AULA – 09

MANIPULANDO TEXTO

Data: 04/08/2023
Prof° Gustavo Guanabara

Teoria

Cadeia de texto

Para o Python, toda cadeia de texto está entre aspas simples e aspas duplas

Forma de atribuir uma string dentro de uma variável;

```
frase = 'Curso em vídeo'
```

Quando esse tipo de atribuição é feito, o python coloca esses dados na memória do computador, mas essa frase não vai inteira, o que ele vai fazer é criar mini espaços dentro da memória dentro do computador e dentro de cada mini espaço ele vai colocar cada uma das letras;

Ex.:
[C][u][r][s][o][ ][e][m][ ][v][í][d][e][o][ ][p][y][t][h][o][n]
[0][1][2][3][4][5][6][7][8][9][10][11][12][13][14][15][16][17][18][19][20]

Perceba que entre "curso" e "em" tem um espaço vazio e esse espaço ocupa esse mini espaço, e cada um desses mini espaços vai receber um índice que é um número sequencial começando de zero e indo até o número de letras que for necessário,

---

Fatiamento

Fatiar uma string é conseguir pegar pedaços dela

Exemplo de fatiamento;

```
Frase[9]
```

o símbolo de colchete é o identificador de uma estrutura de dados do python chamada lista, então se eu mandá-lo escrever print frase, ele vai mandar escrever a frase completa que seria

‘Curso em vídeo Pyhon’, mas se eu mandar escrever frase[9] ele vai conseguir identificar dentro da cadeia de caracteres somente o caractere 9 que no caso é o décimo caractere já que toda string começa pelo zero, no caso a letra 9 seria o [V].

Outra forma de fatiar e a seguinte;

frase[9:13]

de forma simples o 9 é o [V] e o 13 é [o]
dessa forma ele vai começar no 9 e vai até o 13, porém o 13 será excluído imprimindo ‘Vide’ se quiser imprimir até a letra o você terá que colocar até o 14, é sempre 1 a menos no final

Outra forma de fatiamento seria;

Frase[9:21]

Você pode achar que vai dar erro pelo fato da frase conter apenas 20 caracteres, mas muito pelo contrário se colocar até o 21 assim como no exemplo anterior o último caractere será ignorado

Outra forma;

Frase[9:21:2]

Dessa forma ele vai seguir começando pelo 9 e finalizando no 21, porém ele irá seguir pulando de 2 em 2

A frase provavelmente ficaria dessa forma;

[V][d][o][P][t][o]

Outra forma;

Frase[:5]

Sabendo que antes dos dois pontos : é onde ele vai começar e depois e onde vai terminar, quando não se coloca nada antes ele naturalmente começa pelo caractere zero

Outra forma;

Frase[15:]

A frase começa no 15 mas como não foi indicado o final o Python vai levar a frase até o final da string

Outra forma;

Frase[9::3]

Levando em conta os exemplos anteriores dessa mesma forma ele vai começar a frase pelo caractere 9 e vai até o final já que não foi especificado depois dos dois pontos, mas ele fara pulando de 3 em 3.

Análise

Analisar uma string é obter informações sobre ela como, qual o tamanho dela, qual letra ela começa ou termina, qual a primeira palavra inteira.

len(frase)

a primeira coisa que vamos fazer é utilizar a função “len” len vem de lenf que significa comprimento, então quando usamos essa função ela vai nos retornar o comprimento da frase, ou seja quantos espaços ou micro espaços ela tem que no caso seria 21 caracteres

frase.count(‘o’)

essa função vai basicamente pedir para que o programa conte quantas vezes aparece a letra o na variável frase, lembrando que ele está pedindo apenas a letra o minúscula, sendo assim as letras O maiúsculas não serão contadas já que o python diferencia as minúsculas das maiúsculas

uma outra forma de usa esse comando seria;

frase.count(‘o’.0,13)

isso vai basicamente fazer a contagem junto do fatiamento, o programa vai considerar do 0 até o 13 todos os o, vale lembrar que o caractere o no espaço 13 não será considerado já que o último espaço é desconsiderado quando fatiado

Outra funcionalidade de analize sria a seguinte;

frase.find(‘deo’)

isso vai me dizer quanta vezes ele encontrou na frase ‘deo’, nesse caso ele vai mostrar em que momento começa ‘deo’ que seria na posição 11

frase.find(‘android’)

Dentro da string não tem string android, se você coloca dentro do find uma string que simplesmente não existe, ele te retorna o valor -1

Operador;

‘Curso’ in frase

Isso basicamente está dizendo ‘Existe a palavra Curso em frase’, como existe essa string na variável frase ele vai te retornar True que é verdadeiro.

Transformação

Por via de regra, uma lista de string é imutável, não sendo possível mexer nela, mas é possível mudar ela através dos métodos

frase.replace(‘Python’, Android)

replace é trocar ou reposicionar no caso o programa vai procurar por ‘Python’ e substituir por ‘Android’

Método

frase.upper()

isso vai fazer tudo que não estiver em maiúsculo ficar em maiúsculo

Método

frase.lower()

esse de forma contraria ao Upper vai fazer tudo ficar em minúsculo

outra funcionalidade;

frase.capitalize()

Isso vai jogar todos os caracteres para minúsculo e só o primeiro caractere vai ficar em maiúsculo

Outra funcionalidade;

frase.title()

De forma parecida com o Capitalize o tilte vai analisar quantas palavras tem a string, ele faz isso pela posição dos espaços, então onde tem espaços ele faz uma quebra de palavra e vai fazer o capitalize palavra por palavra.

Outra funcionalidade;

frase.strip()

O strip vai remover todos os espaços inúteis no início e no final da string, tome como exemplo alguém leigo em um site e essa pessoa antes de digitar seu nome aperte espaço ou aperte no final do nome o espaço, ficando esse espaço excedente sem motivo algum, essa funcionalidade vai excluir esse espaço, porém somente o espaço do começo e do final da string.

frase.rstrip()

de forma similar ao strip o r na frente que é de right que é direita, dessa forma ele vai remover apenas os últimos espaços que no caso fica à direita.

frase.lstrip()

de forma similar ao rstrip o l na frente que é de left que é esquerda, dessa forma ele vai remover apenas os espaços do início que no caso fica à esquerda.

Divisão

frase.split()

basicamente vai ocorrer uma divisão em sua string considerando os espaços, ele vai pegar onde tem espaço e vai criar uma divisão

Ficaria assim de certa forma;

[C][u][r][s][o] [e][m] [V][i][d][e][o] [P][y][t][h][o][n]
[0][1][2][3][4] [0][1] [0][1][2][3][4] [0][1][2][3][4][5]

Como pode ver a numeração sofreu alteração depois da divisão fazendo cada palavra receber indexação nova e cada uma dessas palavras é colocada dentro de uma outra lista, então o split basicamente cria uma lista com todas as palavras de uma cadeia de caracteres

Ele vai separar essas palavras que foram separadas e essas palavras vão ter numerações

[Curso] [em] [Vídeo] [Python]
[0] [1] [2] [3]

Junção

se eu tenho nomes separados em listas eu posso utilizar o;

‘-’.join(frase)

Que serve para juntar uma coisa na outra, percebe-se também que tem ‘-’ na frente do Join, isso significa que vai juntar todos os elementos de frase e vai usar esse separador ‘-’ que ia gerar uma string única com a seguinte configuração;

Curso-em-Vídeo-Python

Caso queira que tenha um espaço no lugar do traço é só colocar o espaço entre o ‘ ’ na frente do join

---

Prática

Forma de imprimir um texto grande

Exemplo

Olá, seja bem-vindo a minha humilde casa, entre e sente-se no sofá, vou preparar algo para que possamos comer, enquanto isso por favor, tome uma xicara de chá

```
print(“””Olá, seja bem-vindo a minha humilde casa, entre e sente-se no sofá, vou preparar algo para que possamos comer, enquanto isso por favor, tome uma xicara de chá”””)
```

Desafios

Desafio 022:

Crie um programa que leia o nome completo de uma pessoa e mostre:

- O nome com todas as letras Maiúsculas.
- O nome com todas minúsculas.
- Quantas letras ao todo (sem considerar espaços).
- Quantas letras tem o primeiro nome.

```
Nome = str(input("Digite seu nome completo: "))
print("Seu nome em maiúsculas é {} ".format(nome.upper()))
print("Seu nome em minúsculas é {} ".format(nome.lower()))
print("Seu nome tem ao todo {} letras ".format(len(nome) – nome.count(' ')))
print("Seu primeiro nome tem {} letras ".format(nome.find(' ')))
```

Desafio 023:

Faça um programa que leia um número de 0 a 9999 e mostre na tela cada um dos dígitos separados.

Ex: Digite um número: 1834

unidade: 4
dezena: 3
centena: 8
milhar: 1

```
num = int(input('informe um número: '))
u = num // 1% 10
d = num // 10 % 10
c = num // 100 % 10
m = num // 1000 % 10
```

```
print(“Analisando o número {} ”.format(num))
print(“Unidade: {}  ”.format(u))
print(“Dezena: {}  ”.format(d))
print(“Centena: {}  ”.format(c))
print(“Milhar: {}  ”.format(m))
```

Desafio 024:

Crie um programa que leia o nome de uma cidade e diga se ela começa ou não com o nome “Santo”.

```
cid = str(input(“Em que cidade você nasceu? ”)).strip()
print(cid[:5].upper() == ‘Santo’)
```

Desafio 025:

Crie um programa que leia o nome de uma pessoa e diga se ela tem “Silva no nome”.

```
nome = str(input(“Qual é seu nome completo? ”)).strip()
print(“Seu nome tem Silva? {} ”.format(‘silva’ in nome.lower()))
```

Desafio 026:

Faça um programa que leia uma frase pelo teclado e mostre:

- Quantas vezes aparece a letra “A”
- Em que posição ela aparece a primeira vez.
- Em que posição ela aparece a última vez.

```
Frase = str(input(“Digite uma frase: ”)).upper().strip()
print(‘A letra A aparece {} vezes na frase. ’.format(frase.count(‘A’)))
print(‘A primeira letra A apareceu na posição {} ’.format(frase.find(‘A’) + 1))
print(‘A ultima letra A apareceu na posição {} ’.format(frase.rfind(‘A’) + 1))
```

Desafio 027:

Faça um programa que leia o nome completo de uma pessoa, mostrando em seguida o primeiro e o último nome separadamente.

Ex: Ana Maria de Souza
primeiro = Ana
último = Souza

```
n = str(input(‘Digite seu nome completo: ’)).strip()
nome = n.split()
print(‘Muito prazer em te conhecer! ’)
print(‘Seu primeiro nome é {} ’.format(nome[0]))
print(‘Seu último nome é {} ’.format(nome[len(nome) - 1]))
```
