Ejercicios_sentencias_control

July 1, 2025

1 SENTENCIAS DE CONTROL - 30-06-2025

Cristopher Santander

2 1. EJERCICIOS

1. Escribe un programa que calcule la suma de todos los elementos de una *lista* dada. La lista sólo puede contener elementos numéricos.
2. Dada una lista con elementos duplicados, escribir un programa que muestre una nueva lista con el mismo contenido que la primera pero sin elementos duplicados. Para este ejercicio, no puedes hacer uso de objetos de tipo 'Set'.
3. Escribe un programa que construya un diccionario que contenga un número (entre 1 y n) de elementos de esta forma: (x, x*x). Ejemplo: para n = 5, el diccionario resultante sería {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
4. Escribe un programa que, dada una lista de palabras, compruebe si alguna empieza por 'a' y tiene más de 9 caracteres. Si dicha palabra existe, el programa deberá terminar en el momento exacto de encontrarla. El programa también debe mostrar un mensaje apropiado por pantalla que indique el éxito o el fracaso de la búsqueda. En caso de éxito, también se mostrará por pantalla la palabra encontrada.
5. Dada una lista L de números positivos, escribir un programa que muestre otra lista (ordenada) que contenga todo índice i que cumpla la siguiente condición: *L[i]* es múltiplo de 3. Por

ejemplo, dada la lista L = [3,5,13,12,1,9] el programa mostrará la lista [0,3,5] dado que *L[0], L[3] y L[5]* son, respectivamente, 3, 12 y 9, que son los únicos múltiplos de 3 que hay en L.

6. Dado un diccionario cuyos elementos son pares de tipo string y numérico (es decir, las claves son de tipo 'str' y los valores son de tipo 'int' o 'float'), escribe un programa que muestre por pantalla la clave cuyo valor asociado representa el valor númerico más alto de todo el diccionario. Por ejemplo, para el diccionario {'a': 4.3, 'b': 1, 'c': 7.8, 'd': -5} la respuesta sería 'c', dado que 7.8 es el valor más alto de los números 4.3, 1, 7.8 y -5.

7. Dada la lista a = [2, 4, 6, 8] y la lista b = [7, 11, 15, 22], escribe un programa que itere las listas a y b y multiplique cada elemento de a que sea mayor que 5 por cada elemento de b que sea menor que 14. El programa debe mostrar los resultados por pantalla.

8. Escribir un programa que pida un valor numérico X al usuario. Para ello podéis hacer uso de la función predefinida 'input'. El programa deberá mostrar por pantalla el resultado de la división 10/X. En caso de que el usuario introduzca valores no apropiados, el programa deberá gestionar correctamente las excepciones, por ejemplo, mostrando mensajes informativos por pantalla.

9. Escribir un programa que cree un *diccionario* cualquiera. Posteriormente, el programa pedirá al usuario (a través de la función predefinida 'input') que introduzca una clave del diccionario. Si la clave introducida es correcta (es decir, existe en el diccionario), el programa mostrará por pantalla el valor asociado a dicha clave. En caso de que la clave no exista, el programa gestionará de manera apropiada el error, por ejemplo, mostrando un mensaje informativo al usuario.

10. Escribe una *list comprehension* que construya una lista con los números *enteros* positivos de una lista de números dada. La lista original puede incluir números de tipo *float*, los cuales deben ser descartados.

11. Escribe una *set comprehension* que, dada una palabra, construya un conjunto que contenga las vocales de dicha palabra.

12. Escribe una *list comprehension* que construya una lista con todos los números del 0 al 50 que contengan el dígito 3. El resultado será: [3, 13, 23, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 43].

13. Escribe una *dictionary comprehension* que construya un diccionario que incluya los tamaños de cada palabra en una frase dada. Ejemplo: el resultado para la frase "Soy un ser humano" será {'Soy': 3, 'un': 2, 'ser': 3, 'humano': 6}

14. Escribe una *list comprehension* que construya una lista que incluya todos los números del 1 al 10 en orden. La primera mitad se mostrarán en formato numérico; la segunda mitad en texto. Es decir, el resultado será: [1, 2, 3, 4, 5, 'seis', 'siete', 'ocho', 'nueve', 'diez'].

[6]:
```python
#Ejercicio 1
# creo la lista
lista = [3,5,7,10,5,6,7,8]
# Variable acumuladora para guardar la suma
sumacont = 0
 # creamos el for el cual va a recorrer la lista y lo va a guardar en sumacont
for numero in lista:
    sumacont += numero     # Sumamos cada número a la variable 'sumacont'
```

```
print("La suma es:", sumacont)
```

```
La suma es: 51
```

[7]:
```
#Ejercicio 2
lista = [1,1,1,2,3,5,6,6,8,9]
#Lista en la que voy a guardar los elementos
lista_sin_duplicados = []
# creamos el for, y se guardaran temporalmente los elementos
# en num y si no esta el elemento en lista_sin_duplicados con el
# .append agrego el elemento al final de la lista
for num in lista:
    if num not in lista_sin_duplicados:
        lista_sin_duplicados.append(num)
print(lista_sin_duplicados)
```

```
[1, 2, 3, 5, 6, 8, 9]
```

[11]:
```
#Ejercicio 3
#Escribe un programa que construya un diccionario que contenga un número (entre␣
 ↪1 y *n*)
#de elementos de esta forma: (x, x*x).
#Ejemplo: para n = 5, el diccionario resultante sería {1: 1, 2: 4, 3: 9, 4: 16,␣
 ↪5: 25}

# n la cantidad de elementos
n = 8
#diccionario vacio en el que se guardara n y su potencia
potencia={}

# creamos la interacion en X - empezando por 1 y N+1 porque el range no coge el␣
 ↪ultimo elemento
for x in range(1,n+1):
    #creamos la sentencia que potencia en el indice x valdra x*x en cada paso
    potencia[x]=x*x
print(potencia)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}
```

[36]:
```
#Ejercicio 4
#Escribe un programa que, dada una lista de palabras, compruebe si alguna␣
 ↪empieza por 'a'
#y tiene más de 9 caracteres. Si dicha palabra existe,
#el programa deberá terminar en el momento exacto de encontrarla.
#El programa también debe mostrar un mensaje apropiado por pantalla que indique␣
 ↪el éxito o el fracaso de la búsqueda.
#En caso de éxito, también se mostrará por pantalla la palabra encontrada.
```

```
lista=["legue of legends", "valorant", "clash royale", "counter strike",␣
 ↪"aventuradamente"]
#esto es como una bandera que inicializo en False porque todavia no encuentra␣
 ↪nada
comprobador = False
for palabra in lista:
    if palabra.startswith("a") and len(palabra)>9:
        print("Palabra encontrada:",palabra)
        comprobador = True
        break
    if not comprobador:
        print("Palabra no encontrada")
```

```
Palabra no encontrada
Palabra no encontrada
Palabra no encontrada
Palabra no encontrada
Palabra encontrada: aventuradamente
```

[45]:
```
#Ejercicio 5
#Dada una lista *L* de números positivos, escribir un programa que muestre otra␣
 ↪lista (ordenada)
#que contenga todo índice *i* que cumpla la siguiente condición: *L[i]* es␣
 ↪múltiplo de 3.
#Por ejemplo, dada la lista *L* = [3,5,13,12,1,9] el programa mostrará la lista␣
 ↪[0,3,5] dado que *L[0], L[3] y L[5]* son,
#respectivamente, 3, 12 y 9,
#que son los únicos múltiplos de 3 que hay en *L*.

L=[3,5,7,12,9,32,0]
#creo otra lista en la que guardare los indices que cumplan la sentencia
indices=[]

for i in range(len(L)):
    if L[i]!=0 and L[i] %3==0: #creamos la sentencia que dice que exluya 0 y␣
 ↪que el resultado de i%3 sea igual a 0
                               #entonces agregamos a la lista vacia el
        indices.append(i)
print(indices)
```

```
[0, 3, 4]
```

[4]:
```
#Ejercicio 6
#Creo el diccionario con las claves y sus valores asignados de tipo int y tipo␣
 ↪float
```

```
diccionario={'a':5,'b':89,'c':3.5,'d':-10}
#creo una variable que guardara el numero mayor, y le digo con la funcion max␣
 ↪que encuentra el valor mas alto que recorra diccionario, pero que compare el␣
 ↪contenido de cada clave con '.get', para que no compare directamente las␣
 ↪claves
nummayor=max(diccionario,key=diccionario.get) # estructura del 'max'␣
 ↪max(iterable, key=funcion)
print("El numero mayor es", nummayor)
```

```
El numero mayor es b
```

[5]:
```
#Ejercicio 7
#
lista1=[2,4,5,6]
lista2=[15,12,6,8]
```