Fundação CECIERJ - **Vice-Presidência de Educação Superior a Distância**

Curso de Tecnologia em Sistemas de Computação
Disciplina Fundamentos de Programação

AD2 – 1° semestre de 2019

IMPORTANTE

- As respostas (programas) deverão ser entregues pela plataforma em um arquivo ZIP contendo todos os arquivos de código fonte (extensão ".py") necessários para que os programas sejam testados. Respostas entregues fora do formato especificado, por exemplo, em arquivos com extensão ".pdf", ".doc" ou outras, não serão corrigidas.
- Serão aceitos apenas soluções escritas na linguagem Python 3. Programas com erro de intepretação não serão corrigidos. Evite problemas utilizando tanto a versão da linguagem de programação (Python 3.X) quanto a IDE (PyCharm) indicadas na Aula 1.
- Quando o enunciado de uma questão inclui especificação de formato de entrada e saída, tal especificação deve ser seguida à risca pelo programa entregue. Atender ao enunciado faz parte da avaliação e da composição da nota final.
- Faça uso de boas práticas de programação, em especial, na escolha de identificadores de variáveis, subprogramas e comentários no código.
- As respostas deverão ser entregues pela atividade "Entrega de AD2" antes da data final de entrega estabelecida no calendário de entrega de ADs. Não serão aceitas entregas tardias ou substituição de respostas após término do prazo.
- As ADs são um mecanismo de avaliação individual. As soluções podem ser buscadas por grupos de alunos, mas a redação final de cada prova tem que ser individual. Respostas plagiadas não serão corrigidas.
- Os exemplos fornecidos nos enunciados das questões correspondem a casos específicos apontados para fins de ilustração e não correspondem ao universo completo de entradas possíveis especificado no enunciado. Os programas entregues devem ser elaborados considerando qualquer caso que siga a especificação e não apenas os exemplos dados. Essa é a prática adotada tanto na elaboração das listas exercícios desta disciplina quanto no mercado de trabalho.

1ª Questão (2,0 pontos)

Faça um programa que leia strings do teclado até que uma string vazia seja digitada. Construa e mostre a lista de todas as strings que são numéricas, ou seja, que seja composta de números inteiros ou de ponto flutuante, considerando o uso de "-" e "+" no início da string.

Entrada

O formato de entrada deve ser deduzido dos exemplos.

Saída

O formato de saída deve ser deduzido dos exemplos.

Exemplos

Entrada Padrão
<linha deixada em branco>
Saída Padrão
`Lista de Números Válidos Lidos = []`

Entrada Padrão
`1+` `+1` `-1` `1-` `+1-` `-1.1+` `1` *<linha deixada em branco>*
Saída Padrão
`Lista de Números Válidos Lidos = ['+1', '-1', '1']`

Entrada Padrão
`1000` `abacaxi` `uva` `15.19` `-1` `+1.45` `123.456-` `12.34+` *<linha deixada em branco>*
Saída Padrão
`Lista de Números Válidos Lidos = ['1000', '15.19', '-1', '+1.45']`

2ª Questão (2,0 pontos)

Utilizando subprogramação, faça um programa que leia o nome de um arquivo texto contendo produtos de um supermercado. Construa um vetor de tuplas, onde cada tupla representa um produto, com campos: (1) “Código”, uma string; (2) “Quantidade”, um inteiro; e (3) “Preço Unitário”, um número de ponto flutuante. Mostre o conteúdo do vetor lido. Mostre o conteúdo do vetor após ordená-lo pelo campo “Código”. Em seguida, mostre o conteúdo do vetor após ordená-lo pelo campo “Quantidade”; e, finalmente, mostre o conteúdo do vetor após ordená-lo pelo campo “Preço Unitário”. Neste caso, siga o formato apresentado no Teste 2, dado a seguir. Caso o arquivo esteja vazio, conforme Teste 1, escreva a mensagem “Arquivo: *nomeDoArquivo* está vazio!!!”.

Entrada

O formato de entrada deve ser deduzido dos exemplos.

Saída

O formato de saída deve ser deduzido dos exemplos.

Exemplos

Entrada	
Entrada Padrão	Arquivo `prods1.txt`
`prods1.txt`	*<arquivo em branco>*
Saída Padrão	
`prods1.txt está vazio!!!`	

Entrada	
Entrada Padrão	Arquivo `prods2.txt`
`prods2.txt`	`beringela 10 1.99` `arroz 5 4.99` `peixe 2 9.99` `abacaxi 100 3.99`
Saída Padrão	

```
----- Listagem dos Produtos Lida do Arquivo: prods2.txt -----
('beringela', 10, 1.99)
('arroz', 5, 4.99)
('peixe', 2, 9.99)
('abacaxi', 100, 3.99)
------------------------------------------------------------------------
----- Listagem dos Produtos Ordenados Pelo Campo: 1 -----
('abacaxi', 100, 3.99)
('arroz', 5, 4.99)
('beringela', 10, 1.99)
('peixe', 2, 9.99)
------------------------------------------------------------------------
```

```
----- Listagem dos Produtos Ordenados Pelo Campo: 2 -----
('peixe', 2, 9.99)
('arroz', 5, 4.99)
('beringela', 10, 1.99)
('abacaxi', 100, 3.99)
-------------------------------------------------------------------------
----- Listagem dos Produtos Ordenados Pelo Campo: 3 -----
('beringela', 10, 1.99)
('abacaxi', 100, 3.99)
('arroz', 5, 4.99)
('peixe', 2, 9.99)
```

3ª Questão (2,0 pontos)

Utilizando subprogramação, faça um programa que produza um dicionário (`dict`) com contagem de ocorrências de todas as palavras que contenham um conjunto (`set`) de caracteres lido do usuário. Inicialmente seu programa deve ler da entrada padrão o nome do arquivo a ser processado, em seguida deve ler, da entrada padrão, uma string da qual será construído o conjunto de caracteres. Processe o arquivo e gere o dicionário solicitado. Ao final, mostre, ordenado pela chave, o conteúdo do dicionário de ocorrências contendo o conjunto de caracteres solicitado. Suponha que cada linha do arquivo possua várias palavras.

Entrada

O formato de entrada deve ser deduzido dos exemplos.

Saída

O formato de saída deve ser deduzido dos exemplos.

Exemplos

Entrada	
Entrada Padrão	Arquivo `teste1.txt`
`teste1.txt` `fdjjdsfjksdsdfdjj`	*<arquivo em branco>*
Saída Padrão	
`O dicionário está vazio!!!`	

Entrada	
Entrada Padrão	Arquivo `carta.txt`
`carta.txt` `anna`	`O tempo perguntou pro tempo` `quanto tempo o tempo tinha` `o tempo respondeu pro tempo` `que o tempo tinha o mesmo tempo` `que o tempo tem`
Saída Padrão	
`Listagem do dicionário das palavras contendo {'n', 'a'}:` `quanto ocorreu 1 vez` `tinha ocorreu 2 vezes`	

Entrada	
Entrada Padrão	Arquivo `carta.txt`
`carta.txt` `ooooooooooo`	`O tempo perguntou pro tempo` `quanto tempo o tempo tinha` `o tempo respondeu pro tempo` `que o tempo tinha o mesmo tempo` `que o tempo tem`
Saída Padrão	
`Listagem do dicionário das palavras contendo {'o'}:` `mesmo ocorreu 1 vez` `o ocorreu 5 vezes` `perguntou ocorreu 1 vez` `pro ocorreu 2 vezes` `quanto ocorreu 1 vez` `respondeu ocorreu 1 vez` `tempo ocorreu 9 vezes`	

4ª Questão (2,0 pontos)

Esta questão é dividia em duas partes. Na primeira parte seu programa deve receber da entrada padrão uma coleção de valores inteiros, todos informados na mesma linha, ordená-los de forma crescente na memória utilizando um dos algoritmos vistos em aula e escrever na saída padrão o resultado ordenado. Visto que essa parte da questão é praticamente igual ao que é apresentado nos slides, tal implementação deve ser entregue, mas não vale pontos. Entretanto, seu trabalho não eserá em vão, pois a primeira parte da solução servirá de ponto de partida para resolver a segunda parte da questão. Na segunda parte da questão você deverá pegar a coleção informada (isto é, sem ordenação) e escreve-la em um arquivo binário chamado “colecao.bin”. Em seguida, a coleção salva no arquivo deverá ser ordenada sobre o próprio arquivo. Ou seja, será como assumir que tal arquivo é tão grande que não poderá ser lido de uma só vez para a memória principal a fim de realizar a ordenação em uma estrutura de lista (`list`). Logo, a ordenação deverá ser feita dentro do próprio arquivo utilizando o arquivo de forma análoga a uma lista alocada na memória principal. Após a ordenação o seu programa

deverá percorrer todo o arquivo, do início até o fim, e mostrar os valores na saída padrão. O resultado esperado é a impressão idêntica das sequências de valores obtidas na primeira e na segunda parte da questão.

Entrada

O formato de entrada deve ser deduzido do exemplo.

Saída

O formato de saída deve ser deduzido do exemplo.

Exemplo

Entrada Padrão
`16 -25 34 2 199 45 67 90`

Saída Padrão
`Ordenação na memória principal:` `-25 2 16 34 45 67 90 199` `Ordenação no arquivo:` `-25 2 16 34 45 67 90 199`

Observações

Se a questão for resolvida considerando arquivos “colecao.bin” textuais então a nota atribuída para a mesma será 0 (zero), mesmo que a solução esteja correta no contexto de arquivos texto.

É proibida a chamada a implementações prontas de métodos de ordenação disponíveis na API do Python 3. Será atribuída nota 0 (zero) para soluções que façam tais chamadas.

Dicas

Os tamanhos (quantidade de bytes) assumidos para formatos nativos de valores inteiros e de valores em ponto flutuante lidos ou escritos de arquivos binários podem variar de plataforma para plataforma. Ou seja, podem ocorrer problemas de compatibilidade entre programas que rodam perfeitamente em computadores que assumem determinados tamanhos para tipos primitivos, mas que não rodam corretamente em computadores que assumem outros tamanhos para o mesmo tipo. Para forçar a leitura e escrita assumindo os tamanhos padrão (standard) que são indicados na Aula 12 e ficar livre de problemas de compatibilidade, inclua o símbolo “=” na frente do formato indicado nas funções `.pack` e `.unpack` de `struct`. Por exemplo, `struct.unpack("i", bloco)` converte o bloco de bytes em um valor inteiro, mas o tamanho do bloco é dependente da plataforma (não é necessariamente de 4 bytes), enquanto que `struct.unpack("=i", bloco)` converte blocos de 4 bytes em valores inteiros, independentemente da plataforma.

5ª Questão (2,0 pontos)

Um baralho lusófono de 52 cartas é composto de 13 cartas de cada um dos 4 naipes franceses. São eles: paus (♣), ouros (♦), copas (♥) e espadas (♠). Por simplicidade, nesta questão você deve assumir que os naipes são identificados pela primeira letra maiúscula de seus nomes em português. Ou seja: P, O, C e E, respectivamente. As 13 cartas de cada naipe são representadas por letras ou números, sendo ordenadas e nomeadas da seguinte forma:

Símbolo	A	2	3	4	5	6	7	8	9	10	J	Q	K
Nome	Ás	Dois	Três	Quatro	Cinco	Seis	Sete	Oito	Nove	Dez	Valete	Dama	Rei

Implemente um programa que: (i) receba como entrada um arquivo texto chamado "cartas.txt" contendo em cada linha a identificação de uma das 52 cartas do baralho lusófono; (ii) leia a coleção para a memória principal; (iii) ordene as cartas de modo crescente utilizando como primeiro critério o naipe (respeitando a ordem informada acima) e como segundo critério o símbolo da carta (respeitando a ordem informada acima); e (iv) escreva no arquivo "cartas_ordenadas.txt" o resultado da ordenação, colocando uma carta por linha.

Entrada

O formato do arquivo de entrada deve ser deduzido do exemplo. Os três pontos (...) representam sequências de cartas omitidas por questões de espaço.

Saída

O formato do arquivo de saída deve ser deduzido do exemplo. Os três pontos (...) representam sequências de cartas omitidas por questões de espaço.

Exemplo

Arquivo "cartas.txt" de Entrada

```
K E
A P
3 C
Q C
4 E
10 P
J O
...
9 P
2 O
A O
7 E
J C
5 P
8 O
```

Arquivo “cartas_ordenadas.txt” de Saída

```
A P
2 P
3 P
4 P
...
10 P
J P
Q P
K P
A O
2 O
3 O
4 O
...
10 O
J O
Q O
K O
A C
2 C
3 C
4 C
...
10 C
J C
Q C
K C
A E
2 E
3 E
4 E
...
10 E
J E
Q E
K E
```

Observações

É proibida a chamada a implementações prontas de métodos de ordenação disponíveis na API do Python 3. Será atribuída nota 0 (zero) para soluções que façam tais chamadas.

Boa Avaliação!