ESCOLA
SUPERIOR
DE MEDIA
ARTES
E DESIGN
POLITÉCNICO
DO PORTO

ALP

PROVA DE AVALIAÇÃO
ATIVIDADE LETIVA

Tecnologias e Sistemas de Informação para a Web			
CURSO			
2022/2023	2022/02/08	16h30	2h
ANO LETIVO	DATA	HORA	DURAÇÃO
Algoritmia e Estruturas de Dados			1º ano
UNIDADE CURRICULAR			ANO
Mário Paulo Teixeira Pinto			Normal - AD
DOCENTE			ÉPOCA

Observações:

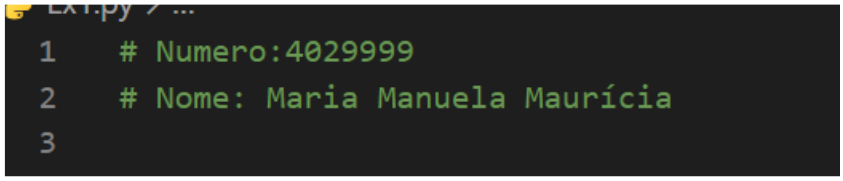
- O Teste é individual e de consulta. Poderá haver lugar a uma prova oral tendo em vista esclarecer dúvidas na resolução dos exercícios, com impacto na avaliação do exame
- Resolva os exercícios recorrendo à linguagem Python

Instruções:

1. Guarde os exercícios resolvidos nessa pasta.
2. No final do teste submeta os exercícios resolvidos no moodle, no objeto *SUBMETER TESTE*.

Parte I (50%)

- Crie um ficheiro com a designação ***EX1_numeroAluno.py*** em que **numeroAluno** é o seu número de aluno na ESMAD.
- No início do ficheiro insira o seu número e nome, sob a forma de comentários, como no exemplo abaixo



Crie um programa que permita gerir os acessos ao parque de estacionamento da ESMAD.

Suponha que existe uma lista de matrículas autorizadas a entrar no parque, definidas numa lista designada:

gessList = ['00-CC-00','01-CC-01','02-CC-02','03-CC-03','04-CC-04', '05-CC-05','06-CC-06','07-CC-07','08-CC-08','09-CC-09']

Pode copiar esta lista (estática) para o seu código, e usá-la a título de exemplo de lista de veículos autorizados
Deve ainda prever uma outra lista, **parkList**, que permita gerir a ocupação do parque de estacionamento (inicialmente vazia).

O seu programa deve simular entradas e saídas de carros no parque, fazendo *inputs* sucessivos de **matrícula** e **tipo de movimento** (E – para entrada, S – para saída). O programa deverá terminar com a introdução da matrícula de '00-00-00'.
Valide o tipo de movimento com uma estrutura de tratamento de exceções.
A cada movimento, o seu programa deve invocar a função:

- **parkValidator**(*matricula, movimento*). Esta função deve devolver um valor booleano que valida se o movimento é válido (True) ou não (False).

  Numa entrada: verifica se a matrícula está autorizada (existe em gessList), e se o veículo não está já no parque (parkList).

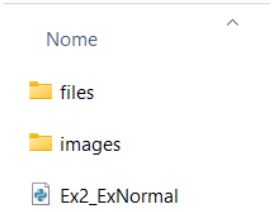
  Numa saída: verifica se a matrícula está no parque de estacionamento(parkList), para dar a sua saída.

  Quando a função parkValidator devolve um valor de False, deve surgir uma mensagem ao utilizador, indicado que o movimento não é possível.

  Quando o movimento é válido deve invocar, em seguida, a função:
  parkManager(*matricula, movimento*). Esta função permite gerir a lista parkList, que contém as matrículas dos carros que estão no parque, num da do momento.
  Numa entrada, deve acrescentar a matrícula à lista; quando se trata de uma saída deve remover a respetiva matrícula da lista.

No final, o seu programa deve indicar quantos carros entraram no parque de estacionamento.

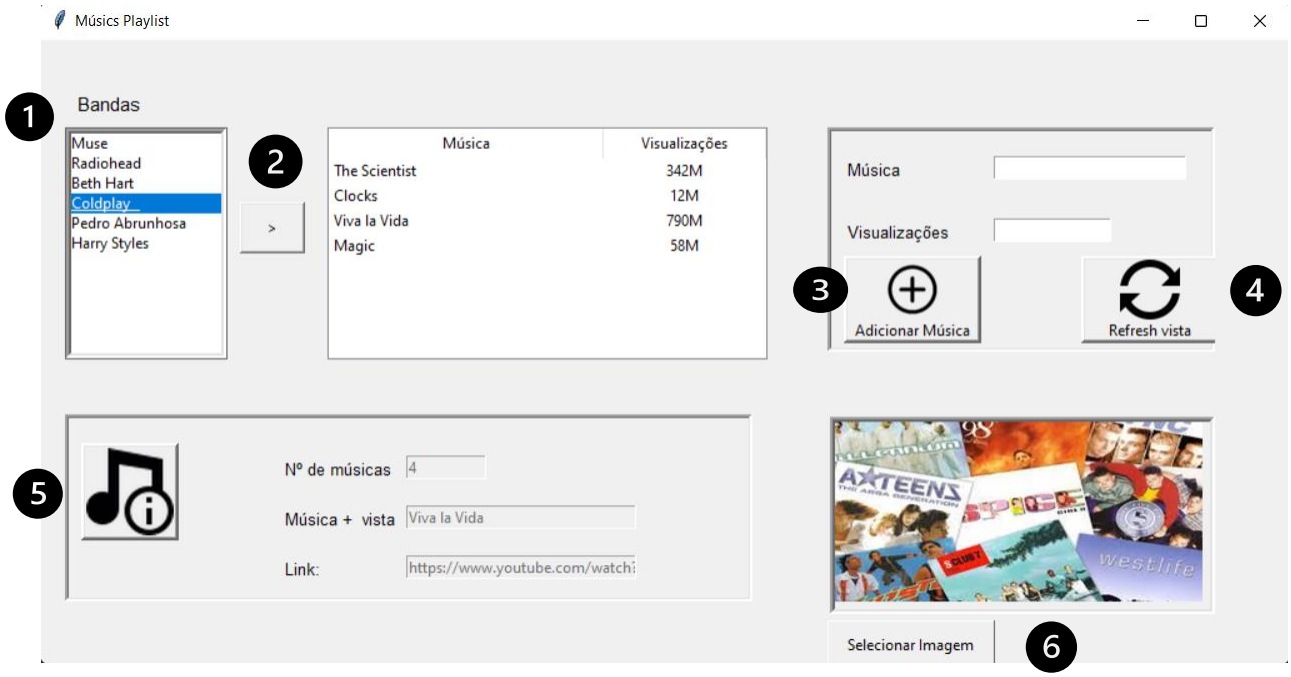
Parte II



- Descarregue do Moodle o ficheiro **ExameNormal**.
- Descompacte a parta. Deve obter a seguir estrutura:
- Altere a designação do ficheiro EX2_ExNormal para ***EX2_numeroAluno*** em que **numeroAluno** é o seu número de aluno na ESMAD.
- No início do ficheiro insira o seu número e nome, sob a forma de comentários, como no exemplo abaixo

```
1   # Numero:4029999
2   # Nome: Maria Manuela Maurícia
3
```

Pretende-se desenvolver uma App com uma *interface* semelhante à abaixo apresentada. A generalidade da interface encontra-se já implementada no ficheiro .py disponibilizado.



1. No arranque da aplicação deve invocar uma função ***lerBandas*()**, com o objetivo de ler o conteúdo do ficheiro ***bandas.txt*** e o renderizar na listBox.

2. Ao clicar no Button, deve invocar a função **lerMusicas()** para ler o conteúdo do ficheiro ***musicas.txt*** e renderizar na treeView as músicas da banda selecionada e o nº de visualizações de cada uma delas.
   O ficheiro ***musicas.txt*** tem a seguinte estrutura: ***banda;musica;nºvisualizações;linkyoutube***

3. O Button deve incluir a imagem adicionar.png, como na imagem acima. Ao clicar neste Button deve invocar a função **adicionaMusica()** para adicionar uma nova linha ao ficheiro ***musicas.txt*** com :
   banda;musica;nºvisualizações; Neste caso, em vez do link para o Youtube inclua um espaço.

4. O Button deve incluir a imagem refresh.png, como na imagem acima. Ao clicar no Button deve invocar a função **refresh()** para atualizar a Treeview com a música que acabou de adicionar ao ficheiro.

5. O Button deve incluir a imagem btn+.png, como se apresenta acima. Ao clicar neste Button deve invocar a função **maisInfo()** que deve mostrar + nº e músicas da banda renderizada na TreeView, a música mais vista e o respetivo link.
   NOTAS:

- o nº de visualizações termina sempre com o **M** (milhões) ou **m** (milhares) que deve ter em consideração para determinar a música mais vista.
- O utilizador não deve poder clicar ou editar as Entrys relativas a este painel.

6. Ao clicar neste Button deve invocar a função **SelecionarImg()** que permita abrir uma FileDialog para selecionar uma imagem (.png ou .gif) para renderizar no componente Canvas.

Boa Sorte 😊