Colecciones. Listas © IMF Smart Education

Índice

Colecciones. Listas

I. Introducción

En esta unidad se va a trabajar con una de las colecciones más importantes en Python, las listas. Primero se explicará cómo se crean las listas, así como su manejo y sus características más relevantes. Dentro de su manejo, veremos cómo acceder a los valores y cómo modificarlos sin usar ninguna función o método específico.

Para comprender mejor qué se puede hacer sobre listas, se verá un concepto que se puede usar siempre que queramos trabajar con partes de la lista. Ese concepto es el "particionado" o "slice". Se verán distintos ejemplos, tanto con índices positivos como negativos.

Por último, se verán un conjunto de métodos y funciones para facilitar el trabajo sobre las listas, como append, pop, insert, etc.

II. Objetivos

1

Conocer las características y la definición de listas.

2

Dominar la creación de listas.

3
Manejar las listas sin funciones ni métodos específicos.
4
Realizar particionado con índices positivos.
5
Realizar particionado con índices negativos.
6
Conocer el manejo de las funciones sobre listas.
7
Conocer el manejo de los métodos sobre listas.

III. Colecciones. Listas

Colecciones. Listas.

Puedes ver las diapositivas relacionas con este vídeo en el siguiente enlace: ./Colecciones_Listas.pdf.

IV. Resumen

En esta unidad se han aprendido las características y la creación de listas. Para ello, se ha practicado usando una lista como ejemplo y realizando sobre ellas distintas operaciones de slice, tanto con índices positivos como negativos

Para poder afianzar aún más el manejo de las listas, se ha practicado con una lista inicial, realizando operaciones con las funciones y los métodos vistos en clase. Se ha añadido el uso de la función y métodos de listas como conceptos FOR e IF de módulos anteriores.

Ejercicios

Caso práctico 1

Se pide

Lee una cadena de texto del usuario y para cada letra indica si es una vocal o una consonante.

Solución

```
cadena = input("Introduce una cadena: ")
vocales = ["a", "e", "i", "o", "u"]

for i in cadena:
    if i in vocales:
        print("el valor %s una vocal"%(i))
    else:
        print("el valor %s es una consonante"%(i))
```

```
Introduce una cadena: hola
el valor h es una consonante
el valor o una vocal
el valor l es una consonante
el valor a una vocal
```

Caso práctico 2

Se pide

Si tenemos una lista con nombres de ciudades, que muestre las ciudades que empiecen con la letra *m*.

Solución

```
lista=["Madrid","Malaga","Barcelona"]

for  i in lista:
    if i.lower().startswith('m'):
        print(i)
```

```
Madrid
Malaga
```

Caso práctico 3

Se pide

Escriba un programa de Python para imprimir los números de una lista especificada después de y que cree una nueva solo con los números pares.

Solución

```
1  lista=[2,3,5,6,7,10]
2  pares=[]
3
4  for i in lista:
5      if i %2==0:
6          pares.append(i)
7  print("lista original",lista)
8  print("lista de numeros pares",pares)
9
```

```
lista original [2, 3, 5, 6, 7, 10]
lista de numeros pares [2, 6, 10]
```

Caso práctico 4

Se pide

Ejercicio 1

Si se tiene la siguiente lista: lista["SGE",3.0,["A","B"],10,"FINAL"], indicar qué devolverá cada uno de los siguientes casos:

a. print (type(lista[2][1])) d.print (lista[0:4])
b. print (lista[2][1]) e.print (lista[1][1])
c. print (lista[0:3:1]) f. lista[2][2]="z"

Ejercicio 2

Si se tiene la siguiente lista: lista["A","B","C",1,2,["LUNES","MARTES"]], indicar qué devolverá cada uno de los siguientes casos:

a. Añadir la letra z b. Indicar la longitud de la lista c.print (lista[::4])
b. print (lista[0:3:2]) e. print (lista[-1][0]) f.print (lista[2::1])

Ejercicio 3

Basándose en la siguiente lista, a = [66.25, 333, 333, 1, 1234.5]:

a. Cuántas veces se repite 333 y 66.25 y 'x'.
b. Insertar en la posición 2 un -1 a.insert(2, -1).
c. Añade al final un nuevo 333 a.append(333).
d. Ordena la lista.

Solución

Caso práctico: Colecciones. Listas.