Exception & Assertion

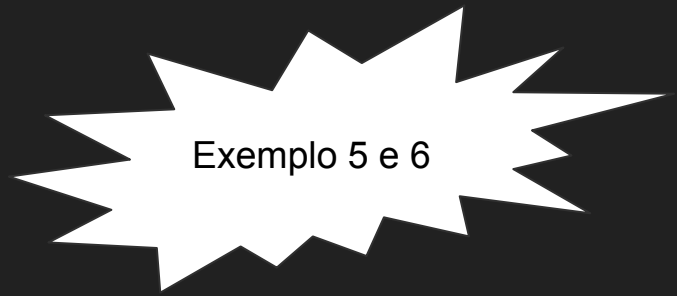
Exception

- Indicam a ocorrência de um evento inesperado durante a execução do código.
- Tratam eventos cuja ocorrência não seja frequente mas que comprometem a execução do código.
- Assim, permitem aos programadores tratarem tais eventos de forma adequada.
- Os eventos podem estar relacionados a erros ou situações inesperadas.
  - L=[1,2,3] print(L[3]) ⇒ Erro: `IndexError`
  - L=[] print(10/len(L)) ⇒ Erro: `ZeroDivisionError`
  - `'x'/5` ⇒ Erro: `TypeError`

Exception

- Formas de lidar com eventos inesperados:
  - `try/exception`
  - `try/exception/else`
  - `try/finally`
- `try` : habilita o tratamento da exceção ao conter o bloco de instruções que podem causar exceções.
- `exception` : determina possíveis causas para as exceções, podendo especificar um identificador para aquele tipo de erro.
- `else` : contém instruções a serem executadas, caso nenhuma exceção seja detectada no bloco `try`.
- `finally` : Um bloco finally define instruções a serem executadas independente da ocorrência ou não de exceções.

Raise Exception



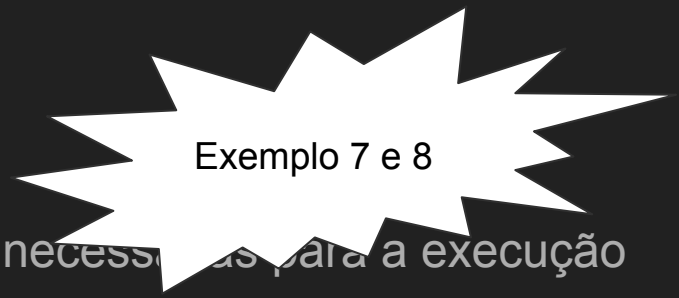
- Permite ao desenvolvedor definir uma exceção
- O programa termina a execução: `raise <nome da exceção> (<argumentos>)`

```
raise ValueError('deu ruim')
```

- O programa não termina a execução:

```
try: ....
    raise ValueError
      ....
except ValueError:
          print('deu ruim')
```

Assertion



- Utilizado para avaliar se determinada condição necessárias para a execução do código ocorre.
- Permite testar se tal condição é verdadeira. Se for falsa, dispara `AssertionError`
- No caso da condição ser falsa, uma mensagem de erro pode ser retornada.

```
assert <condição> , <mensagem>
```