

Folha prática 4 Introdução à Programação, DCC/FCUP, 2009/2010

1. 0 1 1 2 3 5 8 11 ...

A *sucessão de Fibonacci* é definida pela recorrência

$$\begin{cases} F(0) = 0 \\ F(1) = 1 \\ F(n) = F(n-1) + F(n-2) \quad \text{para } n \geq 2 \end{cases}$$

Os termos iniciais da sucessão são pois: 0, 1, 1, 2, 3, 5, 8, 13 ...

- (a) Escreva uma função recursiva `nfib(n)` que calcule o n -ésimo termo da sucessão. A função deve ser directamente baseada na definição dada.
- (b) Execute no interpretador `nfib(10)`, `nfib(20)`, `nfib(30)` e `nfib(40)`. O que acontece ao tempo de execução quando n aumenta?
- (c) Escreva uma função `tfib(n)` que calcule o número de Fibonacci de ordem n de forma iterativa. *Sugestão*: cada termo é a soma dos dois anteriores; use 2 variáveis contendo o último e o penúltimo termos calculados.
- (d) Experimente calcular `tfib(40)` e compare o tempo de execução com a versão recursiva.

2. É uma letra?

Escreva uma função `testaLetra(c)` que testa se um caracter `c` é uma letra maiúscula ou minúscula (não acentuada) e retorna um valor lógico. Exemplos:

```
>>> testaLetra('a')
True
>>> testaLetra('E')
True
>>> testaLetra('1')
False
```

Nota. Não se assume que os códigos das letras são sequenciais; use o predicado `'in'` e as variáveis

```
minus = "abcdefghijklmnopqrstuvwxyz"
maius = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
```

3. Quantas letras?

O programa seguinte conta o número de letras `'a'` numa `'string'`:

```
pal = "banana"
conta = 0
for char in pal:
    if char == 'a':
        conta = conta + 1
print conta
```

Escreva uma função chamada `contaCar(txt,c)` que generalize o programa dado para contar o número de ocorrências do caracter `c` numa 'string' `txt`.

4. **Sem sinais...**

Escreva uma função `removeSinais(txt)` que remova de uma 'string' os sinais de pontuação "-", ".", ",", ";", "!" e "?". Exemplo:

```
>>> removeSinais('Ola!, -- disse ele...')
'Ola disse ele'
```

Nota: pode testar se `x` é um sinal de pontuação com

```
x in "-.,:~!?"
```

5. **Sem espaços a mais...**

Escreva uma função `removeEspacos(txt)` que remova espaços seguidos duma 'string' `txt`, substituindo-os por um único espaço. Por exemplo:

```
>>> removeEspacos(' Ola, polvo !')
'Ola, polvo !'
```

6. **Ao contrário...**

Escreva uma função `reverso(txt)` que calcule a 'string' dada por ordem inversa. Por exemplo:

```
>>> reverso('Viva o povo!')
'!ovop o aviV'
```

7. **arara**

Uma palavra é um *palíndromo* se se lê de forma igual da esquerda para a direita e vice-versa. Exemplo: 'reviver' é um palíndromo.

- (a) Escreva uma função `palindromo(txt)` que verifica se uma 'string' é um palíndromo. O resultado deve ser `True` ou `False`.
- (b) Mais geralmente, uma frase é palíndromo se se lê da mesma forma nos dois sentidos, permitindo-se o ajustamento de espaços entre letras, sinais de pontuação e/ou a troca de maiúsculas e minúsculas. Assim, são palíndromos:

```
"Amora me tem aroma."
"Madam, I'm Adam."
"A man, a plan, a canal: Panama"
```

Altere a função `palindromo` para testar se uma frase é um palíndromo neste sentido mais geral.

8. Codificar mensagens...

A *cifra de César* é um dos métodos mais simples para codificar um texto: cada letra é substituída por outra a uma distância fixa k no alfabeto. Por exemplo, para $k = 3$, a substituição é

<i>A</i>	<i>B</i>	<i>C</i>	<i>D</i>	<i>E</i>	...	<i>W</i>	<i>X</i>	<i>Y</i>	<i>Z</i>
↓	↓	↓	↓	↓	...	↓	↓	↓	↓
<i>D</i>	<i>E</i>	<i>F</i>	<i>G</i>	<i>H</i>	...	<i>Z</i>	<i>A</i>	<i>B</i>	<i>C</i>

e o texto "ATAQUE DE MADRUGADA" é transformado em "DWDTXH GH PDGUXJDGD".

- (a) Escreva a função `cifra(txt,k)` que codifica uma 'string' usando o deslocamento $0 \leq k \leq 25$. Para simplificar, pode assumir que o texto está todo em letras maiúsculas.
- (b) Escreva a função `decifra(txt,k)` para decodificar um texto cifrado. Consegue definir esta função à custa da anterior?

9. Revisão das listas...

Prever o comportamento de cada um dos seguintes comandos Python. Teste as suas previsões no interpretador de comandos.

```
>>> a=[1,[],5,[[2]],9,"ui"]
>>> a[0]
--> ???
>>> a[3][0]
--> ???
>>> a[3][0][0]
--> ???
>>> len(a)
--> ???
>>> a[3]=a[0]
>>> a
--> ???
>>> b=range(1,5)
>>> b
>>> [b[2],b[3]] = [b[3],b[2]]
>>> b
--> ???
>>> a+b
--> ???
```