Status	Finished
Started	Tuesday, 22 October 2024, 1:39 PM
Completed	Tuesday, 22 October 2024, 1:41 PM
Duration	1 min 55 secs
Marks	4.75/5.00
Grade	9.50 out of 10.00 (95 %)

Information

Las **listas** son estructuras de datos diseñadas para almacenar secuencias de elementos.

Para crear una lista en Python, ponemos los elementos separados por comas y encerrados entre corchetes:

```
temperatures = [36.5, 36.7, 37.1, 37.1, 37.5, 38.5]
students = ['Ada Lovelace', 'Alan Turing', Edsger W. Dijkstra', 'Donald Knuth', 'Niklaus Wirth', 'Ole-Johan Dahl', 'Kristen
Nygaard']
years = [1945, 1969, 1971, 1978, 1980, 1984, 2012, 2015, 2019]
incoming_person = ['Susan', 'Slovakia', 2019, 52.18, True]
x = 17
y = 9
z = 1989
values = [x, y, z]
```

Como se ve en el ejemplo, en Python una lista puede tener todos los elementos del mismo tipo, o mezclar elementos de tipos diferentes.

Una lista vacía se puede crear de dos formas.

```
a = []
```

^

```
a = list()
```

La <u>función</u> type() devuelve el tipo list cuando se le pasa una lista:

```
>>> type(a)
<class 'list'>
```



Marque todas las <u>variable</u>s que <u>referencia</u>n listas.

Select one or more:

- c = ['spring', 'summer', 'autumn', 'winter']
- e = (0, 10, 'X', 120.5)
- f = []
- b = (1, 2, 3, 4, 5, 6, 7)
- a = [True]
- d = list()

Information

Para acceder a los elementos de una lista se usan <u>índice</u>s. El <u>índice</u> del primer elemento es 0. Igual que en las <u>string</u>s o las tuplas, se pueden usar <u>índice</u>s negativos que <u>referencian</u> desde el final de la lista (el <u>índice</u> negativo del último elemento es -1).

```
a = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
print(a[0])
print(a[-1])
print(a[-2])
print(a[-2])
```

Resultado de la ejecución del código anterior:

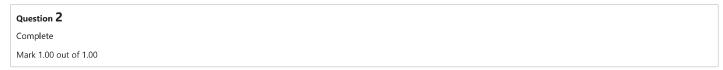
Monday

Wednesday

Sunday

Saturday

Sunday



Marque las sentencias que dan acceso al elemento 'red':

```
colors = ['red', 'green', 'blue', 'white', 'black']
```

Select one or more:

- colors[0]
- colors[1]
- colors[-5]
- colors[len(colors)-1]
- ['red', 'green', 'blue', 'white', 'black'][0]

Information

Concatenación de listas

Se pueden concatenar dos listas usando el operador +. Como resultado creamos una <u>nueva</u> lista conteniendo todos los elementos de la primera lista, seguidos de todos los elementos de la segunda lista, conservando siempre el orden original.

```
>>> a = [1, 2, 3, 4, 5, 6]
>>> b = [6, 7, 8]
>>> a + b
[1, 2, 3, 4, 5, 6, 6, 7, 8]
>>> students = ['John', 'Martin']
>>> students
= ['Joseph'] + students
>>> students
['Joseph', 'John', 'Martin']
>>> students + []
['Joseph', 'John', 'Martin']
```

Question 3

Complete

Mark 1.00 out of 1.00

¿Qué valor muestra el siguiente código?

```
list1 = [1] + [2, 3] + [4] + [5] + []
list2 = [1] + [5] + [2, 3, 4]
print(len(list1) == len(list2))
```

Answer: True

Information

El operador *

El operador * se usa para concatenar una misma lista repetidamente.

```
>>> languages = ['C', 'Java', 'Python'] + ['C', 'Java', 'Python'] + ['C', 'Java', 'Python']
>>> languages
['C', 'Java', 'Python', 'C', 'Java', 'Python', 'C', 'Java', 'Python']
>>> languages = ['C', 'Java', 'Python'] * 3
>>> languages
['C', 'Java', 'Python', 'C', 'Java', 'Python', 'C', 'Java', 'Python']
```

El operador * facilita la inicialización de una lista:

```
>>> counters = [0] * 10
>>> counters
[0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
Question 4
Complete
Mark 1.00 out of 1.00
```

¿Qué muestra el siguiente código?

```
greetings = 4 * ['Hallo']
greetings = greetings + ['Bye'] * 3
print(greetings[len(greetings) // 2])
```

```
Answer: Hallo
```

Information

Operadores in y not in

Podemos comprobar si un elemento determinado está o no contenido en una lista usando los operadores in y not in:

```
>>> languages = ['C', 'Java', 'Python']
>>> 'Pascal' in languages
False
>>> 'Python' in languages
True
>>> 'Java' not in languages
False
```

Question 5
Complete
Mark 1.00 out of 1.00

Observe el siguiente código y marque las opciones que son verdaderas:

```
words = ['We', 'can', 'do', 'it']
if ['can','it'] in words:
    print('yes')
else:
    print('no')
```

Select one or more:

- La <u>variable</u> words <u>referencia</u> a una lista que solo contiene <u>string</u>s
- El programa muestra no
- El programa muestra yes
- El operador in no puede usarse en una condición

Information

Anotación de variables lista

A partir de la versión 3.9 de Python se puede anotar usando el nombre real (en minúsculas) de tipos no básicos:

```
lista: list  # La <u>variable</u> lista se anota de tipo list
vector: list[float] # La <u>variable</u> vector se anota de tipo lista floats
```

En versiones anteriores es necesario usar un nombre del módulo typing:

```
from typing import List
lista: List
vector: List[float]
```

Recordemos que Python por sí mismo no impide que a una variable se les asignen valores de un tipo distinto del anotado.