Apostila 4: Python - Exception

1. Introdução às Exceções

Em Python, exceções são erros que ocorrem durante a execução do programa, interrompendo seu fluxo normal. O tratamento de exceções permite que o código lide com erros de maneira controlada, evitando falhas inesperadas.

Exemplo de uma exceção comum:

```
print(10 / 0)  # Gera um erro ZeroDivisionError
```

2. Tratamento de Exceções com try e except

Para evitar que o programa seja interrompido por um erro, utilizamos `try` e `except` para capturar exceções e tratá-las.

```
try:
    resultado = 10 / 0
except ZeroDivisionError:
    print("Erro: divisão por zero não é permitida!")
```

3. Capturando Múltiplas Exceções

Podemos capturar diferentes tipos de exceções usando múltiplos blocos `except`.

```
try:
    numero = int(input("Digite um número: "))
    resultado = 10 / numero
except ZeroDivisionError:
    print("Erro: Não é possível dividir por zero!")
except ValueError:
    print("Erro: Entrada inválida. Digite um número inteiro.")
```

4. Usando finally

O bloco `finally` é executado independentemente de ocorrer uma exceção ou não. Ele é útil para liberar recursos, como fechar arquivos.

```
try:
    arquivo = open("dados.txt", "r")
```

```
    conteudo = arquivo.read()
    print(conteudo)
except FileNotFoundError:
    print("Erro: Arquivo não encontrado!")
finally:
    arquivo.close()
```

5. Levantando Exceções com raise

Podemos gerar exceções manualmente usando `raise`.

```
def dividir(a, b):
    if b == 0:
        raise ValueError("O divisor não pode ser zero!")
    return a / b

try:
    print(dividir(10, 0))
except ValueError as e:
    print("Erro:", e)
```

6. Criando Exceções Personalizadas

Podemos definir nossas próprias exceções criando classes que herdam de `Exception`.

```
class MeuErro(Exception):
    pass

try:
    raise MeuErro("Ocorreu um erro personalizado!")
except MeuErro as e:
    print("Erro capturado:", e)
```

7. Exercícios

1. Escreva um programa que solicite ao usuário um número e trate possíveis erros de entrada.
2. Crie uma função que abra um arquivo e trate a exceção caso o arquivo não exista.
3. Implemente um programa que divida dois números e levante uma exceção se o divisor for zero.

4. Desenvolva uma classe de exceção personalizada e use-a para capturar um erro específico.

Essa apostila cobre os fundamentos do tratamento de exceções em Python. Caso precise de mais detalhes ou exemplos, me avise!