Departamento de Ciências e Tecnologias da Informação	Programação Ano letivo 2020/2021 1º Semestre **1ª Época**

Duração da prova: 2h
Seja organizado na elaboração das respostas, pois respostas ilegíveis não serão classificadas
É estritamente proibido manipular quaisquer aparelhos eletrónicos (telemóveis, tablets, etc).
Só é permitido entregar ou desistir após a primeira hora.
No caso de faltar espaço para a resposta use o espaço extra das últimas páginas (d)

Número: _____ Nome: _____ Turma: _____

Parte I

Para cada uma das seguintes perguntas classifique com V(erdadeiro) ou F(also) cada uma das alíneas. Coloque a sua classificação imediatamente antes de cada alínea. Exemplo: V (a) ... F (b) ... F (c) ....

1. (1v) Considere a seguinte definição e classifique com V/F cada uma das afirmações.

```
score = {"Mariana":27, "Pedro":21, "Rita":18}
```

   (a) trata-se de um tuplo
   (b) trata-se de um objeto imutável
   (c) a seguinte seguinte instrução imprime o valor 18:

```
print(score[-1])
```

   (d) a seguinte instrução adiciona um novo elemento à variável *score*:

```
score["Rui"] = score["Rita"]
```

   (e) a seguinte instrução altera o valor de um elemento da variável *score*:

```
score["Pedro"] = 99.2
```

2. (1v) Considere a seguinte definição e classifique com V/F cada uma das afirmações.

```
def verifica(v):
   for i in v:
      if i < 0:
         return False
   return True
```

   (a) trata-se de um procedimento
   (b) se no parâmetro *v* for recebido um vetor com zero elementos (for vazio), devolve *False*
   (c) se todos os elementos de *v* forem zero, devolve *True*
   (d) se algum dos elementos de *v* for negativo, devolve *False*
   (e) se no parâmetro *v* for recebido uma lista ou um tuplo, origina um erro

3. (1v) Dada a seguinte classe, classifique com *V* as instruções válidas e com *F* as que dão erro.

```
class Cebola:
    def __init__(self, a, b = 0):
        self.__a = a
        self.b = b
```

(a)
```
a1 = Cebola(20, 10)
```

(b)
```
a2 = Cebola(1.2)
```

(c)
```
a1 = Cebola()
print(a1.a)
```

(d)
```
a1 = Cebola(20, 10)
print(a1.a)
```

(e)
```
a1 = Cebola(10, 20)
a1.b = 0.1
```

4. (1v) Considere a seguinte definição e classifique com V/F cada uma das afirmações.

```
scores = [2, 3, 5, 7]
```

(a) trata-se de uma lista com 4 elementos

(b) a seguinte instrução altera o valor do último elemento:

```
scores[-1] = 8
```

(c) a seguinte instrução adiciona um novo elemento à variável *scores*:

```
scores[5] = 9
```

(d) a seguinte instrução origina uma exceção:

```
4 in scores
```

(e) a seguinte instrução adiciona um novo elemento à variável *scores*:

```
scores.append("super")
```

5. (1v) Considere a seguinte definição e classifique com V/F cada uma das afirmações.

```
scores = (2, 3, 5, 7)
```

(a) trata-se de um tuplo

(b) trata-se de um objeto imutável

(c) a seguinte instrução altera o valor do último elemento:

```
scores[-1] = 8
```

(d) a seguinte instrução escreve o valor **5** no ecrã:

```
print(scores[-2])
```

(e) a seguinte instrução escreve o valor **7** no ecrã:

```
print(scores[4])
```

6. (1v) Classifique com V/F cada uma das afirmações.

   (a) qualquer uma das seguintes instruções cria um vector com 5 elementos:

   ```
   a1 = numpy.ones(5)
   a2 = numpy.zeros(5)
   a3 = numpy.array([9.1, -2, 3, 40, 5555])
   ```

   (b) a seguinte sequência de instruções produz o resultado [4 4 4]:

   ```
   a1 = numpy.array([1, 2, 3])
   a2 = numpy.array([3, 2, 1])
   print(a1+a2)
   ```

   (c) a seguinte sequência de instruções produz o resultado (3, 2):

   ```
   a = numpy.array([[1, 2], [3, 4], [5, 6]])
   print(a.shape)
   ```

   (d) a seguinte sequência de instruções produz o resultado 2:

   ```
   a = numpy.array([[1, 2], [3, 4], [5, 6]])
   print(a[1,2])
   ```

   (e) a seguinte sequência de instruções produz o resultado [3 4]:

   ```
   a = numpy.array([[1, 2], [3, 4], [5, 6]])
   print(a[1])
   ```

7. (1v) Classifique com V as funções que permitem calcular a soma de todos os elementos de uma lista e com F as que não o fazem

   (a)
   ```
   def sum(lst):
       t = 0
       for e in lst:
           t = t + e
       return t
   ```

   (b)
   ```
   def sum(lst):
       return lst[0] + sum(lst[1:])
   ```

   (c)
   ```
   def sum(lst):
       if len(lst) == 0:
           return 0
       return lst[0] + sum(lst[1:])
   ```

   (d)
   ```
   def sum(lst):
       t = 0
       i = 0
       while i < len(lst):
           i = i + 1
           t = t + lst[i]
       return t
   ```

   (e)
   ```
   def sum(lst):
       t = 0
       i = 0
       while i < len(lst):
           t = t + lst[i]
           i = i + 1
       return t
   ```

8. (1v) Classifique com V/F cada uma das afirmações.

   (a) a seguinte sequência de instruções escreve o valor 4 no ecrã:

   ```
   a = ["a", "b", ["Maria", 1234], "c"]
   print( len(a) )
   ```

   (b) a seguinte sequência de instruções origina um erro:

   ```
   a = ("a", "b", ["Maria", 1234], "c")
   print( len(a) )
   ```

   (c) a seguinte sequência de instruções origina uma exceção, que termina execução do programa:

   ```
   def say(x):
       assert( isinstance(x, str) )
       print(x)

   say(10)
   ```

   (d) a seguinte sequência de instruções faz terminar execução do programa:

   ```
   def divide(x, y):
       try:
           return x / y
       except:
           return 0

   res = divide(1, 0)
   ```

   (e) a seguinte sequência de instruções atribui o valor 0 à variável *res*:

   ```
   def divide(x, y):
       try:
           return x / y
       except:
           return 0

   res = divide(1, 0)
   ```

Parte II

1. (2v) Desenvolva uma função que recebe duas listas e devolve verdadeiro se se verificarem as seguintes condições: a) as listas têm o mesmo número de elementos; e b) a soma dos elementos maiores do que 0 (zero) de cada uma delas é igual nas duas listas. Devolve falso, caso contrário.

```
def similar(x1, x2):
```

2. (2v) Desenvolva uma função que dada uma matriz quadrada, representada por um objeto do tipo *numpy.ndarray*, devolve a diferença entre a soma dos elementos da diagonal principal e a soma dos restantes elementos.

   Por exemplo, se a função for aplicada à matriz seguinte devolve 1, pois a soma dos elementos da diagonal principal é 6, enquanto que a soma dos restantes elementos é 5:

```
[[3 2 0]
 [0 2 0]
 [2 1 1]]
```

```
def diagonal_difference(m):
```

Parte III

Considere uma loja de venda de automóveis usados em que o seu stock é constituído por um conjunto de viaturas, em que a cada uma das viaturas tem as seguintes propriedades: matrícula, descrição, preço de custo e preço de venda.

1. (2v) Defina uma classe para representar o conceito de viatura. Esta classe deve dispor de um construtor com os seguintes parâmetros: matrícula, descrição, preço de custo e preço de venda. Deve ter em conta que a matrícula não pode ser alterada posteriormente.

   - Deve ser possível consultar cada uma das suas propriedades, bem como o **lucro** (propriedade adicional) que será obtido com a venda desta viatura (preço de venda - preço de custo).
   - Adicione um método que permita transformar uma viatura (*v*) numa cadeia de caráteres, por forma a que possa visualizar todas as suas propriedades ao usar a instrução `print(v)`.

```
class Viatura:
```

2. (6v) Defina uma classe que representa o conceito de loja. Um objeto deste tipo deve guardar informação sobre o stock de viaturas. O construtor desta classe coloca a estrutura que representa o stock da loja com um valor vazio. Deve também definir os métodos necessários para:

   (a) [`existe`] verificar se uma viatura está presente no stock, dada a sua matrícula;

   (b) [`adicionar`] adicionar uma viatura. O método recebe as 4 propriedades da viatura como parâmetros. Tenha em atenção que não é permitido adicionar uma viatura com a mesma matrícula de uma já existente;

   (c) [`lucro_total`] devolver o lucro total das viaturas em stock;

   (d) [`lucro_medio`] devolver o lucro médio das viaturas em stock. Note que se o número de viaturas em stock for zero, o lucro médio será também zero;

   (e) [`filtrar_por_preco`] dado um valor mínimo e máximo, imprimir todas as viaturas com preço de venda dentro desse intervalo;

   (f) [`viatura_mais_cara`] devolver a viatura mais cara (mesmo que existam mais do que uma, devolve apenas uma). Se não existirem viaturas em stock devolve *None*.

Espaço extra

Boa Sorte