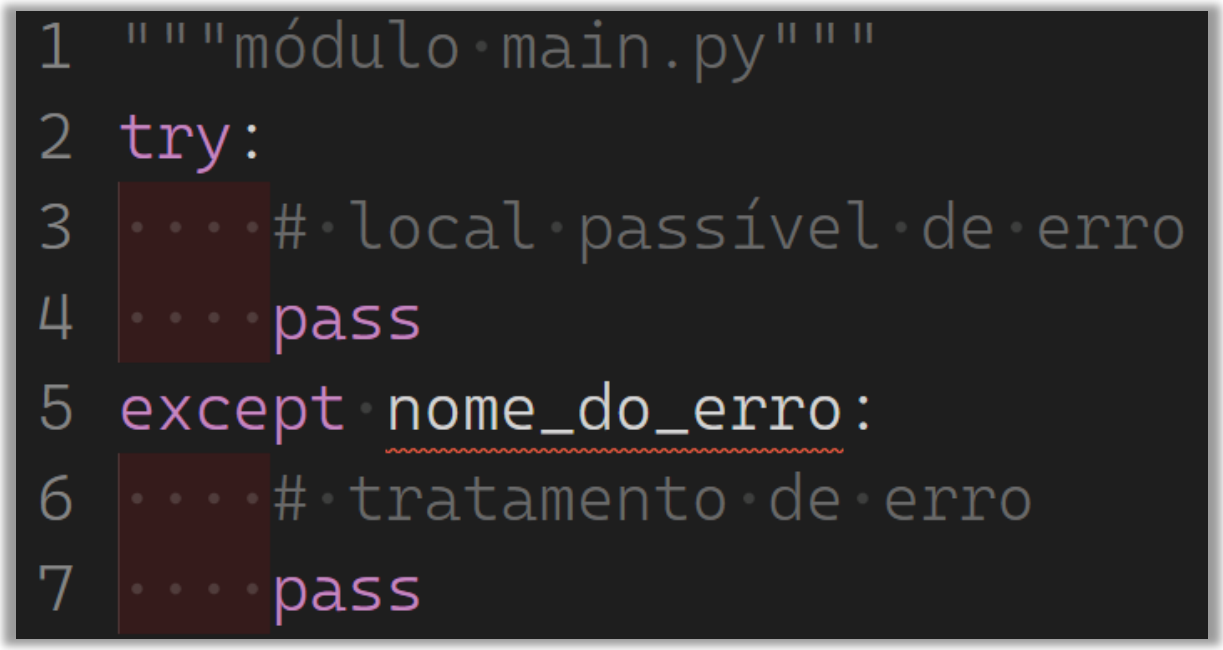
# Aula 04

## Tratamento de Erros

O "try-except" é um recurso fundamental da linguagem de programação Python que permite lidar com exceções em tempo de execução de uma maneira mais controlada. Basicamente, o bloco de código "try" tenta executar uma determinada operação que pode gerar uma exceção, enquanto o bloco de código "except" lida com a exceção caso ela seja levantada.

O formato básico de um bloco "try-except" em Python é o seguinte:




O bloco "try" contém o código que será executado e que pode levantar uma exceção. Se uma exceção for levantada, o Python interrompe a execução do bloco "try" e passa para o bloco "except".

O argumento "nome_do_erro" é opcional e especifica qual exceção o bloco "except" deve lidar. Se nenhum argumento for fornecido, o bloco "except" lidará com todas as exceções. No entanto, é geralmente recomendável especificar o tipo de exceção que o bloco "except" deve lidar, pois isso pode ajudar a identificar erros mais facilmente.

Se uma exceção for levantada, o bloco "except" será executado, e qualquer código contido nele será executado. O bloco "except" deve conter um código que lide com a exceção de alguma forma, como exibir uma mensagem de erro, tentar corrigir o erro ou simplesmente sair do programa.

É possível ter vários blocos "except" para lidar com diferentes tipos de exceções. Além disso, o bloco "try" pode ser seguido por um bloco "finally" opcional, que contém um código que será sempre executado, independentemente de ter ocorrido uma exceção ou não.

Abaixo temos alguns erros comuns:

### SyntaxError

Ocorre quando o código Python contém uma sintaxe inválida. Por exemplo, esquecer um dois-pontos ou um parêntese pode causar um SyntaxError.

### NameError

Ocorre quando uma variável ou função é referenciada, mas não foi definida anteriormente. Por exemplo, tentar usar uma variável que não foi atribuída um valor ou chamar uma função que não existe.

```python
"""módulo main.py"""
entrada = input("Digite algo : ")

try:
    print(ent)
except NameError as ne:
    print(f"\nAconteceu um erro : {ne}\n")
```

```
PS C:\Users\gutoh\OneDrive\0.Python> python main.py
Digite algo : asdf

Aconteceu um erro : name 'ent' is not defined

PS C:\Users\gutoh\OneDrive\0.Python>
```

### TypeError

Ocorre quando uma operação é realizada em um objeto do tipo errado. Por exemplo, tentar adicionar um número a uma string ou indexar um número.

```python
"""módulo main.py"""
num = 42
string = 'planeta'

try:
    num + string
except TypeError as te:
    print(f"\nAconteceu um erro : {te}\n")
```

```
PS C:\Users\gutoh\OneDrive\0.Python> python main.py

Aconteceu um erro : unsupported operand type(s) for +: 'int'
and 'str'

PS C:\Users\gutoh\OneDrive\0.Python>
```

### IndexError

Ocorre quando uma lista ou sequência é indexada com um valor fora de seu intervalo válido. Por exemplo, tentar acessar o quinto elemento de uma lista que tem apenas quatro elementos.

```python
"""módulo main.py"""
lista = [range(10)]

try:
    print(lista[10])
except IndexError as ie:
    print(f"\nAconteceu um erro : {ie}\n")
```

```
PS C:\Users\gutoh\OneDrive\0.Python> python main.py

Aconteceu um erro : list index out of range

PS C:\Users\gutoh\OneDrive\0.Python>
```

### ValueError

Ocorre quando uma função recebe um argumento com um valor inválido. Por exemplo, tentar converter uma string em um número inteiro que não é um número válido.

```python
"""módulo main.py"""
palavra = 'planeta'

try:
    numero = int(palavra)
except ValueError as ve:
    print(f"\nAconteceu um erro : {ve}\n")
```

```
PS C:\Users\gutoh\OneDrive\0.Python> python main.py

Aconteceu um erro : invalid literal for int() with base 10: '
planeta'

PS C:\Users\gutoh\OneDrive\0.Python>
```

### KeyError

Ocorre quando um dicionário é indexado com uma chave que não existe. Por exemplo, tentar acessar um valor em um dicionário que não possui a chave especificada.

```python
"""módulo main.py"""
calculo = {'num_1': 40, 'num_2': 2, 'operacao': '+', 'resultado': 42}

try:
    print(calculo['valor'])
except KeyError as ke:
    print(f"\nAconteceu um erro : {ke}\n")
```

```
PS C:\Users\gutoh\OneDrive\0.Python> python main.py

Aconteceu um erro : 'valor'

PS C:\Users\gutoh\OneDrive\0.Python>
```

### AttributeError

Ocorre quando um objeto não possui um atributo ou método solicitado. Por exemplo, tentar chamar um método em um objeto que não tem esse método.

```python
"""módulo main.py"""
lista = [range(10)]

try:
    print(lista.strip())
except AttributeError as ae:
    print(f"\nAconteceu um erro : {ae}\n")
```

```
PS C:\Users\gutoh\OneDrive\0.Python> python main.py

Aconteceu um erro : 'list' object has no attribute 'strip'

PS C:\Users\gutoh\OneDrive\0.Python>
```

### ZeroDivisionError

Ocorre quando uma divisão por zero é tentada. Por exemplo, tentar dividir um número por zero.

```python
"""módulo main.py"""
try:
    print(10 / 0)
except ZeroDivisionError as zde:
    print(f"\nAconteceu um erro : {zde}\n")
```

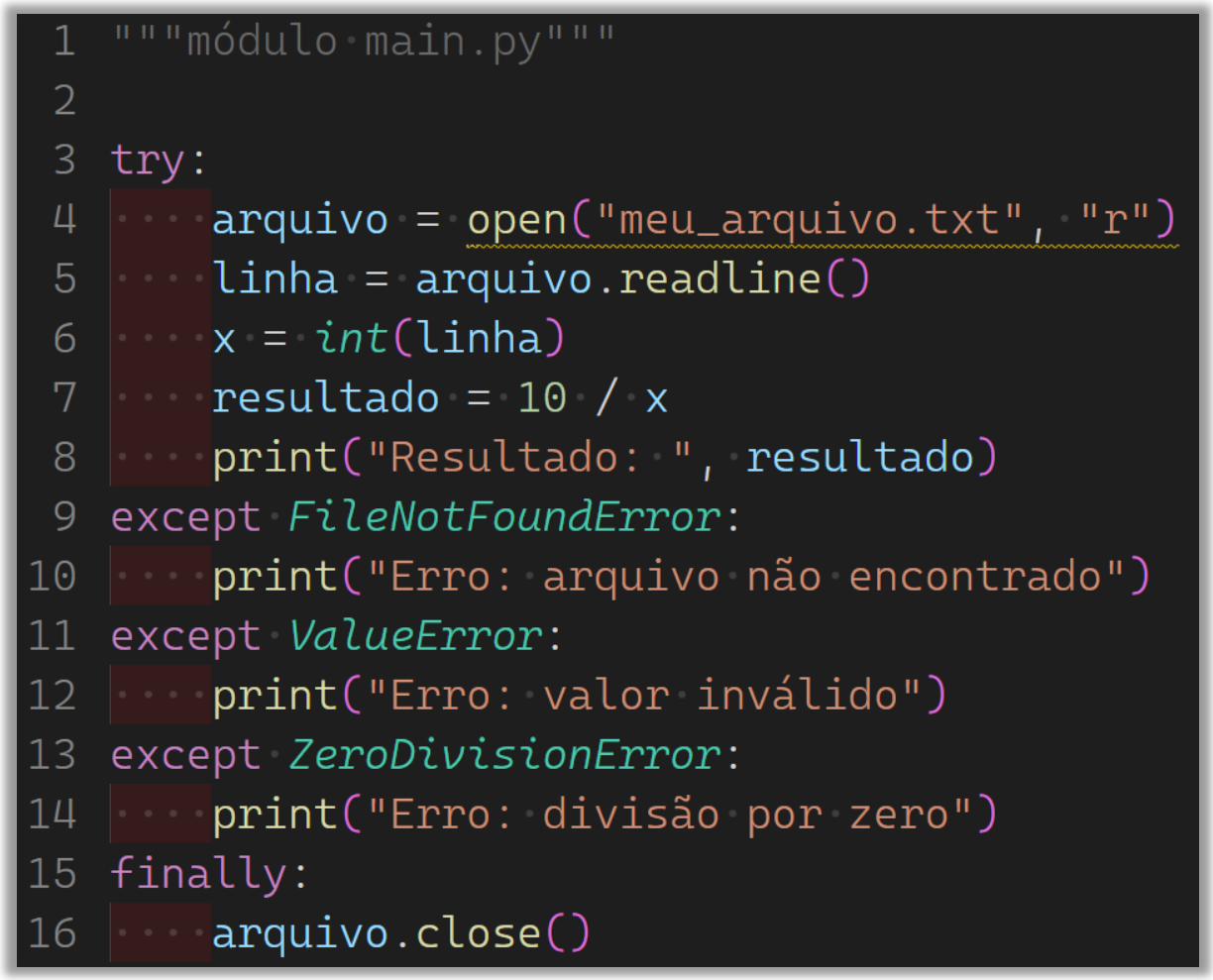
```
PS C:\Users\gutoh\OneDrive\0.Python> python main.py

Aconteceu um erro : division by zero

PS C:\Users\gutoh\OneDrive\0.Python>
```

## Erros Múltiplos

O uso ideal dos blocos "try...except" é sempre deixar o mais específico possível, como os exemplos anteriores. Acontece que nem sempre isso será possível, então, nesses casos, podemos encadear diversos exceptions encadeados.
Veja o exemplo abaixo:



```python
"""módulo main.py"""

try:
    arquivo = open("meu_arquivo.txt", "r")
    linha = arquivo.readline()
    x = int(linha)
    resultado = 10 / x
    print("Resultado: ", resultado)
except FileNotFoundError:
    print("Erro: arquivo não encontrado")
except ValueError:
    print("Erro: valor inválido")
except ZeroDivisionError:
    print("Erro: divisão por zero")
finally:
    arquivo.close()
```

Neste exemplo, o bloco "try" tenta abrir um arquivo, ler uma linha e dividir 10 pelo número lido. Se ocorrer uma exceção, o bloco "except" correspondente será executado. O bloco "finally" é usado para garantir que o arquivo seja sempre fechado, independentemente de ocorrer uma exceção ou não.

## try...except...else...finally

O try...except...else...finally é a estrutura completa de tratamento de erros do Python que é usada para gerenciar exceções (erros) que podem ocorrer durante a execução do código.

### try

O bloco try é usado para envolver o código que pode gerar exceções. Quando uma exceção é lançada no bloco try, a execução do código é interrompida e passa para o bloco except correspondente.

### except

O bloco except é usado para lidar com exceções que são lançadas no bloco try. Você pode especificar o tipo de exceção que deseja lidar usando a cláusula except <tipo de exceção>:, por exemplo, except ValueError:. Se você não especificar um tipo de exceção, o bloco except lidará com qualquer exceção lançada no bloco try.

Dentro do bloco except, você pode escrever código que será executado quando a exceção for lançada. Por exemplo, você pode imprimir uma mensagem de erro ou executar alguma ação para corrigir o problema.

### else

O bloco else é opcional e é executado somente se nenhum erro ocorrer no bloco try. Ou seja, se nenhuma exceção for lançada no bloco try, a execução do código passará para o bloco else.

Dentro do bloco else, você pode escrever código que deve ser executado quando nenhuma exceção é lançada. Por exemplo, você pode imprimir uma mensagem informando que o código foi executado com sucesso.

### finally

O bloco finally é opcional e é sempre executado, independentemente de ter ocorrido ou não uma exceção. Isso significa que, se houver um bloco finally, o código dentro dele será executado, mesmo que uma exceção tenha sido lançada no bloco try e não tenha sido tratada.

Dentro do bloco finally, você pode escrever código que deve ser executado independentemente do que acontecer. Por exemplo, você pode fechar um arquivo que foi aberto no bloco try, ou liberar recursos que foram alocados durante a execução do código.

```python
"""módulo main.py"""

try:
    arquivo = open("meu_arquivo.txt", "r")
except FileNotFoundError:
    print("Erro: arquivo não encontrado")
else:
    linhas = arquivo.readlines()
    print("Resultado: ", linhas)
    arquivo.close()
finally:
    print('Sempre serei executado.')
```

Neste exemplo, tentamos abrir um arquivo. Se o arquivo não existir, então irá gerar um erro. Se o arquivo for aberto, ele será lido e depois será fechado. Sempre no final, independentemente de ter tido sucesso ou não na abertura do arquivo, será executado o finally.

```
PS C:\Users\gutoh\OneDrive\0.Python> python main.py
Erro: arquivo não encontrado
Sempre serei executado.
PS C:\Users\gutoh\OneDrive\0.Python>
PS C:\Users\gutoh\OneDrive\0.Python> python main.py
Resultado:  ['uma linha dentro do meu_arquivo.txt\n']
Sempre serei executado.
PS C:\Users\gutoh\OneDrive\0.Python>
```

A grande vantagem de usarmos os blocos try...except é evitar que nossa execução seja interrompida caso aconteça um erro.

## Outros Usos

Também podemos usar para outros usos, como usar para controlar um laço de repetição até que o usuário digite um número válido.

```
"""módulo main.py"""

while True:
    try:
        num = int(input("Digite um número : "))
        break
    except ValueError:
        print('Eita!')
        print('Isso não foi um número válido.')
        print('Tente de novo ... ')
```

## Comando raise

O comando raise é usado em Python para sinalizar uma exceção (ou erro) que ocorreu durante a execução do código. Quando uma exceção é levantada, o programa para de executar o código normal e começa a procurar por um tratador (handler) de exceção que possa lidar com a exceção levantada. Se nenhum tratador for encontrado, o programa é encerrado e uma mensagem de erro é exibida.

A sintaxe básica do comando raise é a seguinte:

```
"""módulo main.py"""

raise Excecao("mensagem de erro")
```

onde Excecao é o tipo de exceção que está sendo levantado e "mensagem de erro" é uma descrição da exceção. Por exemplo, o seguinte código levanta uma exceção ValueError com uma mensagem de erro personalizada

```
"""módulo main.py"""

try:
    idade = int(input("Digite sua idade: "))
    if idade < 18:
        raise ValueError("Você precisa ter 18 anos ou mais.")
except ValueError as ve:
    print("Erro:", ve)
```

```
PS C:\Users\gutoh\OneDrive\0.Python> python main.py
Digite sua idade: 15
Erro: Você precisa ter 18 anos ou mais.
PS C:\Users\gutoh\OneDrive\0.Python>
PS C:\Users\gutoh\OneDrive\0.Python> python main.py
Digite sua idade: 20
PS C:\Users\gutoh\OneDrive\0.Python>
```

Além disso, é possível levantar exceções sem fornecer uma mensagem de erro. Por exemplo, o seguinte código levanta uma exceção ValueError sem nenhuma mensagem de erro:

```python
"""módulo main.py"""

try:
    idade = int(input("Digite sua idade: "))
    if idade < 18:
        raise ValueError
except ValueError:
    print("Erro.")
```

## Raise dentro do except

Também podemos usar o raise dentro do bloco except para ativarmos outra exceção mais externa.

```python
"""módulo main.py"""

try:
    try:
        lista = [1, 2, 3]
        elemento = lista[4]
    except IndexError as ie:
        print("Ocorreu um erro:", ie)
        raise ValueError("Índice inválido. A lista tem apenas 3 elementos.")
except ValueError as ve:
    print("Ocorreu um erro:", ve)
```

```
PS C:\Users\gutoh\OneDrive\0.Python> python main.py
Ocorreu um erro: list index out of range
Ocorreu um erro: Índice inválido. A lista tem apenas 3 elementos.
PS C:\Users\gutoh\OneDrive\0.Python>
```

Neste exemplo, o código tenta acessar o quinto elemento da lista "lista". Como a lista tem apenas três elementos, o código levanta uma exceção IndexError. Em seguida, o try/except captura a exceção IndexError e levanta uma exceção ValueError com uma mensagem de erro personalizada. O try/except externo captura essa exceção ValueError e lida com ela mostrando outra mensagem.

# Exercício para Entregar

1. Calculadora de Números Romanos.
    a. Crie uma calculadora das operações +, -, *, /, **.
        i. O intervalo de entrada e resultado deve ser (-4000,4000).
        ii. isto é, o valor mínimo recebido e calculado será -3999.
        iii. e o valor máximo recebido e calculado será 3999.
    b. O resultado da divisão deve ser representado como quociente e resto.
    c. A entrada dos dois números deve ser em números romanos.
    d. O resultado deve ser mostrado em números romanos.
    e. Use blocos try / except onde achar necessário, mas seja ESPECÍFICO.
    f. Você deve usar funções, que por sua vez devem estar em um módulo separado.
    g. Todos os cálculos devem ser salvos em um dicionário, que deve ser armazenado em uma lista.
    h. Ao encerrar o programa, todos os cálculos realizados devem ser salvos em um arquivo json.
2. Modelos:
    a. Operação tradicional

```
calculo = {'n1': 'X', 'n2': 'IV', 'op': '+', 'res': 'XIV'}
```

    b. Operação onde foi entrado um valor maior que 3999 ou menor que -3999.

```
calculo = {'n1': 'erro', 'n2': 'MM', 'op': '/', 'res': 'erro'}
```

    c. Modelo usando subtração com algum valor negativo.

```
calculo = {'n1': '-C', 'n2': 'X', 'op': '-', 'res': '-CX'}
```

    d. Modelo de divisão.

```
calculo = {'n1': 'XI', 'n2': 'III', 'op': '/', 'res': 'III', '%': 'II'}
```

# Exercícios para Praticar

## Exercícios try...except

1. Escreva um programa que solicita ao usuário que digite um número e, em seguida, imprima o número. Se o usuário digitar um valor não numérico, exiba uma mensagem de erro apropriada.
2. Escreva um programa que solicita ao usuário que digite dois números e, em seguida, imprima a divisão desses números. Se o segundo número for zero, exiba uma mensagem de erro apropriada.
3. Escreva um programa que cria uma lista vazia e, em seguida, tenta acessar o primeiro elemento da lista. Exiba uma mensagem de erro apropriada.
4. Escreva um programa que cria um dicionário e, em seguida, tenta acessar um valor no dicionário usando uma chave que não existe. Exiba uma mensagem de erro apropriada.
5. Escreva um programa que solicita ao usuário que digite um número inteiro e, em seguida, imprima o número mais um. Se o usuário digitar um valor que não é um número inteiro, exiba uma mensagem de erro apropriada.
6. Escreva um programa que importe o módulo math e tente usar uma função que não existe. Em seguida, exiba uma mensagem de erro apropriada.
7. Escreva um programa que cria uma lista com três elementos e, em seguida, tenta acessar o quarto elemento da lista. Exiba uma mensagem de erro apropriada.
8. Escreva um programa que cria um dicionário com duas chaves e, em seguida, tenta acessar uma chave que não existe. Exiba uma mensagem de erro apropriada.
9. Escreva um programa que solicita ao usuário que digite um número e, em seguida, imprima o logaritmo desse número. Se o usuário digitar um valor negativo, exiba uma mensagem de erro apropriada.
10. Escreva um programa que solicita ao usuário que digite uma palavra e, em seguida, imprima o primeiro caractere dessa palavra. Se a palavra estiver vazia, exiba uma mensagem de erro apropriada.

11. Escreva um programa que cria uma lista com dois elementos e, em seguida, tenta somar essa lista com um número. Se a lista não puder ser somada com um número, exiba uma mensagem de erro apropriada.
12. Escreva um programa que solicita ao usuário que digite um número inteiro e, em seguida, imprima o número menos um. Se o usuário digitar um valor que não é um número inteiro, exiba uma mensagem de erro apropriada.
13. Escreva um programa que cria um dicionário com três chaves e, em seguida, tenta acessar uma chave que não existe. Se a chave não existir, exiba uma mensagem de erro apropriada.
14. Escreva um programa que solicita ao usuário que digite um número e, em seguida, imprima o fatorial desse número. Se o usuário digitar um valor negativo, exiba uma mensagem de erro apropriada.
15. Escreva um programa que cria uma lista vazia e, em seguida, tenta acessar o último elemento da lista. Se a lista estiver vazia, exiba uma mensagem de erro apropriada.
16. Escreva um programa que cria um dicionário vazio e, em seguida, tenta acessar um valor no dicionário usando uma chave que não existe. Se a chave não existir, exiba uma mensagem de erro apropriada.
17. Escreva um programa que solicita dois números ao usuário e, em seguida, imprima o primeiro número dividido pelo segundo. Se o usuário digitar zero, exiba uma mensagem de erro apropriada. Se o usuário digitar um valor que não é um número, exiba uma mensagem de erro apropriada.
18. Escreva um programa que solicita ao usuário que digite um número inteiro e, em seguida, imprima o número elevado ao quadrado. Se o usuário digitar um valor que não é um número inteiro, exiba uma mensagem de erro apropriada.
19. Escreva um programa que cria uma lista com dois elementos e, em seguida, tenta multiplicar essa lista por um número. Se a lista não puder ser multiplicada por um número, exiba uma mensagem de erro apropriada.
20. Escreva um programa que solicita ao usuário que digite uma palavra e, em seguida, imprima o último caractere dessa palavra. Se a palavra estiver vazia, exiba uma mensagem de erro apropriada.
21. Escreva um programa que cria uma lista com três elementos e, em seguida, tenta acessar o elemento em um índice que não existe. Se o índice não existir, exiba uma mensagem de erro apropriada.
22. Escreva um programa que solicita ao usuário que digite um número e, em seguida, imprima o seno desse número. Se o usuário digitar um valor que não é um número, exiba uma mensagem de erro apropriada. Se o módulo não for importado, exiba uma mensagem de erro apropriada.
23. Escreva um programa que solicita ao usuário que digite um número inteiro e, em seguida, imprima o número dividido por um número inteiro aleatório entre 1 e 10. Se o usuário digitar um valor que não é um número inteiro, exiba uma mensagem de erro apropriada. Se o módulo não for importado, exiba uma mensagem de erro apropriada.
24. Escreva um programa que cria uma lista vazia e, em seguida, tenta remover o último elemento da lista. Se a lista estiver vazia, exiba uma mensagem de erro apropriada.
25. Escreva um programa que cria um dicionário com três chaves e, em seguida, tenta remover uma chave que não existe. Se a chave não existir, exiba uma mensagem de erro apropriada.
26. Escreva um programa que solicita ao usuário que digite um número e, em seguida, imprima o logaritmo natural desse número. Se o usuário digitar um valor negativo ou zero, exiba uma mensagem de erro apropriada.
27. Escreva um programa que cria uma lista com três elementos e, em seguida, tenta remover um elemento usando um índice que não existe. Se o índice não existir, exiba uma mensagem de erro apropriada.
28. Escreva um programa que cria um dicionário com três chaves e, em seguida, tenta acessar um valor usando uma chave que não existe. Se a chave não existir, exiba uma mensagem de erro apropriada.
29. Escreva um programa que solicita ao usuário que digite duas strings e, em seguida, concatene essas strings usando o operador de adição (+). Se uma das strings for vazia, exiba uma mensagem de erro apropriada.

## Exercícios de try...except...else...finally

(para os exercícios abaixo, tente usar os 4 blocos de código)

30. Crie um programa que solicite ao usuário que digite um número inteiro. Se o usuário digitar algo que não é um número, o programa deve exibir uma mensagem de erro apropriada. Se for, mostre a soma do número com ele mesmo.

31. Crie um programa que solicite ao usuário que digite um número inteiro positivo. Se o usuário digitar um número negativo ou uma string, o programa deve exibir uma mensagem de erro apropriada. Se for positivo, mostre o número dividido por 2.
32. Crie um programa que solicite ao usuário que digite dois números inteiros. O programa deve dividir o primeiro número pelo segundo. Se o segundo número for zero, o programa deve exibir uma mensagem de erro apropriada.
33. Crie um programa que leia um arquivo e exiba o conteúdo na tela. Se o arquivo não existir, o programa deve exibir uma mensagem de erro apropriada.
34. Crie um programa que solicite ao usuário que digite um número. Se o número for maior que 10, o programa deve exibir uma mensagem de erro apropriada.
35. Crie um programa que solicite ao usuário que digite o nome de um arquivo. O programa deve abrir o arquivo e exibir seu conteúdo na tela. Se o arquivo não existir, o programa deve exibir uma mensagem de erro apropriada.
36. Crie um programa que solicite ao usuário que digite um número. O programa deve verificar se o número é par ou ímpar. Se o usuário digitar algo que não é um número, o programa deve exibir uma mensagem de erro apropriada.
37. Crie um programa que solicite ao usuário que digite um número. O programa deve verificar se o número é divisível por 3. Se o usuário digitar algo que não é um número, o programa deve exibir uma mensagem de erro apropriada.
38. Crie um programa que solicite ao usuário que digite um número inteiro. O programa deve verificar se o número é um múltiplo de 4. Se o usuário digitar algo que não é um número, o programa deve exibir uma mensagem de erro apropriada.
39. Crie um programa que solicite ao usuário que digite uma senha. Se a senha tiver menos de 8 caracteres, o programa deve exibir uma mensagem de erro apropriada.
40. Crie um programa que solicite ao usuário que digite uma idade. Se a idade for menor que 18 anos, o programa deve exibir uma mensagem de erro apropriada.
41. Crie um programa que solicite ao usuário que digite um número. O programa deve verificar se o número está entre 1 e 100. Se o usuário digitar algo que não é um número, o programa deve exibir uma mensagem de erro apropriada.
42. Crie um programa que solicite ao usuário que digite um número. O programa deve verificar se o número é um quadrado perfeito. Se o usuário digitar algo que não é um número, o programa deve exibir uma mensagem de erro apropriada.
43. Crie um programa que solicite ao usuário que digite um número inteiro. O programa deve verificar se o número é primo. Se o usuário digitar algo que não é um número, o programa deve exibir uma mensagem de erro apropriada.
44. Crie um programa que solicite ao usuário que digite o nome de um arquivo. O programa deve ler o arquivo e contar o número de linhas. Se o arquivo não existir, o programa deve exibir uma mensagem de erro apropriada.
45. Crie um programa que solicite ao usuário que digite um número e um operador (+, -, *, /). O programa deve executar a operação e exibir o resultado. Se o usuário digitar um operador inválido ou algo que não é um número, o programa deve exibir uma mensagem de erro apropriada.
46. Crie um programa que solicite ao usuário que digite um número. O programa deve verificar se o número é um número romano válido. Se o usuário digitar algo que não é um número, o programa deve exibir uma mensagem de erro apropriada.
47. Crie um programa que solicite ao usuário que digite uma data no formato dd/mm/aaaa. O programa deve verificar se a data é válida. Se o usuário digitar algo que não é uma data válida, o programa deve exibir uma mensagem de erro apropriada.
48. Crie um programa que solicite ao usuário que digite uma frase. O programa deve contar o número de palavras na frase. Se o usuário digitar algo que não é uma frase válida, o programa deve exibir uma mensagem de erro apropriada.
49. Crie um programa que solicite ao usuário que digite uma string. O programa deve inverter a string e exibir o resultado. Se o usuário digitar algo que não é uma string, o programa deve exibir uma mensagem de erro apropriada.

50. Crie um programa que solicite ao usuário que digite um número. O programa deve calcular o fatorial do número e exibir o resultado. Se o usuário digitar algo que não é um número, o programa deve exibir uma mensagem de erro apropriada.

## Exercícios de raise

51. Crie uma função que solicite ao usuário um número inteiro entre 1 e 10. Se o usuário digitar um número fora deste intervalo, levante uma exceção ValueError.
52. Crie uma função que recebe duas listas como parâmetros. Se as listas tiverem tamanhos diferentes, levante uma exceção ValueError.
53. Crie uma função que recebe um número e verifica se ele é positivo. Se o número for negativo, levante uma exceção ValueError.
54. Crie uma função que recebe uma lista de números e retorna a média aritmética. Se a lista estiver vazia, levante uma exceção ValueError.
55. Crie uma função que recebe um número e verifica se ele é par. Se o número for ímpar, levante uma exceção ValueError.
56. Crie uma função que recebe um texto e verifica se ele contém a palavra "senha". Se a palavra for encontrada, levante uma exceção ValueError.
57. Crie uma função que recebe um dicionário e uma chave. Se a chave não estiver presente no dicionário, levante uma exceção KeyError.
58. Crie uma função que recebe uma lista de strings e verifica se todas as strings têm pelo menos 5 caracteres. Se alguma string tiver menos de 5 caracteres, levante uma exceção ValueError.
59. Crie uma função que recebe um número e verifica se ele é divisível por 3. Se o número não for divisível por 3, levante uma exceção ValueError.
60. Crie uma função que recebe um número inteiro positivo e calcula seu fatorial. Se o número for negativo, levante uma exceção ValueError.
61. Crie uma função que recebe uma string e verifica se ela contém pelo menos um caractere maiúsculo. Se a string não contiver nenhum caractere maiúsculo, levante uma exceção ValueError.
62. Crie uma função que recebe uma lista de números e retorna o produto de todos eles. Se a lista estiver vazia, levante uma exceção ValueError.
63. Crie uma função que recebe um número inteiro e verifica se ele é primo. Se o número não for primo, levante uma exceção ValueError.
64. Crie uma função que recebe uma lista de números e retorna o segundo maior número da lista. Se a lista tiver menos de dois elementos, levante uma exceção ValueError.
65. Crie uma função que recebe uma string e verifica se ela é uma palavra palíndroma. Se a string não for palíndroma, levante uma exceção ValueError.
66. Crie uma função que recebe um dicionário e uma lista de chaves. Se alguma chave não estiver presente no dicionário, levante uma exceção KeyError.
67. Crie uma função que recebe uma lista de números e retorna o menor número da lista. Se a lista estiver vazia, levante uma exceção ValueError.
68. Crie uma função que recebe uma string e verifica se ela contém apenas caracteres alfanuméricos. Se a string contiver algum caractere não alfanumérico, levante uma exceção ValueError.
69. Crie uma função que recebe um número e verifica se ele é um quadrado perfeito. Se o número não for um quadrado perfeito, levante uma exceção ValueError.
70. Crie uma função que recebe uma lista de números e retorna a soma dos números pares. Se a lista estiver vazia, levante uma exceção ValueError.