iscte TECNOLOGIAS E ARQUITETURA Departamento de Ciências e Tecnologias da Informação	Programação Ano letivo 2021/2022 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. (a)

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. Classifique com V/F as funções que permitem calcular a soma de todos os números naturais pares até *n*, inclusive.

```
def f1(n):
    a = 0
    b = 0
    while b != n + 1:
        if b%2 == 0:
            a = a + b
        b = b + 1
    return a
```

```
def f2(n):
    a = 0
    b = 0
    while b != n:
        if b%2 == 0:
            a = a + b
        b = b + 1
    return a
```

```
def f3(n):
    a = 0
    b = 0
    while b < n + 1:
        a = a + b
        b = b + 2
    return a
```

```
def f4(n):
    a = 0
    b = 0
    while b <= n:
        a = a + b
        b = b + 2
    return a
```

```
def f5(n):
    a = 0
    b = 0
    while b != n + 1:
        a = a + b
        b = b + 2
    return a
```

(a) `f1()`

(b) `f2()`

(c) `f3()`

(d) `f4()`

(e) `f5()`

2. (1v) Considere a classe *Rational* abaixo definida. Sabendo que produto de dois números racionais é um número racional, em que o numerador é a multiplicação dos numeradores e o denominador é a multiplicação dos denominadores, indique com V/F quais dos métodos abaixo implementam o produto de dois racionais.

```
class Rational:
    def __init__(self, n, d = 1):
        self.__n = n
        self.__d = d
```

(a)
```
def product(self, r):
    return Rational(self.__n*r.__n, self.__d*r.__d)
```

(b)
```
def product(self, r):
    return Rational(__n*r.__n, __d*r.__d)
```

(c)
```
def product(self, r):
    return self.__init__(self.__n*r.__n, self.__d*r.__d)
```

(d)
```
def product(self, r):
    n = self.__n*r.__n
    d = self.__d*r.__d
    return Rational(n, d)
```

(e)
```
def product(self, r):
    return Rational(self.__n*r.self.__n, self.__d*r.self.__d)
```

3. (1v) Considerando a classe *Circle* abaixo definida, indique se as instruções são ou não válidas (V/F).

```
class Circle:
    def __init__(self, radius):
        self.__radius = radius

    @property
    def radius(self):
        return self.__radius
```

(a)
```
c = Circle(2.0)
```

(b)
```
Circle c(2.0)
```

(c) assuma que a variável c é do tipo *Circle* e que a instrução abaixo está fora do contexto da classe *Circle*:

```
print(c.radius)
```

(d) assuma que a variável c é do tipo *Circle* e que a instrução abaixo está fora do contexto da classe *Circle*:

```
print(c.__radius)
```

(e) assuma que a variável c é do tipo *Circle* e que a instrução abaixo está fora do contexto da classe *Circle*:

```
c.radius = 3.0
```

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

```
def all_odd(v):
    for e in v:
        if e%2 == 0:
            return False
        else:
            return True
```

   (a) a função analisa sempre todos os elementos do vetor *v*

   (b) se o primeiro elemento de *v* for par (even), a função devolve *False*

   (c) se todos os elementos de *v* forem ímpares (odd), a função devolve *True*

   (d) se no parâmetro *v* for recebido uma lista ou um tuplo, origina um erro

   (e) se o primeiro elemento de *v* for ímpar (odd) e o segundo for par (even) a função devolve *False*

5. (1v) Classifique com V/F cada uma das afirmações.

   (a) a seguinte sequência de instruções escreve ***True, False*** no ecrã:

```
class Circle:
    def __init__(self, radius):
        self.__radius = radius

a = Circle(3)
b = Circle(3)
print(f'{a == b}, {a is b}')
```

   (b) a seguinte sequência de instruções escreve ***True, True*** no ecrã:

```
u = [1, 2, 3, 4, 5]
v = u
print(f'{u == v}, {u is v}')
```

   (c) a seguinte sequência de instruções escreve o valor 3 no ecrã:

```
u = [1, 2, 3, 4, 5]
v = u
v[2] = 0
print(u[2])
```

   (d) a seguinte sequência de instruções escreve o valor 13 no ecrã:

```
def inc(n):
    n += 1

n = 12
inc(n)
print(n)
```

   (e) a seguinte sequência de instruções escreve [2, 3, 4, 5, 6] no ecrã:

```
def inc(v):
    for i in range(len(v)):
        v[i] += 1

v = [1, 2, 3, 4, 5]
inc(v)
print(v)
```

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

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

   ```
   u = numpy.zeros(5)
   v = numpy.array([0, 0, 0, 0, 0])
   ```

   (b) a seguinte sequência de instruções produz o resultado [1 2 3 3 2 1]:

   ```
   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.size)
   ```

   (d) a seguinte sequência de instruções produz o resultado 6:

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

   (e) a seguinte sequência de instruções produz o resultado [2 4 6]:

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

7. (1v) Classifique com V/F cada uma das afirmações.

```
v1 = ('ar', 'terra', 'água', 'fogo')
v2 = ['ar', 'terra', 'água', 'fogo']
```

   (a) v1 é um tuplo e v2 é uma lista

   (b) o valor da seguinte expressão é `True`

   ```
   v1[1] in v2
   ```

   (c) após executar a seguinte instrução, o conteúdo de v1 fica `('ar', 'água', 'água', 'fogo')`

   ```
   v1[1] = v1[2]
   ```

   (d) após executar a seguinte sequência de instruções, o conteúdo de v2 fica `['ar', 'água', 'terra', 'fogo']`

   ```
   v2[1] = v2[2]
   v2[2] = v2[1]
   ```

   (e) após executar a seguinte instrução, o conteúdo de v2 fica `['ar', 'água', 'água', 'fogo']`

   ```
   v2[1] = v2[2]
   ```

8. (1v) Considerando os seguintes procedimentos, classifique com V/F cada uma das afirmações.

```
def uga():
    try:
        f = open('x', 'r')
    except:
        print('ai!')
        return
    print('ya!')
```

```
def buga():
    try:
        f = open('x', 'w')
    except:
        print('ai!')
        return
    print('ya!')
```

(a) `uga()` escreve "ai!" quando o ficheiro 'x' não existe

(b) `buga()` escreve "ai!" quando o ficheiro 'x' não existe

(c) `uga()` escreve "ya!" quando o ficheiro 'x' existe

(d) `buga()` escreve "ya!" quando o ficheiro 'x' existe mas apenas pode ser lido

(e) `uga()` e `buga()` escrevem "ya!" se o ficheiro 'x' existe e puder ser escrito

Parte II

1. (2v) Escreva uma função que recebe uma matriz de números reais, representada por um objeto do tipo *numpy.ndarray*, e devolve o número de elementos cuja parte decimal é zero. Por exemplo, se a função for aplicada à matriz seguinte devolve 3, correspondendo aos valores *1.0*, *0.0* e *8.0*.

```
[[ 1.1  0.5  1.0  1.4]
 [ 1.2  0.0  3.8  6.9]
 [ 1.7  9.7  0.8  8.0]]
```

```
def conta_numeros_redondos(m):
```

2. (2v) Escreva uma função que recebe um dicionário com o *rating* de vários jogos (a chave é o nome do jogo e o valor é o *rating* desse jogo) e devolve uma lista com os nomes dos jogos cujo *rating* está acima da média dos *ratings*. Por exemplo, se *ratings* = `{"Minecraft":4.9, "Pokemon Go":4.1, "Hay Day":4.2, "Call of Duty":4.5}`, a função devolverá: `[“Minecraft”, “Call of Duty”]`.

```
def melhores_jogos(ratings):
```

Parte III

1. (2v) Defina uma classe *Evento* que permite representar um evento em cartaz. Um evento é caracterizado por um *nome* (cadeia de carateres), uma *data* (cadeia de carateres), um *local* (cadeia de carateres), número *total de bilhetes* e o número de *bilhetes vendidos* (assume sempre o valor 0 quando o objeto é criado). Após a criação de um objeto deste tipo, não deve ser possível alterar o *nome* e a *data*, mas os restantes atributos podem ser alterados. Deverá ainda ser possível:

   - consultar qualquer um dos atributos;
   - comparar adequadamente dois objetos deste tipo, considerando que dois *eventos* são iguais se tiverem o mesmo nome e a mesma data;
   - mostrar os detalhes do evento no ecrã usando a instrução `print()`.

```
class Evento:
```

2. (6v) Defina a classe *Cartaz*, que guarda informação sobre eventos, sendo um evento caracterizado pela informação acima indicada. Cada objeto do tipo *Cartaz* terá uma lista de eventos, que deverá ser vazia quando o objeto é criado. Implemente os seguintes métodos. Deve usar a classe da pergunta anterior, assumindo que se encontra definida mesmo que não a tenha conseguido implementar:

   (a) `existe(nome, data)`: caso exista um evento com o nome e data indicados devolve *verdadeiro*. Caso contrário devolve *falso*;

   (b) `adiciona(nome, data, local, n_bilhetes)`: adiciona um novo evento, garantindo que não há dois eventos iguais (mesmo nome e mesma data). Se já existir outro evento igual deve lançar uma exceção;

   (c) `eventos_por_local(expressao)`: mostra no ecrã uma listagem dos eventos, cujo *local* contém a expressão indicada. Exemplo de eventos para a expressão "Lisboa":

   ```
   4 AMIGOS ; 25/01/2022 21:00; Aula Magna, Lisboa; 120; 112
   Evanescence; 18/04/2022 21:00; Altice Arena, Lisboa; 600; 2
   ```

   (d) `bilhetes_disponiveis(nome, data, qtd)`: Devolve *verdadeiro* se o número de bilhetes disponíveis é suficiente para cobrir a quantidade (qtd) em causa e *falso* caso contrário;

   (e) `vende_bilhetes(nome, data, qtd)`: atualiza o número de bilhetes vendidos para o evento em causa, adicionando *qtd* ao número de bilhetes vendidos. Se o número de bilhetes disponíveis for inferior a *qtd* deverá lançar uma exceção.

   (f) `apaga_evento(nome, data)`: apaga um evento dado o seu nome e a sua data.

Espaço extra

Boa Sorte