Programação Orientada a Objetos

Tratamento de Exceções

Paulo Viniccius Vieira
paulo.vieira@faculdadeimpacta.com.br

Exceções

- Quando um programa encontra situações não previstas durante sua execução, diz-se que ocorreu uma **exceção** (condição excepcional)
  - Um erro é uma exceção, mas nem toda exceção é um erro

- Se a condição excepcional não é prevista (e tratada), o programa exibe uma mensagem de erro e é interrompido

Exceções

- Um programa bem elaborado precisa detectar exceções e tratá-las para que a execução não seja abortada

- Em Python, é possível **detectar** a ocorrência de uma situação excepcional e **tratar** essa exceção
  - ou seja, definir o que fazer quando ocorre a exceção

Tratando exceções

Para identificar e tratar uma exceção gerada por um trecho de código, pode-se usar a construção **try/except**, conforme a sintaxe abaixo:

```
try:
    comando 1
    comando 2
except TipoDeExceção:
    tratamento da exceção
```

```
try:
    a = int(input("Entre com um numero: "))
    b = int(input("Entre com outro numero: "))
    print(a/b)
except ZeroDivisionError:
    print("Ops, segundo numero não pode ser zero!")
```

Principais Tipos de Exceções

Classe	Descrição
`Exception`	Classe genérica para todas as exceções
`ArithmeticError`	Erros Aritméticos
`AttributeError`	Falha no acesso ou atribuição a atributo de classe
`ZeroDivisionError`	Erro em divisão por zero
`IOError`	Falha no acesso a arquivos
`IndexError`	Índice inexistente em listas ou tuplas
`KeyError`	Chave inexistente de dicionário
`NameError`	Variável Inexistente
`TypeError`	Atribuição de valor a objeto de tipo errado
`ValueError`	Conversão de tipos inválidos
`SyntaxError`	Erro de sintaxe (código errado)

Tratando exceções

A instrução **try** pode ter mais de uma cláusula **except** para diferentes tipos de exceções.

```
try:
    a = int(input("Entre com um numero: "))
    b = int(input("Entre com outro numero: "))
    print(a/b)
except ZeroDivisionError:
    print("Ops, segundo numero não pode ser zero!")
except ValueError:
    print("O valor informado é de um tipo inválido!")
```

```
Entre com um numero: 10
Entre com outro numero: 0
Ops, segundo numero não pode ser zero!
```

```
Entre com um numero: 10
Entre com outro numero: a
O valor informado é de um tipo inválido!
```

Tratando exceções

Se quisermos nos prevenir contra qualquer tipo de erro, podemos usar uma exceção 'genérica', do tipo **Exception**

```
try:
    a = float(input("Entre com um numero: "))
    b = float(input("Entre com outro numero: "))
    print(a/b)
except ZeroDivisionError:
    print("Ooops, divisão por zero")
except ValueError:
    print("Ooops, você não informou o número")
except Exception:
    print("Algo errado aconteceu")
```

Cláusula *else*

É possível completar um comando try com uma cláusula **else** que introduz um trecho de código que só é executado quando nenhuma exceção ocorre:

```
try:
    a = int(input('Informe um número: '))
    b = int(input('Informe outro número: '))
    c = a/b
except ZeroDivisionError:
    print("Erro: divisao por zero")
except Exception:
    print("Algum erro ocorreu")
else:
    print(c)
```

Cláusula *finally*

A cláusula **finally** pode ser usada para assegurar que mesmo que ocorra algum erro (ou não), uma determinada sequência de comandos vai ser executada

```
try:
    a = int(input('Informe um número: '))
    b = int(input('Informe outro número: '))
    c = a/b
except ZeroDivisionError:
    print("Erro: divisao por zero")
except Exception:
    print("Algum erro ocorreu")
else:
    print(c)
finally:
    print("Final do programa")
```

Gerando exceções

Para gerar/provocar uma exceção é usado o comando **raise**, que tem a seguinte forma:

```
raise TipoDaExceção
```

```
try:
    a = int(input('Informe um número Positivo: '))
    if a <= 0:
        raise ValueError
except ValueError:
    print("Erro: O numero deve ser positivo")
except Exception:
    print("Algum outro erro ocorreu")
```

Gerando exceções

- Exemplos:

```
try:
    nota = float(input('Informe uma nota: '))
    if nota < 0.0 or nota > 10.0:
        raise ValueError
except ValueError:
    print("Erro: A nota é invalida")
else:
    if nota >= 6:
        print("Aprovado")
    else:
        print("Reprovado")
```

Gerando validações

```
continua = True
while continua:
    try:
        n = int(input('Informe um número inteiro positivo:'))
        if (n <= 0):
            raise TypeError
    except ValueError:
        print('O valor informado não é do tipo inteiro')
    except TypeError:
        print('O número informado deve ser positivo')
    except Exception:
        print('Ocorreu um erro.')
    else:
        continua = False

print('O numero informado é válido.')
print('O programa continua....')
```