Computational Thinking With Python

Prof. Gilberto Alexandre das Neves
profgilberto.neves@fiap.com.br

Tratamento de Erros

Tratamento de erros

Opcionalmente podemos utilizar no bloco **try-except** o bloco **finally**. Comandos no bloco **finally** serão executados independentemente se ocorreu erro ou não.

Podemos utilizar o comando **raise** para “jogar” (*throw* ou *raise*) uma exceção se uma determinada condição ocorra.

Exemplo 1

FUPQ peça para o usuário digitar dois números inteiros e exiba os resultados divisão realizas com estes números (faça verificação para permitir apenas números inteiros e o segundo número digitado não pode ser menor ou igual a zero). Exiba ao final uma mensagem de despedida independente se ocorreu um erro ou não.

Exemplo 1

```
try :
    print("Digite 2 números inteiros")
    num1 = int(input())
    if not type(num1) is int:
        raise ValueError
    num2 = int(input())
    if not type(num2) is int:
        raise ValueError
    if num2 <= 0 :
        raise ZeroDivisionError
    print(f"Divisão: {num1//num2}")
except ValueError :
    print("Erro: Somente números inteiros são permitidos")
except ZeroDivisionError :
    print("Erro: Segundo número deve ser maior do que zero")
finally :
    print("Obrigado por utilizar nosso programa. Volte em breve!")
```

Módulo RegEx

Python RegEx

Uma **RegEx**, ou **Regular Expression**, é uma sequencia de caracteres que formam um determinado padrão desejado.

Podemos usar **RegEx** para checar se uma ***string*** possui um determinado padrão definido.

Para utilizarmos **RegEx** devemos usar o comando **import re** no início do código.

Funções do RegEx

Ao utilizar o **RegEx** (**import re**) temos acesso a um conjunto de funções que nos permite procurar ocorrências de texto em um objeto ***string***.

Função	Descrição
findall	Retorna um **List** contendo todas as ocorrências procuradas.
search	Retorna **Match** se existe a ocorrência buscada em qualquer parte da ***string***
split	Retorna um **List** de ***strings***, dividindo o texto original na ocorrência do caractere informado.
sub	Substitui uma ou mais ocorrências buscadas na ***string***.

RegEx - Metacharacters

Metacharacters são caracteres com um significa específico em uma expressão regular.

Caracter	Descrição	Exemplo
[]	Um conjunto de caracteres	“[a-m]”
\	Indica uma **sequencia especial** a ser usada	“\d”
.	Qualquer caractere	“he..o”
^	Inicia com...	“^hello”
$	Termina com...	“world$”
{}	Exato número de ocorrência	“he.{2}o”

RegEx – Special Sequence

Uma **Sequencia Especial** inicia com **** e é seguida de um dos caracteres abaixo:

Caracter	Descrição	Exemplo
\d	Retorna **Match** se a ***string*** contém dígitos (0-9)	"\d"
\D	Retorna **Match** se a ***string*** **NÃO** contém dígitos	"\D"
\s	Retorna **Match** se a ***string*** contém espaços	"\s"
\S	Retorna **Match** se a ***string*** **NÃO** contém espaços	"\S"
\w	Retorna **Match** se a ***string*** contém caracteres (alfabeto A-Z, dígitos 0-9 e o underscore _)	"\w"
\W	Retorna **Match** se a ***string*** **NÃO** contém caracteres (alfabeto A-Z, dígitos 0-9 e o underscore _)	"\W"

Exemplo 2

FIAP

Digite o exemplo abaixo e observe seu funcionamento.

```
1  import re
2  frase = "Quem ri por último, ri melhor"
3  print(re.findall("ri", frase))
4  print(re.findall("chora", frase))
5  busca = re.search("último", frase)
6  print(busca)
7  print(busca.start())
8  print(busca.end())
9  print(re.search("segundo", frase))
10 print(re.split("\s", frase))
11 print(re.split("\s", frase, 2))
12 print(re.sub("ri", "chora", frase))
13 print(re.sub("ri", "chora", frase, 1))
```

Exemplo 3

FUPQ peça para o usuário digitar seu nome e CEP da residência (faça verificação para permitir apenas letras para o nome e exatamente 8 dígitos para o CEP).

Exiba os dados informados caso não ocorra erros.

Exiba ao final uma mensagem de fim de programa independente se ocorreu um erro ou não.

Exemplo 3

```
1  import re
2  try :
3      print("Digite seu nome")
4      nome = input()
5      if re.search("\d", nome) :
6          erro = "Nomes não podem conter números"
7          raise ValueError
8      print("Digite seu CEP (somente dígitos)")
9      cep = input()
10     if not re.search("\d{8}", cep) or len(cep) > 8 :
11         erro = "CEP deve conter 8 dígitos"
12         raise ValueError
13     print(f"Nome: {nome}\nCEP: {cep}")
14 except ValueError :
15     print(erro)
16 finally :
17     print("Fim de programa!")
```

Exercícios

Exercícios

1. FUPQ peça para o usuário digitar seu nome e telefone, faça verificação para permitir apenas letras para o nome e somente dígitos para o telefone (com DDD e pode ser residencial ou celular).

Exiba os dados informados caso não ocorra erros e o telefone deve ser exibido seguindo o seguinte formato: **(***00***)** *00000-0000*.

Exiba ao final uma mensagem de fim de programa independente se ocorreu um erro ou não.

Exercícios

2. FUPQ peça para o usuário digitar seu nome e a placa do carro, faça verificação para permitir apenas letras para o nome e se a placa está no modelo do Mercosul (3 letras, um número, uma letra, dois números, exemplo: XYZ9X57). Atenção que as letras da placa são maiúsculas.

Exiba os dados informados caso não ocorra erros e a placa deve ser exibida seguindo o seguinte formato: ***XYZ 9X57*** (com um espaço após a 3 primeiras letras).

Exiba ao final uma mensagem de fim de programa independente se ocorreu um erro ou não.

Referências

Introdução à programação com Python. Nilo Menezes. Novatec, 2019.

Curso Intensivo de Python: Uma introdução prática e baseada em projetos à programação. Eric Matthes. Novatec, 2016.

Até breve!