CONQUER BLOCKS

PYTHON

LISTAS Y ESTRUCTURAS ITERATIVAS (PARTE 2)

REPASO CLASE ANTERIOR

1. Listas como una colección dinámica de elementos o objetos

    - Acceder a los elementos
    - Añadir elementos
    - Borrar elementos

2. Estructuras iterativas o bucles

    - Bucle while
    - Bucle for
    - Uso y contexto de *break* y *continue*

LISTAS Y ESTRUCTURAS ITERATIVAS

RECORRER UNA LISTA CON UN BUCLE

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

```
coches = ['bmw', 'audi', 'seat']
for coche in coches:
    print(coche)
```

✓ 0.1s

```
bmw
audi
seat
```

LISTAS Y ESTRUCTURAS ITERATIVAS

RECORRER UNA LISTA CON UN BUCLE

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

```
for i in range(0,len(lista)):
    print(lista[i])
```

```
coches = ['bmw', 'audi', 'seat']
for coche in coches:
    print(coche)
```

✓ 0.1s

```
bmw
audi
seat
```

```
coches = ['bmw', 'audi', 'seat']
for i in range(0,len(coches)):
    print(coches[i])
```

✓ 0.7s

```
bmw
audi
seat
```

LISTAS Y ESTRUCTURAS ITERATIVAS

RECORRER UNA LISTA CON UN BUCLE

```
coches = ['bmw', 'audi', 'seat']
for i in range(0,len(coches)):
    print('este coche es un ',coches[i].title())
print('Lista de coches terminada')
```

✓ 0.1s

```
este coche es un  Bmw
este coche es un  Audi
este coche es un  Seat
Lista de coches terminada
```

LISTAS Y ESTRUCTURAS ITERATIVAS

RECORRER UNA LISTA CON UN BUCLE

```
coches = ['bmw', 'audi', 'seat']
for i in range(0,len(coches)):
    print('este coche es un ',coches[i].title())
print('Lista de coches terminada')
```

✓ 0.1s

```
este coche es un  Bmw
este coche es un  Audi
este coche es un  Seat
Lista de coches terminada
```

DENTRO DEL BUCLE

FUERA DEL BUCLE

LISTAS Y ESTRUCTURAS ITERATIVAS

RECORRER UNA LISTA CON UN BUCLE

```
coches = ['bmw', 'audi', 'seat']
for i in range(0,len(coches)):
    print('este coche es un ',coches[i].title())
    print('Lista de coches terminada')
```

✓ 0.4s

```
este coche es un  Bmw
Lista de coches terminada
este coche es un  Audi
Lista de coches terminada
este coche es un  Seat
Lista de coches terminada
```

DENTRO DEL BUCLE

DENTRO DEL BUCLE

LISTAS Y ESTRUCTURAS ITERATIVAS

DEBUGGING

```
coches = ['bmw', 'audi', 'seat']
for i in range(0,len(coches)):
print('este coche es un ',coches[i].title())
```

⊗ 0.7s

```
  Cell In[66], line 3
    print('este coche es un ',coches[i].title())
    ^
IndentationError: expected an indented block
```

FALTA DE SANGÍA TRAS EL LA DECLARACIÓN DEL FOR

LISTAS Y ESTRUCTURAS ITERATIVAS

DEBUGGING

```
coches = ['bmw', 'audi', 'seat']
for i in range(0,len(coches))
    print('este coche es un ',coches[i].title())
```

⊗ 0.8s

```
  Cell In[69], line 2
    for i in range(0,len(coches))
                                 ^
SyntaxError: invalid syntax
```

ERROR DE SINTAXIS. FALTAN LOS DOS PUNTOS TRAS EL FOR

LISTAS Y ESTRUCTURAS ITERATIVAS

DEBUGGING

```
    print('Aqui no hay sangría')
        print('Aquí hay sangría')
⊗ 0.1s

  Cell In[67], line 2
    print('Aquí hay sangría')
    ^
IndentationError: unexpected indent
```

SANGRÍA INESPERADA/
INJUSTIFICADA

LISTAS NUMÉRICAS

```
numeros = list(range(1,6))
print(numeros)
```

✓ 0.1s

```
[1, 2, 3, 4, 5]
```

```
numeros_cuadrados = []
for valor in range(1,11): # irá del 0 al 10
    cuadrado = valor**2
    numeros_cuadrados.append(cuadrado)
print(numeros_cuadrados)
```

✓ 0.9s

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

```
numeros_pares = list(range(2,11,2))
print(numeros_pares)
```

✓ 0.7s

```
[2, 4, 6, 8, 10]
```

Podemos usar la función **range()** para crear listas numéricas

LISTAS NUMÉRICAS - COMPRESIÓN

Declaración extendida

```
numeros_cuadrados = []
for valor in range(1,11): # irá del 0 al 10
    cuadrado = valor**2
    numeros_cuadrados.append(cuadrado)
print(numeros_cuadrados)
```

✓ 0.9s

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Declaración comprimida

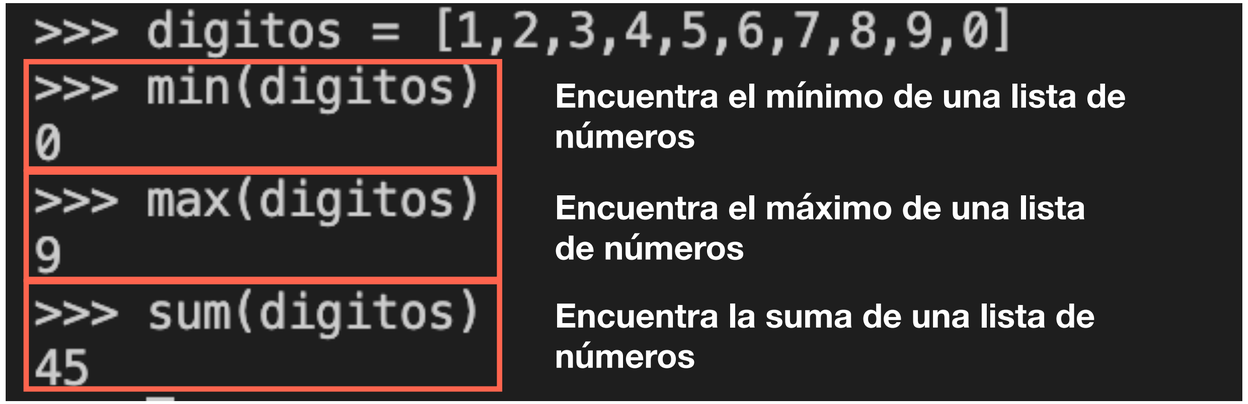
```
numeros_cuadrados = [valor**2 for valor in range(1,11)]
print(numeros_cuadrados)
```

✓ 0.4s

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

LISTAS NUMÉRICAS

En Python existen funciones específicas para tratar con listas numéricas.



PARTES DE UNA LISTA

Porción de una lista

```
digitos = [1,2,3,4,5,6,7,8,9,0]
algunos_digitos = digitos[2:5]
print(algunos_digitos)
```

✓ 0.8s

```
[3, 4, 5]
```

PARTES DE UNA LISTA

Porción de una lista

```
                0 1 2 3 4 5 6 7 8 9
digitos = [1,2,3,4,5,6,7,8,9,0]
algunos_digitos = digitos[2:5]
print(algunos_digitos)
```

✓ 0.8s

```
[3, 4, 5]
```

PARTES DE UNA LISTA

Porción de una lista

```
                0 1 2 3 4 5 6 7 8 9
digitos = [1,2,3,4,5,6,7,8,9,0]
algunos_digitos = digitos[2:5]
print(algunos_digitos)
```

✓ 0.8s

```
[3, 4, 5]
```

```
jugadores = ['Alejandro', 'Felipe', 'Samuel', 'Juan Marcos', 'Lucas', 'David']
equipo_A = jugadores[0:3]
equipo_B = jugadores[3:6]

print(equipo_A)
print(equipo_B)
```

✓ 0.1s

```
['Alejandro', 'Felipe', 'Samuel']
['Juan Marcos', 'Lucas', 'David']
```

PARTES DE UNA LISTA

Porción de una lista

```
                0 1 2 3 4 5 6 7 8 9
digitos = [1,2,3,4,5,6,7,8,9,0]
algunos_digitos = digitos[2:5]
print(algunos_digitos)
```

✓ 0.8s

```
[3, 4, 5]
```

```
jugadores = ['Alejandro', 'Felipe', 'Samuel', 'Juan Marcos', 'Lucas', 'David']
equipo_A = jugadores[:3]
equipo_B = jugadores[0:3]

print(equipo_A)
print(equipo_B)
```

✓ 0.2s

```
['Alejandro', 'Felipe', 'Samuel']
['Alejandro', 'Felipe', 'Samuel']
```

PARTES DE UNA LISTA

Porción de una lista

```
                0 1 2 3 4 5 6 7 8 9
digitos = [1,2,3,4,5,6,7,8,9,0]
algunos_digitos = digitos[2:5]
print(algunos_digitos)
```

✓ 0.8s

```
[3, 4, 5]
```

```
jugadores = ['Alejandro', 'Felipe', 'Samuel', 'Juan Marcos', 'Lucas', 'David']
equipo_A = jugadores[3:]
equipo_B = jugadores[3:6]

print(equipo_A)
print(equipo_B)
```

✓ 0.1s

```
['Juan Marcos', 'Lucas', 'David']
['Juan Marcos', 'Lucas', 'David']
```

PARTES DE UNA LISTA

Porción de una lista

```
                0 1 2 3 4 5 6 7 8 9
digitos = [1,2,3,4,5,6,7,8,9,0]
algunos_digitos = digitos[2:5]
print(algunos_digitos)
```

✓ 0.8s

```
[3, 4, 5]
```

```
jugadores = ['Alejandro', 'Felipe', 'Samuel', 'Juan Marcos', 'Lucas', 'David']
equipo_A = jugadores[3:6]
equipo_B = jugadores[-3:]

print(equipo_A)
print(equipo_B)
```

✓ 0.1s

```
['Juan Marcos', 'Lucas', 'David']
['Juan Marcos', 'Lucas', 'David']
```

PARTES DE UNA LISTA

Porción de una lista

```
                0 1 2 3 4 5 6 7 8 9
digitos = [1,2,3,4,5,6,7,8,9,0]
algunos_digitos = digitos[2:5]
print(algunos_digitos)
```

✓ 0.8s

```
[3, 4, 5]
```

```
jugadores = ['Alejandro', 'Felipe', 'Samuel', 'Juan Marcos', 'Lucas', 'David']
equipo_A = jugadores[3:6]
equipo_B = jugadores[-3:]

print(equipo_A)
print(equipo_B)
```

✓ 0.1s

```
['Juan Marcos', 'Lucas', 'David']
['Juan Marcos', 'Lucas', 'David']
```

PARTES DE UNA LISTA

Porción de una lista

```
                0 1 2 3 4 5 6 7 8 9
digitos = [1,2,3,4,5,6,7,8,9,0]
algunos_digitos = digitos[2:5]
print(algunos_digitos)
```

✓ 0.8s

```
[3, 4, 5]
```

```
                -6        -5        -4          -3          -2        -1
jugadores = ['Alejandro', 'Felipe', 'Samuel', 'Juan Marcos', 'Lucas', 'David']
equipo_A = jugadores[3:6]
equipo_B = jugadores[-3:]

print(equipo_A)
print(equipo_B)
```

✓ 0.1s

```
['Juan Marcos', 'Lucas', 'David']
['Juan Marcos', 'Lucas', 'David']
```

RECORRER UNA PARTE DE UNA LISTA

```
jugadores = ['Alejandro', 'Felipe', 'Samuel', 'Juan Marcos', 'Lucas', 'David']

print('Estos son los jugadores del equipo A:')
for jugador in jugadores[0:3]:
    print(jugador)
```

✓ 0.1s

```
Estos son los jugadores del equipo A:
Alejandro
Felipe
Samuel
```

RECORRER UNA PARTE DE UNA LISTA

```
jugadores = ['Alejandro', 'Felipe', 'Samuel', 'Juan Marcos', 'Lucas', 'David']

print('Estos son los jugadores del equipo A:')
for jugador in jugadores[0:3]:
    print(jugador)
```

✓ 0.1s

```
jugadores = ['Alejandro', 'Felipe', 'Samuel', 'Juan Marcos', 'Lucas', 'David']

print('Estos son los jugadores del equipo A:')
for i in range(len(jugadores[0:3])):
    print(jugadores[i])
```

✓ 0.2s

```
Estos son los jugadores del equipo A:
Alejandro
Felipe
Samuel
```

COPIAR UNA LISTA

Podemos crear nuevas listas a partir de listas ya existentes

```
mi_comida = ['pizza', 'carne', 'tarta de queso']
comida_invitado = mi_comida
print(mi_comida)
print(comida_invitado)
```

✓ 0.1s

```
['pizza', 'carne', 'tarta de queso']
['pizza', 'carne', 'tarta de queso']
```

COPIAR UNA LISTA

Podemos crear nuevas listas a partir de listas ya existentes

```
mi_comida = ['pizza', 'carne', 'tarta de queso']
comida_invitado = mi_comida
comida_invitado.append('helado')
print(mi_comida)
print(comida_invitado)
```

✓ 0.2s

```
['pizza', 'carne', 'tarta de queso', 'helado']
['pizza', 'carne', 'tarta de queso', 'helado']
```

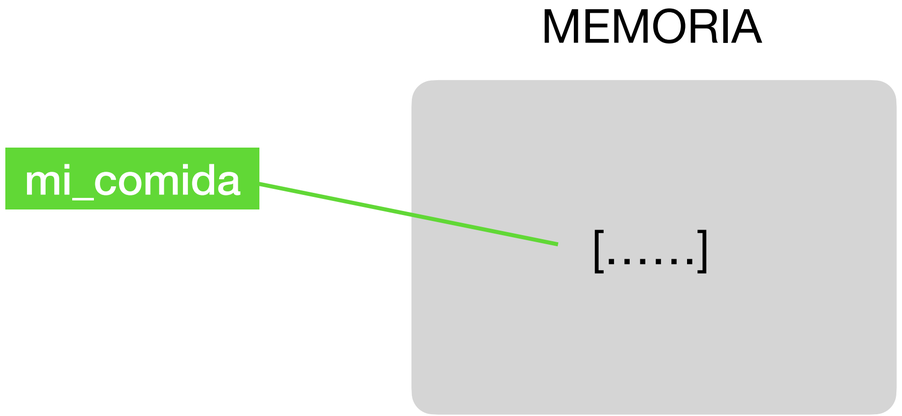
COPIAR UNA LISTA

Podemos crear nuevas listas a partir de listas ya existentes

```
mi_comida = ['pizza', 'carne', 'tarta de queso']
comida_invitado = mi_comida
comida_invitado.append('helado')
print(mi_comida)
print(comida_invitado)
```

✓ 0.2s

```
['pizza', 'carne', 'tarta de queso', 'helado']
['pizza', 'carne', 'tarta de queso', 'helado']
```



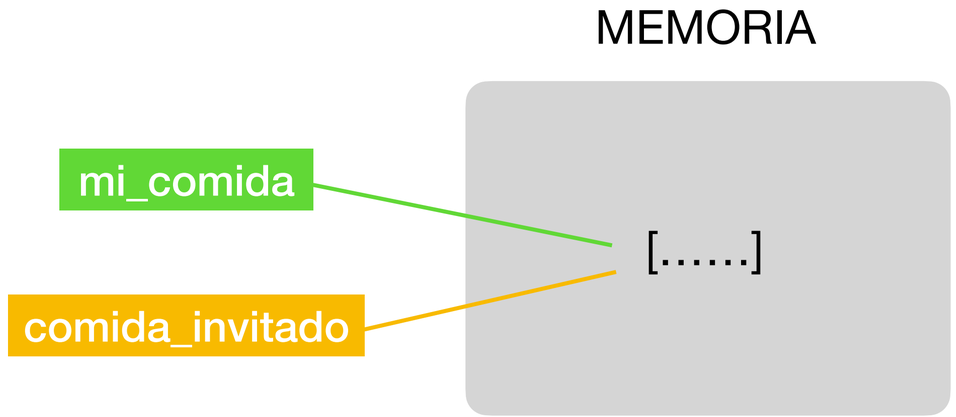
COPIAR UNA LISTA

Podemos crear nuevas listas a partir de listas ya existentes

```
mi_comida = ['pizza', 'carne', 'tarta de queso']
comida_invitado = mi_comida
comida_invitado.append('helado')
print(mi_comida)
print(comida_invitado)
```

✓ 0.2s

```
['pizza', 'carne', 'tarta de queso', 'helado']
['pizza', 'carne', 'tarta de queso', 'helado']
```



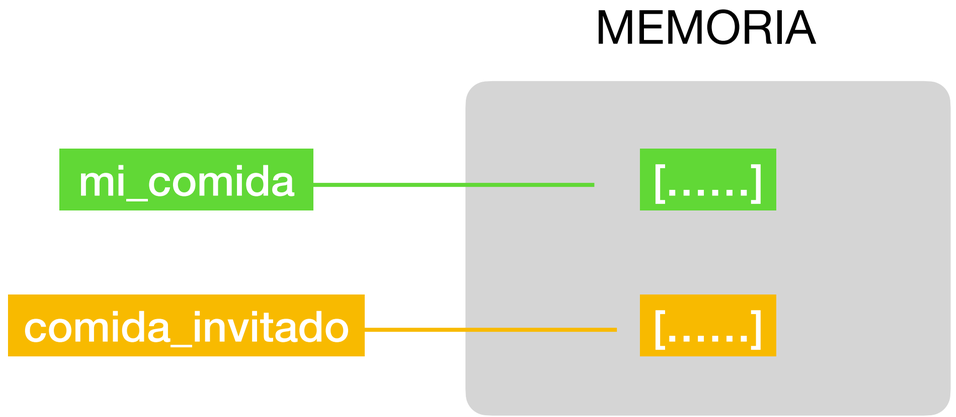
COPIAR UNA LISTA

Podemos crear nuevas listas a partir de listas ya existentes

```
mi_comida = ['pizza', 'carne', 'tarta de queso']
comida_invitado = mi_comida[:]
comida_invitado.append('helado')
print(mi_comida)
print(comida_invitado)
```

✓ 0.1s

```
['pizza', 'carne', 'tarta de queso']
['pizza', 'carne', 'tarta de queso', 'helado']
```



LISTAS ANIDADAS

Las listas son colecciones de objetos, pero a su vez son un objeto también. Igual que una lista puede contener objetos como strings o números también puede contener otras listas:

```
datos_alumnos = [['David', 27], ['Jose', 22], ['Lucas',23]]
print(type(datos_alumnos))
```

✓ 0.9s

```
<class 'list'>
```

LISTAS ANIDADAS

Las listas son colecciones de objetos, pero a su vez son un objeto también. Igual que una lista puede contener objetos como strings o números también puede contener otras listas:

```
datos_alumnos = [['David', 27], ['Jose', 22], ['Lucas',23]]
print(type(datos_alumnos))
print(datos_alumnos[0])
print(type(datos_alumnos[0]))
```

✓ 0.5s

```
<class 'list'>
['David', 27]
<class 'list'>
```

LISTAS ANIDADAS

Las listas son colecciones de objetos, pero a su vez son un objeto también. Igual que una lista puede contener objetos como strings o números también puede contener otras listas:

```
datos_alumnos = [['David', 27], ['Jose', 22], ['Lucas',23]]
print(type(datos_alumnos))
print(datos_alumnos[0][0])
print(datos_alumnos[0][1])
print(type(datos_alumnos[0][0]))
```

✓ 0.6s

```
<class 'list'>
David
27
<class 'str'>
```

LISTAS ANIDADAS

Las listas son colecciones de objetos, pero a su vez son un objeto también. Igual que una lista puede contener objetos como strings o números también puede contener otras listas:

```
datos_alumnos = [['David', 27], ['Jose', 22], ['Lucas',23]]
print(type(datos_alumnos))
print(datos_alumnos[0:2])
```

✓ 0.1s

```
<class 'list'>
[['David', 27], ['Jose', 22]]
<class 'str'>
```

REPASO

Listas = colecciones de objetos

- Acceder a los elementos
- Usar los elementos
- Modificar los elementos
- Añadir elementos
- Borrar elementos
- Ordenar elementos dentro de la lista
- Longitud de la lista

— Listas numéricas + funciones específicas
— Acceder a partes de una lista
— Copiar listas y partes de listas
— Anidamiento de listas

Estructura condicional + Lista → Comprobar existencia de elementos

REPASO

Bucles For y While

- Funcionamiento general
- Uso de *range*
- Usos de *break*
- Usos de *continue*

Estructura iterativa + Lista ⟶ Recorrer los elementos

CONQUER BLOCKS