Mestrado em Informação e Sistemas Empresariais

Introdução à Programação
2023/2024

Projeto 2

Data enunciado: 12 de fevereiro de 2024
Data Limite de Entrega: 25 de fevereiro 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 tarefas de uma empresa. Este sistema permitirá aos gestores a atribuição de tarefas aos empregados e permite aos empregados a gestão das suas tarefas.

Deverá ter especial cuidado em cumprir todas as especificações dadas no que respeita à funcionalidade de cada função, bem como ao formato em que a informação deverá ser guardada nas estruturas de dados ou devolvida pelas funções. A correção da funcionalidade das funções implementadas será avaliada por um sistema automático, que atribuirá a classificação de 0 caso algum detalhe não esteja conforme o especificado.

Durante este projeto, as datas serão representadas usando o pacote incluído na biblioteca padrão do Python `datetime`. A razão para esta opção é tornar mais fácil a manipulação de datas.

```
import datetime
```

Por forma a criar uma data, basta fazer:

```
data = datetime.date(2024, 1, 1)
```

Alternativamente, é possível fazer:

```
data = datetime.date.fromisoformat('2024-01-01')
```

Para converter a data para uma string no formato `'AAAA-MM-DD'`, basta fazer:

```
data_str = str(data)
```

Por forma a incrementar a data em um dia, basta fazer:

```
data = data + datetime.timedelta(days=1)
```

Note ainda que as datas neste formato podem ser comparadas com `>=`, `<=`, `>`, `<` e `==`.

Durante a realização deste projeto, será necessário processar ficheiros CSV. Um ficheiro CSV (Comma-separated values) é um tipo de ficheiro de texto usado regularmente em Informática para representar dados de forma estruturada. Estes ficheiros são usados por serem facilmente lidos por programas informáticos. Para lidar com ficheiros CSV, aconselha-se a utilização do pacote incluído na biblioteca padrão do Python `csv`. A razão da utilização deste pacote é facilitar o tratamento de casos excepcionais relacionados com ficheiros CSV (por exemplo, quando o delimitador entre colunas aparece nos próprios valores).

```
import csv
```

Para realizar a leitura de cada linha de um ficheiro CSV, basta fazer (a instrução print é meramente ilustrativa):

```
with open('test.csv') as csvfile:
    reader = csv.reader(csvfile)
    for row in reader:
        print(row)
```

Para escrever para um ficheiro CSV, basta fazer:

```
with open('test.csv', 'w') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['Produto', 'Preço', 'Quantidade'])
    writer.writerow(['Atum', 2.50, 10])
```

Os pacotes `csv` e `datetime` deverão ser os **únicos** pacotes importados para a realização deste projeto.

Durante o desenvolvimento de todo o projeto, caso seja passado para uma função **um argumento com um tipo diferente do explicitado** no enunciado, deve ser lançada uma excepção do tipo `ValueError` com a mensagem `'argumento com tipo inválido'`.

2 Modelo de Dados

As duas principais entidades a modelar neste projeto são o **empregado**, que representa um empregado da empresa, e a **tarefa**, que representa uma tarefa que poderá ser atribuída a um empregado. Cada empregado poderá ter várias tarefas nas suas mãos. Neste sistema, a cada tarefa é associado um conjunto de estados, correspondente aos vários estados pela qual a tarefa passou. Cada tarefa e cada empregado é identificado por um número inteiro, designado por **identificador**, que é um número inteiro positivo gerado de forma sequencial, por um processo que descreveremos adiante.

A entidade **empregado** será modelada por um dicionário, contendo os seguintes pares chave/valor:

- `'nif'`: número de identificação fiscal do empregado (string com no máximo 9 caracteres);
- `'nome'`: nome do empregado (string com no máximo 50 caracteres);
- `'data_nasc'`: data de nascimento do empregado (datetime.date);
- `'cargo'`: cargo na empresa (string) que apenas poderá conter um dos seguintes valores:
  - `'EMPREGADO'`
  - `'GESTOR'`
- `'tarefas_id'`: lista contendo os identificadores de todas as tarefas atribuídas ao empregado (lista de números inteiros).

Exemplo

```
{
  'nif': '987654321',
  'nome': 'Manuel Silva',
  'data_nasc': datetime.date(2000, 1, 1),
  'cargo': 'GESTOR',
  'tarefas_id': [1, 3]
}
```

A entidade **tarefa** será modelada por um dicionário, contendo os seguintes pares chave/valor:

- `'descricao'`: descrição textual da tarefa (string com no máximo 256 caracteres);
- `'prazo'`: data máxima para a finalização da tarefa (datetime.date);

- `'estados'`: lista de tuplos contendo a informação relativa a todos os estados pelos quais a tarefa passou; cada tarefa será modelada por um tuplo com os seguintes elementos (por esta ordem):
  - tipo (string) que poderá conter um dos seguintes valores:
    * `'POR ATRIBUIR'`: corresponde ao estado da tarefa quando é criada;
    * `'ATRIBUÍDA'`: corresponde ao estado da tarefa após ser atribuída a um empregado;
    * `'EM PROGRESSO'`: corresponde ao estado atribuído pelo empregado responsável pela tarefa quando inicia o processo de resolução da tarefa;
    * `'FINALIZADA'`: corresponde ao estado atribuído pelo empregado responsável pela tarefa quando esta foi finalizada.
  - data (datetime.date) que corresponde à data de alteração da tarefa para o estado atual;
  - identificador do empregado responsável pela tarefa (número inteiro); caso o estado não seja do tipo `'ATRIBUÍDA'`, o valor deste elemento será `None`.

  Na lista, os estados deverão estar ordenados do estado mais antigo para o estado mais recente.
- `'empregado_id'`: `None` caso a tarefa não esteja atribuída a nenhum empregado; caso contrário é um número inteiro correspondente ao identificador do empregado a que a tarefa foi atribuída.

Exemplo

```
{
    'descricao': 'Resolver bug #31 reportado pela empresa XPTO',
    'prazo': datetime.date(2024, 3, 1),
    'estados': [
        ('POR ATRIBUIR', datetime.date(2024, 1, 5), None),
        ('ATRIBUÍDA', datetime.date(2024, 1, 7), 1)
    ],
    'empregado_id': 1
}
```

Deverá introduzir no início do seu programa as seguintes definições:

```
import csv
import datetime

estado_programa = {
```

```
    'hoje': datetime.date.today(),
    'empregado_id': 1,
    'tarefa_id': 1
}

empregados = {}
tarefas = {}
```

O dicionário `estado_programa` contém os seguintes pares chave/valor:

- `'hoje'`: data atual; esta data é inicializada com a data do dia em que o programa é executado, através da chamada à função `datetime.date.today()`, mas poderá ser alterada através de uma função a implementar, definida mais à frente;
- `'empregado_id'`: identificador a ser atribuído ao próximo empregado a ser criado, inicializado com o valor `1` (número inteiro); cada vez que um identificador é atribuído a um empregado, este valor deverá ser incrementado em uma unidade;
- `'tarefa_id'`: identificador a ser atribuído à próxima tarefa a ser criada, inicializado com o valor `1` (número inteiro); cada vez que um identificador é atribuído a uma tarefa, este valor deverá ser incrementado em uma unidade.

O dicionário `empregados` contém a informação sobre todos os empregados da empresa. A chave é o identificador do empregado e o valor é um dicionário com a informação sobre o empregado, no formato já descrito anteriormente.

Exemplo

```
empregados = {
  1: {
        'nif': '987654321',
        'nome': 'Manuel Silva',
        'data_nasc': datetime.date(2000, 1, 1),
        'cargo': 'GESTOR',
        'tarefas_id': [1, 3]
    },
  2: {
        'nif': '524353422',
        'nome': 'Maria Santos',
        'data_nasc': datetime.date(2010, 3, 11),
        'cargo': 'EMPREGADO',
        'tarefas_id': [2, 4]
    }
}
```

O dicionário `tarefas` contém a informação sobre todas as tarefas. A chave é o identificador da tarefa e o valor é um dicionário com a informação sobre a tarefa, no formato já descrito anteriormente.

Exemplo

```
tarefas = {
  1: {
        'descricao': 'Resolver bug #31 reportado pela empresa XPTO',
        'prazo': datetime.date(2024, 3, 1),
        'estados': [
            ('POR ATRIBUIR', datetime.date(2024, 1, 5), None),
            ('ATRIBUÍDA', datetime.date(2024, 1, 7), 1)
        ],
        'empregado_id': 1
    },
  2: {
        'descricao': 'Concluir venda do produto Y',
        'prazo': datetime.date(2024, 5, 1),
        'estados': [
            ('POR ATRIBUIR', datetime.date(2024, 2, 2), None),
        ],
        'empregado_id': None
    }
}
```

3 Gestão de Tarefas

3.1 Função `cria_tarefa` [2 valores]

`cria_tarefa(descricao, prazo, data_criacao)`

Argumentos:

- `descricao`: string correspondente à descrição da tarefa. Caso a string tenha mais de 256 caracteres, uma excepção do tipo `ValueError` deverá ser lançada com a mensagem `'descrição excede o limite de 256 caracteres'`;
- `prazo`: string, no formato `'AAAA-MM-DD'`, correspondente ao prazo da tarefa. Caso o formato da string esteja incorreto, deverá ser lançada uma excepção do tipo `ValueError` com a mensagem `'prazo com formato inválido'`.

- `data_criacao`: string, no formato `'AAAA-MM-DD'`, correspondente à data considerada para a criação da tarefa. Caso o formato da string esteja incorreto, deverá ser lançada uma excepção do tipo `ValueError` com a mensagem `'data de criação com formato inválido'`.

Valor: a função deverá devolver o identificador da tarefa criada (número inteiro).

Esta função deverá efetuar a criação de uma nova tarefa, com descrição `descricao` e prazo `prazo`, e adicioná-la ao dicionário `tarefas`. A lista de estados da tarefa criada deverá ter um único estado `'POR ATRIBUIR'` com data correspondente a `data_criacao`. Note-se que apesar dos argumentos `prazo` e `data_criacao` da função serem strings, as strings deverão ser transformadas num `datetime.date`. Dado o estado `'POR ATRIBUIR'` da tarefa criada, o campo `empregado_id` da nova tarefa deverá ser preenchido com `None`.

3.2 Função `modifica_estado` [3.5 valores]

```
modifica_estado(tarefa_id, estado, data_modificacao, empregado_id = None)
```

Argumentos:

- `tarefa_id`: identificador da tarefa a modificar (número inteiro);
- `estado`: tipo de estado (string); caso a string não corresponda a um dos valores possíveis para tipos de estados, deverá ser lançada uma excepção do tipo `ValueError` com a mensagem `'estado inválido'`.
- `data_modificacao`: string, no formato `'AAAA-MM-DD'`, correspondente à data considerada para a modificação do estado. Caso o formato da string esteja incorreto, deverá ser lançada uma excepção do tipo `ValueError` com a mensagem `'data de modificação com formato inválido'`;
- `empregado_id`: número inteiro correspondente ao identificador de um empregado; este argumento é opcional.

Esta função deverá modificar o estado atual da tarefa com identificador `tarefa_id`. Caso não exista uma tarefa com identificador `tarefa_id` deverá ser lançada uma excepção do tipo `ValueError` com a mensagem `'tarefa inexistente'`. Para modificar o estado atual da tarefa, deverá ser adicionado um novo estado ao fim da lista de estados da tarefa. Esse estado deverá ter como tipo o argumento `estado` e data `data_modificacao`. Note-se que apesar do argumento `data_modificacao` da função ser uma string, a string deverá ser transformada num `datetime.date`.

Uma tarefa que esteja no estado `'FINALIZADA'` não pode ter o seu estado alterado. Caso `tarefa_id` corresponda ao identificador de uma tarefa finalizada, deverá ser lançada uma excepção do tipo `ValueError` com a mensagem `'tarefa encontra-se finalizada'`.

Caso uma tarefa tenha o seu estado alterado para `'POR ATRIBUIR'`, o campo `empregado_id` da tarefa deverá ser alterado para `None`. Caso essa tarefa estivesse previamente atribuída a um empregado, o identificador da tarefa deverá ser removido da lista de tarefas do empregado em questão.

Caso uma tarefa tenha o seu estado alterado para `'ATRIBUÍDA'`, o campo `empregado_id` da tarefa deverá ser alterado para o argumento `empregado_id` passado para a função `modifica_estado`. O identificador da tarefa deve também ser adicionado à lista de tarefas do novo empregado responsável. Caso a tarefa estivesse previamente atribuída a outro empregado, o identificador da tarefa deverá ser removido da lista de tarefas do anterior responsável. No caso do argumento `estado` corresponder à string `'ATRIBUÍDA'`, então o argumento opcional `empregado_id` deverá necessariamente ser preenchido com um número inteiro. Caso não seja, deverá ser lançada uma excepção do tipo `ValueError` com a mensagem `'identificador do empregado responsável não pode ser vazio'`. Caso o número inteiro não corresponda ao identificador de um empregado, deverá ser lançada uma excepção do tipo `ValueError` com a mensagem `'empregado inexistente'`.

Uma tarefa apenas pode ser alterada para o estado `'EM PROGRESSO'` ou `'FINALIZADA'`, caso já tenha um empregado responsável. Caso contrário, deverá ser lançada uma excepção do tipo `ValueError` com a mensagem `'tarefa não tem empregado responsável'`.

3.3 Função `carrega_tarefas` [2 valores]

```
carrega_tarefas(ficheiro_tarefas, ficheiro_estados)
```

Argumentos:

- `ficheiro_tarefas`: nome do ficheiro CSV com a listagem de tarefas (string).
- `ficheiro_estados`: nome do ficheiro CSV com a listagem de estados de tarefas (string).

Esta função deverá processar dois ficheiros CSV `ficheiro_tarefas` e `ficheiro_estados`, contendo um conjunto de tarefas e de estados de tarefas, respetivamente. A primeira linha de ambos os ficheiros corresponde ao cabeçalho do ficheiro CSV como demonstrado nos exemplos abaixo. Cada linha do ficheiro CSV `ficheiro_tarefas` contém a informação relativa a uma tarefa, no seguinte formato:

- descrição da tarefa (string);
- prazo da tarefa (string, no formato `'AAAA-MM-DD'`);
- data de criação da tarefa (string, no formato `'AAAA-MM-DD'`).

Exemplo

```
descricao,prazo,criacao
Resolver bug #31 reportado pela empresa XPTO,2024-03-01,2024-01-05
Concluir venda do produto Y,2024-05-01,2024-02-02
```

Cada linha do ficheiro CSV `ficheiro_estados` contém a informação relativa ao estado de uma tarefa, no seguinte formato:

- identificador da tarefa (número inteiro);
- tipo de estado (string correspondente a um dos valores possíveis para tipos de estado);
- data de modificação do estado (string, no formato `'AAAA-MM-DD'`).
- identificador do empregado (número inteiro); caso o tipo de estado não seja 'ATRIBUÍDA', o valor no ficheiro CSV corresponderá a uma string vazia.

Exemplo

```
tarefa_id,tipo,data,empregado_id
1,ATRIBUÍDA,2024-01-07,1
1,EM PROGRESSO,2024-01-08,
```

As tarefas no ficheiro `ficheiro_tarefas` encontram-se ordenadas por ordem crescente do identificador númerico, ou seja, a tarefa na linha 2 corresponderá sempre à tarefa com identificador 1, a tarefa na linha 3 corresponderá à tarefa com identificador 2, etc. Os estados no ficheiro `ficheiro_estados` estão ordenados por ordem crescente do identificador númerico da tarefa. Para estados pertencentes à mesma tarefa, é seguida a ordem de inserção na lista `estados` da tarefa em questão. **Note-se que o estado inicial `POR ATRIBUIR` de cada tarefa não está presente no ficheiro `ficheiro_estados` dado que esse estado é criado com a criação da tarefa. No entanto, posteriores estados `POR ATRIBUIR` estão presentes neste ficheiro.**

Esta função deverá chamar a função `cria_tarefa` para cada uma das linhas do ficheiro `ficheiro_tarefas` e a função `modifica_estado` para cada uma das linhas do ficheiro `ficheiro_estados`. Não deve ser feito nenhum tratamento de excepções para os valores lidos do ficheiro CSV para além do tratamento já feito pela função `cria_tarefa` e `modifica_estado`.

Esta função deverá apagar completamente o conteúdo do dicionário `tarefas` e substituí-lo pela informação contida nos ficheiros a ler. O valor `estado_programa['tarefa_id']` também deverá ser reinicializado para 1 antes do processamento das tarefas.

Caso não seja possível abrir um dos ficheiros, a função não deverá mexer no conteúdo do dicionário `tarefas` nem do dicionário `estado_programa`, e deverá gerar uma exceção do tipo `ValueError`, com a mensagem `'erro a abrir o ficheiro'`.

3.4 Função `guarda_tarefas` [1.5 valores]

```
guarda_tarefas(ficheiro_tarefas, ficheiro_estados)
```

Argumentos:

- `ficheiro_tarefas`: nome a dar ao ficheiro CSV para a listagem de tarefas (string).
- `ficheiro_estados`: nome a dar ao ficheiro CSV para a listagem de estados de tarefas (string).

Esta função deverá criar dois ficheiros CSV com nomes `ficheiro_tarefas` e `ficheiro_estados`. Estes ficheiros deverão seguir o mesmo formato apresentado na função `carrega_tarefas`. O ficheiro `ficheiro_tarefas` deverá conter uma listagem de todas as tarefas no dicionário `tarefas` ordenadas por ordem crescente do identificador númerico. O ficheiro ficheiro_estados deverá conter uma listagem de todos os estados de cada tarefa ordenados por ordem crescente do identificador númerico da tarefa. Para estados pertencentes à mesma tarefa, deve ser seguida a ordem de inserção na lista `estados` da tarefa em questão.

3.5 Função `gera_resumo_diario` [3.5 valores]

```
gera_resumo_diario(data)
```

Argumentos:

- `data`: data para a qual o resumo é gerado (string, no formato `'AAAA-MM-DD'`); Caso o formato da string esteja incorreto, deverá ser lançada uma excepção do tipo `ValueError` com a mensagem `'data de modificação com formato inválido'`.

Valor: a função deverá devolver um tuplo contendo a informação do resumo para o dia considerado (tuplo).

Esta função deverá gerar e devolver um resumo das tarefas do dia `data`, o qual deverá ser modelado por um tuplo com os seguintes elementos (por esta ordem):

- data do resumo, correspondente a `data` (string, no formato `'AAAA-MM-DD'`);
- data em que o resumo foi gerado, correspondente a `estado_programa['hoje']` (string, no formato `'AAAA-MM-DD'`); note que terá que transformar `estado_programa['hoje']` numa string;
- uma lista de tuplos contendo informação relativa às várias alterações de estado de tarefas realizadas no dia `data`. Cada tuplo deverá conter os seguintes elementos (por esta ordem):

  - descrição da tarefa associada ao estado (string com no máximo 256 caracteres);
  - tipo do estado (string com um dos possíveis valores para tipos de estados);
  - nome do empregado responsável pela tarefa no momento em que a alteração de estado foi realizada (string com, no máximo, 50 caracteres). Note que o empregado responsável pela tarefa no momento em que a alteração foi realizada não é necessariamente o empregado que é atualmente responsável pela tarefa (a tarefa pode ter sido atribuída a outro empregado depois da alteração de estado ou pode ter sido removida a atribuição). Caso não exista nenhum empregado responsável, a string deverá ser vazia ''.

- uma lista de tuplos contendo informação relativa à lista de tarefas em atraso, ou seja, a lista de todas as tarefas não finalizadas cuja data `data` ultrapassa o `prazo` da tarefa; Cada tuplo deverá conter os seguintes elementos (por esta ordem):
  - descrição da tarefa associada ao estado (string com no máximo 256 caracteres);
  - prazo da tarefa (string, no formato `'AAAA-MM-DD'`);
  - o estado da tarefa na data do resumo (string com um dos possíveis valores para tipos de estados); note que o estado da tarefa na data do resumo pode não corresponder ao estado atual da tarefa (isto é, ao último estado).
  - nome do empregado responsável pela tarefa na data do resumo (string com, no máximo, 50 caracteres). Note que o empregado responsável pela tarefa na data do resumo não é necessariamente o empregado que é atualmente responsável pela tarefa (a tarefa pode ter sido atribuída a outro empregado depois da data do resumo ou pode ter sido removida a atribuição). Caso não exista nenhum empregado responsável, a string deverá ser vazia ''.

Recomenda-se a criação de duas funções auxiliares que permitam:

- obter o empregado responsável por uma dada tarefa numa data específica;
- obter o estado de uma dada tarefa numa data específica.

3.6 Função `imprime_resumo` [1 valor]

```
imprime_resumo(resumo)
```

Argumentos:

- `resumo`: tuplo correspondente a um resumo diário gerado pela função `gera_resumo_diario` (tuplo). Não é necessário validar o argumento `resumo`.

Esta função deverá imprimir a informação contida num resumo diário, no seguinte formato (cada item numa linha):

- o caracter `'-'` repetido 50 vezes;
- a string `'DATA DO RESUMO:'`; um espaço; a data do resumo, no formato `'AAAA-MM-DD'`;
- a string `'DATA DE CRIAÇÃO:'`; um espaço; a data em que o resumo foi gerado, no formato `'AAAA-MM-DD'`;
- o caracter `'-'` repetido 50 vezes;
- 13 espaços; a string `'** ESTADO DAS TAREFAS **'`;
- o caracter `'-'` repetido 50 vezes;
- para cada alteração de estado, colocar numa linha: a descrição da tarefa; o tipo do estado; o nome do empregado responsável pela tarefa (caso não exista um empregado responsável, não deverá ser impressa nenhuma linha); uma linha em branco;
- o caracter `'-'` repetido 50 vezes;
- 13 espaços; a string `'** TAREFAS EM ATRASO **'`;
- o caracter `'-'` repetido 50 vezes;
- para cada tarefa em atraso, colocar numa linha: a descrição da tarefa; a string `'PRAZO:'`, seguida de um espaço e depois o prazo da tarefa; o estado da tarefa na data do resumo; o nome do empregado responsável pela tarefa na data do resumo (caso não exista um empregado responsável, não deverá ser impressa nenhuma linha); uma linha em branco;
- o caracter `'-'` repetido 50 vezes.

Exemplo

```
--------------------------------------------------
DATA DO RESUMO: 2024-04-01
DATA DE CRIAÇÃO: 2024-04-02
--------------------------------------------------
             ** ESTADO DAS TAREFAS **
--------------------------------------------------
Resolver bug #31 reportado pela empresa XPTO
EM PROGRESSO
Manuel Silva

Concluir venda do produto Z
POR ATRIBUIR
```

```
------------------------------------------------
             ** TAREFAS EM ATRASO **
------------------------------------------------
Resolver bug #31 reportado pela empresa XPTO
PRAZO: 2024-03-01
EM PROGRESSO
Manuel Silva

------------------------------------------------
```

4 Gestão de Empregados

4.1 Função `cria_empregado` [1 valor]

`cria_empregado(nif, nome, data_nasc, cargo)`

Argumentos:

- `nif`: número de identificação fiscal do empregado (string com no máximo 9 caracteres);
- `nome`: nome do empregado (string com no máximo 50 caracteres);
- `data_nasc`: data de nascimento do empregado (string, no formato `'AAAA-MM-DD'`);
- `cargo`: cargo na empresa (string com valor `'EMPREGADO'` ou `'GESTOR'`).

Valor: a função deverá devolver o identificador do empregado criado (número inteiro).

Esta função deverá criar um novo empregado, com NIF `nif`, com nome `nome`, com data de nascimento `data_nasc` e cargo `cargo`, devolvendo o respetivo identificador. Deverá também adicionar o empregado ao dicionário `empregados`. Note que a string `data_nasc` deverá ser transformada numa `datetime.date`. Inicialmente, a lista de tarefas do empregado é vazia `[]`. Deverá ser lançada uma excepção do tipo `ValueError` nas seguintes situações:

- caso a string `nif` tenha mais de 9 caracteres (a mensagem deverá ser `'nif não pode exceder os 9 caracteres'`);
- caso a string `nome` tenha mais de 50 caracteres (a mensagem deverá ser `'nome não pode exceder os 50 caracteres'`);

- caso a string `data_nasc` não tenha o formato indicado (a mensagem deverá ser `'data de nascimento com formato inválido'`);
- caso a string `cargo` não corresponda a um dos valores indicados (a mensagem deverá ser `'cargo inválido'`).

4.2 Função `carrega_empregados` [2 valores]

```
carrega_empregados(nome_ficheiro)
```

Argumentos:

- `nome_ficheiro`: nome do ficheiro CSV com a listagem de empregados (string).

Esta função deverá processar um ficheiro CSV com nome `nome_ficheiro` contendo uma listagem de empregados. A primeira linha do ficheiro corresponde ao cabeçalho do ficheiro CSV como demonstrado no exemplo abaixo. Cada linha do ficheiro CSV `nome_ficheiro` contém a informação relativa a um empregado no seguinte formato:

- NIF (string);
- nome (string);
- data de nascimento (string, no formato 'AAAA-MM-DD');
- cargo (string).

Exemplo

```
nif,nome,data_nasc,cargo
987654321,Manuel Silva,2000-01-01,GESTOR
524353422,Maria Santos,2010-03-11,EMPREGADO
```

Os empregados no ficheiro `nome_ficheiro` encontram-se ordenados por ordem crescente do identificador númerico, ou seja, o empregado na linha 2 corresponderá sempre ao empregado com identificador 1, o empregado na linha 3 corresponderá ao empregado com identificador 2, etc.

Esta função deverá apagar completamente o conteúdo do dicionário `empregados` e substituí-lo pela informação contida no ficheiro a ler. Para isso, deverá ser chamada a função `cria_empregado` para cada uma das linhas do ficheiro CSV `nome_ficheiro`. O valor `estado_programa['empregado_id']` também deverá ser reinicializado para 1 antes

do processamento dos empregados. Não deve ser feito nenhum tratamento de excepções para os valores lidos do ficheiro CSV para além do tratamento já feito pela função `cria_empregado`.

Caso não seja possível abrir o ficheiro, a função não deverá mexer no conteúdo do dicionário `empregados` nem do dicionário `estado_programa`, e deverá gerar uma exceção do tipo `ValueError`, com a mensagem `'erro a abrir o ficheiro'`.

4.3 Função `guarda_empregados` [1.5 valores]

```
guarda_empregados(nome_ficheiro)
```

Argumentos:

- `nome_ficheiro`: nome a dar ao ficheiro CSV para a listagem de empregados (string).

Esta função deverá criar um ficheiro CSV com nomes `nome_ficheiro`. Este ficheiro deverá seguir o mesmo formato apresentado na função `carrega_empregados`. O ficheiro `nome_ficheiro` deverá conter uma listagem de todos os empregados no dicionário `empregados` ordenados por ordem crescente do identificador númerico.

4.4 Função `imprime_tarefas` [1 valor]

```
imprime_tarefas(empregado_id)
```

Argumentos:

- `empregado_id`: identificador do empregado (número inteiro)

Esta função deverá imprimir informação sobre todas as tarefas não finalizadas atribuídas ao empregado com identificador `empregado_id`. Esta informação deverá ser imprimida no seguinte formato (cada item numa linha):

- o caracter `'-'` repetido 50 vezes;
- a string `'NOME DO EMPREGADO:'`; um espaço; nome do empregado com identificador `empregado_id`;
- o caracter `'-'` repetido 50 vezes;
- 13 espaços; a string `'** TAREFAS A REALIZAR **'`;
- o caracter `'-'` repetido 50 vezes;

- para cada tarefa não finalizada do empregado com identificador `empregado_id`, colocar numa linha: a descrição da tarefa; o tipo do estado atual da tarefa; o prazo da tarefa, no formato 'AAAA-MM-DD'; uma linha vazia;
- o caracter `'-'` repetido 50 vezes.

Caso não exista um empregado com identificador `empregado_id` deverá ser lançada uma excepção do tipo `ValueError` com a mensagem `'empregado inexistente'`.

Exemplo

```
--------------------------------------------------
NOME DO EMPREGADO: Manuel Silva
--------------------------------------------------
             ** TAREFAS A REALIZAR **
--------------------------------------------------
Resolver bug #31 reportado pela empresa XPTO
ATRIBUÍDA
2024-03-01

--------------------------------------------------
```

5 Outras Funcionalidades

5.1 Função `inicia_dia` [1 valor]

`inicia_dia(data = None)`

Argumentos:

- `data` nova data atual (string, no formato `'AAAA-MM-DD'`); este argumento é opcional. Caso o argumento seja uma string e o formato da string esteja incorreto, deverá ser lançada uma excepção do tipo `ValueError` com a mensagem `'data com formato inválido'`.

Valor: a função deverá devolver um resumo diário da nova data atual (tuplo).
Esta função corresponde ao processamento que é efetuado de madrugada, no início de um novo dia. Em primeiro lugar, a data atual, guardada em `estado['hoje']` deverá ser atualizada para a data dada por `data` (após transformação numa `datetime.date`). Caso o argumento `data` não seja fornecido, a data dada por `estado['hoje']` será incrementada em 1 dia. Para calcular o tuplo de retorno, esta função deverá chamar a função `gera_resumo_diario`.

6 Aspetos a considerar

- Deverá ler o projeto com antecedência e com calma. 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 não é complexo e é relativamente pequeno. Sempre que tiver dúvidas relativas a código que já implementou, não coloque o código no forum. 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 `p2.py` correspondente ao seu projeto.

  ```
  python tester_p2.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, com excepção dos pacotes `csv` e `datetime`. 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.

7 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

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 `p2.py`, contendo, no início, as definições indicadas na Secção 2. Poderá criar outras funções suas, para além das funções pedidas, mas **o ficheiro não deverá conter qualquer outro código fora das funções**. O ficheiro deverá conter, em comentário, na primeira linha, o número e o nome do aluno. No código não devem ser utilizados caracteres acentuados ou qualquer caracter que não pertença à tabela ASCII. Em comentários e cadeias de caracteres poderá utilizar caracteres acentuados. Programas que não cumpram este requisito serão penalizados em 2 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 3 valores.

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.