

Curso de Iniciação à Programação em Python para Profissionais de Saúde



Luís Vieira e José Ferrão

Instituto Nacional de Saúde Doutor Ricardo Jorge

17-26 Março 2025

Sessão 4: Funções

- **Conteúdos:**
 - Funções
 - Abrir e guardar ficheiros csv
 - Operadores '*in*' e '*not in*'
 - Exercícios

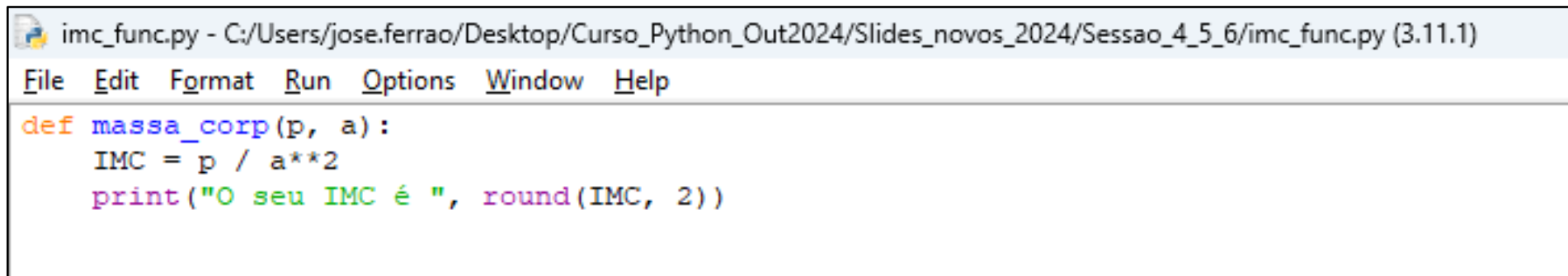
Funções

- O *Python* possui dezenas de funções pré-definidas, como o *print()*, *len()*, *max()* ou *type()*
- No entanto, o *Python* também permite que o utilizador defina as suas próprias funções, como se fossem outras funções *built-in*
- No *Python*, cada **definição de função** tem o seguinte formato:
`def nome_da_função (lista de parâmetros formais):`
 corpo da função
- *def* é uma palavra reservada que informa o *Python* que vai ser definida uma função. O nome da função é apenas um nome escolhido pelo programador para referenciar a função (que tem cor própria). O corpo da função é qualquer sequência de código do *Python*

Funções

- Por exemplo, pode definir-se uma função no editor do *Python* para calcular o índice de massa corporal, usando o peso (p) e a altura (a) como parâmetros formais:

```
def massa_corp (p, a):  
    IMC = p / a**2  
    print ("O seu IMC é", round(IMC, 2))
```



The screenshot shows a window titled "imc_func.py - C:/Users/jose.ferrao/Desktop/Curso_Python_Out2024/Slides_novos_2024/Sessao_4_5_6/imc_func.py (3.11.1)". The menu bar includes File, Edit, Format, Run, Options, Window, and Help. The code editor contains the following Python code:

```
def massa_corp(p, a):  
    IMC = p / a**2  
    print("O seu IMC é ", round(IMC, 2))
```

- Quando a função é usada, os parâmetros formais ficam ligados aos argumentos da **invocação da função**

Funções

- Para invocar a função, pode ser acrescentada uma linha no final com o nome da função e respectivos argumentos (em que o parâmetro *p* liga-se ao objecto 65 e o parâmetro *a* liga-se ao objecto 1.70)

```
imc_func.py - C:/Users/jose.ferrao/Desktop/Curso_Python_Out2024/Slides_novos_2024/Sessao_4_5_6/imc_func.py (3.11.1)
File Edit Format Run Options Window Help
def massa_corp(p, a):
    IMC = p / a**2
    print("O seu IMC é ", round(IMC, 2))

massa_corp(65, 1.70)
|
```

- A execução do código acima produz o seguinte resultado na *Shell* do *Python*:

```
IDLE Shell 3.11.1
File Edit Shell Debug Options Window Help
===== RESTART: C:/Users/jo
se.ferrao/Desktop/Curso_Python_Out2024/Slides_novos_2024/Sessao_4_5_6/imc_func.py =====
O seu IMC é  22.49
>>> |
```

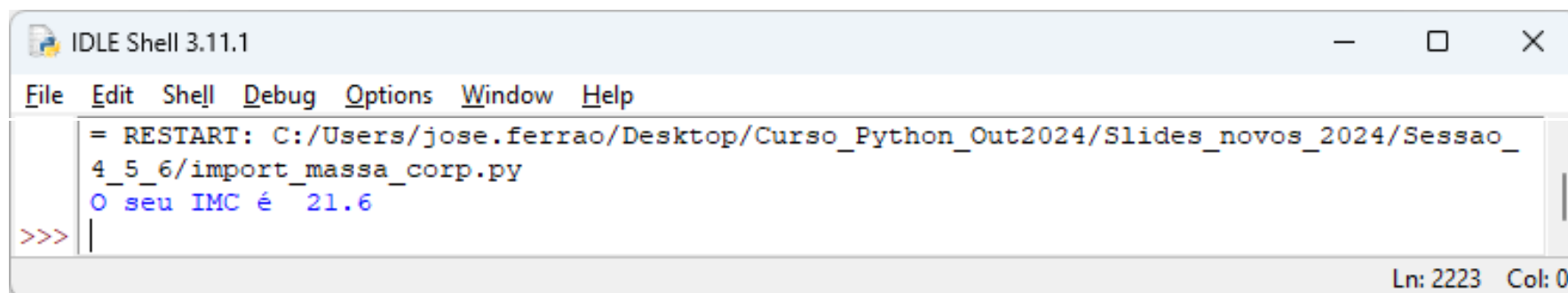
Ln: 2187 Col: 0

Funções

- A função *massa_corp* também pode ser invocada fora do ficheiro onde está definida
- Num novo ficheiro do editor do *Python*, pode criar-se o seguinte código:

```
from imc_func import massa_corp
massa_corp (70, 1.80)
```

- A execução deste código vai invocar a função *massa_corp*, a partir do ficheiro *imc_func.py*, com os respectivos argumentos para o peso e altura, produzindo o seguinte resultado no editor do *Python*:



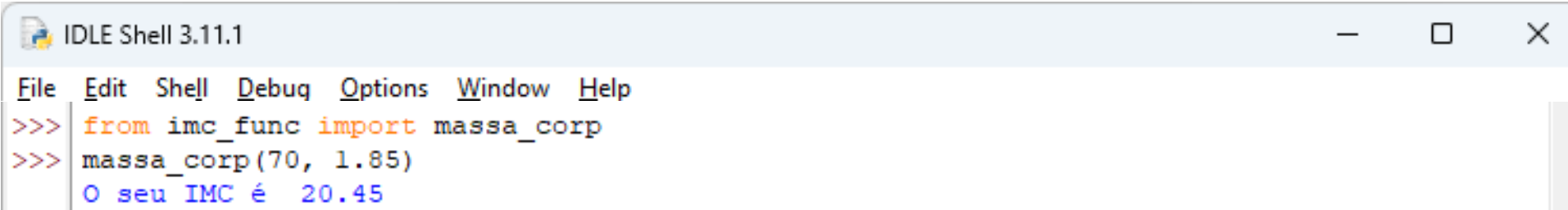
```
IDLE Shell 3.11.1
File Edit Shell Debug Options Window Help
= RESTART: C:/Users/jose.ferrao/Desktop/Curso_Python_Out2024/Slides_novos_2024/Sessao_
4_5_6/import_massa_corp.py
O seu IMC é 21.6
>>>
Ln: 2223 Col: 0
```

Funções

- A função *massa_corp* também pode ser invocada e executada directamente na *Shell* do *Python* usando os seguintes comandos*:

```
>>> from imc_func import massa_corp
```

```
>>> massa_corp(70, 1.85)
```



The screenshot shows a window titled 'IDLE Shell 3.11.1' with a menu bar (File, Edit, Shell, Debug, Options, Window, Help). The shell contains the following text:

```
>>> from imc_func import massa_corp
>>> massa_corp(70, 1.85)
O seu IMC é 20.45
```

* Nota: Os ficheiros *imc_func.py* e o ficheiro *import_massa_corp.py* devem estar localizados no directório de instalação do *Python*. Em alternativa, redefinir directório de trabalho.

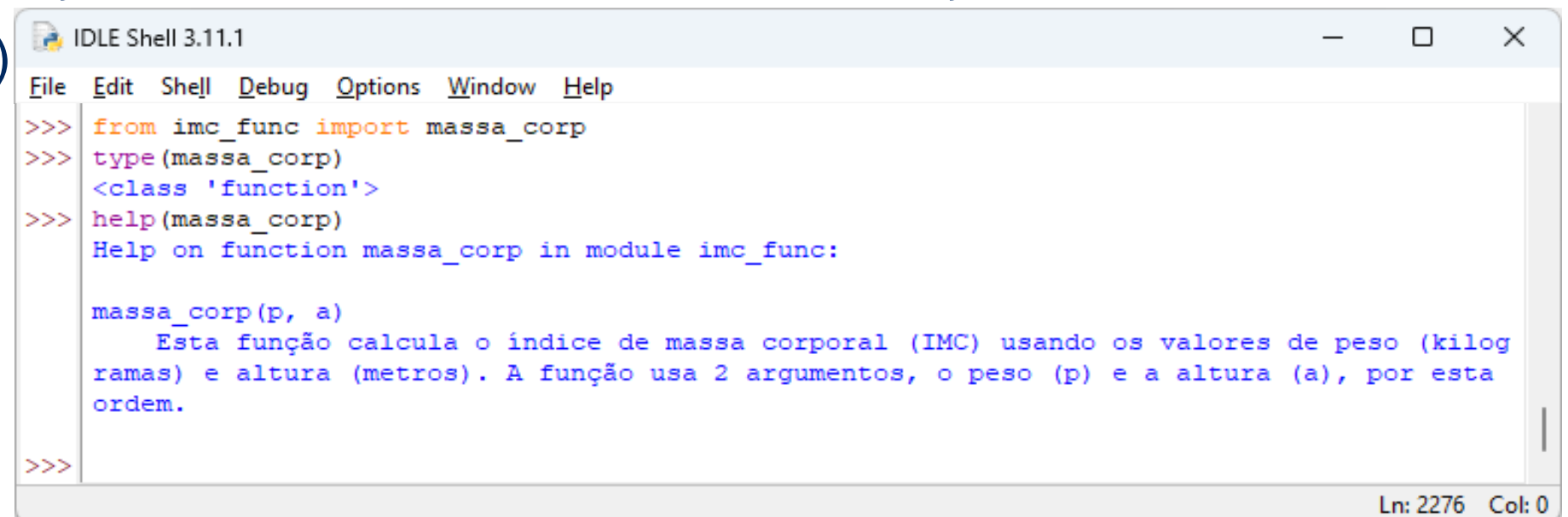
Funções

- Assim como nas funções *built-in* do *Python*, é possível obter informação sobre as funções criadas por um programador. Por exemplo, no caso da função `massa_corp`, podemos correr o comando `type()` para confirmar que se trata de uma função:

```
>>> type(massa_corp)
```

- As funções podem conter um texto explicativo colocado entre aspas triplas abaixo da linha de definição da função (**docstring**). Ao usar o comando `help()`, o utilizador pode visualizar a informação sobre a função sem ter de abrir o ficheiro da função:

```
>>> help(massa_corp)
```



```
IDLE Shell 3.11.1
File Edit Shell Debug Options Window Help
>>> from imc_func import massa_corp
>>> type(massa_corp)
<class 'function'>
>>> help(massa_corp)
Help on function massa_corp in module imc_func:

massa_corp(p, a)
    Esta função calcula o índice de massa corporal (IMC) usando os valores de peso (kilogramas) e altura (metros). A função usa 2 argumentos, o peso (p) e a altura (a), por esta ordem.

>>>
```


Funções

- **Considerações importantes no uso das funções**
 - Os parâmetros formais e argumentos podem ligar-se usando o método posicional (o 1º parâmetro formal liga-se ao 1º argumento, o 2º parâmetro formal liga-se ao 2º argumento, etc.) ou através de argumentos palavra-chave, em que a ligação é feita usando o nome do parâmetro formal. Neste último caso, os parâmetros formais podem ser escritos em qualquer ordem
 - Algumas variáveis podem existir no corpo de uma função e noutra secção do código com o mesmo nome, mas com objectos diferentes. Os parâmetros formais e as variáveis do corpo da função existem apenas no âmbito da definição da função, não afectando outras variáveis existentes fora do âmbito da função. A isto chama-se scoping
 - A invocação da função termina se existir uma declaração **return** no corpo da função

Funções

- Scope: Variável global vs variável local

```
*imc_func.py - C:/Users/jose.ferrao/Desktop/Curso_Python_Out2024/Slides_novos_2024/Sessao_4_5_6/imc_func.py (3.11.1)*
File Edit Format Run Options Window Help
x = 100 # Variável global

def massa_corp(p, a):
    """
    Esta função calcula o índice de massa corporal (IMC) usando os valores de peso (quilogramas) e altura (metros).
    A função usa 2 argumentos, o peso (p) e a altura (a), por esta ordem.
    """
    x = 500 # Variável local
    IMC = p / a**2
    print("O seu IMC é ", round(IMC, 2))
    print("o valor da variável x dentro da função, ou seja, como variável local, é ", x)
massa_corp(65, 1.70)

print("o valor da variável x fora da função, ou seja, como variável global, é ", x)
```

```
IDLE Shell 3.11.1
File Edit Shell Debug Options Window Help
= RESTART: C:/Users/jose.ferrao/Desktop/Curso_Python_Out2024/Slides_novos_2024/Sessao_4_5_6/imc_func.py
O seu IMC é 22.49
o valor da variável x dentro da função, ou seja, como variável local, é 500
o valor da variável x fora da função, ou seja, como variável global, é 100
>>>
```

Funções

- A **especificação** de uma função define um contracto entre o implementador da função e todos aqueles (clientes) que a vão usar nos seus programas. O contracto deverá conter 2 partes:
 - Pressupostos: Descrevem, por exemplo, o conjunto de tipos de objectos aceitáveis para cada um dos parâmetros formais e, por vezes, os valores permitidos para cada parâmetro. Esta informação deve ser colocada na *docstring*
 - Garantias: Descrevem as condições que a função deve satisfazer (por ex., resultados que produz) se tiver sido chamada respeitando os pressupostos

Abrir ficheiros csv

- O formato *csv* (*comma-separated values*) é o mais utilizado para importar e exportar dados de folhas de cálculo e de bases de dados
- O módulo **csv** do *Python* implementa classes para ler e escrever dados em formato tabular (por ex., *Microsoft Excel*)
- O módulo **csv** é importado para o *Python* da seguinte forma:

```
>>> import csv
```

- As linhas do ficheiro *csv* são lidas com a função *csv.reader*, que produz uma **lista de strings** por cada linha do ficheiro de origem. O exemplo seguinte mostra como usar a função *csv.reader* para imprimir na *Shell* cada linha do ficheiro *csv*:

```
data = open("nome_do_ficheiro_de_origem.csv", newline="")
```

```
reader = csv.reader(data)
```

```
for row in reader:
```

```
    print(row)
```

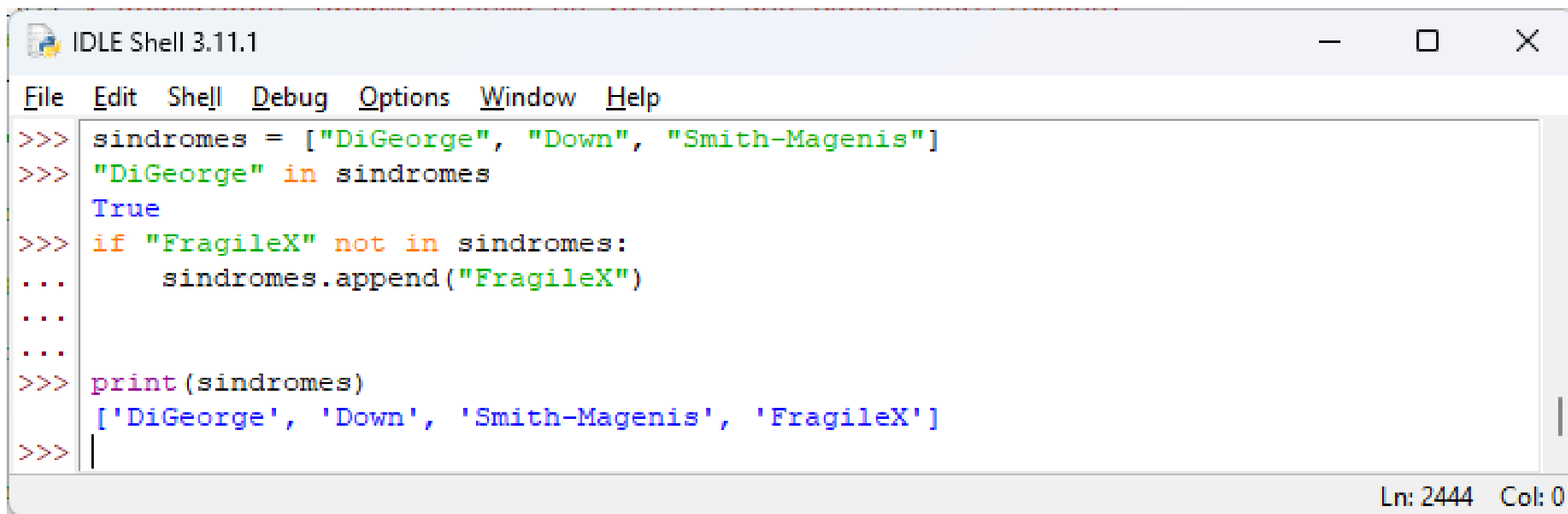
Guardar ficheiros csv

- Os dados do utilizador também podem ser guardados usando o formato `csv` através da função `csv.writer` do módulo **csv**
- A função `csv.writer` converte os dados do utilizador em *strings* delimitadas no ficheiro de destino. O exemplo seguinte mostra a utilização da função `csv.writer` para guardar uma lista de *strings* separadas por espaços, num ficheiro `csv`:

```
import csv
open("nome_do_ficheiro_de_destino.csv", "w", newline="")
writer = csv.writer(data, delimiter = " ")
writer.writerows(["string1", "string2", "string3"...])
data.close()
```

- As funções `csv.reader` e `csv.writer` possuem vários parâmetros de formatação dos dados que podem ser conhecidos usando a função `help()`

Operadores '*in*' e '*not in*'



```

IDLE Shell 3.11.1
File Edit Shell Debug Options Window Help
>>> sindromes = ["DiGeorge", "Down", "Smith-Magenis"]
>>> "DiGeorge" in sindromes
True
>>> if "FragileX" not in sindromes:
...     sindromes.append("FragileX")
...
...
>>> print(sindromes)
['DiGeorge', 'Down', 'Smith-Magenis', 'FragileX']
>>> |
Ln: 2444 Col: 0

```


Exercícios

- O SINAVE é um sistema de vigilância de saúde pública que permite a monitorização do estado de saúde da população ao longo do tempo, nomeadamente a incidência das doenças transmissíveis sujeitas a declaração obrigatória permitindo determinar o risco de transmissão destas doenças. Os dados de 2014-2018 encontram-se no ficheiro (“doencas-de-declaracao-obrigatoria.csv”), mas não é fácil extrair a informação necessária dada a elevada dimensão da tabela.

Construa uma função do *Python* que receba 1 único argumento (nome do ficheiro Excel) e devolva os seguintes dados ao utilizador:

1. N° total de doenças e de casos notificados. Imprima também o nome das doenças.
2. N° de casos de Febre Q notificados em residentes da região do Algarve (NUTS III)
3. N° de casos de Gonorreia notificados em indivíduos do sexo masculino entre 15-24 anos de idade residentes na Área Metropolitana de Lisboa (NUTS III)

Fim da sessão 4

