Nesta aula…

Conteúdo

1 Cadeias de caracteres

Cadeias de caracteres

- compostas por caracteres individuais
- podemos tratá-las como um entidade única
- podemos também aceder aos caracteres individuais

```
>>> fruta = 'Banana'
>>> fruta[0]
'B'
>>> fruta[1]
'a'
>>> fruta[2]
'n'
>>> len(fruta)
6
```

Índices

`txt` ⟶	`'B`	`a`	`n`	`a`	`n`	`a'`
	0	1	2	3	4	5

- índices: `0`, `1`, `2`, `3`, `4`, `5`
- em geral: `0`, `1`, …, `len(txt)-1`
- caracteres: `txt[0]`, `txt[1]`, …
- último caracter: `txt[len(txt)-1]`
- penúltimo caracter: `txt[len(txt)-2]`

Fatias de cadeias de caracteres

`txt[i:j]` cadeia entre indices i e $j-1$ inclusíve

`txt[i:]` cadeia desde índice i até ao final

`txt[:j]` cadeia desde o ínicio até ao índice $j-1$ inclusíve

```
>>> fruta = 'Banana'
>>> fruta[:3]
'Ban'
>>> fruta[3:]
'ana'
>>> fruta[2:5]
'nan'
```

Cadeias de caracteres são imutáveis

- não se pode alterar os caracteres duma cadeia
- mas pode-se construir *outra* cadeia

```
>>> fruta = 'Bamana'
>>> fruta[2] = 'n'
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
TypeError: object doesn't support item assignment

>>> fruta = fruta[:2]+'N'+fruta[3:]
>>> print fruta
BaNana
```

Comparação de cadeias

- `txt1 <= txt2` é verdade se e só se `txt1` ocorre primeiro do que `txt2` segundo a *ordem lexicográfica*
- comparação entre caracteres é feita pelos código númericos ASCI/UTF-8

```
>>> 'abba' <= 'banana'
True
>>> 'abade' <= 'abacaxi'
False
>>> 'B' <= 'a'
True
>>> print ord('B'), ord('a')
66 97
```

Teste de ocorrência numa cadeia

- `txt1 in txt2` é verdade se e só se `txt1` ocorre dentro de `txt2`

```
>>> 'ana' in 'Banana'
True

>>> 'mana' in 'Banana'
False
```

2 Processamento de cadeias

Percorrer uma cadeia

Usando um ciclo `while`

```
# invariante: 0<=i<=len(fruta)-1
i = 0
while i<len(fruta):
  c = fruta[i]
  print c
  i = i + 1
```

Percorrer uma cadeia (2)

Usando um ciclo `for`

```
for c in fruta:
  print c
```

- evita a utilização explícita do índice
- mais geralmente: o ciclo `for` percorre quaisquer *sequências* (próxima aula)

Procurar a primeira ocorrência

Queremos uma função `primeira(c,txt)` que:

- procure se um caracter `c` ocorre em na cadeia `txt`;
- em caso afirmativo, retorne o índice da primeira ocorrência;
- em caso negativo, retorne -1.

```
>>> primeira('n', 'banana')
2
>>> primeira('m', 'banana')
-1
```

Procurar a primeira ocorrência

```
def primeira(c, txt):
    "Índice da 1ª ocorrência de c em txt."
    i = 0
    while i < len(txt):
        if txt[i] == c: # encontrou
            return i    # retorna o índice
        i = i + 1       # próximo índice

    return -1   # não encontrou: retorna -1
```

Variantes

- procurar a última ocorrência
- contar o número de ocorrências (exercício)

Procurar a última ocorrência

Basta percorrer índices de len(txt)-1 até 0...

```
def ultima(c, txt):
    "Índice da última ocorrência de c em txt."
    i = len(txt)-1
    while i >= 0:
        if txt[i] == c: # encontrou
            return i    # retorna o índice
        i = i - 1       # próximo índice

    return -1   # não encontrou: retorna -1
```

Funções pré-definidas

A linguagem Python tem pré-definidas várias operações sobre cadeias no módulo `string`.

find(txt1,txt2) índice da 1ª ocorrência de `txt2` em `txt1`

replace(txt1,txt2,txt3) substituir ocorrências de `txt2` por `txt3` em `txt1`

upper(txt) substituir letras minúsculas por maiúsculas

lower(txt) substituir letras maiúsculas por minúsculas

letters todas as letras

digits todos os algarismos

whitespace espaços branos

punctuation sinais de pontuação

help(string) para obter mais informarção

Funções pré-definidas

```
>>> import string
>>> fruit = 'banana'
>>> string.find(fruit,'a')
1
>>> string.find(fruit,'nan')
2
>>> string.replace(fruit,'ana','lua')
'bluana'
>>> string.upper(fruit)
'BANANA'
>>> string.letters
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKL
MNOPQRSTUVWXYZ'
```