Lists in Python

January 25, 2023

1 Lists in Python

- List is an ordered collection of elements (of same type (Homogeneous) or of different types (Heterogeneous).
- List elements are enclosed within square braces []
- Homogeneous list is a list that contains same type of elements. Ex: [10, 20, 30], [12.5, 15.5], [“abc”, “def”, “ghi”]
- Heterogeneous list is a list that contains different types of elements. Ex: [10, 20.2, “Hello”, True]
- List elements have indices using which they can be accessed
- List index starts from 0
- Which means 1st element in the list exists at 0 index, 2nd element at 1 index, 3rd element at 2 index and so on
- Ex: [10, 20, 30, 40, 50]
- Ind 0 1 2 3 4
- We can use indexes to access list elements
- Accessing the list element using index: list_name[index_value]
- Ex: lst = [10, 20, 30, 40, 50], then accessing 30 would be like lst[2], accessing 50 would be like lst[4]

[3]:
```
lst = [10, 20, 30, 40]
print(lst)
```

```
[10, 20, 30, 40]
```

[4]:
```
a = 10
b = 12.2
c = 'hello'
d = True
x = [a, b, c, d]
print(x)
```

```
[10, 12.2, 'hello', True]
```

[8]:
```
list_int = [10, 20, 30, 40] # len(list_int) = 4
# ind        0  1   2   3 --> Positive indexing
print(list_int) # indexes start from 0
print(list_int[3])
print(list_int[1])
```

```
[10, 20, 30, 40]
40
20
```

[10]:
```
list_int = [10, 20, 30, 40, 50] # len(list_int) = 4
# ind        0  1   2   3   4--> Positive indexing
print(list_int) # indexes start from 0
print(list_int[1], list_int[-4])
print(list_int[3], list_int[-2])
```

```
[10, 20, 30, 40, 50]
20 20
40 40
```

[15]:
```
names = ['Amarendra Baahubali', 'Sivagami', 'Kalakeya', 'Avantika',
        'Kattappa', 'Devasena', 'Bhallaladeva', 'Mahendra Baahubali']
# Devasena loves Amarendra Baahubali
print(f'{names[5]} loves {names[0]}')
print(f'{names[-3]} loves {names[-8]}')
```

```
Devasena loves Amarendra Baahubali
Devasena loves Amarendra Baahubali
```

[16]:
```
names = ['Amarendra Baahubali', 'Sivagami', 'Kalakeya', 'Avantika',
        'Kattappa', 'Devasena', 'Bhallaladeva', 'Mahendra Baahubali']
# Kattappa kills Amarendra Baahubali
print(f'{names[4]} kills {names[0]}')
print(f'{names[-4]} kills {names[-8]}')
```

```
Kattappa kills Amarendra Baahubali
Kattappa kills Amarendra Baahubali
```

[17]:
```
names = ['Amarendra Baahubali', 'Sivagami', 'Kalakeya', 'Avantika',
        'Kattappa', 'Devasena', 'Bhallaladeva', 'Mahendra Baahubali']
# Sivagami sacrifices herself to save Mahendra Baabhubali
print(f'{names[1]} sacrifices herself to save {names[7]}')
print(f'{names[-7]} sacrifices herself to save {names[-1]}')
```

```
Sivagami sacrifices herself to save Mahendra Baahubali
Sivagami sacrifices herself to save Mahendra Baahubali
```

1.1 Mutability of lists

- Mutability is the ability to be changed even after creation
- Lists are mutable

[18]:
```
names = ['Rocky', 'Gadura', 'Vaanaram', 'Shatni', 'Adhera', 'Rnia']
print(f'Before change: {names}')
names[1] = 'Garuda'
names[3] = 'Shanti'
```

```
names[4] = 'Adheera'
names[-1] = 'Rina'
print(f'After change: {names}')
```

```
Before change: ['Rocky', 'Gadura', 'Vaanaram', 'Shatni', 'Adhera', 'Rnia']
After change: ['Rocky', 'Garuda', 'Vaanaram', 'Shanti', 'Adheera', 'Rina']
```

1.2 Built_in functions that can be applied on lists

[19]:
```
nums = [12, 14, 5, 6, 7]
# how many elements
print(len(nums))
```

```
5
```

[20]:
```
print(len('aaabbbcccdddee'))
```

```
14
```

[21]:
```
print(len(range(10, 100, 10))) #
```

```
9
```

[22]:
```
nums = [10, 20, 30, 40]
print(max(nums))
```

```
40
```

[23]:
```
nums = [10, 20, 30, 40]
print(min(nums))
```

```
10
```

[24]:
```
my_list = [10, 12.2, 'hello', True]
print(len(my_list))
```

```
4
```

[25]:
```
my_list = [10, 12.2, 'hello', True]
print(max(my_list))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [25], in <cell line: 2>()
      1 my_list = [10, 12.2, 'hello', True]
----> 2 print(max(my_list))

TypeError: '>' not supported between instances of 'str' and 'float'
```

[26]:
```
names = ['Rocky', 'Gadura', 'Vaanaram', 'Shatni', 'Adhera', 'Rnia']
print(len(names))
```

```
6
```

[28]:
```
names = ['Rocky', 'Gadura', 'Vaanaram', 'Shatni', 'Adhera', 'Rnia']
print(max(names))
# Ad Ga Rnia Rocky Shan Vaan
```

```
Vaanaram
```

[29]:
```
names = ['Rocky', 'Gadura', 'Vaanaram', 'Shatni', 'Adhera', 'Rnia']
print(min(names))
# Ad Ga Rnia Rocky Shan Vaan
```

```
Adhera
```

1.3 Traversing through a list

- Accessing the elements of a list using loops

1.3.1 Element Based Access

- We can access the elements of a list using following syntax Syntax:

```
for i in list_name:
    print(i)
```

[30]:
```
nums = [10, 20, 30, 40, 50]
# ind   0   1   2   3   4
for i in nums: # i = 20
    print(i) # 10 20 30 40 50
```

```
10
20
30
40
50
```

[33]:
```
# Print the squares of all elements in the list
nums = [10, 20, 30, 40, 50]
for i in nums:
    print(i ** 2)
```

```
100
400
900
1600
2500
```

[38]:
```
# Print all the even numbers in the given list
lst = [34, 20, 91, 86, 21, 35, 37, 51, 88, 66]
for i in lst:
    if i % 2 == 0:
        print(i, end = ' ')
```

```
34 20 86 88 66
```

[42]:
```
# Print the elements in the following list who ends with a 1
lst = [19, 55, 6, 15, 21, 41, 31, 77, 44, 59]
for i in lst:
    if i % 10 == 1:
        print(i, end = ' ')
```

```
21 41 31
```

1.3.2 Index Based Access - Mostly Used

- We can access the list elements by running a loop over it's indexes

```
for i in range(len(list_name)):
    print(list_name[i])
```

[45]:
```
# index based access
lst = [19, 55, 6, 15, 21, 41, 31] # len(lst) = 7
#ind   0   1   2  3   4   5   6
for i in range(len(lst)):
    print(lst[i], end = ' ') # i = lst[0] lst[1] lst[2] 3
```

```
19 55 6 15 21 41 31
```

[46]:
```
# Print all the even numbers in the given list
lst = [34, 20, 91, 86, 21, 35, 37, 51, 88, 66]
for i in range(len(lst)):
    if lst[i] % 2 == 0:
        print(lst[i], end = ' ')
```

```
34 20 86 88 66
```

[51]:
```
# Print all the odd numbers in the given list
lst = [9, 18, 39, 23, 32, 2, 67, 87, 99, 90]
for i in lst:
    if i % 2 != 0:
        print(i, end = ' ')
```

```
9 39 23 67 87 99
```

[52]:
```
# Print all the odd numbers that are present at odd indexes in the given list
lst = [9, 18, 39, 23, 32, 2, 67, 87, 99, 90] # len(lst) = 10
# ind  0  1   2   3   4   5  6   7   8   9 # len(lst) - 1
for i in range(len(lst)):
```

```
    if i % 2 == 1 and lst[i] % 2 == 1:
        print(lst[i], end = ' ')
```

```
23 87
```

[57]:
```
# find out how many odd numbers in the given list are present
# between exactly two even numbers
x = [33, 91, 26, 46, 42, 19, 45, 68, 7, 90, 72, 82, 84, 95, 86]
#in  0   1   2   3   4   5   6   7   8  9   10  11  12  13  14
cnt = 0
for i in range(1, len(x) - 1): # i = 1
    if x[i] % 2 != 0 and x[i - 1] % 2 == 0 and x[i + 1] % 2 == 0:
        print(x[i], end = ' ')
        cnt += 1
print(cnt)
```

```
7 95 2
```

1.3.3 Traversing a list from the end

[61]:
```
#nind  -5  -4  -3  -2  -1
lst = [10, 20, 30, 40, 50]
#ind   0   1   2   3   4
# index based access
for i in range(len(lst)-1, -1, -1):
    print(lst[i], end = ' ')
```

```
50 40 30 20 10
```

[65]:
```
#nind  -5  -4  -3  -2  -1
lst = [10, 20, 30, 40, 50]
#ind   0   1   2   3   4
# index based access
for i in range(-1, -6, -1): # -2 -> -2-1 -> 1
    print(lst[i], end = ' ')
```

```
50 40 30 20 10
```

[66]:
```
#nind  -5  -4  -3  -2  -1
lst = [10, 20, 30, 40, 50]
#ind   0   1   2   3   4
# index based access
for i in range(1, 6): # -2 -> -2-1 -> 1
    print(lst[-i], end = ' ')
```

```
50 40 30 20 10
```

[67]:
```
# find out the last odd number in the list
lst = [33, 91, 26, 46, 42, 19, 45, 68, 7, 90, 72, 82, 84, 95, 86]
for i in lst:
```

```
    if i % 2 == 1:
        o = i
print(o)
```

95

[69]:
```
# find out the last odd number in the list
lst = [33, 91, 26, 46, 42, 19, 45, 68, 7, 90, 72, 82, 84, 95, 86]
for i in range(len(lst) - 1, -1, -1):
    if lst[i] % 2 == 1:
        print(lst[i])
        break
```

95

1.4 list of elements as input from user

[47]:
```
# list of integers
lst = list(map(int, input().split()))
print(lst)
```

```
10 20 30 40 50
[10, 20, 30, 40, 50]
```

[48]:
```
# list of floating point values
lst = list(map(float, input().split()))
print(lst)
```

```
12.2 14.6 17.8 100.7
[12.2, 14.6, 17.8, 100.7]
```

[49]:
```
# list of strings
lst = list(map(str, input().split()))
print(lst)
```

```
class is over you can leave
['class', 'is', 'over', 'you', 'can', 'leave']
```

[ ]:
```

```

1.5 List methods

- Addition to list
  - list.append()
  - list.exetend()
  - list.insert()
- Deletions from list
  - list.pop()
  - list.remove()
  - list.clear()

- Other Operations
  - list.reverse()
  - list.sort()
  - list.index()
  - list.count()

1.5.1 list.append()

- used to add an object at the end of the list

[2]:
```
marks = [35, 45, 75]
# append()
marks.append(85)
print(marks)
```

```
[35, 45, 75, 85]
```

[3]:
```
marks = []
marks.append(55)
marks.append(65)
print(marks)
```

```
[55, 65]
```

[4]:
```
marks = [55, 65]
new_marks = [75, 85]
marks.append(new_marks) # [55, 65, [75, 85]]
print(marks)
```

```
[55, 65, [75, 85]]
```

[5]:
```
marks = [55, 65]
marks.append('hello')
print(marks)
print(len(marks))
```

```
[55, 65, 'hello']
3
```

[6]:
```
marks = [55, 65]
marks.append(55, 65, 75)
print(marks)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [6], in <cell line: 2>()
      1 marks = [55, 65]
----> 2 marks.append(55, 65, 75)
      3 print(marks)
```

```
TypeError: list.append() takes exactly one argument (3 given)
```

[10]:
```
lst = []
n = int(input()) # howmany list elements # 5
for i in range(n): # i = 1 2 3 4
    x = int(input()) # x = 20
    lst.append(x) # lst.append(20) [10, 20, 30, 40, 50]
print(lst)
```

```
5
10
20
30
40
50
[10, 20, 30, 40, 50]
```

[12]:
```
lst = [33, 91, 26, 46, 42, 19, 45, 68, 7, 90, 72, 82, 84, 95, 86]
# Create a new list with only even numbers from above list
evens = []
for i in lst:
    if i % 2 == 0:
        evens.append(i)
print(evens)
```

```
[26, 46, 42, 68, 90, 72, 82, 84, 86]
```

[ ]:

[9]:
```
lst = list(map(int, input().split()))
print(lst)
```

```
10
[10]
```

[23]:
```
names = ['rocky', 'garuda', 'vaanaram', 'shanti', 'adheera', 'rina']
new = []
for i in names:
    t = (max(i), min(i), len(i))
    new.append(t) # tuple
print(new)
```

```
[('y', 'c', 5), ('u', 'a', 6), ('v', 'a', 8), ('t', 'a', 6), ('r', 'a', 7),
('r', 'a', 4)]
```

[17]:
```
x = 'rocky'
# print(max(x))
# print(min(x))
```

```
# print(len(x))
t = (max(x), min(x), len(x))
print(t)
```

```
('y', 'c', 5)
```

1.5.2 list.extend()

- Used extend an existing with list an iterable
- extend() takes every element from the iterable object and appends it to the existing list

[25]:
```
marks = [56, 72] # [56, 72, 89, 72, 46]
new_marks = [89, 72, 46]
marks.extend(new_marks)
print(marks)
```

```
[56, 72, 89, 72, 46]
```

[27]:
```
x = [10, 20] # [10, 20, 'p', 'y', 't', 'h', 'o', 'n']
x.extend('python')
print(x) #
print(len(x))
```

```
[10, 20, 'p', 'y', 't', 'h', 'o', 'n']
8
```

[28]:
```
x = [10, 20] # [10, 20, 'python']
x.append('python')
print(x) #
print(len(x))
```

```
[10, 20, 'python']
3
```

[29]:
```
# we can only pass iterables as arguments to extend()
x = [10, 20]
x.extend(123)
print(x)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [29], in <cell line: 3>()
      1 # we can only pass iterables as arguments to extend()
      2 x = [10, 20]
----> 3 x.extend(123)
      4 print(x)

TypeError: 'int' object is not iterable
```

[30]:
```
# we can only pass iterables as arguments to extend()
x = [10, 20]
x.extend('123')
print(x)
```

```
[10, 20, '1', '2', '3']
```

[32]:
```
# range() is also an iterable object
nums = []
nums.extend(range(100, 0, -1))
print(nums)
```

```
[100, 99, 98, 97, 96, 95, 94, 93, 92, 91, 90, 89, 88, 87, 86, 85, 84, 83, 82,
81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62,
61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42,
41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22,
21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

1.5.3 list.insert()

- Used to insert a value at a particular index position

[34]:
```
x = [10, 30]
x.insert(1, 20)
print(x)
```

```
[10, 20, 30]
```

[38]:
```
char = ['a', 'c', 'e']
# ind    0    1    2
# insert 'b' between a and c
char.insert(1, 'b') # ['a', 'b', 'c', 'e']
# insert 'd' between c and e
char.insert(3, 'd')
print(char)
```

```
['a', 'b', 'c', 'd', 'e']
```

[44]:
```
x = [10, 20] # >=len(lst) at the end
x.insert(12, 30)
print(x)
```

```
[10, 20, 30]
```

1.5.4 list.index()

- Used to find the index of a particular value in the list
- Always gives the index of first occurance of the value

[46]:
```
lst = [10, 20, 30]
print(lst.index(30))
```

```
2
```

[47]:
```
lst = [10, 20, 30, 10, 20, 30, 10, 20, 30]
print(lst.index(30))
```

```
2
```

[49]:
```
lst = [10, 20, 30, 10, 20, 30, 10, 20, 30]
print(lst.index(30, 3))
```

```
5
```

[50]:
```
lst = [10, 20, 30, 10, 20, 30, 10, 20, 30]
print(lst.index(20, 5))
```

```
7
```

[52]:
```
x = [10, 20, 30, 40]
# ind 0  1   2
print(x.index(40, 0, 4))
```

```
3
```

[53]:
```
z = [100, 200, 300]
print(z.index(500))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [53], in <cell line: 2>()
      1 z = [100, 200, 300]
----> 2 print(z.index(500))

ValueError: 500 is not in list
```

1.5.5 list.pop()

- Used to pop an element at specified index. If no index specified pops the last element from the list
- Returns the popped element

[3]:
```
lst = [10, 20, 30, 40, 50]
lst.pop()
print(lst)
```

```
[10, 20, 30, 40]
```

[4]:
```
lst = [10, 20, 30, 40, 50]
lst.pop(2)
print(lst)
```

```
[10, 20, 40, 50]
```

[5]:
```
lst = [10, 20, 30, 40, 50]
lst.pop(2, 3, 4)
print(lst)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [5], in <cell line: 2>()
      1 lst = [10, 20, 30, 40, 50]
----> 2 lst.pop(2, 3, 4)
      3 print(lst)

TypeError: pop expected at most 1 argument, got 3
```

[7]:
```
lst = [10, 20, 30, 40, 50]
lst.pop(2) # [10, 20, 40, 50]
lst.pop(3) # [10, 20, 40]
print(lst)
```

```
[10, 20, 40]
```

1.5.6 list.remove()

- Removes the first occurence of the specified element from the list

[8]:
```
lst = [10, 20, 30, 40, 50]
# lst.remove() is value based deletion
lst.remove(20)
print(lst)
```

```
[10, 30, 40, 50]
```

[9]:
```
lst = [10, 20, 30, 40, 50, 20, 40, 50, 20]
# lst.remove() is value based deletion
lst.remove(20)
print(lst)
```

```
[10, 30, 40, 50, 20, 40, 50, 20]
```

[12]:
```
# remove all the occurences of 10 from the given list
lst = [10, 20, 30, 40, 50, 10, 20, 40, 50, 20, 10, 20, 10, 10]
while True:
    if 10 not in lst:
        break
    else:
        lst.remove(10)
```

```
print(lst)
```

```
[20, 30, 40, 50, 20, 40, 50, 20, 20]
```

[13]:
```
# remove all the occurences of 10 from the given list
lst = [10, 20, 30, 40, 50, 10, 20, 40, 50, 20, 10, 20, 10, 10]
while 10 in lst:
    lst.remove(10)
print(lst)
```

```
[20, 30, 40, 50, 20, 40, 50, 20, 20]
```

[14]:
```
lst = [10, 20, 30]
lst.remove(100)
print(lst)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [14], in <cell line: 2>()
      1 lst = [10, 20, 30]
----> 2 lst.remove(100)
      3 print(lst)

ValueError: list.remove(x): x not in list
```

[ ]:
```
index based deletion - pop()
element based deletion - remove()
```

1.5.7 list.clear()

- Clears the list and leaves the empty list

[17]:
```
lst = [10, 20, 30]
lst.clear()
print(lst)
```

```
[]
```

1.5.8 list.count()

- Returns the number of occurances of an element in a list

[19]:
```
lst = [10, 20, 30, 40, 10, 30, 40, 10, 30]
print(lst.count(10))
print(lst.count(20))
print(lst.count(100))
```

```
3
1
0
```

1.5.9 list.reverse()

- Reverses the list in-place

[20]:
```
lst = [10, 20, 30, 40, 10, 30, 40, 10, 30]
lst.reverse()
print(lst)
```

```
[30, 10, 40, 30, 10, 40, 30, 20, 10]
```

1.5.10 list.sort()

- Sorts the given list in ascending order (in-place)
- If the sort has to happen in descending order use the following list.sort(reverse = True)

[21]:
```
lst = [10, 20, 30, 40, 10, 30, 40, 10, 30]
lst.sort() # [10, 10, 10, 20, 30, 30, 30, 40, 40]
print(lst)
```

```
[10, 10, 10, 20, 30, 30, 30, 40, 40]
```

[24]:
```
lst = [10, 20, 30, 40, 10, 30, 40, 10, 30]
lst.sort(reverse = True) # [40, 40, 30, 30, 30, 20, 10, 10, 10]
print(lst)
```

```
[40, 40, 30, 30, 30, 20, 10, 10, 10]
```

1.5.11 list.copy()

Deep Copy Vs. Shallow Copy

- If two object are deep copied, every change we make on one object will be reflected on the other object
- If two object are shallow copied, every change we make on one object will not be reflected on the other object

[34]:
```
lst1 = [10, 20, 30]
lst2 = lst1 # deep copy
lst2.extend('hello')
print(lst1)
print(lst2)
```

```
[10, 20, 30, 'h', 'e', 'l', 'l', 'o']
[10, 20, 30, 'h', 'e', 'l', 'l', 'o']
```

[33]:
```
lst1 = [10, 20, 30]
lst2 = lst1.copy() # shallow copy
lst2.extend('hello')
print(lst1)
print(lst2)
```

```
[10, 20, 30]
[10, 20, 30, 'h', 'e', 'l', 'l', 'o']
```

1.6 List Slicing

- Slicing is a way to get sub-parts from an existing list
- Slicing can be done using slicing operator :
- Syntax

`list_name[start_index:stop_index:index_jump]`

- Every slicing operation returns a new list
- Defaults in slicing
  - start_index –> 0
  - stop_index –> len(list)
  - index_jump –> 1

[36]:
```
lst = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
# ind  0   1   2   3   4   5   6   7   8   9
new_list = lst[2:5:1]
print(new_list)
print(lst)
```

```
[30, 40, 50]
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

[38]:
```
lst = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
# ind  0   1   2   3   4   5   6   7   8   9
new_list = lst[::] # ommitable
print(new_list)
```

```
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

[39]:
```
lst = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
# ind  0   1   2   3   4   5   6   7   8   9
new_list = lst[:5:] # ommitable
print(new_list)
```

```
[10, 20, 30, 40, 50]
```

[40]:
```
lst = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
# ind  0   1   2   3   4   5   6   7   8   9
new_list = lst[:8:2] # ommitable
print(new_list)
```

```
[10, 30, 50, 70]
```

[41]:
```
lst = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
# ind  0   1   2   3   4   5   6   7   8   9
new_list = lst[::2] # ommitable
print(new_list)
```

```
[10, 30, 50, 70, 90]
```

[42]:
```
lst = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
# ind  0   1   2   3   4   5   6   7   8   9
new_list = lst[::-1] # reverse the list
print(new_list)
```

```
[100, 90, 80, 70, 60, 50, 40, 30, 20, 10]
```

[43]:
```
lst = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
# ind  0   1   2   3   4   5   6   7   8   9
new_list = lst[:] # reverse the list
print(new_list)
```

```
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

[44]:
```
lst = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
# ind  0   1   2   3   4   5   6   7   8   9
new_list = lst[:3] # reverse the list
print(new_list)
```

```
[10, 20, 30]
```

[45]:
```
lst = [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
# ind  0   1   2   3   4   5   6   7   8   9
new_list = lst[5:] # reverse the list
print(new_list)
```

```
[60, 70, 80, 90, 100]
```

[51]:
```
# lst = [10, 20, 30, 40] # len --> even
# [10, 20], [30, 40]
# [20, 10], [40, 30]
# output_list = [20, 10, 40, 30]
lst = [10, 20, 30, 40, 50, 60] # n = 4
# ind  0   1   2   3   4   5
n = len(lst)
s1 = lst[:n//2]
s2 = lst[n//2:]
s1.reverse()
s2.reverse()
print(s1 + s2)
```

```
[30, 20, 10, 60, 50, 40]
```

[52]:
```
# lst = [10, 20, 30, 40] # len --> even
# [10, 20], [30, 40]
# [20, 10], [40, 30]
# output_list = [20, 10, 40, 30]
lst = [10, 20, 30, 40, 50, 60] # n = 4
# ind  0   1   2   3   4   5
n = len(lst)
s1 = lst[:n//2]
s2 = lst[n//2:]
print(abs(sum(s1) - sum(s2)))
```

```
90
```

[53]:
```
names = ['Rocky', 'Garuda', 'Vaanaram', 'Shanti', 'Adheera', 'Rina']
# ['Rocky', 'Vaanaram', 'Adheera']
print(names[::2])
```

```
['Rocky', 'Vaanaram', 'Adheera']
```

1.7 List Comprehensions

[57]:
```
ages = [31, 69, 8, 27, 49, 45, 35, 89,
        42, 6, 37, 25, 19, 45, 70, 2, 58,
        79, 21, 46, 92, 77, 45, 55, 41, 95, 21,
        79, 18, 97]
teen = []
old = []
for i in ages:
    if i >= 13 and i <= 19:
        teen.append(i)
print(teen)
for i in ages:
    if i > 50:
        old.append(i)
print(old)
```

```
[19, 18]
[69, 89, 70, 58, 79, 92, 77, 55, 95, 79, 97]
```

[58]:
```
# Generate a list of elements from 1 to 100
nums = []
for i in range(1, 101):
    nums.append(i)
print(nums) # [1, 2, 3, 4, 5, ..., 100]
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62,
63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82,
```

```
83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]
```

[59]:
```python
# Generate a list of multiples of 7 from 1 to 100
nums = []
for i in range(1, 101):
    if i % 7 == 0:
        nums.append(i)
print(nums) # [7, 14, 21, 28, ......, 98]
```

```
[7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98]
```

[60]:
```python
# Generate a list of multiples of 7 from 1 to 100
nums = []
for i in range(7, 101, 7):
    nums.append(i)
print(nums) # [7, 14, 21, 28, ......, 98]
```

```
[7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98]
```

[61]:
```python
# Generate a list of elements from 1 to 100
nums = [i for i in range(1, 101)] # List comprehension
print(nums) # [1, 2, 3, 4, 5, ..., 100]
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42,
43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62,
63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82,
83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100]
```

[63]:
```python
# Generate a list of multiples of 7 from 1 to 100
nums = [i for i in range(1, 101) if i % 7 == 0]
print(nums) # [7, 14, 21, 28, ......, 98]
```

```
[7, 14, 21, 28, 35, 42, 49, 56, 63, 70, 77, 84, 91, 98]
```

[66]:
```python
ages = [31, 69, 8, 27, 49, 45, 35, 89,
        42, 6, 37, 25, 19, 45, 70, 2, 58,
        79, 21, 46, 92, 77, 45, 55, 41, 95, 21,
        79, 18, 97]
teen = [i for i in ages if i >= 13 and i <= 19]
old = [i for i in ages if i > 50]
print(teen)
print(old)
```

```
[19, 18]
[69, 89, 70, 58, 79, 92, 77, 55, 95, 79, 97]
```

[68]:
```python
nums = [10, 20, 30, 40]
nums_squares = [i * i for i in nums]
print(nums_squares)
```

```
[100, 400, 900, 1600]
```

[70]:
```
cities = ['berlin', 'tokyo', 'palermo', 'nairobi', 'denver',
          'rio', 'lisbon', 'stockholm', 'bogota', 'helsinki']
#
new_cities = [i for i in cities if len(i) > 5]
print(new_cities)
```

```
['berlin', 'palermo', 'nairobi', 'denver', 'lisbon', 'stockholm', 'bogota',
'helsinki']
```

[71]:
```
cities = ['berlin', 'tokyo', 'palermo', 'nairobi', 'denver',
          'rio', 'lisbon', 'stockholm', 'bogota', 'helsinki']
lengths = [len(i) for i in cities]
print(lengths)
```

```
[6, 5, 7, 7, 6, 3, 6, 9, 6, 8]
```

[72]:
```
cities = ['berlin', 'tokyo', 'palermo', 'nairobi', 'denver',
          'rio', 'lisbon', 'stockholm', 'bogota', 'helsinki']
lengths = [max(i) for i in cities]
print(lengths)
```

```
['r', 'y', 'r', 'r', 'v', 'r', 's', 't', 't', 's']
```

[74]:
```
# Create list of all numbers from 1 to 300 that leaves a remainder 5 when␣
 ↪divided by 8
lst = [i for i in range(1, 301) if i % 8 == 5]
print(lst)
```

```
[5, 13, 21, 29, 37, 45, 53, 61, 69, 77, 85, 93, 101, 109, 117, 125, 133, 141,
149, 157, 165, 173, 181, 189, 197, 205, 213, 221, 229, 237, 245, 253, 261, 269,
277, 285, 293]
```

[76]:
```
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True

primes = [i for i in range(1, 100) if is_prime(i) == True]
print(primes)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73,
79, 83, 89, 97]
```

[79]:
```
def is_palin(n: int) -> bool:
    s = str(n)
```

```
    if s == s[::-1]:
        return True
    else:
        return False

palindromes = [i for i in range(1, 200) if is_palin(i) == True]
print(palindromes)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 22, 33, 44, 55, 66, 77, 88, 99, 101, 111, 121,
131, 141, 151, 161, 171, 181, 191]
```

[4]:
```
x = [i for i in range(10, 101) if abs(i%10 - i//10) == 1]
print(x)
```

```
[10, 12, 21, 23, 32, 34, 43, 45, 54, 56, 65, 67, 76, 78, 87, 89, 98]
```

[6]:
```
lst = [i for i in range(100, 201) if i % 10 == 6 or i % 10 == 7]
print(lst)
```

```
[106, 107, 116, 117, 126, 127, 136, 137, 146, 147, 156, 157, 166, 167, 176, 177,
186, 187, 196, 197]
```

[15]:
```
def digit_sum(n):
    s = 0
    for i in str(n): # '123'
        s += int(i)
    return s

lst = [i for i in range(250, 351) if digit_sum(i) == 8]
print(lst)
```

```
[251, 260, 305, 314, 323, 332, 341, 350]
```

[17]:
```
# list
# string
lst = [10, 20, 30, 40, 50, 60]
st1 = 'python'
print(len(lst), len(st1))
print(lst[1], st1[1])
```

```
6 6
20 y
```

[18]:
```
lst = [10, 20, 30, 40, 50, 60]
for i in lst: # element based
    print(i)
```

```
10
20
30
```

```
40
50
60
```

[19]:
```
string = 'python'
for i in string: # element (character) based access
    print(i)
```

```
p
y
t
h
o
n
```

[20]:
```
lst = [10, 20, 30, 40, 50, 60]
for i in range(len(lst)): # element based
    print(lst[i])
```

```
10
20
30
40
50
60
```

[22]:
```
string = 'python'
for i in range(len(string)):
    print(string[i])
```

```
p
y
t
h
o
n
```

[24]:
```
lst = [10, 20, 30, 40, 50, 60]
string = 'python'
print(lst[1:5])
print(string[1:5])
```

```
[20, 30, 40, 50]
ytho
```

[26]:
```
string = 'python'
          #012345
for i in range(len(string)-1, -1, -1):
    print(string[i])
```

```
n
o
h
t
y
p
```

[ ]:

[27]:
```
cities = ['berlin', 'tokyo', 'palermo', 'nairobi', 'denver',
          'rio', 'lisbon', 'stockholm', 'bogota', 'helsinki']
new_list = [i for i in cities if len(i) <= 5]
print(new_list)
```

```
['tokyo', 'rio']
```

[30]:
```
cities = ['berlin', 'tokyo', 'palermo', 'nairobi', 'denver',
          'rio', 'lisbon', 'stockholm', 'bogota', 'helsinki']
new_list = [i for i in cities if max(i) == 'r']
print(new_list)
```

```
['berlin', 'palermo', 'nairobi', 'rio']
```

[33]:
```
cities = ['berlin', 'tokyo', 'palermo', 'nairobi', 'denver',
          'rio', 'lisbon', 'stockholm', 'bogota', 'helsinki']
new_list = [i for i in cities if i[-1] == 'o']
print(new_list)
```

```
['tokyo', 'palermo', 'rio']
```

[32]:

[32]:
```
'n'
```

[37]:
```
lst = []
for i in range(1, 4):
    for j in range(1, 4):
        lst.append((i, j))
print(lst)
```

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

[38]:
```
lst = [(i, j) for i in range(1, 4) for j in range(1, 4)]
print(lst)
```

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

[39]:
```
lst = [(i, j) for i in range(1, 4) for j in range(1, 4) if i + j == 4]
print(lst)
```

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

[40]:
```
lst = [(i, j) for i in range(1, 4) for j in range(1, 4) if i + j == 4 and i !=␣
↪j]
print(lst)
```

```
[(1, 3), (3, 1)]
```

[42]:
```
lst = []
for i in range(1, 4):
    for j in range(1, 4):
        for k in range(1, 4):
            lst.append((i, j, k))
print(lst)
```

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

[43]:
```
lst = [(i, j, k) for i in range(1, 4) for j in range(1, 4) for k in range(1, 4)]
print(lst)
```

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

1.8 Nested lists and Matrices

[48]:
```
marks = [[25, 55, 75, 89, 64],
         [55, 78, 47, 97, 31],
         [27, 89, 53, 32, 46],
         [96, 16, 69, 77, 28]]
print(len(marks))
print(marks[2])
```

```
4
[27, 89, 53, 32, 46]
```

[49]:
```
marks = [[25, 55, 75, 89, 64],
         [55, 78, 47, 97, 31],
         [27, 89, 53, 32, 46],
         [96, 16, 69, 77, 28]]
print(marks[2][3])
```

```
32
```

[51]:
```
#   0 1 2
# 0 1 2 3
# 1 4 5 6
# 2 7 8 9
```

```
mat = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
print(mat[1][1])
```

```
5
```

[52]:
```
# Accessing elements in nested lists
marks = [[25, 55, 75, 89, 64],
         [55, 78, 47, 97, 31],
         [27, 89, 53, 32, 46],
         [96, 16, 69, 77, 28]]
for i in marks:
    print(i)
```

```
[25, 55, 75, 89, 64]
[55, 78, 47, 97, 31]
[27, 89, 53, 32, 46]
[96, 16, 69, 77, 28]
```

[53]:
```
# Accessing elements in nested lists
marks = [[25, 55, 75, 89, 64],
         [55, 78, 47, 97, 31],
         [27, 89, 53, 32, 46],
         [96, 16, 69, 77, 28]]
for i in marks:
    print(sum(i))
```

```
308
308
247
286
```

[54]:
```
# Accessing elements in nested lists
marks = [[25, 55, 75, 89, 64],
         [55, 78, 47, 97, 31],
         [27, 89, 53, 32, 46],
         [96, 16, 69, 77, 28]]
for i in marks:
    print(max(i))
```

```
89
97
89
96
```

[56]:
```
# Accessing elements in nested lists
marks = [[25, 55, 75, 89, 64],
         [55, 78, 47, 97, 31],
         [27, 89, 53, 32, 46],
```

```
          [96, 16, 69, 77, 28]]
for i in range(len(marks)):
    print(marks[i])
```

```
[25, 55, 75, 89, 64]
[55, 78, 47, 97, 31]
[27, 89, 53, 32, 46]
[96, 16, 69, 77, 28]
```

[57]:
```
# Accessing elements in nested lists
marks = [[25, 55, 75, 89, 64],
         [55, 78, 47, 97, 31],
         [27, 89, 53, 32, 46],
         [96, 16, 69, 77, 28]]
for i in range(len(marks)): # i = 0 1 2 3
    for j in range(len(marks[i])): # j = 2 3 4
        print(marks[i][j], end = ' ') # marks[0][2]
    print()
```

```
25 55 75 89 64
55 78 47 97 31
27 89 53 32 46
96 16 69 77 28
```

1.8.1 Reading a 2D-list from user as input

- Matrix Reading
- Matrix Element Accessing

[ ]:
```
r = 3, c =
1 2 3 -> Row1 -> 1D list
4 5 6 -> Row2 -> 1D list
7 8 9 -> Row3 -> 1D list
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

[2]:
```
mat = []
for i in range(3):
    lst = list(map(int, input().split()))
    mat.append(lst)
print(mat)
```

```
1 2 3
4 5 6
7 8 9
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

[4]:
```
# matrix reading
r, c = map(int, input().split())
mat = []
```

```
for i in range(r):
    lst = list(map(int, input().split()))
    mat.append(lst)
print(mat)
```

```
4 3
1 2 3
4 5 6
7 8 9
10 11 12
[[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12]]
```

[5]:
```
# matrix reading
r, c = map(int, input().split())
mat = []
for i in range(r):
    lst = list(map(int, input().split()))
    mat.append(lst)
print(mat)

# matrix accessing
for i in range(r):
    for j in range(c):
        print(mat[i][j], end=" ")
    print()
```

```
3 5
1 2 3 4 5
6 7 8 9 0
1 2 3 4 5
[[1, 2, 3, 4, 5], [6, 7, 8, 9, 0], [1, 2, 3, 4, 5]]
1 2 3 4 5
6 7 8 9 0
1 2 3 4 5
```

[7]:
```
n = int(input())
mat = []
for i in range(n):
    lst = list(map(int, input().split()))
    mat.append(lst)
print(mat)

for i in range(n):
    for j in range(n):
        print(mat[i][j], end = ' ')
    print()
```

```
3
1 2 3
```

```
4 5 6
7 8 9
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
1 2 3
4 5 6
7 8 9
```

[12]:
```python
# Principal Diagonal
n = int(input())
# mat = [list(map(int, input().split())) for i in range(n)]
mat = [[int(i) for i in input().split()] for j in range(n)]
# Diagonal
for i in range(n):
    for j in range(n):
        if i == j:
            print(mat[i][j], end = ' ')
    print()
```

```
3
1 2 3
4 5 6
7 8 9
3
5
7
```

[ ]:
```python
# Secondary Diagonal
n = int(input())
# mat = [list(map(int, input().split())) for i in range(n)]
mat = [[int(i) for i in input().split()] for j in range(n)]
# Diagonal
for i in range(n):
    for j in range(n):
        if i + j == n - 1:
            print(mat[i][j], end = ' ')
    print()
```

[7]:
```python
x = [1, 2, 3, 4, 5]
print(*x[5:])
```

[8]:
```python
from random import *
mat = [[randint(-20, 20) for  i in range(4)] for j in range(4)]
for i in range(len(mat)):
    for j in range(len(mat[i])):
        print(mat[i][j], end = ' ')
    print()
```

```
-8 11 -4 -13
15 9 -5 -2
-14 3 -2 12
10 -8 -18 5
```

[17]:
```
mat = [[4, 11, -4, -13],
    [15, 9, -5, -2],
    [-14, 3, -2, 12],
    [10, -8, -18, 5]] # mat = [[list(map(int, input().split()))] for i in␣
 ↪range(4)]
print(mat)
n = len(mat)
sums = []
for i in range(1, n - 1):
    for j in range(1, n - 1):
        s = mat[i-1][j-1] + mat[i-1][j+1] + mat[i+1][j-1] + mat[i+1][j+1]
        sums.append(s)
print(max(sums))
```

```
[[4, 11, -4, -13], [15, 9, -5, -2], [-14, 3, -2, 12], [10, -8, -18, 5]]
13
```