Autores: María Inmaculada Rodríguez García, María Gema Carrasco García, Javier González Enrique, Juan Jesús Ruiz Aguilar, Ignacio J. Turias Domínguez. Universidad de Cádiz

1.4. Estructuras de datos
1.4.1. Listas

Python. Listas (vectores):

En otros lenguajes se conocen como vectores. Las listas admiten cambios en sus elementos. También se pueden añadir nuevos elementos o eliminar elementos.

Instrucciones: Python Lists

- **En MATLAB las estructuras de datos** que se asemejan a las listas en Python son las celdas (*cells*) y las matrices (*arrays*). Sin embargo, hay algunas diferencias clave entre las listas de Python y estas estructuras de datos de MATLAB.

  Celdas (*Cells*):
  - Las celdas en MATLAB son estructuras de datos que pueden contener datos de diferentes tipos y tamaños.
  - Cada elemento de una celda puede ser una matriz, una cadena, o incluso otra celda, permitiendo una mayor flexibilidad.
  - Puedes acceder a los elementos de una celda utilizando llaves {}.

```
% Ejemplo de celdas en MATLAB
ciudades = {'Jerez', 'Puerto Real', 'Cádiz'};
contaminantes = {'CO', 'NO_{2}', 'SO_{2}', 'Ozono'};
datos = {ciudades, contaminantes};
```

Ejercicio 32_01. Declaración de una lista en Python

```
# declaración de una lista
lista1 = []
lista2 = list()
# lista con valores. El primer elemento está en la posición 0
lista3 = [1,2, True]
print("Lista 3")
print(lista3)
print("Elemento cero de la lista 3 es: %d" % lista3[0])
# otra forma de declarar una lista con valores
lista4 = list([4,9, False, 'valor 1'])
print("Los contenidos de la lista4 son: ")
print(lista4)
```

```
Lista 3
     [1, 2, True]
Elemento cero de la lista 3 es: 1
Los contenidos de la lista4 son:
    [4, 9, False, 'valor 1']
```

Ejercicio 32_02. Declaración de una lista en MATLAB

```
% Declaración de un vector vacío
lista1 = [];
% Declaración de un vector vacío de otra forma
```

```
lista2 = [];
% Declaración de un vector con valores
lista3 = [1, 2, true];
fprintf('Lista 3\n');
disp(lista3);
fprintf('Elemento cero de la lista 3 es: %d\n', lista3(1));  % En MATLAB, los índices
comienzan en 1
% Otra forma de declarar un vector con valores
% Crear una celda con diferentes tipos de datos
lista4 = {4, 9, false, 'valor 1'};
% Mostrar los contenidos de la celda
fprintf('Los contenidos de la lista4 son: \n');
disp(lista4);

Lista 3
     1     2     1
Elemento cero de la lista 3 es: 1
Los contenidos de la lista4 son:
    {[4]}    {[9]}    {[0]}    {'valor 1'}
```

Ejercicio 33_01. Modificar un elemento de una lista en Python

```
lista4[3] = "valor 2"
print(lista4)

[4, 9, False, 'valor 2']
```

Ejercicio 33_02. Modificar un elemento de una lista en MATLAB

```
% Modificar un elemento del vector
lista4{3} = 'valor 2';
fprintf('La lista4 después de la modificación es:\n');
disp(lista4);

La lista4 después de la modificación es:
    {[4]}    {[9]}    {'valor 2'}    {'valor 1'}
```

Ejercicio 34_01. Mostrar los contenidos de una lista en Python

```
# Mostrar los contenidos de una lista
print("Primera forma de mostrar el contenido de una lista: ")
for elemento in lista4:
      print(elemento)
print("Segunda forma de mostrar el contenido de una lista: ")
# Mostrar contenido de una lista forma 2 recorriendo los índices
# len nos devuelve el tamaño de la lista
for i in range(len(lista4)):
      print(lista4[i])

Primera forma de mostrar el contenido de una lista:
4
9
False
```

```
valor 1
Segunda forma de mostrar el contenido de una lista:
4
9
False
valor 1
```

Ejercicio 34_02. Mostrar los contenidos de una lista en MATLAB

```
% Mostrar los contenidos de un vector
fprintf('Primera forma de mostrar el contenido de una lista:\n');
for elemento = lista4
    disp(elemento);
end
fprintf('Segunda forma de mostrar el contenido de una lista:\n');
% Mostrar contenido de un vector forma 2 recorriendo los índices
% numel nos devuelve el tamaño del vector
for i = 1:numel(lista4) % o también for i = 1:length(lista4)
    disp(lista4{i});
end
```

```
Primera forma de mostrar el contenido de un vector:
    {[4]}
    {[9]}
    {'valor 2'}
    {'valor 1'}
Segunda forma de mostrar el contenido de un vector:
      4
      9
valor 2
valor 1
```

Ejercicio 35_01. Añadir elementos a una lista en Python

```
# Añadimos nuevo elemento en la lista4 al final
lista4.append(7)
#añadir un nuevo elemento en la lista en una posición concreta
lista4.insert(0, "nuevo") # se inserta en la posición 0
print(lista4)
```

```
['nuevo', 4, 9, False, 'valor 2', 7]
```

Ejercicio 35_02. Añadir elementos a una lista en MATLAB

```
% Añadir un nuevo elemento al final de la lista
lista4{end + 1} = 7;
% Añadir un nuevo elemento en una posición concreta (en este caso, al principio)
nuevoElemento = 'nuevo';
lista4 = [nuevoElemento, lista4];
fprintf('La lista 4 después de añadir elementos es:\n');
disp(lista4);
```

```
La lista 4 después de añadir elementos es:
    {'nuevo'}    {[4]}    {[9]}    {'valor 2'}    {'valor 1'}    {[7]}
```

Ejercicio 36_01. Eliminar elementos de una lista según su posición en Python

```
lista4.pop(1) # Eliminamos el elemento de la posición 1
print("Lista 4 con el elemento de la posición 1 eliminado: ")
print(lista4)

Lista 4 con el elemento de la posición 1 eliminado:
    ['nuevo', 9, False, 'valor 2', 7]
```

Ejercicio 36_02. Eliminar elementos de una lista según su posición en MATLAB

```
posicion = 1; % Posición del elemento a eliminar
lista4(posicion) = []; % Eliminar el elemento de la posición 1
% Mostrar la celda después de eliminar el elemento
fprintf('La lista4 con el elemento de la posición %d eliminado:\n', posicion);
disp(lista4);

La lista4 con el elemento de la posición 1 eliminado:
    {[4]}    {[9]}    {'valor 2'}    {'valor 1'}    {[7]}
```

Ejercicio 37_01. Eliminar elementos de una lista según su valor en Python

```
try:
       lista4.remove(9)
except:
   print("Ese elemento no está en la lista")
 print(lista4)

['nuevo', False, 'valor 2', 7]
```

Ejercicio 37_02. Eliminar elementos de una lista según su valor en MATLAB

```
% Valor que deseas eliminar
valor_a_eliminar = 9;
% Encontrar índices de elementos iguales al valor a eliminar
indices = find(cellfun(@(x) isequal(x, valor_a_eliminar), lista4));
% Verificar si se encontraron elementos para eliminar
if ~isempty(indices)
    % Encontrados, eliminar los elementos en los índices encontrados
    lista4(indices) = [];
else
    % No encontrados, mostrar mensaje
    disp('Ese elemento no está en la lista');
end
% Mostrar el cell array resultante
disp(lista4);
```

```
    {[4]}    {'valor 2'}    {'valor 1'}    {[7]}
```

1.4.2. Matrices

Python. Listas (matrices):

En este caso tenemos listas con filas y columnas.

Instrucciones: Matrices en Python

- **Matrices (*Arrays*):**
    - En MATLAB, las matrices son estructuras de datos bidimensionales que pueden contener números, cadenas de texto u otros tipos de datos.
    - Puedes acceder a los elementos de una matriz utilizando índices, y las operaciones entre matrices son operaciones de matriz estándar.
    - La longitud de una matriz en una dimensión dada es fija (cuadrada (nxn), rectangular (nxm),… etc) y debe especificarse al crearla.

```
% Ejemplo de matriz en MATLAB
A = [1, 2, 3; 4, 5, 6; 7, 8, 9]; % Definición de matriz 3x3
```

Ambas estructuras, tanto **matrices** (*arrays*) como **celdas** (*cells*) en MATLAB pueden cumplir funciones similares a las listas de Python, dependiendo de los requisitos específicos de tu código. Las celdas son más flexibles porque pueden contener diferentes tipos de datos y tamaños, mientras que las matrices están más orientadas a operaciones numéricas y requieren que las dimensiones sean especificadas.

Ejercicio 38_01. Definir una matriz en Python

```
# Definimos una matriz de dimensión 3x4
A = [[1,2,3,4],[5,6,7,8],[9,10,11,12]]
print("Mostramos la matriz A que acabamos de definir: ")
print(A)
# Recorremos la matriz A y mostramos los elementos
filas = len(A)
col = len(A[0])
for i in range(filas):
   for j in range(col):
       print(A[i][j])
```

```
Mostramos la matriz A que acabamos de definir:
[[1,2,3,4],[5,6,7,8],[9,10,11,12]]
     1
     2
     3
     4
     5
     6
     7
     8
```

```
     9
    10
    11
    12
```

Ejercicio 38_02. Definir una matriz en MATLAB

```
% Definimos una matriz de dimensiones 3x4
A = [1, 2, 3, 4; 5, 6, 7, 8; 9, 10, 11, 12];
% Mostramos la matriz A que acabamos de definir
disp('Mostramos la matriz A que acabamos de definir:');
disp(A);
% Recorremos la matriz A y mostramos los elementos uno a uno
[filas, col] = size(A);
for i = 1:filas
    for j = 1:col
        disp(A(i, j));
    end
end
```

```
Mostramos la matriz A que acabamos de definir:
     1     2     3     4
     5     6     7     8
     9    10    11    12

     1
     2
     3
     4
     5
     6
     7
     8
     9
    10
    11
    12
```

1.4.3. Diccionarios y *cell arrays*

Python. Diccionarios:

Los diccionarios son estructuras de datos en las que se mapean entradas con valores (claves-valor). No están ordenados. Las claves son inmutables y no pueden estar duplicadas en el diccionario. Podemos introducir listas como elementos de un diccionario.

Instrucciones: Diccionarios Python

- **MATLAB no es un lenguaje de programación que utilice diccionarios como en Python. En su lugar, puedes utilizar celdas o *cell arrays***

Autores: María Inmaculada Rodríguez García, María Gema Carrasco García, Javier González Enrique, Juan Jesús Ruiz Aguilar, Ignacio J. Turias Domínguez. Universidad de Cádiz

Ejercicio 39_01. Declaración de diccionarios en Python

```
# En los diccionarios se mapean entradas con valores (clases-valor).
# Declaración de diccionarios:
# Declaramos un diccionario vacío:
diccionario1 = {}
# Definimos un diccionario con datos de clave - valor
miDiccionario = {"a":1,"b":2,"c":3}
```

Ejercicio 39_02. Declaración de *cell arrays* en MATLAB

```
% En MATLAB, se utilizan las estructuras para realizar tareas similares a los
diccionarios en Python.
% Declaramos una estructura vacía:
estructura1 = struct();
% Definimos una estructura con datos de campo - valor
miEstructura.a = 1;
miEstructura.b = 2;
miEstructura.c = 3;
```

Ejercicio 40_01. Acceso a los datos de un diccionario en Python

```
# Acceso a los datos de un diccionario mediante la clave:
print("Valor de a: %d" %miDiccionario["a"])

Valor de a: 1
```

Ejercicio 40_02. Acceso a los datos de un *cell array* en MATLAB

```
% Acceso a los datos de una estructura mediante el campo:
fprintf('Valor de a: %d\n', miEstructura.a);

Valor de a: 1
```

Ejercicio 41_01. Modificar un valor de un elemento del diccionario en Python

```
miDiccionario["a"] = 5
print("Valor de a tras el cambio: %d" % miDiccionario["a"])

Valor de a tras el cambio: 5
```

Ejercicio 41_02. Modificar un valor de un elemento del *cell array* en MATLAB

```
% Modificar un valor de un campo de la estructura:
miEstructura.a = 5;
fprintf('Valor de a tras el cambio: %d\n', miEstructura.a);

Valor de a tras el cambio: 5
```

Ejercicio 42_01. Acceso a un diccionario en Python

```
# Podemos acceder mediante el método get:
print(miDiccionario.get("b")) # muestra el valor para la clave b
```

```
2
```

Ejercicio 42_02. Acceso a un *cell array* en MATLAB

```
% Podemos acceder a un campo utilizando un enfoque similar a través del operador de
punto:
fprintf('%d\n', miEstructura.b);  % muestra el valor para el campo b

2
```

Ejercicio 43_01. Número de elementos del diccionario en Python

```
# Número de elementos de un diccionario:
print(len(miDiccionario))

3
```

Ejercicio 43_02. Número de elementos del *cell array* en MATLAB

```
% Número de elementos de una estructura:
fprintf('Número de elementos de una estructura: %d\n',
length(fieldnames(miEstructura)));

Número de elementos de una estructura: 3
```

Ejercicio 44_01. Mostrar contenido del diccionario en Python

```
# Mostrar el contenido de un diccionario:
# Importante %s para strings
print("Imprimimos todas las claves: %s" %miDiccionario.keys())
print("Imprimimos todas las claves: %s" %miDiccionario.values())
print("Imprimimos las parejas clave-valor del diccionario: ")
for i in miDiccionario:
   print(i, "-", miDiccionario[i])

Imprimimos todas las claves: dict_keys(['a', 'b', 'c'])
Imprimimos todos los valores: dict_values([5, 2, 3])
Imprimimos las parejas clave-valor del diccionario:
a - 5
b - 2
c - 3
```

Ejercicio 44_02. Mostrar contenido del *cell array* en MATLAB

```
% Mostrar el contenido de una estructura:
% Importante usar %s para strings
fprintf('Imprimimos todas las claves: %s\n', strjoin(fieldnames(miEstructura),', '));
values = struct2cell(miEstructura);
fprintf('Imprimimos todos los valores: %s\n', strjoin(string(values),', '));
fprintf('Imprimimos las parejas clave-valor de la estructura: \n');
fields = fieldnames(miEstructura);
for i = 1:length(fields)
```

```
    fprintf('%s - %d\n', fields{i}, miEstructura.(fields{i}));
end

Imprimimos todas las claves: a, b, c
Imprimimos todos los valores: 5, 2, 3
Imprimimos las parejas clave-valor de la estructura:
a - 5
b - 2
c - 3
```

Ejercicio 45_01. Borrar un elemento del diccionario en Python

```
# Borra un elemento por su clave:
del miDiccionario["a"]
print("Imprimimos todas las claves: %s" %miDiccionario.keys())
print("Imprimimos todos los valores: %s" %miDiccionario.values())

Imprimimos todas las claves: dict_keys(['b', 'c')]
Imprimimos todos los valores: dict_values[(2, 3)]
```

Ejercicio 45_02. Borrar un elemento del *cell array* en MATLAB

```
% Eliminar un campo por su nombre:
miEstructura = rmfield(miEstructura, 'a');
fprintf('Imprimimos todas las claves: %s\n', strjoin(fieldnames(miEstructura),', '));
values = struct2cell(miEstructura);
fprintf('Imprimimos todos los valores: %s\n', strjoin(string(values), ', '));

Imprimimos todas las claves: b, c
Imprimimos todos los valores: 2, 3
```

1.4.4. Diccionarios con listas como elementos

Ejercicio 46_01. Incluir listas como elementos de un diccionario en Python

```
# Definimos las listas
cuidades = ["Jerez", "Puerto Real", "Cádiz"]
conta =  ["CO", "NO2", "SO2", "Ozono"]
# Definimos el diccionario
datos = {"estaciones": ciudades, "contaminantes": conta}
print(datos["estaciones"][0]) # mostramos la primera estación
print(datos["contaminantes"][3]) # mostramos el cuarto contaminante

Jerez
Ozono

# Mostramos todos los elementos de la lista asociados a una clave:
datos["estaciones"]

['Jerez', 'Puerto Real', 'Cádiz']

datos["estaciones"][0]) = "Algeciras" # mostramos la primera estación
```

```
datos["estaciones "]

['Algeciras', 'Puerto Real', 'Cádiz']
```

Ejercicio 46_02. Incluir listas como elementos de un *cell array* en MATLAB

```
% Podemos incluir celdas como elementos de una estructura en MATLAB:
% Definimos las celdas
ciudades = {'Jerez', 'Puerto Real', 'Cádiz'};
contaminantes = {'CO', 'NO_{2}', 'SO_{2}', 'Ozono'};
% Definimos la estructura
datos = {ciudades, contaminantes}
% Mostramos la primera estación y el cuarto contaminante
disp(datos{1}{1});
disp(datos{2}{4});

Jerez
Ozono

% Mostramos todos los elementos de la lista asociados a una clave
datos{:} % muestra todos a la vez

datos{1}
  1×3 cell array
   {'Jerez'}    {'Puerto Real'}    {'Cádiz'}
datos{2}
  1×4 cell array
    {'CO'}    {'NO_{2}'}    {'SO_{2}'}    {'Ozono'}

% Mostramos la primera estación
fprintf('%s\n', datos{1}{1});
% Mostramos el cuarto contaminante
fprintf('%s\n', datos{2}{4});
% Mostramos todos los elementos de la celda asociados a una clave:
disp(datos.estaciones);
```

1.4.5. Tuplas o vectores

Python. Tuplas:

Las tuplas son colecciones de elementos que son de sólo lectura (listas o vectores inmutables).

Instrucciones: Tuplas Python

- **En Matlab las tuplas equivalen a vectores o matrices de una fila.**

Ejercicio 47_01. Definir una tupla en Python

```
# Las Tuplas son vectores inmutables:
a = (1,2,3)
print("El contenido de la tupla es: ")
```

Autores: María Inmaculada Rodríguez García, María Gema Carrasco García, Javier González Enrique, Juan Jesús Ruiz Aguilar, Ignacio J. Turias Domínguez. Universidad de Cádiz

```
print(a)
print("Recorremos la tupla elemento a elemento: ")
for i in range(len(a)):
  print(a[i], sep="\n")

El contenido de la tupla es:
     1     2     3
Recorremos la tupla elemento a elemento:
     1
     2
     3
```

Ejercicio 47_02. Definir un tupla como un vector en MATLAB

```
% Las tuplas no son directamente comparables a las estructuras de MATLAB.
% Sin embargo, podrían simularse con matrices de una fila, es decir, vectores.
a = [1, 2, 3];
disp('El contenido del vector es: ');
disp(a);
disp('Recorremos el vector elemento a elemento: ');
for i = 1:length(a)
    disp(a(i));
end

El contenido del vector es:
     1     2     3
Recorremos el vector elemento a elemento:
     1
     2
     3
```