Python List

- Python list is a data structure which is used to store various types of data.
- In Python, lists are **mutable** i.e., Python will not create a new list if we modify an element of the list.
- It works as a container that holds other objects in a given order. We can perform various operations like insertion and deletion on list.
- A list can be composed by storing a sequence of different type of values separated by commas.
- Python list is enclosed between square(**[]**) brackets and elements are stored in the index basis with starting index 0.
- It can have any number of items and they may be of different types (integer, float, string etc.).

Lists

- Ordered sequence of values
- Written as a sequence of comma-separated values between square brackets
- Values can be of different types
  - usually the items all have the same type

```
>>> lst = [1,2,3,4,5]
>>> lst
[1, 2, 3, 4, 5]
>>> type(lst)
<type 'list'>
```

Lists

- List is also a sequence type
  - Sequence operations are applicable

Lists

- List is also a sequence type
  - Sequence operations are applicable

```
>>> [0] + fib # Concatenation
```

Nested List

- Also, a list can even have another list as an item. This is called nested list.

- # nested list

```
my_list = ["mouse", [8, 4, 6], ['a']]
```

List Index

- We can use the index operator [] to access an item in a list. Index starts from 0. So, a list having 5 elements will have index from 0 to 4.

- Trying to access an element other than that this will raise an IndexError. The index must be an integer. We can't use float or other types, this will result into TypeError.

- Nested list are accessed using nested indexing.

Negative Indexing & Slice

- Negative indexing Python allows negative indexing for its sequences. The index of -1 refers to the last item, -2 to the second last item and so on.

- Slice List in Python
- We can access a range of items in a list by using the slicing operator (colon).

- List are mutable, meaning, their elements can be changed unlike string or tuple.
- We can use assignment operator (=) to change an item or a range of items.

append() and extend()

- We can add one item to a list using **append()** method or add several items using **extend()** method.
- We can also use **+ operator** to combine two lists. This is also called concatenation.
- The *** operator** repeats a list for the given number of times.
- We can delete one or more items from a list using the keyword del. It can even delete the list entirely.

remove(), pop() & clear()

- We can use remove() method to remove the given item or pop() method to remove an item at the given index.
- The pop() method removes and returns the last item if index is not provided. This helps us implement lists as stacks (LIFO data structure).
- We can also use the clear() method to empty a list.
- **List Membership Test** We can test if an item exists in a list or not, using the keyword ‘in/ not in’.

Append()	Add an element to the end of the list
Extend()	Add all elements of a list to another list
Insert()	Insert an item at the defined index
Remove()	Removes an item from the list
Pop()	Removes and returns an element at the given index
Clear()	Removes all items from the list
Index()	Returns the index of the first matched item
Count()	Returns the count of the number of items passed as an argument
Sort()	Sort items in a list in ascending order
Reverse()	Reverse the order of items in the list
copy()	Returns a copy of the list

reduce()	apply a particular function passed in its argument to all of the list elements stores the intermediate result and only returns the final summation value
sum()	Sums up the numbers in the list
ord()	Returns an integer representing the Unicode code point of the given Unicode character
cmp()	This function returns 1 if the first list is “greater” than the second list
max()	return maximum element of a given list
min()	return minimum element of a given list
all()	Returns true if all element is true or if the list is empty
any()	return true if any element of the list is true. False, if the list is empty.
len()	Returns length of the list or size of the list
enumerate()	Returns enumerate object of the list
accumulate()	apply a particular function passed in its argument to all of the list elements returns a list containing the intermediate results
filter()	tests if each element of a list is true or not
map()	returns a list of the results after applying the given function to each item of a given iterable
lambda()	This function can have any number of arguments but only one expression, which is evaluated and returned.

Loop Through a List

```
thislist =["apple", "banana", "cherry"]
for x in thislist:
  print(x)
```

Loop Through the Index Numbers

```
thislist = ["apple", "banana", "cherry"]
for i in range(len(thislist)):
  print(thislist[i])
```

Shorthand for loop

```
thislist = ["apple", "banana", "cherry"]
[print(x) for x in thislist]
```

```
>>> colors = ["red", "green", "burnt sienna", "blue"]
>>> colors[2]
```

What is the output of the colors[2] expression?

A. 'blue'

B. It causes a run-time error.

C. 'red'

D. 'green'

E. 'burnt sienna'

```
>>> colors = ["red", "green", "burnt sienna", "blue"]
>>> "yellow" in colors
```

What is the result of the yellow in colors expression?

A. 3

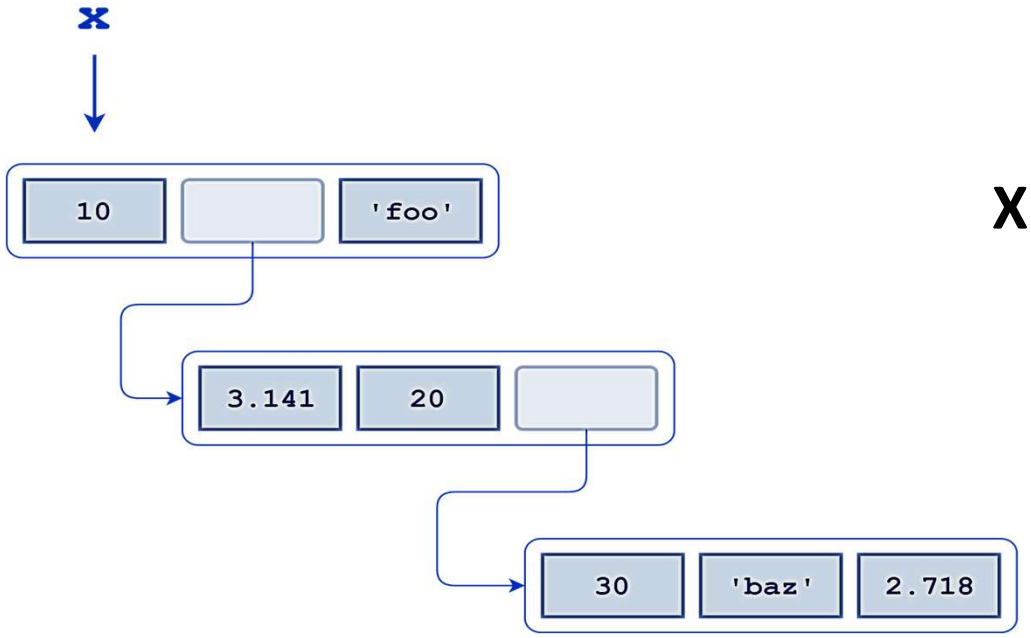
B. 4

C. False

D. ValueError: 'yellow' is not in list

Consider the following nested list definition:

x = [10, [3.141, 20, [30, 'baz', 2.718]], 'foo']
A schematic for this list is shown below:

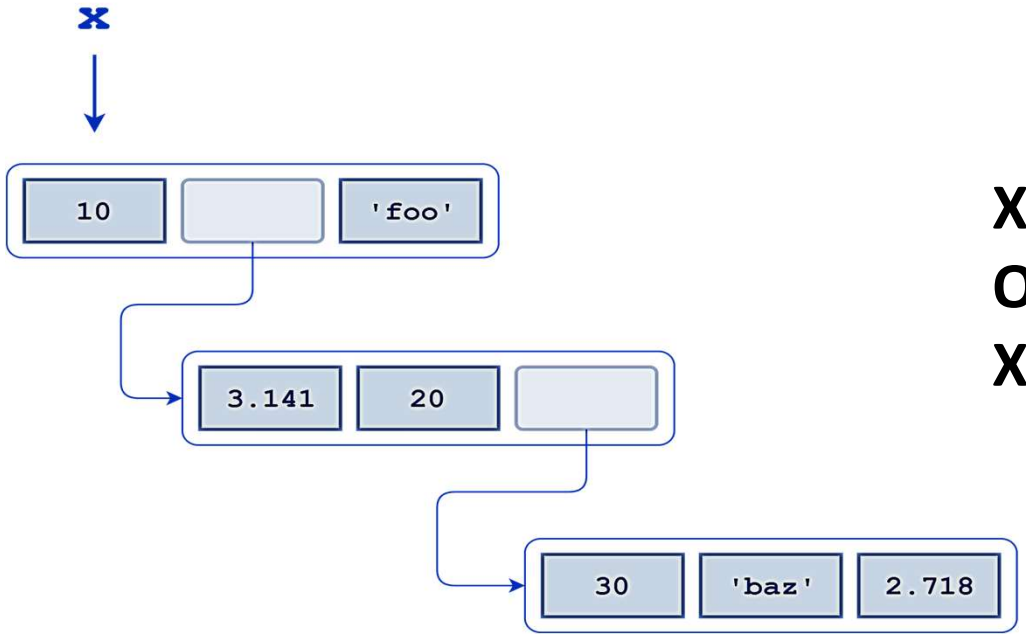


X[1][2][1][2]

Nest list example
What is the expression that returns the 'z' in 'baz'?

Consider the following nested list definition:

x = [10, [3.141, 20, [30, 'baz', 2.718]], 'foo']
A schematic for this list is shown below:



X[1][2][1:]
Or
X[1][2][1:3]

Nest list example
What is the expression that returns the list ['baz', 2.718]?

Which of the following are true of Python lists?

A. There is no conceptual limit to the size of a list

B. All elements in a list must be of the same type

C. A list may contain any type of object except another list

D. These represent the same list:
['a', 'b', 'c']
['c', 'a', 'b']

E. A given object may appear in a list more than once

You have a list A defined as follows:

A = [1, 2, 7, 8]
Write a Python statement using slice assignment that will fill in the missing values so that a equals [1, 2, 3, 4, 5, 6, 7, 8].

a[2:2] = [3, 4, 5, 6]