Mestrado em Informação e Sistemas Empresariais

Introdução à Programação 2023/2024

Projeto 1

Data Enunciado: 22 de janeiro de 2024
Data Limite de Entrega: 28 de janeiro de 2024 (23:59)

1 Descrição do problema

Neste projeto pretende-se que os alunos desenvolvam um conjunto de funções destinadas a manipular estruturas de dados para gestão de uma oficina. A oficina recebe veículos para reparação provenientes de vários clientes. Existem duas componentes a serem consideradas: a gestão dos clientes e dos seus veículos e a gestão das reparações da oficina.

2 Gestão dos clientes

A oficina mantém um registo dos seus clientes. No seu programa, a informação sobre os clientes deverá ser guardada numa lista, designada pela variável `clientes`, definida no quadro global. Para tal, deverá defini-la no início do seu ficheiro, fora de qualquer função, da seguinte forma:

```
clientes = []
```

Esta lista irá conter tuplos, cada um dos quais guardará a informação relativa a cada um dos clientes da oficina. No trabalho a desenvolver, a informação descritiva de cada cliente será apenas o seu nome completo, NIF e o seu número de telemóvel. Assim, para cada cliente, será guardada a seguinte informação:

- **nome completo**: cadeia de caracteres com comprimento máximo de 255 caracteres;

- **número de identificação fiscal (NIF)**: cadeia de caracteres constituída por 9 dígitos, sendo os primeiros 8 dígitos sequenciais e o último um dígito de controlo;
- **número de telemóvel**: cadeia de caracteres com 9 dígitos, sendo que o primeiro dígito é sempre o número 9;
- **veículos**: veículos do cliente que foram reparados na oficina; lista, em que cada elemento da lista é um tuplo contendo o tipo de veículo, a matrícula do veículo e a sua marca;
  - **tipo de veículo**: cadeia de caracteres com dois valores possíveis, "MOTOCICLO"ou "CARRO";
  - **matrícula do veículo**: cadeia de caracteres que segue a estrutura "YY-YY-YY"; de modo a simplificar a implementação, Y pode ser qualquer número de 0 a 9 ou qualquer letra maíscula de A a Z, sem restrições na sequência de caracteres;
  - **marca**: cadeia de caracteres com comprimento máximo de 25 caracteres;
  - **reparacoes**: reparações do veículo; lista, em que cada elemento da lista é uma lista contendo o ID da reparação, a data de entrada na oficina, a data estimada para entrega do veículo e a data de entrega do veículo;
    * ID da reparação; número inteiro positivo;
    * tuplo, contendo 3 elementos, em que cada elemento representa o dia (1-31), mês (1-12) e ano, respetivamente, da data em que o veículo entrou na oficina;
    * tuplo, contendo 3 elementos, em que cada elemento representa o dia (1-31), mês (1-12) e ano, respetivamente, da data estimada para entrega do veículo ao cliente;
    * tuplo, contendo 3 elementos, em que cada elemento representa o dia (1-31), mês (1-12) e ano, respetivamente, da data de entrega do veículo ao cliente; **caso o veículo ainda não tenha sido entregue, este campo deverá estar preenchido com um tuplo vazio**.

A informação relativa a cada cliente deverá ser guardada num tuplo em que os campos descritos acima devem encontrar-se organizados pela mesma ordem pela qual foram apresentados. Cada elemento da lista representada pela variável `clientes` irá guardar o tuplo correspondente a um cliente. Apresenta-se abaixo um exemplo de lista `clientes`, contendo a informação de dois clientes:

```
clientes = [
    ('António Manuel da Silva', '219072230', '912345678', [
        ('MOTOCICLO', 'AA-01-02', 'SUZUKI', [
            [253, (21,2,2021), (27,2,2021), (28,2,2021)],
```

```
            [794, (1,1,2024), (3,1,2024), ()]
        ])
    ]),
    ('José Maria Rebelo de Sousa', '248537474', '918765432', [])
]
```

De seguida apresenta-se a descrição das funções relativas à gestão de clientes que deverá implementar.

2.1 Função `valida_nif` [2 valores]

`valida_nif(nif)`

Esta função, que recebe como único argumento uma cadeia de carateres `nif` com um NIF, deverá devolver um valor booleano, `True` ou `False`, conforme `nif` seja um NIF válido ou não.

O NIF é um número de 9 dígitos que tem como finalidade identificar uma entidade fiscal. Este número é constituído por 8 dígitos sequenciais e um dígito final de validação (*check digit*), que serve apenas para validar se não houve engano na introdução dos 8 dígitos anteriores. O dígito de validação é calculado da seguinte forma:

1. Multiplique o 8º dígito por 2, o 7º dígito por 3, o 6º dígito por 4, o 5º dígito por 5, o 4º dígito por 6, o 3º dígito por 7, o 2º dígito por 8 e o 1º dígito por 9;
2. Some os resultados;
3. Calcule o resto da divisão do número por 11;
4. Se o resto for 0 ou 1, o dígito de controlo será 0;
5. Caso contrário, o dígito de controlo será o resultado de subtrair o resto da divisão a 11 (11 − resto da divisão);

Por exemplo, para o NIF 219072230, a soma resultante do primeiro e segundo passo é $3 \times 2 + 2 \times 3 + 2 \times 4 + 7 \times 5 + 0 \times 6 + 9 \times 7 + 1 \times 8 + 2 \times 9 = 144$. Em seguida, o resto da divisão por 11 é $144 \,\%\, 11 = 1$. Neste caso, como o resto é 1, o dígito de controlo será 0 (segundo o 4º passo), verificando-se que o NIF é válido, dado que o último dígito do NIF corresponde ao dígito que acabámos de calcular. Assim, a função `valida_nif` deveria devolver o valor `True`.

2.2 Função `adiciona_veiculo` [2 valores]

`adiciona_veiculo(cliente, veiculo)`

Esta função associa um veículo a um cliente. O veículo é identificado pela sua matrícula e é guardada a informação sobre o tipo e marca do veículo. Esta função deverá adicionar o veículo ao final da lista de veículos do cliente `cliente`.
A função deverá ainda tratar as seguintes situações:

- validar que o tipo de veículo corresponde à cadeia de caracteres "MOTOCICLO"ou "CARRO"; caso não corresponda, deverá lançar uma exceção do tipo `ValueError`, com a mensagem `'tipo de veículo inválido'`;
- verificar que a matrícula do veículo é válida, ou seja, que satisfaz as condições apresentadas na Secção 2; caso não seja, deverá lançar uma exceção do tipo `ValueError`, com a mensagem `'matrícula inválida'`;
- validar que a marca do veículo é uma cadeia de caracteres que não excede o comprimento máximo de 25 caracteres; caso não seja, deverá lançar uma exceção do tipo `ValueError`, com a mensagem `'marca inválida'`;
- caso um veículo com a mesma matrícula já esteja associado ao cliente `cliente`, deverá lançar uma exceção do tipo `ValueError`, com a mensagem `'veículo já se encontra associado a este cliente'`.

Exemplo

O exemplo abaixo apresenta o valor de `cliente` antes e depois da chamada à função.

```
cliente = ('António Manuel da Silva', '219072230', '912345678', [
    ('MOTOCICLO', 'AA-01-02', 'SUZUKI', [])
])

adiciona_veiculo(cliente, ('CARRO', 'BB-01-02', 'BMW', []))

cliente = ('António Manuel da Silva', '219072230', '912345678', [
    ('MOTOCICLO', 'AA-01-02', 'SUZUKI', []), ('CARRO', 'BB-01-02', 'BMW', [])
])
```

2.3 Função `adiciona_cliente` [3 valores]

`adiciona_cliente(nome_completo, nif, telemovel, veiculos)`

Esta função adiciona um novo cliente ao registo da oficina. O cliente é identificado pelo seu NIF `nif` e é guardada a informação sobre o seu nome completo `nome_completo`, número de telemóvel `telemovel` e os veículos que lhe pertencem `veiculos`. Caso o cliente ainda não exista nos registos da oficina, esta função deverá adicioná-lo ao final da lista `clientes`. Caso já exista nos registos da oficina um cliente com o mesmo NIF, a função deverá apenas adicionar os veículos `veiculos` à lista de veículos do cliente.
A função deverá ainda tratar as seguintes situações:

- verificar que o NIF `nif` é válido; caso não seja, deverá lançar uma exceção do tipo `ValueError`, com a mensagem `'NIF inválido'`;
- validar que o nome completo `nome_completo` é uma cadeia de caracteres que não excede os 255 caracteres de comprimento; caso não seja, deverá lançar uma exceção do tipo `ValueError`, com a mensagem `'nome completo inválido'`;
- validar que o número de telemóvel é uma cadeia de caracteres constituída por 9 dígitos sendo o primerio dígito o número '9'; caso não seja, deverá lançar uma exceção do tipo `ValueError`, com a mensagem `'número de telemóvel inválido'`;
- cada veículo em `veiculos` deverá sofrer das mesmas validações da função `adiciona_veiculo`;
- caso o cliente já exista no registo da oficina, verificar se o nome completo fornecido `nome_completo` é igual ao nome completo do cliente existente no registo da oficina com o mesmo NIF; caso não seja, deverá lançar uma exceção do tipo `ValueError` com a mensagem `'nome completo incoerente'`.

Exemplo

O exemplo abaixo apresenta o valor de `clientes` antes e depois da chamada à função.

```
clientes = [
    ('José Maria Rebelo de Sousa', '248537474', '918765432', [])
]

adiciona_cliente('António Manuel da Silva', '219072230', '912345678', [
    ('MOTOCICLO', 'AA-01-02', 'SUZUKI', [])
])
```

```
clientes = [
    ('José Maria Rebelo de Sousa', '248537474', '918765432', []),
    ('António Manuel da Silva', '219072230', '912345678', [
        ('MOTOCICLO', 'AA-01-02', 'SUZUKI', [])
    ])
]
```

2.4 Função `remove_cliente` [2 valores]

`remove_cliente(nif)`

Esta função remove o cliente com NIF `nif` do registo de clientes da oficina. Caso o cliente tenha veículos com reparações em curso, deverá ser lançada uma exceção do tipo `ValueError`, com a mensagem `'reparações em curso'`. Caso o NIF `nif` seja inválido, deverá ser lançada uma exceção do tipo `ValueError`, com a mensagem `'NIF inválido'`. Caso não exista nenhum cliente com o NIF indicado, a função não faz nada, nem devolve qualquer exceção.

Exemplo

```
clientes = [
    ('José Maria Rebelo de Sousa', '248537474', '918765432', []),
    ('António Manuel da Silva', '219072230', '912345678', [
        ('MOTOCICLO', 'AA-01-02', 'SUZUKI', [])
    ])
]

remove_cliente('219072230')
remove_cliente('219072230')

clientes = [
    ('José Maria Rebelo de Sousa', '248537474', '918765432', [])
]
```

2.5 Função `procura_veiculo` [1 valores]

`procura_veiculo(matricula)`

Esta função deverá procurar nos veículos dos clientes da lista `clientes` o veículo com matrícula `matricula` e devolvê-lo. Considere que não podem existir dois veículos com a

mesma matrícula em clientes diferentes. Caso a matrícula `matricula` seja inválida, deverá ser lançada uma exceção do tipo `ValueError` com a mensagem `'matrícula inválida'`. Caso um veículo com matrícula `matricula` não exista, a função deverá lançar uma exceção do tipo `ValueError` com a mensagem `'veículo não encontrado'`.

Exemplo

O exemplo abaixo apresenta o valor de `clientes`, bem como o valor devolvido pela função quando é efetuada a procura pela matrícula `'AA-01-02'`.

```
clientes = [
    ('José Maria Rebelo de Sousa', '248537474', '918765432', []),
    ('António Manuel da Silva', '219072230', '912345678', [
        ('MOTOCICLO', 'AA-01-02', 'SUZUKI', [])
    ])
]

procura_veiculo('AA-01-02')
('MOTOCICLO', 'AA-01-02', 'SUZUKI', [])
```

3 Gestão de reparações

Quando um cliente coloca um veículo para reparar na oficina, é criado um novo registo de reparação no sistema de gestão da oficina. Os registos de reparação são modelados e guardados segundo a estrutura apresentada na Secção 2. Os IDs das reparações são atribuídos de forma sequencial, ou seja, se o ID da última reparação for 253, o ID da reparação registada imediatamente a seguir será 254. O ID da próxima reparação deverá ser guardado numa variável `id_reparacao`, definida no quadro global, e atualizada conforme novos registos de reparações são inseridos no sistema. Para tal, deverá definir a variável `id_reparacao` no ínicio do seu ficheiro, fora de qualquer função, da seguinte forma:

```
id_reparacao = 0
```

Caso pretenda modificar esta variável global dentro de uma função, deverá usar a palavra-chave `global` do Python.

```
def foo():
    global id_reparacao
    id_reparacao += 1
```

Aconselha-se prudência ao utilizar variáveis globais e a palavra-chave `global`, dado que o seu uso pode complicar a manutenção de código a longo prazo. Em projetos reais é

recomendável procurar alternativas que evitem a utilização de variáveis globais, ainda que, no âmbito específico deste exercício, estas sejam adotadas por simplificação.

Assume-se que as datas e IDs passados como argumentos das funções desta secção têm sempre o tipo e formato correto.

3.1 Função `inicia_reparacao` [3 valor]

`inicia_reparacao(matricula, data_entrada, data_estimada)`

Esta função inicia uma nova reparação no veículo com matrícula `matricula` com data de entrada `data_entrada` e data estimada de entrega `data_entrega`. A data de entrega da reparação criada deverá ser preenchida com um tuplo vazio. Caso o veículo já se encontre a ser reparado, a data estimada de entrega deve ser alterada para a nova data `data_entrega`. Se a data de entrada da reparação em curso no sistema não coincidir com `data_entrada`, deverá ser lançada uma exceção do tipo `ValueError`, com a mensagem `'data de entrada incoerente'`.
A função deverá ainda tratar as seguintes situações:

- caso não exista nenhum veículo com a matrícula indicada, o comportamento deverá coincidir com o comportamento da função `procura_veiculo`;
- caso a data de entrada seja superior à data estimada de entrega, deverá lançar uma exceção do tipo `ValueError`, com a mensagem `'data de entrada não pode ser superior à data estimada de entrega'`;

Exemplo

O exemplo abaixo apresenta o valor de `clientes` e de `id_reparacao` antes e depois da chamada à função.

```
id_reparacao = 0
clientes = [
    ('José Maria Rebelo de Sousa', '248537474', '918765432', []),
    ('António Manuel da Silva', '219072230', '912345678', [
        ('MOTOCICLO', 'AA-01-02', 'SUZUKI', [])
    ])
]

inicia_reparacao('AA-01-02', (1,1,2022), (3,1,2022))

id_reparacao = 1
clientes = [
```

```
    ('José Maria Rebelo de Sousa', '248537474', '918765432', []),
    ('António Manuel da Silva', '219072230', '912345678', [
        ('MOTOCICLO', 'AA-01-02', 'SUZUKI', [[0, (1,1,2022), (3,1,2022), ()]])
    ])
]
```

3.2 Função `anula_reparacao` [1 valor]

`anula_reparacao(id)`

Esta função elimina a reparação com ID `id`. No caso da reparação com este ID não exisitr, a função não faz nada. Quando uma reparação é anulada, o ID da próxima reparação a ser inserida não é alterado.

Exemplo

O exemplo abaixo apresenta o valor de `clientes` e de `id_reparacao` antes e depois da chamada à função.

```
id_reparacao = 1
clientes = [
    ('José Maria Rebelo de Sousa', '248537474', '918765432', []),
    ('António Manuel da Silva', '219072230', '912345678', [
        ('MOTOCICLO', 'AA-01-02', 'SUZUKI', [[0, (1,1,2022), (3,1,2022), ()]])
    ])
]

anula_reparacao(0)

id_reparacao = 1
clientes = [
    ('José Maria Rebelo de Sousa', '248537474', '918765432', []),
    ('António Manuel da Silva', '219072230', '912345678', [
        ('MOTOCICLO', 'AA-01-02', 'SUZUKI', [])
    ])
]
```

3.3 Função `procura_reparacao` [1 valor]

`procura_reparacao(id)`

Esta função deverá procurar pela reparação com ID `id` e devolver um tuplo com os seguintes elementos (por esta ordem): NIF do cliente que pediu a reparação, matrícula do veículo reparado/a reparar e a própria reparação. Caso não exista nenhuma reparação com ID `id`, a função deverá lançar uma exceção do tipo `ValueError` com a mensagem `'reparação não encontrada'`.

Exemplo

O exemplo abaixo apresenta o valor de `clientes`, bem como o valor devolvido pela função quando é efetuada a procura pelo ID `0`.

```
clientes = [
    ('José Maria Rebelo de Sousa', '248537474', '918765432', []),
    ('António Manuel da Silva', '219072230', '912345678', [
        ('MOTOCICLO', 'AA-01-02', 'SUZUKI', [[0, (1,1,2022), (3,1,2022), ()]])
    ])
]

procura_reparacao(0)
('219072230', 'AA-01-02', [0, (1,1,2022), (3,1,2022), ()])
```

3.4 Função `reparacao_atrasada` [1.5 valores]

`reparacao_atrasada(data)`

Esta função recebe um tuplo contendo a data atual `data`, utilizando o mesmo formato já definido anteriormente para outra datas, e devolve um tuplo em que cada elemento será também um tuplo contendo o NIF de um cliente, a matrícula de um veículo e uma reparação em que a data estimada foi ultrapassada (i.e. quando `data` corresponde a um dia posterior à data de entrega estimada para a reparação). Os elementos devolvidos devem estar ordenados primeiramente pelo NIF do cliente e no caso de existirem NIFs iguais, o desempate deve ser feito pela matrícula do veículo. A ordem considerada para os NIFs é a ordem pela qual estes aparecem na variável `clientes`. A ordem considerada para as matrículas é a ordem pela qual estas aparecem na lista de veículos de um dado cliente. Esta função não efetua qualquer modificação na variável `clientes`.

Exemplo

O exemplo abaixo apresenta o valor devolvido pela função quando é passada a data atual de 04/01/2022.

```
clientes = [
    ('José Maria Rebelo de Sousa', '248537474', '918765432', [
        ('CARRO', 'CC-01-02', 'BMW', [[0, (1,1,2022), (3,1,2022), ()]]),
        ('CARRO', 'DD-01-02', 'BMW', [
            [0, (1,1,2022), (3,1,2022), (3,1,2022)]
        ]),
        ('CARRO', 'EE-01-02', 'BMW', [[0, (1,1,2022), (5,1,2022), ()]])
    ]),
    ('António Manuel da Silva', '219072230', '912345678', [
        ('MOTOCICLO', 'AA-01-02', 'SUZUKI', [[0, (1,1,2022), (3,1,2022), ()]]),
        ('MOTOCICLO', 'BB-01-02', 'SUZUKI', [[0, (1,1,2022), (2,1,2022), ()]])
    ])
]

reparacao_atrasada((4, 1, 2022))
(
    ('248537474', 'CC-01-02', [0, (1,1,2022), (3,1,2022), ()]),
    ('219072230', 'AA-01-02', [0, (1,1,2022), (3,1,2022), ()]),
    ('219072230', 'BB-01-02', [0, (1,1,2022), (2,1,2022), ()]),
)
```

3.5 Função `finaliza_reparacao` [2.5 valores]

`finaliza_reparacao(id, data)`

Esta função finaliza a reparação com ID `id` e emite o respetivo recibo contendo a informação relativa à reparação. O recibo da reparação, a ser devolvido pela função, deverá ser modelado por um tuplo com os seguintes elementos (por esta ordem): NIF do cliente que pediu a reparação, matrícula do veículo reparado, data de entrada na oficina e data de entrega do veículo. Esta função altera a data de entrega da reparação com ID `id` para a data `data`.
A função deverá ainda tratar as seguintes situações:

- caso a data de entrada do veículo seja superior à data de entrega, deverá lançar uma exceção do tipo `ValueError`, com a mensagem `'data de entrada não pode ser superior à data de entrega'`;
- caso a reparação com ID `id` já tenha sido finalizada antes da chamada à função, deverá lançar uma exceção do tipo `ValueError`, com a mensagem `'reparação já foi finalizada'`;
- caso não exista nenhuma reparação com ID `id`, o comportamento deverá coincidir com o comportamento da função `procura_reparacao`.

Exemplo

O exemplo abaixo apresenta o valor de `clientes` antes e depois da chamada à função e o valor devolvido pela função.

```
clientes = [
    ('José Maria Rebelo de Sousa', '248537474', '918765432', []),
    ('António Manuel da Silva', '219072230', '912345678', [
        ('MOTOCICLO', 'AA-01-02', 'SUZUKI', [[0, (1,1,2022), (3,1,2022), ()]])
    ])
]

finaliza_reparacao(0, (4,1,2022))
('219072230', 'AA-01-02', (1,1,2022), (4,1,2022))

clientes = [
    ('José Maria Rebelo de Sousa', '248537474', '918765432', []),
    ('António Manuel da Silva', '219072230', '912345678', [
        ('MOTOCICLO', 'AA-01-02', 'SUZUKI', [[0, (1,1,2022), (3,1,2022), (4,1,2022)]])
    ])
]
```

3.6 Função `imprime_recibo` [1 valor]

`imprime_recibo(recibo)`

Esta função escreve no ecrã a informação de um recibo `recibo`, com o seguinte formato:

- uma linha com o caráter `'-'` repetido 49 vezes;
- uma linha com a cadeira de carateres `' Cliente:'` (notar o espaço no início), seguida de um espaço e seguida do NIF do cliente;
- uma linha com a cadeira de carateres `' Veículo:'` (notar o espaço no início), seguida de um espaço e seguida da matrícula do veículo;
- uma linha com a cadeira de carateres `'Data de entrada:'`, seguida de um espaço e seguida da data de entrada do veículo na oficina no formato `dd/mm/aaaa`;
- uma linha com o caráter `'-'` repetido 49 vezes;
- uma linha com a cadeira de carateres `'Data de entrega:'`, seguida de um espaço e seguida da data de entrega do veículo no formato `dd/mm/aaaa`;

- uma linha com o caráter `'-'` repetido 49 vezes;

Cada uma das linhas deverá ser terminada com o caráter `'\n'`. Não é necessário verificar a validade do recibo.

Exemplo

Por exemplo, a informação escrita pela função para o recibo considerado no exemplo anterior, deverá ser:

```
-------------------------------------------------
 Cliente: 219072230
 Veículo: AA-01-02
Data de entrada: 01/01/2022
-------------------------------------------------
Data de entrega: 04/01/2022
-------------------------------------------------
```

4 Aspetos a considerar

- Deverá ler o projeto com antecedência e com calma. A descrição das funções é, na maior parte dos casos, acompanhada de exemplos que ajudam a compreender a funcionalidade pretendida. Se, após uma segunda leitura, ainda tiver dúvidas, deverá esclarecê-las junto do docente. O enunciado pode parecer longo, mas o código de cada uma das funções é simples e pequeno. Sempre que tiver dúvidas relativas a código que já implementou, não coloque o código no fórum. Contacte diretamente o docente.

- Deverá proceder ao desenvolvimento das funções, uma a uma, desde aquela que lhe parecer mais fácil, até à que lhe parecer mais difícil. **Após escrever o código de uma função deverá testar se essa função realmente implementa a funcionalidade pretendida**. Não complique desnecessariamente o código, implementando funciondalidades ou efetuando validações que não são pedidas. Não teste apenas quando já tiver implementado todas as funções. Ao longo do enunciado foram apresentados alguns exemplos que o poderão auxiliar nessa tarefa. Verifique se, após a execução da função, as estruturas de dados são atualizadas de acordo com o esperado.

- Será fornecido um ficheiro com testes para o auxiliar na despistagem de alguns problemas no seu código. Os testes fornecidos não serão exaustivos, por isso deverá efetuar testes adicionais, com valores criados por si, para testar todas as hipóteses possíveis, e assim garantir que o seu código funciona corretamente para todos os casos. Para correr os testes basta executar o comando abaixo. Certifique-se que o

ficheiro de testes se encontra na mesma pasta que o ficheiro `p1.py` correspondente ao seu projeto.

```
python tester_p1.py
```

- O código deverá ser devidamente comentado, mas não em excesso. Para cada função, a respetiva funcionalidade deverá ser comentada na forma de uma cadeia de caracteres de ajuda (doc_string), que poderá ser acedida através do comando `help(<função>)`. O exemplo abaixo apresenta um modelo para doc_strings.

```
def foo():
    """Descrição da funcionalidade da função

    Argumentos:
        x (tipo do argumento): descrição do argumento
        y (tipo do argumento): descrição do argumento

    Devolve:
        tipo de retorno: descrição do retorno

    Levanta:
        tipo da excepção: descrição do(s) caso(s) em que a excepção
                          é levantada
    """
    pass
```

- Na sua implementação do projeto deverá utilizar apenas as funções básicas do Python e das estruturas de dados utilizadas. Não deverá importar/utilizar outras bibliotecas externas. Não deve copiar código da internet ou de outras fontes (e.g. colegas). Essas situações são facilmente detetáveis e serão penalizadas.

- Não pense que o projeto se pode fazer nos últimos dias. Se apenas iniciar o seu trabalho neste período, irá ver a Lei de Murphy em ação: 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 possível.

5 Classificação

A cada função a desenvolver foi atribuída uma cotação que se encontra indicada. Essa cotação será ponderada pelos seguintes fatores:

- **execução correta (70%)**: esta parte da avaliação é efetuada recorrendo a um programa de avaliação automática, pelo que as suas funções deverão cumprir exatamente o que é pedido: nem menos, nem mais.
- **estilo de programação (30%):** esta parte avalia os seguintes aspetos: qualidade, e não quantidade, de comentários, nomes de variáveis bem escolhidos, facilidade de leitura, simplicidade e eficiência do código implementado

Projetos não originais, iguais ou muito semelhantes, serão penalizados com a classificação de zero. O corpo docente da unidade curricular será o único juiz do que se considera ou não copiar num projeto.

6 Entrega

A entrega do projeto será efectuada exclusivamente por via electrónica no Moodle. Deverá submeter o seu projeto, tal como tem feito para os exercícios práticos de programação, até à data definida. **Após esta data não se aceitam projetos seja qual for o pretexto**.

Deverá submeter um único ficheiro com o nome `p1.py`, contendo uma linha com a definição no quadro global de `clientes` como uma lista vazia, seguida da definição da variável `id_reparacao` e da definição das funções solicitadas e, eventualmente, de funções auxiliares que considere necessárias. O ficheiro não deverá conter qualquer outro código. O ficheiro deverá conter, em comentário, na primeira linha, o número e o nome do aluno. No código não devem ser utilizados carateres acentuados ou qualquer caráter que não pertença à tabela ASCII. Em comentários e cadeias de carateres poderá utilizar carateres acentuados. Programas que não cumpram este requisito serão penalizados em dois valores.

Certifique-se de que o seu programa não origina erros quando é carregado. Os programas que tiverem de ser corrigidos para poderem ser avaliados serão penalizados em **três valores**.