Chapter 10 , {exercises}

شایسته گیوه ای

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

>> Yes

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

>> It's return the value from end of list .

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

>> sort() method

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]

(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? 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 ?

>> Square brackets []

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 fales

(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.

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]	0	5
[2, 4, 6, 8, 10]	[]		
[2, 4, 6, 8, 10]	[10, 8, 6, 4, 2]	-1	-6
[2, 4, 6, 8, 10]	[2, 4, 6]	0	3
[2, 4, 6, 8, 10]	[6, 8, 10]	2	5
[2, 4, 6, 8, 10]	[2, 10]	1	-1
[2, 4, 6, 8, 10]	[4, 6, 8]	1	4

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']

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

(e) 3 * ([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 ['a','b'] for y in range(3)]

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

>> in lst --> Return True or False

14. What does reversed do?

>> Physically reverses the elements in the list. The list is modified.

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): # Add your code... sum = 0 for i in a : if i < 0 : pass else : sum += i print ("Sum of positive number is %i"%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(lst): # Add your code... count = 0 if len(lst) == 0 : return 0 break for i in lst: if lst % 2 == 0: count += 1 if count > 0 : print("We have %i even number :)"%count) else: print("We dont have any even number "

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 i in lst:
        if i >= num:
            print(i)
# example usage
print_big_enough([1, 2, 3, 4, 5], 3) # prints 3, 4, 5
```

18. Write a function named next_number that accepts a list 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 integer not in 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.sort()
    next_num = 1
    for num in lst:
        if num == next_num:
            next_num += 1
return next_num
# Example usage
```

```
print(next_number([5, 3, 1])) # Output: 2
print(next_number([5, 4, 1, 2])) # Output: 3
print(next_number([2, 3])) # Output: 1
print(next_number([])) # Output: 1
```

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):
    left = 0
    right = len(a) – 1

    while left < right:
        a[left], a[right] = a[right], a[left]
        left += 1
        right -= 1
# Example usage
lst = [1,2,3,4]
reverse(lst)
print(lst) # Output: [4,3,2,1]
```

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 i in range(6)]
print(m)
m[2][4] = 0
print(m)
```

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.

```
>> 1. Using a list comprehension: lst = [i for i in range(1, 11)]
```

2. Using the range() function and converting it to a list: lst = list(range(1, 11))

3. Concatenating two lists: lst = [1, 2, 3, 4, 5] + [6, 7, 8, 9, 10]

4. Using the extend() method: lst = [] lst.extend(range(1,11))

5. Using the append() method in a loop: lst = [] for i in range(1,11): lst.append(i)

22. In a square 2D list the number of rows equals the nnumber 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.

>> Here's one possible implementation:

python

```
def check_square_list(square_list):
    n = len(square_list)
    for i in range(n):
        row = square_list[i]
        col = [square_list[j][i] for j in range(n)]
        if row == col:
            return True
```

return False

This function takes a square 2D list as input and first determines its size n. It then iterates over each row index i and extracts the corresponding row and column using list comprehension. If the row and column are equal, it means that the left-to-right contents of the row match the top-to-bottom contents of the column, so we can return True immediately. If no such match is found after checking all rows, we return False.

Note that this implementation assumes that all elements in the square list are comparable for equality (i.e., they implement the == operator). If this is not the case, you may need to modify the function accordingly.

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(board):
    # check rows
    for row in board:
        if row[0] == row[1] == row[2] and row[0] != ' ':
            return row[0]
     # check columns
    for col in range(3):
        if board[0][col] == board[1][col] == board[2][col] and
board[0][col] != ' ':
            return board[0][col]
     # check diagonals
    if board[0][0] == board[1][1] == board[2][2] and board[0][0]
!= ' ':
        return board[0][0]
      if board[2][0] == board[1][1] == board[0][2] and board[2][0]
!= ' ':
        return board[2][0]


                                    return ' '
```