USC
UNIVERSIDADE DE SANTIAGO DE COMPOSTELA

5. TIPOS ESTRUCTURADOS: SECUENCIAS

Programación I

Grado en Inteligencia Artificial

Curso 2022/2023

Contenidos

- Cadenas
  - Secuencias de escape
  - Recorrido de cadenas
  - Operador de corte
- Listas
  - Comparación, modificación y ordenación
  - El operador **is**
- Matrices

Cadenas

- Son *secuencias* de caracteres
- Operadores de concatenación (**+**) y repetición (*****)
- Funciones: **int()**, **float()**, **str()**, **ord()**, **chr()**, **len()**…
- Métodos: **lower()**, **upper()**, **title()**, **format()**, **find**()…

Secuencias de escape

- Permiten integrar **caracteres especiales**
- Se especifican usando la **barra invertida ** (“backslash”):
  - **\n**: salto de línea
  - **\t**: tabulador
  - **\a**: *campana*
  - ****: barra invertida
  - **\'** y **\"**: comillas

Acceso a los caracteres

- Carácter que ocupa la posición i+1 en una cadena s: **s[i]**
  - Primero: s[0]
  - Último: s[len(s) – 1]
- Se pueden usar índices negativos:
  - Último: s[-1]
  - Penúltimo: s[-2]
  - Primero: s[-len(s)]

Recorrido de cadenas

- Pueden recorrerse con bucles **for – in**:

```
s = 'En un lugar'
for c in s:
    print(c)
for i in range(len(s)):
    print(s[i])
```

Hacia delante

```
s = 'En un lugar'
for i in range(len(s)):
    print(s[len(s)-i-1])
for i in range(len(s)-1, -1, -1):
    print(s[i])
```

Hacia atrás

Ejemplo: recuento de palabras

```
cadena = input('Su frase: ')
cambios = 0
ant = ' '
for c in cadena:
    if c == ' ' and ant != ' ':
        cambios += 1
    ant = c
if cadena[-1] == ' ':
    cambios -= 1
palabras = cambios + 1
print('Palabras:', palabras)
```

Otro ejemplo: binario a decimal

```
bits = input('¿Su binario? ‘)

#se asume cadena bien formada!
n = len(bits)
dec = 0
for bit in bits:
    if bit == '1':
        dec = dec + 2 ** (n-1)
    n -= 1

print('Decimal:', dec)
```

Inversión de cadenas

```
cadena = input('¿Su cadena? ')

inversion = ''
for c in cadena:
    inversion = c + inversion

print('Inversión:', inversion)
```

Operador de corte (“slicing”)

```
cadena = input('¿Cadena? ')
i = int(input('¿Inicio? '))
j = int(input('¿Fin? '))
#faltan comprobaciones!
subcadena = ''
for k in range(i, j):
    subcadena += cadena[k]

print('Subcadena:',
subcadena)
```

La misma subcadena se puede obtener con esta expresión: **cadena[i:j]**

Se puede omitir cualquiera de los dos índices

Referencias a cadenas

- Ver sección 5.1.11 de Marzal et al.
- Al **asignar** a una variable la cadena contenida en otra, únicamente se copia la *referencia*
- **Concatenación** o **corte** de cadenas da lugar a una nueva reserva de memoria
- La **liberación** de memoria sucede de forma automática al dejar de referenciar