Aula 11 - Tratamento de Erros e Exceções

June 10, 2021

1 Tratamento de Erros e Exceções

Segundo a documentação do Python há pelo menos dois tipos distintos de erros: erros de sintaxe e exceções.

Os erros de sintaxe apontam para a linha de nosso script e, normalmente são fáceis de se corrigir, veja um exemplo abaixo. Neste erro eu esqueço de iniciar a string com o sinal de aspas ".

[17]:
```
# Erro -> Ocasionado pelo programador -> Código não compila por erro nosso na␣
↪hora de programar
print(Esqueci o sinal de aspas")
```

```
  File "<ipython-input-17-a611b03e9f14>", line 2
    print(Esqueci o sinal de aspas")
                  ^
SyntaxError: invalid syntax
```

Por outro lado, mesmo que seu código esteja sintaticamente correto, é possível ocorrer um erro em tempo de execução. Também denominadas simplesmente por exceções.

O tratamento dessas exceções devem nos auxiliar a entender e depurar o código.

Veja abaixo um exemplo de exceção não tratada.

[16]:
```
# Exceção -> "Não dava" para prever -> Foi algo acidental -> Foi algo␣
↪intencional -> Sintaxe "Correta"
numero1 = 1 # Input
numero2 = 0 # Input

def divisao():
    return numero1 / numero2

divisao()
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-16-e461342a3391> in <module>
      6     return numero1 / numero2
```

```
      7
----> 8 divisao()

<ipython-input-16-e461342a3391> in divisao()
      4
      5 def divisao():
----> 6     return numero1 / numero2
      7
      8 divisao()

ZeroDivisionError: division by zero
```

Para evitar exceções nós podemos realizar tratamentos prévios.

Sendo assim, tentamos prever o erro e evitar que ele ocorra ou ao menos avisar ao usuário sobre o erro.

Para isso usaremos **try** e **except**, onde:
try: tentar executar uma ação
except: caso der erro, entra no except

[21]:
```
try: # Tentar
    print(1/0)
except ZeroDivisionError: # Deu erro -> Deu ruim!
    print("Por favor, não divida por zero!")
```

```
Por favor, não divida por zero!
```

Para executar apenas se não der erro podemos colocar o bloco **else**.
Else só entrará caso não entrar no **except**.

[35]:
```
try:
    n = int(input("Número: ")) # Tento isso
except ValueError:
    print("Coloque apenas números inteiros") # Se der erro, eu mostro isso
else:
    print(n) # Por fim, se não der erro eu faço isso
```

```
Número:
Coloque apenas números inteiros
```

Podemos tratar mais de uma exceção ao mesmo tempo

[41]:
```
try:
    n = int(input("Numero: "))
    m = int(input("Numero: "))
    x = n / m
except ZeroDivisionError:
    print("Não divida por zero")
except ValueError:
```

```
    print("Insira apenas números inteiros")
```

```
Numero: 1
Numero: a
Insira apenas números inteiros
```

[43]:
```
try:
    n = int(input("Numero: "))
    m = int(input("Numero: "))
    x = n / m
except (ZeroDivisionError,ValueError):
    print("Erro: Coloque números inteiros diferentes de zero!")
```

```
Numero: 1
Numero: a
Erro: Coloque números inteiros diferentes de zero!
```

O **finally** sempre executa no final do tratamento, tenha ele entrado no except ou não

[52]:
```
try:
    n = int(input("Numero: "))
    m = int(input("Numero: "))
    x = n / m
except (ZeroDivisionError,ValueError):
    print("Erro: Coloque números inteiros diferentes de zero!")
else:
    print("O valor de X é",x)
    print("Eu não tive erro dividindo")
finally:
    print("Finalmente eu terminei essa execução")
```

```
Numero: 1
Numero: 1
O valor de X é 1.0
Eu não tive erro dividindo
Finalmente eu terminei essa execução
```

1.1 Desafios do Try/Except

1.1.1 Desafio de arrumar a calculadora

[57]:
```
class Calculadora(object):
    def soma(self,primeiro_valor,segundo_valor):
        return primeiro_valor + segundo_valor;

    def subtracao(self,primeiro_valor,segundo_valor):
        return primeiro_valor - segundo_valor

    def divisao(self,primeiro_valor,segundo_valor):
        try:
```

```
            return primeiro_valor / segundo_valor
        except ZeroDivisionError:
            print("Você não pode dividir por zero.")

    def multiplicacao(self,primeiro_valor,segundo_valor):
        return primeiro_valor * segundo_valor
```

[61]:
```
calc = Calculadora()
calc.divisao(2,1)
```

[61]: 2.0

1.1.2 Desafio de arrumar a classe Alunos

[88]:
```
class Aluno(object):
    def __init__(self, nome):
        self.nome = nome

    def inserir_notas(self,nota1,nota2):
        try:
            self.nota1 = float(nota1)
            self.nota2 = float(nota2)
        except ValueError:
            print("As notas devem ser números reais")

    def calcular_media(self):
        try:
            return (self.nota1 + self.nota2) / 2
        except AttributeError:
            print("Não é possível calcular a média até que tenha inserido todas␣
 ↪as notas")

    def mostrar_informacoes(self):
        try:
            if(self.calcular_media() >= 6):
                print(f"O aluno {self.nome} foi aprovado")
            else:
                print(f"O aluno {self.nome} não foi aprovado")
        except TypeError:
            print("Não é possível mostrar informações até que tenha calculado a␣
 ↪média")
```

[91]:
```
felipe = Aluno('Felipe')
felipe.inserir_notas(10,9.5)
felipe.mostrar_informacoes()
```

```
O aluno Felipe foi aprovado
```

[ ]: