Universidade do Minho
Licenciatura em Física

Tarefa 5

Professores:	Cristiana Neto, Pedro Oliveira, Vitor Alves
Disciplina:	Linguagens para Computação Numérica
Tema:	Dicionários e Estruturas repetitivas
Data:	Março de 2022

1 Introdução

Nas aulas anteriores foram introduzidos diferentes tipos de dados sequenciais em Python: `strings`, `listas` e `tuplos`. Outra estrutura de dados muito útil em Python é o dicionário, cujo tipo é `dict`.

1.1 Dicionários

Ao contrário das sequências que são indexadas por inteiros, os dicionários são indexados por chaves (`keys`), que podem ser de qualquer tipo imutável (como strings e inteiros). Os dicionários são delimitados por chavetas: {}, e contêm uma lista de pares `chave:valor` separada por vírgulas. A figura seguinte mostra um exemplo de um dicionário e manipulação do mesmo.

```
cartao_cidadao = {'rita': 12344566, 'pedro': 22323344}

#adicionar novo elemento ao dicionário caso a chave não exista, se existir modifica o valor:
cartao_cidadao['maria'] = 8765453          #o dicionario passa a ser: {'rita': 12344566, 'pedro': 22323344, 'maria': 8765453}

#ir buscar o valor associado a uma chave (indexação)
cartao_cidadao['rita']                     #retorna 12344566

#eliminar um elemento (par chave:valor) do dicionario através da chave:
del cartao_cidadao['pedro']               #o dicionario passa a ser: {'rita': 12344566, 'maria': 8765453}

#mostrar a lista das chaves do dicionário
list(cartao_cidadao) #retorna ['rita', 'maria']
```

Com a utilização de strings, listas, tuplos e dicionários surge várias vezes a necessidade de executar um conjunto de instruções para cada um dos seus elementos, não sendo suficiente o uso dos métodos apresentados pelo Python. Assim, surgem as estruturas repetitivas que permitem repetir múltiplas vezes o mesmo conjunto de instruções. Resumindo, estas estruturas permitem a execução de instruções repetidas vezes enquanto uma determinada condição for verdadeira ou para cada elemento de uma lista, tuplo, dicionário ou string. Para o primeiro caso, usamos um ciclo `while` (“enquanto”, em inglês), para os outros, usamos um ciclo `for` (“para”, em inglês).

1.2 Ciclo `for`

O comando `for` vem acompanhado de uma variável, a palavra-chave `in` e objetos do tipo listas, tuplos, strings ou dicionários. De seguida temos um exemplo utilizando uma lista.

```
frutas = ["uva", "banana", "cereja"]
for fruta in frutas:
  print('Sou a fruta: ', fruta.upper())
```

```
Sou a fruta:  UVA
Sou a fruta:  BANANA
Sou a fruta:  CEREJA
```

Neste exemplo, a cada iteração (repetição), a variável `fruta` guardará um elemento da lista `frutas`, por ordem, e de acordo com a linha seguinte, vai ser impresso o elemento em letras maiúsculas. Ocorrerão tantas iterações quantos elementos a lista tiver.

Nos ciclos `for` é muito comum a utilização da função `range()` do Python que retorna uma sequência de números a começar no zero (por default) e acabar no número fornecido. Por exemplo, o progama apresentado de seguida imprime cada um dos elementos de uma sequência de 5 números.

```
for n in range(6):
  print('Eu sou o número ', n)
```

```
Eu sou o número  0
Eu sou o número  1
Eu sou o número  2
Eu sou o número  3
Eu sou o número  4
Eu sou o número  5
```

1.3 Ciclo `while`

Por sua vez, o comando `while` vem acompanhado por uma condição que é verificada em cada iteração. O ciclo termina quando a condição for falsa. O exemplo seguinte mostra o mesmo problema apresentado anteriormente mas resolvido agora com um `while`.

```
frutas = ["uva", "banana", "cereja"]
pos = 0
while pos < len(frutas):
  print('Sou a fruta: ', frutas[pos].upper())
  pos = pos + 1
```

```
Sou a fruta:  UVA
Sou a fruta:  BANANA
Sou a fruta:  CEREJA
```

Podemos ver que o código faz exatamente o mesmo, porém foi escrito de forma diferente. Além de ter uma variável a guardar a lista, também criámos a variável `pos`. Esta variável guardará a posição do elemento que queremos imprimir. A posição irá desde 0 até uma posição anterior ao comprimento da lista, que corresponderá à posição do último elemento. Outra coisa que podemos notar é a instrução `pos = pos + 1`. O que estamos a dizer aqui, basicamente, é: guardar na variável `pos` o valor que `pos` tiver nesse momento somado de 1. Inicialmente `pos` guarda o valor 0, após essa instrução, `pos` guardará o valor 1 e assim sucessivamente.

Neste exemplo, o comando `while` é acompanhado pela condição `pos < len(frutas)` que compara a posição nesse momento com o comprimento da lista `frutas`, que é 3. O bloco while será executado sempre que `pos` for menor que 3, ou seja, até 2, que será a posição do último elemento. Dentro do bloco é impresso o elemento da posição nesse momento e, de seguida, a posição é incrementada numa unidade. Uma vez que a condição deixa de ser verdadeira (`pos` deixa de ser menor do que o comprimento da lista), o bloco deixa de ser executado e passam a ser executadas, se houver, as instruções que o seguem.

Nota: o comando `break` pode ser utilizado para sair de um ciclo `for` ou `while` antes de este terminar.

2 Exercícios

Com esta ficha de trabalho pretende-se apresentar alguns problemas exemplificativos da utilização de estruturas de controlo repetitivas e de algumas técnicas específicas, relacionadas com a sua utilização, para a resolução de problemas.

1. Considere o seguinte dicionário:

   ```
   populacao = "Amares": 19853, "Barcelos": 124555, "Braga": 176154, "Cabeceiras de Basto": 17635, "Celorico de Basto": 19767, "Esposende": 35552, "Fafe": 53600, "Guimarães": 162636, "Póvoa de Lanhoso": 24230, "Terras de Bouro": 7506, "Vieira do Minho": 14077, "Vila Nova de Famalicão": 134969, "Vila Verde": 49171, "Vizela": 24477
   ```

   (a) Use um ciclo `for` para mostrar o nome dos concelhos do dicionário `populacao`.

   (b) Use o mesmo ciclo `for` e mostre apenas os concelhos do dicionário `populacao` que têm mais de 50000 habitantes.

   (c) Use um ciclo `for` para percorrer o dicionário `populacao` e mostrar os concelhos que seriam precisos para juntar no mínimo 200 000 habitantes.

   (d) Use um ciclo `for` para percorrer o dicionário populacao e mostrar os três primeiros concelhos que tenham o nome formado por mais do que uma palavra (como "Cabeceiras de Basto", por exemplo.

2. Dada uma sequência de n números, calcular os seguintes resultados:

   (a) o máximo valor da sequência;

   (b) o mínimo valor da sequência;

   (c) os três maiores/menores valores da sequência;

   (d) o somatório dos valores da sequência;

   (e) o número de valores superiores a 10;

   (f) a percentagem de valores superiores a 10;

   (g) a média dos valores da sequência;

   (h) a média dos valores superiores a 10.

3. Calcular os resultados pretendidos no exercício anterior, mas agora utilizando uma sequência cujo tamanho não é conhecido à partida (utilize, por exemplo, uma sentinela para marcar o fim da sequência).

4. Dada a equação de uma recta do tipo $y = m \times x + b$ com m e b constantes conhecidas, calcular os valores de y para valores de x compreendidos entre 0 e 50, de 5 em 5.

5. Calcular os múltiplos de 3, compreendidos no intervalo situado entre 6 e um dado limite superior.

6. Dado um número n, calcular o seu factorial.

7. Dado um número n, calcular os números de Fibonacci que lhe são inferiores. Os números de Fibonacci são representados pela seguinte fórmula de recorrência:

   - $F_{n+2} = F_{n+1} + F_n$ com $F_0 = 0$, $F_1 = 1$ sendo $n \geq 0$

8. Certos núcleos atómicos são instáveis e, com o tempo, decairão através da emissão de radiação. Chamamos a esses núcleos atómicos: radioativos. O processo de decaimento radioativo é completamente aleatório, mas para grandes coleções de átomos, podemos modelar o quanto resta da matéria original depois de um certo tempo. De uma massa original N_0 de um material radioativo, a massa restante após um tempo t (em segundos) é dado pela equação para o decaimento radioativo:

$$N(t) = N_0 \times e^{t/\tau} \tag{1}$$

τ é a chamada 'vida média' do material radioativo e representa a vida útil média de um único núcleo no material radioativo.

   (a) Faça um cliclo while que preencha um dicionário em que as chaves representam os pontos no tempo t e os valores são os valores de $N(t)$ nesses momentos. O cliclo deve ser executado até que a massa restante de material esteja abaixo de 50% da massa original. Comece em $t = 0$ s e use espaços de tempo de 60 s. Considere uma massa $N_0 = 4.5$ kg de carbono-11, que tem uma constante de tempo $\tau = 1760$ s.

   (b) Ao abortar o loop quando metade do material desapareceu, o último elemento no nosso dicionário deve ser a meia-vida do carbono-11, $t_{1/2}$. A meia-vida de um material é simplesmente o tempo que leva para metade do material decair. Teste se isso é verdade imprimindo e comparando a maior chave do seu dicionário com a meia-vida do carbono-11, definido como:

$$t_{\frac{1}{2}} = \tau \times ln(2) \tag{2}$$

   Lembre-se de que, como o seu programa usa espaços de tempo de um minuto, a sua meia-vida medida pode ter um erro de até 60 segundos.

9. A lei da gravidade universal de Newton descreve como a gravidade atua como uma força atrativa entre dois objetos:

$$F = G \times \frac{m_1 \times m_2}{r^2} \tag{3}$$

onde m_1 e m_2 são as massas dos dois objetos que se atraem e r é a distância entre eles. A constante G é a constante de gravitação universal que tem o seguinte valor:

$$G = 6.674 \times 10^{-11} m^3 kg^{-2} s^{-1} \tag{4}$$

Consideremos um objeto com massa $M = 3kg$ que é influenciado por N objetos com massa $m_1, m_2, ..., m_N$. O objeto número i tem massa $m_i = \frac{i}{6} + 2$ kg e distância $r_i = \sqrt{(\frac{i}{4})^2 + 10}$ metros do objeto com massa M.

O número de objetos a interagir é $N = 10$.

Escreva um programa que calcule a força total $\sum_{i=1}^{N} F_i = F_1 + F_2 + ... + F_N$ que afeta o objecto com massa M.