1. **Can a Python list hold a mixture of integers and strings?**
   Yes , it can . (. میتواند ، بله)
2. **What happens if you attempt to access an element of a list using a negative index?**
   When we use a negative index to access the elements, we actually call the elements of the list from the end to the beginning, for example: -1 means the last element of the list or -2 means the second element of the list from the end and...

   وقتی از شاخص منفی برای دسترسی به عناصر استفاده میکنیم در واقع ما عناصر لیست را از انتها به ابتدا فرا خوانی میکنیم برای مثال : 1- یعنی آخرین عنصر لیست یا 2- یعنی دومین عنصر لیست از آخر و ...

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

list = [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]

(c) What is lst[0]?
The first element of the list : 10

اولین عنصر لیست : 10

(d) What is lst[3]?
It includes the fourth element of the list (because the number of elements starts from 0) : 29

شامل چهارمین عنصر لیست میشود (چون شماره ی عناصر از 0 شروع میشوند) : 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?

No, because it's the result :

TypeError: list indices must be integers or slices, not float .

نه، چون این نتیجه است :

عرور برای نوع داده است : شاخص های لیست باید اعداد صحیح یا برش باشند، نه شناور .

5. Given the statements

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

x = 2

valuate 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() function

تابع len()

7. What expression represents the empty list?

For show empty list we use : [ ]

از [ ] برای نمایش لیست خالی استفاده می کنیم .

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.

Original List	Target List	Slice indices	
		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]`		

Original List [2,4,6,8,10]

Target List [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

m = len(a), n = len(a) + 5 a[m:n] = [12,14,16,18,20]

Modified List [2,4,6,8,10,12,14.16.18.20]

Target List [-10,-8,-6,-4,-2.0.2.4.6.8.10]

m = -11,n = -1 a[m:n] = [-10,-8,-6,-4,-2,O.2.4.6.8.lO]

Modified List [-10.-8.-6.-4.-2.O.2A.A.A.lO]

Target List [2.3A.A.A.A.A.lO]

m = l,n=7 a[m:n]=[2,A,A,A,A,lO]

Modified List [2,A,A,A.A.lO]

Target List [2A.A.cA.lOJ

m=3,n=6 a[m:n]=['a','b','c']

Modified List [2A.A.'a','b','c',8,lO]

Target List []

m=n=0 a[m:n]=[]

Modified List []

Target List [l0.B.B.B.BJ

m=-1,n=-6 a[m:n]=[l0,B,B,B]

Modified List[2A.A.'a','b','c',B,B,l0]

Target List [246J

m=0,n=3 a[m:n]=[2A.,4.,6.]

Modified list[246810]

Target list[6810J

m=3,n=len(a) a[m:n]=[8,10]

Modified list[2A.A.8.lO]

Target List [2.10]

m=0,n=2 a[m:n]=[2,lO]

Modified List [2A.A.8.lO]

Target List [468J

m=1,n=4 a[m:n]=[4,6,8]

Modified List[2A.,4.,6.,8,lO]

_

Original List [2,4,6,8,10] Target List m:n

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20] 5:11

[-10, -8, -6, -4, -2, 0, 2, 4, 6, 8, 10] -12:11

[2 ,3 ,4 ,5 ,6 ,7 ,8 ,10] :

[2 ,4 ,6 ,'a' ,'b' ,'c' ,8 ,10] 3:6

[] :

[10 ,8 ,6 ,4 ,2] ::-1

[2 ,4 ,6][:3]

[6 ,8 ,10][:-1]

[2 ,10][::len([2])]

[4 ,6 ,8][:-1]

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]) : [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)] :

[(char, num) for char in ['a', 'b'] for num in range(3)]

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

X in lst => If X is in lst, the result will be true, otherwise the result will be false . (boolean)

اگر X در لیست باشد نتیجه درست خواهد بود ،درغیر این صورت نتیجه غلط خواهد بود .

14. What does reversed do?

The reversed() function in Python returns a reverse iterator. It reverses the order of the elements in a sequence such as a list, tuple, or string. The original sequence is not modified, but a new reversed sequence is created. The reversed() function can be used with a for loop or converted to a list using the list() function.

این تابع در پایتون یک تکرار کننده معکوس برمی گرداند. ترتیب عناصر را در یک دنباله مانند لیست، تاپل یا رشته معکوس می کند. دنباله اصلی اصلاح نشده است، اما یک دنباله معکوس جدید ایجاد می شود. این تابع را می توان با حلقه (فور) استفاده کرد یا با استفاده از تابع لیست به لیست تبدیل کرد.

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...

```
def sum_positive(a):
    total = 0
    for num in a:
        if num > 0:
            total += num
    return total if a else 0
```

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...

```
def count_evens(lst):
    count = 0
    for num in lst:
        if num % 2 == 0:
            count += 1
    return 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.

```
# code for function
def print_big_enough(numbers, minimum):
    for num in numbers:
        if num >= minimum:
            print(num)
```

```
#call function like this

data = [3, 5, 7, 2, 8, 4]

print_big_enough(data, 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 = sorted(lst)
    next_num = 1
    for num in lst:
        if num == next_num:
            next_num += 1
    return next_num
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):
    """
    Reverses the contents of a list in place.
    """
    i = 0
    j = len(a) - 1
    while i < j:
        a[i], a[j] = a[j], a[i]
        i += 1
        j -= 1
my_list = [1, 2, 3, 4, 5]
reverse(my_list)
print(my_list) # Output: [5, 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 = [[0.1]*9 for i in range(7)]
m[2][4] = 0
for row in m:
    print(row)
```

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. Using the append() method in a for loop:

```
lst = []
for i in range(1, 11):
    lst.append(i)
```

4. Using the extend() method with an empty list and the range() function:

```
lst = []
lst.extend(range(1, 11))
```

5. Using the * operator to repeat a tuple and then converting it to a list:

```
tup = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
lst = list(tup * 1)
```

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.

```
def check_square_list(square_list):
    n = len(square_list)
    for i in range(n):
        row_sum = sum(square_list[i])
        col_sum = sum(square_list[j][i] for j in range(n))
        if row_sum == col_sum:
            return True
return False
```

If you want to add error checking to ensure that this is the case, you can add something like this at the beginning of the function:

```
def check_square_list(square_list):
    n = len(square_list)
    if any(len(row) != n for row in square_list):
        raise ValueError("Input must be a square 2D list")
    # rest of function...
```

This will raise a ValueError if any row has a different length than the number of rows/columns.

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.count("X") == 3:
            return "X"
        elif row.count("O") == 3:
            return "O"

    # Check columns
    for i in range(3):
        column = [board[j][i] for j in range(3)]
        if column.count("X") == 3:
            return "X"
        elif column.count("O") == 3:
            return "O"

    # Check diagonals
    diagonal1 = [board[i][i] for i in range(3)]
    diagonal2 = [board[i][2-i] for i in range(3)]
    if diagonal1.count("X") == 3 or diagonal2.count("X") == 3:
        return "X"
    elif diagonal1.count("O") == 3 or diagonal2.count("O") == 3:
        return "O"
    # No winner found
    return " "
```