

DESENVOLVIMENTO RÁPIDO DE APLICAÇÕES EM PYTHON

Aula 04 - Manipulação de strings. Tratamento de exceções.



Prof. Ronaldo Candido
ronaldo.candido@estacio.br



Estácio

2023.2

DESENVOLVIMENTO RÁPIDO DE APLICAÇÕES EM PYTHON



Estácio

Objetivos da aula

- **Reconhecer as funções de manipulação de strings.**
- **Descrever as exceções na manipulação de arquivos e outras operações.**

Conteúdo Programático

1. Funções de manipulação de strings.
2. Tratamento de exceções.
3. Exceções na manipulação de arquivos.
4. Aplicação em Python.
5. Atividade verificadora de aprendizagem.
6. Aprenda+.



Prof. Ronaldo Candido


AULA 04

DESENVOLVIMENTO RÁPIDO DE APLICAÇÕES EM PYTHON 

Sala de Aula Virtual (SAVA)




Prof. Ronaldo Candido AULA 04

DESENVOLVIMENTO RÁPIDO DE APLICAÇÕES EM PYTHON 

Situação-problema

- Conforme o conteúdo digital da disciplina, Tema "Manipulação de Dados", Módulo 2: "Funções de manipulação de strings" e "Exceções na manipulação de arquivos e outras operações", durante a vida de programador é muito comum nos depararmos com situações em que precisamos realizar alguns ajustes e operações sobre os textos lidos de arquivos.
- Quais são os métodos presentes nos objetos do tipo **str (string)**, muito utilizados em conjunto com a manipulação de arquivos ?



Prof. Ronaldo Candido AULA 04

Manipulação de strings

- Podemos definir um conteúdo de texto associando-o a uma variável do tipo texto.

- Exemplo:

```
frase = 'Esta é a primeira linha\n\
Esta é a segunda linha\n\
Esta é a terceira linha'
print (frase)
```

- Resultado:

```
Esta é a primeira linha
Esta é a segunda linha
Esta é a terceira linha

...Program finished with exit code 0
Press ENTER to exit console.
```

Funções básicas em Python

- Usamos `print()` para saída de dados (exibição) e `input()` para entrada de dados (digitação). Para conversão de dados usa-se `int()` para números inteiros e `str()` para strings. Exemplo:

#variáveis

```
nome = "Ronaldo"
sobrenome = "Candido"
numero1 = 10
numero2 = 5
```

#exibir

```
print(nome + ' ' + sobrenome)
print(numero1 + numero2)
```

#entrada de dados

```
x = int(input('Digite um número: '))
y = int(input('Digite outro número: '))
```

#operações matemáticas

```
print('Soma = ' + str(x + y))
print('Subtração = ' + str(x - y))
print('Multiplicação = ' + str(x * y))
print('Divisão = ' + str(x / y))
```

```
Ronaldo Candido
15
Digite um número: 10
Digite outro número: 3
Soma = 13
Subtração = 7
Multiplicação = 30
Divisão = 3.3333333333333335
```

Operações em strings

- Exemplos:

```
st = 'estou'
s = ' em casa'
print(st + s) # estou em casa
s = 'estou' ' em casa'
print(s)
#Interpolação:
st = 'vida boa'
print("o comprimento de %s é %d" % (st,len(st)) ) #o comprimento de vida boa é 8
#Uso da string como sequencia:
for c in s:
    print(c)
```

```
#particionando
st = 'Departamento de Sistemas e Computacao'
print(st[:15]) #Departamento de
print(st[27:]) #Computacao
print(st[:15],st[27:]) #Departamento de Computacao
print(st[15:27]) # Sistemas e
print(st[::-1]) #oacatupmoC e sametsiS ed otnematrapeD
```

Funções de Manipulação de strings

- Exemplos:

```
teste = 'Esta é uma simples mensagem'
#len() usado para encontrar o tamanho de uma string contando espaços e caracteres especiais.
print(len(teste)) #27
#replace() substitui uma string por outra.
teste = teste.replace('simples', 'curta')
print(teste) #Esta é uma curta mensagem
#count() exibe o número de vezes que uma string aparece dentro de outra string
print(teste.count('s')) #2
#find() exibe a posição que uma string esta ocupando dentro de outra string
print(teste.find('uma')) #7
```

Funções avançadas de Manipulação de strings

- Exemplos:

```
teste = ' Outra simples mensagem '
```

#strip() remove os espaços no início e no final da string.

```
print(teste.strip()) #Outra simples mensagem
```

#split() divide uma string em uma lista onde cada palavra é um item de lista.

```
x = teste.split()
print(x) #exibe ['Outra', 'simples', 'mensagem']
```

#join() junta todos os itens de uma tupla em uma string, usando um caractere de hash como separador

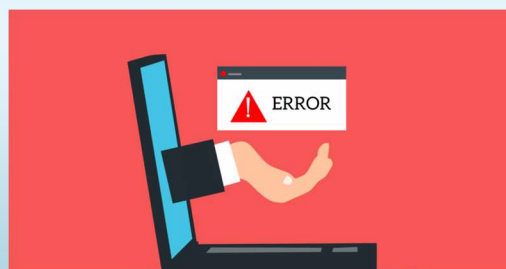
```
minhaTupla = ("John", "Peter", "Vicky")
x = "#".join(minhaTupla)
print(x) #exibe : John#Peter#Vicky
```

Tuplas podem conter elementos de qualquer tipo, porém são imutáveis !!!

Tratamento de exceções

- Diferentemente dos erros de sintaxe, no qual são erros de digitação do código que bloqueiam a execução do programa, as exceções são falhas que podem ocorrer durante a execução do programa e por isso utilizamos **try-except** para evitar que o programa deixe de executar devido a esses erros. Exemplo:

```
a = [ 1, 2, 3 ]
try:
    print(a[5]) #tentando acesso indevido
except IndexError: #captura a exceção
    print("Posição inexistente!")
finally:
    print("Fim do teste")
```



Tratamento de exceções (continuação)

- Várias exceções podem ser tratadas ao mesmo tempo. Exemplo:

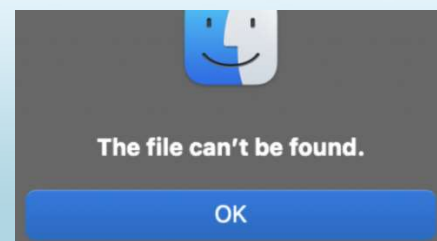
```
try:
    a = int(input('Digite um número:'))
    b = int(input('Digite um número para dividir com o primeiro:'))
    n = a/b
except ZeroDivisionError: #divisão por zero
    print('Não é possível dividir um número por zero')
except ValueError: #erro de valor
    print('Tivemos um problema com os dados fornecidos')
except KeyboardInterrupt: #interrompeu programa com CTRL C
    print('O usuário interrompeu a execução do programa')
else:
    print(n) #foi tudo bem
finally:
    print('Obrigado, volte sempre!') #sempre é executado
```

Tratamento de exceções (continuação)

- Garantindo tratamento automático em arquivos. Exemplo:

```
#sem tratamento
for linha in open('meuarquivo.txt','r'):
    print(linha, end='')

#com tratamento e fechando o arquivo automaticamente após o uso
try:
    with open('meuarquivo.txt','r') as f: #indica o apelido f
        for linha in f:
            print(linha, end='') #sem pular linha
except FileNotFoundError:
    print('Arquivo não encontrado')
```



Atividade verificadora de aprendizagem

- Voltando ao cenário que trata de um sistema de registro de notas de alunos em uma pequena instituição de ensino, propor situações em que **strings** fornecidas como entrada para o aplicativo precisam de tratamento para que sejam corretamente armazenadas em arquivo. Realizar também requisitos de tratamentos de exceções a serem implementadas. Os alunos devem incrementar o sistema desenvolvido na aula anterior para incluir esta funcionalidade, conforme requisitos fornecidos pelo docente.

Exemplos de requisitos são:

(A) Manipulações de Strings:

- (i) tratar entradas vazias, (ii) tratar nomes de alunos que contém números e (iii) tratar número de matrículas com caracteres.

(B) Tratamento de exceções:

- (i) Tratar fechamento de arquivos que não foram abertos, (ii) tentar escrever em arquivos com permissão para somente leitura, (iii) tentar ler dados de arquivos inexistentes.

Aprenda +

- Assistir o vídeo: Curso em vídeo. “Manipulando Texto”, Disponível em <<https://www.youtube.com/watch?v=a7DH88vk2Sk>>. Acesso em: 14 mar. 2023.

- Atividade Autônoma Aura:

Questão 1) A interpolação de String é uma característica bastante útil quando queremos fazer uma substituição dentro de um texto. O Python trata a interpolação de Strings de modo a facilitar o trabalho do desenvolvedor.

A respeito da interpolação de Strings no Python 3, selecione a opção correta:

- Pode ser feita com @ (arroba), ou com \$ (Cifrão)
- Podemos aplicar usar\$ (Cifrão), ou # (Hashtag)
- O modo correto é usando # (Hashtag), ou % (percentual)
- A forma é usando % (percentual), ou { } (chaves)
- Podemos utilizar { } (chaves), ou " (Aspas duplas)



Aprenda + (continuação)

Questão 2) As operações com arquivos são fundamentais na programação, pois os arquivos são um recurso muito útil para manter a persistência de dados. O Python oferece comandos que permitem que os desenvolvedores possam fazer essas manipulações.


A respeito da manipulação de arquivos no Python, selecione a opção CORRETA :

- a) Para que um programa que manipula arquivos funcione é necessário fazer o tratamento de exceções.
- b) Apesar de não ser obrigatório o tratamento de exceções, ele deve ser usado sempre para evitar problemas ao longo da execução do programa
- c) Por se tratar de um recurso que já faz parte do Python padrão, o tratamento de exceções não é necessário.
- d) Para abrir um arquivo texto, por exemplo, sempre é necessário verificar se ele não está corrompido antes, pois, caso contrário, pode gerar inconsistência no programa.
- e) Os arquivos podem gerar muitos problemas, portanto são um recurso de uso apenas didático. Aplicações reais são desenvolvidas com bancos de dados.

Para a próxima aula...

- Leitura e resolução dos exercícios propostos no livro **"BANIN, S. L. Python 3 Conceitos e Aplicações. Uma Abordagem Didática"**. Capítulo 8: Python 3 com Banco de Dados SQLite.
- Conteúdo digital da disciplina, Tema 3 - "PYTHON COM BANCO DE DADOS", Módulo 1 "Frameworks e bibliotecas para gerenciamento de banco de dados".
- Estudar a Aula 05 de **DESENVOLVIMENTO RÁPIDO EM PYTHON** no SAVA previamente.





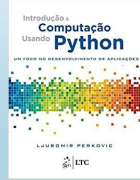
DESENVOLVIMENTO RÁPIDO DE APLICAÇÕES EM PYTHON


Referências

BANIN, S. L. Python 3 Conceitos e Aplicações. Uma Abordagem Didática. 1a. ed. São Paulo: Érica, 2018. Disponível em: <https://integrada.minhabiblioteca.com.br/#/books/9788536530253/>. Acesso em: 16 fev. 2022.

MACIEL, F. M. de B. Python e Django. Desenvolvimento web ágil e moderno. 1a. ed. Rio de Janeiro: Alta Books, 2020. Disponível em: <https://integrada.minhabiblioteca.com.br/reader/books/9786555200973>. Acesso em : 14 mar. 2022.

PERKOVIC, L. Introdução à Computação Usando Python - Um Foco no Desenvolvimento de Aplicações. 1a. ed. Rio de Janeiro: LTC, 2016. Disponível em: <https://integrada.minhabiblioteca.com.br/reader/books/9788521630937/>. Acesso em: 16 fev. 2022.

Prof. Ronaldo Candido
AULA 04

DESENVOLVIMENTO RÁPIDO DE
APLICAÇÕES EM PYTHON

Dúvidas, sugestões ou análises ???



Estácio

Prof. Ronaldo Candido
ronaldo.candido@estacio.br