Capítulo 3

Funções

1. Escreva uma função com o nome `cinco` que tem o valor `True` se o seu argumento for 5 e `False` no caso contrário. Não pode utilizar uma instrução `if`.

2. Defina uma função com o nome `horas_dias` que recebe um inteiro correspondente a um certo número de horas e que tem como valor um número real que traduz o número de dias correspondentes ao seu argumento. Por exemplo

   ```
   >>> horas_dias(48)
   2.0
   >>> horas_dias(10)
   0.4166666666666667
   ```

3. Defina uma função com o nome `area_circulo` que recebe o valor do raio de um círculo e tem como valor a área do círculo. Note-se que a área do círculo cujo raio é r é dada por πr^2. Use o valor 3.14 para o valor de π.

4. Utilizando a função `area_circulo` do exercício anterior, escreva uma função com o nome `area_coroa` que recebe dois argumentos, `r1` e `r2`, e tem como valor a área da coroa circular de raio interior `r1` e raio exterior `r2`. A sua função deverá gerar um erro de valor (`ValueError`) se o valor de `r1` for maior que o valor de `r2`.

5. Usando um ciclo for, escreva uma função em Python que recebe uma quantia em Euros e calcula o número de notas de 50 €, 20 €, 10 €, 5 € e moedas de 2 €, 1 €, 50 cêntimos, 20 cêntimos, 10 cêntimos, 5 cêntimos, 2 cêntimos e 1 cêntimo, necessário para perfazer, essa quantia, utilizando sempre o máximo número de notas e moedas para cada quantia, da mais elevada, para a mais baixa.

6. Escreva uma função em Python com o nome `bissexto` que recebe um número inteiro correspondente a um ano e que devolve `True` se o ano for

bissexto e `False` em caso contrário. Um ano é bissexto se for divisível por 4 e não for divisível por 100, a não ser que seja também divisível por 400. Por exemplo, 1984 é bissexto, 1100 não é, e 2000 é bissexto. por exemplo:

```
>>> bissexto(1984)
True
>>> bissexto(1985)
False
>>> bissexto(2000)
True
```

7. Defina uma função com o nome `dias_mes` que recebe uma cadeia de caracteres, correspondentes às 3 primeiras letras (minúsculas) do nome de um mês e um ano, e tem como valor um número inteiro correspondendo ao número de dias desse mês. No caso de uma cadeia de caracteres inválida, a sua função deverá gerar um erro de valor (`ValueError`). Use a função `bissexto` do exercício anterior. A sua função deve permitir gerar a interação:

```
>>> dias_mes('jan', 2017)
31
>>> dias_mes('fev', 2016)
29
>>> dias_mes('MAR', 2017)
ValueError: Mes não existe
```

8. Quando se efectua um depósito a prazo de uma quantia q com uma taxa de juros j $(0 < j < 1)$, o valor do depósito ao fim de n anos é dado por:

$$q \times (1 + j)^n$$

   (a) Escreva em Python a função `valor` que recebe como argumentos um número inteiro positivo q correspondente à quantia depositada, um real j no intervalo $]0, 1[$ correspondente à taxa de juros e um inteiro positivo n correspondente ao número de anos que o dinheiro está a render, e, verificando a correcção dos argumentos, devolve um real correspondente ao valor do depósito ao fim desse número de anos. Caso os argumentos não estejam correctos, deverá gerar um erro. Por exemplo,

```
>>> valor(100, 0.03, 4)
112.55088100000002
```

   (b) Usando a função da alínea anterior, escreva uma função que calcula ao fim de quantos anos consegue duplicar o seu dinheiro. Não é necessário validar os dados de entrada. Por exemplo,

```
>>> duplicar(100, 0.03)
24
```

9. Escreva uma função em Python com o nome `serie_geom` que recebe um inteiro `r` e um inteiro não negativo `n`, e devolve a soma dos primeiros `n+1` termos da série geométrica $1 + r + r^2 + r^3 + \ldots + r^n$. Por exemplo,

```
>>> serie_geom(2, 4)
31
>>> serie_geom(100, 0)
1
>>> serie_geom(100, -1)
ValueError: serie_geom: argumento incorrecto
```

10. A *congruência de Zeller* é um algoritmo inventado pelo matemático alemão Julius Christian Zeller (1822–1899) para calcular o dia da semana para qualquer dia do calendário. Para o nosso calendário, o *calendário Gregoriano*, a congruência de Zeller é dada por:

$$h = \left(q + \left\lfloor \frac{13(m+1)}{5} \right\rfloor + K + \left\lfloor \frac{K}{4} \right\rfloor + \left\lfloor \frac{J}{4} \right\rfloor - 2J \right) \mod 7$$

em que h é o dia da semana (0 = Sábado, 1 = Domingo, ...), q é o dia do mês, m é o mês (3 = março, 4 = abril, ..., 14 = fevereiro) – os meses de janeiro e fevereiro são contados comos os meses 13 e 14 do ano anterior, K é o ano do século ($ano \mod 100$), J é o século ($\lfloor ano/100 \rfloor$). Esta expressão utiliza a função matemática, *chão*, denotada por $\lfloor x \rfloor$, a qual converte um número real x no maior número inteiro menor ou igual a x. A definição formal desta função é $\lfloor x \rfloor = \max\{m \in \mathbb{Z} \mid m \leq x\}$. A expressão utiliza também a função módulo, em que $a \bmod b$ representa o resto da divisão de a por b.

Escreva uma função em Python, chamada `dia_da_semana`, que recebe três inteiros correspondentes a um dia, um mês e um ano e que devolve o dia da semana em que calha essa data. A sua função deve utilizar outras funções auxiliares a definir por si. Por exemplo,

```
>>> dia_da_semana(18, 1, 2014)
'sabado'
```