El futuro digital es de todos

MinTIC

Listas

Rogerio Orlando Beltrán Castro

Universidad Industrial de Santander

UIS

Mision TIC 2022

Listas

Presentaremos ahora una nueva estructura de datos: la lista. Usaremos listas para poder modelar datos compuestos pero cuya cantidad y valor varían a lo largo del tiempo. Son secuencias mutables y vienen dotadas de una variedad de operaciones muy útiles.

La notación para lista es una secuencia de valores encerrados entre corchetes y separados por comas. Por ejemplo, si representamos a los alumnos mediante su número de padrón, se puede tener una lista de inscritos en la materia como la siguiente: i [78455, 89211, 66540, 45750]. Al abrirse la inscripción, antes de que hubiera inscritos, la lista de inscritos se representará por una lista vacía: [].

Longitud de la lista. Elementos y segmentos de listas

Como a las secuencias ya vistas, a las listas también se les puede aplicar la función len() para conocer su longitud.

Para acceder a los distintos elementos de la lista se utilizará la misma notación de índices de cadenas y tuplas, con valores que van de 0 a la longitud de la lista −1.

```
xs=[78455, 89211, 66540, 45750]

print('Lista 0:',xs[0])

print('Longitude de Lista:',len(xs))

print('Lista 4:',xs[4])

print('Lista 3:',xs[3])
```

Para obtener una sublista a partir de la lista original, se utiliza la notación de rangos, como en las otras secuencias.

Para obtener la lista que contiene sólo a quién se inscribió en segundo lugar podemos escribir:

```
print('Lista 1:2:',xs[1:2])
```

Para obtener la lista que contiene al segundo y tercer inscritos podemos escribir:

```
print('Lista 1:2:',xs[1:3])
```

Para obtener la lista que contiene al primero y segundo inscritos podemos escribir:

```
print('Lista 1:2:',xs[:2])
```

Cómo crear listas

Las dos maneras principales de crear una lista son:
usar una lista literal, con los valores entre corchetes:

```
primos = [2, 3, 5, 7, 11]
print('Lista:',primos)
print('Lista:',[])
print('Lista:',[1.0 + 2.0, 3.0 + 4.0 + 5.0])
print('Lista:',['hola ' + 'mundo', 24 * 7, True or False])
```

usar la función list aplicada sobre un iterable:

```
print('Lista',list('hola'))
print('Lista',list(range(10)))
print('Lista',list())
```

Cómo mutar listas

Dijimos antes que las listas son secuencias mutables. Para lograr la mutabilidad Python provee operaciones que nos permiten cambiarle valores, agregarle valores y quitarle valores.

Para cambiar una componente de una lista, se selecciona la componente mediante su índice y se le asigna el nuevo valor:

```
xs[1]=79211

print('Lista:',xs)
```

Para agregar un nuevo valor al final de la lista se utiliza la operación append(). Escribimos xs.append(47890) para agregar el dato 47890 al final de xs.

```
xs.append(47890)

print('Lista:',xs)
```

Cómo mutar listas

Para insertar un nuevo valor en la posición cuyo índice es k (y desplazar un lugar el resto de la lista) se utiliza la operación insert().

Escribimos xs.insert(2, 54988) para insertar el padrón 54988 en la tercera posición de xs.

```
xs.insert(2, 54988)
print('Lista:',xs)
```

Nota: Las listas no controlan si se insertan elementos repetidos, si necesitamos exigir unicidad, debemos hacerlo mediante el código de nuestros programas.

Cómo mutar listas

Para eliminar un valor de una lista se utiliza la operación remove().

Escribimos xs.remove(45750) para borrar el padrón 45750 de la lista de inscritos:

```
xs.remove(45750)
print('Lista:',xs)
```

Si el valor a borrar está repetido, se borra sólo su primera aparición:

```
xs.remove(78455)
print('Lista:',xs)
```

Advertencia: Si el valor a borrar no existe, se produce un error:

Cómo buscar dentro de las listas

Queremos poder formular dos preguntas más respecto de la lista de inscritos:

¿Está la persona cuyo padrón es v inscripta en esta materia?

¿En qué orden se inscribió la persona cuyo padrón es v?.

Veamos qué operaciones sobre listas se pueden usar para lograr esos dos objetivos:

Para preguntar si un valor determinado es un elemento de una lista usaremos la operación in:

```
xs= [78455, 79211, 54988, 66540, 47890]
print('Lista:',xs)
print('78 in xs',78 in xs)
print('66540 in xs',66540 in xs)
```

Nota: Esta operación se puede utilizar para todas las secuencias, incluyendo tuplas y cadenas

Cómo buscar dentro de las listas

Para averiguar la posición de un valor dentro de una lista usaremos la operación index().

```
print('Posición de 78455:',xs.index(78455))
print('Posición de 47890:',xs.index(47890))
```

Advertencia: Si el valor no se encuentra en la lista, se producirá un error.

Si el valor está repetido, el índice que devuelve es el de la primera aparición:

```
xs=[10,20,10]
print('Lista:',xs)
print('xs.index(10)',xs.index(10))
```

Nota: Esta operación está disponible en cadenas, pero no en tuplas.

Secuencias

En Python, las listas, las tuplas y las cadenas son parte del conjunto de las secuencias. Todas las secuencias cuentan con las siguientes operaciones:

Operación	Resultado
`x in s`	Indica si la variable `x` se encuentra en `s`
`s + t`	Concantena las secuencias `s` y `t`
`s * n`	Concatena `n` copias de `s`
`s[i]`	Elemento `i` de `s`, empezando por `0`
`s[i:j]`	Porción de la secuencia `s` desde `i` hasta `j` (no inclusive)
`s[i:j:k]`	Porción de la secuencia `s` desde `i` hasta `j` (no inclusive), con paso `k`
`len(s)`	Cantidad de elementos de la secuencia `s`
`min(s)`	Mínimo elemento de la secuencia `s`
`max(s)`	Máximo elemento de la secuencia `s`

Métodos en las listas

Un comentario al margen: append es un método. Los métodos son funciones que están dentro de un objeto. Cada lista tiene su propia función append. Es importante tener esta distinción clara, ya que hay operaciones que están implementadas como funciones y otras como métodos.

sum(x) entrega la suma de los valores de la lista:

```
print('Lista',sum([2, 3, 5, 7, 11]))
```

l1 + l2 concatena las listas l1 y l2:

```
print('Lista',list('perro') + [2, 3, 4])
```

l * n repite n veces la lista l:

```
print('Lista',[3.14, 6.28, 9.42] * 2)
```

l.count(x) cuenta cuántas veces está el elemento x en la lista:

```
letras = list('paralelepipedo')
print('Lista',letras)
print('Lista busca p:',letras.count('p'))
```

Ordenar listas

Nos puede interesar que los elementos de una lista estén ordenados: una vez que finalizó la inscripción en un curso, tener a los padrones de los alumnos por orden de inscripción puede ser muy incómodo, siempre será preferible tenerlos ordenados por número para realizar cualquier comprobación.

Python provee dos operaciones para obtener una lista ordenada a partir de una lista desordenada.

Para dejar la lista original intacta pero obtener una nueva lista ordenada a partir de ella, se usa la función sorted.

```
xs=[5,2,4,2]

cs=sorted(xs)

print('Lista Original:',xs)

print('Lista Ordenada:',cs)
```

Ordenar listas

Para modificar directamente la lista original usaremos la operación sort().

```
ds=[5,3,4,5]

print('Lista Original:',ds)

ds.sort()

print('Lista Ordenada:',ds)
```

Ordenar listas

l.reverse() invierte la lista:

```
l = [7, 0, 3, 9, 8, 2, 4]
print('Lista',l)
l.reverse()
print('Lista despues del reverse',l)
```

Listas y cadenas

A partir de una cadena de caracteres, podemos obtener una lista con sus componentes usando la función split.

Si queremos obtener las palabras (separadas entre sí por espacios) que componen la cadena xs escribiremos simplemente xs.split():

```
c = " Una cadena con espacios "

print('Frase:',ds)

print('Frase por palabra:',c.split())
```

En este caso split elimina todos los blancos de más, y devuelve sólo las palabras que conforman la cadena

Listas y cadenas

La casi–inversa de split es una función join que tiene la siguiente sintaxis:

<separador>.join(<lista de componentes a unir>)

y que devuelve la cadena que resulta de unir todas las componentes separadas entre sí por medio del separador:

```
xs = ['aaa', 'bbb', 'cccc']
print('Frase:',xs)
print('Frase:'," ".join(xs))
print('Frase:',", ".join(xs))
print('Frase:',"@@".join(xs))
```

Ordenar listas

Nos puede interesar que los elementos de una lista estén ordenados: una vez que finalizó la inscripción en un curso, tener a los padrones de los alumnos por orden de inscripción puede ser muy incómodo, siempre será preferible tenerlos ordenados por número para realizar cualquier comprobación.

Python provee dos operaciones para obtener una lista ordenada a partir de una lista desordenada.

Para dejar la lista original intacta pero obtener una nueva lista ordenada a partir de ella, se usa la función sorted.

```
xs=[5,2,4,2]

cs=sorted(xs)

print('Lista Original:',xs)

print('Lista Ordenada:',cs)
```

Resumen

- Python nos provee con varias estructuras que nos permiten agrupar los datos que tenemos. En particular, las tuplas son estructuras inmutables que permiten agrupar valores al momento de crearlas, y las listas son estructuras mutables que permiten agrupar valores, con la posibilidad de agregar, quitar o reemplazar sus elementos.

- Las tuplas se utilizan para modelar situaciones en las cuales al momento de crearlas ya se sabe cuál va a ser la información a almacenar. Por ejemplo, para representar una fecha, una carta de la baraja, una ficha de dominó.

- Las listas se utilizan en las situaciones en las que los elementos a agrupar pueden ir variando a lo largo del tiempo. Por ejemplo, para representar un las notas de un alumno en diversas materias, los inscritos para un evento o la clasificación de los equipos en una competencia

Ejercicios

Ejercicio 1. Escribir una función que reciba una tupla de elementos e indique si se encuentran ordenados de menor a mayor o no.

Ejercicio 2. Dominó.

Escribir una función que indique si dos fichas de dominó encajan o no. Las fichas son recibidas en dos tuplas, por ejemplo: (3,4) y (5,4).

Escribir una función que indique si dos fichas de dominó encajan o no. Las fichas son recibidas en una cadena, por ejemplo: 3-4 2-5. Nota: utilizar la función split de las cadenas.

Ejercicio 3. Campaña electoral

Escribir una función que reciba una tupla con nombres, y para cada nombre imprima el mensaje Estimado , vote por mí.

Escribir una función que reciba una tupla con nombres, una posición de origen p y una cantidad n, e imprima el mensaje anterior para los n nombres que se encuentran a partir de la posición p.

Modificar las funciones anteriores para que tengan en cuenta el género del destinatario, para ello, deberán recibir una tupla de tuplas, conteniendo el nombre y el género.

Ejercicio 4. Dada una lista de números enteros, escribir una función que:

Devuelva una lista con todos los que sean primos.

Devuelva la sumatoria y el promedio de los valores.

Devuelva una lista con el factorial de cada uno de esos números.

Ejercicios

Ejercicio 5 Dada una lista de números enteros y un entero k, escribir una función que:

Devuelva tres listas, una con los menores, otra con los mayores y otra con los iguales a k.

Devuelva una lista con aquellos que son múltiplos de k.

Ejercicio 6. Escribir una función que reciba una lista de tuplas (Apellido, Nombre, Inicial_ segundo_ nombre) y devuelva una lista de cadenas donde cada una contenga primero el nombre, luego la inicial con un punto, y luego el apellido.

Ejercicio 7. Inversión de listas

Realizar una función que, dada una lista, devuelva una nueva lista cuyo contenido sea igual a la original pero invertida. Así, dada la lista ['Di', 'buen', 'día', 'a', 'papa'], deberá devolver ['papa', 'a', 'día', 'buen', 'Di'].

Realizar otra función que invierta la lista, pero en lugar de devolver una nueva, modifique la lista dada para invertirla, si usar listas auxiliares.

Ejercicio 8. Escribir una función empaquetar para una lista, donde epaquetar significa indicar la repetición de valores consecutivos mediante una tupla (valor, cantidad de repeticiones). Por ejemplo, empaquetar ([1, 1, 1, 3, 5, 1, 1, 3, 3]) debe devolver [(1, 3) , (3, 1) , (5, 1), (1, 2), (3, 2)].