Status	Finished
Started	Saturday, 23 November 2024, 5:14 PM
Completed	Saturday, 23 November 2024, 5:16 PM
Duration	2 mins 17 secs
Marks	2.00/2.00
Grade	**10.00** out of 10.00 (**100**%)

Information

Procesando varias listas al mismo tiempo

En muchas ocasiones se necesita procesar dos o más listas simultáneamente. El siguiente ejemplo intercala los elementos de dos listas, generando una nueva:

```
def interleave(list1, list2):
    result = []

    for i in range(len(list1)):
        result.append(list1[i])
        result.append(list2[i])

    return result

numbers = [10, 20, 30]
chars= ['A', 'B', 'C']
print(interleave(numbers, chars)) # [10, 'A', 20, 'B', 30, 'C']
```

Como se observa en el ejemplo, no hace falta un bucle independiente para cada lista, ya que de lo que se trata es de recorrerlas al mismo tiempo, tomando un elemento de cada una para pasarlo al resultado, esto es, 1 recorrido (de 1, 2, o más listas a la vez) == 1 bucle.

En la función del ejemplo se supone que ambas listas son del mismo tamaño, o que *len(list2) >= len(list1)*. De no ser así, se produciría un error (*Index out of range*).

Question 1

Complete

Mark 1.00 out of 1.00

Dadas las siguientes listas:

```
list1 = ['red', 'blue', 'yellow']
list2 = ['X', 'Y', 'Z']
```

¿Qué muestra cada uno de los siguientes trozos de código?

```
result = []

for i in range(len(list1)):
    result.append(list1[i])
    result.append(list2[i])

print(result)
```

['red', 'X', 'blue', 'Y', 'yellow', 'Z']

```
result = []

for i in range(len(list1)):
    result.append(list1[i])
    for j in range(len(list2)):
        result.append((list2[i]))

print(result)
```

['red', 'X', 'X', 'X', 'blue', 'Y', 'Y', 'Y', 'yellow', 'Z', 'Z', 'Z']

```
result = []

for i in range(len(list1)):
    result.append(list1[i])
for i in range(len(list2)):
    result.append(list2[i])

print(result)
```

['red', 'blue', 'yellow', 'X', 'Y', 'Z']

Information

Procesando varias listas al mismo tiempo con zip

Si no necesitamos tener acceso a los índices, una forma de procesar dos o más listas al mismo tiempo es usar la función *zip()*:

```
def interleave1(list1, list2):
    result = []

    for item1, item2 in zip(list1, list2):
        result.append(item1)
        result.append(item2)

    return result

list1 = [10, 20, 30]
list2 = ['A', 'B', 'C']
print(interleave1(list1, list2)) # [10, 'A', 20, 'B', 30, 'C']
```

La función predefinida *zip()* permite iterar por una secuencia de tuplas, formada de tal manera que el *i*-ésimo elemento de esa secuencia es una tupla formada por los i-ésimos elementos de cada una de las listas (en general, secuencias) que se le pasan como parámetros. En el ejemplo, el primer elemento devuelto por *zip* es una tupla formada por el primer elemento de *list1* y el primer elemento de *list2 (10, 'A')* como segundo elemento *(20, 'B')* y como tercer y último elemento *(30, 'C')*. Note que en cada iteración del bucle, cada elemento/tupla de la secuencia zip es disgregado en las variables item1 e item2. Para la tupla *(10, 'A')* *item1* tomará el valor *10* e *item2* el valor *'A'*.

En el caso de que las listas fuesen de diferentes longitudes, *zip()* procesaría sólo hasta la longitud de la más corta.

Information

Procesando simultáneamente listas de distintos tamaños con zip_longest

La función *zip()* permite recorrer simultáneamente dos o más listas, pero sólo hasta el tamaño de la más corta. Si se quiere recorrer hasta el tamaño de la más larga, se puede usar la función *zip_longest()* del módulo *itertools*:

```
import itertools
def interleave2(list1, list2):
    result = []

    for item1, item2 in itertools.zip_longest(list1, list2, fillvalue='-'):
        result.append(item1)
        result.append(item2)

    return result

list1 = [10, 20, 30]
list2 = ['A', 'B', 'C', 'D']
print(interleave2(list1, list2)) # [10, 'A', 20, 'B', 30, 'C', '-', 'D']
```

Los elementos que faltan en la lista más cortas se sustituyen con el valor especificado en el parámetro *fillvalue* (el caracter '-' en el ejemplo). Si se omite el parámetro *fillvalue*, se usa el valor *None*.

Information

Procesando varias listas de diferentes tamaños a la vez usando índices

Si se quiere recorrer dos listas de diferentes tamaños a la vez usando índices en vez de *zip_longest()*, es conveniente, como en el siguiente ejemplo, separar la iteración en tres bucles de los que sólo se ejecutan, como máximo, dos: el primero, mientras las dos listas tienen elementos, y uno de los otros dos, para terminar de procesar los elementos de la lista más larga.

```
def interleave3(list1, list2):
    result = []
    min_len = min(len(list1), len(list2))

    for i in range(min_len):
        result.append(list1[i])
        result.append(list2[i])
```

```
    for i in range(min_len, len(list1)):
        result.append(list1[i])
        result.append('-')
```

```
    for i in range(min_len, len(list2)):
        result.append('-')
        result.append(list2[i])
```

```
    return result
```

Alternativamente, se pueden usar sentencias *while* en su lugar, aprovechando el valor del índice a la salida del primer bucle:

```
def interleave3b(list1, list2):
    result = []
    i = 0

    while i < min(len(list1), len(list2)):
        result.append(list1[i])
        result.append(list2[i])
        i += 1

    while i < len(list1):
        result.append(list1[i])
        result.append('-')
        i += 1

    while i < len(list2):
        result.append('-')
        result.append(list2[i])
        i += 1

    return result
```

De hacer un solo bucle, hasta la longitud más larga, habría que incluir dentro del bucle preguntas para saber si se ha acabado alguna de las listas, cuyo coste de ejecución se sumaría en cada ejecución, pero sería fácilmente extensible a cualquier número de listas:

```
def interleave3c(list1, list2):
    result = []

    for i in range(max(len(list1), len(list2))):
        if i < len(list1):
            result.append(list1[i])
        else:
            result.append('-')
        if i < len(list2):
            result.append(list2[i])
        else:
            result.append('-')

    return result
```

Una alternativa a esto, en Python, es usar *zip_longest()* junto con *enumerate()*, para dos o más listas:

```
from itertools import zip_longest

def interleave3d(list1, list2):
    result = []

    for i, items in enumerate(zip_longest(list1, list2, fillvalue='-')):
        result.append(items[0])
        result.append(items[1])

    return result
```

Question 2

Complete

Mark 1.00 out of 1.00

Dadas las siguientes listas:

```
num = [1, 2, 3]
color = ['red', 'white', 'black']
value = [255, 256]
```

¿Qué muestra cada uno de los siguientes trozos de código?

```
from itertools import zip_longest
for (a, b, c) in zip_longest(num, color, value):
    print(a, b, c, end = ' ')
```

1 red 255 2 white 256 3 black None

```
for (a, b, c) in zip(num, color, value):
    print(a, b, c, end = ' ')
```

1 red 255 2 white 256