FUNDAMENTOS DE PROGRAMAÇÃO

String

Prof. Bruno Góis Mateus

Índice

O TIPO DE DADOS STRING

O tipo de dados String

- Uma String é uma sequência de caracteres
  - Uma String literal deve estar entre aspas, simples ou duplas
  - Uma String que contém número, ainda é um string
    - É possível converter essa string numérica em números utilizando as funções int e float
  - O operador + signficado concatenação
    - Emendar

O tipo de dados String

```
>>> str1 = "Hello"
>>> str2 = 'there'
>>> bob = str1 + str2
>>> print bob
Hellothere
>>> str3 = '123'
>>> type(str3)
<type 'str'>
>>> str3 = str3 + 1
Traceback (most recent call last):  File "<stdin>", line 1, in
<module>TypeError: cannot concatenate 'str' and 'int' objects
>>> x = int(str3) + 1
>>> print x
124
```

Lendo e convertendo

- Em geral nós lemos String e depois convertemos para o tipo de dados que precisamos
  - Permite o maior controle más entradas de dados
  - O números informados via raw_input devem ser convertidos

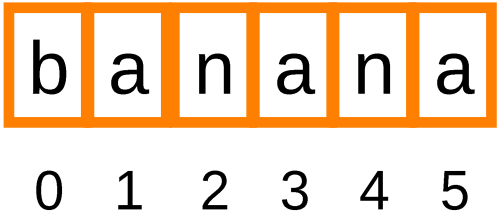
Lendo e convertendo

```
>>> name = raw_input('Enter:')
Enter:Chuck
>>> print name
Chuck
>>> apple = raw_input('Enter:')
Enter:100
>>> x = apple - 10
Traceback (most recent call last):  File "<stdin>", line 1, in
<module>TypeError: unsupported operand type(s) for -: 'str' and
'int'
>>> x = int(apple) - 10
>>> print x
90
```

OLHANDO PARA DENTRO DAS STRING

Olhando para dentro das string

- Nós podemos acessar uma caractere qualquer na string
  - Devemos informar o índice
    - Deve ser um inteiro
    - Inicia em zero
    - Pode ser informado através de um expressão
  - O índice deve ser posto entre colchetes




```
>>> fruit = 'banana'
>>> letter = fruit[1]
>>> print letter
a
>>> n = 3
>>> w = fruit[n - 1]
>>> print w
n
```

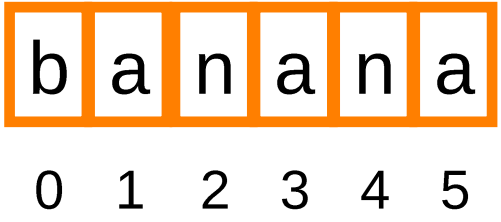
Olhando para dentro das string

- Um erro ocorre se tentarmos acesso um índice fora da string

```
>>> zot = 'abc'
>>> print zot[5]
Traceback (most recent call last):  File
"<stdin>", line 1, in <module>IndexError: string
index out of range
>>>
```

Toda string tem um tamanho

- Existe uma função built-in len
  - Ela informa o tamanho da string




```
>>> fruit = 'banana'
>>> print len(fruit)
6
```

PERCORRENDO STRINGS

Percorrendo strings

- Podemos percorrer um string utilizando while
  - A ideia é olhar para todos os caracteres
- Para isso precisamos de:
  - Variável de iteração
  - Função len

```
fruit = 'banana'
index = 0
while index < len(fruit):
    letter = fruit[index]
    print index, letter
    index = index + 1
```

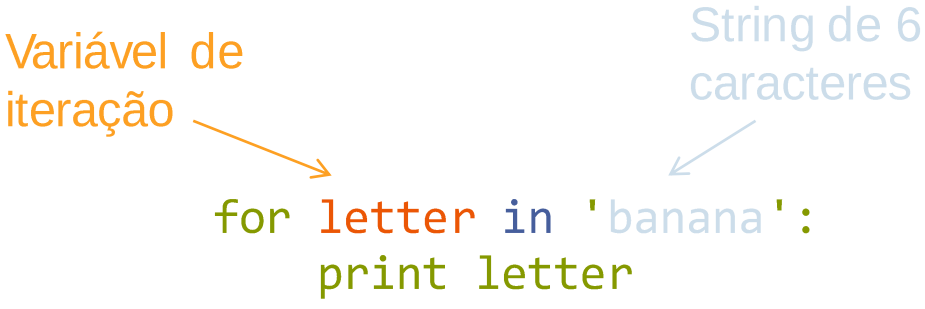
Percorrendo strings

- Utilizando um laço definido com o for
  - Não precisamos nos preocupar com a variável de iteração

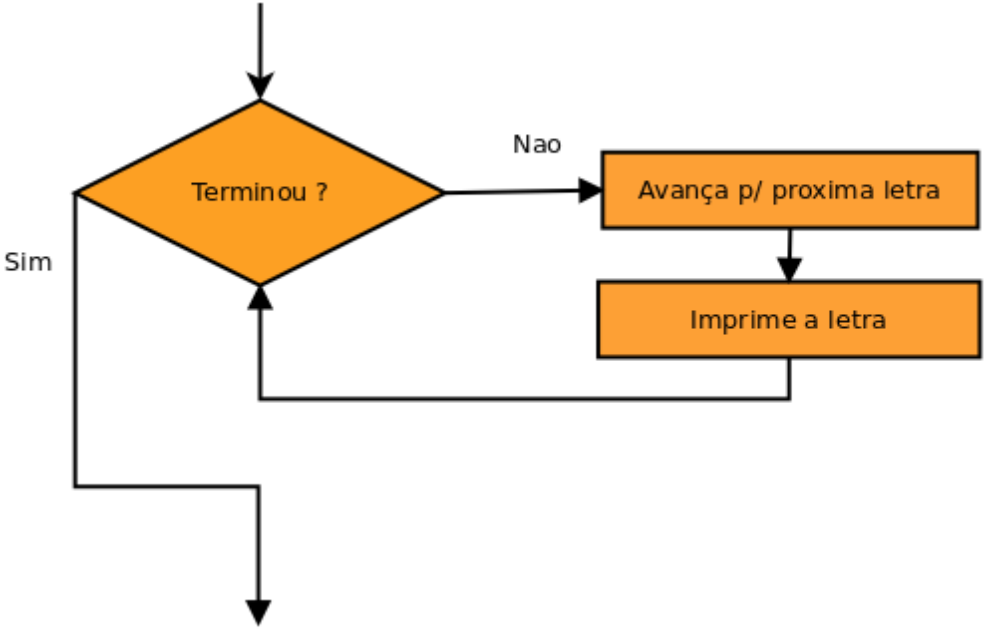
```
fruit = 'banana'
for letter in fruit:
    print letter
```

O operador in

- A variável de iteração percorre um sequência ordenada
- O bloco do for é executado uma vez para cada valor na sequência
- A variável de iteração percorre todos os valores da sequência



O operador in




```
for letter in 'banana':
    print letter
```

O operador in como operador lógico

- A palavra reservada in pode ser utilizada se uma string está contida em outra
- Nesse caso a expressão in é um operador lógico
  - Retorna sempre True ou False

```
>>> fruit = 'banana'
>>> 'n' in fruit
True
>>> 'm' in fruit
False
>>> 'nan' in fruit
True
>>> if 'a' in fruit :
...     print 'Found it!'
...
Found it!
>>>
```

Percorrendo strings e contando

- Contando quantas vezes um caractere ocorre em uma string

```
word = 'banana'
count = 0
for letter in word :
    if letter == 'a' :
        count = count + 1
print count
```

SLICING

Slicing

- Operado para facilitar o trabalho com substring
- Usa o operador :
  - Permite percorrer um sequência continua
  - O primeiro número informar o índice inicial
  - O segundo número informa o índice final
    - Este elemento não será incluído
    - Se o índice for maior que o tamanho da string o operador para no final

Slicing


```
>>> s = 'Monty Python'
>>> print s[0:4]
Mont
>>> print s[6:7]
P
>>> print s[6:20]
Python
```

Slicing

- Se o primeiro ou o último número forem omitidos será considerado o início e o fim da string respectivamente

```
>>> s = 'Monty Python'
>>> print s[:2]
Mo
>>> print s[8:]
thon
>>> print s[:]
Monty Python
```

CONCATENAÇÃO

Concatenação

- O operador + quando utilizado no contexto das strings significa concanetação

```
>>> a = 'Hello'
>>> b = a + 'There'
>>> print b
HelloThere
>>> c = a + ' ' + 'There'
>>> print c
Hello There
>>>
```

COMPARAÇÃO

Comparação

```
if word == 'banana':
    print  'All right, bananas.'

if word < 'banana':
    print 'Your word,' + word + ', comes before banana.'
elif word > 'banana':
    print 'Your word,' + word + ', comes after banana.'
else:
    print 'All right, bananas.'
```

A BIBLIOTECA STRING

A biblioteca String

- Python possui uma série de funções para trabalhar com string
  - Tais funções estão contidas na String library
  - São funções contida em cada string
    - Podemos chamá-las adicionado o nome da função ao final da string
  - Essas funções não modificam a string original
    - Retornam uma nova string modificada

```
>>> greet = 'Hello
Bob'
>>> zap = greet.lower()
>>> print zap
hello bob
>>> print greet
Hello Bob
>>> print 'Hi
There'.lower()
hi there
>>>
```

A biblioteca string

```
>>> stuff = 'Hello world'
>>> type(stuff)
<type 'str'>
>>> dir(stuff)
['capitalize', 'center', 'count', 'decode', 'encode', 'endswith',
'expandtabs', 'find', 'format', 'index', 'isalnum', 'isalpha',
'isdigit', 'islower', 'isspace', 'istitle', 'isupper', 'join',
'ljust', 'lower', 'lstrip', 'partition', 'replace', 'rfind',
'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
'splitlines', 'startswith', 'strip', 'swapcase', 'title',
'translate', 'upper', 'zfill']
```

- https://docs.python.org/2/library/stdtypes.html#string-methods

Procurando uma string

- Utilizamos a função find para procurar uma substring
  - Retorna o índice da primeira ocorrência da substring
  - Retorna -1 quando a substring não está contida na string

```
>>> fruit = 'banana'
>>> pos =
fruit.find('na')
>>> print pos
2
>>> aa = fruit.find('z')
>>> print aa
-1
```

Maiúsculo e Minúsculo

- Podemos criar um cópia da string
  - Com todas as letras maiúscula
  - Com todas as letras minúsculas

```
>>> greet = 'Hello Bob'
>>> nnn = greet.upper()
>>> print nnn
HELLO BOB
>>> www = greet.lower()
>>> print www
hello bob
>>>
```

Procurando e substituindo

- A função replace, tem a mesma funcionalidade de procurar e substituir de um editor de texto
  - Substitui todas as ocorrência de uma determinada string por outra

```
>>> greet = 'Hello Bob'
>>> nstr =
greet.replace('Bob','Jane')
>>> print nstr
Hello Jane
>>> nstr =
greet.replace('o','X')
>>> print nstr
HellX BXb
>>>
```

Procurando e substituindo

- String em python são imutáveis

```
greeting = "Hello, world!"
greeting[0] = 'J'              # ERROR!
print(greeting)
```

```
greeting = "Hello, world!"
newGreeting = 'J' + greeting[1:]
print(newGreeting)
print(greeting)
```

Removendo espações em branco

- Algumas vezes é necessário que nossas string não possuam espaços em branco no início e no final
  - lstrip, remove os espaços no início de uma string
  - rstrip, remove os espaços no fim de uma string
  - strip, remove os espaços no início e no final de uma string

Prefixo

```
>>> line = 'Please have a nice day'
>>> line.startswith('Please')
True
>>> line.startswith('p')
False
```

O que vem por aí

- Listas