INSTITUTO FEDERAL
Catarinense
Campus Blumenau

Algoritmos

Bacharelado em Ciência da Computação
Prof. Paulo César Rodacki Gomes

Lista de exercícios -Alg-09
Exercícios sobre Arquivos em Python

Estes exercícios devem ser entregues no Google Classroom. Para cada um dos exercícios, crie um arquivo fonte Python com o respectivo nome de acordo com a seguinte regra: `SUASINICIAIS-Alg-09-Ex-num.py`. Por exemplo, se o professor resolvesse o exercício número 3, o nome do arquivo seria `PCRG-AER-09-Ex-03.py`.

Introdução

Os arquivos nos permitem trabalhar com os dados, sem precisar inseri-los cada vez que nosso programa é executado. Os arquivos também nos permitem armazenar os resultados do nosso programa de uma forma mais permanente. Esses recursos são freqüentemente usados na criação de programas maiores. Ao completar os exercícios desta lista, você vai aprender a:

- Abrir um arquivo para leitura e/ou gravação
- Ler dados de um arquivo
- Gravar dados em um novo arquivo
- Usar os valores fornecidos ao programa como parâmetros de linha de comando
- Detectar e tratar erros, tais como tentar abrir um arquivo que não existe
- Detectar e tratar erros que não estão necessariamente relacionados a arquivos

Alguns dos exercícios nesta lista envolvem a leitura de arquivos existentes, como uma lista de palavras, nomes, etc. Você pode baixar esses arquivos no Google Classroom.

Questões:

1. **Passagem de argumentos para o programa em linha de comando**. Neste exercício, vamos aprender a passar argumentos para programas Python em linha de comando. Para tanto, siga os seguintes passos:

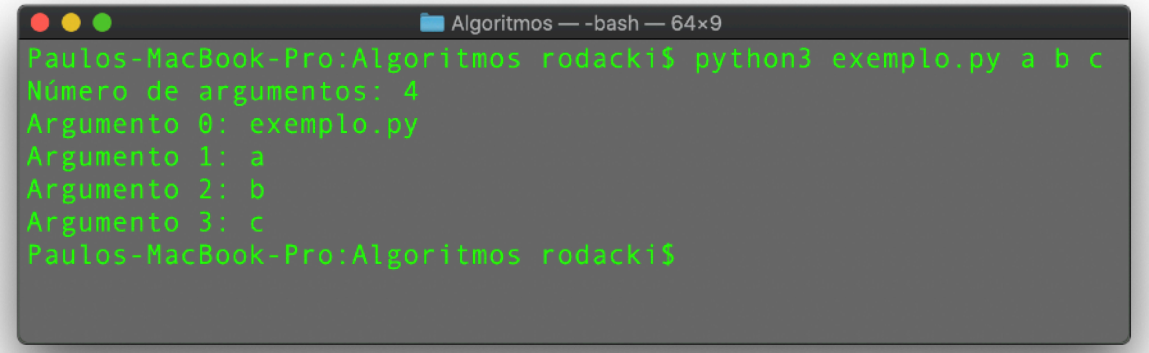
   a) Escreva o seguinte código Python, que lê e exibe os argumentos passados em linha de comando (no meu caso, eu chamei este programa de exemplo.py):

   ```python
   import sys

   if __name__ == "__main__":
       print(f"Número de argumentos: {len(sys.argv)}")
       for i, arg in enumerate(sys.argv):
           print(f"Argumento {i}: {arg}")
   ```

   b) Faça um teste de execução do seu programa em linha de comando. A imagem abaixo mostra um teste realizado pelo professor, passando argumentos **a**, **b**, e **c** com o seguinte comando: `python3 exemplo.py a b c`

Exemplo de saída do programa, executado em linha de comando:



2. **Exibir o cabeçalho de um arquivo texto.** Os sistemas operacionais baseados em Unix normalmente possuem um comando chamado `head`, que exibe as 10 primeiras linhas de um arquivo cujo nome é passado como argumento em linha de comando. Escreva um programa Python que apresente o mesmo comportamento. O programa deve exibir uma mensagem de erro caso não exista o arquivo ou caso não tenha sido passado nenhum argumento para o programa em linha de comando.

3. **Exibir a "cauda" de um arquivo texto.** Os sistemas operacionais baseados em Unix normalmente também possuem um comando chamado `tail`, que exibe as 10 últimas linhas de um arquivo cujo nome é passado como argumento em linha de comando. Escreva um programa Python que apresente o mesmo comportamento. O programa deve exibir uma mensagem de erro caso não exista o arquivo ou caso não tenha sido passado nenhum argumento para o programa em linha de comando.

   Dica: Existem algumas abordagens diferentes para resolver este problema. Uma opção é copiar todo o conteúdo do arquivo para uma lista, e exibir os seus 10 últimos elementos. Outra opção consiste em ler duas vezes o conteúdo do arquivo, a primeira para contar a quantidade de linhas, e a segunda para exibir as últimas 10 linhas. Porém, ambas as opções não são boas para arquivos grandes. Existe uma terceira solução que requer apenas uma leitura do arquivo e o armazenamento de 10 linhas apenas. Se quiser topar um desafio extra, você pode tentar desenvolver esta última opção.

4. **Concatenar múltiplos arquivos.** Os sistemas operacionais baseados em Unix normalmente possuem um comando chamado `cat`, que é a abreviação de “concatenate". Sua finalidade é concatenar e exibir um ou mais arquivos passados como argumentos. Os arquivos são exibidos na mesma ordem em que são passados como argumentos. Note que este coando não modifica os arquivos originais e não cria um novo arquivo, ele apenas exibe o conteúdo dos arquivos na sequencia. Escreva um programa Python que apresente o mesmo comportamento. O programa deve exibir uma mensagem de erro caso algum arquivo não puder ser lido, e então passa para a leitura do próximo arquivo. O programa também deve exibir mensagem de erro caso não tenha sido passado nenhum argumento para o programa em linha de comando.

5. **Numerar as linhas de um arquivo.** Escreva um programa Python que adicione números de linhas em um arquivo. O programa deve receber do usuário os nomes do arquivo de entrada e do arquivo de saída (que será um novo arquivo criado pelo programa). Cada linha do arquivo de saída deve começar com leu número, seguido de dois pontos, um espaço em branco, seguido do conteúdo da linha do arquivo original.

6. **Encontrar a palavra mais longa de um arquivo.** Neste exercício você deve desenvolver um programa Python para identificar a(s) palavra(s) mais longa(s) de um arquivo. Seu programa deve exibir uma mensagem que inclua o tamanho da palavra mais longa, juntamente com

todas as palavras daquele tamanho que existem no arquivo. Trate cada grupo de caracteres sem espaço como uma palavra, mesmo se esse grupo contiver números ou sinais de pontuação. **Dica**: vai ser bem mais fácil desenvolver uma solução se você usar estruturas do Python tais como listas, conjuntos e dicionários.

7. **Frequência de letras.** Uma técnica que pode ser usada para quebrar formas simples de encriptação é a análise de frequência. Essa análise examina o texto encriptado para determinar quais caracteres são mais frequentes. Depois ela tenta mapear as letras mais comuns na língua portuguesa, tais como A, R, S, etc.. para os caracteres mais frequentes do texto encriptado.

   Escreva um programa Python que faz a primeira parte deste processo, determinando e exibindo a frequência (percentual) de ocorrência de todas as letras em um arquivo. Ignore espaços, números e sinais de pontuação. Seu programa não deve diferenciar letras maiúsculas e minúsculas (portento deve tratar **A** e **a** como sendo a mesma letra). O usuário deve fornecer o nome do arquivo como argumento em linha de comando. O programa deve exibir uma mensagem de erro caso não consiga ler o arquivo ou caso não tenha sido passado nenhum argumento para o programa em linha de comando.

8. **Palavras que ocorrem com maior frequência.** Escreva um programa que exiba a palavra (ou palavras) que ocorrem com mais frequência em um arquivo. Seu programa deve começar lendo o nome do arquivo do usuário. Em seguida, ele deve encontrar a (s) palavra (s) dividindo cada linha do arquivo em cada espaço. Finalmente, quaisquer sinais de pontuação à esquerda ou à direita devem ser removidos de cada palavra. Além disso, seu programa deve ignorar a capitalização. Como resultado, **porta**, **porta!**, **Porta** e **PoRTA** devem ser tratados como a mesma palavra, por exemplo. Você provavelmente verá que sua solução para o exercício 8 da lista 6 (sobre strings) é útil para resolver este problema.

9. **Remoção de comentários.** Python usa o caractere # para marcar o início de um comentário. O comentário termina no final da linha que contém o caractere #. Neste exercício, você deve criar um programa que remove todos os comentários de um arquivo-fonte Python. Verifique cada linha no arquivo para determinar se um caractere # está presente. Se for, então seu programa deve remover todos os caracteres do caractere # até o final da linha (ignoraremos a situação em que o caractere de comentário ocorre dentro de uma string). Salve o arquivo modificado usando um novo nome que será inserido pelo usuário. O usuário também irá inserir o nome do arquivo de entrada. Certifique-se de que uma mensagem de erro apropriada seja exibida se for encontrado um problema ao acessar os arquivos.

10. **Um livro sem “e”**. O romance “Gadsby”, de Ernest Vincent Wright, tem mais de 50.000 palavras. Embora 50.000 palavras não sejam normalmente notáveis para um romance, neste caso é porque nenhuma das palavras do livro usa a letra “e”. Isso é particularmente notável quando se considera que "e" é a letra mais comum em inglês. Escreva um programa que leia uma lista de palavras de um arquivo e determine qual proporção das palavras usam cada letra do alfabeto. Exibe o resultado para todas as 26 letras. Inclua uma mensagem adicional identificando a letra usada na menor quantidade de palavras. Seu programa deve ignorar quaisquer sinais de pontuação e deve tratar letras maiúsculas e minúsculas como equivalentes.

11. **Corretor ortográfico**. Um corretor ortográfico pode ser uma ferramenta útil para pessoas que têm dificuldade em escrever palavras corretamente. Neste exercício, você escreverá um programa que lê um arquivo e exibe todas as palavras com erros ortográficos contidas nele. Palavras com erros ortográficos serão identificadas comparando cada palavra do arquivo com uma lista de palavras conhecidas do Português. Quaisquer palavras no arquivo do usuário que não aparecerem na lista de palavras conhecidas serão relatadas como erros ortográficos. O usuário deve fornecer como parâmetro de linha de comando o nome do arquivo para verificar se há erros de ortografia. Seu programa deve exibir uma mensagem de erro apropriada se o parâmetro da linha de comando estiver ausente. Uma mensagem de erro também deve ser exibida se o seu programa não conseguir abrir o arquivo do usuário. Use a solução do exercício 8 da lista 6 ao criar a solução para este exercício, de forma que palavras seguidas

de vírgula, ponto ou outro sinal de pontuação não sejam relatadas como erros ortográficos. Ignore a capitalização das palavras ao verificar a ortografia. No google classroom há um arquivo texto com todas as palavras do Português Brasileiro para você usar neste exercício.

Dica: Embora você possa carregar em uma lista todas as palavras em português do conjunto de dados de palavras, a pesquisa em uma lista é lenta se você usar o operador **in** do Python. É muito mais rápido verificar se uma chave está presente em um dicionário ou se um valor está presente em um conjunto. Se você usar um dicionário, as palavras serão as chaves. Os valores podem ser o inteiro 0 (ou qualquer outro valor) porque os valores nunca serão usados.