ACH5531

Introdução à Computação

Cadeias de caracteres

Prof. Dr. Grzegorz Kowal

grzegorz.kowal@usp.br

https://sites.google.com/usp.br/ach5531

1° sem 2019 – sexta-feira, 14h00-15h45 – CB, Bloco 3, 2° andar, Lab. 6

Cadeias de caracteres

Cadeias de caracteres (**strings**) é qualquer sequência de caracteres. Caractere é a menor unidade de todo o texto. Os caracteres são classificados em 3 grupos: **numérico**, **letras do alfabeto** e **caracteres especiais**. É comum nos referirmos ao conjunto de caracteres numéricos e as letras com o termo alfanumérico.

Na programação, é comum a manipulação de cadeias de caracteres, isso porque, a maior parte da informação que o ser humano trabalha, está no formato de texto. Assim, processamos caracteres a todo instante e rapidamente descobrimos, a grande necessidade em tratar e processar textos. E também, rapidamente aprendemos que existem centenas de maneiras para tratarmos e reconhecermos trechos de texto. Por exemplo, toda linguagem de programação precisa ser interpretada e, para isso, deve haver o reconhecimento de diversos nomes, diversas partes. Cada parte, ou melhor, cada segmento de texto reconhecido possuíra significado diferente, algumas vezes serão expressões, noutras, apenas sinais.

Por essas razões, o entendimento e o domínio da manipulação de caracteres e de conjuntos de caracteres é tão essencial.

Cadeias de caracteres

Para o Python, **todo texto** ou **caractere** é um objeto do tipo *str*. Portanto, se precisarmos utilizar determinado texto como sendo um outro tipo de informação, é nossa responsabilidade como programador, transformar, ou melhor, converter do tipo *str* para o novo tipo. Vamos supor que desejamos que determinada *String* seja reconhecido pela linguagem como sendo um número, então, precisamos fazer a conversão e essa pode ser feita da seguinte forma:

Exemplo 1:

```
texto1 = “10”
texto2 = “20.2”
num = int(texto1)
valor = float(texto2)
```

O conceito de caractere e string

Caractere é a unidade mínima de toda *string*. Um caractere pode ser numérico, letras do alfabeto ou um caracteres especial. É importante observar, que no Python, diferentemente de outras linguagens, nós não temos o tipo *char*, ou seja, o tipo caractere. Para o Python, tudo são *strings*, indiferente se a mesma contiver 1 ou "n" caracteres.

No trecho de código a seguir, utilizamos a função type() para saber qual o tipo de alguns objetos

Exemplo 2:

```
>>> type("a")
<class 'str'>
>>> type("abc")
<class 'str'>
>>> s = “Lista de caracteres”
```

Operações sobre strings

Quando manipulamos cadeias de caracteres, isto é, string, utilizamos algumas terminologias especificas que são, de fato, o nome da ação que será realizada. A primeira e mais comum, é o termo **concatenar**. O segundo, menos utilizado verbalmente, porém, muito utilizado em nossos códigos é **interpolar**. Ambos os termos são verbos, e não raramente ouvirás ou lerás expressões em que falamos: "concatenar valores", "concatenação de variáveis", "interpolação de dados", "interpolação de valores" e etc.

Além de concatenação e interpolação temos outras formas de manipular as cadeias de caracteres como: **fatiamento**, **análise**, **transformação, divisão** e **junção**.

Exemplo 3:

```
for i in range(0,255):
  print("%d = '%s'" % ( i, chr(i) ))
```

Concatenação de texto

Concatenar é a junção de 2 cadeias de caracteres e que dá origem a uma nova string que é formada pela junça das 2 partes. A concatenação, pode ser a direita ou então a esquerda de outra string. Então por exemplo, a seguir, temos um exemplo em que fazemos alguns tipos de concatenações.

```
a = 'texto1'
b = 'texto2'
c = a + b
```

Exemplo 4:

```
nome = 'Pedro'
sobrenome = 'Santos'
print(nome + ' ' + sobrenome)
```

Interpolação de texto

Interpolar é a inserção de um trecho de texto dentro de outro. A interpolação, ocorre de várias formas, e, algumas linguagens e bibliotecas, proporcionam maneiras bastante interessantes e diferentes para interpolarmos valores. O Python fornece um conjunto de ferramentas bastante poderosas para esse fim.

```
>>> sexo = "masculino"
>>> nome = "Cláudio"
>>> interpolar = "Nome %s é %s " % (nome, sexo)
```

Exemplo 5:

```
nome = 'Pedro'
sobrenome = 'Santos'
print("%s %s" % (nome, sobrenome))
```

Fatiamento de texto

Fatiamento é a extração de uma parte de texto usando as posições de caracteres no texto (índices). A contagem de posições começa com 0. Podemos usar posições negativas também. -1 significa o último caractere de texto.

```
texto[i] # extração de i-esimo caractere
texto[i:j] # extração de um intervalo de caracteres, entre i e j
texto[i:j:k] # extração de um intervalo de caracteres, entre i e
j a cada k caracteres
```

Exemplo 6:

```
nome = 'Pedro'
nome[2] # mostra 'd'
nome[2:4] # mostra 'de'
nome[-2:-1] # mostra 'ro'
nome[::2] # mostra 'Pdo'
```

Análise de variável de texto

Análise é por exemplo a verificação de comprimento de texto, ou contagem de um caractere em uma variável de texto, ou verificação se o texto contem um caractere ou uma expressão.

```
len(frase) # retorna número de caracteres em variável 'frase'
frase.count('o') # retorna o número de caracteres 'o' em 'frase'
frase.find('deo') # mostra quantas vezes ele encontrou 'deo'.
Mostra o momento que começou o 'deo'
```

Exemplo 7:

```
texto = 'caractere'
len(texto) # mostra 9
texto.count('a') # mostra 2
texto.find('c') # 2
```

Transformação de texto

Existem vários operadores de transformação de texto. Alguns deles são listados abaixo.

```
frase.replace('Python', 'Android') # substitui todas palavras 'Python' na frase por 'Android'
frase.lower() # converte todas as letras da frase para letras minusculas
frase.upper() # converte todas as letras da frase para letras maiúsculas
frase.capitalize() # deixa a somente primeira letra em maiúsculo
frase.strip() # elimina todos os espaços no inicio e no fim
frase.lstrip() # elimina todos os espaços somente no inicio
frase.rstrip() # elimina todos os espaços somente no fim
```

Divisão e junção de texto

Divisão separa uma variável de texto em lista de palavras.

```
frase = 'Hoje temos aula'
lista = frase.split()
print(lista) # mostra lista ['Hoje', 'temos', 'aula']
```

Junção junta os elemento de frase usando como separador o – separa uma variável de texto em lista de palavras.

```
c = frase.split()
frasejunta = ''.join(c)
print(frasejunta) # mostra string 'Hojetemosaula'
```

Exemplo 8

Escreva um programa que solicita usuário fornecer uma frase e conta o número de palavras e vogais nela.

```
print("Programa conta número de palavras e vogais na frase fornecida por usuário.\n")


frase = input("Escreva uma frase: ")


npalavras = len(frase.split())
lfrase = frase.lower()
nvogais =  lfrase.count('a') + lfrase.count('e') + lfrase.count('i') + lfrase.count('o') + lfrase.count('u')


print("Número de palavras: ", npalavras)
print("Número de vogais: ",  nvogais)
```

Exemplo 9

Escreva um programa que usando a data de nascimento fornecida em formato ‘ano/mes/dia’, imprime ela no formato ‘dia de mês de ano’. Verifique que a data fornecida tem formato correto e se não for repete a solicitação.

Exercícios

10. Escreva um programa que mostra o número de cada letra usada na palavra ou frase fornecida pelo usuário.

11. Escreva um programa que solicita uma lista de números separados por espaço ou vírgula, calcula a média destes números, e acha o valor mínimo e máximo entre eles.