Computação II - Python
Aula 5 - Tratamento de exceções

Carla A. D. M. Delgado

João C. P. da Silva

Dept. Ciência da Computação - UFRJ

Tratamento de exceções

Exceções na programação

- Em programação, o termo *exceção* se refere a um erro que acontece durante a **execução** de um programa ou trecho de código.

```
In [1]: 10 * (1/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero

In [2]: 4 + nun*3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'nun' is not defined

In [3]: '2' + 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object to str implicitly
```

Tratamento de exceções

Exceções na programação

```
In [1]: 10 * (1/0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ZeroDivisionError: division by zero

In [2]: 4 + nun*3
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'nun' is not defined

In [3]: '2' + 2
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object to str implicitly
```

- A última linha da mensagem de erro indica o que aconteceu.
- Existem diferentes tipos de exceções classificados (ZeroDivisionError, NameError, TypeError...). Essas exceções são ditas *built-in*.

Tratamento de exceções

Exceções na programação

- Em programação, o termo *exceção* se refere a um erro que acontece durante a **execução** de um programa ou trecho de código.
- As *exceções* denotam sitações que fogem o que seria esperado para o funcionamento do programa.
- Os mecanismos de tratamento de exceção são construtos que algumas linguagens de programação oferecem para lidar sistematicamente com erros, principalmente os já previstos.

Tratamento de exceções

Exceções na programação

- Algumas exceções podem ser previstas;

```
1
2 ...
3 n = int(input("Entre com um numero inteiro: "))
4 ...
```

- De forma que a execução do programa não seja interrompida abruptamente :-/

```
1 Entre com um numero: 23.5
2 Traceback (most recent call last):
3   File "<stdin>", line 1, in <module>
4 ValueError: invalid literal for int() with base 10: '23.5'
```

Tratamento de exceções

Exceções na programação

- O mecanismo de tratamento de exceções **deve** ser usado para tratar exceções previstas;

```
...
while True:
    try:
        x = int(input("Entre com um numero inteiro: "))
        break
    except ValueError:
        print("Opa!  Isso nao eh um inteiro valido.  Tente novamente...")
print("A entrada foi validada.")

...
```

- De forma que a execução do programa não seja interrompida abruptamente :-)

```

...
Entre com um numero inteiro: abc
Opa!  Isso nao eh um inteiro valido.  Tente novamente...
Entre com um numero inteiro: 42.0
Opa!  Isso nao eh um inteiro valido.  Tente novamente...
Entre com um numero inteiro: 42
A entrada foi validada.
...
```

Tratamento de exceções

Mecanismos de tratamento de exceções no Python

- O código que tem o risco de causar uma exceção deve estar dentro de uma estrutura chamada bloco *try*;
- Um bloco *exception*, pareado com o bloco *try*, conterá o código que deve fazer o tratamento da exceção, ou seja, aquilo que você programador acha que deve fazer para contornar o problema ocorrido.

Tratamento de exceções

O mecanismo de tratamento de exceções em Python

- Primeiro, o bloco de comandos associados ao *try* é executado.
- Caso nenhuma exceção ocorra, os comandos associados à cláusula *except* **não** são executados.
- Caso uma exceção ocorra durante a execução do bloco associado ao *try*:
  - Os demais comandos do bloco *try* **não** serão feitos.
  - Se a exceção que aconteceu foi do tipo de exceção citada na cláusula *except*, então o bloco de comandos associados ao *except* será executado.
  - Se a exceção não for do mesmo tipo citado na cláusula *except*, busca-se passar para um bloco *try* mais externo, caso haja.
  - Se não houver nenhum, dizemos que a exceção não está tratada e a execução do programa é interrompida.
- Depois, a execução do programa continua normalmente, após o *except*.

Tratamento de exceções

Exceções na programação - Estudo dirigido

- Analise as funções *divi1* e *divi2*. Elas fazem a mesma coisa? Quais as principais diferenças entre elas?

```
1
2  def divi1(n1, n2):
3      return n1/n2
4
5  def divi2(n1, n2):
6      try:
7          return float(n1)/float(n2)
8      except ZeroDivisionError:
9          print("divisao por zero, operacao invalida")
10         return None
```

- Verifique qual seria o resultado de cada uma das funções para as seguintes entradas:
  - $1, 2$
  - $6.7, 2$
  - $"a", 2$
  - $2, 0$
  - $0, 2$

Tratamento de exceções

Exceções na programação - Estudo dirigido

- Podemos ter mais de uma cláusula *exception* para uma mesma cláusua *try*

```
1
2  def divi1(n1, n2):
3      return n1/n2
4
5  def divi2(n1, n2):
6      try:
7          return float(n1)/float(n2)
8      except ZeroDivisionError:
9          print("divisao por zero, operacao invalida")
10         return None
11     except ValueError:
12         print("a funcao trabalha com duas entradas numericas")
13         return None
```

- Verifique qual seria o resultado da função div2 novamente para as entradas:
  - $1, 2$
  - $6.7, 2$
  - $"a", 2$
  - $2, 0$
  - $0, 2$

Tratamento de exceções

Exceções na programação

A lista completa de exceções *built-in* (exceções catalogadas) pode ser vista na documentação do Python, em: https://docs.python.org/3/library/exceptions.html

Alguns exemplos:

- ImportError
- IndexError
- TypeError
- ValueError
- ZeroDivisionError

Tratamento de exceções

Exceções na programação - Exercício

- Faça uma função que receba como entrada uma lista de números, e um número inteiro n. Sua função deve retornar o elemento da lista que esteja na posição indexada por n.
- Certifique-se que sua função não tenha sua execução interrompida caso n não seja um índice válido da lista. Nesse caso, sua função deveria retornar "None".

Tratamento de exceções

Os mecanismos de tratamento de exceções da programação, em geral

- salvam o estado da execução do programa no momento em que o problema ocorre;
- interrompem o fluxo normal de execução do programa;
- iniciam a execução de uma função ou pedaço do código especialmente feito para lidar com a situação de erro (conhecido como tratador de exceções - *exception handler*;
- dependendo do tipo de erro que foi identificado, o tratador pode contornar o problema e o programa pode ser continuado a seguir, com as informações salvas anteriormente.

Computação II - Python
Aula 5 - Tratamento de exceções

Carla A. D. M. Delgado

João C. P. da Silva

Dept. Ciência da Computação - UFRJ