</ Ayudantía 7:
Listas 1 />


Rolando Rojas

1 0 1 1  0 1 1  0 1  1 0 1 1 0 0 1  1 0  1 1 0 1 1  0 1 1  0 1  1 1 0 1 1 0  1 1 0 1 1 1  1 1 0 1

# </ Temas a tratar

{01}
Introducción a Listas

{02}
Creación, acceso y modificación de listas

{03}
Operaciones con listas

{04}
Nuevas funciones para listas

{05}
Recorriendo listas

{06}
Ejercicios
Preguntas
Tarea

1 0 1 1  0 1 1  0 1  1 0 1 1 0 0 1  1 0  1 1 0 1 1  0 1 1  0 1  1 1 0 1 1 0  1 1 0 1 1 1  1 1 0 1

# </ Introducción a Listas

*Una lista es una estructura de* ***datos*** *que permite almacenar una colección ordenada de elementos, que pueden ser de diferentes tipos (números, cadenas de texto, objetos, etc.), y que se identifican mediante índices numéricos.*
*Se definen utilizando corchetes [ ] y los elementos se separan por comas.*

```
string = "Quiero un completo"
lista = ["Quiero","un","completo"]
          0       1       2
```

```
>>> print(lista[0])
Quiero
```

```
>>> print(lista[0][1])
u
```

1 0 1 1  0 1 1  0 1  1 0 1 1 0 0 1  1 0  1 1 0 1 1  0 1 1  0 1  1 1 0 1 1 0  1 1 0 1 1 1  1 1 0 1

# </ Creación, acceso y modificación de listas

*Las listas, a diferencia de los strings, son **mutables**, lo que significa que se pueden modificar después de su creación mediante la adición, eliminación o modificación de elementos.*
*Las listas son una de las estructuras de datos más utilizadas en Python para el almacenamiento y manipulación de datos.*

Creación de lista vacia:

```
listaVacia = []
```

Creación y modificación:

```
>>> lista = ["Quiero","un","completo"]
>>> print(lista)
['Quiero', 'un', 'completo']
>>> lista[2] = "churrasco"
>>> print(lista)
['Quiero', 'un', 'churrasco']
```

1 0 1 1  0 1 1  0 1  1 0 1 1 0 0 1  1 0  1 1 0 1 1  0 1 1  0 1  1 1 0 1 1 0  1 1 0 1 1 1  1 1 0 1

# </ Operaciones con listas

| Operador | | Ejemplo |
|---|---|---|
| a + b | Concatena dos listas | [1, 2] + [3, 4]<br>= [1, 2, 3, 4] |
| n * a | Concatena consigo misma n veces | 3 * [1, 2, 3]<br>= [1, 2, 3, 1, 2, 3, 1, 2, 3] |
| a[·] | Elemento de la lista a en la posición [·] | list = [1, 2, 3, 4]<br>list [2] = [3] |
| [a:b] | Sublista a la existente de [a] a [b-1] | sublist = list[0:2]<br>sublist = [1, 2] |
| [a:] | Sublista a la existente de [a] en adelante | sublist = list[1:]<br>sublist = [2, 3, 4] |
| [a:b:paso] | Sublista a la existente de [a] a [b-1] saltando paso | sublist = list[0:3:2]<br>sublist = [1, 3] |

1 0 1 1  0 1 1  0 1  1 0 1 1 0 0 1  1 0  1 1 0 1 1  0 1 1  0 1  1 1 0 1 1 0  1 1 0 1 1 1  1 1 0 1

# </ Consideraciones (1/2): Nuevas funciones para listas

```
>>> lista = ["string",16,True,10.4]
```

## l.append(x)

Sirve para añadir un elemento x a una lista:

```
>>> lista.append(1000)
>>> print(lista)
['string', 16, True, 10.4, 1000]
```

## l.remove(x)

Sirve para borrar un elemento x de una lista: (borrará solamente el primero que coincida)

```
>>> lista.remove(16)
>>> print(lista)
['string', True, 10.4, 1000]
```

## l.reverse()

Revierte el orden de la lista:

```
>>> lista.reverse()
>>> print(lista)
[1000, 10.4, True, 'string']
```

## l.sort()

Ordena los elementos de menor a mayor:

```
>>> lista_numeros = [1000,3,35,64,20]
>>> lista_numeros.sort()
>>> print(lista_numeros)
[3, 20, 35, 64, 1000]
```

1 0 1 1  0 1 1  0 1  1 0 1 1 0 0 1  1 0  1 1 0 1 1  0 1 1  0 1  1 1 0 1 1 0  1 1 0 1 1 1  1 1 0 1

# </ Consideraciones (2/2): Nuevas funciones para listas

```
>>> lista = ["string",16,True,10.4]
```

**l.count(x)** Sirve para contar la cantidad de elementos x en una lista:

```
>>> lista.count(16)
1
```

**l.index(x)** Sirve para buscar el índice del elemento x (obtendrá el primero que aparezca)

```
>>> lista.index(True)
2
```

**l.insert(i,x)** Inserta el elemento x en la posición i:

```
>>> lista.insert(1,"hola")
>>> print(lista)
['string', 'hola', 16, True, 10.4]
```

1 0 1 1  0 1 1  0 1  1 0 1 1 0 0 1  1 0  1 1 0 1 1  0 1 1  0 1  1 1 0 1 1 0  1 1 0 1 1 1  1 1 0 1

# </ Recorriendo listas con ciclos

```
>>> lista = ["string",16,True,10.4]
```

0 1 2 3

Ciclo for:

```
>>> lista = ["string",16,True,10.4]
>>> for elemento in lista:
...     print(elemento)
...
string
16
True
10.4
```

Ciclo while:

```
>>> lista = ["string",16,True,10.4]
>>> i=0
>>> while i < len(lista):
...     print(lista[i])
...     i+=1
...
string
16
True
10.4
```

1 0 1 1  0 1 1  0 1  1 0 1 1 0 0 1  1 0  1 1 0 1 1  0 1 1  0 1  1 1 0 1 1 0  1 1 0 1 1 1  1 1 0 1

# </ Ejercicio 1

*La parada de camioneros PyTrucks, dedicada a prestar estacionamiento y hospedajes a los camioneros de todo Chile, ha decidido cambiar su política de cobros, intentando hacerla más justa.*
*El nuevo sistema consiste en cobrar en base al tiempo exacto de estadía en horas y minutos de un camionero cualquiera. Para ello han decidido contratarlos a ustedes, los estudiantes de programación, para crear la función tiempo(hora_ingreso,hora_salida) donde hora_ingreso y hora_salida corresponden a listas de la forma [hh,mm,ss stop] con hh, mm, ss en entero, y retorne la diferencia de tiempo en el mismo formato.*
*Considere que un camionero puede llegar un día e irse al siguiente, pero nunca pasará más de 24 horas en la parada.*

*Ejemplo:*

```
>>> hora_ingreso=[23, 11, 25]
>>> hora_salida=[7, 22, 54]
>>> tiempo(hora_ingreso, hora_salida)
(8, 11, 29)
```

1 0 1 1  0 1 1  0 1  1 0 1 1 0 0 1  1 0  1 1 0 1 1  0 1 1  0 1  1 1 0 1 1 0  1 1 0 1 1 1  1 1 0 1

# </ Ejercicio 1: solución

```
def tiempo(hora_ingreso, hora_salida):
        h1, m1, s1 = hora_ingreso
        h2, m2, s2 = hora_salida

        t1 = (h1*60 + m1)*60 + s1
        t2 = (h2*60 + m2)*60 + s2
        delta = t2 - t1

        if hora_ingreso > hora_salida:
                delta += 24*60*60

        h_delta = delta // 3600
        m_delta = (delta % 3600) // 60
        s_delta = (delta % 3600) % 60

        return (h_delta, m_delta, s_delta)

print ('Tiempo resultante:', tiempo( [23, 11, 25], [7, 22, 54]))
```

1 0 1 1  0 1 1  0 1  1 0 1 1 0 0 1  1 0  1 1 0 1 1  0 1 1  0 1  1 1 0 1 1 0  1 1 0 1 1 1  1 1 0 1

# </ Ejercicio 2

*La búsqueda binaria encuentra la posición de un elemento en una lista ordenada, comparando el elemento con el valor de al medio de la lista, si no son iguales, la mitad en la cual el valor no puede estar es eliminada y la búsqueda continúa en la mitad restante hasta que el valor se encuentre.*
*Por ejemplo, se desea encontrar el 10 en la siguiente lista:*

| 4 | 6 | 10 | 12 | 17 | 25 | 29 |
|---|---|---|---|---|---|---|

*Comparar el elemento buscado con el valor central: 10 < 12, es menor, entonces descartar el lado derecho:*

| 4 | 6 | 10 | | | | |
|---|---|---|---|---|---|---|

*Comparar el elemento buscado con el valor central: 10 > 6, es mayor, entonces descartar el lado izquierdo:*

| | 6 | | | | | |
|---|---|---|---|---|---|---|

*Comparar el elemento buscado con el valor central: 10 = 10, es igual, entonces se encontró. Si hubiesen sido distintos, como no quedan más elementos, el valor buscado no está en la lista.*

*Escriba la función busqueda_binaria(lista, elemento)que recibe una lista ordenada y un elemento que se desea encontrar. La función debe retornar True si encuentra el elemento en la lista utilizando una búsqueda binaria. Si no se encuentra retornar False.*

```
>>> busqueda_binaria([0, 1, 3, 8, 14, 18, 19, 34, 52], 3)
True
>>> busqueda_binaria([0, 1, 3, 8, 14, 18, 19, 34, 52], 17)
False
```

1 0 1 1  0 1 1  0 1  1 0 1 1 0 0 1  1 0  1 1 0 1 1  0 1 1  0 1  1 1 0 1 1 0  1 1 0 1 1 1  1 1 0 1

# </ Ejercicio 2: solución

```
def busqueda_binaria(lista, elemento):
    inicio = 0
    fin = len(lista) - 1

    while inicio <= fin:
        medio = (inicio + fin) // 2
        if lista[medio] == elemento:
            return True
        elif lista[medio] < elemento:
            inicio = medio + 1
        else:
            fin = medio - 1
    return False
```

1 0 1 1  0 1 1  0 1  1 0 1 1 0 0 1  1 0  1 1 0 1 1  0 1 1  0 1  1 1 0 1 1 0  1 1 0 1 1 1  1 1 0 1

# </ Ejercicio 3

*Saul Pydman tiene una agenda muy ocupada luego de haber ganado sus casos en la corte. Es por eso que necesita un estudiante de programación que le ayude con su agenda.*

*Para eso, él tiene una lista con todos los interesados de la semana y su día de preferencia, en el siguiente formato "[contacto, día]"*

*Además, tiene otra lista con sus horarios disponibles.*

*A continuación, ayuda a Jimmy dejando su lista ordenada por día, considera que la lista de interesados está ordenada por orden de llamada, en caso de no tener disponibilidad en ese día, agenda una hora para el día siguiente*

*Ejemplo:*

```
>>> Reservas_Saul
[['Jhonny Kurt', 'lunes'], ['Joseph Pyrdiola', 'viernes'], ['Daniel Pydman', 'jueves'], ['J
oseph Curry', 'martes'], ['Vicent Van Gogh', 'domingo'], ['Chusky Cherlier', 'lunes'], ['Ju
stin Pymbeler', 'martes'], ['Kanye Pyst', 'sabado'], ['Javiera Mena', 'lunes'], ['Clepher F
olbünt', 'miercoles'], ['James Hopskins', 'jueves'], ['Pedro Machucha', 'viernes'], ['Nicol
as Salas', 'jueves'], ['Jose Cego', 'martes']]
```

```
>>> dias_disponibles
['lunes', 'miercoles', 'jueves', 'sabado', 'domingo']
```

*Resultado:*

```
>>> Ayuda_Saul(Reservas_Saul,dias_disponibles)
[['lunes', 'Jhonny Kurt'], ['lunes', 'Chusky Cherlier'], ['lunes', 'Javiera Mena'], ['mierc
oles', 'Joseph Curry'], ['miercoles', 'Justin Pymbeler'], ['miercoles', 'Clepher Folbünt'],
['miercoles', 'Jose Cego'], ['jueves', 'Daniel Pydman'], ['jueves', 'James Hopskins'], ['ju
eves', 'Nicolas Salas'], ['sabado', 'Joseph Pyrdiola'], ['sabado', 'Kanye Pyst'], ['sabado'
, 'Pedro Machucha'], ['domingo', 'Vicent Van Gogh']]
```

1 0 1 1  0 1 1  0 1  1 0 1 1 0 0 1  1 0  1 1 0 1 1  0 1 1  0 1  1 1 0 1 1 0  1 1 0 1 1 1  1 1 0 1

# </ Ejercicio 3: solución

```
def Ayuda_Saul(lista_reservas,lista_dia):
    agenda_saul = []
    dias = ["lunes","martes","miercoles","jueves","viernes","sabado","domingo"]
    for contacto in lista_reservas:
        persona = contacto[0]
        dia = contacto[1]
        if dia in lista_dia:
            persona_agendada = [dia,persona]
            agenda_saul.append(persona_agendada)
        else:
            indice_dia = dias.index(dia)
            dia_siguiente = dias[indice_dia+1]
            persona_agendada = [dia_siguiente,persona]
            agenda_saul.append(persona_agendada)
    lista_ordenada = []
    for day in dias:
        for agendado in agenda_saul:
            dia = agendado[0]
            if day == dia:
                lista_ordenada.append(agendado)
    return lista_ordenada
```

1 0 1 1  0 1 1  0 1  1 0 1 1 0 0 1  1 0  1 1 0 1 1  0 1 1  0 1  1 1 0 1 1 0  1 1 0 1 1 1  1 1 0 1

# No hay tarea esta semana

1 0 1 1  0 1 1  0 1  1 0 1 1 0 0 1  1 0  1 1 0 1 1  0 1 1  0 1  1 1 0 1 1 0  1 1 0 1 1 1  1 1 0 1

# </ Ayudantía 7: Listas 1 />


**¿Dudas?**
rolando.rojass@usm.cl

Rolando Rojas

1 0 1 1  0 1 1  0 1  1 0 1 1 0 0 1  1 0  1 1 0 1 1  0 1 1  0 1  1 1 0 1 1 0  1 1 0 1 1 1  1 1 0 1