TÉCNICO LISBOA

Fundamentos da Programação
Ano letivo 2022-23
Primeiro Projeto
30 de Setembro de 2022

P22

Neste primeiro projeto de Fundamentos da Programação os alunos irão desenvolver as funções que permitam resolver três tarefas independentes: 1) Justificação de textos; 2) Cálculo de mandatos pelo método de Hondt; e 3) Solução de sistemas de equações.

1 Justificação de textos

1.1 Descrição do problema

A primeira tarefa consiste em justificar um texto utilizando uma largura de coluna configurável. Isto é, dado um texto qualquer representado por uma cadeia de carateres não vazia e um inteiro representando a largura da coluna, pretende-se obter um tuplo formado por cadeias de carateres de tamanho igual à largura pretendida. Todos os elementos do tuplo devem começar e terminar por um caráter diferente do caráter espaço (' '), com a eventual excepção das cadeias formadas apenas por uma palavra e a última cadeia do tuplo. Para atingir o tamanho pretendido devem ser inseridos espaços (' ') de forma uniforme entre as palavras, acrescentado espaços adicionais nos intervalos entre palavras, da esquerda para a direita. No caso das cadeias formadas apenas por uma palavra ou a última cadeia do tuplo, os espaços são inseridos no fim. Por exemplo, dado o seguinte texto exemplo:

> *Computers are incredibly fast, accurate, and stupid. Human beings are incredibly slow, inaccurate, and brilliant. Together they are powerful beyond imagination.*

Para uma largura de coluna igual a 60, pretende-se obter o seguinte tuplo:

```
('Computers  are  incredibly fast, accurate, and stupid. Human',
 'beings  are  incredibly  slow,  inaccurate,  and  brilliant.',
 'Together they are powerful beyond imagination.             ')
```

Como o texto inicial pode conter qualquer caráter, numa fase inicial é necessário *limpar* o texto. Isto é, todos os carateres brancos ASCII:

0x09, HT *horizontal tab* (`\t`)

0x0a, LF *line feed* (`\n`)

0x0b, VT *vertical tab* (`\v`)

0x0c, FF *form feed* (`\f`)

0x0d, CR *carriage return* (`\r`)

0x20, espaço (`' '`)

devem ser substituídos por espaços. Também deve ser eliminada qualquer repetição de espaços consecutivos e espaços iniciais e finais. Considere que uma palavra é qualquer sequência de carateres não-brancos.

1.2 Trabalho a realizar

O objetivo desta tarefa é escrever um programa em Python que permita justificar um texto arbitrário conforme descrito anteriormente. Para isso, é necessário definir o conjunto de funções solicitadas, assim como algumas funções auxiliares adicionais, caso seja necessário. Apenas as funções para as quais a verificação da correção dos argumentos é explicitamente pedida devem verificar a validade dos argumentos, para as outras assume-se que estão corretos.

1.2.1 limpa_texto: cad. carateres → cad. carateres (1,0 valores)

Esta função recebe uma cadeia de carateres qualquer e devolve a cadeia de carateres *limpa* que corresponde à remoção de carateres brancos conforme descrito.

1.2.2 corta_texto: cad. carateres × inteiro → cad. carateres × cad. carateres (1,0 valores)

Esta função recebe uma cadeia de carateres e um inteiro positivo correspondentes a um texto *limpo* e uma largura de coluna respetivamente, e devolve duas subcadeias de carateres *limpas*: a primeira contendo todas as palavras *completas* desde o início da cadeia original (incluindo os espaços separadores entre elas) até um comprimento máximo total igual à largura fornecida, e a segunda cadeia contendo o resto do texto de entrada. Ambas estas cadeias não devem começar ou terminar com um espaço e podem ser vazias dependendo do valor dos parâmetros concretos de entrada.

1.2.3 insere_espacos: cad. carateres × inteiro → cad. carateres (1,0 valores)

Esta função recebe uma cadeia de carateres e um inteiro positivo correspondentes a um texto *limpo* e uma largura de coluna respetivamente, e no caso da cadeia de entrada conter duas ou mais palavras devolve uma cadeia de carateres de comprimento igual à largura da coluna formada pela cadeia original com espaços entre palavras conforme descrito. Caso contrário, devolve a cadeia de comprimento igual à largura da coluna formada pela cadeia original seguida de espaços.

1.2.4 justifica_texto: cad. carateres × inteiro → tuplo (2,0 valores)

Esta função recebe uma cadeia de carateres não vazia e um inteiro positivo correspondentes a um texto qualquer e uma largura de coluna respetivamente, e devolve um tuplo de cadeias de carateres justificadas, isto é, de comprimento igual à largura da coluna com espaços entre palavras conforme descrito. Esta função deve verificar a validade dos seus argumentos, gerando um `ValueError` com a mensagem '`justifica_texto: argumentos invalidos`' caso os seus argumentos não sejam válidos. Para este fim, considere também que todas as palavras do texto de entrada devem ter tamanho igual ou menor à largura da coluna pretendida.

1.3 Exemplo

```
>>> limpa_texto('  Fundamentos\n\tda      Programacao\n')
'Fundamentos da Programacao'
>>> corta_texto('Fundamentos da Programacao', 15)
('Fundamentos da', 'Programacao')
>>> insere_espacos('Fundamentos da Programacao!!!', 30)
'Fundamentos  da Programacao!!!'
>>> insere_espacos('Fundamentos da Programacao!!!', 40)
'Fundamentos       da      Programacao!!!'
>>> cad = ('Computers are incredibly  \n\tfast,     \n\t\taccurate'
' \n\t\t\tand  stupid.   \n    Human beings are incredibly  slow  '
'inaccurate, and brilliant. \n     Together  they  are powerful   '
'beyond imagination.')
>>>  print(cad)
Computers are incredibly
        fast,
                accurate
                        and  stupid.
    Human beings are incredibly  slow  inaccurate, and brilliant.
     Together  they  are powerful   beyond imagination.
>>> justifica_texto(cad, 60)
('Computers  are  incredibly  fast, accurate and stupid. Human',
 'beings   are  incredibly  slow  inaccurate,  and  brilliant.',
 'Together they are powerful beyond imagination.             ')
>>> for l in justifica_texto(cad, 60): print(l)
Computers  are  incredibly  fast, accurate and stupid. Human
beings   are  incredibly  slow  inaccurate,  and  brilliant.
Together they are powerful beyond imagination.
```

2 Método de Hondt

2.1 Descrição do problema

A segunda tarefa consiste em obter os resultados de umas eleições utilizando o método de Hondt[1], um modelo matemático utilizado para converter votos em mandatos em sistemas eleitorais de representação proporcional. Este é o método mais utilizado no mundo, amplamente implementado em inúmeros países democráticos. Em particular, em Portugal a lei eleitoral da Assembleia da República segue o sistema de representação proporcional e utiliza o método de Hondt.

O método usa a divisão sucessiva do número total de votos obtidos por cada candidatura pelos divisores (1, 2, 3, 4, 5 etc. até o número de mandatos a eleger) e a atribuição dos mandatos em disputa por ordem decrescente aos quocientes mais altos que resultarem das divisões operadas. Em Portugal, caso falte atribuir o último mandato e se verifique igualdade do quociente em duas listas diferentes, tal mandato é atribuído à lista que em termos de resultados totais tenha obtido menor número de votos. Por exemplo, considerando um círculo eleitoral que tem direito a eleger 7 deputados e onde concorrem 4 partidos: A, B, C e D. Apurados os votos, a distribuição é a seguinte: A - 12.000 votos; B - 7.500 votos; C - 5.250 votos; e D - 3.000 votos. Da aplicação do método de Hondt resulta a série de quocientes mostrado na Tabela 1, sendo os quocientes correspondentes a mandatos assinalados a verde, de acordo com a seguinte distribuição:

- Partido A - 3 deputados, correspondentes aos quocientes 12000 (1.º eleito), 6000 (3.º eleito) e 4000 (5.º eleito). Note-se que apesar do quociente resultante da divisão por 4 ser 3000, igual aos votos obtidos pelo partido D, o mandato é atribuído ao menos votado, isto é, ao Partido D.
- Partido B - 2 deputados, correspondentes aos quocientes 7500 (2.º eleito) e 3750 (6.º eleito).
- Partido C - 1 deputado, correspondente ao quociente 5250 (4.º eleito).
- Partido D - 1 deputado, correspondente ao quociente 3000 (7.º e último eleito), beneficiando da regra que em igualdade atribui o lugar ao partido menos votado.

divisor	Partido			
	A	B	C	D
1	12000	7500	5250	3000
2	6000	3750	2625	1500
3	4000	2500	1750	1000
4	3000	1875	1312.5	750

Tabela 1: Quocientes para a distribuição de mandatos seguindo o método de Hondt (divisores maiores omitidos por não serem necessários).

[1] `https://www.cne.pt/content/metodo-de-hondt`

Considere que a informação sobre as eleições num território com vários círculos eleitorais é representada como um dicionário com um par chave/valor por cada círculo, onde a chave é igual ao nome do círculo (cadeia de carateres não vazia) e o valor é por sua vez um dicionário com número de deputados (chave 'deputados') e o apuramento dos votos (chave 'votos'):

```
{'Endor': {'deputados': 7,
  'votos': {'A':12000, 'B':7500, 'C':5250, 'D':3000}},
 'Hoth':  {'deputados': 6,
  'votos': {'A':9000, 'B':11500, 'D':1500, 'E':5000}},
 'Tatooine': {'deputados': 3,
  'votos': {'A':3000, 'B':1900}}}
```

Em cada círculo, o valor da chave 'deputados' é um inteiro positivo e o da chave 'votos' é um dicionário com pelo menos uma entrada de pares chave/valor correspondentes ao nome de um partido (cadeia de carateres não vazias) e aos votos obtidos (inteiro não negativo), respetivamente.

2.2 Trabalho a realizar

O objetivo desta tarefa é escrever um programa em Python que permita obter os resultados de umas eleições conforme descrito anteriormente. Para isso, deverá definir o conjunto de funções solicitadas, assim como algumas funções auxiliares adicionais, caso considere necessário. Apenas as funções para as quais a verificação da correção dos argumentos é explicitamente pedida devem verificar a validade dos argumentos.

2.2.1 calcula_quocientes: dicionário × inteiro → dicionário (1,0 valores)

Esta função recebe um dicionário com os votos apurados num círculo (com pelo menos um partido) e um inteiro positivo representando o número de deputados; e devolve o dicionário com as mesmas chaves do dicionário argumento (correspondente a partidos) contendo a lista (de comprimento igual ao número de deputados) com os quocientes calculados com o método de Hondt ordenados em ordem decrescente. Esta função não deve de alterar o dicionário que é passado como argumento de entrada.

2.2.2 atribui_mandatos: dicionário × inteiro → lista (2,0 valores)

Esta função recebe um dicionário com os votos apurados num círculo e um inteiro representando o número de deputados, e devolve a lista ordenada de tamanho igual ao número de deputados contendo as cadeias de carateres dos partidos que obtiveram cada mandato, isto é, a primeira posição da lista corresponde ao nome do partido que obteve o primeiro deputado, a segunda ao partido que obteve o segundo deputado, etc. Considere que no caso de existirem dois ou mais partidos com igual quociente, os mandatos são distribuídos por ordem ascendente às listas menos votadas. Esta função não deve alterar o dicionário que é passado como argumento de entrada.

2.2.3 obtem_partidos: dicionário → lista (1,0 valores)

Esta função recebe um dicionário com a informação sobre as eleições num território com vários círculos eleitorais como descrito, e devolve a lista por ordem alfabética com o nome de todos os partidos que participaram nas eleições. Esta função não deve de alterar o dicionário que é passado como argumento de entrada.

2.2.4 obtem_resultado_eleicoes: dicionário → lista (2,0 valores)

Esta função recebe um dicionário com a informação sobre as eleições num território com vários círculos eleitorais como descrito, e devolve a lista ordenada de comprimento igual ao número total de partidos com os resultados das eleições. Cada elemento da lista é um tuplo de tamanho 3 contendo o nome de um partido, o número total de deputados obtidos e o número total de votos obtidos. A lista está ordenada por ordem descendente de acordo ao número de deputados obtidos e, em caso de empate, de acordo ao número de votos. Esta função deve verificar a validade dos seus argumentos, gerando um `ValueError` com a mensagem `'obtem_resultado_eleicoes: argumento invalido'` caso o seu argumento não seja válido. Para este fim, considere que o dicionário contêm a informação de um ou mais círculos eleitorais como descrito anteriormente e que cada círculo eleitoral têm no mínimo um deputado e um ou mais partidos que receberam votos. Esta função não deve de alterar o dicionário que é passado como argumento de entrada.

2.3 Exemplo

```
>>>  calcula_quocientes({'A':12000, 'B':7500, 'C':5250, 'D':3000}, 7)
{'A': [12000.0, 6000.0, 4000.0, 3000.0, 2400.0, 2000.0, 1714.285714],
 'B': [7500.0, 3750.0, 2500.0, 1875.0, 1500.0, 1250.0, 1071.428571],
 'C': [5250.0, 2625.0, 1750.0, 1312.5, 1050.0, 875.0, 750.0],
 'D': [3000.0, 1500.0, 1000.0, 750.0, 600.0, 500.0, 428.571428]}
>>> atribui_mandatos({'A':12000, 'B':7500, 'C':4500, 'D':3000}, 7)
['A', 'B', 'A', 'C', 'A', 'B', 'D']
>>> info = {
'Endor':   {'deputados': 7,
            'votos': {'A':12000, 'B':7500, 'C':5250, 'D':3000}},
'Hoth':    {'deputados': 6,
            'votos': {'B':11500, 'A':9000, 'E':5000, 'D':1500}},
'Tatooine': {'deputados': 3,
            'votos': {'A':3000, 'B':1900}}}
>>> obtem_partidos(info)
['A', 'B', 'C', 'D', 'E']
>>> obtem_resultado_eleicoes(info)
[('A', 7, 24000), ('B', 6, 20900), ('C', 1, 5250),
 ('E', 1, 5000), ('D', 1, 4500)]
```

3 Solução de Sistemas de Equações

3.1 Descrição do problema

Esta terceira tarefa aborda o problema da solução de sistemas de equações lineares da forma:

$$\begin{cases} f_0(\mathbf{x}) &= c_0 \\ f_1(\mathbf{x}) &= c_1 \\ f_2(\mathbf{x}) &= c_2 \\ & \vdots \\ f_n(\mathbf{x}) &= c_n \end{cases}$$

em que cada f_i é uma função linear, c_i uma constante, e $\mathbf{x}$ é o vetor de variáveis a determinar, em número igual ao de equações. De notar que estes sistemas de equações podem ser escritos de forma concisa em termos matriciais $\mathbf{A} \times \mathbf{x} = \mathbf{c}$, em que $\mathbf{A}$ é a matriz dos coeficientes e $\mathbf{c}$ o vetor das constantes. Com esta representação o cálculo de $f_i(\mathbf{x})$ corresponde ao produto interno entre a linha i da matriz $\mathbf{A}$ e o vetor $\mathbf{x}$.

Um dos métodos mais simples para o cálculo da solução consiste em começar com uma estimativa inicial para a solução e ir aproximando esta cada vez mais da solução do sistema (método de Jacobi). Dada a estimativa da iteração k, a estimativa da iteração seguinte é calculada usando:

$$x_i^{(k+1)} = x_i^{(k)} + (c_i - f_i(\mathbf{x}^{(k)}))/a_{i,i}$$

em que $a_{i,i}$ é o coeficiente da variável x_i da função f_i.

Considere-se por exemplo o sistema seguinte de dimensão 3, com a matriz $\mathbf{A}$ e o vetor $\mathbf{c}$ equivalente em notação matricial:

$$\begin{cases} 2x_0 - x_1 - x_2 = -8 \\ 2x_0 - 9x_1 + 7x_2 = 8 \\ -2x_0 + 5x_1 - 9x_2 = -6 \end{cases} \qquad \mathbf{A} = \begin{bmatrix} 2 & -1 & -1 \\ 2 & -9 & 7 \\ -2 & 5 & -9 \end{bmatrix} \qquad \mathbf{c} = \begin{bmatrix} -8 \\ 8 \\ -6 \end{bmatrix}$$

Se o valor inicial for tudo 0s, a evolução da estimativa da solução está ilustrada na Tabela 2. Em geral, é definida uma precisão ϵ para a solução e o algoritmo termina quando todas as equações tiverem um erro inferior a essa precisão.

Cuidados a ter com este algoritmo:

1. a variável x_i tem que estar presente na equação f_i, isto é, não pode ter um coeficiente nulo. Esta condição é equivalente a dizer que a matriz $\mathbf{A}$ não pode ter zeros na diagonal.

2. embora não convirja para todos os sistemas de equações, a convergência é garantida para sistemas com matriz $\mathbf{A}$ diagonalmente dominante, isto é, o valor absoluto do valor da diagonal ser maior ou igual à soma dos restantes valores absolutos da linha: $|a_{ii}| \geq \sum_{j \neq i} |a_{ij}|$ para todas as linhas i.

Iteração	x_0	x_1	x_2
0	0	0	0
1	-4,000	-0,889	0,667
2	-4,111	-1,259	1,062
3	-4,099	-0,977	0,881
4	-4,048	-1,115	1,035
5	-4,040	-0,984	0,947
6	-4,018	-1,050	1,018
7	-4,016	-0,990	0,976
8	-4,007	-1,022	1,009
9	-4,006	-0,994	0,989
10	-4,003	-1,000	1,005
11	-4,003	-0,997	0,995
12	-4,001	-1,004	1,002
13	-4,001	-0,998	0,998
14	-4,000	-1,002	1,001
15	-4,000	-0,999	0,999
16	-4,000	-1,001	1,001
17	-4,000	-1,000	1,000

Tabela 2: Exemplo ilustrativo da convergência do método de Jacobi.

Adicionalmente, de forma a ter uma solução única, o número de equações deve ser igual ao número de variáveis, e nenhuma equação pode ser obtida por combinações lineares de outras linhas. Esta condição é equivalente a dizer que o determinante da matriz **A** não pode ser zero. Este problema (o cálculo do determinante) não será abordado nesta tarefa.

3.2 Trabalho a realizar

Nesta tarefa deverá escrever um programa em Python que implemente o método iterativo de Jacobi para a resolução de um sistema de equações. Antes do início da resolução, o programa deverá, por esta ordem:

1. verificar a validade dos argumentos;
2. eliminar entradas a zero da diagonal por troca de linhas da matriz;
3. garantir que o método converge testando se a matriz é diagonal dominante.

Deverá definir o conjunto de funções solicitadas, assim como algumas funções auxiliares adicionais, caso considere necessário. Apenas as funções para as quais a verificação da correção dos argumentos é explicitamente pedida devem verificar a validade dos argumentos.

3.2.1 produto_interno: tuplo × tuplo → real (0,5 valores)

Esta função recebe dois tuplos de números (inteiros ou reais) com a mesma dimensão representando dois vetores e retorna o resultado do produto interno desses dois vetores.

3.2.2 verifica_convergencia: tuplo × tuplo × tuplo × real → booleano (0,75 valores)

Esta função recebe três tuplos de igual dimensão e um valor real positivo. O primeiro tuplo é constituído por um conjunto de tuplos cada um representando uma linha da matriz quadrada $\mathbf{A}$, e os outros dois tuplos de entrada contêm valores representando respetivamente o vetor de constantes $\mathbf{c}$ e a solução atual $\mathbf{x}$. O valor real de entrada indica a precisão pretendida ϵ. A função deverá retornar `True` caso o valor absoluto do erro de todas as equações seja inferior à precisão, $|f_i(x) - c_i| < \epsilon$, e `False` caso contrário.

3.2.3 retira_zeros_diagonal: tuplo × tuplo → tuplo × tuplo (1,0 valores)

Esta função recebe um tuplo de tuplos, representando a matriz de entrada no mesmo formato das funções anteriores, e um tuplo de números, representando o vetor das constantes. A função deverá retornar uma nova matriz com as mesmas linhas que a de entrada, mas com estas reordenadas de forma a não existirem valores 0 na diagonal. O segundo parâmetro de saída é também o vetor de entrada com a mesma reordenação de linhas que a aplicada à matriz. Para este efeito, e de forma a uniformizar o resultado, deverá inspecionar por ordem todas as linhas da matriz à procura de um 0 na diagonal. Se na linha i se encontrar um 0 na diagonal, esta linha deve ser trocada pela primeira linha j que, procurando desde o início da matriz, não contenha um 0 na coluna i, sempre que na linha i não haja um 0 na coluna j.

3.2.4 eh_diagonal_dominante: tuplo → booleano (0,75 valores)

Esta função recebe um tuplo de tuplos representando uma matriz quadrada no mesmo formato das funções anteriores. Deverá retornar `True` caso seja uma matriz diagonalmente dominante, e `False` caso contrário.

3.2.5 resolve_sistema: tuplo × tuplo × real → tuplo (2,0 valores)

Esta função recebe um tuplo de tuplos representando uma matriz quadrada no mesmo formato das funções anteriores correspondente aos coeficientes das equações do sistema, um tuplo de números representando o vetor das constantes, e um valor real positivo correspondente à precisão pretendida para a solução. Deverá retornar um tuplo que é a solução do sistema de equações de entrada aplicando o método de Jacobi descrito. Considere para este efeito que o valor inicial para todas as variáveis do sistema de equações é igual a 0. Em caso de erro deverá gerar um `ValueError` com as seguintes mensagens consoante o erro:

- argumentos invalidos: `'resolve_sistema: argumentos invalidos'`
- diagonal não-dominante: `'resolve_sistema: matriz nao diagonal dominante'`

3.3 Exemplo

```
>>> produto_interno((1,2,3,4,5),(-4,5,-6,7,-8))
-24.0
>>> A1, c1 = ((1, -0.5), (-1, 2)), (-0.4, 1.9)
>>> verifica_convergencia(A1, c1, (0.1001, 1), 0.00001)
False
>>> verifica_convergencia(A1, c1, (0.1001, 1), 0.001)
True
>>> A2, c2 = ((0, 1, 1), (1, 0, 0), (0, 1, 0)), (1, 2, 3)
>>> retira_zeros_diagonal(A2, c2)
(((1, 0, 0), (0, 1, 0), (0, 1, 1)), (2, 3, 1))
>>> A3 = ((1, 2, 3, 4, 5),(4, -5, 6, -7, 8), (1, 3, 5, 3, 1),
         (-1, 0, -1, 0, -1), (0, 2, 4, 6, 8))
>>> eh_diagonal_dominante(A3)
False
>>> eh_diagonal_dominante(((1, 0, 0), (0, 1, 0), (0, 1, 1)))
True
>>> A4, c4 = ((2, -1, -1), (2, -9, 7), (-2, 5, -9)), (-8, 8, -6)
>>> resolve_sistema(A4, c4, 1e-20)
(-4.0, -1.0, 1.0)
```

4 Condições de Realização e Prazos

- A entrega do 1º projeto será efetuada exclusivamente por via eletrónica. Deverá submeter o seu projeto através do sistema Mooshak, até às **17:00 do dia 28 de Outubro de 2022**. Depois desta hora, não serão aceites projetos sob pretexto algum.
- Deverá submeter um único ficheiro com extensão *.py* contendo todo o código do seu projeto.
- O sistema de submissão assume que o ficheiro está codificado em UTF-8. Alguns editores podem utilizar uma codificação diferente, ou a utilização de alguns carateres mais estranhos (nomeadamente nos comentários) não representáveis em UTF-8 pode levar a outra codificação. Se todos os testes falharem, pode ser um problema da codificação usada. Nesse caso, deverá especificar qual é a codificação do ficheiro na primeira linha deste[2].
- Submissões que não corram nenhum dos testes automáticos por causa de pequenos erros de sintaxe ou de codificação, poderão ser corrigidos pelo corpo docente, incorrendo numa penalização de três valores.

[2] `https://www.python.org/dev/peps/pep-0263/`

- Não é permitida a utilização de qualquer módulo ou função não disponível built-in no Python 3.
- Pode, ou não, haver uma discussão oral do trabalho e/ou uma demonstração do funcionamento do programa (será decidido caso a caso).
- Lembre-se que no Técnico, a fraude académica é levada muito a sério e que a cópia numa prova (projetos incluídos) leva à reprovação na disciplina e eventualmente a um processo disciplinar. Os projetos serão submetidos a um sistema automático de deteção de cópias[3], o corpo docente da cadeira será o único juiz do que se considera ou não copiar num projeto.

5 Submissão

A submissão e avaliação da execução do projeto de FP é feita utilizando o sistema Mooshak[4]. Para obter as necessárias credenciais de acesso e poder usar o sistema deverá:

- Obter a senha para acesso ao sistema, seguindo as instruções na página: `http://acm.tecnico.ulisboa.pt/~fpshak/cgi-bin/getpass-fp22`. A senha ser-lhe-á enviada para o email que tem configurado no Fenix. A senha pode não chegar de imediato, aguarde.
- Após ter recebido a sua senha por email, deve efetuar o login no sistema através da página: `http://acm.tecnico.ulisboa.pt/~fpshak/`. Preencha os campos com a informação fornecida no email.
- Utilize o botão *"Browse..."*, selecione o ficheiro com extensão *.py* contendo todo o código do seu projeto. O seu ficheiro *.py* deve conter a implementação das funções pedidas no enunciado. De seguida clique no botão *"Submit"* para efetuar a submissão.
  Aguarde (20-30 seg) para que o sistema processe a sua submissão!!!
- Quando a submissão tiver sido processada, poderá visualizar na tabela o resultado correspondente. Receberá no seu email um relatório de execução com os detalhes da avaliação automática do seu projeto podendo ver o número de testes passados/falhados.
- Para sair do sistema utilize o botão *"Logout"*.

Submeta o seu projeto atempadamente, dado que as restrições seguintes podem não lhe permitir fazê-lo no último momento:

- Só poderá efetuar uma nova submissão 5 minutos depois da submissão anterior.

[3] `https://theory.stanford.edu/~aiken/moss`

[4] A versão de Python utilizada nos testes automáticos é Python 3.7.3.

- O sistema só permite 10 submissões em simultâneo pelo que uma submissão poderá ser recusada se este limite for excedido [5].
- Não pode ter submissões duplicadas, ou seja, o sistema pode recusar uma submissão caso seja igual a uma das anteriores.
- Será considerada para avaliação a **última** submissão (mesmo que tenha pontuação inferior a submissões anteriores). Deverá, portanto, verificar cuidadosamente que a última entrega realizada corresponde à versão do projeto que pretende que seja avaliada. Não há excepções!
- Cada aluno tem direito a **15 submissões sem penalização** no Mooshak. Por cada submissão adicional serão descontados 0,1 valores na componente de avaliação automática.

6 Classificação

A nota do projeto será baseada nos seguintes aspetos:

1. **Avaliação automática (80%).** A avaliação da correta execução será feita através do sistema Mooshak. O tempo de execução de cada teste está limitado, bem como a memória utilizada.
   Serão usados um conjunto de testes públicos (disponibilizados na página da disciplina) e um conjunto de testes privados. O resultado da avaliação automática é calculado em função da fração de testes privados que o programa passa. Como a avaliação automática vale 80% (equivalente a 16 valores) da nota do projeto, uma submissão obtém a nota máxima de 1600 pontos.
   O facto de um projeto completar com sucesso os testes públicos fornecidos (naturalmente, só devem submeter depois do programa passar todos estes testes) não implica que esse projeto esteja totalmente correto, pois estes não são exaustivos. É da responsabilidade de cada aluno garantir que o código produzido está de acordo com a especificação do enunciado usando testes próprios adicionais, de forma a completar com sucesso os testes privados.
2. **Avaliação manual (20%).** Estilo de programação e facilidade de leitura. Em particular, serão consideradas as seguintes componentes:
   - Boas práticas (1,5 valores): serão considerados entre outros a clareza do código, a integração de conhecimento adquirido durante a UC e a criatividade das soluções propostas.
   - Comentários (1 valor): deverão incluir a assinatura das funções definidas, comentários para o utilizador (*docstring*) e comentários para o programador.

[5] Note que o limite de 10 submissões simultâneas no sistema Mooshak implica que, caso haja um número elevado de tentativas de submissão sobre o prazo de entrega, alguns alunos poderão não conseguir submeter nessa altura e verem-se, por isso, impossibilitados de submeter o código dentro do prazo.

- Tamanho de funções, duplicação de código e abstração procedimental (1 valor).
- Escolha de nomes (0,5 valores).

7 Recomendações e aspetos a evitar

As seguintes recomendações e aspetos correspondem a sugestões para evitar maus hábitos de trabalho (e, consequentemente, más notas no projeto):

- Leia todo o enunciado, procurando perceber o objetivo das várias funções pedidas. Em caso de dúvida de interpretação, utilize o horário de dúvidas para esclarecer as suas questões.
- No processo de desenvolvimento do projeto, comece por implementar as várias funções dentro de cada tarefa pela ordem apresentada no enunciado, seguindo as metodologias estudadas na disciplina. Como as tarefas são independentes umas das outras, estas podem ser resolvidas por qualquer ordem e com independência umas das outras.
- Para verificar a funcionalidade das suas funções, utilize os exemplos fornecidos como casos de teste. Tenha o cuidado de reproduzir fielmente as mensagens de erro e restantes *outputs*, conforme ilustrado nos vários exemplos.
- Não pense que o projeto se pode fazer nos últimos dias. Se apenas iniciar o seu trabalho neste período irá sentir a Lei de Murphy em funcionamento (todos os problemas são mais difíceis do que parecem; tudo demora mais tempo do que nós pensamos; e se alguma coisa puder correr mal, ela vai correr mal, na pior das alturas possíveis).
- Não duplique código. Se duas funções são muito semelhantes é natural que estas possam ser fundidas numa única, eventualmente com mais argumentos.
- Não se esqueça que as funções excessivamente grandes são penalizadas no que respeita ao estilo de programação.
- A atitude "vou pôr agora o programa a correr de qualquer maneira e depois preocupo-me com o estilo" é totalmente errada.
- Quando o programa gerar um erro, preocupe-se em descobrir qual a causa do erro. As "marteladas" no código têm o efeito de distorcer cada vez mais o código.