Essa é uma versão dos Guias de Estudo desenvolvida para auxiliar o processo de impressão e para facilitar a leitura dos guias por programas que fornecem a leitura automatizada para suporte a todos os alunos que necessitem. Dessa forma, não apresentaremos ilustrações nesse arquivo.* *#ParaTodosLerem****.*

Módulo #6

Guias de estudo

Módulo #6

Até o momento foram apresentados conteúdos sobre lógica e técnicas de programação. Você deve ter percebido em todos os códigos que implementamos que armazenamos dados em alguma estrutura como ***variável****,* ***lista****,* ***dicionário****, entre outras; armazenam esses dados no momento da execução e, quando fechamos o programa e abrimos, os dados não estão mais disponíveis? Isso ocorre porque guardamos estes dados em uma memória temporária. Para que possamos acessar os dados quando quisermos é necessário armazenar na memória física do computador e podemos fazer isso por meio do uso de bancos de dados (que veremos mais a seguir) e uma outra opção é utilizar arquivos. Como foi visto em aula, os arquivos podem ser criados, lidos e alterados.*

Um outro problema comum que ocorre quando estamos desenvolvendo um programa são os famosos ***erros****. Quem aqui nunca passou horas tentando descobrir o motivo de um erro ter ocorrido? Parece que fiz tudo certo e o Python apresenta erro, como assim?*

Bom, temos alguns tipos de erros. Os ***erros de sintaxe*** *ocorrem quando escrevemos algo errado. Neste caso pode ser um alinhamento incorreto, a falta de uma vírgula ou de dois pontos, escrita errada, entre outros.*

Um outro tipo de erro que pode ocorrer é um erro de lógica, que é quando o Python não apresenta um erro porém a resposta do programa não é aquilo que esperávamos. Isso acontece porque algo na lógica de programação não ficou correto.

Sempre que desenvolvemos algum programa precisamos pensar que o usuário pode errar no momento da iteração com o sistema. Por exemplo: suponha que vamos criar uma calculadora que realiza operações entre dois números informados pelo usuário. Então, a entrada esperada são dois valores inteiros ou float. Imagine que houve um erro de digitação e o usuário informou uma letra. Logo, o Python vai apresentar um erro porque não é possível realizar tal operação matemática.

*Esse é um exemplo de **erro de exceção**, que é quando o programa está correto mas ocorre um erro durante sua execução .*

Vale lembrar que as mensagens de erro retornadas pelo Python não serão compreendidas pelo usuário então precisamos tratar os erros e deixar mensagens claras para o usuário compreender.

Aí vai uma dica muito importante:

Sempre que estiver desenvolvendo um programa pense em todas as possibilidades de erro que um usuário possa cometer e trate todos eles.

Como fazer o tratamento?

Em Python usamos o Try...Except..Else..Finally. Veja como você pode utilizar estas funções:

> O compilador executa o código que está no bloco try, se ocorrer um erro, o bloco except é acionado, onde o erro deve ter sido tratado. Por isso, no bloco except você deve inserir as mensagens de erro.

> O bloco finally é executado se houver ou não um erro no bloco try. Nele você pode inserir mensagens do tipo "Programa finalizado" ou algum código que encerra o programa.

> O bloco else, usado depois do try, é executado se não houver algum erro. Ou seja, se algo der errado no código que está no bloco try, except é acionado, se não houver erro, else é acionado. No bloco else você pode colocar mensagens do tipo: "programa executado com sucesso", exibir resultados, entre outros.

Vamos ver um exemplo:

```
try:
 x=float(input('Informe o primeiro valor: '))
 y=float(input('Informe o segundo valor: '))

 operacao= input('Digite: \n + para somar \n - para subtrair \n * 
para multiplicar ou \n / para dividir')

 if operacao =='+':
   resultado = x + y

 elif operacao =='-':
   resultado = x - y
```

```
 elif operacao =='*':
   resultado = x * y

 elif operacao =='/':
   resultado = x / y

 else:
   print('Informe uma operação válida!')

except ValueError:
 print("Dados de entrada incorretos")

except ZeroDivisionError:
 print("divisão por zero!")

else:
 print("Resultado: ", resultado)

finally:
 print("Obrigada por utilizar nosso sistema!")
```

Neste exemplo implementamos uma calculadora no bloco try em que o usuário deve informar dois números e a operação que deseja realizar. Três tipos de erro podem ocorrer:

1. *O usuário informar uma operação diferente de **+**, **-**, *****, **/**. Este erro pode ser facilmente tratado na condicional else dentro do Try.*
2. *O usuário pode informar outro tipo de dado no momento de informar os números. Este erro foi tratado como uma exceção em `except ValueError`*
3. *Na matemática não se pode dividir um número por zero e caso ocorra. O erro foi tratado como uma exceção em `except ZeroDivisionError`*

Se tudo ocorrer conforme esperado no bloco Try, o bloco else será executado e o resultado será mostrado na tela. Independente se deu erro ou não, o bloco finally é executado e mostrará a seguinte mensagem: `Obrigada por utilizar nosso sistema!`

Lembre-se

Sempre que estiver desenvolvendo um sistema pense em todas as possibilidades de erro que podem ocorrer no momento da iteração entre o programa e o usuário e trate todos eles.