Tipos de datos estructurados
Fundamentos de la Programación

Salvador Sánchez

Universidad de Alcalá

Noviembre de 2021

Licencia

Colecciones de datos

Definición

Tipos de datos que representan grupos de elementos.

- Abstracciones que facilitan el manejo y acceso a los datos.
- Cada tipo de colección posee características específicas en lo relativo a la homogeneidad o heterogeneidad de los elementos, orden, acceso por posición, tamaño fijo o variable, y otros.
- Las colecciones disponibles varían de un lenguaje a otro.
  - A veces la colección que necesitamos está directamente disponible en el lenguaje. Ej.: arrays en C.
  - Otras veces el programador debe adaptar y modelar los tipos disponibles. Ej.: arrays en Python.

Colecciones de tamaño fijo

- **Arrays:** Colección cuyos elementos, todos del mismo tipo, pueden ser accedidos por posición.
  - En los lenguajes que los implementan es necesario definir de antemano su tamaño y el tipo de sus elementos.
  - Pueden tener una o más dimensiones: vectores, matrices, etc.
- **Cadenas:** Array unidimensional cuyo elemento base es un carácter.
- **Registros:** Colección formada por un conjunto de campos a cuyos elementos, potencialmente de tipos distintos, se accede por nombre de campo.
  - Dan soporte al concepto de "ficha" sobre un elemento: libro (título, autor, editorial), coche (matrícula, color, marca, modelo), etc.
  - Frecuentemente utilizada como dato base de colecciones más estructuradas: array de registros, pila de registros, etc.

Colecciones de tamaño variable

- **Pilas / Colas:** Colecciones donde lo más importante es el orden de entrada y salida de los elementos.
  - En una pila, el último elemento en entrar es el primero en salir (LIFO).
  - En una cola, el primer elemento en entrar es el primero en salir (FIFO).
  - Variantes: cola con prioridades, cola multinivel, etc.
- **Lista:** Colección de elementos a los que se accede por posición mediante un índice (a menudo entero).
  - En la mayoría de lenguajes es una estructura homogénea.
  - Variantes: tupla, lista ordenada.
- **Conjunto:** Agrupación no ordenada de elementos únicos.

Colecciones complejas de tamaño variable

- **Diccionario:** Colección formada por pares clave-valor en la que el acceso a los elementos se realiza por clave.
- **Archivo:** Almacenamiento externo (no en memoria principal) de datos bien homogéneos o bien heterogéneos.
  - Debe ser abierto antes de su manejo y cerrado al término del mismo.
  - Variantes: archivo de texto, binario, indizado, etc.
- **Otros:** Tablas hash, arrays dinámicos (arraylists), etc.

Colecciones de datos en Python

- Al no tener un sistema estricto de tipos de datos, en Python las colecciones son heterogéneas: pueden contener elementos de tipos distintos.
- Cada posición puede referenciar un elemento de tipo simple (entero, real, booleano, string) o compuesto (otras secuencias, instancias de una clase, archivos, etc.).

Mutabilidad

Definición

Aquellos objetos cuyo valor puede cambiar se dice que son **mutables**, a diferencia de aquellos cuyo valor no puede cambiar una vez creados, los cuales se denominan **inmutables**.

- Son inmutables los tipos de datos cuyas variables solo pueden cambiar el valor de todo el objeto (var= valor).
- Los tipos compuestos mutables, permiten cambiar el valor de los componentes, de forma independiente, o añadir nuevos elementos
- Mutables en Python: listas, diccionarios.
- Inmutables: números, cadenas y tuplas.

Listas

Definición

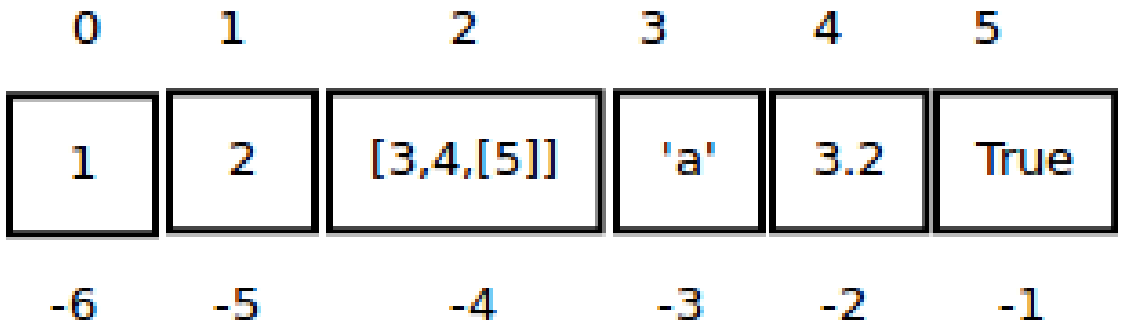
Conjunto mutable y organizado de elementos.

- Se declara y representa encerrando los miembros entre corchetes.
- Los elementos de una lista son variables, pudiendo añadir, modificar y eliminar elementos en cualquier momento.
- Se acceder a los elementos por posición mediante un índice entero
- Rango de los elementos: 0 a longitud-1
- Puede haber elementos repetidos

Lista

```
lista = [1, 2, [3,4,[5]], 'a', 3.2, True]
```

- Representación interna:



- El acceso por posición puede hacerse de izquierda a derecha o de derecha a izquierda, utilizando índices enteros positivos o negativos.

Operaciones enfocadas a elementos

- `lista.index(elem)` - Retorna la posición del elemento en la lista. Si no está se produce una excepción.
- `lista[i]` - Accede al elemento en la posición i.
- `elem in lista` - Determina si un elemento está o no en la lista.
- `lista.count(elem)` - Número de veces que aparece un elemento en la lista.

Operaciones sobre listas

- `len(lista)` - retorna el tamaño de la lista (num. elementos).
- `lista.append(elemento)` - añade un elemento al final de la lista.
- `L1.extend(L2)` - añade a L1 todos los elementos de L2.
- `lista1 + lista2` - produce una lista con los elementos de ambas.
- `lista.insert(pos,elem)` - inserta un elemento en una posición.
- `lista.remove(elem)` - Elimina la primera aparición del elemento.
- `lista.sort()` - Ordena la lista
- `lista.reverse()` - Invierte el orden de los elmentos de la lista
- `lista.pop()` - Elimina y retorna el último elemento
- + (concatenar) y * (repetir): Ej. lista1 = lista2 * 3

Iteración

- Las secuencias son el elemento base de la iteración con `for`.

```
frutero = ['pera', 'higo', 'fresa', 'caqui']
for fruta in frutero:
    print ("Una pieza de fruta: ", fruta)
```

Slicing

- Forma abreviada de obtener subsecuencias a partir de secuencias existentes
- Sintaxis: `secuencia [inicio_incluido:fin_excluido:salto]`
- Si no se especifica inicio, se entiende desde 0, si no se especifica fin, se entiende hasta el final de la secuencia
- Ejemplos:

```
preposiciones = ['a', 'ante', 'bajo', 'cabe', 'con', '
    contra', 'de', 'desde', 'en']
a = preposiciones [1:5]  # copia de la posicion 1 a la 4
     a=['ante', 'bajo', 'cabe', 'con
b = preposiciones [4:]  # copia de la posicion 4 al final
     b=['con','contra', 'de', 'desde', 'en']
c = preposiciones [:3]  # copia de la posicion 0 a la 2
    c=['a', 'ante', 'bajo']
d = preposiciones[:] # hace un duplicado
```

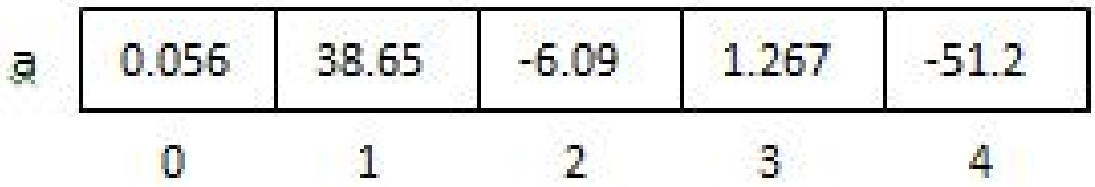
Tupla

Definición

Conjunto inmutable y ordenado de elementos.

- Se declara y representa encerrando los miembros entre paréntesis.
- Número fijo de elementos: sólo se pueden añadir y eliminar reasignando la tupla completa.
- Ejemplo:

$$a = (0.056,\ 38.65, -6.09, 1.267, -51.2)$$



Operaciones con tuplas

- Sus métodos son todos de *solo-lectura*.
  - No pueden añadirse elementos (no tienen métodos `append` ni `extend`).
  - No pueden eliminarse elementos (no tienen `remove` ni `pop`).
- Pueden utilizarse como claves en un diccionario (las listas no).
- Pueden 'convertirse' en listas: `list()` toma una tupla y devuelve una lista con los mismos elementos
  - Es posible también 'convertir' una lista a tupla: `tuple()`
  - Se dice que `tuple` "congela" una lista, y que a su vez `list` "descongela" una tupla.

String

Definición

Colección homogénea cuyos elementos son caracteres de texto.

- Número fijo de elementos: no se pueden añadir ni eliminar.
- Sus elementos no se pueden modificar.
- Creación: texto entre comillas simples o dobles.
- Ejemplo:

```
cadena1 = "Universidad de Alcala"
cadena2 = 'Alcala de Henares'
```

Operaciones con cadenas

- `lista.upper()` - Pone en mayúsculas
- `lista.lower()` - Pone en minúsculas
- `lista.find(car)` - Posición del carácter `car` en la cadena
- `lista.replace(car,car2)` - Sustituye un carácter por otro
- `lista.split()` - Retorna una lista con las partes de la cadena separadas (usando espacios)
- Se puede usar `split()` con cualquier separador si se indica: `split('.')`, `split(':')`, etc.
- Ya conocidas: `+` (concatenar cadenas), `*i` (repetir i veces la cadena)

Diccionarios

Definición

Estructura de datos sin orden entre los elementos, y en la que el acceso viene determinado por una clave única que se asocia a cada valor.

- La clave es a menudo una cadena de texto, si bien puede ser cualquiera de los tipos inmutables de Python.
- Tipos de claves: : boolean, integer, float, tupla, string ...
- Los diccionarios son mutables: se pueden agregar, eliminar y cambiar sus elementos.

Crear diccionarios

- Para crear un diccionario se emplean corchetes ({}) alrededor de pares `clave:valor` separados por comas.
- El diccionario más simple es un diccionario vacío, el cual no contiene ninguna clave o valor en absoluto:

```
>>> empty_dict = {}
>>> empty_dict
{}
```

Ejemplo de diccionarios

- Ejemplo preliminar de diccionario con citas del Diccionario del Diablo de Ambrose Bierce:

```
>>> bierce = {
    "dia": "Periodo de veinticuatro horas, casi todas desperdiciadas",
    "paciencia": "forma menor de desesperanza, a menudo disfrazada de virtud",
    "paz": "En asuntos internacionales, tiempo de mentiras entre dos periodos de
        lucha",
    }
>>>
```

- Al escribir el nombre del diccionario en el intérprete se imprimirán sus claves y valores:

```
>>> bierce
{'dia': 'Periodo de veinticuatro horas, casi todas desperdiciadas', 'paciencia': '
    forma menor de desesperanza, a menudo disfrazada de virtud', 'paz': 'En
    asuntos internacionales, tiempo de mentiras entre dos periodos de lucha'}
>>>
```

Operaciones con diccionarios

- **Agregar** un elemento: basta con referirse al ítem por su clave y asignarle un valor.
  - Si la clave ya estaba en el diccionario, el valor existente se reemplaza por el nuevo, si la clave es nueva, se agrega al diccionario con su valor.
  - Imposible errar por especificar un índice fuera de rango.
- **Eliminar** un elemento: utilizar `del`. Ejemplo: `del bierce['dia']`
- **Reiniciar** eliminando todos los elementos: `clear`. Ejemplo: `bierce.clear()`
- **Pertenencia** de una clave: `in`. Ejemplo: `'dia' in bierce`
- **Número de elementos:** `len(diccionario)`
- **Réplica:** `nuevo_diccionario = diccionario.copy()`

Más operaciones con diccionarios

- **Lista de claves**: `keys`. Ejemplo: `bierce.keys()`
  - Retorna un objeto de tipo `dict_keys`, que es una forma iterable de lista. Puede convertirse a lista con `list()`.
- **Lista de valores**: `values`. Ejemplo: `bierce.values()`
  - Retorna un objeto de tipo `dict_values`, una forma iterable de lista.
  - Ejemplo: `for item in x.values(): print item`
- **Lista de items**: lista de tuplas (clave,valor) que al igual que `values` y `keys`, es iterable: `diccionario.items()`

Resumen

- Las colecciones son abstracciones que facilitan el manejo de datos agrupados.
- Todo lenguaje de programación implementará unicamente algunas colecciones, otras deben ser modeladas por el programador
- Una lista en Python es una secuencia de elementos mutable (número de elementos variable + elementos modificables)
- Una tupla en Python es una colección de elementos inmutable (número de elementos fijo + elementos no modificables)
- en Python el acceso a las posiciones y muchas otras operaciones sobre secuencias son comunes a todas las secuencias
- Las cadenas de texto son secuencias inmutables de elementos homogéneos
- Los diccionarios contienen parejas clave-valor