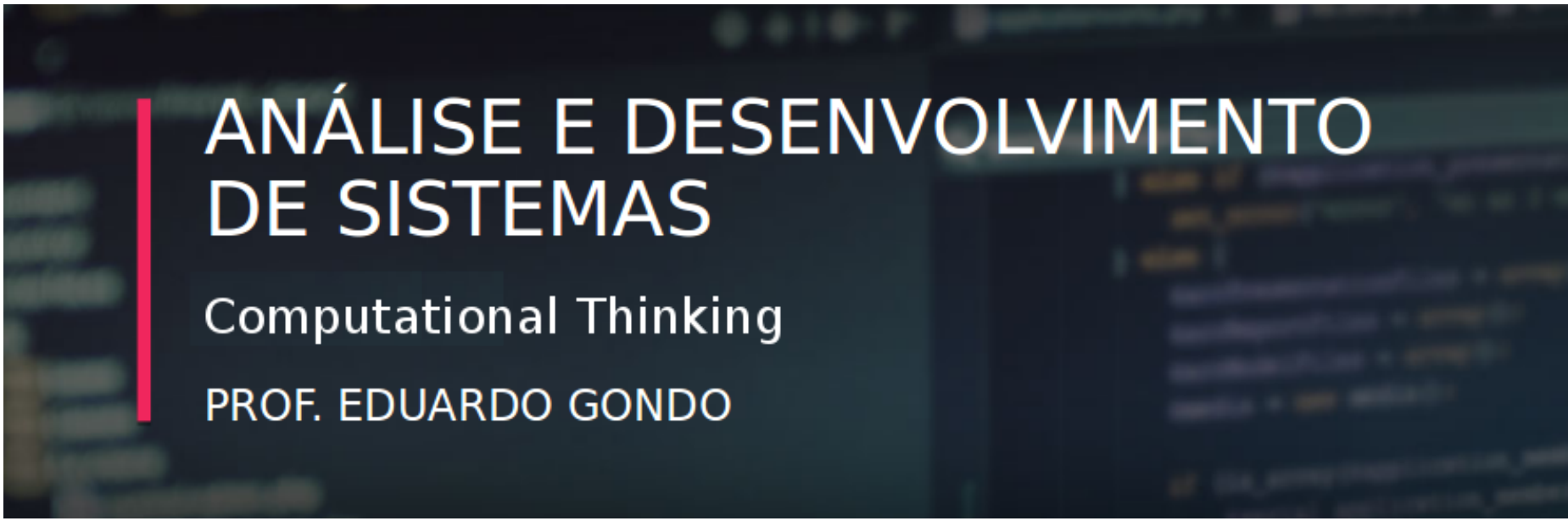
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS
Computational Thinking
PROF. EDUARDO GONDO

O que são exceções?

- Exceções são erros que ocorrem durante a execução dos programas
- Eles são causados por vários fatores como entrada de dados, falha de rede, acesso a recursos (arquivos ou banco de dados), etc.
- Quando uma exceção ocorre, o programa para a execução e uma mensagem de erro aparece
- Python provê uma maneira de manipular as exceções e continuar com a execução do programa se for possível

Por quê devemos tratar as Exceções?

- Tratamento de exceções é uma maneira elegante de recuperar erros
- Ela previne o a quebra do programa e permite que o usuário saiba o motivo do erro
- Ajuda no debug do algoritmo por causa da informação do erro

Bloco try-except

- A sintaxe para tratamento de exceções é o bloco `try-except`
- O bloco `try` contém o código que pode disparar uma exceção
- O bloco `except` contém o código para tratamento da exceção
- Caso uma exceção seja disparada no bloco `try`, o programa pula para o bloco `except` desde que o ele esteja apto a tratar a exceção.

Bloco try-except-finally

- Outro comando que podemos adicionar no bloco try/except é o `finally`
- as instruções dentro do bloco finally sempre são executadas
- útil quando temos recursos que precisam ser fechados ou liberados
- Veja abaixo uma estrutura de exemplo do finally:

```
1  try:
2      #codigo que pode lancar uma excecao
3      ....
4  except:
5      #bloco que e executado caso uma excecao aconteca
6      ....
7  finally:
8      #as instrucoes do finally sempre sao executadas
           independente se houver excecao
9      .....
```

Bloco try-except-else

- Outro comando que podemos adicionar no bloco try/except é o `else`
- as instruções dentro do bloco else serão executadas quando não houver erro
- por exemplo, quando não acontece o erro e queremos mostrar alguma informação para o usuário ou fornecer a resposta do algoritmo
- Veja abaixo uma estrutura de exemplo do else:

```
1  try:
2      #codigo que pode lancar uma excecao
3      ....
4  except:
5      #bloco que e executado caso uma excecao aconteca
6      ....
7  else:
8      #instrucoes a serem executadas quando nao acontece o
           erro
9      .....
```

Exemplo 1: ValueError

```
1  try:
2      a = int(input("Digite um numero: "))
3      print(f"O numero e {a}.")
4  except ValueError:
5      print("Entrada invalida. Digite somente numeros
           inteiros.")
```

Exemplo 2: ZeroDivisionError

```
1  try:
2      a = int(input("Digite um numero: "))
3      b = int(input("Informe outro numero: "))
4      c = a / b
5  except ZeroDivisionError:
6      print("Erro: Divisao por zero")
7  else:
8      print("Resultado:", c)
```

Exemplo 3: KeyError

```
1  d = {'a': 1, 'b': 2}
2  try:
3      x = d['c']
4  except KeyError:
5      print("Error: Key not found in dictionary")
6  else:
7      print("Value:", x)
```

Exemplo 4: Disparando exceções

- como na linguagem Java, nosso programa em Python também pode lançar exceções
- isso é feito através do comando `raise`
- veja abaixo um exemplo:

```
1  def divide(a, b):
2      if b == 0:
3          raise ZeroDivisionError("divisao por zero")
4      return a / b
5
6  try:
7      result = divide(10, 0)
8  except ZeroDivisionError as e:
9      print("Erro:", e)
10 else:
11     print("Resultado:", result)
```

Tratamento de múltiplos exceções

```
1  try:
2      a = int(input("Entre um numero: "))
3      b = int(input("Entre outro numero: "))
4      c = a / b
5  except (ValueError, ZeroDivisionError) as e:
6      print("Erro:", e)
7  else:
8      print("Resultado:", c)
```

Uma outra alternativa

```
1  try:
2      a = int(input("Entre um numero: "))
3      b = int(input("Entre outro numero: "))
4      c = a / b
5  except (ValueError as e:
6      print("Erro:", e)
7
8  except ZeroDivisionError as f:
9      print("Erro: ", f)
10 else:
11     print("Resultado:", c)
```

Nenhuma exceção especificada

```
1  try:
2      a = int(input("Entre um numero: "))
3      b = int(input("Entre outro numero: "))
4      c = a / b
5  except:
6      print("Algum erro aconteceu")
7  else:
8      print("Resultado:", c)
```

Referência Bibliográfica

- Puga e Rissetti - Lógica de Programação e Estrutura de Dados
- Ascêncio e Campos - Fundamentos da Programação de Computadores
- Forbelone e Eberspacher - Lógica de programação: a construção de algoritmos e estruturas de dados
- Documentação do Python - https://docs.python.org/3.8/
- Python Programming For Beginners: Learn The Basics Of Python Programming (Python Crash Course, Programming for Dummies) (English Edition). Kindle
- Python: 3 Manuscripts in 1 book: - Python Programming For Beginners - Python Programming For Intermediates - Python Programming for Advanced (English Edition). Kindle

Copyleft