Taller de Física Computacional

Secuencias: listas, tuplas y rangos

Cristián G. Sánchez y Carlos J. Ruestes

2020

Secuencias

Como vimos existen tres tipos de secuencias incorporadas en Python:

tuplas

Secuencias inmutables de elementos de distinto tipo †.

listas

Secuencias mutables de elementos del mismo tipo †.

rangos

Secuencias inmutables de enteros ordenados.

Se pueden definir otros objetos que se clasifican como secuencias si implementan una serie de métodos especiales.

† El lenguaje no fuerza que los elementos cumplan esto.

Listas

Listas

Se puede crear una lista en python de la siguientes formas:

```
l_vacia = [] # crea una lista vacia
l_vacia = list()

# el argumento del constructor puede ser cualquier iterable
l_importantes = list((3.14,2.71))

l_enteros = [-3, -2, -1, 0, 1, 2, 3] # buen ejemplo

# un ejemplo de la anarquia pytonica
l_cosas = [3, 3.0, 'cadena', 1+3j, [-1,4,'mas cosas']]

# algo que se parece a una matriz
A = [[0.2, 0.3, 0.4], [0.5, 0.9, 3.4], [0.0, 1.0, 4.0]]
```

Tuplas

Tuplas

- Las tuplas en Python se usan fundamentalmente para devolver más de un valor en una función. Se utilizan en casos especiales para pasar a algunas funciones parámetros que contienen otros parámetros.
- Al ser inmutables su utilidad dentro de la programación científica (en nuestra opinión) es limitada, pero como son *hasheables* pueden utilizarse como índices en los diccionarios (los veremos más adelante).

Tuplas

Tuplas

Se pueden crear tuplas de la siguiente forma:

```
# directamente
planta = ('mimbulus','mimbletonia')
alumno = ('Cedric','','Diggory','Hufflepuff')
# la coma hace a la tupla
enemigo = 'Tom','Marvolo','Riddle','Slythrin'



# a partir de una función que
# devuelve varios valores
def f(x):
    return x,x**2,x**3

potencias_de_dos = f(2)
```

Rangos

Los rangos **representan** (no contienen) una secuencia de enteros ordenados. Se generan usando el constructor de la clase, la función `range(start,stop,step)`, la cual se puede llamar con uno, dos o tres argumentos. Ejemplos:

```
# Enteros de 0 a (stop-1)
rango_1 = range(10)

# Enteros de start a (stop-1)
rango2 = range(-10,11)

# Enteros de start a (stop-1) de dos en dos (step)
rango2 = range(-10,11,2)
```

Indexación

Indexación

Todas las secuencias son indexables:

- `seq[i]` es el i-ésimo elemento de la secuencia.
- `seq[0]` es el primer elemento de la secuencia.
- `seq[i:j]` es una *rebanada* de la secuencia, es la sub-secuencia desde el índice i al $j-1$.
- `seq[-1]` es el último elemento. Cualquier índice negativo sirve para indexar desde el final de la secuencia.
- Una lista de listas se idexa de la forma `seq[i][j]`

Si bien las secuencias son más generales podemos pensarlas (y usarlas) como un vector de n componentes.

Operaciones implementadas para secuencias (mutables o inmutables)

Operación	Resultado
`x in s`	`True` si un elemento de s es igual a x, sino `False`
`x not in s`	`False` si un elemento de s es igual a x, sino `True`
`s + t`	la concatenation de s y t
`s * n` o `n * s`	equivalente a sumar s a si misma n veces
`s[i]`	*iésimo elemento de s, el primero es el 0*
`s[i:j]`	rebanada de s de i a j
`s[i:j:k]`	rebanada de s de i a j con paso k
`len(s)`	largo de s
`min(s)`	el menor elemento de s
`max(s)`	el mayor elemento de s
`s.index(x, i[, j]])`	índice de la primera ocurrencia de x en s (a o antes de i y antes de j)
`s.count(x)`	número total de ocurrencias de x in s

Operaciones implementadas para secuencias mutables (listas)

Operación	**Resultado**
`s[i] = x`	reemplaza el ítem *i* de *s* por *x*
`s[i:j] = t`	la rebanada de *s* desde *i* a *j* se reemplaza por el contenido del iterable *t*
`del s[i:j]`	equivalente a `s[i:j] = []`
`s[i:j:k] = t`	los elementos de `s[i:j:k]` son reemplazados por los de *t*
`s *= n`	actualiza *s* con su contenido repetido *n* veces

Operaciones implementadas para secuencias mutables (listas)

Operation	**Result**
`s.append(x)`	agrega *x* al final de la secuencia
`s.clear()`	remueve todos los elementos de *s*
`s.copy()`	crea una copia superficial de *s*
`s.extend(t)` o `s += t`	extiende *s* con el contenido de *t*
`s.insert(i, x)`	inserta *x* en *s* en el índice *i*
`s.pop([i])`	devuelve y remueve de *s* el elemento en *i*
`s.remove(x)`	remueve el primer elemento de *s* cuyo valor es *x*
`s.reverse()`	invierte el orden de los elementos de *s* sin hacer una copi

Síntesis y recursos:

- Manual de referencia de Python
- Manual de la Librería estándar de Python