

# Taller de Física Computacional

## Secuencias: listas, tuplas y rangos

Cristián G. Sánchez y Carlos J. Ruestes

2021

# 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

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 pythonica
```

```
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

- 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

Se pueden crear tuplas de la siguiente forma:

*# directamente*

```
planta = ('mimulus', '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

Todas las secuencias son indexables:

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

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



# Operaciones implementadas para secuencias mutables (listas)

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

# Operaciones implementadas para secuencias mutables (listas)

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

# Síntesis y recursos:

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