## **Estrutura TRY...EXCEPTION Objetivo:**

Evitar que o programa quebre com erros inesperados ao lidar com entradas ou operações inválidas.

## **Estrutura básica:**

try:

# Código que pode causar erro

...

except TipoDoErro:

# Código a ser executado se o erro ocorrer

...

## **Exemplo simples: Entrada numérica protegida**

try:

numero = int(input("Digite um número inteiro: "))

print(f"Você digitou: {numero}")

except ValueError:

print("Erro: Você não digitou um número inteiro.")

* Se o usuário digitar "abc", o programa não vai travar. Ele mostra a mensagem do except.

## **Com while para repetir até acertar:**

while True:

try:

idade = int(input("Digite sua idade: "))

break

except ValueError:

print("Erro: Digite apenas números.")

## **Usando except genérico (não recomendado em casos grandes):**

try:

# código

except:

print("Ocorreu um erro.")

**Cuidado**: isso captura **qualquer erro**, até os que você não esperava.

## **Lidando com vários tipos de erro:**

try:

numero = int(input("Digite um número: "))

resultado = 10 / numero

except ValueError:

print("Erro: Entrada inválida, digite um número.")

except ZeroDivisionError:

print("Erro: Não é possível dividir por zero.")

## Dica extra – else e finally:

try:

x = int(input("Número: "))

except ValueError:

print("Valor inválido!")

else:

print("Tudo certo!")

finally:

print("Fim da execução.")

* else: roda **se não houve erro**
* finally: roda **sempre**, com ou sem erro

**Tipos de erros (Exceptions) mais comuns**

### ValueError

**Quando ocorre:** Quando você tenta converter um tipo de dado, mas o valor não é compatível.

int("abc") # Erro

**Mensagem típica:** ValueError: invalid literal for int() with base 10

**Use para validar entradas numéricas.**

### TypeError

**Quando ocorre:** Quando você usa um tipo de dado de forma incorreta.

soma = "texto" + 10 # Erro

**Mensagem típica:** TypeError: can only concatenate str (not "int") to str

**Use para proteger operações entre tipos diferentes.**

### ZeroDivisionError

**Quando ocorre:** Quando você tenta dividir qualquer número por zero.

x = 10 / 0 # Erro

**Mensagem típica:** ZeroDivisionError: division by zero

**Use para verificar divisores antes de calcular.**

### IndexError

**Quando ocorre:** Quando você tenta acessar um índice que não existe em uma lista.

lista = [1, 2, 3]

print(lista[5]) # Erro

**Mensagem típica:** IndexError: list index out of range

**Use para proteger acessos a listas com índices variáveis.**

### KeyError

**Quando ocorre:** Quando você tenta acessar uma chave que não existe em um dicionário.

d = {"nome": "João"}

print(d["idade"]) # Erro

**Mensagem típica:** KeyError: 'idade'

Use .get() ou in para verificar chaves antes de acessar.

print(d.get("idade", "Chave não encontrada"))

### AttributeError

**Quando ocorre:** Quando você tenta usar um método ou atributo que não existe para aquele tipo.

numero = 5

numero.append(10) # Erro

**Mensagem típica:** AttributeError: 'int' object has no attribute 'append'

**Use para verificar o tipo de objeto antes de usar métodos.**

### ImportError / ModuleNotFoundError

**Quando ocorre:** Quando um módulo ou biblioteca não pode ser importado.

import pandasz # Erro

**Mensagem típica:** ModuleNotFoundError: No module named 'pandasz'

**Verifique o nome e se o módulo está instalado.**

### FileNotFoundError

**Quando ocorre:** Quando tenta abrir um arquivo que não existe.

open("arquivo\_inexistente.txt") # Erro

**Mensagem típica:** FileNotFoundError: [Errno 2] No such file or directory

**Use os.path.exists() antes de abrir arquivos.**

### IndentationError (Erro de indentação)

**Quando ocorre:** Quando a indentação (espaços/tabulação) está incorreta.

def funcao():

print("Erro") # Erro

**Mensagem típica:** IndentationError: expected an indented block

**Use sempre 4 espaços por nível. Evite misturar tab e espaço.**

## Dica Final:

Você pode capturar múltiplos erros juntos:

try:

x = int(input("Número: "))

y = 10 / x

except (ValueError, ZeroDivisionError) as e:

print(f"Ocorreu um erro: {e}")

Conclusão:

* Use try/except para proteger seu código de erros em tempo de execução.
* Sempre **especifique o tipo de erro** (como ValueError, ZeroDivisionError).
* Combine com while para forçar repetições até uma entrada válida.