

Tipos de datos estructurados

Ricardo Pérez López

IES Doñana, curso 2019/2020



1. Introducción
2. Secuencias
3. Estructuras no secuenciales
4. Iterables

1. Introducción

1.1 Conceptos

1.1. Conceptos

1.1. Conceptos

- ▶ Un **dato estructurado** o **dato compuesto** es un dato formado, a su vez, por otros datos llamados **componentes** o **elementos**.
- ▶ Un **tipo de dato estructurado**, también llamado **tipo compuesto**, es aquel cuyos valores son datos estructurados.
- ▶ Frecuentemente se puede acceder de manera individual a los elementos que componen un dato estructurado y a veces, también, se pueden modificar de manera individual.
- ▶ El término **estructura de datos** se suele usar como sinónimo de **tipo de dato estructurado**, aunque nosotros haremos una distinción:
 - Usaremos **tipo de dato estructurado** cuando usemos un dato sin conocer sus detalles internos de implementación.
 - Usaremos **estructura de datos** cuando nos interesen esos detalles internos.

2. Secuencias

2.1 Concepto de secuencia

2.2 Operaciones comunes

2.3 Inmutables

2.4 Mutables

2.1. Concepto de secuencia

2.1. Concepto de secuencia

- ▶ Una secuencia es una estructura de datos que:
 - permite el acceso eficiente a sus elementos usando índices enteros, y
 - se le puede calcular su longitud mediante la función `len`.
- ▶ Las secuencias se dividen en:
 - **Inmutables:** cadenas (`str`), tuplas (`tuple`), rangos (`range`).
 - **Mutables:** listas (`list`), principalmente.

2.2. Operaciones comunes

2.2. Operaciones comunes

- ▶ Todas las secuencias (ya sean cadenas, listas, tuplas o rangos) comparten un conjunto de operaciones comunes.
- ▶ Además de estas operaciones, las secuencias del mismo tipo admiten comparaciones. Las tuplas y las listas se comparan lexicográficamente elemento a elemento.
 - Eso significa que dos secuencias son iguales si cada elemento es igual y las dos secuencias son del mismo tipo y tienen la misma longitud.
- ▶ La siguiente tabla enumera las operaciones sobre secuencias, ordenadas por prioridad ascendente. `s` y `t` son secuencias del mismo tipo, `n`, `i`, `j` y `k` son enteros y `x` es un dato cualquiera que cumple con las restricciones que impone `s`.

Operación	Resultado
<code>x in s</code>	<code>True</code> si algún elemento de <code>s</code> es igual a <code>x</code>
<code>x not in s</code>	<code>False</code> si algún elemento de <code>s</code> es igual a <code>x</code>
<code>s + t</code>	La concatenación de <code>s</code> y <code>t</code>
<code>s * n</code>	Equivale a añadir <code>s</code> a sí mismo <code>n</code> veces
<code>n * s</code>	
<code>s[i]</code>	El <i>i</i> -ésimo elemento de <code>s</code> , empezando por 0
<code>s[i:j]</code>	Rodaja de <code>s</code> desde <i>i</i> hasta <i>j</i>
<code>s[i:j:k]</code>	Rodaja de <code>s</code> desde <i>i</i> hasta <i>j</i> con paso <i>k</i>
<code>len(s)</code>	Longitud de <code>s</code>
<code>min(s)</code>	El elemento más pequeño de <code>s</code>
<code>max(s)</code>	El elemento más grande de <code>s</code>
<code>s.index(x[, i[,j]])</code>	El índice de la primera aparición de <code>x</code> en <code>s</code> (desde el índice <i>i</i> inclusive y antes del <i>j</i>)
<code>s.count(x)</code>	Número total de apariciones de <code>x</code> en <code>s</code>

2.3. Inmutables

2.3.1 Cadenas (`str`)

2.3.2 Tuplas

2.3.3 Rangos

Funciones

Métodos

Expresiones regulares

Tuplas

- ▶ Las tuplas son secuencias inmutables, usadas frecuentemente para almacenar colecciones de datos heterogéneos (de tipos distintos).
- ▶ También se usan en aquellos casos en los que se necesita una secuencia inmutable de datos homogéneos (por ejemplo, para almacenar datos en un conjunto o un diccionario).
- ▶ Las tuplas se pueden crear:
 - Con paréntesis vacíos, para representar la tupla vacía: `()`
 - Usando una coma detrás de un único elemento:
`a,`
`(a,)`
 - Separando los elementos con comas:
`a, b, c`
`(a, b, c)`
 - Usando el constructor `tuple()`

- ▶ Observar que lo que construye la tupla es realmente la coma, no los paréntesis.
- ▶ Los paréntesis son opcionales, excepto en dos casos:
 - La tupla vacía: `()`
 - Cuando son necesarios para evitar ambigüedad.

Por ejemplo, `f(a, b, c)` es una llamada a función con tres argumentos, mientras que `f((a, b, c))` es una llamada a función con un único argumento que es una tupla de tres elementos.
- ▶ Las tuplas implementan todas las operaciones comunes de las secuencias.

2.4. Mutables

2.4.1 Listas

Listas

- En la siguiente tabla, `s` es una instancia de un tipo de secuencia mutable (en nuestro caso, una lista), `t` es cualquier dato iterable y `x` es un dato cualquiera que cumple con las restricciones que impone `s`:

Operación	Resultado
<code>s[i] = x</code>	El elemento <code>i</code> de <code>s</code> se sustituye por <code>x</code>
<code>s[i:j] = t</code>	La rodaja de <code>s</code> desde <code>i</code> hasta <code>j</code> se sustituye por el contenido del iterable <code>t</code>
<code>del s[i:j]</code>	Igual que <code>s[i:j] = []</code>
<code>s[i:j:k] = t</code>	Los elementos de <code>s[i:j:k]</code> se sustituyen por los de <code>t</code>
<code>del s[i:j:k]</code>	Elimina de la secuencia los elementos de <code>s[i:j:k]</code>

Operación	Resultado
<code>s.append(x)</code>	Añade <code>x</code> al final de la secuencia; es igual que <code>s[len(s):len(s)] = [x]</code>
<code>s.clear()</code>	Elimina todos los elementos de <code>s</code> ; es igual que <code>del s[:]</code>
<code>s.copy()</code>	Crea una copia <i>superficial</i> de <code>s</code> ; es igual que <code>s[:]</code>
<code>s.extend(t)</code>	Extiende <code>s</code> con el contenido de <code>t</code> ; es como hacer <code>s[len(s):len(s)] = t</code>
<code>s += t</code>	
<code>s *= n</code>	Modifica <code>s</code> repitiendo su contenido <code>n</code> veces
<code>max(s)</code>	El elemento más grande de <code>s</code>
<code>s.insert(i, x)</code>	Inserta <code>x</code> en <code>s</code> en el índice <code>i</code> ; es igual que <code>s[i:i] = [x]</code>
<code>s.pop([i])</code>	Extrae el elemento <code>i</code> de <code>s</code> y lo devuelve (por defecto, <code>i</code> vale <code>-1</code>)
<code>s.remove(x)</code>	Quita el primer elemento de <code>s</code> que sea igual a <code>x</code>
<code>s.reverse()</code>	Invierte los elementos de <code>s</code>

3. Estructuras no secuenciales

3.1 Conjuntos (`set` y `frozenset`)

3.2 Diccionarios (`dict`)

3.1. Conjuntos (set y frozenset)

3.2. Diccionarios (dict)

3.2.1 *Hashables*

4. Iterables

4.1 Iteradores

4.1. Iteradores