LIST

While going for a Shopping ….!!!???

Imagine you have the scores in “Python Programming” for 100 students. If you want to find the average score in Python…?

Simple Statistics

- Many programs deal with large collections of similar information.
  - Words in a document
  - Students in a course
  - Data from an experiment
  - Customers of a business
  - Graphics objects drawn on the screen
  - Cards in a deck

Indexing, Slicing

```
>>> L = ['spam', 'Spam', 'SPAM!']

>>> L[2]              # Offsets start at zero

'SPAM!'

>>> L[−2]             # Negative: count from the right

'Spam'

>>> L[1:]             # Slicing fetches sections

['Spam', 'SPAM!']
```

Insertion, Deletion and Replacement

```
>>> L = [1, 2, 3]
>>> L[1:2] = [4, 5]        # Replacement/insertion
>>> L
[1, 4, 5, 3]
>>> L[1:1] = [6, 7]    # Insertion (replace nothing)
>>> L
[1, 6, 7, 4, 5, 3]
>>> L[1:2] = []            # Deletion (insert nothing)
>>> L
[1, 7, 4, 5, 3]
```

Insertion, Deletion and Replacement

```
>>> L = [1]

>>> L[:0] = [2, 3, 4]
# Insert all at :0, an empty slice at front
>>> L
[2, 3, 4, 1]
>>> L[len(L):] = [5, 6, 7]
 # Insert all at len(L):, an empty slice at end
>>> L
[2, 3, 4, 1, 5, 6, 7]
```

List method calls

```
>>> L = ['eat', 'more', 'SPAM!']

>>> L.append('please')

# Append method call: add item at end

>>> L

['eat', 'more', 'SPAM!', 'please']

>>> L.sort()                    # Sort list items ('S' < 'e')

>>> L

['SPAM!', 'eat', 'more', 'please']
```

More on Sorting Lists

```
>>> L = ['abc', 'ABD', 'aBe']

>>> L.sort()                          # Sort with mixed case

>>> L

['ABD', 'aBe', 'abc']

>>> L = ['abc', 'ABD', 'aBe']

>>> L.sort(key=str.lower)             # Normalize to lowercase

>>> L

['abc', 'ABD', 'aBe']
```

More on Sorting Lists

```
>>> L.sort(key=str.lower, reverse=True)

# Change sort order

>>> L ['aBe', 'ABD', 'abc']
```

Other common list methods

```
>>> L = [1, 2]

>>> L.extend([3, 4, 5])

# Add many items at end (like in-place +)

>>> L [1, 2, 3, 4, 5]

>>> L.pop()

# Delete and return last item

5
```

Other common list methods

```
>>> L [1, 2, 3, 4]

>>> L.reverse()                    # In-place reversal method

>>> L [4, 3, 2, 1]

>>> list(reversed(L))

# Reversal built-in with a result (iterator)

[1, 2, 3, 4]
```

Other common list methods

```
>>> L = ['spam', 'eggs', 'ham']
>>> L.index('eggs')       # Index of an object (search/find)
1
>>> L.insert(1, 'toast') # Insert at position
>>> L
['spam', 'toast', 'eggs', 'ham']
>>> L.remove('eggs')         # Delete by value
>>> L
['spam', 'toast', 'ham']
```

Other common list methods

```
>>> L.pop(1)                    # Delete by position 'toast'

>>> L

['spam', 'ham']

>>> L.count('spam')             # Number of occurrences 1

1
```

Other common list methods

```
>>> L = ['spam', 'eggs', 'ham', 'toast']
>>> del L[0]                          # Delete one item
>>> L ['eggs', 'ham', 'toast']
>>> del L[1:]                         # Delete an entire section
>>> L                                 # Same as L[1:] = []
['eggs‘]
```

Other common list methods

```
>>> L = ['Already', 'got', 'one']

>>> L[1:] = []

>>> L

['Already']

>>> L[0] = []

>>> L

[[]]
```

List

- A list is a sequence of items stored as a single object.
- Items in a list can be accessed by indexing, and sub lists can be accessed by slicing.
- Lists are mutable; individual items or entire slices can be replaced through assignment statements.
- Lists support a number of convenient and frequently used methods.
- Lists will grow and shrink as needed.

Other methods

- `max(list)`
  - returns the maximum element from the list

```
>>>L = [34,45,12]
>>>L1 = ['ab','xc','df']
>>>L2 = ['ab','xc',1]
>>>max(L)
  45
>>>max(L1)
  'xc'
>>>max(L2)
Error!!
```

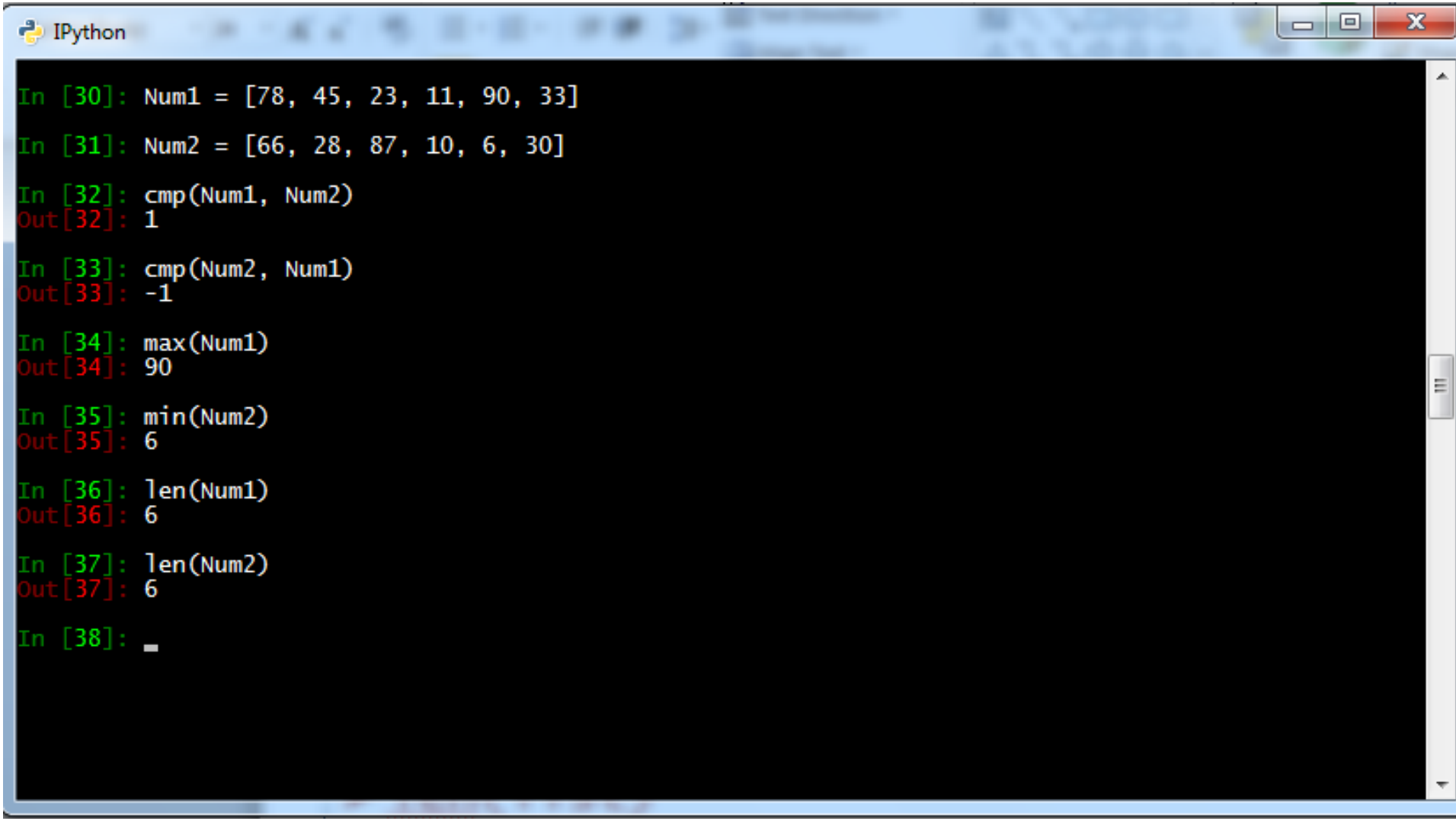
Other methods

- `min(list)`
  - returns the minimum element from the list

```
>>>L = [34,45,12]
>>>min(L)
12
```

- `len(list)`
  - returns the number of items in the list

```
>>> len(L)
3
```


IPython
In [30]: Num1 = [78, 45, 23, 11, 90, 33]
In [31]: Num2 = [66, 28, 87, 10, 6, 30]
In [32]: cmp(Num1, Num2)
Out[32]: 1
In [33]: cmp(Num2, Num1)
Out[33]: -1
In [34]: max(Num1)
Out[34]: 90
In [35]: min(Num2)
Out[35]: 6
In [36]: len(Num1)
Out[36]: 6
In [37]: len(Num2)
Out[37]: 6
In [38]:

Strings and Lists

```
>>> S = 'spammy'
>>> L = list(S)
>>> L
['s', 'p', 'a', 'm', 'm', 'y']
>>> L[3] = 'x'                    # Works for lists, not strings
>>> L[4] = 'x‘
>>> L
['s', 'p', 'a', 'x', 'x', 'y']
>>> S = ''.join(L)      #uses ‘’ for joining elements of list
>>> S
'spaxxy‘
```

Strings and Lists

```
>>> 'SPAM'.join(['eggs', 'sausage', 'ham', 'toast'])
'eggsSPAMsausageSPAMhamSPAMtoast‘
# uses ‘SPAM’ for joining elements of list
>>> line = 'aaa bbb  ccc'
>>> cols = line.split()
>>> cols
['aaa', 'bbb', 'ccc']
```

Statistics using List

Find the mean, standard deviation on a set of numbers

Examples

- Apple, Banana, Berry, Mango
- Football, Basketball, Throwball, Tennis, Hockey
- Sunrise, Sugar, Cheese, Butter, Pickle, Soap, Washing Powder, Oil….
- Agra, Delhi, Kashmir, Jaipur, Kolkata…

Introduction

- Contains multiple values that are logically related
- List is a type of mutable sequence in Python
- Each element of a list is assigned a number – index / position
- Can do indexing, slicing, adding, multiplying, and checking for membership
- Built-in functions for finding length of a sequence and for finding its largest and smallest elements

What is a List?

- Most versatile data type in Python
- Comma-separated items can be collected in square brackets
- Good thing is..
  - THE ITEMS IN THE LIST NEED NOT BE OF SAME TYPE

Creating a list

- Creating an EMPTY list

  listname = []

Example:

L1 = []

MyList = []

Books = []

- Creating a list with items

listname = [item1, item2, ….]

Example:

Temp = [100, 99.8, 103, 102]

S = [‘15BIT0001’, ‘Achu’, 99.9]

L2 = [1, 2, 3, 4, 5, 6, 7]

Course = [‘Python’, ‘C’, ‘C++’, ‘Java’]

Accessing Values

- Using index or indices

```
>>>L1 = [1, 2, 3, 4, 5, 6]
>>>print (L1[3]) #indexing
>>>4
>>>print (L1[2:5]) #slicing
>>>[3, 4, 5]
```

Updating Elements

- Update an element in list using index

```
>>>L1 = [1, 2, 3, 4, 5, 6]
>>>L1[2] = 111
>>>L1
    [1, 2, 111, 4, 5, 6]
```

Deleting Elements

- Delete an element in list using index

```
>>>L1 = [1, 2, 3, 4, 5, 6]
>>>del L1[4]
>>>L1
    [1, 2, 3, 4, 6]
```

Reading Elements

```
Lst=[]
N=int(input(“Enter N: “))
for i in range(1,N):
    lst.append(int(input("Enter the number")))
print(lst)
```

Basic Operations in List

- >>> len([1, 2, 3]) # Length

  3

- >>> [1, 2, 3] + [4, 5, 6] # Concatenation

  [1, 2, 3, 4, 5, 6]

- >>> ['Ni!'] * 4 # Repetition

  ['Ni!', 'Ni!', 'Ni!', 'Ni!']

Basic Operations in List

- >>> str([1, 2]) + "34" # Same as "[1, 2]" + "34"
  '[1, 2]34'

- >>> [1, 2] + list("34")
  # Same as [1, 2] + ["3", "4"]
  [1, 2, '3', '4‘]

List Iteration

- >>> 3 in [1, 2, 3] # Membership

  True

- >>> for x in [1, 2, 3]:

  print(x, end=' ')

Iteration (2.X uses: print x,) ... 1 2 3

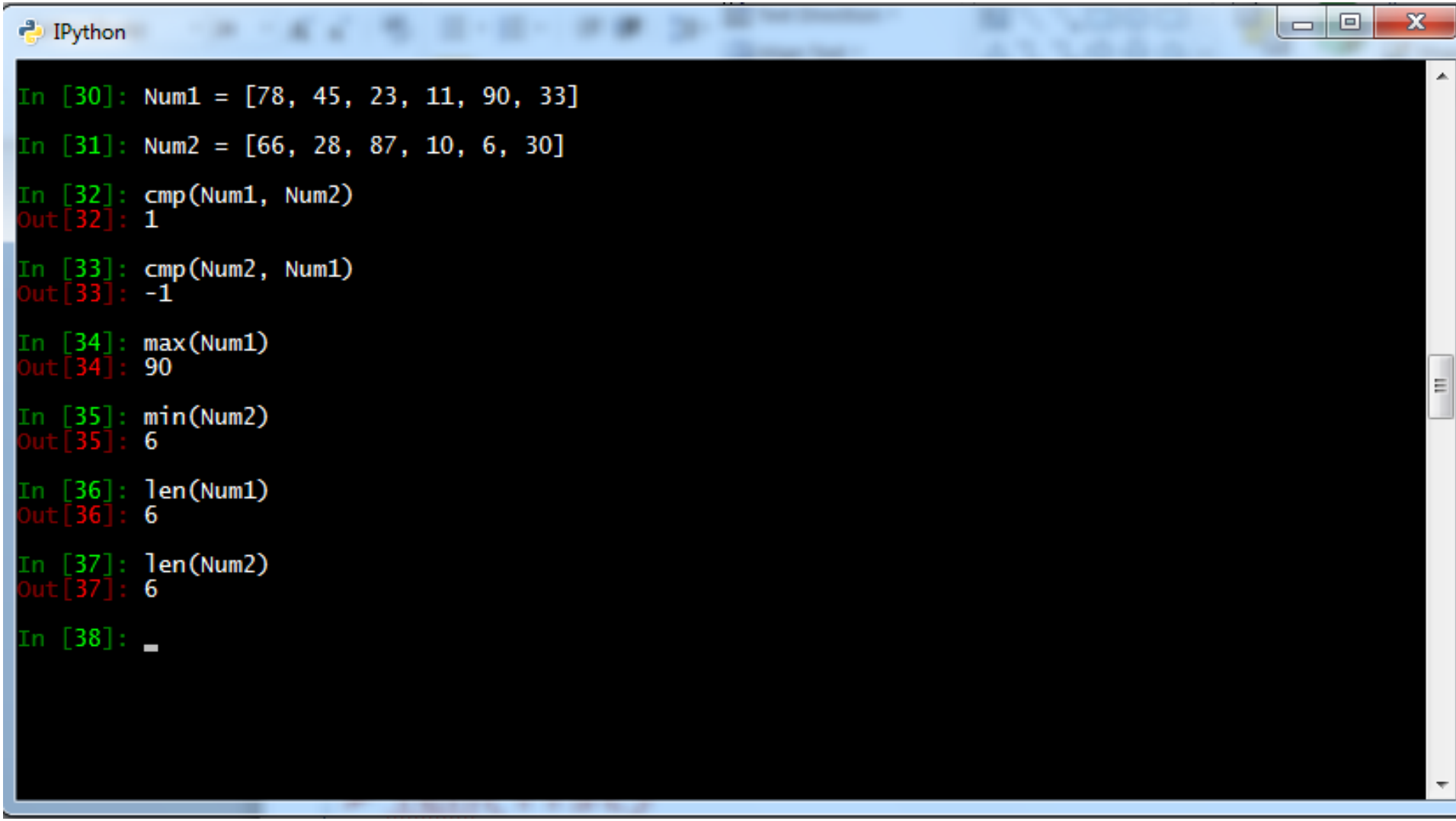
Table 8-1. Common list literals and operations

Operation	Interpretation
`L = []`	An empty list
`L = [123, 'abc', 1.23, {}]`	Four items: indexes 0..3
`L = ['Bob', 40.0, ['dev', 'mgr']]`	Nested sublists
`L = list('spam')`	List of an iterable's items, list of successive integers
`L = list(range(-4, 4))`	
`L[i]`	Index, index of index, slice, length
`L[i][j]`	
`L[i:j]`	
`len(L)`	
`L1 + L2`	Concatenate, repeat

Operation	Interpretation
`L * 3`	
`for x in L: print(x)`	Iteration, membership
`3 in L`	
`L.append(4)`	Methods: growing
`L.extend([5,6,7])`	
`L.insert(i, X)`	
`L.index(X)`	Methods: searching
`L.count(X)`	
`L.sort()`	Methods: sorting, reversing,
`L.reverse()`	copying (3.3+), clearing (3.3+)
`L.copy()`	
`L.clear()`	
`L.pop(i)`	Methods, statements: shrinking
`L.remove(X)`	
`del L[i]`	
`del L[i:j]`	
`L[i:j] = []`	
`L[i] = 3`	Index assignment, slice assignment
`L[i:j] = [4,5,6]`	
`L = [x**2 for x in range(5)]`	List comprehensions and maps (Chapter 4, Chapter 14, Chapter 20)
`list(map(ord, 'spam'))`	

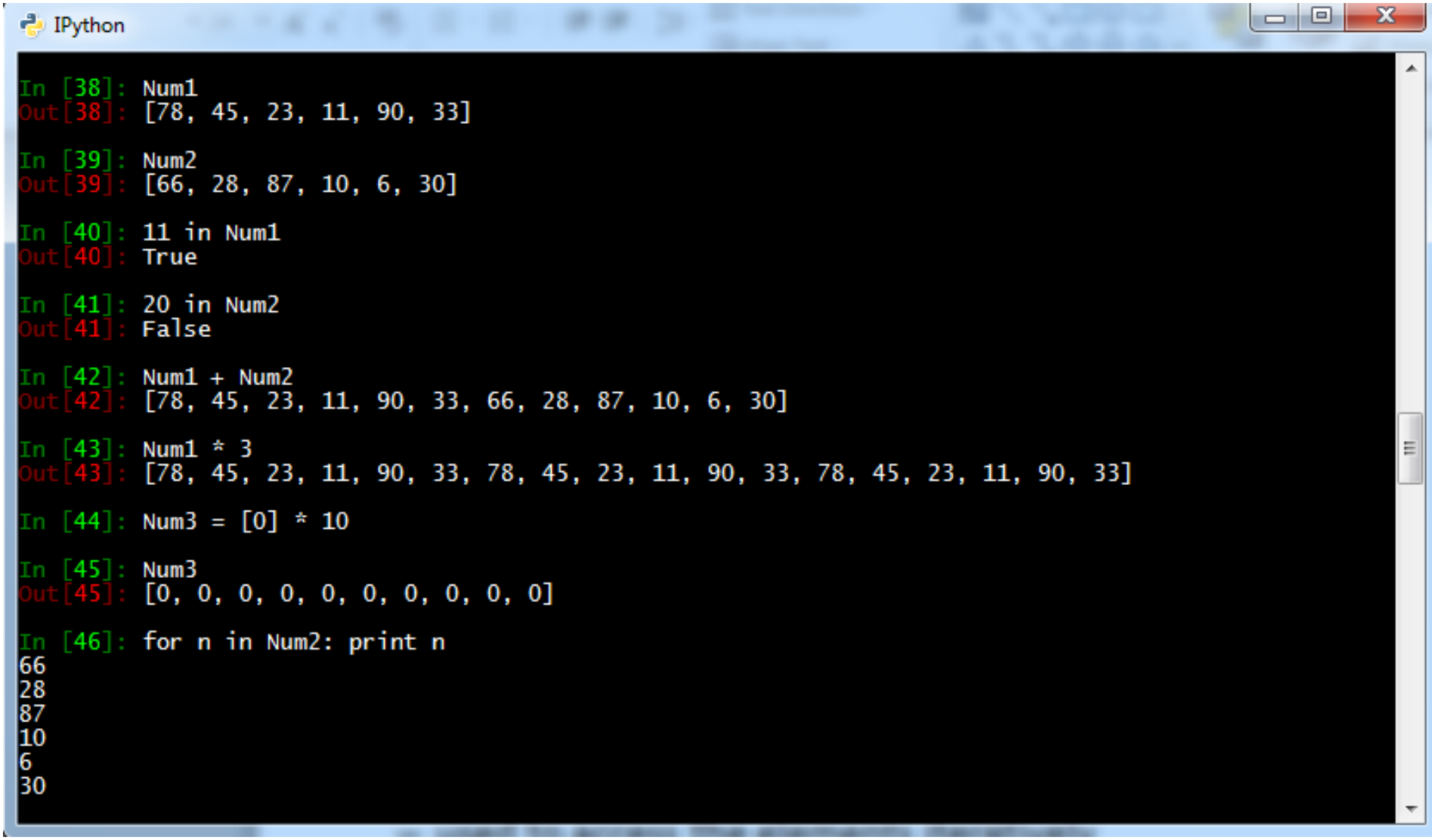
Other methods

- `cmp(list1, list2)`
  - compares elements of both list1 and list2
- `max(list)`
  - returns the maximum element from the list
- `min(list)`
  - returns the minimum element from the list
- `len(list)`
  - returns the number of items in the list


IPython
In [30]: Num1 = [78, 45, 23, 11, 90, 33]
In [31]: Num2 = [66, 28, 87, 10, 6, 30]
In [32]: cmp(Num1, Num2)
Out[32]: 1
In [33]: cmp(Num2, Num1)
Out[33]: -1
In [34]: max(Num1)
Out[34]: 90
In [35]: min(Num2)
Out[35]: 6
In [36]: len(Num1)
Out[36]: 6
In [37]: len(Num2)
Out[37]: 6
In [38]:

Other methods

- `in (membership)`
  - returns True if the item is in the list
- `+ (concatenation)`
  - merges two lists
- `* (repetition)`
  - repeats the list elements specified number of times
- `for (iteration)`
  - used to access the elements iteratively


IPython
In [38]: Num1
Out[38]: [78, 45, 23, 11, 90, 33]
In [39]: Num2
Out[39]: [66, 28, 87, 10, 6, 30]
In [40]: 11 in Num1
Out[40]: True
In [41]: 20 in Num2
Out[41]: False
In [42]: Num1 + Num2
Out[42]: [78, 45, 23, 11, 90, 33, 66, 28, 87, 10, 6, 30]
In [43]: Num1 * 3
Out[43]: [78, 45, 23, 11, 90, 33, 78, 45, 23, 11, 90, 33, 78, 45, 23, 11, 90, 33]
In [44]: Num3 = [0] * 10
In [45]: Num3
Out[45]: [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
In [46]: for n in Num2: print n
66
28
87
10
6
30

Strings and Lists

```
>>> S = 'spammy'
>>> L = list(S)
>>> L ['s', 'p', 'a', 'm', 'm', 'y']
>>> L[3] = 'x'                    # Works for lists, not strings
>>> L[4] = 'x‘
>>> L ['s', 'p', 'a', 'x', 'x', 'y']
>>> S = ''.join(L) #uses ‘’ for joining elements of list
>>> S
'spaxxy‘
```

- Given the marks of N students in a class, write a program to find the class average.

- Given a list of integer values, find the count of positive numbers, negative numbers and zeroes in the list.

```
lst=[]
countpos=0
countzero=0
countneg=0
N=int(input(“Enter N: “))
for i in range(1,N):
    lst.append(int(input("Enter the number")))
for n in lst:
    if(n==0):
        countzero+=1
    elif(n>0):
        countpos+=1
    else:
        countneg+=1
print("Number of Zero %d" %(countzero))
print("Number of positive numbers %d" %(countpos))
print("Number of negative numbers %d"  %(countneg))
```

Read the names of N students and display the name list in alphabetical order.

```
names=[]
for i in range(1,4):
    names.append(input("Enter the name"))
print("The name list")
print(names)
print("The name list in alphabetical order")
names.sort()
print(names)
names.reverse()
print("The reversed name list")
print(names)
```

List Comprehensions

```
>>> res = [c * 4 for c in 'SPAM']
# List comprehensions
>>> res
['SSSS', 'PPPP', 'AAAA', 'MMMM']
```

- expression is functionally equivalent to a for loop that builds up a list of results manually
- list comprehensions are simpler to code and likely faster to run today:

List Comprehensions

```
# List comprehension equivalent ...
>>> res = []
>>> for c in 'SPAM':
        res.append(c * 4)
>>> res
['SSSS', 'PPPP', 'AAAA', 'MMMM']
```

- res=[ord(c) for c in "span"]
- Lst=[65,67,49,97,99]

  res =[chr(c) for c in Lst]
- list1=[0 for j in range(col)]
- MatrixA=[[0 for j in range(col)] for i in range(row)]

```
N=5
L1=[int(input()) for i in range(N)]
print(L1)
```

Matrices

- a basic 3 × 3 two-dimensional list-based array:
  >>> matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
- With one index, you get an entire row (really, a nested sublist), and with two, you get an item within the row:

```
>>> matrix[1]
[4, 5, 6]
```

Matrixes

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

Matrix Addition

```
row=int(input("enter the no of rows :"))
col=int(input("enter the no of cols :"))
print("Enter the first matrix");
list1=[[int(input()) for j in range(col)] for i in range(row)]
print("Enter the second matrix");
list2=[[int(input()) for j in range(col)] for i in range(row)]
print(list1)
print(list2)
summat=[[list1[i][j]+list2[i][j] for j in range(col)] for i in
    range(row)]
print(summat)
```

- Watson gives Sherlock a list of *N numbers*. Then he asks him to determine if there exists an element in the list such that the sum of the elements on its left is equal to the sum of the elements on its right. If there are no elements to the left/right, then the sum is considered to be zero.

```
lst=[10,20,30,10,10,10]
N=len(lst)
for i in range(N)):
    if(sum(lst[0:i])==sum(lst[i+1:N])):
        print("Element = %d" %(lst[i]))
        break

else:
    print("Not found")
```

- Sunny and Johnny together have *M* dollars they want to spend on ice cream. The parlor offers *N* flavors, and they want to choose two flavors so that they end up spending the whole amount.

You are given the cost of these flavors. The cost of the *ith* flavor is denoted by *ci*. You have to display the indices of the two flavors whose sum is *M*.

```
c=[10,20,30,10,10,10]
M=30
for i in range(len(c)-1):
    for j in range((i+1),len(c)):
                if(c[i]+c[j]==M):
                    print("Flavors %d and %d add upto %d"
%(i,j,M))
```

- In a supermarket there are two sections S1 and S2. The sales details of item_1 to item_n of section1 and item_1 to item_p of section2 are maintained in a sorted order. Write a program to merge the elements of the two sorted lists to form the consolidated list.

```
L1=[]
L2=[]
N=int(input("Enter the no of item in list1"))
M=int(input("Enter the no of item in list2"))
for i in range(1,N+1):
    L1.append(int(input("Enter the item in list1")))

for i in range(1,M+1):
    L2.append(int(input("Enter the item in list2")))

L1.extend(L2)
L1.sort()
print(L1)
```

```
para=input("Enter the text")
words=para.split()
count=0
for w in words:
    if w.lower().strip()=="the":
        count+=1
print(count)
```

- Given a positions of coins of player1 and player2 in a 3X3 Tic Tac Toc board, write a program to determine if the board position is a solution and if so identify the winner of the game. In a Tic Tac Toc problem, if the coins in a row or column or along a diagonal is of the same player then he has won the game. Assume that player1 uses ‘1’ as his coin and player2 uses '2' as his coin. '0' in the board represent empty cell.

```
board=[[0 for col in range(3)] for row in range(3)]
winner=0
print("Enter the first matrix");
#enter '0' if the position is empty, '1' if it contains the coin of player1 and '2' if it contains coin of player2

for row  in range(3):
   for col in range(3):
      board[row][col]=int(input("enter (0/1/2) :"))
#scan row by row to check if the coins in the row are of same player
for row1 in range(0,3):
   if (board[row1][0]==1 and board[row1][1]==1 and board[row1][2]==1):
      winner = 1
   elif (board[row1][0]==2 and board[row1][1]==2 and board[row1][2]==2):
      winner = 2

#scan column by column to check if the coins in the column are of same player
for col1 in range(0,3):
   if (board[0][col1]==1 and board[1][col1]==1 and board[2][col1]==1):
      winner = 1
   elif (board[0][col1]==2 and board[1][col1]==2 and board[2][col1]==2):
      winner = 2
```

```
#scan diagonal1 to check if the coins in the column are of same player
if (board[0][0]==1 and board[1][1]==1 and board[2][2]==1):
        winner = 1
    elif (board[0][0]==2 and board[1][1]==2 and board[2][2]==2):
        winner = 2

#scan diagonal2 to check if the coins in the column are of same player
if (board[0][2]==1 and board[1][1]==1 and board[2][0]==1):
        winner = 1
    elif (board[0][2]==2 and board[1][1]==2 and board[2][0]==2):
        winner = 2

print(winner)
```