python™

Rafael Vieira Coelho

rafaelvc2@gmail.com

PARTE 1

7 - Strings

7.1 Um tipo de dado composto

Até aqui, vimos três diferentes tipos de dado: int, float e string.

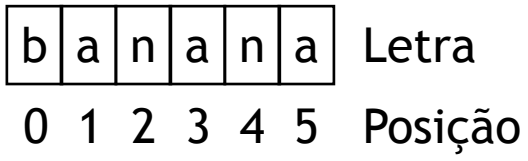
Strings são diferentes dos outros dois tipos porque são feitas de pedaços menores - caracteres.

Tipos que consistem de pedaços menores são chamados tipos de **dados compostos**.

Dependendo do que estejamos fazendo, pode ser que precisemos tratar um tipo de dado composto como uma coisa única, ou pode ser que queiramos acessar suas partes.

O operador colchete seleciona um único caractere de uma string:

```
>>> fruta = "banana"
>>> letra = fruta[1]
>>> print (letra)
```



python™

7.1 Um tipo de dado composto

- Uma string é simplesmente uma série de caracteres. Tudo que estiver entre aspas é considerada uma string em Python.

- Você pode usar aspas simples ou duplas em torno de suas strings.

- Diferença entre aspas duplas(“..”) e aspas simples(‘..’):

```
print(“Este é um exemplo com ‘aspas simples’ na string”)

print('Este é um exemplo com "aspas simples" na string')
```

7.2 Comprimento

A função len retorna o número de caracteres de uma string:

```
fruta = "banana"
len(fruta)
6
```

Como alternativa, podemos usar índices negativos, os quais contam de trás pra frente os elementos da string. A expressão fruta[-1] resulta a última letra, fruta[-2] resulta a penúltima (a segunda de trás para frente), e assim por diante.

Tenha cuidado!

Um tipo composto começa na posição 0 e vai até a N-1, tendo N posições.

python™

Repetição com o for

Usar um índice para percorrer um conjunto de valores é tão comum que Python oferece uma sintaxe alternativa
simplificada - o loop for:

```
for letra in fruta:
        print (letra)
```

Tarefa para casa: escreva uma função que tome uma string como argumento e devolva suas letras de trás para frente, uma por linha. Sua solução pode usar um comando while ou um for combinado com a função range().

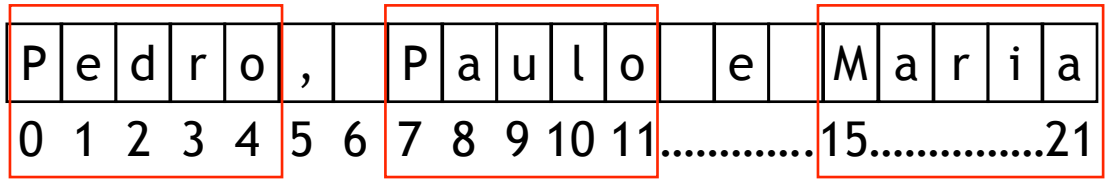
Ex: Mostra de 3, 2 e 1 na tela (o passo é -1, o padrão é +1).

```
for x in range(3, 0, -1):
    print (x)
```

7.3 Fatias de strings

Um segmento de uma string é chamado de uma fatia. Selecionar uma fatia é similar a selecionar um caractere:

```
s = "Pedro, Paulo e Maria"
```




```
print (s[0:5])
```
Pedro

```
print (s[7:12])
```
Paulo

```
print (s[15:21])
```
Maria

python™

7.3 Fatias de strings

Se você omitir o primeiro índice (antes dos dois pontos “:”), a fatia começa do início da string. Se você omitir o segundo índice, a fatia vai até o final da string.

Assim:

fruta = "banana"

fruta[:3]
’ban’

fruta[3:]
’ana’

O que você acha de s[:] significa?

```
>>> fruta = "banana"
>>> fruta[:3]
'ban'
>>> fruta[3:]
'ana'
>>> fruta[:]
'banana'
```

python™

Concatenação de strings

Muitas vezes, será conveniente combinar strings. Por exemplo, você pode querer armazenar um primeiro nome e um sobrenome em variáveis separadas e, então, combiná-las quando quiser exibir o nome completo de alguém:

Exemplo:

```
first_name = "ada"
last_name = "lovelace"
full_name = first_name + " " + last_name
print(full_name)
```

```
ada lovelace
```

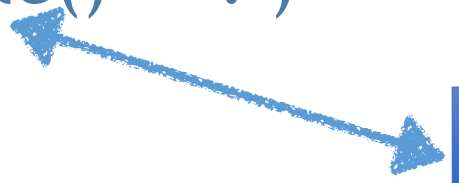
Concatenação de strings

Podemos usar concatenação para compor mensagens completas usando informações armazenadas em uma variável.

Exemplo:

```
first_name = "ada"
last_name = "lovelace"
full_name = first_name + " " + last_name
print("Hello, " + full_name.title() + "!")
```

Hello, Ada Lovelace!



A função title() modifica a primeira letra de cada palavra para maiúscula.

Acrescentando espaços em branco em strings com tabulações ou quebras de linha

Em programação, espaços em branco se referem a qualquer caractere que não é mostrado, como espaços, tabulações e símbolos de fim de linha.

Para acrescentar uma tabulação e uma quebra de linha em seu texto, utilize a combinação de caracteres \t e \n como mostrado abaixo.

Exemplo 1:

```
>>> print("Python")
Python
>>> print("\t Python")
        Python
>>> print("\n Python")

 Python
```

Exemplo 2:

```
>>> print("Languages:\n\tPython\n\tC\n\tJavaScript")
Languages:
  Python
  C
  JavaScript
```

Removendo Espaços em Branco

- Nesse exemplo, começamos com um valor que tem espaços em branco no início e no fim.
- Então removemos os espaços extras do lado direito, do lado esquerdo em e de ambos os lados.
- No mundo real, essas funções de remoção são usadas com mais frequência para limpar entradas de usuário antes de armazená-las em um programa.

```
 >>> favorite_language = ' python
 >>> favorite_language.rstrip()
' python'
 >>> favorite_language.lstrip()
'python '
 >>> favorite_language.strip()
'python'
```

7.4 Comparação de strings

O operador de comparação funciona com strings. Para ver se duas strings são iguais:

```
if palavra == "banana":
        print ("Sim, nós não temos bananas!")
```

Outras operações de comparação são úteis para colocar palavras em ordem alfabética:

```
if palavra < "banana":
        print ("Sua palavra," + palavra + ", vem antes de banana.")
elif palavra > "banana":
        print ("Sua palavra," + palavra + ", vem depois de banana.")
else:
        print ("Sim, nós não temos bananas!")
```

Tenha cuidado!

Em python, letras maiúsculas vem antes das minúsculas.

python™

7.5 Uma função find (procurar)

O que faz a seguinte função?:

```
def encontrar(palavra, letra):
        indice = 0
        while indice < len(palavra):
                if palavra[indice] == letra:
                        return indice
                indice = indice + 1
        return -1
```

Tarefa para casa: modifique a função encontrar de modo que ela receba um terceiro parâmetro, o índice da string por onde ela deve começar sua procura.

Este padrão de computação é às vezes chamado de travessia “eureka”, porque tão logo ele encontra (find) o que está procurando, ele pode gritar “Eureka!” e parar de procurar.

7.6 O módulo string

Toda string inclui uma função chamada find (encontrar) que faz a mesma coisa que a função que escrevemos.

Para chamá-la, temos que especificar o nome do módulo e o nome da função usando a notação de ponto.:

```
fruta = "banana"
indice = fruta.find("a")
print (indice)
```

1

Outros usos da função find

A função find pode encontrar substrings, não apenas caracteres:

```
"banana".find("na")
2
```

Além disso, ela recebe um argumento adicional que especifica o índice pelo qual ela deve começar sua procura:

```
"banana".find("na", 3)
4
```

Ou ela pode receber dois argumentos adicionais que especificam o intervalo de índices:

```
"bob".find("b", 1, 2)
-1
```

Neste exemplo, por que é retornado -1?

7.7 Classificação de caracteres

A string string.lowercase contém todas as letras que o sistema considera como sendo minúsculas. Similarmente, string.uppercase contém todas as letras maiúsculas. Tente o seguinte e veja o que você obtém:

```
import string

print (string.ascii_letters)
print (string.ascii_lowercase)
print (string.ascii_uppercase)
print (string.digits)
```

https://docs.python.org/3.6/library/string.html

```
10  print (string.ascii_letters)
11  print (string.ascii_lowercase)
12  print (string.ascii_uppercase)
13  print (string.digits)
```

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ
0123456789
```

python™

7.7 Classificação de caracteres

A string string.lowercase contém todas as letras que o sistema considera como sendo minúsculas. Similarmente, string.uppercase contém todas as letras maiúsculas. Tente o seguinte e veja o que você obtém:

```
import string

print (string.ascii_letters)
print (string.ascii_lowercase)
print (string.ascii_uppercase)
print (string.digits)
```

```
def main():
        letra = input("Qual o caractere?")
        if (ehMinusculo(letra)):
                print("É minusculo")
        else:
                print("Não é minusculo")
```

https://docs.python.org/3.6/library/string.html

Nós podemos usar essas constantes e find (encontrar) para classificar caracteres:

```
def ehMinusculo(letra):
        return string.ascii_lowercase.find(letra) != -1
```

Tarefa para Casa: crie a função ehMaiusculo(ch) e ehNumero(ch)

python™

Alternativas...

Como uma alternativa, podemos tirar vantagem do operador in, que determina se um caractere aparece em uma string:

```
def ehMinusculo2(ch):
        return ch in string.ascii_lowercase
```

Ainda, como uma outra alternativa, podemos usar o operador de comparação:

```
def ehMinusculo3(ch):
        return 'a' <= ch <= 'z'
```

Se ch estiver entre a e z, ele deve ser uma letra minúscula.

7.7 Funções auxiliares: Teste

islower(). Retorna True se string é composta por minúsculas
isupper() Retorna True se string é composta por maiúsculas
isspace() Retorna True se string contém espaço em branco

Exemplo:

```
texto = "palavra"
texto.islower()
True
texto.isupper()
False
texto.isspace()
False
```

7.7 Funções auxiliares: Teste

isalnum()	Retorna True se string é alfanumérica (contém apenas números e letras)
isalpha()	Retorna True se string contém apenas letras
isdigit()	Retorna True se string contém apenas números

```
>>> texto = "palavra12"
>>> texto.isalnum()
True
>>> texto.isdigit()
False
>>> texto.isalpha()
False
```

```
>>> texto = "palavra 12"
>>> texto.isalnum()
False
>>> texto.isalpha()
False
>>> texto.isdigit()
False
```

```
>>> texto = "palavra"

>>> texto.isalnum()
True
>>> texto.isalpha()
True
>>> texto.isdigit()
False
```

```
>>> texto = "12"
>>> texto.isalnum()
True
>>> texto.isalpha()
False
>>> texto.isdigit()
True
```

python™

7.7 Funções auxiliares: Pesquisa

- **endswith(s1: str): bool** Retorna True se a string testada termina com a substring s1
- **startswith(s1: str): bool** Retorna True se a string testada começa com a substring s1
- **count(substring): int** Conta o número de ocorrências da substring na string testada
- **find(s1): int** Retorna o menor índice (primeiro) onde aparece a string s1. Caso não encontrada, retorna -1.
- **rfind(s1): int** Retorna o maior índice (último) onde aparece a string s1. Caso não encontrada, retorna -1.

Exemplo:

```
texto = "palavra"
quantidade = texto.count(a")
print(quantidade)
```

7.7 Funções auxiliares: Conversão

- **lower(): str** transforma toda string para letras minúsculas.
- **upper(): str** transforma toda string para letras maiúsculas.
- **swapcase(): str** troca maiúsculas por minúsculas e vice-versa.
- **replace(old, new): str** substitui toda ocorrência de *old* por *new* na string.

Exemplo:

```
name = "Ada Lovelace"
print(name.upper())
print(name.lower())
```

Essas instruções exibirão o seguinte:

```
ADA LOVELACE
ada lovelace
```

python™

Exercícios

1) Nome ao contrário em maiúsculas. Faça um programa que permita ao usuário digitar o seu nome e em seguida mostre o nome do usuário de trás para frente utilizando somente letras maiúsculas.

Dica: lembre–se que ao informar o nome o usuário pode digitar letras maiúsculas ou minúsculas.

2) Nome na vertical. Faça um programa que solicite o nome do usuário e imprima-o na vertical em maiúscula.

F
U
L
A
N
O

Exercícios

3) Nome na vertical em escada. Modifique o programa anterior de forma a mostrar o nome em formato de escada.

F
FU
FUL
FULA
FULAN
FULANO

4) Escreva uma função que tome uma string como argumento e devolva suas letras de trás para frente, uma por linha.

DICA: A opção end='' permite não dar ENTER no final de cada print:
Ex: print(palavra, end = '')

5) Modifique a função encontrar (crie a encontrar2) de modo que ela receba um terceiro parâmetro, o índice da string por onde ela deve começar sua procura.

```
def encontrar(palavra, letra):
        indice = 0
        while indice < len(palavra):
                if palavra[indice] == letra:
                        return indice
                indice = indice + 1
        return -1

def main():
        palavra = input("Qual a palavra?")
        letra = input("Qual a letra a ser procurada?")
        posicao_inicial = int(input("Qual a posição inicial?"))
        posicao = encontrar2(palavra, letra, posicao_inicial)
        if (posicao == -1):
                print("Não foi encontrada")
        else:
                print("Indice: " + str(posicao))
main()
```

Exercícios

Exercícios

6) Crie a função ehMaiusculo(ch) e ehNumero(ch)

```
import string

def ehMinusculo(letra):
        return string.ascii_lowercase.find(letra) != -1

def ehMinusculo2(ch):
        return ch in string.ascii_lowercase

def ehMinusculo3(ch):
        return 'a' <= ch <= 'z'
```