# Computação II

Departamento de Ciência da Computação – UFRJ

Aula 4

Professora: *Fernanda Duarte Vilela Reis de Oliveira*

2019/01

# Tópicos

1. Tratamento de exceção:
   a. Definições;
   b. Múltiplas exceções;
   c. try – except – else – finally ;
   d. Gerando exceções com raise.

# 1. Tratamento de Exceções

- Exceções são eventos que modificam o fluxo do programa.
- Uma exceção ocorre quando o programa encontra alguma dificuldade de execução devido a alguma condição anormal ou inesperada.
- Erros geram exceções:
  - Tipos de variáveis incompatíveis;
  - Erros aritméticos, como divisão por zero;
  - Arquivos que não existem;
  - Acesso a posições maiores que o tamanho da lista;
- Exceções que não são erros: interrupção com o teclado (ctrl+C), warnings...

# 1. Tratamento de Exceções

- Bloco try/except: tratar a exceção dizendo ao Python para tentar executar um código e identificar se esse código gerou alguma exceção.
- Tem como objetivo identificar erros previstos pelo programador.
- Sintaxe:

```
try:
    # Código que pode gerar uma exceção.
except:
    # Código executado se houver alguma exceção.
else:
    # Código executado se NÃO houver exceção.
finally:
    # Código que sempre é executado.
```

# 1. Tratamento de Exceções

- Bloco try/except: tratar a exceção dizendo ao Python para tentar executar um código e identificar se esse código gerou alguma exceção.
- Tem como objetivo identificar erros previstos pelo programador.
- Sintaxe:

```
try:
    # Código que pode gerar uma exceção.
except 'Nome da exceção que desejamos tratar.':
    # Código executado se houver alguma exceção.
else:
    # Código executado se NÃO houver exceção.
finally:
    # Código que sempre é executado.
```

# 1. Tratamento de Exceções

- Bloco try/except: tratar a exceção dizendo ao Python para tentar executar um código e identificar se esse código gerou alguma exceção.
- Tem como objetivo identificar erros previstos pelo programador.
- Sintaxe:

```
try:
    # Código que pode gerar uma exceção.
except 'Nome da exceção que desejamos tratar.':
    # Código executado se houver alguma exceção
    # - onde o erro pode ser informado ou mesmo tratado.
else:
    # Código executado se NÃO houver exceção.
    # - tudo o que deve ser executado se o fluxo do código for
    #   de acordo com o desejado.
finally:
    # Código que sempre é executado.
```

# 1. Tratamento de Exceções

- Bloco try/except: tratar a exceção dizendo ao Python para tentar executar um código e identificar se esse código gerou alguma exceção.
- Tem como objetivo identificar erros previstos pelo programador.

Código que pode gerar uma exceção

```
try:
    f = open('teste.txt')
except:
    print('O arquivo nao existe.')
```

# 1. Tratamento de Exceções

- Bloco try/except: tratar a exceção dizendo ao Python para tentar executar um código e identificar se esse código gerou alguma exceção.
- Tem como objetivo identificar erros previstos pelo programador.

```
try:
    f = open('teste.txt')
except:
    print('O arquivo nao existe.')
```

Tratamento da exceção

# 1. Tratamento de Exceções

- Bloco try/except: tratar a exceção dizendo ao Python para tentar executar um código e identificar se esse código gerou alguma exceção.
- Tem como objetivo identificar erros previstos pelo programador.

```
try:
    f = open('teste.txt')
except:
    print('O arquivo nao existe.')
```

Utilizar somente *except* é muito impreciso. Qualquer tipo de erro pode gerar essa exceção.

# 1. Tratamento de Exceções

```
try:
    print(a)
    f = open('teste.txt')
except:
    print('O arquivo nao existe.')
```

Se *a* não existir, uma exceção será gerada antes da linha para abertura do arquivo.

O usuário pode interromper o programa antes de entrar com um valor, o que vai gerar uma exceção antes da linha para abertura do arquivo.

```
try:
    a = eval(input())
    print('Input executado corretamente')
    f = open('teste.txt')
except:
    print('O arquivo nao existe.')
```

# 1. Tratamento de Exceções

- Adicionar exceções mais específicas.
- É possível adicionar mais de um *except* para o mesmo *try* – múltiplas exceções.

```
try:
    print(a)
    f = open('teste.txt')
except IOError:
    print('O arquivo nao existe.')
except NameError:
    print('Uma variavel nao foi definida.')
```

Erro ao abrir o arquivo

Erro ao acessar uma variável inexistente.

# 1. Tratamento de Exceções

- Exceções mais genéricas devem estar no final da lista, pois após executar uma exceção, o Python desconsidera as demais.

```
try:
    print(a)
    f = open('teste.txt')
except IOError:
    print('O arquivo nao existe.')
except NameError:
    print('Uma variavel nao foi definida.')
except:
    print('Erro indefinido')
```

Exceção sem especificação no final

# 1. Tratamento de Exceções

- Imprimindo a informação da exceção gerada pelo Python:

```
try:
    a = eval(input())
    f = open('teste.txt')
except IOError as excecao:
    print(excecao)
except NameError as excecao:
    print(excecao)
except Exception as excecao:
    print(excecao)
except:
    print('Excecao inesperada.')
```

*excecao* será uma string definida pelo python que depende da exceção gerada.

# 1. Tratamento de Exceções

- try – except – else –finally

```
try:
    a = eval(input())
    print(a)
except NameError:
    print('Uma variavel nao foi definida.')
except IOError:
    print('O arquivo nao existe.')
else:
    print('Else eh executado quando nenhuma\n\
    excecao eh disparada.')
finally:
    print('Sempre eh executado')
```

# 1. Tratamento de Exceções

- try – except – else –finally

```
try:
    a = eval(input())
    print(a)
except NameError:
    print('Uma variavel nao foi definida.')
except IOError:
    print('O arquivo nao existe.')
else:
    print('Else eh executado quando nenhuma\n\
    excecao eh disparada.')
finally:
    print('Sempre eh executado')
```

Else: se nenhuma exceção for disparada, o conteúdo de else é executado.

# 1. Tratamento de Exceções

- try – except – else –finally

Finally: sempre é executado. É útil para garantir a liberação de espaço de memória, o fechamento de arquivos....

```
try:
    a = eval(input())
    print(a)
except NameError:
    print('Uma variavel nao foi definida.')
except IOError:
    print('O arquivo nao existe.')
else:
    print('Else eh executado quando nenhuma\n\
    excecao eh disparada.')
finally:
    print('Sempre eh executado')
```

# 1. Tratamento de Exceções

- Gerando exceções personalizadas: *raise*

```
try:
    a = eval(input())
    if type(a) != str:
        raise TypeError
except TypeError as excecao:
    print('Somente variaveis do tipo str sao aceitas.')
except NameError as excecao:
    print(excecao)
except Exception as excecao:
    print(excecao)
except:
    print('Excecao inesperada.')
```

Gera uma exceção do tipo TypeError apesar do Python não ter gerado essa exceção automaticamente.

# 1. Tratamento de Exceções

- Gerando exceções personalizadas: *raise*

```
try:
    a = eval(input())
    if type(a) != str:
        raise TypeError('Somente variaveis do tipo str sao aceitas.')
except TypeError as excecao:
    print(excecao)
except NameError as excecao:
    print(excecao)
except Exception as excecao:
    print(excecao)
except:
    print('Excecao inesperada.')
```

Incluindo a mensagem de erro ao gerar a exceção.

# 1. Tratamento de Exceções

- Gerando exceções personalizadas: *raise*

```
try:
    a = eval(input())
    if type(a) != str:
        raise TypeError('Somente variaveis do tipo str sao aceitas.')
except TypeError as excecao:
    print(excecao)
except NameError as excecao:
    print(excecao)
except Exception as excecao:
    print(excecao)
except:
    print('Excecao inesperada.')
```

As exceções geradas por *raise* devem ser instâncias do tipo *exception* ou subclasses de *Exception*.

# 1. Tratamento de Exceções

- Gerando exceções personalizadas: *raise*

```
class myError(Exception):
    pass
```

Novas exceções podem ser definidas criando uma classe herdeira de Exception.

```
try:
    a = eval(input())
    if type(a) != str:
        raise myError('Somente variaveis do tipo str sao aceitas.')
except myError as excecao:
    print(excecao)
except NameError as excecao:
    print(excecao)
except Exception as excecao:
    print(excecao)
except:
    print('Excecao inesperada.')
```