Programação em Python

CEFET-MG

Programação em Python

- **Arquivos:**
  - **Lendo dados de um arquivo**

```
with open('pi.txt') as file_object:
    contents = file_object.read()
print(contentes)
print(contentes.rstrip())
print(contentes.strip())
```

#with: Python decide quando fechar o arquivo. (mais seguro)

close(): fecha o arquivo

#open-as: abre o arquivo.

#read(): lê todo o conteúdo do arquivo.

#strip(): remove espaços em branco no início e no final de uma string

Programação em Python

- **Arquivos:**
  - **Lendo dados de um arquivo**

```
with open(“pi.txt”, ”r”, encoding="utf-8") as file_object:
```

r: modo somente leitura (padrão)

Para reconhecer caracteres especiais, símbolos, acentos, emojis,...

Programação em Python

Modo	Descrição	Pode Ler?	Pode Escrever?	Cria Arquivo se Não Existir?	Sobrescreve?	Posicionamento Inicial
"r"	Leitura (padrão)	✓	✗	✗	✗	Início do arquivo
"r+"	Leitura e escrita	✓	✓	✗	✗	Início do arquivo
"w"	Escrita (substitui conteúdo)	✗	✓	✓	✓	Início (arquivo zerado)
"w+"	Leitura e escrita (substitui conteúdo)	✓	✓	✓	✓	Início (arquivo zerado)

Programação em Python

Modo	Descrição	Pode Ler?	Pode Escrever?	Cria Arquivo se Não Existir?	Sobrescreve?	Posicionamento Inicial
"a"	Acrescentar (append)	✗	✓	✓	✗	Fim do arquivo
"a+"	Leitura e escrita no fim (append)	✓	✓	✓	✗	Fim do arquivo
"x"	Criação exclusiva	✗	✓	✓ (erro se existir)	✗	Início

Programação em Python

Modo	Descrição	Arquivos
"b"	Modo binário (usar junto)	.mp3, .zip, .jpg
"t"	Modo texto (padrão)	Texto (stg)

- **Exemplo:**

```
with open("exemplo.txt", "r", encoding="utf-8") as arq:
    conteudo = arq.read()
    print(conteudo)
    print(type(conteudo))
```

Programação em Python

Modo	Descrição	Arquivos
"b"	Modo binário (usar junto)	.mp3, .zip, .jpg
"t"	Modo texto (padrão)	Texto (stg)

- **Exemplo:**

```
with open("exemplo.txt", "rb") as arq:
    conteudo = arq.read()
    print(conteudo)
    print(type(conteudo))
```

Programação em Python

- **Arquivos:**
  - **Lendo dados linha a linha:**

```
file_name = 'txt/arq.txt'
with open(file_name) as file_object:
    for lin in file_object:
        print(lin)

with open(file_name) as file_object:
    linhas = arquivo.readlines()
    print(linhas)
```

Programação em Python

- **Arquivos:**
  - **Lendo dados valor por valor:**

```
dist = list()
with open('txt/arq2.dat‘) as arquivo:
    for i in arquivo:
        valores = list(map(int, i.strip().split()))
        dist.append(valores)
print(dist)
```

 # strip().split(): Remove espaços extras e divide os números

 # map: define cada elemento da lista como int

 # Para ler elementos de matriz numérica: numpy

Programação em Python

- **Arquivos:**
  - **Módulo pathlib:**
    - Mais moderno que o open.
    - Mais usado para as strings.
    - Comando read_file
      - Retorna o conteúdo do arquivo como texto.

Programação em Python

- **Arquivos:**
  - **Módulo pathlib:**
    - Ex.:

```
from pathlib import*


caminho = Path(‘pi.txt')
texto = caminho.read_text()
print(texto)
```

#read_text(encoding='utf-8'): para arquivos grandes ou com caracteres especiais

Programação em Python

- **Exercícios:**

- **Aprendendo Python**: Abra um arquivo em branco em seu editor de texto e escreva algumas linhas que sintetizem o que você aprendeu sobre Python até agora. Comece cada linha com a expressão Em Python podemos.... Salve o arquivo como learning_python.txt no mesmo diretório em que estão seus exercícios deste capítulo. Escreva um programa que leia o arquivo e mostre o que você escreveu, três vezes. Exiba o conteúdo uma vez lendo o arquivo todo, uma vez percorrendo o objeto arquivo com um laço e outra armazenando as linhas em uma lista e então trabalhando com ela fora do bloco with.

Programação em Python

```
# Criação do arquivo learning_python.txt com algumas linhas

#Ao invés do editor de texto

with open("learning_python.txt", "w", encoding="utf-8") as arquivo:
    arquivo.write("Em Python podemos criar programas simples e complexos.\n")
    arquivo.write("Em Python podemos manipular arquivos com facilidade.\n")
    arquivo.write("Em Python podemos usar estruturas de dados como listas e dicionários.\n")
    arquivo.write("Em Python podemos programar de forma orientada a objetos.\n")
    arquivo.write("Em Python podemos automatizar tarefas e analisar dados.\n")
```

Programação em Python

```
# --- 1ª forma: lendo o arquivo todo de uma vez ---
print("=== Leitura completa do arquivo ===")
with open("learning_python.txt", "r", encoding="utf-8") as arquivo:
    conteudo = arquivo.read()
print(conteudo)

# --- 2ª forma: percorrendo o arquivo linha por linha ---
print("=== Leitura linha a linha com laço ===")
with open("learning_python.txt", "r", encoding="utf-8") as arquivo:
    for linha in arquivo:
        print(linha.strip())
```

Programação em Python

```
# --- 3ª forma: lendo as linhas em uma lista e exibindo fora do bloco with ---
print("=== Leitura armazenando as linhas em uma lista ===")
with open("learning_python.txt", "r", encoding="utf-8") as arquivo:
    linhas = arquivo.readlines()

for linha in linhas:
    print(linha.strip())
```

Programação em Python

- **Exercícios:**

- **Aprendendo C**: Você pode usar o método replace() para substituir qualquer palavra por uma palavra diferente em uma string. Eis um exemplo rápido que mostra como substituir a palavra 'dog' por 'cat' em uma frase:

```
>>> message = "I really like dogs.“
>>> message.replace('dog', 'cat') 'I really like cats.'
```

- Leia cada linha do arquivo learning_python.txt que você acabou de criar e substitua a palavra Python pelo nome de outra linguagem, por exemplo, C. Mostre cada linha modificada na tela.

Programação em Python

```
# Lê o arquivo e substitui "Python" por "C"
with open("learning_python.txt", "r", encoding="utf-8") as arquivo:
    for linha in arquivo:
        linha_modificada = linha.replace("Python", "C")
        print(linha_modificada.strip())
```

Programação em Python

- **Arquivos:**
  - **Escrevendo dados em um arquivo vazio:**

```
filename = 'txt/teste.txt'

with open(filename, 'w') as arq:

    arq.write("I adoro Python.")
```

#r: somente leitura
#r+: leitura e escrita
#w: somente escrita (apaga pré existente)
#a: concatenação

#write: escreve uma string

Programação em Python

- **Exercícios:**

- **Lista de convidados:** Escreva um laço while que pergunte o nome aos usuários. Quando fornecerem seus nomes, apresente uma saudação na tela e acrescente uma linha que registre a visita do usuário em um arquivo chamado guest_book.txt. Certifique-se de que cada entrada esteja em uma nova linha do arquivo.

- Crie uma função que leia o arquivo e mostre o conteúdo na tela

Programação em Python

```
def registrar_visitantes():
    """Pergunta o nome dos usuários e registra no arquivo guest_book.txt"""
    while True:
        nome = input("Digite seu nome (ou 'sair' para encerrar): ")
        if nome.lower() == 'sair':
            break
        print(f"Olá, {nome}! Seja bem-vindo(a)!")

        # Abre o arquivo no modo de adição e escreve o nome
        with open("guest_book.txt", "a", encoding="utf-8") as arquivo:
            arquivo.write(f"{nome}\n")
```

Programação em Python

```
def mostrar_conteudo():
    """Lê o conteúdo do arquivo guest_book.txt e mostra na tela"""
    try:
        with open("guest_book.txt", "r", encoding="utf-8") as arquivo:
            conteudo = arquivo.read()
            print("\n Visitantes registrados:")
            print(conteudo)
    except FileNotFoundError:
        print("O arquivo guest_book.txt ainda não existe.")


# Programa principal
registrar_visitantes()
mostrar_conteudo()
```

Programação em Python

- **Exercícios:**

- **Enquete sobre programação:** Escreva um laço while que pergunte às pessoas por que elas gostam de programação. Sempre que alguém fornecer um motivo, acrescente-o em um arquivo que armazene todas as respostas.

- Crie uma função que leia o arquivo e mostre o conteúdo na tela.

Programação em Python

```
def registrar_motivos():
    """Pergunta às pessoas por que elas gostam de programação e grava as respostas em um
       arquivo."""
    while True:
        motivo = input("Por que você gosta de programar? (digite 'sair' para encerrar): ")
        if motivo.lower() == 'sair':
            break

        # Adiciona o motivo no arquivo
        with open("motivos_programacao.txt", "a", encoding="utf-8") as arquivo:
            arquivo.write(f"{motivo}\n")

        print("Motivo registrado! Obrigado por compartilhar.\n")
```

Programação em Python

```
def mostrar_motivos():
    """Lê o arquivo e mostra todos os motivos registrados."""
    try:
        with open("motivos_programacao.txt", "r", encoding="utf-8") as arquivo:
            conteudo = arquivo.read().strip()
            if conteudo:
                print("\n Motivos pelos quais as pessoas gostam de programar:\n")
                print(conteudo)
            else:
                print("Ainda não há motivos registrados.")
    except FileNotFoundError:
        print("O arquivo motivos_programacao.txt ainda não existe.")



# Programa principal
registrar_motivos()
mostrar_motivos()
```

Programação em Python

- **Exercícios:**

- Visite o site: https://gutenberg.org/ e baixe o arquivo texto (*.txt) do livro Mobi Dick.

- Quantas palavras aparecem neste arquivo?

Programação em Python

```
def contar_palavras(arquivo):
    """Conta o número de palavras em um arquivo de texto."""
    try:
        with open(arquivo, encoding="utf-8") as f:
            texto = f.read()
    except FileNotFoundError:
        print(f"O arquivo '{arquivo}' não foi encontrado.")
        return

    palavras = texto.split()
    num_palavras = len(palavras)
    print(f"O arquivo '{arquivo}' contém aproximadamente {num_palavras:,} palavras.")


# Programa principal
contar_palavras("2701-0.txt")  # ou o nome do arquivo que você baixou
```

Programação em Python

- **Exceções:**
  - Erros durante a execução.
    - Memória.
    - Tipo de dado.
    - Entrada não esperada.
    - Comportamento anômalo.

Programação em Python

- **Exceções:**
  - **Bloco try-except**
    - O código espera um determinado comportamento (entrada).
    - Caso ocorra uma exceção: ele diz o que fazer.

```
try:
        instrução
except TipoDeErro:
        instrução em caso de erro
```

Programação em Python

- **Exceções:**

  - **Bloco try-except**
    - Ex.: Divisão por zero

```
try:
    print(5/0)
except ZeroDivisionError:
    print(“Você não pode divider por zero!")
```

#pass: não faz nada

Programação em Python

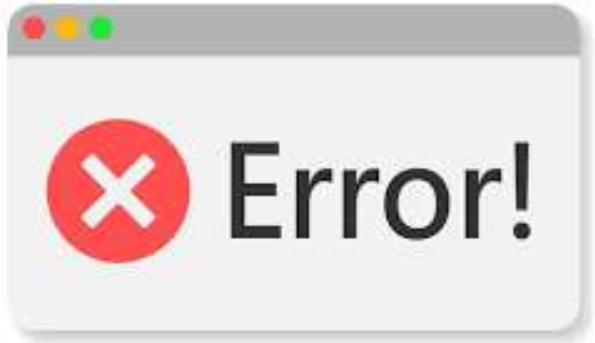
- **Exceções:**

  - **Bloco try-except-else**

```
Try:
    instrução
except TipoDeError:
    instrução
else:
    instrução
```

Programação em Python

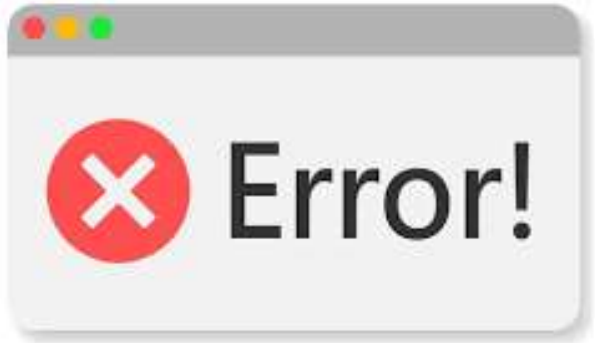
- **Exceções:**
  - **Principais Exceções:**



    - FileNotFoundError: Arquivo não encontrado.
    - OverflowError: Resultado de cálculo numérico muito grande para ser representado.
    - RuntimeError: Erro geral que ocorre em tempo de execução.
    - ModuleNotFoundError: Módulo não encontrado.

Programação em Python

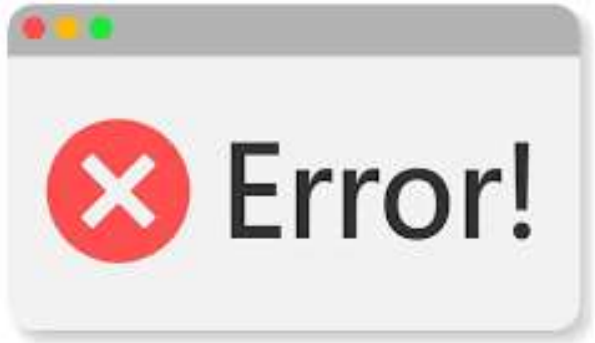
- **Exceções:**
  - **Principais Exceções:**



    - ImportError: Módulo ou função não encontrado durante a importação
    - AttributeError: Atributo inválido ou inexistente de um objeto
    - KeyError: Chave não encontrada em um dicionário
    - IndexError: Índice fora do intervalo de uma sequência (lista, tupla, etc.).

Programação em Python

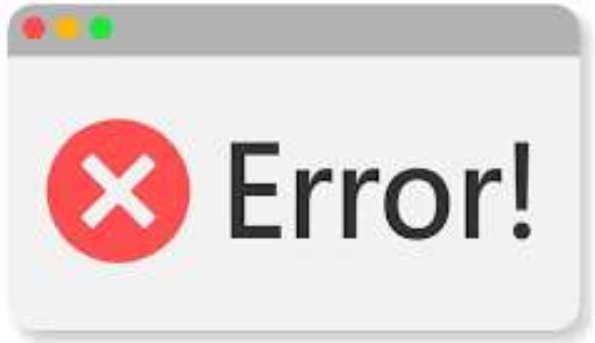
- **Exceções:**
  - **Principais Exceções:**



    - ValueError: Valor inválido, embora o tipo seja correto.
    - TypeError: Tipo de dado inválido para uma operação.
    - NameError: Variável ou nome de função não definido.
    - MemoryError: Memória insuficiente.
    - E se não sei o tipo de erro?

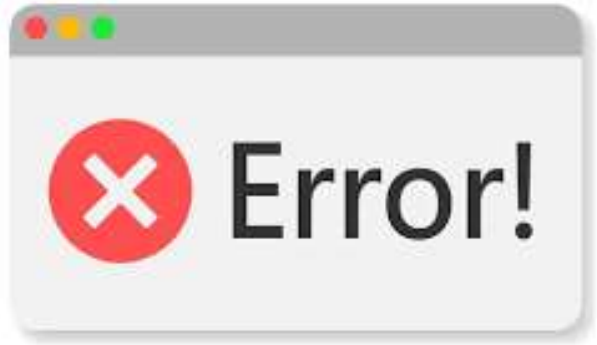
Programação em Python

- **Exceções:**
  - **E se não sei o tipo de erro?**
    - Erro Genérico
    - P/ a maioria dos erros (Mas não todos)
    - try:
      - Instrução
    - except Exception as e:
      - print(e) **#Mostra o tipo de erro.**



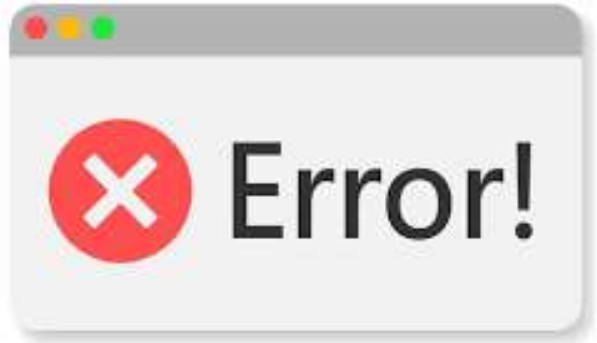
Programação em Python

- **Exceções:**
  - Tipo de Erro criado pelo dev:
    - Cria-se uma classe filha de da classe Exception
    - Cria-se uma função para verificar a condição.
    - Utilize a função dentro do bloco try-except
    - Exemplo: Erro se valor maior que 10.



Programação em Python

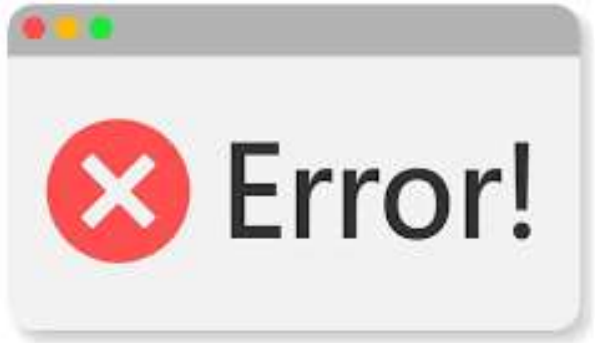
- **Exceções:**
  - Tipo de Erro criado pelo dev:
    - Exemplo: Erro se valor maior que 10.




```
class NumeroMaiorQueDezError(Exception):
    """Exceção lançada quando o número é maior que 10."""
    def __init__(self):
        super().__init__("O número é maior que 10!")
```

Programação em Python

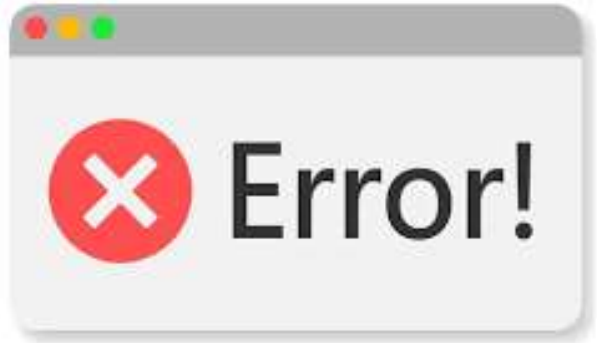
- **Exceções:**
  - Tipo de Erro criado pelo dev:
    - Exemplo: Erro se valor maior que 10.




```
# Função que verifica o número
def verificar_numero(n):
    if n > 10:
        # Lança a exceção personalizada
        raise NumeroMaiorQueDezError()
```

Programação em Python

- **Exceções:**
  - Tipo de Erro criado pelo dev:
    - Exemplo: Erro se valor maior que 10.




```
# Uso com try/except
try:
    numero = int(input("Digite um número: "))
    verificar_numero(numero)

except NumeroMaiorQueDezError as e:
    print("Erro capturado:", e)
```

Programação em Python

- **Exercícios:**

- **Adição**: Um problema comum quando pedir entradas numéricas ocorre quando as pessoas fornecem texto no lugar de números. Ao tentar converter a entrada para um int, você obterá um ValueError. Escreva um programa que peça dois números ao usuário. Some-os e mostre o resultado. Capture o ValueError caso algum dos valores de entrada não seja um número e apresente uma mensagem de erro simpática. Teste seu programa fornecendo dois números e, em seguida, digite um texto no lugar de um número. Coloque um laço while para que o usuário possa continuar fornecendo números, mesmo se cometerem um erro e digitarem um texto no lugar de um número.

Programação em Python

```
def somar_numeros():
    """Solicita dois números ao usuário, soma e trata entradas inválidas."""
    while True:
        try:
            num1 = input("Digite o primeiro número (ou 'sair' para encerrar): ")
            if num1.lower() == 'sair':
                print("Encerrando o programa. Até mais!")
                break

            num1 = int(num1)
            num2 = input("Digite o segundo número: ")
            num2 = int(num2)
            resultado = num1 + num2
            print(f"A soma de {num1} + {num2} é {resultado}.\n")

        except ValueError:  # Captura quando a conversão para int falha
            print("Ops! Você precisa digitar números válidos. Tente novamente.\n")
```

```
# Executa o programa
somar_numeros()
```

Programação em Python

- **Exercícios:**

- **Gatos e cachorros:** Crie dois arquivos usando Python, cats.txt e dogs.txt. Armazene pelo menos três nomes de gatos no primeiro arquivo e três nomes de cachorro no segundo arquivo. Escreva um programa que tente ler esses arquivos e mostre o conteúdo do arquivo na tela. Coloque seu código em um bloco try-except para capturar o erro FileNotFound e apresente uma mensagem simpática caso o arquivo não esteja presente. Mova um dos arquivos para um local diferente de seu sistema e garanta que o código no bloco except seja executado de forma apropriada.

Programação em Python

```
# Criando os arquivos com alguns nomes
def cria_arquivos():
    with open("cats.txt", "w", encoding="utf-8") as cats:
        cats.write("Mimi\n")
        cats.write("Luna\n")
        cats.write("Simba\n")

    with open("dogs.txt", "w", encoding="utf-8") as dogs:
        dogs.write("Rex\n")
        dogs.write("Bolt\n")
        dogs.write("Toby\n")
```

Programação em Python

```
def mostrar_arquivo(nome_arquivo):
    """Tenta ler um arquivo e mostra seu conteúdo. Trata FileNotFoundError."""
    try:
        with open(nome_arquivo, "r", encoding="utf-8") as arquivo:
            conteudo = arquivo.read()
            print(f"\nConteúdo do arquivo {nome_arquivo}:\n{conteudo}")
    except FileNotFoundError:
        print(f"\nOps! O arquivo '{nome_arquivo}' não foi encontrado. 😿")


# Testando os arquivos
cria_arquivos()
mostrar_arquivo("cats.txt")
mostrar_arquivo("dogs.txt")
```

Programação em Python

- Exercícios:

- **Leitura Simples com Tratamento de Erros:** Escreva um programa que solicita ao usuário o nome de um arquivo de texto e tenta abri-lo. Caso o arquivo não exista, capture a exceção e exiba uma mensagem amigável ao usuário.

Programação em Python

```
def abrir_arquivo():
    """Solicita o nome de um arquivo e tenta abri-lo, mostrando uma mensagem amigável caso não exista."""
    nome_arquivo = input("Digite o nome do arquivo de texto que deseja abrir: ")

    try:
        with open(nome_arquivo, "r", encoding="utf-8") as arquivo:
            conteudo = arquivo.read()
            print(f"\n Conteúdo do arquivo '{nome_arquivo}':\n")
            print(conteudo)
    except FileNotFoundError:
        print(f"\nOps! O arquivo '{nome_arquivo}' não foi encontrado. Verifique o nome e tente novamente.")

# Executa o programa
abrir_arquivo()
```

Programação em Python

- Exercícios:

- **Escrita em Arquivo com Validação**

- Crie um programa que pede ao usuário para inserir várias linhas de texto, que serão salvas em um arquivo. Utilize try-except para tratar erros como a falta de permissões para escrever no arquivo.

- Instruções:
  - Solicite o nome do arquivo de saída ao usuário.
  - Capture possíveis exceções como PermissionError ao tentar gravar no arquivo.

Programação em Python

```
def salvar_texto():
    """Solicita várias linhas de texto do usuário e grava em um arquivo com tratamento de erros."""

    nome_arquivo = input("Digite o nome do arquivo de saída (ex: meu_texto.txt): ")

    print("Digite o texto que deseja salvar no arquivo. Digite 'FIM' em uma nova linha para encerrar.")

    linhas = []
    while True:
        linha = input()
        if linha.upper() == "FIM":
            break
        linhas.append(linha)

    try:
        with open(nome_arquivo, "w", encoding="utf-8") as arquivo:
            for linha in linhas:
                arquivo.write(linha + "\n")
        print(f"\n✅ Texto salvo com sucesso no arquivo '{nome_arquivo}'!")
    except PermissionError:
        print(f"\n⚠️ Não foi possível gravar no arquivo '{nome_arquivo}'. Verifique suas permissões.")
    except Exception as e:
        print(f"\nOcorreu um erro inesperado: {e}")


# Executa o programa
salvar_texto()
```

Programação em Python

- Exercícios:

- **LOG:** Crie um programa que Tenta ler um arquivo de configuração chamado config.txt.
- Caso o arquivo esteja corrompido, não exista ou vazio, capture a exceção apropriada e crie um arquivo e registre o erro neste arquivo de log (errors.log).
- Dica: use o módulo OS e a função os.path.exists()
  - os.path.exists() é uma função em Python que verifica se um caminho (arquivo ou diretório) existe no sistema de arquivos. Retorna True ou False.

Programação em Python

```
import os

def ler_config():
    """Tenta ler o arquivo config.txt e trata exceções, registrando erros em errors.log."""

    nome_config = "config.txt"
    nome_log = "errors.log"

    try:
        if not os.path.exists(nome_config):
            raise FileNotFoundError("Arquivo de configuração não encontrado.")

        with open(nome_config, "r", encoding="utf-8") as arquivo:
            conteudo = arquivo.read().strip()
            if not conteudo:
                raise ValueError("Arquivo de configuração está vazio.")

            # Aqui você poderia adicionar validação adicional do conteúdo
            print(f"Arquivo de configuração lido com sucesso:\n{conteudo}")
```

Programação em Python

```
    except (FileNotFoundError, ValueError) as e:
        print(f" Problema ao ler o arquivo de configuração: {e}")
        # Registra o erro no arquivo de log
        with open(nome_log, "a", encoding="utf-8") as log:
            log.write(f"Erro: {e}\n")
        print(f"Detalhes do erro foram registrados em '{nome_log}'.")
    except Exception as e:
        print(f" Um erro inesperado ocorreu: {e}")
        with open(nome_log, "a", encoding="utf-8") as log:
            log.write(f"Erro inesperado: {e}\n")
        print(f"Detalhes do erro inesperado foram registrados em '{nome_log}'.")


# Executa a função
ler_config()
```

Programação em Python

- **Exercícios:**

- Visite o site: https://gutenberg.org/ e baixe o arquivo texto (*.txt) do livro Mobi Dick.

- Quantas vezes a palavra ‘the’ aparece neste arquivo?

- **Dica: use o comando .count(‘the’).**

Programação em Python

```
def contar_the(arquivo):
    """Conta quantas vezes a palavra 'the' aparece no arquivo."""
    try:
        with open(arquivo, "r", encoding="utf-8") as f:
            texto = f.read().lower()  # converte tudo para minúsculas
    except FileNotFoundError:
        print(f"O arquivo '{arquivo}' não foi encontrado.")
        return

    # Conta apenas a palavra 'the' exata, evitando partes de outras palavras
    palavras = texto.split()
    contador = sum(1 for palavra in palavras if palavra.strip(".,;:!?\"") == "the")

    print(f"A palavra 'the' aparece aproximadamente {contador:,} vezes no arquivo '{arquivo}'.")

# Executa a função
contar_the("2701-0.txt")
```

Programação em Python

- **DESAFIO: Gerenciamento de Arquivos**
- Implemente um programa que gerencie múltiplos arquivos de texto. O programa deve:
  - Permitir ao usuário criar, abrir e editar arquivos.
  - Utilizar try-except para tratar os seguintes erros:
    - Arquivo inexistente ao tentar abrir ou editar.
    - Permissões insuficientes ao tentar abrir ou editar.
  - Garantir que erros inesperados sejam registrados em um arquivo de log (gerenciamento_erros.log).

Programação em Python

```
import os

LOG_ERROS = "gerenciamento_erros.log"

def registrar_erro(mensagem):
    """Registra a mensagem de erro no arquivo de log."""
    with open(LOG_ERROS, "a", encoding="utf-8") as log:
        log.write(mensagem + "\n")


def criar_arquivo():
    """Cria um novo arquivo de texto."""
    nome_arquivo = input("Digite o nome do arquivo a ser criado: ")
    try:
        with open(nome_arquivo, "w", encoding="utf-8") as f:
            print(f"Arquivo '{nome_arquivo}' criado com sucesso!")
    except PermissionError:
        print(f" Não há permissão para criar o arquivo '{nome_arquivo}'.")
        registrar_erro(f"PermissionError ao criar arquivo: {nome_arquivo}")
    except Exception as e:
        print(f" Erro inesperado: {e}")
        registrar_erro(f"Erro inesperado ao criar arquivo {nome_arquivo}: {e}")
```

Programação em Python

```
def abrir_arquivo():
    """Abre e mostra o conteúdo de um arquivo existente."""
    nome_arquivo = input("Digite o nome do arquivo que deseja abrir: ")
    try:
        with open(nome_arquivo, "r", encoding="utf-8") as f:
            conteudo = f.read()
            print(f"\n Conteúdo do arquivo '{nome_arquivo}':\n")
            print(conteudo)
    except FileNotFoundError:
        print(f" O arquivo '{nome_arquivo}' não existe.")
        registrar_erro(f"FileNotFoundError ao abrir arquivo: {nome_arquivo}")
    except PermissionError:
        print(f" Sem permissão para abrir o arquivo '{nome_arquivo}'.")
        registrar_erro(f"PermissionError ao abrir arquivo: {nome_arquivo}")
    except Exception as e:
        print(f" Erro inesperado: {e}")
        registrar_erro(f"Erro inesperado ao abrir arquivo {nome_arquivo}: {e}")
```

Programação em Python

```
def editar_arquivo():
    """Permite editar (adicionar linhas) em um arquivo existente."""
    nome_arquivo = input("Digite o nome do arquivo que deseja editar: ")
    try:
        with open(nome_arquivo, "a", encoding="utf-8") as f:
            print("Digite o texto que deseja adicionar (digite 'FIM' para encerrar):")
            while True:
                linha = input()
                if linha.upper() == "FIM":
                    break
                f.write(linha + "\n")
        print(f"Arquivo '{nome_arquivo}' editado com sucesso!")
    except FileNotFoundError:
        print(f"O arquivo '{nome_arquivo}' não existe.")
        registrar_erro(f"FileNotFoundError ao editar arquivo: {nome_arquivo}")
    except PermissionError:
        print(f"Sem permissão para editar o arquivo '{nome_arquivo}'.")
        registrar_erro(f"PermissionError ao editar arquivo: {nome_arquivo}")
    except Exception as e:
        print(f"Erro inesperado: {e}")
        registrar_erro(f"Erro inesperado ao editar arquivo {nome_arquivo}: {e}")
```

Programação em Python

```
def menu():
    """Exibe o menu e gerencia as opções do usuário."""
    while True:
        print("\n=== Gerenciador de Arquivos ===")
        print("1. Criar arquivo")
        print("2. Abrir arquivo")
        print("3. Editar arquivo")
        print("4. Sair")
        escolha = input("Escolha uma opção: ")
         if escolha == "1":
                criar_arquivo()
            elif escolha == "2":
                abrir_arquivo()
            elif escolha == "3":
                editar_arquivo()
            elif escolha == "4":
                print("Saindo do gerenciador de arquivos...")
                break
            else:
                print("Opção inválida. Tente novamente.")

# Executa o programa
if __name__ == "__main__":
    menu()
```