

Capítulo 4

Tuplos e Ciclos Contados

1. Considere as seguintes instruções em Python:

```
soma = 0
i = 20
while i > 0:
    soma = soma + 1
    i = i - 2
print('Soma =',soma)
```

Escreva instruções equivalentes utilizando um ciclo `for`.

2. Escreva em Python a função `explode` que recebe um número inteiro positivo, verificando a correção do seu argumento, e devolve o tuplo contendo os dígitos desse número, pela ordem em que aparecem no número. Por exemplo

```
>>> explode(34500)
(3, 4, 5, 0, 0)
>>> explode(3.5)
ValueError: explode: argumento não inteiro
```

3. Escreva em Python a função `implode` que recebe um tuplo contendo algarismos, verificando a correção do seu argumento, e devolve o número inteiro contendo os algarismos do tuplo, pela ordem em que aparecem. Por exemplo

```
>>> implode((3, 4, 0, 0, 4))
34004
>>> implode((2, 'a', 5))
ValueError: implode: elemento não inteiro
```

Escreva duas versões da sua função, uma utilizando um ciclo `while` e outra utilizando um ciclo `for`.

4. Escreva uma função em Python que recebe um número inteiro positivo maior do que zero e que devolve um tuplo contendo os algarismos codificados desse número do seguinte modo: (a) cada algarismo par é substituído pelo número par seguinte, entendendo-se que o número par seguinte a 8 é o 0; (b) cada algarismo ímpar é substituído pelo número ímpar anterior, entendendo-se que o número ímpar anterior a 1 é o 9. Por exemplo,

```
>>> num_para_seq_cod(1234567890)
(9, 4, 1, 6, 3, 8, 5, 0, 7, 2)
```

5. Escreva em Python a função `filtra_pares` que recebe um tuplo contendo algarismos, verificando a correção do seu argumento, e devolve o tuplo contendo apenas os algarismos pares. Por exemplo

```
>>> filtra_pares((2, 5, 6, 7, 9, 1, 8, 8))
(2, 6, 8, 8)
```

6. Duas palavras de igual comprimento dizem-se “amigas” se o número de posições em que os respetivos caracteres diferem for inferior a 10%. Escreva uma função em Python que recebe como argumentos duas cadeias de caracteres e devolve verdadeiro se os seus argumentos corresponderem a palavras amigas e falso em caso contrário. Por exemplo:

```
amigas('amigas', 'amigas')
True
amigas('amigas', 'asigos')
False
```

7. Escreva em Python uma função, `cc_para_int`, que converte uma cadeia de caracteres num inteiro usando o código ASCII. Use o facto que a representação de um carácter tem três algarismos. Por exemplo

```
>>> cc_para_int('bom dia')
98111109032100105097
```

8. Considere a gramática em notação BNF:

```
<idt> ::= <letras> <numeros>
<letras> ::= <letra> |
            <letra> <letras>
<numeros> ::= <num> |
             <num> <numeros>
<letra> ::= A | B | C | D
<num> ::= 1 | 2 | 3 | 4
```

Escreva uma função em Python, chamada `reconhece`, que recebe como argumento uma cadeia de caracteres e devolve *verdadeiro* se o seu argumento corresponde a uma frase da linguagem definida pela gramática e *falso* em caso contrário. Por exemplo,

```
>>> reconhece('A1')
True
>>> reconhece('ABBBBCDDDD23311')
True
>>> reconhece('ABC12C')
False
```

9. Considere a seguinte gramática em notação BNF, na qual o símbolo inicial é $\langle \text{prim} \rangle$:

```
 $\langle \text{prim} \rangle ::= a \langle \text{seg} \rangle a \mid a \langle \text{prim} \rangle a$ 
 $\langle \text{seg} \rangle ::= b \langle \text{seg} \rangle b \mid b \langle \text{ter} \rangle b$ 
 $\langle \text{ter} \rangle ::= c \mid c \langle \text{ter} \rangle$ 
```

Escreva uma função em Python, chamada **reconhece**, que recebe como argumento uma cadeia de caracteres e devolve *verdadeiro* se o seu argumento corresponde a uma frase da linguagem definida pela gramática e *falso* em caso contrário. Por exemplo,

```
>>> reconhece('abccccba')
True
>>> reconhece('abccccb')
False
>>>> reconhece('abc')
False
```

10. Escreva uma função em Python que recebe como parâmetros duas cadeias de caracteres de comprimento 1 (**C1** e **C2**) e escreve todas as possíveis combinações de três letras contendo os caracteres compreendidos entre **C1** e **C2**. Por exemplo, se **C1** for 'a' e **C2** for 'c' então algumas das combinações possíveis são: **aaa aab aac aba abb abc aca acb acc baa ...**
11. Um método básico para codificar um texto corresponde a isolar os caracteres nas posições pares para um lado e os caracteres nas posições ímpares para outro, juntando depois as duas partes anteriormente obtidas. Por exemplo, o texto **abcde** é codificado por **acebd**.

- (a) Defina uma função que codifica uma cadeia de caracteres de acordo com o algoritmo apresentado. Não é necessário validar os dados de entrada. Por exemplo,

```
>>> codifica('abcde')
'acebd'
```

- (b) Defina uma função que descodifica uma cadeia de caracteres de acordo com o algoritmo apresentado. Não é necessário validar os dados de entrada. Por exemplo,

```
>>> descodifica('acebd')
'abcde'
```