Chapter 10 Answersheet

1.yes.

2. A negative list index represents a negative offset from an imaginary element one past the end of the list. For list a, the expression a[-1] represents the last element in a. The
expression al-2] represents the next to the last element, and so forth. If a contains n elements, the expression a[0] corresponds to Ist[-n].

3.lst=[45,-3,16,8]

4.
(a)10
(b)29
(c)10
(d)29
(e)-4
(f)29
(g)10
(h)illegal

5.
(a)3
(b)5
(c)1
(d)5
(e)5
(f)illegal
(g)0
(h)3

6.len

7.make_zero

8.
(a)[20,1,-34,40,-8,60,1,3]
(b)[20,1,-34]
(c)[-8,60,1,3]
(d)[-8,60,1,3]
(e)[40,-8]
(f)[20,1,-34]
(g) [-8,60,1,3]
(h)[20,1,-34,40,-8,60,1,3]
(i)[20,1,-34,40]
(j)[1,-34,40,-8]
(k)True
(l)False
(m)8

9.
-Target List
-m n

[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
0 10

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

[2, 3, 4, 5, 6, 7, 8, 10]
0 8

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

[2, 4, 6, 8, 10]
0 5

[]
0 0

[10, 8, 6, 4, 2]
4 0

[2, 4, 6]
0 3

[6, 8, 10]
2 5

[2, 10]
0 2

[4, 6, 8]
1 4

10.
(a)[8,8,8,8]
(b)[2,7,2,7,2,7,2,7,2,7,2,7]
(c)[1,2,3,'a','b','c','d']
(d)[1,2,1,2,1,2,4,2]
(e)[1,2,4,2,1,2,4,2,1,2,4,2]

11.
(a)[3,5,7,9]
(b)[50,60,70,90]
(c)[12,15,18]
(d)[(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)[(0,0),(0,2),(1,1),(1,3),(2,0),(2,2)]

12.

(a)[x**2 for x in rang (6)]
(b)[x/4 for x in rang (7)]
(c)[(x,y) for x in[‘a’,’b’] for y in rang(3)]

13.

If Ist is a list, the expression x in 1st evaluates to True if x in an element in 1st; otherwise, the expression is False. Similarly, the expression × not in 1st evaluates to True if x is not an element in 1st; otherwise, the expression is False

14.
Physically reverses the elements in the list. The list is modified.

15.
```
def sum_positive(a):
  result = 0
  for x in a:
    if x > 0:
     result += x
  return result
```

16.
```
def count_evens(Ist):
  count = 0
  for x in Ist:
     if x % 2 == 0:
        count += 1
   return count
```

17.
```
def print_big_enough(numbers, threshold):
   for num in numbers:
       if num >= threshold:
           print(num)
```

18.
```
def next_number(numbers):
   numbers = set(numbers)
   next_num = 1
   while next_num in numbers:
       next num += 1
   return next_num
```

19.
```
def reverse(a):
  left = 0
  right = len(a) - 1
  while left < right:
     a[left], a[right] = a[right], a[left]
     left + = 1
     right -= 1
```

20.

```
matrix = [[1 for j in range(10)] for i in
range(7)]
  for i in range(5): for j in range(9):
      print(matrix[i][j], end=” “)
  print()
```

21.
1. Using a list comprehension:

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

2.Using the list() function and the
'range() function:

```
lst = list(range(1, 11))
```

3. Using the append() method and a loop:

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

4. Using the ' extend()' method and a list of values:

```
lst = []
lst.extend([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
```

5. Using the ‘*’operator to repeat a tuple of values, and then converting to a list:

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

22.

```
def check_square(matrix):
  n = len(matrix)
  for i in range(n):
  row_sum = sum(matrix[i])
  col_sum = 0
  for j in range(n):
    col_sum += matrix[i]li]
  if row_sum == col_sum:
    return True
  return False
```

23.

```
def check winner(board):
   for row in board:
      if row[0] == row[1] == row[2]!= " "
        return row[0]

   for i in range(3):
     if board[0][i] == board[1][i] == board [2][1] != " ".
       return board[0][i]

   if board[0][0] == board [1][1] == board[2] [2] != "
     return board[01[01
   elif board[0][2] == board[1][1] == board [2][01 != "".
     return board [0][2]
```

```
return ""
```