UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Guía Rápida de Listas

IWI 131: Programación

Departamento de Informática
Universidad Técnica Federico Santa María

Creación de listas

`[ ]`	Lista vacía
`[1,2,3]`	Lista con elementos
`list(s)`	Lista de caracteres (*strings*) creado con los caracteres que forman el *string* `s`

Creación de listas como Rangos

`list(range(n))`	Crea una lista con los números naturales desde 0 hasta $n-1$; el parámetro n es positivo
`list(range(a,b))`	Crea una lista con los números enteros entre a y $b-1$; se supone que $a < b$
`list(range(a,b,paso))`	Crea una lista con los números enteros entre a y $b-1$; saltando *paso* entre uno y otro; si $a < b$, entonces *paso* es positivo, de lo contrario *paso* debe ser negativo

Funciones sobre listas

`len(l)`	Retorna la longitud de la lista `l`, es decir, la cantidad de elementos que contiene
`sum(l)`	Retorna la suma de los elementos de la lista `l`, suponiendo que son números y que es posible sumarlos
`min(l)`	Retorna el valor mínimo de los elementos de la lista `l`, suponiendo que la comparación es posible
`max(l)`	Retorna el valor máximo de los elementos de la lista `l`, suponiendo que la comparación es posible

Notas

A diferencia de los *strings*, las listas sí son mutables, es decir, se puede asignar nuevos valores a los elementos de una lista existente, se pueden agregar nuevos elementos o eliminar elementos existentes.

Operadores sobre listas

`+`	Concatena dos listas, creando una nueva Ej: `lista = [1,5] + [2,4]`
`*`	Concatena una lista, consigo misma, una cantidad de veces, creando una nueva Ej: `lista = 3 * [1,2,3]`
`[ ]`	Recupera el elemento que se encuentra en una posición particular de una lista; el índice debe ser válido Ej: `print(numeros[5])` Índices de izq. a der.: $0, 1, 2, \ldots$ Índices de der. a izq.: $-1, -2, -3, \ldots$
`[a:b]`	Crea una nueva lista como una *rebanada* de una lista existente; comenzando en el índice `a` y terminando uno antes de `b`. Ej: `sublista = lista[5:10]`
`[:b]`	Crea una nueva lista como una *rebanada* al inicio de una lista existente, terminando uno antes de `b`. Ej: `prefijo = lista[:10]`
`[a:]`	Crea una nueva lista como una *rebanada* al final de una lista existente, comenzando en el índice `a`. Ej: `sufijo = lista[a:]`
`[a:b:paso]`	Crea una nueva lista como una *rebanada* de una lista existente; comenzando en el índice `a` y terminando uno antes de `b`, saltando `paso` entre cada par de elementos; por ejemplo, si `paso` es 2, lo hace elemento de por medio; si $a > b$ entonces `paso` debiera ser negativo Ej: `sublista = lista[2:20:3]` Ej: `sublista = lista[20:2:-2]`
`e in l`	Retorna `True` si el elemento `e` está contenido en la lista `l`; de lo contrario retorna `False` Ej: `if 5 in numeros:` Puede utilizarse `not in` para preguntar en negativo
`del l[ind]`	Elimina el elemento con índice `ind` de la lista `l`, desplazando hacia la izquierda los elementos restantes; índice debe ser válido

Métodos sobre listas

`l.append(e)`	Agrega el elemento `e` al final de la lista `l` Ej: `numeros.append(5)`
`l.insert(pos,e)`	Inserta el elemento `e` en la posición `pos` de la lista `l`, desplazando los elementos hacia la derecha Ej: `numeros.insert(3,5)`
`l.index(e)`	Retorna el índice de la primera ocurrencia del elemento `e` en la lista `l`; si no está se produce un error Ej: `pos = numeros.index(10)`
`l.count(e)`	Retorna la cantidad de veces que el elemento `e` aparece en la lista `l`; si no está retorna 0 Ej: `veces = numeros.count(10)`
`l.remove(e)`	Elimina la primera ocurrencia del elemento `e` en la lista `l`; si no está se produce un error Ej: `numeros.remove(10)`
`l.sort()`	Ordena ascendentemente (de menor a mayor) la lista `l`; utilizando los criterios de ordenamiento que correspondan Ej: `edades.sort()` # números Ej: `nombres.sort()` # strings
`l.reverse(e)`	Invierte los elementos de la lista `l`; se puede usar después de `l.sort()` para ordenar descendentemente (de mayor a menor) Ej: `edades.reverse()`

Recorrido de listas con `while`

```
i = 0
while i < len(lista):
   print(lista[i])
   i = i + 1
```

Recorrido de listas con `for`

```
for elemento in lista:
   print(elemento)
```