In the name of god

kian-nazari

1. Can a Python list hold a mixture of integers and strings

? *yes.*

2. Cant happens if you attempt to access an element of a list using a negative index?

negative indexes start from the end of the list. So -1 would be the last item in the list.

3. Cant Python statement produces a list containing the values 45,–3, 16 and 8, in that order?

[45, -3, 16, 8]

4. Given the statement

lst = [10, -4, 11, 29]

(a) What expression represents the very first element of lst?

lst[0]

(b) What expression represents the very last element of lst?

lst[-1] or lst[3]

(c) What is lst[0]? 10

(d) What is lst[3]? 29

(e) What is lst[1]? -4

(f) What is lst[-1]? -29

(g) What is lst[-4]? 10

(h) Is the expression lst[3.0] legal or illegal?

 it either should be a integer or a slice, so illegal.

5. Given the statements

lst = [3, 0, 1, 5, 2]

x = 2

evaluate the following expressions:

(a) lst[0]? 3

(b) lst[3]? 5

(c) lst[x]? 1

(d) lst[-x]? 5

(e) lst[x + 1]? 5

(f) lst[x] + 1? 2

(g) lst[lst[x]]? 0

(h) lst[lst[lst[x]]]? 3

6. What function returns the number of elements in a list
? *len*

7. What expression represents the empty list
? *[]*

8. Given the list
lst = [20, 1, -34, 40, -8, 60, 1, 3]
evaluate the following expressions:

(a) lst : *[20,1,-34,40,-8,60,1,3]*

(b) lst[0:3] : *[20,1,-34]*

(c) lst[4:8] : *[-8,60,1,3]*

(d) lst[4:33] : *[-8,60,1,3]*

(e) lst[-5:-3] : *[40,-8]*

(f) lst[-22:3] : *[20,1,-34]*

(g) lst[4:] : *[-8,60,1,3]*

(h) lst[:] : *[20,1,-34,40,-8,60,1,3]*

(i) lst[:4] : *[20,1,-34,40]*

(j) lst[1:5] : *[1,-34,40,-8]*

(k) -34 in lst : *True*

(l) -34 not in lst : *False*

(m) len(lst) : 8

9. An assignment statement containing the expression a[m:n] on the left side and a list on the right side can modify list a. Complete the following table by supplying the m and n values in the slice assignment statement needed to produce the indicated list from the given original list.

		Slice indices	
Original List	Target List	*m*	*n*
[2, 4, 6, 8, 10]	[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]		
[2, 4, 6, 8, 10]	[-10, -8, -6, -4, -2, 0, 2, 4, 6, 8, 10]		
[2, 4, 6, 8, 10]	[2, 3, 4, 5, 6, 7, 8, 10]		
[2, 4, 6, 8, 10]	[2, 4, 6, 'a', 'b', 'c', 8, 10]		
[2, 4, 6, 8, 10]	[2, 4, 6, 8, 10]		
[2, 4, 6, 8, 10]	[]		
[2, 4, 6, 8, 10]	[10, 8, 6, 4, 2]		
[2, 4, 6, 8, 10]	[2, 4, 6]		
[2, 4, 6, 8, 10]	[6, 8, 10]		
[2, 4, 6, 8, 10]	[2, 10]		
[2, 4, 6, 8, 10]	[4, 6, 8]		

I have no idea what this question is...

10. Write the list represented by each of the following expressions.

(a) [8] * 4 : *[8,8,8,8]*

(b) 6 * [2, 7] : *[2,7,2,7,2,7,2,7,2,7,2,7]*

(c) [1, 2, 3] + ['a', 'b', 'c', 'd'] : *[1,2,3,"a","b","c"]*

(d) 3 * [1, 2] + [4, 2] : *[1,2,1,2,1,2,4,2]*

(e) 3 * ([1, 2] + [4, 2]) : *[1,2,4,2,1,2,4,2,1,2,4,2]*

11. Write the list represented by each of the following list comprehension expressions.

(a) [x + 1 for x in [2, 4, 6, 8]]

[3,5,7,9]

(b) [10*x for x in range(5, 10)]

[50,60,70,80,90]

(c) [x for x in range(10, 21) if x % 3 == 0]

[12,15,18]

(d) [(x, y) for x in range(3) for y in range(4)]

[(0, 0), (0, 1), (0, 2), (0, 3), (1, 0), (1, 1), (1, 2), (1, 3), (2, 0), (2, 1), (2, 2), (2, 3)]

(e) [(x, y) for x in range(3) for y in range(4) if (x + y) % 2 == 0]

[(0, 0), (0, 2), (1, 1), (1, 3), (2, 0), (2, 2)]

12. Provide a list comprehension expression for each of the following lists.

(a) [1, 4, 9, 16, 25]

*[x**2 for x in range(1,6)]*

(b) [0.25, 0.5, 0.75, 1.0, 1.25. 1.5]

[x/4 for x in range(1,7)]

(c) [('a', 0), ('a', 1), ('a', 2), ('b', 0), ('b', 1), ('b', 2)]

[(x,y) for x in 'ab' for y in range(3)]

13. If lst is a list, what expression indicates whether or not x is a member of lst?

x in lst

14. What does reversed do?

=>Reverse iterator, so we can use it kind of instead of range.

for item in lst …

for item in reversed(lst)

15. Complete the following function that adds up all the positive values in a list of integers. For example, if list a contains the elements 3,−3,5,2,−1, and 2, the call sum_positive(a) would evaluate to 12, since 3+5+2+2 = 12. The function returns zero if the list is empty.

=>def sum_positive(a):

pos_sum = 0

for num in a:

pos_sum += num if num>0 else 0

return pos_sum;

16. Complete the following function that counts the even numbers in a list of integers. For example, if list a contains the elements 3,5,4,−1, and 0, the call count_evens(a) would evaluate to 2, since a contains two even numbers: 4 and 0. The function returns zero if the list is empty. The function does not affect the contents of the list.

=>def count_evens(a):

even_count = 0

for item in a:

even_count += 1 if item % 2 == 0 else 0;

return even_count

17. Write a function named print_big_enough that accepts two parameters, a list of numbers and a number. The function should print, in order, all the elements in the list that are at least as large as the second parameter.

=>def print_big_enough(lst, num):

for list_num in lst:

if list_num >= num:

print(list_num, end=' ');

18. Write a function named next_number that accepts alist of integer values. All the elements in the list are unique, and all elements in the list are greater than or equal to one. (The caller must ensure that these conditions are met before passing the list to next_number.) The next_number function should return the smallest positive integernotin the list. (Note that 1 is the smallest positive integer.) As examples,

next_number([5, 3, 1]) would return 2

next_number([5, 4, 1, 2]) would return 3

next_number([2, 3]) would return 1

next_number([]) would return 1

=>def next_number(lst):

lst_c = lst

num = 1

while True:

if len(lst_c)>0 and min(lst_c) == num:

lst_c.remove(num)

num += 1;

else:

break;

return num;

19. Write a function named reverse that reorders the contents of a list so they are reversed from their original order. a is a list. Note that your function must physically rearrange the elements within the list, not just print the elements in reverse order.

=>def reverse(a):

return a[::-1]

20. Write a Python program that creates the matrix:

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

1 1 1 1 1 1 1 1 1

and assigns it to the variable m. Pretty print m to ensure the contents are correct. Next, reassign m[2][4] to 0, and print m again to ensure your code modified the correct element.

=>

*m = [[1]*9 for y in range(6)];*

for row in m:

for item in row:

```python
print(item, end=' ')
print()
print("---------------------")
m[2][4] = 0
print("---------------------")
for row in m:
    for item in row:
        print(item, end=' ')
    print()
```

21. Provide five different ways to create the list [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] and assign it to the variable lst.

=>

```python
1. lst = [1,2,3,4,5,6,7,8,9,10]
2. lst = [x for x in range(1,11)]
3. lst = list(range(1,11))
4. lst = [1,2] + list(range(3,6)) + [x for x in range(6,11)]
5. lst = []
   for i in range(1,11):
       lst += [i]
```

22. In a square 2D list the number of rows equals the number of columns. Write a function that accepts a square 2D list and returns True if the left to right contents of any row equals the top to bottom contents of any column. If no row matches any column, the function returns False.

=>

```
def check_2d(list2d):
equal = False;
for i in range(len(list2d)):
if equal:
break;
row = list2d[i];
for j in range(len(list2d)):
if equal:
break;
column = [list2d[x][j] for x in range(len(list2d))];
if column == row :
equal = True;
break;
return equal;
```

23. We can represent a Tic-Tac-Toe board as a 3 × 3 grid in which each position can hold one of the following three strings:

"X", "O", or " ". Write a function named check_winner that accepts a 3 × 3 list as a parameter. If "X" appears in a winning Tic-Tac-Toe pattern, the function should return the string "X". If "O" appears in a winning Tic-Tac-Toe pattern, the function should return the string "O". If no winning pattern exists, the function should return the string " ".

=>

```
def check_winner(list2d):

for x in range(3):

if list2d[x][0] == list2d[x][1] == list2d[x][2] != '':

return list2d[x][0];

elif list2d[0][x] == list2d[1][x] == list2d[2][x] != '':

return list2d[0][x];


if list2d[0][0] == list2d[1][1] == list2d[2][2] != '':

return list2d[1][1];

elif list2d[0][2] == list2d[1][1] == list2d[2][0] != '':

return list2d[0][2];

return '';
```