta solução com a apresentada neste capítulo?

### Exercício 12.2 F

Um algoritmo simples de conversão de um número inteiro na base 10 para a base 2, baseia-se na ideia de ir dividindo o número sucessivamente por 2 e guardando o resultado do resto. Implemente o algoritmo recorrendo ao TDA pilha para guardar o número binário.

### Exercício 12.3 F

Suponha que quer limitar o tamanho de uma fila. Como pode modificar o TDA apresentado para ter essa limitação em linha de conta?

### Exercício 12.4 M

Na vida real as filas podem estar ordenadas por prioridade, de modo que um elemento quando vai ser inserido na fila não fica necessariamente na última posição, mas antes na posição imediatamente atrás do primeiro elemento da fila que tem uma prioridade maior ou igual à sua. Implemente o respectivo TDA.

### Exercício 12.5 F

Altere o código do avaliador de expressões em notação pósfixa, por forma

a poder ser usado também com números em vírgula flutuante.

### Exercício 12.6 M

Desenvolva um programa que avalie expressões aritméticas simples representadas em notação prefixa.

### Exercício 12.7 D

Desenvolva um programa que lhe permita transformar expressões aritméticas simples em notação infixa na correspondente em notação pós-fixa.

### Exercício 12.8 F

Uma **Deque** é uma coleção ordenada de objectos em que os elementos podem ser inseridos e eliminados nas suas duas *extremidades*. Implemente o respectivo TDA.

### Exercício 12.9 M

Recorre ao TDA **deque** para implementar um programa que resolva o problema de saber se uma dada cadeia de caracteres é ou não uma capicua.

### Exercício 12.10 F

Usando o TDA vector implemente uma operação que lhe permita efectuar o produto escalar de dois vectores.

### Exercício 12.11 F

Usando o TDA vector implemente uma operação que lhe permita efectuar a translação de um vector.

### Exercício 12.12 M

Usando o TDA vector implemente uma operação que lhe permita efectuar a rotação de um vector.

### Exercício 12.13 D

Em programação existe muitas vezes a necessidade de lidar com estruturas bi-dimensionais, por exemplo, matrizes e imagens. Socorrendo-se do TDA Vector implemente um novo TDA, denominado **Array2D**, para objectos bi-dimensionais, cujo interface deve conter um conjunto minimalista de métodos.

### Exercício 12.14 M

Usando o TDA **Array2D** desenvolva um programa que permita efectuar

o produto de duas matrizes.

### Exercício 12.15 F

Desenvolva um programa que implemente o método de travessia em ordem.

### Exercício 12.16 M

Sabemos como inserir um elemento numa árvore binária. Discuta o problema inverso, ou seja, eliminar um nó da árvore e apresente a respectiva implementação.

### Exercício 12.17 M

Duas árvores binárias,  $A$  e  $B$ , dizem-se **isomórficas** se uma das seguintes condições se verificar:

1. As duas árvores são vazias, ou
2. As duas árvores têm o mesmo número de sub-árvores não vazias, e estas são isomórficas entre si.

Desenvolva um programa que dadas duas árvores binárias diga se elas são ou não isomórficas.

### Exercício 12.18 D

As **árvores binárias de procura** são uma estrutura de dados importante em computação. São um caso especial de árvore binária (AB). Se tivermos uma árvore binária, de raiz  $elem$ , sub-árvore esquerda  $AB_1$  e sub-árvore direita  $AB_2$ , ela diz-se uma árvore binária de procura se satisfizer as seguintes condições:

- 1) os elementos de  $AB_1$  são todos inferiores a  $elem$  e se  $AB_1$  for também uma AB de procura;
- 2) os elementos de  $AB_2$  forem todos superiores a  $elem$  e se  $AB_2$  for também uma AB de procura.

Defina o novo tipo de dados **ArvoreBinariaProcura**. Use como referência o TDA **ArvoreBinaria**.

### Exercício 12.19 M

Escreva um programa recursivo que dado um elemento e uma árvore binária de procura indique se o elemento pertence ou não à árvore. Altere a

sua solução por forma a também devolver a sub-árvore que tem esse elemento por raíz.

**Exercício 12.20 M**

Desenvolva um programa que permita determinar se um dado elemento está presente ou não numa dada árvore binária de procura.

# Capítulo 13

## Programação Orientada aos Objectos

### Exercício 13.1 MF

Suponha que define uma classe **Ponto**, que representa pontos num espaço 2D pelas suas coordenadas. Admita ainda que lhe vamos associar dois métodos, um para mover o ponto e outro para calcular a distância entre dois pontos.

```
class Ponto:

    def __init__(self, x, y):
        self._x = x
        self._y = y

    def obtem_x(self):
        return self._x

    def obtem_y(self):
        return self._y

    def move(self,x,y):
        self._x = x
        self._y = y

    def distancia(self,outro):
        dif_x = (self._x - outro.obtem_x()) ** 2
        dif_y = (self._y - outro.obtem_y()) ** 2
        return (dif_x + dif_y) ** (1/2)
```

Execute o código seguinte, observe os resultados obtidos e comente.

```
if __name__ == '__main__':
    p_1 = Ponto(2,3)
    p_2 = Ponto(5,8)
    print(p_1.obtem_x())
    print(p_2.obtem_y())
    p_2.move(8,2)
    print(p_2.obtem_y())
    print(p_1.distancia(p_2))
    print(p_2.distancia(p_1))
```

**Exercício 13.2 F**

Retome o problema 13.1. Suponha agora que pretende ser possível criar um ponto sem indicar expressamente as suas coordenadas. Diga como ficaria implementada a classe.

**Exercício 13.3 F**

Pretende-se uma nova alteração à classe ponto por forma a ser possível criar um ponto através das suas coordenadas como no exercício 13.1, ou no caso de não serem dadas as coordenadas estas serem, por defeito, 0. Modifique a definição da classe.

**Exercício 13.4 F**

Considere o jogo do poker de cartas. De um modo simples, o jogo envolve um baralho de cartas, vários jogadores e um distribuidor. Identifique as classes envolvidas e suas relações. Desenhe o respectivo diagrama de classes.

**Exercício 13.5 M**

Retome o problema das pessoas ligadas a uma instituição académica discutido no livro. Complete o desenho identificando os diferentes atributos e métodos de cada classe. Identifique o modo como usou o mecanismo de herança. Existe interesse em usar polimorfismo?

**Exercício 13.6 M**

Suponha que é dono de um pequeno super-mercado e quer implementar uma pequena aplicação para registar a compras dos clientes e emitir a respectiva factura. Para tal vai implementar uma classe, *CaixaRegistadora*. Identifique os atributos e os métodos respectivos, sendo que, para cada cliente, deve ser possível acrescentar os produtos um a um, saber o valor actual a pagar, saber o valor final a pagar, eliminar o último produto introduzido

ou limpar todo o registo.

### **Exercício 13.7 M**

Suponha que quer implementar um contador à semelhança do que foi discutido no livro. Agora no entanto pretende prever um método que lhe permita decrementar o contador. Implemente a sua solução que deve ter em atenção que não pode decrementar o contador mais vezes do que o incrementou.

### **Exercício 13.8 M**

Suponha que quer implementar uma solução para o problema de ter várias entradas num recinto desportivo e querer controlar o número total de entradas. Admita que tem um contador associado a cada uma das entradas. Construa uma solução baseada na ideia de uma classe chamada **Contador** de que cada um dos contadores de cada entrada é uma instância. A sua solução deve prever um limite máximo de entradas.

### **Exercício 13.9 M**

Suponha que quer desenvolver uma aplicação para gerir o voto nas eleições para Presidente da República. A sua aplicação deve permitir as acções mais básicas tais como o voto, a contagem dos votos em qualquer momento. Não deve ser permitido qualquer tipo de batota. Identifique as classes da sua aplicação, e para cada uma delas quais os seus atributos e métodos.

### **Exercício 13.10 F**

Considere o diagrama de classes da figura 13.1. Implemente em **Python** e verifique o funcionamento do polimorfismo.

### **Exercício 13.11 F**

Suponha que tem que administrar contas bancárias. admita que as contas podem ser contas à ordem, contas a prazo ou contas poupança, todas com um custo de manutenção associado embora com diferentes modo de serem calculados. Apresente uma solução para o problema baseada na ideia de (super) classe abstracta.

### **Exercício 13.12 F**

Apresentámos um exemplo simples de um robô que habita um mundo que é uma grelha quadrada. Altere a solução de modo a que a vizinhança seja agora formada por todas as oito células envolventes.

### **Exercício 13.13 M**

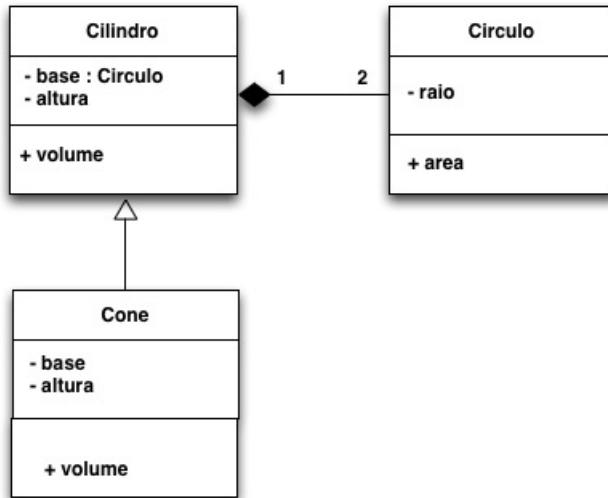


Figura 13.1: Circulos, cilindros e cones

Apresentámos um exemplo simples de um robô que habita um mundo que é uma grelha quadrada. Altere a solução inicial apresentada por forma a que:

- possa existir mais do que um robô na mesma célula
- o construtor do robô incluir uma chamada ao método de registo do mundo
- em vez de termos uma grelha na forma de matriz inicializada com o indicador de célula vazia ('\*'), termos em sua substituição uma lista dos robôs existentes no mundo.

#### **Exercício 13.14 M**

Apresentámos um exemplo simples de um robô que habita um mundo que é uma grelha quadrada. O robô pode movimentar-se de modo aleatório, mas a implementação apresentada apenas faz uma tentativa para encontrar

uma célula vazia na vizinhança. Rescreva este método por forma a que o robô tente tantas vezes quantas as necessárias a procura de uma célula apropriada.

### Exercício 13.15 M

Apresentámos um exemplo simples de um robô que habita um mundo que é uma grelha quadrada. O mundo é fechado, pelo que nem todas as células têm o mesmo número de vizinhos. Admita uma nova versão em que o mundo tem uma forma toroidal, pelo que todas as células terão sempre o mesmo número de vizinhos (a figura 13.2 ilustra a ideia.). Implemente uma nova versão do sistema só com um tipo de robôs para este tipo de mundo.

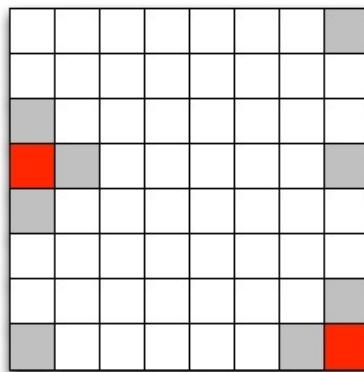


Figura 13.2: Vizinhanças num mundo toroidal.

### Exercício 13.16 M

Apresentámos um exemplo que envolve robôs do tipo predador e presa e que habitam um mundo que é uma grelha quadrada. Complete a implementação das Presas por forma a terem um comportamento de **fuga** sempre que tiverem um predador na sua vizinhança. Pense numa solução em que a presa escolhe uma célula célula da sua vizinhança que não faz parte da vizinhança do predador.

### Exercício 13.17 D

Apresentámos um exemplo que envolve robôs do tipo predador e presa e que habitam um mundo que é uma grelha quadrada. Acrescente aos robôs o comportamento de reprodução. Por reprodução entende-se a criação de um novo robô do mesmo tipo que é colocado numa das células vizinhas que esteja livre. Com a reprodução o progenitor perde metade da sua energia

que é passada para o sucessor. Existe um tempo mínimo necessário para que um robô se possa reproduzir.

**Exercício 13.18 MD**

Considere de novo o mundo habitado por robôs agora com todas as características seguintes:

- mundo toroidal
- vizinhança formada pelas oito células envolventes
- mais do que uma entidade pode habitar uma mesma célula
- o mundo em vez de uma matriz é representado pela lista das entidades que o habitam
- para além de predadores e presas podem existir outras entidades como plantas comestíveis
- as plantas crescem ao longo do tempo ocupando as células vizinhas livres
- predadores e presas movem-se e reproduzem-se
- existe um tempo mínimo para a reprodução, podendo ser diferente para predadores e presas
- os predadores comem presas e plantas, dando prioridade às primeiras
- as presas comem plantas e fogem dos predadores

Implemente uma solução e inclua um simulador que permita:

- introduzir o conceito de tempo de simulação
- definir o número de predadores, presas e plantas no mundo inicial
- definir aleatoriamente as posições iniciais dos predadores, presas e plantas
- dar prioridade a comer face a mover
- visualizar a evolução do mundo ao longo do tempo

**Exercício 13.19 M**

Retome o problema das contas bancárias e desenvolva uma aplicação que permite criar contas dos diversos tipos, cada uma com um número apropriado. Identifique os atributos e os métodos que permitam as operações básicas com as contas (por exemplo, depósitos, pagamentos, consulta de saldos). Em que medida usou variáveis de classe?

**Exercício 13.20 M**

Uma empresa é formada para além do edifício, por diversos departamentos (e.g., gestão, marketing, produção), por funcionários (e.g., operários, chefes de departamento, CEO) cada um com diferentes funções. A empresa produz algo e depende de fornecedores. Identifique as diferentes classes relevantes de uma aplicação que modele a empresa e as respectivas relações entre classes. Implemente o respectivo sistema.

**Exercício 13.21 MD**

Vamos supor quer pretendemos implementar um simulador de circuitos lógicos como o da figura 13.3. O simulador deve ser genérico, isto é, deve funcionar para qualquer circuito.

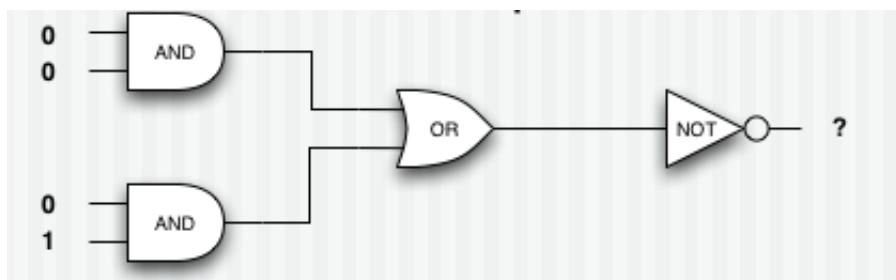


Figura 13.3: Exemplo de circuito lógico

O simulador o que faz é fornecer o valor da saída conhecidos os valores das entradas. Os circuitos lógicos são constituídos por diferentes tipos de portas lógicas ligadas de uma certa maneira. A porta **AND** produz uma saída 1 quando ambas as entradas são também 1, sendo 0 em todas as outras combinações possíveis. A porta **OR** tem como saída 0 sempre que ambas as entradas forem 0, sendo 1 em todos os outros casos. A porta **NOT** gera à saída o sinal inverso da entrada. As portas lógicas formam uma hierarquia como a identificada na figura 13.4.

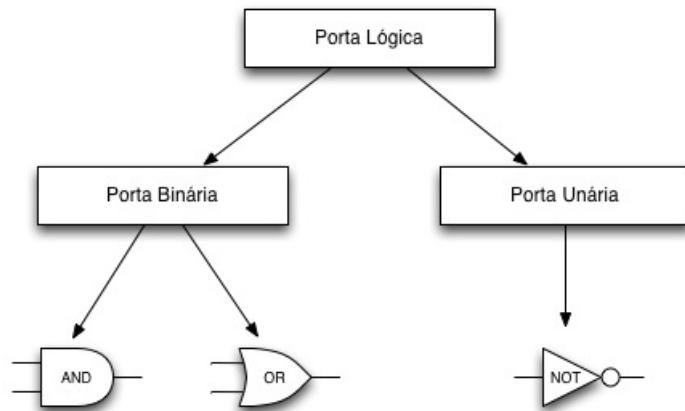


Figura 13.4: A hierarquia das portas lógicas

Identifique os diferentes objectos e respectivas classes, determinando ainda os atributos e comportamentos de cada uma. Implemente o sistema.

# Capítulo 14

## Interfaces Gráficas com o Utilizador

### Exercício 14.1 MF

Desenvolva uma aplicação gráfica cuja interface é a da figura 14.1.



Figura 14.1: GUI simples

### Exercício 14.2 MF

Escreva um programa que construa uma interface gráfica composta por dois botões, um de nome OK e outro de nome KO.

### Exercício 14.3 F

Desenvolva uma aplicação gráfica que permita determinar, conhecida a distância a percorrer e o tempo disponível, qual a velocidade média a que nos devemos deslocar. Em que medida pode alterar a sua implementação para calcular qualquer um dos elementos conhecidos os outros dois?

### Exercício 14.4 F

Escreva um programa que construa uma interface gráfica composta por dois botões, um de nome OK e outro de nome KO. Sempre que carregar no botão OK deve ser impressa a mensagem *Botão OK pressionado!*, se for o botão KO a mensagem deve ser *Botão KO pressionado*.

### Exercício 14.5 F

Desenvolva o programa que permite construir a interface gráfica da figura 14.2. Use cores distintas para os diferentes elementos.

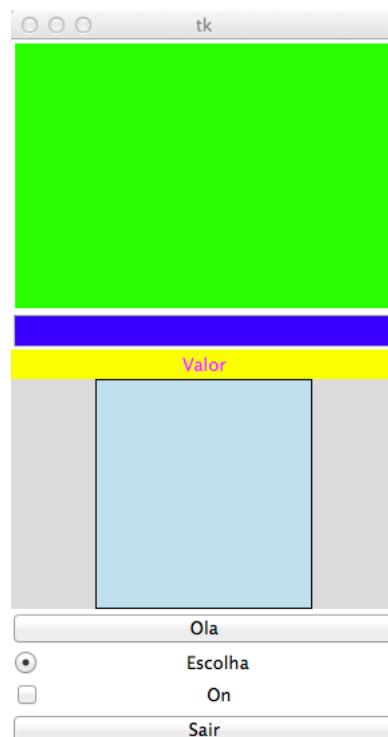


Figura 14.2: Vários componentes

### Exercício 14.6 F

Desenvolva uma aplicação gráfica que permita calcular o salário bruto, o salário líquido e os descontos, conhecidos o número de horas que foram trabalhadas e o custo de uma hora de trabalho.

### Exercício 14.7 M

Use o gestor de posicionamento `grid` para gera a interface da figura 14.3.

### Exercício 14.8 M

Desenvolva uma interface gráfica que lhe permita apresentar/construir a ficha pessoa de alguém como se ilustra na figura 14.4. Notar o alinhamento das etiquetas.

### Exercício 14.9 M



Figura 14.3: Usando Grid



Figura 14.4: Ficha Pessoal

Desenvolva uma aplicação gráfica que lhe permite desenhar um quadrado como o da figura 14.5 cada vez que carrega no botão.

### Exercício 14.10 M

Desenvolva uma aplicação gráfica que lhe permite desenhar um quadrado como o da figura 14.5 cada vez que carrega no botão *Tabuleiro*. Cada vez que carregar no botão *Piao* deve desenhar uma circunferência colorida num dos quadrados dos tabuleiro (ver figura 14.6).

### Exercício 14.11 M

Desenvolva um programa com uma interface gráfica que permita alterar a cor de uma tela. A figura 14.7 exemplifica o pretendido.

Sempre que introduzir o nome de uma cor, essa cor cobre a tela.

### Exercício 14.12 M

Apresentámos neste capítulo um exemplo de animação que consistia num texto a mover-se horizontalmente numa tela. Altere o programa por forma ser possível controlar o movimento. Por exemplo, deve ser possível parar e

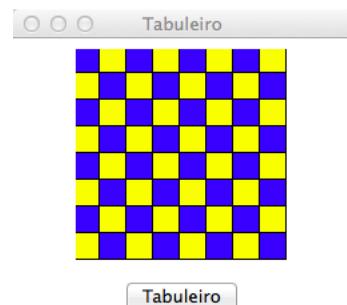


Figura 14.5: Tabuleiro colorido

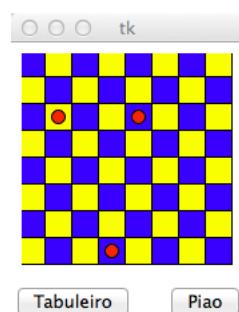


Figura 14.6: Piões num tabuleiro

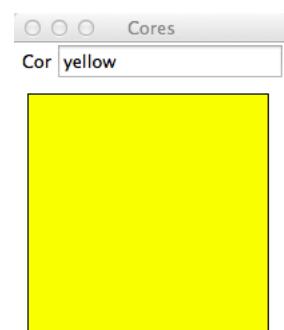


Figura 14.7: Mudando de cor

retomar o movimento, acelerar ou travar o movimento.

### **Exercício 14.13 M**

Desenvolva uma aplicação com interface gráfico para converter uma temperatura de graus Fahrenheit para Celsius. Altere a sua solução para que seja possível converter nos dois sentidos.

### **Exercício 14.14 M**

O ADN de cada um de nós pode ser resumido a uma longa sequência de quatro símbolos (bases): A, T, C e G. Pretende-se desenvolver uma aplicação que conte o número de vezes que cada símbolo ocorre. A aplicação deve ser suportada por um GUI que permita indicar a cadeias de ADN, um botão para desencadear a contagem e um campo para mostrar cada um dos resultados.

### **Exercício 14.15 F**

Desenvolva uma aplicação com interface gráfica que permite verificar a autenticidade de uma *password* introduzida pelo utilizador e que, no caso de erro mostre uma mensagem de erro.

### **Exercício 14.16 M**

Neste capítulo apresentámos um pequeno exemplo de editor de texto que nos permite criar um ficheiro e salvá-lo recorrendo a um menu na barra de menus. O exemplo dado era muito primitivo. Pretende-se que complete o editor por forma a abrir também um ficheiro já existente, editá-lo e guardar a nova versão. Procure que o seu código seja robusto, por exemplo, que ele possa funcionar correctamente no caso de querer abrir um ficheiro inexistente.

### **Exercício 14.17 M**

Vamos construir um editor de texto muito simples. A interface gráfica da aplicação deve conter, três botões, um para abrir, outro salvar e ainda outro criar um ficheiro novo, um campo para introduzir o nome do ficheiro e um campo para visualizar e alterar o ficheiro. Deve ser possível passar no texto por meio de uma *scrollbar* vertical.

### **Exercício 14.18 M**

Descrevemos o essencial da nossa implementação do jogo do galo mas ainda falta juntar muitas peças do puzzle. Mesmo sabendo que pode obter o código no sítio do livro, implemente uma versão completa do jogo, para a descrição apresentada

### **Exercício 14.19 D**

Neste capítulo apresentámos uma solução para uma calculadora simples. Vamos agora desenhar uma que tenha uma forma mais próxima das calculadoras reais. A figura 14.8 exemplifica o que se pretende.

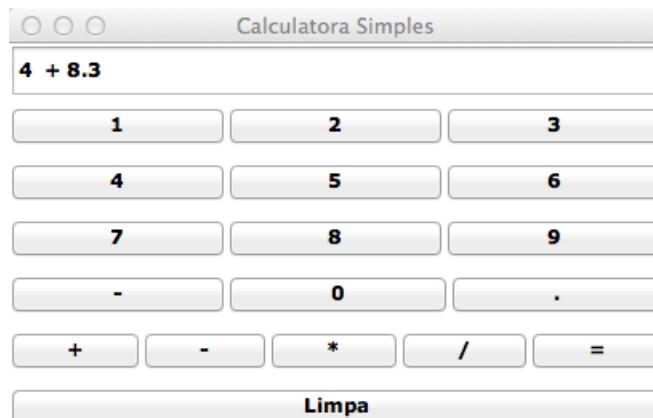


Figura 14.8: Calculadora Simples

### Exercício 14.20 D

A versão do jogo do galo deste capítulo supõe dois jogadores humanos. Implemente uma nova versão em que um humano joga contra o computador.

### Exercício 14.21 MD

Quando abordámos o tema da recursividade no livro, apresentámos o exemplo das **Torres de Hanói**. Pretende-se que desenvolva um programa baseado no `tkinter` que apresente uma animação da construção da solução.

### Exercício 14.22 MD

Os autómatos celulares (AC) são modelos formais que permitem simular diferentes tipos de sistema. Procure informação sobre os AC e construa uma interface gráfica que permita definir os seus parâmetros e visualizar o seu funcionamento. Admita que se trata de um AC a uma dimensão.

### Exercício 14.23 MD

Os Sistemas de Lindenmayer são sistemas formais usados para compreender fenómenos como o desenvolvimento das plantas. São também muito usados na construção de ambientes artificiais para jogos e cinema. Procure na internet a sua definição. Desenvolva um programa com uma interface gráfica que lhe permita introduzir os parâmetros do sistema, simular o seu

funcionamento e visualizar o resultado usando a interpretação baseada no conceito de tartaruga.



## Bibliografia