



# Fundamentos de Programação (T)

Licenciatura em Videojogos  
Ano 1 / Semestre 1



# Sumário

- Ficheiros
  - Json
- Executáveis
- Excepções
  - Erros Clássicos vs. Excepções
  - Tratamento
  - Lançamento
  - Propagação
  - Parametrização
- Asserções



# Ficheiros

# Ficheiros

- Vamos ver como conseguimos tratar ficheiros
- Começamos pela leitura:
  - Para conseguirmos ler o ficheiro, primeiro temos que o abrir

```
file = open("lorem.txt", "rt")
```

Nome do  
ficheiro

Modo de abertura:  
r = read  
t = ficheiro de texto

- Sintaxe:



# Ficheiros

- Ler ficheiros:

- Para abrir o ficheiro:

```
file = open("lorem.txt", "rt")
```

- Depois lemos a primeira linha:

```
first_line = file.readline()
```

- No final, temos sempre que fechar o ficheiro:

```
file.close()
```

# Ficheiros

```
file = open("lorem.txt", "rt")  
  
first_line = file.readline()  
  
print(first_line)  
  
file.close()
```

Porquê este espaço?

Porque nós lemos uma linha (que acaba num `return "\n"`) e o `print` também coloca outro no fim.

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas> & C:/Users/filip/AppData/Local/Programs/Python/Python312/python.exe c:/Users/filip/Aula7_fp.py  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam sed blandit rigula. Etiam semper lorem consectetur iaculis facilisis.  
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
```

# Ficheiros

- Ler ficheiros:

- Primeiro temos que o abrir:

```
file = open("lorem.txt", "rt")
```

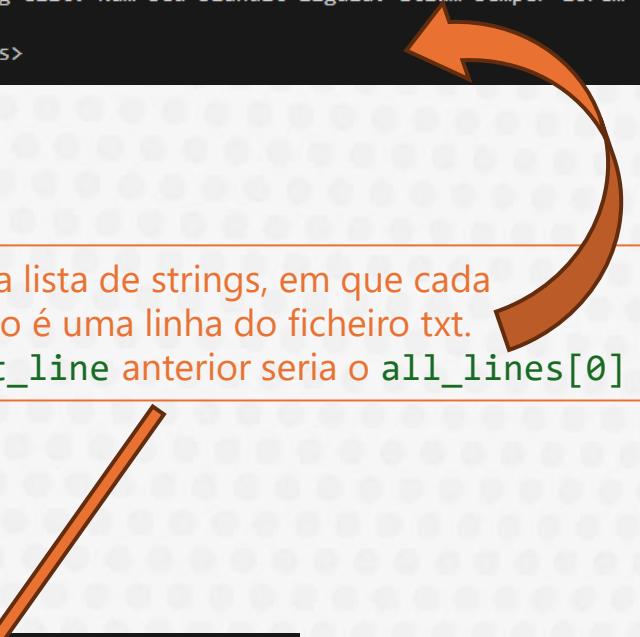
- Para lermos todas as linhas:

```
all_lines = file.readlines()
```

```
[ 'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam sed blandit ligula. Etiam semper lorem consectetur iaculis facilisis. \n', 'Duis magna tortor, mollis nec ultrices eu, porttitor ut libero. Mauris ex eros, pellentesque a odio non, posuere ultrices purus. \n', 'Aliquam dapibus tellus mi, ultricies ullamcorper turpis consequat id. Aliquam erat volutpat. Suspendisse imperdiet iaculis finibus. \n', 'Aliquam suscipit turpis vitae lectus maximus rhoncus. Quisque congue posuere congu . Morbi id condimentum ipsum. \n', 'Vestibulum nisi mi, mattis ut mauris nec, mattis euismod dolor.\n', '\n', 'Sed pellentesque dolor quis arcu rutrum vulputate. Nu lan sit amet lacus quis nisl sagittis finibus. Etiam dictum et est venenatis ullamcorper. \n', 'Ut auctor nunc ac nulla varius mollis. Curabitur vehicula fermentum odales. Ut vel fermentum diam. In hac habitasse platea dictumst.\n', '\n', 'Quisque tellus lorem, pulvinar in malesuada ac, viverra vitae erat. Suspendisse facilisi metus lacinia sagittis fermentum.\n', 'Nunc pellentesque nunc ac sem bibendum efficitur. Vivamus dapibus ligula mauris, vel euismod metus faucibus at. Donec fermentum posuere ex id auctor. \n', 'Nulla vestibulum nisi eget molestie tincidunt. Nullam dictum laoreet consectetur. Pellentesque eget eleifend nunc. \n', 'Aenean tempo , nisi vel elementum elementum, arcu urna auctor mauris, in aliquet ligula dui nec diam. Morbi tempor sodales congue. \n', 'Integer quis metus aliquam lorem egestas egestas. In velit elit, posuere quis iaculis et, malesuada et tellus. \n', 'Vestibulum facilisis elit in nunc sagittis semper quis eget eros.\n', '\n', 'Integer ege consectetur lectus, a lacinia dui. Maecenas blandit volutpat augue, sit amet bibendum nunc faucibus eget. \n', 'Praesent pharetra gravida nisi sed tempor. Donec eu placerat est, quis scelerisque purus. Sed tincidunt sem vel lacus mollis dictum. \n', 'Integer quis arcu mauris. Nam libero libero, pretium ut dictum nec, tempor in diam. Nulla nec urna lectus. \n', 'Duis blandit mi ac lacus pretium, at lobortis erat venenatis. Donec rhoncus neque id magna luctus, sed fermentum eros finibus. \n , 'Sed sem odio, rhoncus quis massa vel, sagittis sagittis lectus.\n', '\n', 'Morbi laoreet odio in aliquam blandit. Curabitur cursus neque cursus purus mollis, non malesuada sem laoreet. Duis ut eros diam. \n', 'Donec eu odio egestas, eleifend libero in, tristique est. Phasellus ultrices velit quam, a sagittis tortor pellentes ue et. \n', 'Maecenas quis tristique nisi. Cras laoreet sit amet diam ac fermentum. In dictum a odio ac varius. Quisque dapibus maximus consectetur. \n', 'Aliquam a ligula ac sem placerat rhoncus.' ]
```

PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>

Cria uma lista de strings, em que cada elemento é uma linha do ficheiro txt.  
O `first_line` anterior seria o `all_lines[0]`



# Ficheiros

```
file = open("lorem.txt", "rt")  
  
all_lines = file.readlines()  
  
for line in all_lines:  
  
    print(line.strip())  
  
file.close()
```

■ Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam sed blandit ligula. Etiam semper lorem consectetur iaculis facilisis. Duis magna tortor, mollis nec ultrices eu, porttitor ut libero. Mauris ex eros, pellentesque a odio non, posuere ultrices purus. Aliquam dapibus tellus mi, ultricies ullamcorper turpis consequat id. Aliquam erat volutpat. Suspendisse imperdiet iaculis finibus. Aliquam suscipit turpis vitae lectus maximus rhoncus. Quisque congue posuere congue. Morbi id condimentum ipsum. Vestibulum nisi mi, mattis ut mauris nec, mattis euismod dolor.

Sed pellentesque dolor quis arcu rutrum vulputate. Nullam sit amet lacus quis nisl sagittis finibus. Etiam dictum et est venenatis ullamcorper. Ut auctor nunc ac nulla varius mollis. Curabitur vehicula fermentum sodales. Ut vel fermentum diam. In hac habitasse platea dictumst.

Quisque tellus lorem, pulvinar in malesuada ac, viverra vitae erat. Suspendisse facilisis metus lacinia sagittis fermentum. Nunc pellentesque nunc ac sem bibendum efficitur. Vivamus dapibus ligula mauris, vel euismod metus faucibus at. Donec fermentum posuere ex id auctor. Nulla vestibulum nisi eget molestie tincidunt. Nullam dictum laoreet consectetur. Pellentesque eget eleifend nunc. Aenean tempor, nisi vel elementum elementum, arcu urna auctor mauris, in aliquet ligula dui nec diam. Morbi tempor sodales congue. Integer quis metus aliquam lorem egestas egestas. In velit elit, posuere quis iaculis et, malesuada et tellus. Vestibulum facilisis elit in nunc sagittis semper quis eget eros.

Integer eget consectetur lectus, a lacinia dui. Maecenas blandit volutpat augue, sit amet bibendum nunc faucibus eget. Praesent pharetra gravida nisi sed tempor. Donec eu placerat est, quis scelerisque purus. Sed tincidunt sem vel lacus mollis dictum. Integer quis arcu mauris. Nam libero libero, pretium ut dictum nec, tempor in diam. Nulla nec urna lectus. Duis blandit mi ac lacus pretium, at lobortis erat venenatis. Donec rhoncus neque id magna luctus, sed fermentum eros finibus. Sed sem odio, rhoncus quis massa vel, sagittis sagittis lectus.

Morbi laoreet odio in aliquam blandit. Curabitur cursus neque cursus purus mollis, non malesuada sem laoreet. Duis ut eros diam. Donec eu odio egestas, eleifend libero in, tristique est. Phasellus ultrices velit quam, a sagittis tortor pellentesque et. Maecenas quis tristique nisi. Cras laoreet sit amet diam ac fermentum. In dictum a odio ac varius. Quisque dapibus maximus consectetur. Aliquam at ligula ac sem placerat rhoncus.

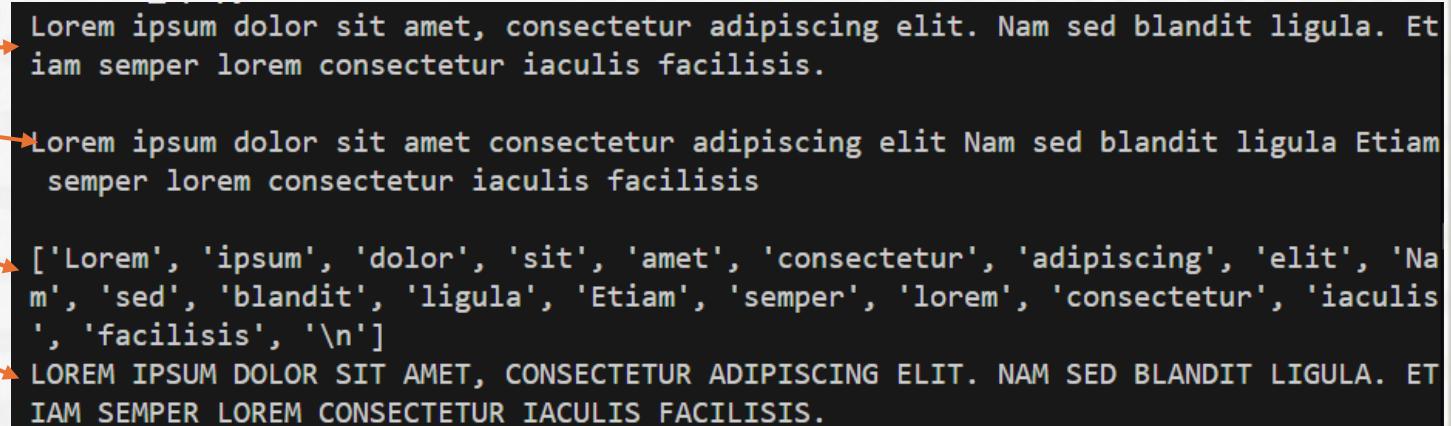
# Ficheiros

- Podemos depois tratar o `readline()` usando operações de strings:

```
file = open("lorem.txt", "rt")

first_line = file.readline()
first_line_no_punctuation = first_line.replace(".", "").replace(", ", "")
first_line_list = first_line_no_punctuation.split(" ")
first_line_caps = first_line.upper()
print(first_line)
print(first_line_no_punctuation)
print(first_line_list)
print(first_line_caps)

file.close()
```



The diagram illustrates the processing of the first line of the 'lorem.txt' file. It shows the original line of text, the line with punctuation removed, the list of words, and the line of text in uppercase. Arrows point from each print statement to its corresponding output.

Original text:  
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam sed blandit ligula. Et iam semper lorem consectetur iaculis facilisis.

Text after punctuation removal:  
Lorem ipsum dolor sit amet consectetur adipiscing elit Nam sed blandit ligula Etiam semper lorem consectetur iaculis facilisis

Text as a list of words:  
['Lorem', 'ipsum', 'dolor', 'sit', 'amet', 'consectetur', 'adipiscing', 'elit', 'Nam', 'sed', 'blandit', 'ligula', 'Etiam', 'semper', 'lorem', 'consectetur', 'iaculis', 'facilisis', '\n']

Text in uppercase:  
LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT. NAM SED BLANDIT LIGULA. ET IAM SEMPER LOREM CONSECTETUR IACULIS FACILISIS.

# Ficheiros

- Modos de abertura de um ficheiro:

- "r" : **read** Lê simplesmente o ficheiro
- "w" : **write** Se o ficheiro existir, será apagado, se não existir será criado
- "a" : **append** Se o ficheiro existir, novo conteúdo será acrescentado no fim, se não existir será criado
- "t" : modo texto
- "b" : modo binário

# Ficheiros

- **Modo texto vs. Modo binário**

- Em sistemas baseados em Posix (Mac, Linux, Solaris, RedHat, etc) não existe diferença nenhuma
- Em sistemas Windows, existe uma questão de conversão de caractéres:
  - Quando se escreve um `\n`, é escrito de facto `\r\n`
  - Quando se lê um `\r\n`, é lido para a string um `\n`
  - Quando se lê um código hexadecimal `0x1a`, o input é considerado que chegou ao fim (legacy do CP/M)
- Resumindo:
  - Modo texto = ler/escrever texto, com codificação (UTF-8) e possíveis traduções de `\n` em Windows.
  - Modo binário = ler/escrever bytes exatamente como estão, usado para ficheiros não textuais.



# Ficheiros

```
file = open("lorem.txt", "wt")

for i in range(0,10):

    file.write("Line " + str(i) + "\n")

file.close()
```

```
lorem.txt
1 Line 0
2 Line 1
3 Line 2
4 Line 3
5 Line 4
6 Line 5
7 Line 6
8 Line 7
9 Line 8
10 Line 9
11 |
```

\*\*\*\*\* OLD VERSION

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam sed blan  
Duis magna tortor, mollis nec ultrices eu, porttitor ut libero. Mauri  
Aliquam dapibus tellus mi, ultricies ullamcorper turpis consequat id.  
Aliquam suscipit turpis vitae lectus maximus rhoncus. Quisque congue  
Vestibulum nisi mi, mattis ut mauris nec, mattis euismod dolor.

Sed pellentesque dolor quis arcu rutrum vulputate. Nullam sit amet la  
Ut auctor nunc ac nulla varius mollis. Curabitur vehicula fermentum s

Quisque tellus lorem, pulvinar in malesuada ac, viverra vitae erat. S  
Nunc pellentesque nunc ac sem bibendum efficitur. Vivamus dapibus lig  
Nulla vestibulum nisi eget molestie tincidunt. Nullam dictum laoreet  
Aenean tempor, nisi vel elementum elementum, arcu urna auctor mauris,  
Integer quis metus aliquam lorem egestas egestas. In velit elit, posu  
Vestibulum facilisis elit in nunc sagittis semper quis eget eros.

Integer eget consectetur lectus, a lacinia dui. Maecenas blandit volu  
Praesent pharetra gravida nisi sed tempor. Donec eu placerat est, qui  
Integer quis arcu mauris. Nam libero libero, pretium ut dictum nec, t  
Duis blandit mi ac lacus pretium, at lobortis erat venenatis. Donec r  
Sed sem odio, rhoncus quis massa vel, sagittis sagittis lectus.

Morbi laoreet odio in aliquam blandit. Curabitur cursus neque cursus  
Donec eu odio egestas, eleifend libero in, tristique est. Phasellus u  
Maecenas quis tristique nisi. Cras laoreet sit amet diam ac fermentum  
Aliquam at ligula ac sem placerat rhoncus.

\*\*\*\*\* USING WT

\*\*\*\*\* NEW VERSION

```
Line 0
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
```

# Ficheiros

```
file = open("lorem.txt", "at")

for i in range(10,20):

    file.write("Line " + str(i) + "\n")

file.close()
```

```
1  Line 0
2  Line 1
3  Line 2
4  Line 3
5  Line 4
6  Line 5
7  Line 6
8  Line 7
9  Line 8
10 Line 9
11 Line 10
12 Line 11
13 Line 12
14 Line 13
15 Line 14
16 Line 15
17 Line 16
18 Line 17
19 Line 18
20 Line 19
```

```
***** NEW VERSION
```

```
Line 0
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
***** USED APPEND
```

```
Line 0
Line 1
Line 2
Line 3
Line 4
Line 5
Line 6
Line 7
Line 8
Line 9
Line 10
Line 11
Line 12
Line 13
Line 14
Line 15
Line 16
Line 17
Line 18
Line 19
```



# JSON

# JSON

- JSON – JavaScript Object Notation
- É um formato aberto que usa texto legível para transferir dados em formato chave-valor (*key-value*)
- A generalidade das linguagens de alto nível têm bibliotecas default para ler e escrever JSON
- O Python tem suporte nativo para JSON, e suporta objectos simples
- Para objectos mais complexos é preferível usar outras bibliotecas

```
{  
    "firstName": "John",  
    "lastName": "Smith",  
    "isAlive": true,  
    "age": 27,  
    "address": {  
        "streetAddress": "21 2nd Street",  
        "city": "New York",  
        "state": "NY",  
        "postalCode": "10021-3100"  
    },  
    "phoneNumbers": [  
        {  
            "type": "home",  
            "number": "212 555-1234"  
        },  
        {  
            "type": "office",  
            "number": "646 555-4567"  
        },  
        {  
            "type": "mobile",  
            "number": "123 456-7890"  
        }  
    "children": [],  
    "spouse": null  
}
```

# JSON

- Primeiro vamos declarar uma estrutura de dados que queremos exportar:

```
data = {  
    "points" : [  
        { "x" : 0, "y" : 1 },  
        { "x" : 1, "y" : 3 },  
        { "x" : 4, "y" : 7 }  
    ],  
    "n_points" : 3  
}
```

**Nós já vimos esta estrutura de dados**

# JSON

- Se escrevemos os dados já obtemos algo semelhante a uma estrutura de um JSON

```
print(data)
```

```
{'points': [{x: 0, y: 1}, {x: 1, y: 3}, {x: 4, y: 7}], 'n_points': 3}
```

**Dicionário em Python e ficheiros JSON seguem a mesma estrutura**

# JSON

- Se quisessemos escrever isto para um ficheiro JSON, podíamos fazer:

```
import json

file = open("data.json", "wt")

json_data = json.dumps(data)
file.write(json_data)

file.close()

{} data.json > ...
1  [{"points": [{"x": 0, "y": 1}, {"x": 1, "y": 3}, {"x": 4, "y": 7}], "n_points": 3}]
```

# JSON

- Podemos tambem ler o JSON e importar os dados para reconstruir o array:

```
import json

file = open("data.json", "rt")
json_data = file.read()
data = json.loads(json_data)
print(data)

file.close()
```

```
• {'points': [{x: 0, y: 1}, {x: 1, y: 3}, {x: 4, y: 7}], 'n_points': 3}
```



# JSON

- Não é possível serializar classes com a biblioteca default
- Serialização é o que se chama a operação de converter estruturas de dados de uma linguagem de programação num formato bom para ficheiros.
- A operação inversa chama-se “desseractialização”
- Há outras bibliotecas para este propósito (por exemplo o marshmallow-dataclass, JSONPickle).
- Para coisas simples, como guardar opções do jogador, ou high-scores, o sistema default serve.
- Resumindo:

**O JSON default no Python só sabe trabalhar com tipos básicos. Se precisarem de guardar classes/objetos, precisam de uma biblioteca extra. Mas para configs de jogo, high-scores ou dados simples, a biblioteca default é mais do que suficiente.**

# JSON - Serialização

- Serializar significa converter uma estrutura de dados num formato linear e “guardável” em JSON
- **Por exemplo:**

```
save = {  
    "player_name" : "Phil",  
    "class" : "rogue",  
    "level" : "77",  
    "map_position" : { "location" : "titan", "x" : 245, "y" : 123, "z" : 45}  
}
```

- Converter isto num JSON seria um processo de serialização

```
{ } save_game.json > ...  
1  [{"player_name": "Phil", "class": "rogue", "level": "77", "map_position": {"location": "titan", "x": 245, "y": 123, "z": 45}}]
```

# JSON - Dessorialização

- Dessorializar significa o processo inverso, converter um ficheiro JSON numa estrutura de dados em Python
- No caso específico do JSON que criámos com save data, poderíamos fazer um “load” quando voltamos ao jogo

```
import json
player_data = {}
file = open("save_game.json", "rt")
json_data = file.read()
player_data = json.loads(json_data)
print("loaded player data:", player_data)
file.close()
```

```
loaded player data: {'player_name': 'Phil', 'class': 'rogue', 'level': '77', 'map_position': {'location': 'titan', 'x': 245, 'y': 123, 'z': 45}}
```



# Executáveis



# Executáveis

- É possível tornar uma aplicação Python num executável “normal” -> um programa
- Um executável otimiza o código, fazendo com que a aplicação corra mais rapidamente consumindo menos recursos
- Criar um executável fará com que o nosso programa fique agarrado a uma plataforma
  - Se criarem um executável para windows, só vai funcionar no windows, e o mesmo para outros sistemas operativos
- Um executável esconde o código, dificultando que seja visível pelos utilizadores
  - O que é sempre importante num jogo

# Executáveis

- Existem várias aplicações para criar executáveis
- Infelizmente, há alguma confusão com versões (Python 2 vs Python 3).
- O PyInstaller é o mais simples de usar, basta instalar:

```
pip install pyinstaller
```

- Depois, só é preciso fazer:

```
python -m PyInstaller --onefile <mainFile.py>
```

- Na pasta dist vai estar um executável com o nosso programa, para a plataforma onde fizemos esta operação.
- Não esquecer de copiar quaisquer ficheiros necessários para a execução (imagens, sons, etc).

```
L MINGW64 /d/projects/fc_2019_exemplos (master)
exe --onefile adventure9.py
taller: 3.5
n: 3.7.3
orm: Windows-10-10.0.17134-SP0
D:\projects\fc_2019_exemplos\adventure9.spec
s not available.
ding PYTHONPATH with paths
\\fc_2019_exemplos', 'D:\\projects\\fc_2019_exemplos'
ing Analysis
ing Analysis because Analysis-00.toc is non existent
alizing module dependency graph...
alizing module graph hooks...
zing base_library.zip ...
ning Analysis Analysis-00.toc
ing Microsoft.Windows.Common-Controls to dependent as
d:\\opt\\python\\python.exe
ning module hooks...
lyzing D:\\projects\\fc_2019_exemplos\\adventure9.py
ding module hooks...
ding module hook "hook-encodings.py"...
ding module hook "hook-pydoc.py"...
ding module hook "hook-xml.py"...
king for ctypes DLLs
lyzing run-time hooks ...
king for dynamic libraries
king for eggs
ng Python library d:\\opt\\python\\python37.dll
nd binding redirects:

nings written to D:\\projects\\fc_2019_exemplos\\build\\a
oh cross-reference written to D:\\projects\\fc_2019_ex
cking PYZ
lding PYZ because PYZ-00.toc is non existent
lding PYZ (ZlibArchive) D:\\projects\\fc_2019_exemplos\\
lding PYZ (ZlibArchive) D:\\projects\\fc_2019_exemplos\\
cking PKG
lding PKG because PKG-00.toc is non existent
lding PKG (CArchive) PKG-00.pkg
lding PKG (CArchive) PKG-00.pkg completed successfully
tloader d:\\opt\\python\\lib\\site-packages\\PyInstaller\\b
cking EXE
lding EXE because EXE-00.toc is non existent
lding EXE from EXE-00.toc
ending archive to EXE D:\\projects\\fc_2019_exemplos\\di
lding EXE from EXE-00.toc completed successfully.
```



# Exercícios

# Exercícios

## 1. Gestão de diálogos

- Usa o ficheiro dialogos.txt no moodle, e faz print de todas as linhas do ficheiro.
- Separa cada linha do ficheiro por nome e fala.
- Constrói um dicionário em que os nomes são as *keys* e as falas os *values*.
- Serializa o dicionário num ficheiro JSON – dialogos.json
- Faz um executável do programa que leia um qualquer dialogos.txt e o transforme num dialogos.json.
  - Esse executável deve pedir ao utilizador um input do ID de um NPC existente e imprimir apenas as falas desse NPC

## 2. Sistema de high scores

- Usa o ficheiro highscore.json do moodle, cria um dicionário com os dados e imprime. Faz uma verificação para confirmar que o ficheiro não está vazio
- Usa o sistema de input para pedir ao utilizador um novo nome e score inteiro. Adiciona esses dados aos highscores
- Ordena a lista de highscores por score descendente (sem usar sort() ou sorted())
- Mantém apenas os 3 melhores resultados
- Serializa novamente a lista de scores como new\_highscores.json
- No final imprime o novo ranking numerado (1º, 2º e 3º)

## 3. Merge de game logs

- Usa os dois ficheiros que estão no moodle – day1.log e day2.log
- Abre os ficheiros e lê todas as linhas de cada um deles
- Cria um merged log com todas as linhas dos dois logs, cada linha deve ter a indicação do dia no formato [DAY1] ou [DAY2]
- Conta o total de linhas de cada um dos dias usando o merged log
- Cria um dicionário estatístico que contenha o número de acções de cada dia e o total de acções de ambos os dias
- Serializa o merged log num log\_stats.json



# Excepções

# Excepções

- Existem duas maneiras normais de gerir erros
  - *Error codes*
  - *Exceptions*
- *Error codes* são mais explícitos:

```
ret = some_function_that_can_fail("param1")
if (ret == False):
    print("Some error!")
    exit()
else:
    print("all good")

Traceback (most recent call last):
  File "c:\Users\filip\OneDrive\Ambiente de Trabalho\Aulas\IC\2526\Aula9_exerc_Python\aula7_fp.py", line 1, in <module>
    ret = some_function_that_can_fail("param1")
          ^^^^^^^^^^^^^^^^^^^^^^
NameError: name 'some_function_that_can_fail' is not defined
```

# Excepções

- Existem duas maneiras normais de gerir erros
  - *Error codes*
  - *Exceptions*
- *Exceptions* são mais estruturadas

```
try:  
    some_function_that_can_fail("param1")  
    print("all good")  
except [Error code]:  
    print("Some error!")  
    exit()
```

```
❸ Traceback (m  
  File "c:\U  
      ret = so  
          ^  
  
NameError: n
```

```
thon313/python.exe "c:/Users/filip/OneDrive/Ambiente de Trabalho/Aulas/IC,  
❸ Some error!
```



# Excepções vs. Erros Clássicos

# Excepções vs. Erros Clássicos

- Excepções permitem separar o código que gera os erros do código que executa normalmente, o que pode fazer o código ficar mais limpo e fácil de ler

```
try:  
    get_value_by_string_index(array, "10")  
except ValueError:  
    print("Failed because a conversion to int failed!")  
    exit()  
except IndexError:  
    print("Failed because the index was out of range!")  
    exit()  
except Exception:  
    print("Some other error!")  
    exit()
```

● Some other error!

# Excepções vs. Erros Clássicos

- Excepções permitem separar o código que gera os erros do código que executa normalmente, o que pode fazer o código ficar mais limpo e fácil de ler

```
array = [1,2,3,4]

try:
    print(array[5])
except ValueError:
    print("Failed because a conversion to int failed!")
    exit()
except IndexError:
    print("Failed because the index was out of range!")
    exit()
except Exception :
    print("Some other error!")
    exit()
```

python3 excepcoes.py

- Failed because the index was out of range!

# Excepções vs. Erros Clássicos

- Excepções permitem separar o código que gera os erros do código que executa normalmente, o que pode fazer o código ficar mais limpo e fácil de ler

```
array = [1,2,3,4]

try:
    array.remove(5)
except ValueError:
    print("Failed because a conversion to int failed!")
    exit()
except IndexError:
    print("Failed because the index was out of range!")
    exit()
except Exception :
    print("Some other error!")
    exit()
```

• Failed because a conversion to int failed!

# Excepções vs. Erros Clássicos

- Erros clássicos evitam o erro antes de acontecer (**if/else**).
- Excepções lidam com o erro depois de ele ocorrer (**try/except**).
- Se forem tratados, o programa continua normalmente.

```
def division_function(a,b):
    if(b == 0):
        print("Can't divide by zero")
    else:
        print(a/b)

for i in range(6, 8):
    for j in range(-1, 2):
        division_function(i,j)
```

```
-6.0
Can't divide by zero
6.0
-7.0
Can't divide by zero
7.0
```

```
def division_function(a,b):
    ret = 0
    try:
        ret = a/b
        print(ret)
    except ZeroDivisionError:
        print("Can't divide by zero")

    return ret
```

```
for i in range(6, 8):
    for j in range(-1, 2):
        division_function(i,j)
```

```
-6.0
Can't divide by zero
6.0
-7.0
Can't divide by zero
7.0
```

VS.

# Excepções vs. Erros Clássicos

- Se não soubermos ou não conseguirmos antecipar o erro específico podemos usar:

```
def division_function(a,b):
    ret = 0
    try:
        ret = a/b
        print(ret)
    except Exception:
        print("something went wrong")
    return ret

for i in range(6,8):
    for j in range(-1,2):
        division_function(i,j)
```

```
● -6.0
something went wrong
6.0
-7.0
something went wrong
7.0
```

# Excepções vs. Erros Clássicos

- As excepções podem (e devem) ser tratadas no ponto em que faz mais sentido, mesmo que seja “longe” da função que originou o erro:

O erro está aqui

As excepções são colocadas fora da função

```
def function_that_runs_a_lot_of_code():
    ...
    get_value_by_string_index(array, "10")
    ...

try:
    function_that_runs_a_lot_of_code()
except ValueError:
    print("Failed because a conversion to int failed!")
    exit()
except IndexError:
    print("Failed because the index was out of range!")
    exit()
except Exception:
    print("Some other error!")
    exit()
```

# Excepções em Python - Tratamento

```
try:  
    # Code to try to run  
except Exception1:  
    # Code to run if previous code throws an exception of  
    # type Exception 1  
except Exception2:  
    # Code to run exception of type Exception 2 is thrown  
except Exception:  
    # Code to run if an exception was thrown, but wasn't  
    # of type Exception1 or Exception2  
finally:  
    # Code to run if any exception was thrown  
  
# Normal code
```



# Excepções em Python - Lançamento

- Por vezes queremos ser nós a lançar (**throw** ou **raise**) uma excepção, a ser tratadas por outra parte do código:

```
raise NameError
```

- Neste caso lançamos um excepção de tipo `NameError`.
- Podemos também definir as nossas próprias excepções.
  - Por exemplo no caso da nossa aventura de texto, o player movimenta-se para uma área não definida:

```
class InvalidPosition(Exception):
```

```
    pass
```

Falaremos mais sobre isto em  
Programação Orientada a Objectos

```
        raise InvalidPosition
```

# Excepções em Python - Lançamento

- Exemplos:

```
def raise_an_exception():
    raise NameError

try:
    raise_an_exception()
except NameError:
    print("There was an error")
```

● There was an error

```
def division_function(a,b):
    if b == 0:
        raise ZeroDivisionError
    else:
        print(a/b)

try:
    for i in range(6,8):
        for j in range(-1,2):
            division_function(i,j)
except ZeroDivisionError:
    print("can not divide by zero")
```

```
ogramas/Python/Python312/python.e
la7_fp.py
● -6.0
can not divide by zero
○ PS C:\Users\filip\Desktop\Video
```

# Excepções em Python - Lançamento

- O **raise** interrompe imediatamente a execução da função onde ocorre, e passa o controlo para o **except** mais próximo

```
def division_function(a,b):
    ret = 0
    try:
        ret = a/b
        print(ret)
    except ZeroDivisionError:
        print("Can't divide by zero")

    return ret

for i in range(6, 8):
    for j in range(-1, 2):
        division_function(i,j)
```

Anteriormente tinhamos isto

O try/except dentro da função detetam o erro clássico, identificam-no e lançam a exceção

```
-6.0
Can't divide by zero
6.0
-7.0
Can't divide by zero
7.0
```

Mas o programa continua a correr

```
def division_function(a,b):
    if b == 0:
        raise ZeroDivisionError
    else:
        print(a/b)

try:
    for i in range(6,8):
        for j in range(-1,2):
            division_function(i,j)
except ZeroDivisionError:
    print("can not divide by zero")
```

Com o raise dentro da função

O try/except fora da função identificam o erro normalmente

A identificação antecipada do código do erro interrompe o programa

```
ograms/Python/Python312/python.e
la7_fp.py
-6.0
can not divide by zero
PS C:\Users\filip\Desktop\Video
```

# Excepções em Python - Lançamento

- Às vezes precisamos de lançar exceções que são específicas do programa que estamos a desenvolver.
  - Python só reconhece por defeito erros genéricos (TypeError, ValueError, IndexError, etc.).
  - Mas alguns erros são próprios da lógica do nosso jogo, o interpretador não os reconhece automaticamente.
- No caso da Aventura de texto:
  - Se o player se mover para uma posição que não existe no mapa, isso é um erro lógico do jogo.
  - Não existe nenhuma exceção que represente esse problema, por isso temos de criar a nossa própria exceção e lançá-la:

```
class InvalidPosition(Exception):  
    pass  
  
    raise InvalidPosition
```



# Excepções em Python - Propagação

- Quando uma excepção é lançada e não é tratada pela função onde ocorreu
  - Ela é automaticamente enviada (propagada) para a função que chamou essa função.
- Este processo continua em cadeia, função acima, depois a seguinte, e assim por diante....
  - Até encontrar um bloco `try/except` capaz de apanhar.
- **Se nenhuma função tratar a exceção, ela chega ao interpretador de Python.**
  - Nesse momento, o programa termina imediatamente e o Python escreve a mensagem de erro no ecrã.
- A propagação é útil porque permite que erros sejam tratados ao nível certo do programa.
  - Não obrigatoriamente no local onde ocorreram.

# Excepções em Python - Propagação

```
def f3():
    print("f3: antes do erro")
    raise ZeroDivisionError
    print("f3: depois do erro")

def f2():
    print("f2: a chamar f3")
    f3()
    print("f2: depois de f3")

def f1():
    print("f1: a chamar f2")
    f2()
    print("f1: depois de f2")

f1()
```

```
f1: a chamar f2
f2: a chamar f3
f3: antes do erro
Traceback (most recent call last):
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 147, in <module>
    f1()
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 144, in f1
    f2()
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 139, in f2
    f3()
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 134, in f3
    raise ZeroDivisionError
ZeroDivisionError
```

# Excepções em Python - Propagação

```
def f3():
    print("f3: antes do erro")
    raise ZeroDivisionError
    print("f3: depois do erro")

def f2():
    print("f2: a chamar f3")
    f3()
    print("f2: depois de f3")

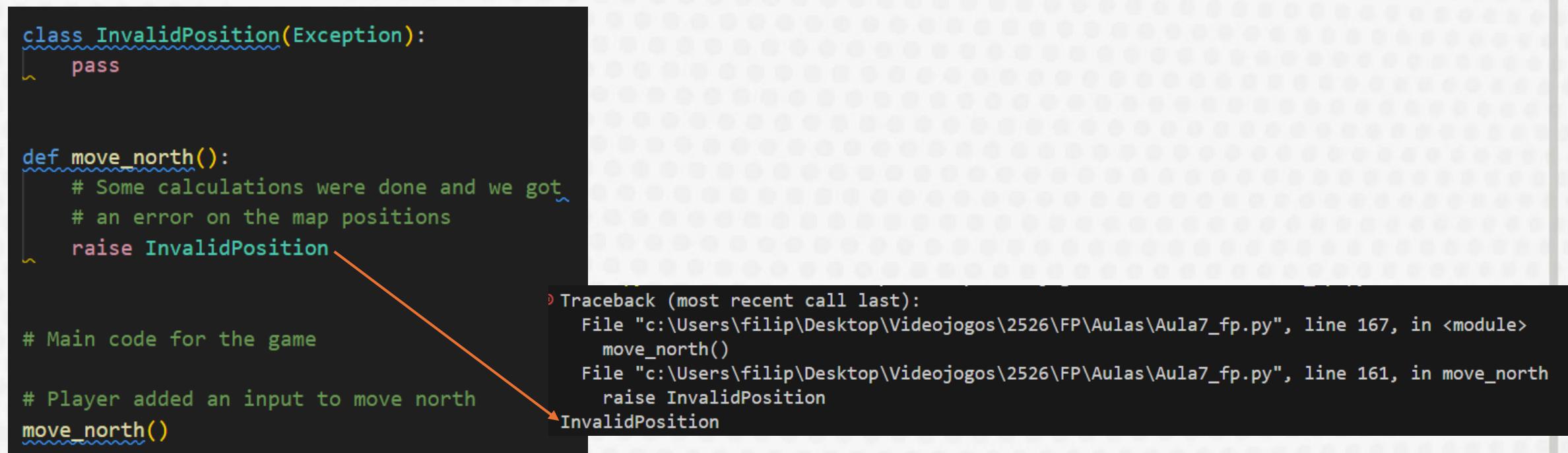
def f1():
    print("f1: a chamar f2")
    f2()
    print("f1: depois de f2")

try:
    f1()
except ZeroDivisionError:
    print("ERRO APANHADO NO TRY EXTERNO!")
```

```
f1: a chamar f2
f2: a chamar f3
f3: antes do erro
ERRO APANHADO NO TRY EXTERNO!
```

# Excepções em Python - Propagação

- Pegando no exemplo da exceção que iríamos criar para o nosso jogo



The diagram illustrates the propagation of an exception. On the left, a code editor shows a Python script with the following content:

```
class InvalidPosition(Exception):
    pass

def move_north():
    # Some calculations were done and we got
    # an error on the map positions
    raise InvalidPosition

# Main code for the game

# Player added an input to move north
move_north()
```

An orange arrow points from the line `move_north()` in the main code block to the `raise InvalidPosition` statement in the `move_north` function definition. To the right, a terminal window displays the resulting traceback:

```
Traceback (most recent call last):
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 167, in <module>
    move_north()
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 161, in move_north
    raise InvalidPosition
InvalidPosition
```

# Excepções em Python - Propagação

- Pegando no exemplo da exceção que iríamos criar para o nosso jogo

```
class InvalidPosition(Exception):
    pass

def move_north():
    # Some calculations were done and we got
    # an error on the map positions
    try:
        raise InvalidPosition
    except InvalidPosition:
        print("That's an invalid position")

# Main code for the game

# Player added an input to move north
move_north()
```

```
PS C:\Users\filip\Desktop\Videojogos\
on312/python.exe c:/Users/filip/Desktop\Videojogos\on312\main.py
That's an invalid position
PS C:\Users\filip\Desktop\Videojogos\on312\
```

# Excepções em Python - Propagação

- Pegando no exemplo da exceção que iríamos criar para o nosso jogo

```
class InvalidPosition(Exception):
    pass

def move_north():
    # Some calculations were done and we got
    # an error on the map positions
    raise InvalidPosition

# Main code for the game

# Player added an input to move north
try:
    move_north()
except InvalidPosition:
    print("Invalid Map Position")
```

```
PS C:\Users\filip\Desktop\Videojogos\2526
on312/python.exe c:/Users/filip/Desktop/V
Invalid Map Position
PS C:\Users\filip\Desktop\Videojogos\2526
```

# Excepções em Python - Parametrização

- Até agora vimos excepções simples, que apenas indicam que aconteceu um erro.
- Em muitos casos, queremos incluir informação adicional dentro da exceção:
  - coordenadas inválidas
  - valores introduzidos pelo jogador
  - o nome do ficheiro em erro
  - o estado do jogo no momento da falha
- Para isso, podemos definir a nossa própria exceção com parâmetros.
- Esta informação pode depois ser lida no bloco `except`, facilitando a resposta ao erro.

# Excepções em Python - Parametrização

- Vamos ver melhor a construção de classes na próxima aula, mas a parametrização de uma exceção poderia se qualquer coisa deste género:

```
class InvalidPosition(Exception):
    def __init__(self, x,y):
        self.x = x
        self.y = y

def move_north(x, y):
    # Some calculations were done and we got
    # an error on the map positions
    raise InvalidPosition(x,y)

# Main code for the game

# Player added an input to move north
try:
    move_north(15,20)
except InvalidPosition as exp:
    print("Invalid Map Position: " + str(exp.x) + "," + str(exp.y))
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas> <
/Aulas/Aula7_fp.py
● Invalid Map Position: 15,20
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
```



# Asserções

# Asserções

- Asserções são ferramentas importantes de *debugging*
- Um asserção permite parar um programa quando um situação que não deveria acontecer, de facto acontece
- Vamos ver um exemplo:

```
import math

def square_root(x):
    return math.sqrt(x)

print(square_root(9))
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas> &
/Aulas/Aula7_fp.py
3.0
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas>
```

# Asserções

- Pegando na mesma função

```
import math

def square_root(x):
    return math.sqrt(x)

print(square_root(-16))
```

```
Traceback (most recent call last):
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 181, in <module>
    print(square_root(-16))
                  ^^^^^^^^^^

  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 179, in square_root
    return math.sqrt(x)
                  ^^^^^^

ValueError: math domain error
```

- O módulo math lida com números reais, onde não existem raízes de número negativos

# Asserções

- Usando uma asserção

```
import math

def square_root(x):
    assert x >= 0
    return math.sqrt(x)

print(square_root(-16))
```

```
❸ Traceback (most recent call last):
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 182, in <module>
    print(square_root(-16))
    ^^^^^^^^^^^^^^^^^^
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 179, in square_root
    assert x >= 0
AssertionError
```

# Asserções

- Podemos usar asserções para escrever mensagens:

```
import math

def square_root(x):
    assert x >= 0, "You shouldn't be using math to find square root of a negative number, you should import cmath instead"
    return math.sqrt(x)

print(square_root(-16))
```

```
PS C:\Users\filip\Desktop\Videojogos\2526\FP\Aulas> & C:/Users/filip/AppData/Local/Programs/Python/Python312/python.exe c:/Aulas/Aula7_fp.py
Traceback (most recent call last):
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 182, in <module>
    print(square_root(-16))
    ^^^^^^^^^^^^^^^^^^
  File "c:\Users\filip\Desktop\Videojogos\2526\FP\Aulas\Aula7_fp.py", line 179, in square_root
    assert x >= 0, "You shouldn't be using math to find square root of a negative number, you should import cmath instead"
AssertionError: You shouldn't be using math to find square root of a negative number, you should import cmath instead
```

# Asserções

```
import cmath

def square_root(x):
    return cmath.sqrt(x)

print(square_root(-16))
```

```
PS C:\Users\filip\Desktop\Videojogos\2
/Aulas/Aula7_fp.py
● 4j
○ PS C:\Users\filip\Desktop\Videojogos\2
```



# Asserções

- Para que servem:
  - Validar suposições
  - Dar erros mais expressivos acerca de situações
  - Alertar acerca de problemas que previmos que podiam dar problemas, mas que não acreditamos que possam acontecer devido à lógica do programa
- Porque não usar ifs ou exceptions?
  - Porque se tivermos a certeza que o programa agora está bem “comportado”, podemos desligar completamente as asserções, aumentando a performance.



# Exercícios

# Exercícios

No moodle, ficheiro `exerc_aula7_fp.py`:

- Cada bloco de código contém erros intencionais que fazem o programa rebentar.
- O objectivo é implementar tratamento de excepções adequado usando:
  - `try / except`
  - `raise`
  - `assert`
- O objectivo não é corrigir os erros apenas tratá-los para que o código fique funcional
- Não alteres a lógica do programa. Apenas torna-o robusto.