Status	Finished
Started	Tuesday, 22 October 2024, 1:42 PM
Completed	Tuesday, 22 October 2024, 1:43 PM
Duration	57 secs
Marks	5.00/6.00
Grade	**8.33** out of 10.00 (**83.33**%)

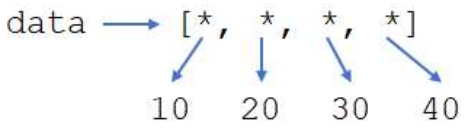
Information

Las listas son mutables

Considere la siguiente instrucción:

```
data = [10, 20, 30, 40]
```

Esta instrucción crea una lista referenciada por la variable *data*.

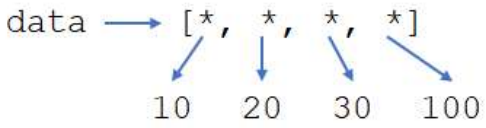


La lista consta de 4 variables: data[0], data[1], data[2] y data[3] que hacen referencia a los cuatro valores *int*.

Las listas son estructuras mutables, esto significa que pueden ser modificadas.

```
data[3] = 100
```

La instrucción anterior cambia el último elemento de la lista. No hemos creado una nueva lista, solo hemos cambiado uno de sus elementos, de ahí que digamos que sea mutable.



Nótese que esto no se puede hacer con una tupla o con una string porque, a diferencia de las listas, ambas son estructuras de datos inmutables.

Question 1

Complete

Mark 1.00 out of 1.00

Observe el siguiente código:

```
a = [2, 0, 0, 8]
a[3] = 9
a[2] += 1
a = a * 2
print(a)
```

¿Qué valor se muestra?

Select one:

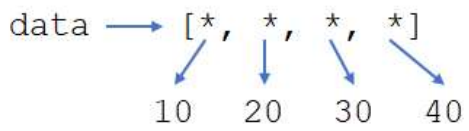
- [4,0,2,18]
- [[2, 0, 1, 9], [2, 0, 1, 9]]
- [2, 0, 1, 9, 2, 0, 1, 9]

Information

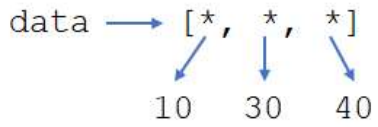
Eliminación de elementos

Para eliminar un elemento de una lista se puede usar la instrucción **del**.

```
data = [10, 20, 30, 40]
```




```
del data[1]
```



Question 2

Complete

Mark 0.00 out of 1.00

Observe el siguiente código y marque todas las opciones correctas:

```
import random

b = [random.randrange(1,10)] * 10

for i in range(5):
    del b[i]

print(b)
```

(Nota: la función *random.randrange* genera un número al azar en el rango especificado)

Select one or more:

- [x] El programa muestra una lista con 10 elementos
- [ ] Todos los elementos que quedan en la lista *b* tienen el mismo valor
- [ ] El programa muestra una lista con 5 elementos
- [x] la instrucción **del** no puede usarse como se hace en el programa

Information

Recorrido de una lista

Para recorrer una lista suele usarse un bucle *for*:

```
friends = ['Peter','Paul', 'Michael', 'George', 'John']
for i in range(len(friends)):
    print(i, "-", friends[i])
```

Resultado:

0 - Peter
1 - Paul
2 - Michael
3 - George
4 - John

Si sólo estamos interesados en los elementos y los índices no son relevantes, podemos usar una notación más concisa:

```
for element in friends:
    print(element, end=' ')
```

En este último ejemplo, además, hemos añadido a la instrucción *print* el parámetro *end*, sustituyendo el salto de línea que se usa por omisión por un espacio; el resultado es:

Peter Paul Michael George John

Question 3

Complete

Mark 1.00 out of 1.00

La lista *temperatures* contiene las temperaturas registradas en Las Palmas de Gran Canaria en la primera semana de julio del año pasado.

```
temperatures = [21.0, 22.0, 25.0, 25.0, 24.0, 21.0, 20.0]
```

Necesitamos calcular la temperatura media en esa semana. Compare estas dos soluciones:

SOLUCIÓN 1

```
sum = 0.0

for i in range(len(temperatures)):
    sum += temperatures[i]

avg_temp = sum / len(temperatures)
```

SOLUCIÓN 2

```
sum = 0.0

for temp in temperatures:
   sum += temp
```

```
avg_temp = sum / len(temperatures)
```

¿Ambas calculan el resultado correctamente?

Select one:

- True
- False

Question 4

Complete

Mark 1.00 out of 1.00

Queremos incrementar los valores de los elementos de la lista *t* en 1.

```
t = [21, 22, 25, 25, 24, 21, 20]
```

Considere las siguientes soluciones:

SOLUCIÓN 1

```
for x in t:
    x += 1

print(t)
```

SOLUCIÓN 2

```
for i in range(len(t)):
    t[i] += 1

print(t)
```

¿Las dos resuelven el problema correctamente?

Select one:

- True
- False

Information

Enumerate

Podemos hacer uso de la función **`enumerate()`** para acceder a ambos, el índice y el valor correspondiente al mismo tiempo:

```
friends = ['Peter', 'Paul', 'Michael', 'George', 'John']

for i, element in enumerate(friends):
   print(i, "-", element)
```

Resultado:

0 - Peter
1 - Paul
2 - Michael
3 - George
4 - John

La función *enumerate()* inicia el contador de la secuencia de posiciones en 0; podemos especificar el valor inicial usando el parámetro *start*:

```
friends = ['Peter', 'Paul', 'Michael', 'George', 'John']

for i, element in enumerate(friends, start = 1):
   print(i, "-", element)
```

Resultado:

1 - Peter
2 - Paul
3 - Michael
4 - George
5 - John

Question 5

Complete

Mark 1.00 out of 1.00

Escriba lo que muestra el siguiente código:

```
for i, element in enumerate(['a', 'b', 'c'], start = 1):
    print(i * element, end = ',')
```

¡Tenga cuidado y no olvide agregar el carácter de coma después de cada uno de los valores calculados!

(Recuerde el funcionamiento con strings del operador *)

Answer: a,bb,ccc,

Information

List comprehension

La *comprensión de listas* de Python es un mecanismo que proporciona una forma concisa de crear nuevas listas. Consiste en una expresión, seguida de una cláusula `for`, seguida a su vez de cero o más cláusulas `for` o `if`; todo ello entre corchetes. Por ejemplo:

```
[expression for element in list]
[expression for element in list if condition]
```

Con esta notación obtenemos una nueva lista, compuesta de los valores que resulten de evaluar repetidamente la expresión (que puede ser compleja) en el contexto de las cláusulas `for` e `if` que la siguen. Aquí hay algunos ejemplos; estudie atentamente los resultados:

```
list1 =  [i for i in range(10)]
#list1 vale: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
list2 = [i ** 2 for i in range(5)]
#list2 vale: [0, 1, 4, 9, 16]
```

none')?'inline':'none'})()" style="color:darkblue">Etimología

Information

List comprehension - más ejemplos

```
import random
a = [random.randint(0, 9) for i in range(10)]
print(a)
```

Salida: [4, 4, 9, 7, 4, 1, 0, 0, 3, 6]

(La función *random.randint()* genera un número al azar en el rango indicado; la salida será diferente en diferentes ejecuciones)

```
words = ['this', 'is', 'a', 'list', 'of', 'words']
b = [word[0].upper() for word in words]
print(b)
```

Salida: ['T', 'I', 'A', 'L', 'O', 'W']

(El método *upper()* devuelve la conversión a mayúsculas de la string asociada)

```
things = ['pen', 'pencil', 'rubber', 'paper']
c = [thing for thing in things if len(thing) > 5]
print(c)
```

Salida: ['pencil', 'rubber']

```
from math import pi
d = [str(round(pi, i)) for i in range(1, 6)]
print(d)
```

Salida: ['3.1', '3.14', '3.142', '3.1416', '3.14159']

(La función *round()* redondea el *float* pasado como primer parámetro al número de decimales indicado por el segundo)

Question 6

Complete

Mark 1.00 out of 1.00

Relacione cada expresión con la lista resultante (referenciada por la variable *a*).

`a = [0 for i in range(10)]`	[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
`a = [x * y for x in range(1, 4) for y in range(1, 4)]`	[1, 2, 3, 2, 4, 6, 3, 6, 9]
`years = [1945, 1969, 1971, 1980, 1984, 2012, 2015]` `a = [y % 100 for y in years]`	[45, 69, 71, 80, 84, 12, 15]