Programação I
Aula 8 — Cadeias de carateres
Pedro Vasconcelos
DCC/FCUP

Nesta aula

1. Cadeias de carateres
2. Exemplo: a cifra de César

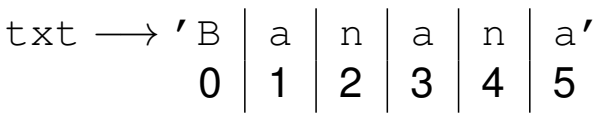
Cadeias de carateres

- São sequências de caracteres
- Podemos tratá-las como uma entidade única
- Mas também podemos aceder aos caracteres individuais

Exemplo

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

Índices



- Índices de `0` até `len(txt)-1`
- Caracteres: `txt[0]`, `txt[1]`, ...
- Último caracter: `txt[len(txt)-1]`
- Penúltimo caracter: `txt[len(txt)-2]`
- Índices negativos contam **do fim para o ínicio**:
  `txt[-1] == txt[len(txt)-1]`
  `txt[-2] == txt[len(txt)-2]`

Índices negativos

```
>>> txt = 'Banana'
>>> txt[5]
'a'
>>> txt[-1]
'a'
>>> txt[6]
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
IndexError: string index out of range
>>> txt[-6]
'B'
```

Fatias (*slices*)

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

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

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

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


Cadeias são imutáveis

- Não podemos modificar caracteres numa cadeia
- Mas podemos construir uma nova cadeia a partir de outras

Exemplo

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

>>> nova = txt[:2] + 'N' + txt[3:]
>>> nova
'BaNana'

>>> txt = txt[:2] + 'n' + txt[3:]
>>> txt
'Banana'
```


Comparação de cadeias

txt1 <= txt2 menor ou igual pela ordem lexicográfica (i.e. ordem do dicionário)

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

A ordem entre carateres corresponde à ordem dos seus *códigos númericos* (sistema *Unicode*).

Testar ocorrência

txt1 in *txt2* testar se *txt1* ocorre dentro de *txt2*

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

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


Percorrer uma cadeia (1)

Usando um ciclo sobre todos os índices válidos:

```
for i in range(len(txt)):
    # i = 0, 1, ..., len(txt)-1
    print(txt[i])
```

Percorrer uma cadeia (2)

Usando um ciclo diretamente sobre os carateres na cadeia:

```
for ch in txt:
  # ch = txt[0], txt[1], ..., txt[len(txt)-1]
  print(ch)
```

- Evita a necessidade da variável índice
- Mais geralmente: o ciclo `for` permite percorrer quaisquer *sequências* (próxima aula)

Procurar a primeira ocorrência

Vamos escrever uma função `primeira(ch,txt)` que:

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

Exemplo

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


Procurar a primeira ocorrência

```
def primeira(ch:str, txt:str) -> int:
    "Procura a primeira ocorrência de c em txt."
    for i in range(len(txt)):
        if txt[i] == ch: # encontrou
            return i     # termina e retorna o índice
    # fim do ciclo
    return -1   # não encontrou: retorna -1
```

Métodos sobre cadeias

- As cadeias suportam várias *operações pré-definidas*
- Usamos a sintaxe de *invocação de métodos*:

$$txt.operação(arg_1, arg_2, \ldots)$$

- Mais à frente: veremos exemplos de outros *objetos* e respectivos *métodos*

Métodos sobre cadeias (cont.)

find	índice da primeira ocorrência
replace	substituir ocorrências
upper	substituir letras minúsculas por maiúsculas
lower	substituir letras maiúsculas por minúsculas

Exemplos:

```
>>> txt = "Banana"
>>> txt.upper()
'BANANA'
>>> txt.find('ana')
1
>>> txt.replace('a','A')
'BAnAnA'
```

Exemplo: a cifra de César

- Um dos métodos mais simples para codificar um texto.
- Cada letra é substituida pela que dista k posições no alfabeto.
- Quando ultrapassa a letra 'z', volta à letra 'a'.

Exemplo: para $k = 3$, a rotação é:

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
d e f g h i j k l m n o p q r s t u v w x y z a b c
```

Logo: "`ataque`" é codificado como "`dwdtxh`".

Códigos de carateres

Vamos usar duas funções pré-definidas para converter entre carateres e códigos numéricos:

ord(c) obter o código numérico dum carater c;

chr(n) obter o carater com código n.

Exemplos:

```
>>> ord('A')
65
>>> chr(66)
'B'
```

Rodar um carater

Definimos uma função para deslocar as letras minúsculas *k* posições; outros caracteres ficam inalterados.

```
def rodar(k:int, ch:str) -> str:
    "Rodar um carater ch por k posições."
    if ch>='a' and ch<='z':
       n = ord(ch) - ord('a')
       return chr((n+k)%26 + ord('a'))
    else:
       return ch
```


Cifrar um texto

Para cifrar, percorremos o texto e aplicamos `rodar` a cada caracter.

```
def cifrar(k, txt):
    "Cifrar txt rodando k posições."
    msg = ""    # mensagem cifrada
    for ch in txt:
        msg = msg + rodar(k,ch)
    return msg
```

Atenção: esta solução não é eficiente—em aulas seguintes veremos soluções melhores!

Cifrar um texto (cont.)

Exemplo:

```
>>> cifrar(3, "ataque de madrugada!")
'dwdtxh gh pdguxjdgd!'
```

Para descodificar, basta usar cifrar com o deslocamento simétrico:

```
>>> cifrar(-3, "dwdtxh gh pdguxjdgd!")
'ataque de madrugada!'
```