Aula 23 - Tratamento de Erros e Exceções
20/07/2022 - 15:51

Erros acontecem...
A diferença está na forma como você lida com eles.
E sobre lidar, a programação nos apresenta jeitos e formas de encarar esse lado da vida tão duríssimo 😭.

Vamos aos exemplos:
primt(x) -> Clássico erro de sintaxe
Mas, como já foi visto até aqui, a programação não lida somente com erros de sintaxe.

Caso ainda seja realizado o comando print(x), se a variável x não for definida, o programa também apresentará erro (semântico). Isto deixa claro como os erros de sintaxe não são os únicos, mas somente alguns dos mais diversos tipos.

Porém, quando um erro que não sintático ocorre, é dado a ele o nome de exceção. Neste caso acima, seria uma exceção NameError.

```
n = int(input('Digite um número: ')
print(f'Você digitou o número {n}')
```

Caso o usuário digite uma string, o programa mostrará outra exceção, chamada de ValueError, pois é esperado um input de um valor inteiro, e não de uma string, mesmo que seja escrito um número por extenso.

```
Elst = [2, 3, 4]
print(Elst[3])
```

Apresenta a exceção IndexError. Se for dicionário, KeyError

Até então, todas as aulas mostraram a figura do programador todo-poderoso, que comanda e, a partir disto, cria seus algoritmos. Porém, o tratamento de erros e exceções instiga o indivíduo a mudar esta postura e inaugurar, para si mesmo, outra: a que testa, a que investiga opções dentro do código e se articula com a própria linguagem.

Esta nova postura toma a seguinte forma:

```
try:
except:
```

“try” pede ao programa que execute o desejado e “except” apresenta uma exceção criada pelo programador.
Em esquema, temos que:

```
try:
    operação
except:
    falhou
else:
    deu certo
```

Se o programa funcionar, o “else” se apresenta para mostrar o que acontece caso o programa funcione corretamente.

Vamos aos exemplos novamente:

```
try:
    a = int(input(‘Digite o numerador ‘)
    b = int(input(‘Digite o denominador’)
    r = a/b
except:
    print(‘Infelizmente ocorreu um erro T-T)

print(f’O resultado da divisão é {r}’)
```

Bom, se o programa tiver somente “try” e “except”, o próprio PyCharm reclamará dizendo que a, b e r não foram definidos. Ainda assim, se o programa funcionar corretamente, nada será apresentado. Porém, se for posto um 0 no denominador propositalmente, além da mensagem de exceção definida no programa, a IDE apresentará um problema de que r não pode ser definido, e não de divisão por 0.
Para que funcione da maneira desejada, o programa precisa ter o “else:”

```
try:
    operação
except:
    falhou
else:
    deu certo
finally:
    deu certo/deu errado
```

“Finally” mostra algo independentemente do programa apresentar falha ou funcionar corretamente, como uma mensagem “Volte sempre!”.

Bom, mas, testando na IDE, você percebeu que expressão “except” é vista

como muito ampla. Então, para mostrar na mensagem, vamos especificar:

```
try:
	a = int(input('Digite o numerador ')
	b = int(input('Digite o denominador')
	r = a/b
except Exception as erro:
	print(f'Infelizmente ocorreu um erro T-T\nTipo: {erro.__class__}')
else:
	print(f'O resultado da divisão é {r}')
finally:
	print('Volte sempre!')
```

except Exception as erro: Nos diz que atribuirá à variável erro a Exception dada pelo Python. Quando, no print, uso "erro.__clas__", o programa mostra a classe da exceção apresentada.

Outra forma de especificação é escrever: except TypeError: para quando a exceção estiver relacionada ao tipo de elemento inserido no programa (str, int, etc.).
Em um programa pode haver mais de um except. Posso criar excepts para cada tipo de erro, criando uma mensagem personalizada

```
try:
	operação
except TypeError:
	falhou
except ValueError:
	falhou
except OSError:
	falhou
else:
	deu certo
finally:
	deu certo/falhou
```

Pode utilizar () para colocar mais de um erro ou usar o except genérico para ver qual foi o tipo de erro obtido

Sobre o exercício 115:

Esse é o último exercício do mundo 3. Não temos continuação a partir daí 😭
Porém, ainda existem lições neste exercício que merecem uma menção.

O exercício consistia em criar um mini-sistema de cadastro de indivíduos, que coletaria dados como nome e idade.
Criou-se um pacote dentro de uma pasta lib que está dentro da pasta do

exercício 115. No momento da importação, usou-se asterisco para não ter que colocar o nome do pacote em todas as vezes em que fosse utilizada uma função.

from ex115.lib.interface import *

Criar e analisar arquivos em python pode parecer complicado, simplesmente pelo fato de que estou aprendendo isto durante a resolução de um exercício. Porém, não é. Vejamos:

1. Teste se o arquivo existe em seu programa
   É super fácil ver se algum arquivo existe. Primeiro, use a função open(filename, mode), que tenta justamente abrir um arquivo com o nome posto como parâmetro string, de acordo com um modo, também em string, que especifica se você quer ler (r), ler arquivo de texto (rt) ou só adicionar alguma string no final da linha (a) - existem mais modos, obviamente. Arquivos em texto aceitam somente strings, lembre-se disso.

```
def arquivo_existe(filename):
    try:
        a = open(filename, 'rt')
        a.close()
    except FileNotFoundError:
        return False
    else:
        return True

# Programa Principal

arq = 'lista.txt'
if arquivo_existe(arq):
    print(f'O arquivo solicitado existe')
else:
    print(f'O arquivo não foi encontrado')
```

E por que isso?
Simples: caso não exista arquivo nenhum, o sistema criará um novo arquivo.

```
def criar_arquivo(nome):
    try:
        a = open(nome, 'wt+')
    except:
        print('Houve um problema durante a criação do arquivo')
    else:
        print(f'Arquivo \"{nome}\" criado com sucesso')
```

Bom, vamos entender o que ocorreu com o open:

- nome é o parâmetro que recebe a string com o nome do arquivo;
- ‘w’ indica que algo será escrito no arquivo. O arquivo é de texto ’t’, logo ‘wt’;
- ‘+’ significa que, se o arquivo não existir, um novo será criado com o mesmo nome definido.

Entendido? Que bom...
No programa principal....

```
arq = ‘lista.txt’
if not arquivo_existe(arq):
        criar_arquivo(arq)
```