



ESCUELA DE INGENIERÍA



Aplicaciones de listas



PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE



Cristian Ruz Ruz

Profesor Asistente Adjunto, Departamento de
Ciencia de la Computación
Facultad de Ingeniería
Pontificia Universidad Católica de Chile

Índice



Ejemplos de uso de listas



Recomendaciones y cuidados

Introducción

- Para qué sirven las listas.
- Qué cuidados debemos tener al usar listas.



Ejemplos de uso de listas



Uso de listas:

1

Elementos homogéneos, ordenados,
que queremos recorrer sistemáticamente

```
estudiantes = ['Aurora', 'Sebastián', 'Rafaela', 'Dario', 'Lisa',  
'Almendra', 'Camilo']  
puntos = [67, 56, 48, 21, 60, 38, 89, 20, 28, 79]  
temperaturas = [16.8, 35.2, 27.9, 19.2, 19.8, 24.8, 34.3, 19.0, 37.3,  
20.6]  
respuestas = ['a', 'c', 'c', 'b', 'd', 'd', 'a', 'c', 'e', 'b']
```



Elementos homogéneos



Elementos ordenados



Recorrido sistemático

¿Cuál es el nombre más largo?

¿Cuál es el promedio de puntos?

¿Cuándo ha habido temperaturas
mayores a 30 grados?

Uso de listas:

Buscar
elemento(s)
que cumple(n)
una condición

2

Buscar nombres más largos

```
estudiantes = ['Aurora', 'Sebastián', 'Rafaela', 'Dario', 'Lisa',  
'Almendra', 'Camilo']
```

¿Cuál es nombre más largo? ¿En qué posición está?

```
pos_mas_largo = 0  
mas_largo = estudiantes[0]  
  
for i in range(1, len(estudiantes)):  
    if len(estudiantes[i]) >= len(mas_largo):  
        pos_mas_largo = i  
        mas_largo = estudiantes[i]  
  
print(f"El nombre más largo es {mas_largo} y se encuentra en la \\  
posición {pos_mas_largo}")
```

El nombre más largo es Sebastián y se encuentra en la posición 1

Uso de listas:

Calcular
promedio de
elementos

3

Calcular promedio (u otra operación) de elementos

```
puntos = [67, 56, 48, 21, 60, 38, 89, 20, 28, 79]
```

¿Cuál es el promedio de puntos?

```
suma = 0
for elem in puntos:
    suma += elem
promedio = suma/len(puntos)
print(f"El promedio de puntos es {promedio}")
```

El promedio de puntos es 50.6

```
promedio = sum(puntos)/len(puntos)
print(f"El promedio de puntos es {promedio}")
```

El promedio de puntos es 50.6

4

Calcular promedio de elementos que cumplen una condición

```
puntos = [67, 56, 48, 21, 60, 38, 89, 20, 28, 79]
```

¿Cuál es el promedio de los puntajes que son mayores a 60?

```
limite_inferior = 60
suma = 0
num_elementos = 0
for elem in puntos:
    if elem >= limite_inferior:
        suma += elem
        num_elementos += 1
promedio = suma / num_elementos
print(f"El promedio de puntos mayores a {limite_inferior} es \
{promedio}")
```

El promedio de puntos mayores a 60 es 73.75

¿Y, si no hay elementos mayores a 60?
¿Cuál es el elemento más cercano al promedio?

Uso de listas:

Calcular
promedio de
elementos
que cumplen
una condición

5

Extraer elementos que cumplen una condición

```
temperaturas = [16.8, 35.2, 27.9, 19.2, 19.8, 24.8, 34.3, 19.0, 37.3, 20.6]
```

¿Cuándo ha habido temperaturas mayores a 30 grados?

```
limite = 30
dias_altas = []
altas = []
for i in range(len(temperaturas)):
    if temperaturas[i] >= limite:
        dias_altas.append(i)
        altas.append(temperaturas[i])
print(f"Los días {dias_altas} hubo temperaturas mayores a {limite} y\
fueron {altas}")
```

Los días [1, 6, 8] hubo temperaturas mayores a 30 y fueron [35.2, 34.3, 37.3]

Uso de listas:

Construir lista
con
elementos
que cumplen
una condición

Uso de listas:

Buscar un
patrón en una
sub lista

6

Buscar un patrón en una porción de la lista

```
respuestas = ['a', 'c', 'c', 'b', 'd', 'd', 'a', 'c', 'e', 'b']
```

¿Hay dos respuestas iguales seguidas?

```
dos_iguales = []  
for i in range(len(respuestas)-1):  
    if respuestas[i] == respuestas[i+1]:  
        dos_iguales = respuestas[i:i+2]  
print(f"Respuestas consecutivas iguales: {dos_iguales}")
```

Respuestas consecutivas iguales: ['d', 'd']

¿Y, si queremos todas las respuestas
consecutivas iguales, y su posición?

Uso de listas:

Buscar un
patrón en una
sub lista

7

Buscar un patrón en una porción de la lista

```
respuestas = ['a', 'c', 'c', 'b', 'd', 'd', 'a', 'c', 'e', 'b']
```

¿Hay dos respuestas iguales seguidas?

```
pos_iguales = []
dos_iguales = []

for i in range(len(respuestas)-1):
    if respuestas[i] == respuestas[i+1]:
        dos_iguales.append(respuestas[i:i+2])
        pos_iguales.append([i, i+1])
print(f"Respuestas consecutivas iguales: {dos_iguales}, en las \
posiciones: {pos_iguales}")
```

Respuestas consecutivas iguales: [['c', 'c'], ['d', 'd']], en las
posiciones: [[1, 2], [4, 5]]

¿Cómo obtener las secuencias de n respuestas iguales consecutivas?

Recomendaciones al usar listas



Modificar la lista mientras se recorre

1

No modificar una lista mientras la recorremos

```
respuestas = ['a', 'c', 'c', 'b', 'd', 'd', 'a', 'c', 'e', 'b']
```

Eliminar las respuestas que son 'c'



```
print(f"Respuestas: {respuestas}")
for elem in respuestas:
    if elem == 'c':
        respuestas.remove(elem)
print(f"Respuestas: {respuestas}")
```

```
Respuestas: ['a', 'c', 'c', 'b', 'd', 'd', 'a', 'c', 'e', 'b']
Respuestas: ['a', 'b', 'd', 'd', 'a', 'c', 'e', 'b']
```

2

No modificar una lista mientras la recorremos

```
respuestas = ['a', 'c', 'c', 'b', 'd', 'd', 'a', 'c', 'e', 'b']
```

Eliminar las respuestas que son 'c'

```
print(f"Respuestas: {respuestas}")
for i in range(len(respuestas)):
    if respuestas[i] == 'c':
        respuestas.pop(i)
print(f"Respuestas: {respuestas}")
```

```
Respuestas: ['a', 'c', 'c', 'b', 'd', 'd', 'a', 'c', 'e', 'b']
Traceback (most recent call last):
... ..
IndexError: list index out of range
```

MORALEJA

NUNCA agregar o sacar elementos de la lista mientras se recorre.

**Modificar la
lista mientras
se recorre**


Modificar la lista mientras se recorre

3

Crear una nueva lista sin los elementos que no queremos

```
respuestas = ['a', 'c', 'c', 'b', 'd', 'd', 'a', 'c', 'e', 'b']
```

Eliminar las respuestas que son 'c'

```
respuestas_sin_c   
print(f"Respuestas: {respuestas}")  
for elem in respuestas:  
    if elem != 'c':  
        respuestas_sin_c.append(elem)  
print(f"Respuestas: {respuestas_sin_c}")
```

```
Respuestas: ['a', 'c', 'c', 'b', 'd', 'd', 'a', 'c', 'e', 'b']  
Respuestas: ['a', 'b', 'd', 'd', 'a', 'e', 'b']
```

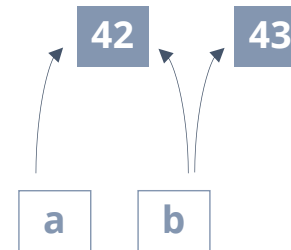

Copiar listas

1

Al asignar listas, se asignan referencias

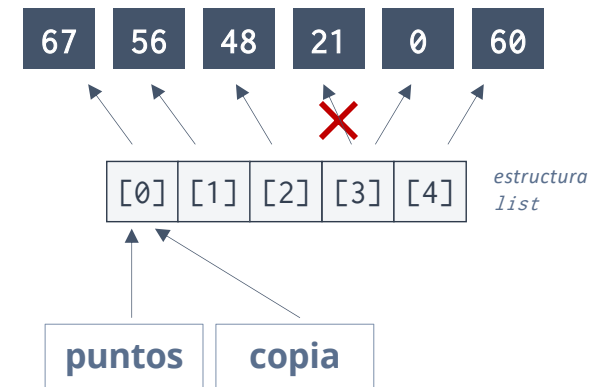
```
a = 42
b = a
b = b+1
print(f"a={a}, b={b}")
```

a=42, b=43



```
puntos = [67, 56, 48, 21, 60]
copia = puntos
copia[3] = 0
print(f"Puntos: {puntos}")
print(f"Copia : {copia}")
```

Puntos: [67, 56, 48, 0, 60]
Copia : [67, 56, 48, 0, 60]



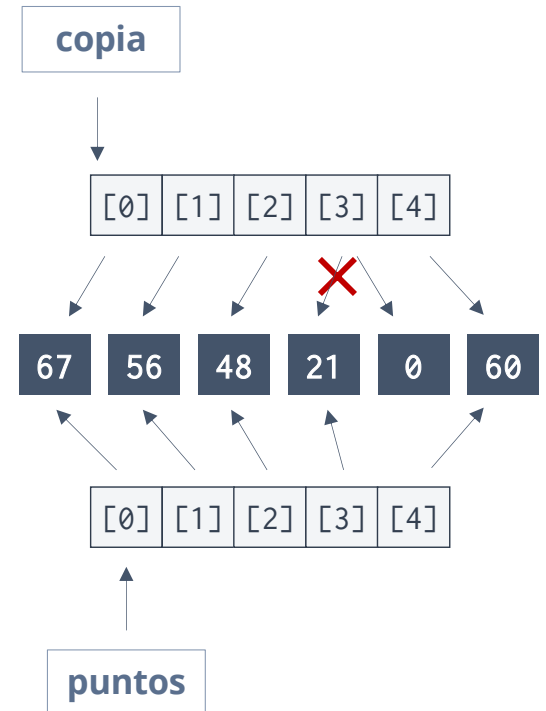
Copiar listas (I)

2

Copiar listas mediante *slicing*

```
puntos = [67, 56, 48, 21, 60]  
copia = puntos[:]  
copia[3] = 0  
print(f"Puntos: {puntos}")  
print(f"Copia : {copia}")
```

Puntos: [67, 56, 48, 21, 60]
Copia : [67, 56, 48, 0, 60]



Copiar listas (II y III)

3

Copiar listas mediante constructor `list()`, o método `copy()`

```
puntos = [67, 56, 48, 21, 60]
copia = list(puntos)
copia[3] = 0
print(f"Puntos: {puntos}")
print(f"Copia : {copia}")
```

```
Puntos: [67, 56, 48, 21, 60]
Copia : [67, 56, 48, 0, 60]
```

```
puntos = [67, 56, 48, 21, 60]
copia = puntos.copy()
copia[4] = 0
print(f"Puntos: {puntos}")
print(f"Copia : {copia}")
```

```
Puntos: [67, 56, 48, 21, 60]
Copia : [67, 56, 48, 21, 0]
```

Ordenar listas (I)

1

Copia ordenada de listas

```
puntos = [67, 56, 48, 21, 60, 38, 89, 20, 28, 79]  
ordenada = sorted(puntos)  
print(f"Puntos: {puntos}")  
print(f"Ordenada: {ordenada}")
```

```
Puntos:  [67, 56, 48, 21, 60, 38, 89, 20, 28, 79]  
Ordenada: [20, 21, 28, 38, 48, 56, 60, 67, 79, 89]
```

Ordenar listas (II)

2

Ordenar una lista

```
puntos = [67, 56, 48, 21, 60, 38, 89, 20, 28, 79]  
puntos.sort()  
print(f"Puntos: {puntos}")
```

Puntos: [20, 21, 28, 38, 48, 56, 60, 67, 79, 89]

```
puntos = [67, 56, 48, 21, 60, 38, 89, 20, 28, 79]  
puntos.sort(reverse=True)  
print(f"Puntos: {puntos}")
```

Puntos: [89, 79, 67, 60, 56, 48, 38, 28, 21, 20]

Síntesis

- Uso de listas.
- Datos homogéneos y ordenados.
- Búsqueda, selección, recorrido sistemático, construcción.
- Modificación y copia de listas.
- Ordenar listas.



Referencias bibliográficas

- Data Structures. The Python Tutorial, v3.8.2rc1. Recuperado de:
<https://docs.python.org/3/tutorial/datastructures.html>



PONTIFICIA
UNIVERSIDAD
CATÓLICA
DE CHILE