

## Exercícios sobre entrada e saída em arquivos com Python

Alguns exercícios desta lista compreendem a leitura de arquivos pré-existentes. Na atividade AER-Alg-28 no Google Classroom, você pode baixar alguns arquivos que podem ser usados para testar seus exercícios.

**OBS.: utilize as mesmas regras das listas anteriores para dar nomes aos arquivos.**

### Exercícios:

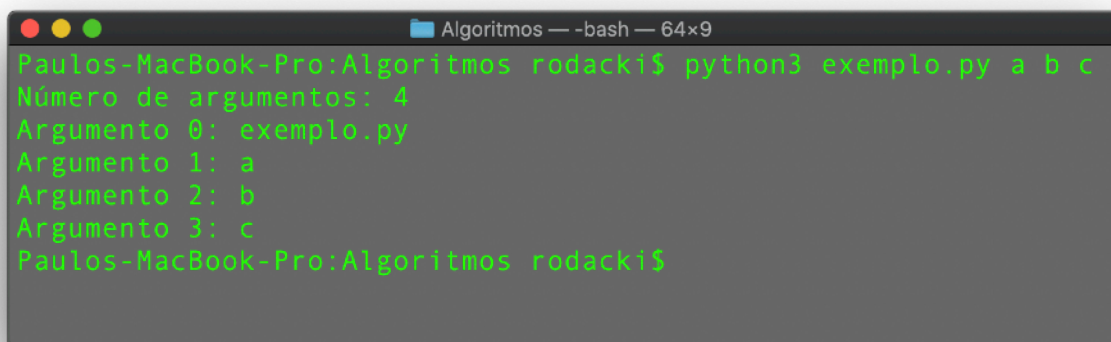
1. **Passagem de argumentos para o programa em linha de comando.** Neste exercício, vamos aprender a passar argumentos para programas Python em linha de comando. Para tanto, siga os seguintes passos:

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):

```
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`



```
Algoritmos — -bash — 64x9
Paulos-MacBook-Pro:Algoritmos rodacki$ python3 exemplo.py a b c
Número de argumentos: 4
Argumento 0: exemplo.py
Argumento 1: a
Argumento 2: b
Argumento 3: c
Paulos-MacBook-Pro:Algoritmos rodacki$
```

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 topiar 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 comando não modifica os arquivos originais e não cria um novo arquivo, ele apenas exibe o conteúdo dos arquivos na sequência. 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 seu 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 (portanto 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.