Lists

Lists

- The most basic data structure in Python is the sequence. Each element of a sequence is assigned a number - its position or index. The first index is zero, the second index is one, and so forth.
- Python has six built-in types of sequences, but the most common ones are lists and tuples, which we would see in this tutorial.
- There are certain things you can do with all sequence types. These operations include indexing, slicing, adding, multiplying, and checking for membership. In addition, Python has built-in functions for finding the length of a sequence and for finding its largest and smallest elements.

Lists

- They are mutable data type
- The list is a most versatile data type available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.
- Creating a list is as simple as putting different comma-separated values between square brackets.

Lists

simple.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/16/simple.py (3.4.4)

File Edit Format Run Options Window Help

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]

print(list1)
print(list2)
print(list3)
```

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 1
tel)] on win32
Type "copyright", "credits" or "license()" for mo
>>>
 RESTART: C:/Documents and Settings/admin/Desktop
e.py
['physics', 'chemistry', 1997, 2000]
[1, 2, 3, 4, 5]
['a', 'b', 'c', 'd']
```

Lists

A list is simply a sequence of values stored in a specific order with each value identified by its position in that order.

So for an example consider the list of names of the elements up to uranium.

Lists

hydrogen, helium, lithium, beryllium, …, protactinium, uranium

A *sequence* of values	The names of the elements
Values stored in order	Atomic number order
Individual value identified by position in the sequence	“helium” is the name of the second element

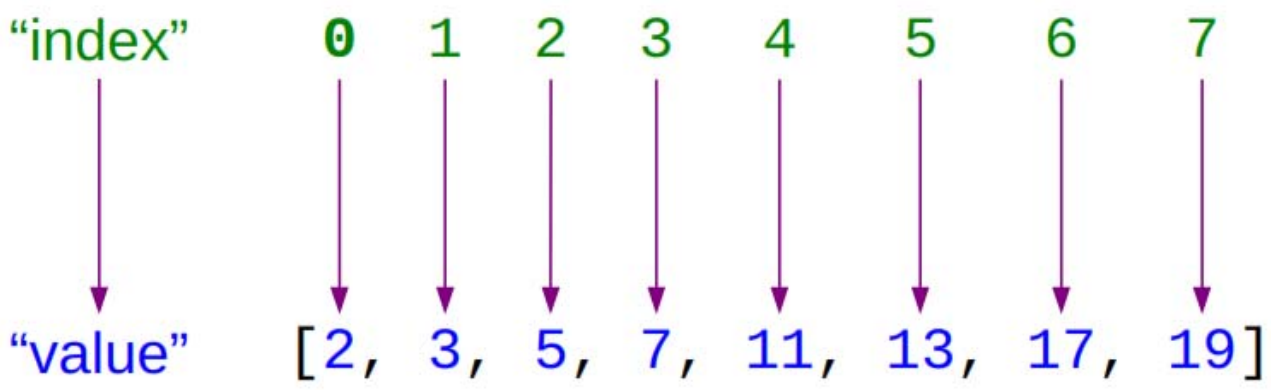
Lists

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59

A *sequence* of values	The prime numbers less than sixty
Values stored in order	Numerical order
Individual value identified by position in the sequence	7 is the fourth prime

Accessing Values in Lists

- To access values in lists, use the square brackets for slicing along with the index or indices to obtain value available at that index.



Accessing Values in Lists

simple.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/16/simple.py (3.4

File Edit Format Run Options Window Help

```
book = ['physics', 'chemistry', 1997, 2000];
prime = [2, 3, 5, 7, 11, 13, 17, 19 ];

print ("book[0]: ", book[0])
print ("prime[1:5]: ", prime[1:5])
```

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015,
tel)] on win32
Type "copyright", "credits" or "license()" for
>>>
 RESTART: C:/Documents and Settings/admin/Deskt
e.py
book[0]:  physics
prime[1:5]:  [3, 5, 7, 11]
```

Counting from the end

```
primes = [ 2, 3, 5, 7, 11, 13, 17, 19]
```

0	1	2	3	4	5	6	7
↓	↓	↓	↓	↓	↓	↓	↓
[2,	3,	5,	7,	11,	13,	17,	19]
↑	↑	↑	↑	↑	↑	↑	↑
-8	-7	-6	-5	-4	-3	-2	-1

simple.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/16/simple.py (3.

File Edit Format Run Options Window Help

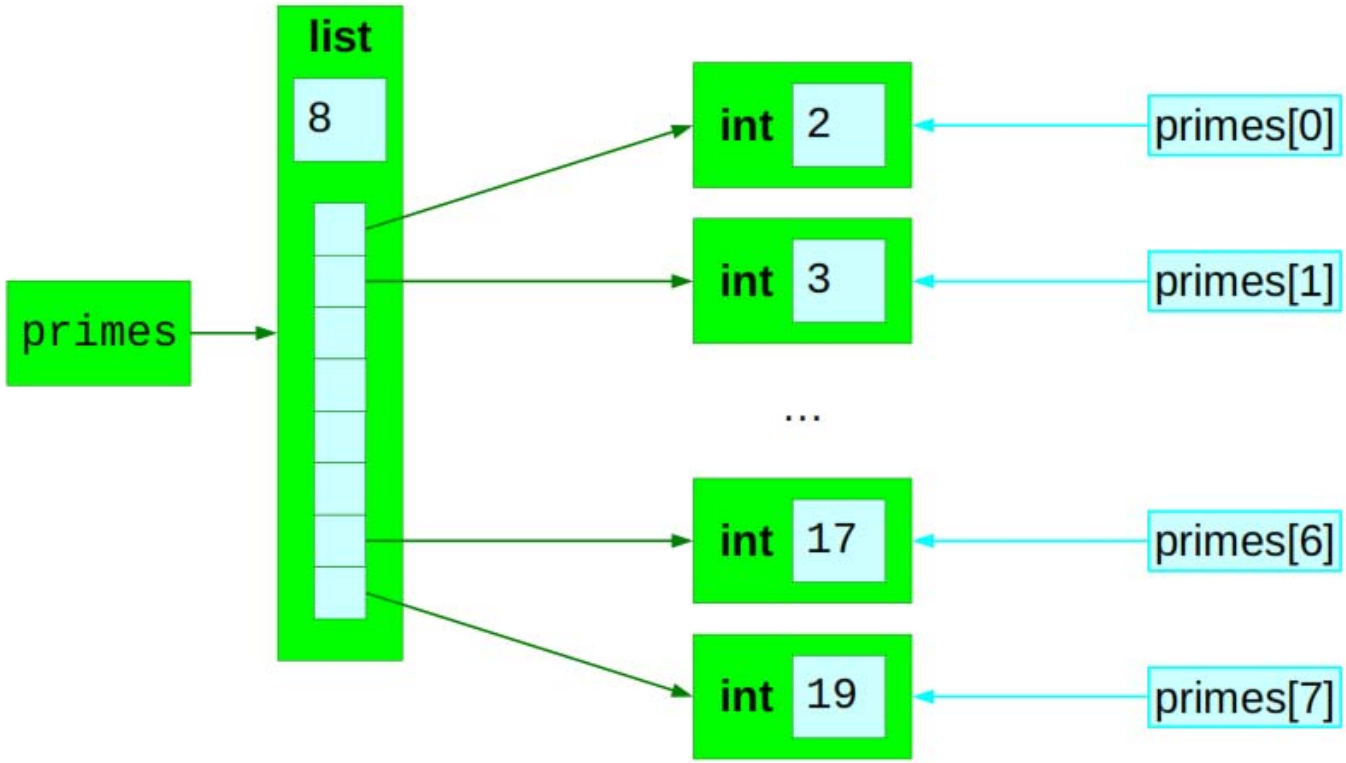
```
book = ['physics', 'chemistry', 1997, 2000];
prime = [2, 3, 5, 7, 11, 13, 17, 19 ];

print ("book[0]: ", book[0])
print ("prime[1:5]: ", prime[1:5])
print ("prime[1:5]: ", prime[-5:-3])
```

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015
tel)] on win32
Type "copyright", "credits" or "license()" for
>>>
 RESTART: C:/Documents and Settings/admin/Desk
e.py
book[0]:  physics
prime[1:5]:  [3, 5, 7, 11]
prime[1:5]:  [7, 11]
```

Inside view of a list



Updating Lists

- You can update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator, and you can add to elements in a list with the append() method..

Updating Lists

simple.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/16/simple.py (3.4.4)

File Edit Format Run Options Window Help

```
list1 = ['physics', 'chemistry', 1997, 2000];

print ("Value available at index 2 : ", end="")
print (list1[2])
list1[2] = 2001;
print ("New value available at index 2 : ", end="")
print (list1[2])
```

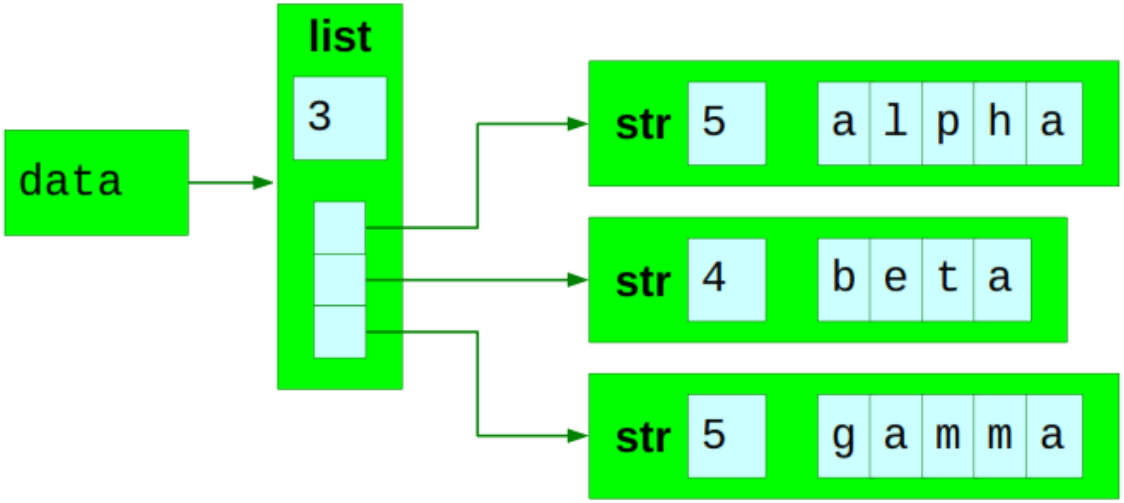
Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:1
tel)] on win32
Type "copyright", "credits" or "license()" for more in
>>>
 RESTART: C:/Documents and Settings/admin/Desktop/intr
e.py
Value available at index 2 : 1997
New value available at index 2 : 2001
```

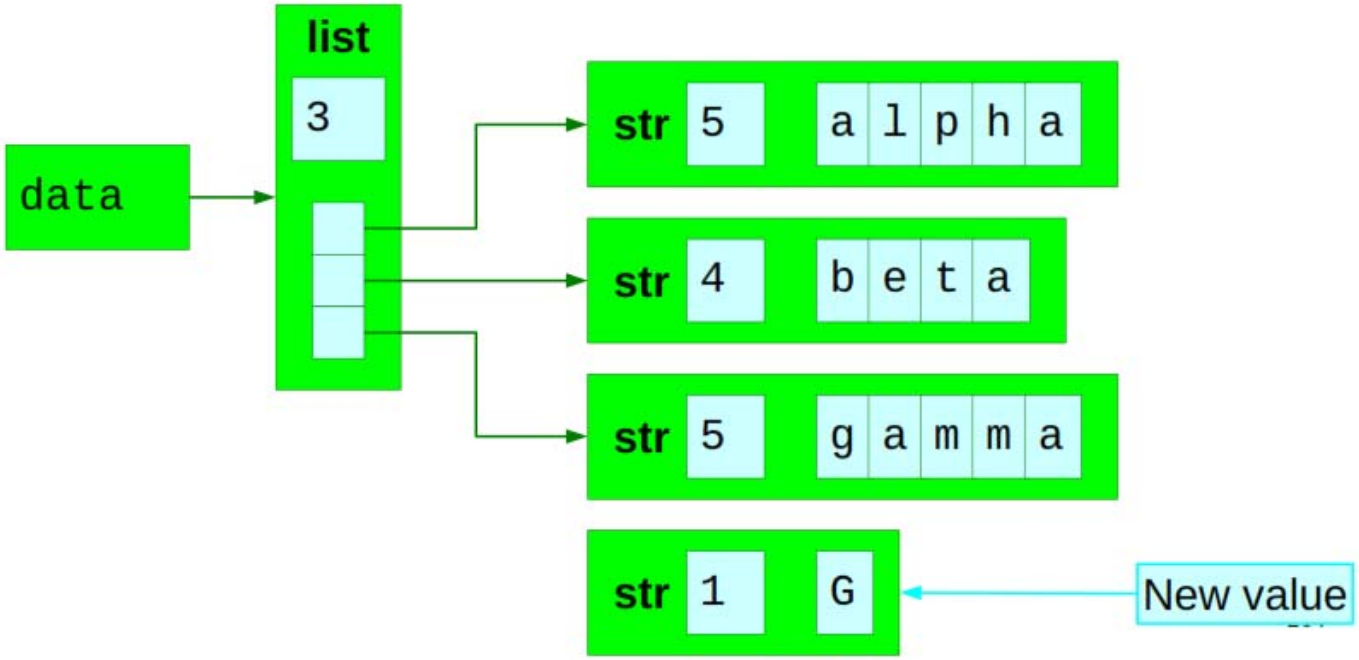
Updating Lists

```
data = ['alpha', 'beta', 'gamma']
```

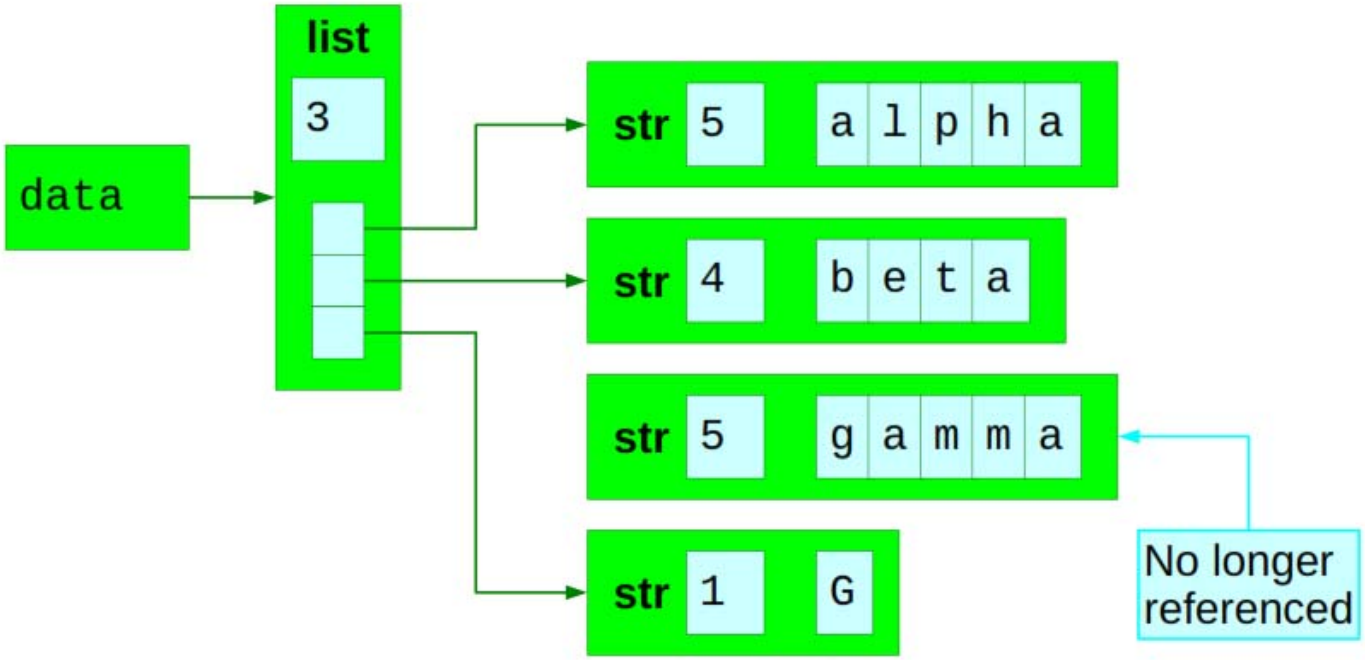


Updating Lists

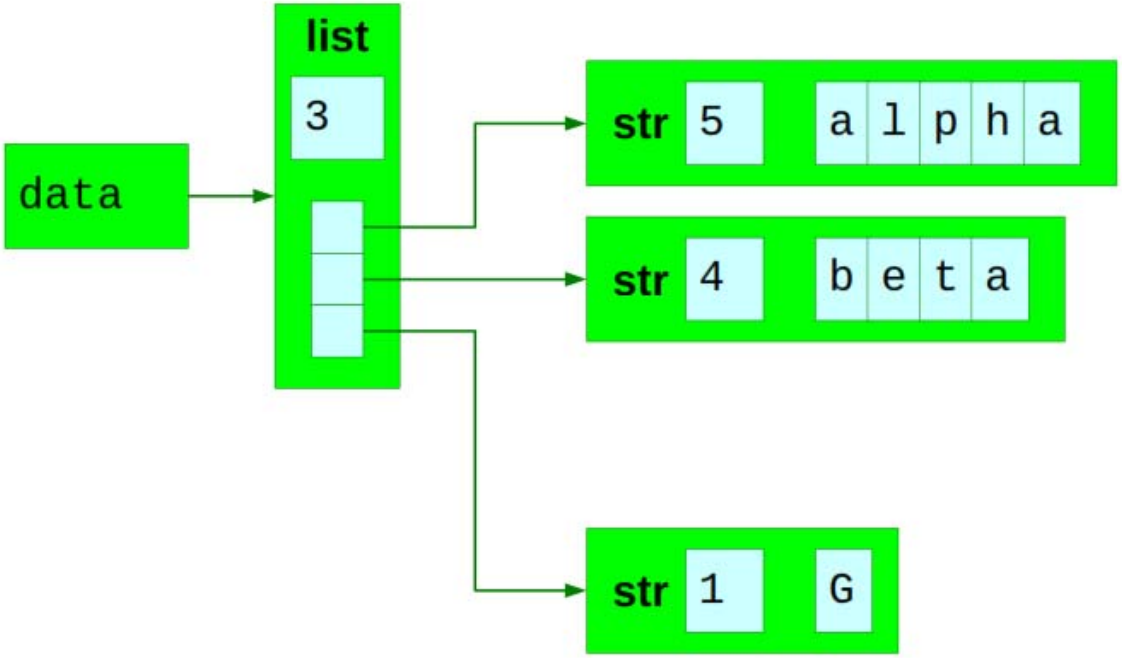
```
data[2] = 'G'
```



Updating Lists



Updating Lists



Delete List Elements

- To remove a list element, you can use either the del statement if you know exactly which element(s) you are deleting or the remove() method if you do not know.

Delete List Elements

simple.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/16/simple.py (3.4.4)

File Edit Format Run Options Window Help

```
list1 = ['physics', 'chemistry', 1997, 2000]


print (list1)
del (list1[2])
print ("After deleting value at index 2 : ")
print (list1)
```

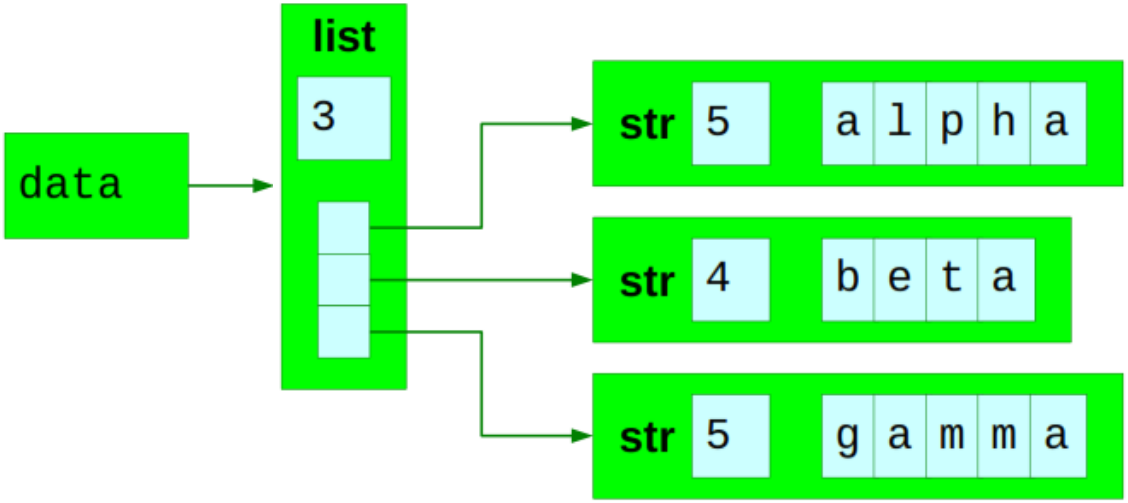
Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:1
tel)] on win32
Type "copyright", "credits" or "license()" for more in
>>>
 RESTART: C:/Documents and Settings/admin/Desktop/intr
e.py
['physics', 'chemistry', 1997, 2000]
After deleting value at index 2 :
['physics', 'chemistry', 2000]
```

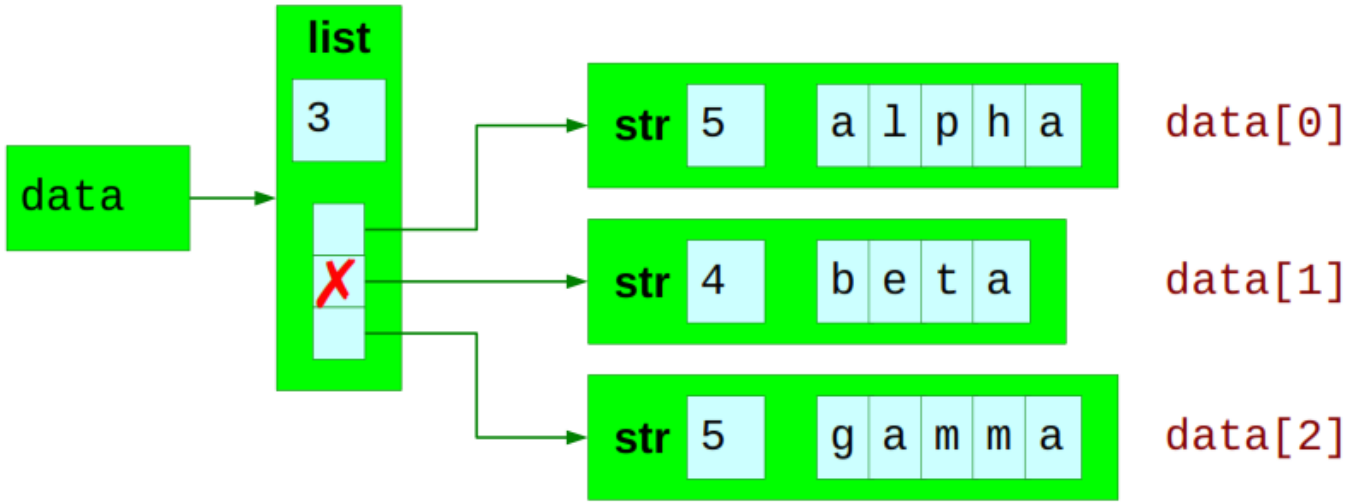
Delete List Elements

```
data = ['alpha', 'beta', 'gamma']
```

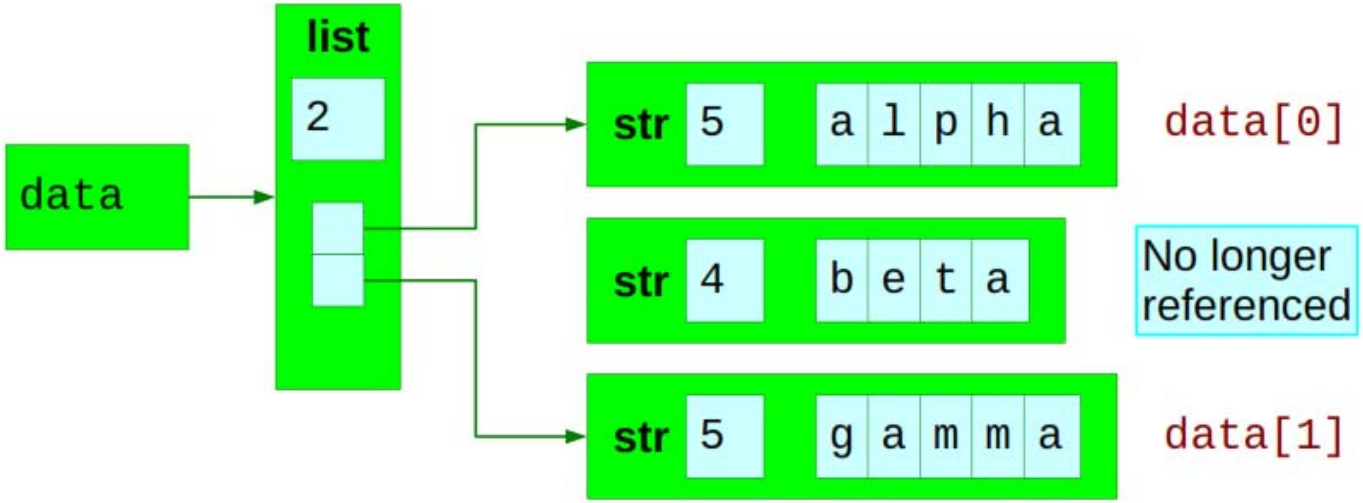


Delete List Elements

```
del data[1]
```



Delete List Elements



Operators for lists

- List operators are:
- +, *
- slicing [:]
- **in** and **not in**

Python Expression	Results	Description
[1, 2, 3] + [4, 5, 6]	[1, 2, 3, 4, 5, 6]	Concatenation
['Hi!'] * 4	['Hi!', 'Hi!', 'Hi!', 'Hi!']	Repetition
3 in [1, 2, 3]	True	Membership
for x in [1, 2, 3]: print x,	1 2 3	Iteration

List operators

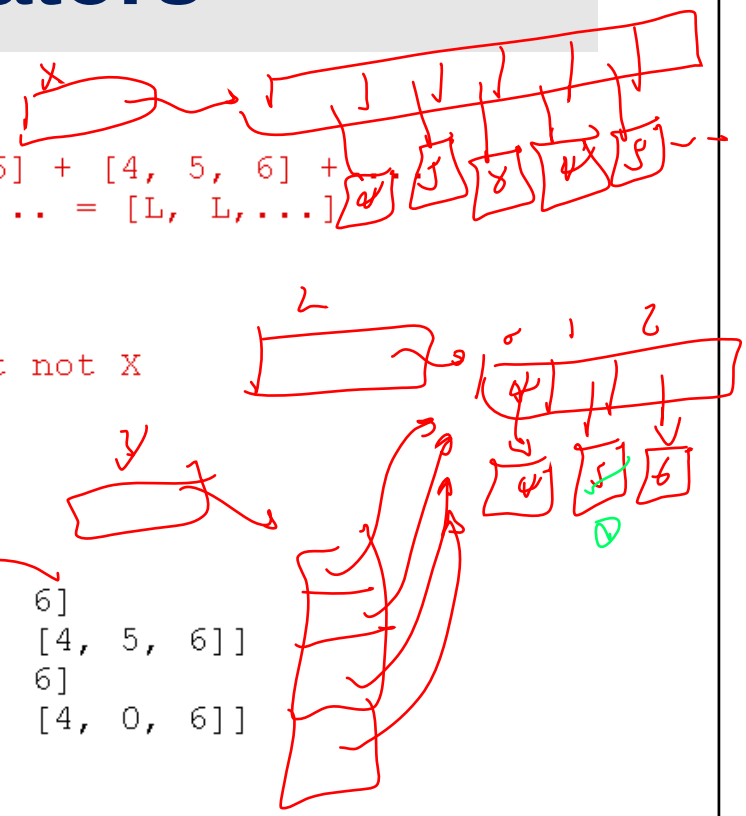
listrep.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\15-list\listrep.py (3.4.4)

File Edit Format Run Options Window Help

```
La=[1, 2, 3, 'A']
Lb=[4, 5, 6, 'B']
X=La + Lb
print(X)
if 'B' in X:
    print('B is in X')

#output

    [1, 2, 3, 'A', 4, 5, 6, 'B']
    B is in X
```

List operators

```
L = [4, 5, 6]
X = L * 4          # Like [4, 5, 6] + [4, 5, 6] + ...
Y = [L] * 4        # [L] + [L] + ... = [L, L,...]
print(X)
print(Y)

L[1] = 0           # Impacts Y but not X
print(X)
print(Y)

#output:
[4, 5, 6, 4, 5, 6, 4, 5, 6, 4, 5, 6]
[[4, 5, 6], [4, 5, 6], [4, 5, 6], [4, 5, 6]]
[4, 5, 6, 4, 5, 6, 4, 5, 6, 4, 5, 6]
[[4, 0, 6], [4, 0, 6], [4, 0, 6], [4, 0, 6]]
```



List operators

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\16\simple.py (3.4.4)

File Edit Format Run Options Window Help

```
L = [1, 2, 3]
L[1:2] = [4, 5]                    # Replacement/insertion
print(L)
L[1:1] = [6, 7]                    # Insertion (replace nothing)
print(L)
L[1:2] = []                        # Deletion (insert nothing)
print(L)
L = [1]
L[:0] = [2, 3, 4]          # Insert all at :0, an empty slice at front
print(L)
L[len(L):] = [5, 6, 7]     # Insert all at len(L):, an empty slice at end
print(L)
print()
```

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:18) [MSC v.1600 32
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
 RESTART: C:\Documents and Settings\admin\Desktop\intro-python\examples\
e.py
[1, 4, 5, 3]
[1, 6, 7, 4, 5, 3]
[1, 7, 4, 5, 3]
[2, 3, 4, 1]
[2, 3, 4, 1, 5, 6, 7]
```

List operators

- **Access item by item**

```
L = [1, 2, 3, 4]
while L:
    front, L = L[0], L[1:]
    print(front, L)

#output:

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

List operators

- **Variable index**

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]

i=0
print(list1[i])
i=i+2
print(list1[i])
print()
```

```
physics
1997
```

List comparing Operator

- The standard comparisons (<, <=, >, >=, ==, !=) apply to lists. These comparisons use the standard item-by-item comparison rules.

simpletuple.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/19/simpletuple

File Edit Format Run Options Window Help

```
l1=[1, [1,4], [3, 5, 9, 11], 19]
l2=[1, [1,4], [3, 5, 7, 11], 19]
if (l1>l2):
    print('List 1 is greater than List 2')

#output
    List 1 is greater than List 2
```

built-in functions for lists

SN	Function with Description
	len(list) Gives the total length of the list.
	max(list) Returns item from the list with max value.
	min(list) Returns item from the list with min value.

built-in functions for lists

simple.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/16/simple.py (3.4.4)

File Edit Format Run Options Window Help

```
list1, list2 = ['xyz', 'zara', 'abc'], [456, 700, 200]

print ("Max value element : ", max(list1))
print ("Min value element : ", min(list2))
```

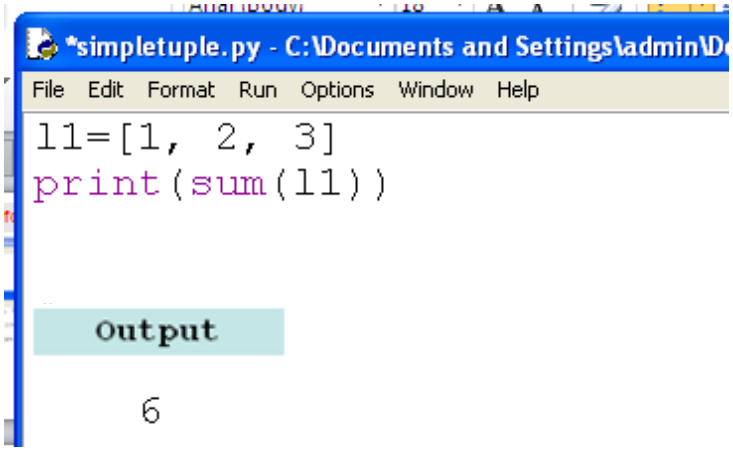
Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 1
tel)] on win32
Type "copyright", "credits" or "license()" for mo
>>>
 RESTART: C:/Documents and Settings/admin/Desktop
e.py
Max value element :  zara
Min value element :  200
```

Sum

- Retrun the sum of all elements in the list.



All

- Return True if all elements of the list are true (or if the list is empty).

*simpletuple.py - /home/nowzari/Desktop/python/python-my/python/ex

File Edit Format Run Options Window Help

```
list1 = [1, 123, 18, 'str', 21]
r = all(list1)
print(r)

# Output
  True


list1 = [1, 123, 0, 'str', 21]
r = all(list1)
print(r)

# Output
   False
```

0 → false

Any

- Return True if any element of the list is true. If the list is empty, return False.

```
list1 = [0, 1, 0, 0, 0]
r = any(list1)
print(r)

# Output
  True


list1 = [0, 0, 0, 0, 0]
r = any(list1)
print(r)

# Output
   False
```

built-in functions for lists

*simple.py - /home/nowzari/Desktop/python/python-my/python/examples/14-list/simple

File Edit Format Run Options Window Help

```
line = 'aaa,bbb,ccccc,dd   \n'
print(line)
linenew=line.split(',')
print(linenew)

line = 'aaa,bbb,ccccc,dd     \n'
# rstrip Remove whitespace characters
# on the right side
linenew=line.rstrip().split(',')     # Combine two operations
print(linenew)

>>>output

aaa,bbb,ccccc,dd

['aaa', 'bbb', 'ccccc', 'dd   \n']
['aaa', 'bbb', 'ccccc', 'dd']
```

built-in functions for lists

```
line = 'aaa,bbb,ccccc,dd     \n'
print('string: ', line)
linenew=line.rstrip().split(',')     # Combine two operations
print('list: ', linenew)
line=','.join(linenew)
print('string: ', line)
```

```
string:  aaa,bbb,ccccc,dd

list:  ['aaa', 'bbb', 'ccccc', 'dd']
string:  aaa,bbb,ccccc,dd
```

built-in functions for lists

```
s1='abc'
s2='123'
l1=s1.join(s2)
l2=s2.join(s1)
print('s1 join s2: ', l1)
print('s2 join s1: ', l2)
```

```
s1 join s2:  1abc2abc3
s2 join s1:  a123b123c
```

Type Conversion functions

simple.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/16/simple.py (3

File Edit Format Run Options Window Help

```
tstr="string"
tlist=list(tstr)
print(tlist)
```

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015
tel)] on win32
Type "copyright", "credits" or "license()" for
>>>
 RESTART: C:/Documents and Settings/admin/Desk
e.py
['s', 't', 'r', 'i', 'n', 'g']
```

Type Conversion functions

str(x): convert x into a string

```
L=[1,2,5]
M=str(L)
print(M[0])
```

"[1,2,5]"

[

Type specific functions for lists

SN	Methods with Description
1	**list.append(obj)** Appends object obj to list
2	**list.count(obj)** Returns count of how many times obj occurs in list
3	**list.extend(seq)** Appends the contents of seq to list
4	**list.index(obj)** Returns the lowest index in list that obj appears
5	**list.insert(index, obj)** Inserts object obj into list at offset index

Type specific functions for lists

6	**list.pop(obj=list[-1])** Removes and returns last object or obj from list
7	**list.remove(obj)** Removes object obj from list
8	**list.reverse()** Reverses objects of list in place
9	**list.sort([func])** Sorts objects of list, use compare func if given

```
menu_item = 0
namelist = []
while menu_item != 9:
    print("--------------------")
    print("1. Print the list")
    print("2. Add a name to the list")
    print("3. Remove a name from the list")
    print("4. Change an item in the list")
    print("9. Quit")
    menu_item =int(input("Pick an item from the menu: "))
    if menu_item == 1:
        current = 0
        if len(namelist) > 0:
            while current < len(namelist):
                print(current, ".", namelist[current])
                current = current + 1
        else:
            print("List is empty")
    elif menu_item == 2:
        name =input("Type in a name to add: ")
        namelist.append(name)
    elif menu_item == 3:
        del_name = input("What name would you like to remove: ")
        if del_name in namelist:
            # namelist.remove(del_name) would work just as fine
            item_number = namelist.index(del_name)
            del namelist[item_number]
            # The code above only removes the first occurrence of
            # the name.  The code below from Gerald removes all.
            # while del_name in namelist:
            #       item_number = namelist.index(del_name)
            #       del namelist[item_number]
        else:
            print(del_name, "was not found")
```

```
    elif menu_item == 3:
        del_name = input("What name would you like to remove: ")
        if del_name in namelist:
            # namelist.remove(del_name) would work just as fine
            item_number = namelist.index(del_name)
            del namelist[item_number]
            # The code above only removes the first occurrence of
            # the name.  The code below from Gerald removes all.
            # while del_name in namelist:
            #       item_number = namelist.index(del_name)
            #       del namelist[item_number]
        else:
            print(del_name, "was not found")
    elif menu_item == 4:
        old_name =input("What name would you like to change: ")
        if old_name in namelist:
            item_number = namelist.index(old_name)
            new_name =input("What is the new name: ")
            namelist[item_number] = new_name
        else:
            print(old_name, "was not found")

print("Goodbye")
```

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:18
tel)] on win32
Type "copyright", "credits" or "license()" for more info
>>>
 RESTART: C:/Documents and Settings/admin/Desktop/intro-
.py
--------------------
1. Print the list
2. Add a name to the list
3. Remove a name from the list
4. Change an item in the list
9. Quit
Pick an item from the menu: 1
List is empty
--------------------
1. Print the list
2. Add a name to the list
3. Remove a name from the list
4. Change an item in the list
9. Quit
Pick an item from the menu: 2
Type in a name to add: abbas
--------------------
1. Print the list
2. Add a name to the list
3. Remove a name from the list
4. Change an item in the list
9. Quit
Pick an item from the menu: 1
0 . abbas
--------------------
1. Print the list
2. Add a name to the list
3. Remove a name from the list
4. Change an item in the list
9. Quit
Pick an item from the menu: 2
```

```
--------------------
1. Print the list
2. Add a name to the list
3. Remove a name from the list
4. Change an item in the list
9. Quit
Pick an item from the menu: 2
Type in a name to add: hassan
--------------------
1. Print the list
2. Add a name to the list
3. Remove a name from the list
4. Change an item in the list
9. Quit
Pick an item from the menu: 2
Type in a name to add: akbar
--------------------
1. Print the list
2. Add a name to the list
3. Remove a name from the list
4. Change an item in the list
9. Quit
Pick an item from the menu: 2
Type in a name to add: sari
--------------------
1. Print the list
2. Add a name to the list
3. Remove a name from the list
4. Change an item in the list
9. Quit
Pick an item from the menu: 1
0 . abbas
1 . hassan
2 . akbar
3 . sari
--------------------
```

```
--------------------
1. Print the list
2. Add a name to the list
3. Remove a name from the list
4. Change an item in the list
9. Quit
Pick an item from the menu: 4
What name would you like to change: akbar
What is the new name: ali
--------------------
1. Print the list
2. Add a name to the list
3. Remove a name from the list
4. Change an item in the list
9. Quit
Pick an item from the menu: 1
0 . abbas
1 . hassan
2 . ali
3 . sari
--------------------
1. Print the list
2. Add a name to the list
3. Remove a name from the list
4. Change an item in the list
9. Quit
Pick an item from the menu: |
```

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\16\simple.py (3.4.4)

File Edit Format Run Options Window Help

```
aList = ['xyz', 'zara', 'abc'];
bList = ['ali', 'manni'];
aList.extend(bList)
print ("Extended List : ", aList)
aList.insert( 3, 'akbar')
print ("Final List : ", aList)

print ("A List : ", aList.pop())
print ("B List : ", aList.pop(2))
print ("Final List : ", aList)

aList.remove('xyz');
print ("List : ", aList)

aList.append('kamran')
aList.reverse();
print ("Revese List : ", aList)
aList.append('yashar')
aList.sort();
print ("List : ", aList)
```

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Type "copyright", "credits" or "license()" for more informatic
>>>
 RESTART: C:\Documents and Settings\admin\Desktop\intro-pythor
e.py
Extended List :  ['xyz', 'zara', 'abc', 'ali', 'manni']
Final List :  ['xyz', 'zara', 'abc', 'akbar', 'ali', 'manni']
A List :  manni
B List :  abc
Final List :  ['xyz', 'zara', 'akbar', 'ali']
List :  ['zara', 'akbar', 'ali']
Revese List :  ['kamran', 'ali', 'akbar', 'zara']
List :  ['akbar', 'ali', 'kamran', 'yashar', 'zara']
```

Type specific functions for lists

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\16\simple.py (3.4.4)

File Edit Format Run Options Window Help

```
greek = ['alpha', 'gamma', 'delta', 'beta']
sg=sorted(greek)
print(sg)
print(greek)
print()
```

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:1
tel)] on win32
Type "copyright", "credits" or "license()" for more in
>>>
 RESTART: C:\Documents and Settings\admin\Desktop\intr
e.py
['alpha', 'beta', 'delta', 'gamma']
['alpha', 'gamma', 'delta', 'beta']
```

Array

- A list which all elements have a specific type is called array.
- A one dimensional array is called Vector
- int array: aint=[1, 3, 4, 7, 9]
- String array: stra=[‘as’, ‘pe’, ‘dk’]

Multidimensional list

- **We can define a multidimensional list.**

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\16\simple.py (3.4.4)

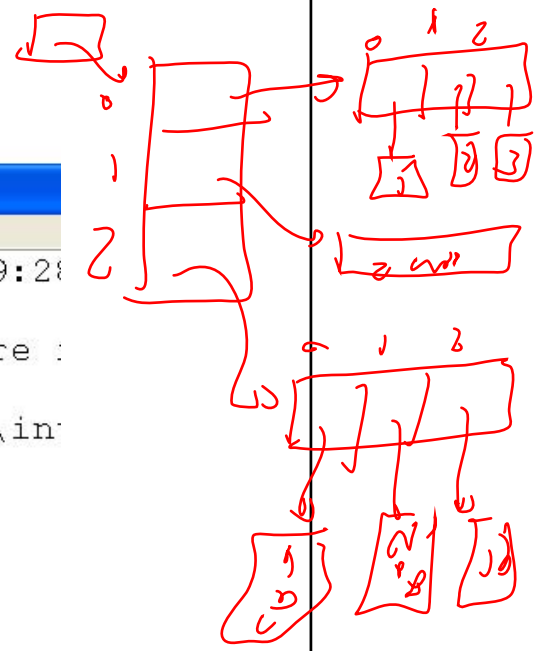
File Edit Format Run Options Window Help

```
aList = [[1, 2, 3], 'zara', ['abc', 'def', 'gh']];
print(aList)
print(aList[0][1])
print(aList[2][1])
```

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:2
tel)] on win32
Type "copyright", "credits" or "license()" for more
>>>
 RESTART: C:\Documents and Settings\admin\Desktop\in
e.py
[[1, 2, 3], 'zara', ['abc', 'def', 'gh']]
2
def
```



Multidimensional list

- simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\16\simple.py (3.4.4)

File Edit Format Run Options Window Help

```
aList = [[1, 2, 3], 'zara', ['abc', 'def', 'gh']];
print(aList)
aList[0][1]=4
print(aList)
```

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
>>>
 RESTART: C:\Documents and Settings\admin\Desktop\intr
e.py
[[1, 2, 3], 'zara', ['abc', 'def', 'gh']]
[[1, 4, 3], 'zara', ['abc', 'def', 'gh']]
```

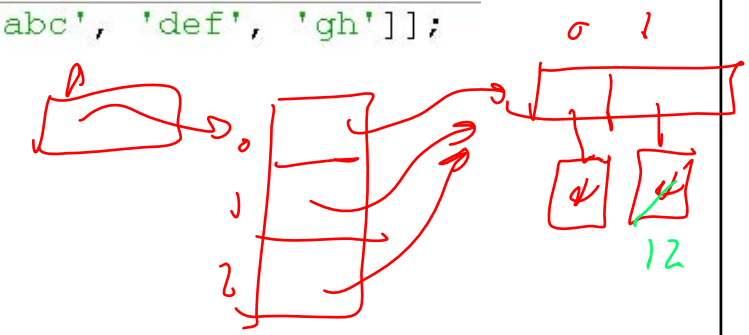
Multidimensional list

- W

simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\16\simple.py (3.4.4)

File Edit Format Run Options Window Help

```
aList = [[1, 2, 3], 'zara', ['abc', 'def', 'gh']];
print(aList)
aList[0][1]=4
print(aList)

A = [[4] * 2] * 3
print(A)
A[2][1]=12
print(A)
```



Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
>>>
 RESTART: C:\Documents and Settings\admin\Desktop\in
e.py
[[1, 2, 3], 'zara', ['abc', 'def', 'gh']]
[[1, 4, 3], 'zara', ['abc', 'def', 'gh']]
[[4, 4], [4, 4], [4, 4]]
[[4, 12], [4, 12], [4, 12]]
```

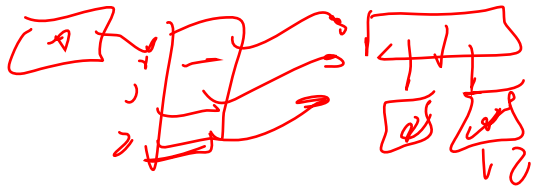
Multidimensional list

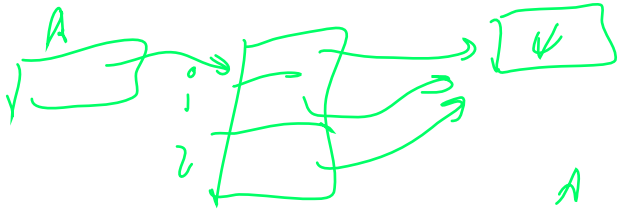
- The reason is that replicating a list with * doesn't create copies, it only creates references to the existing objects. The *3 creates a list containing 3 references to the same list of length two. Changes to one row will show in all rows, which is almost certainly not what you want.

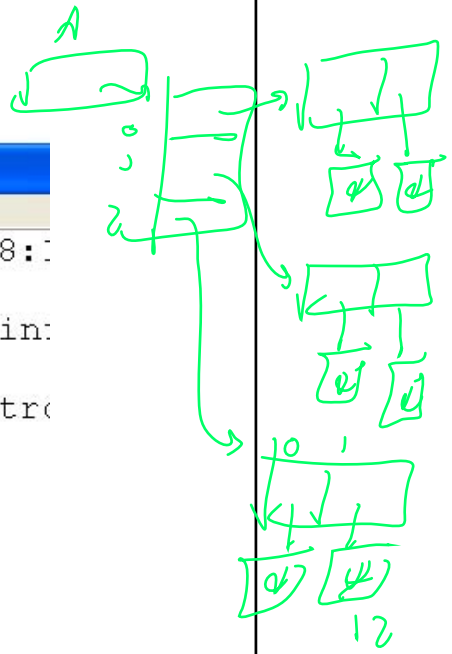
simple.py - C:\Documents and Settings\admin\Desktop\intro-python\examples\16\simple.py (3.4.4)

File Edit Format Run Options Window Help

```
A = [[4] * 2] * 3
print(A)
A[2][1]=12
print(A)
print()
A = [4] * 3
i=0
while (i< 3):
    A[i] = [4] * 2
    i=i+1
print(A)
A[2][1]=12
print(A)
```







Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:1
tel)] on win32
Type "copyright", "credits" or "license()" for more in
>>>
 RESTART: C:\Documents and Settings\admin\Desktop\intr
e.py
[[4, 4], [4, 4], [4, 4]]
[[4, 12], [4, 12], [4, 12]]

[[4, 4], [4, 4], [4, 4]]
[[4, 4], [4, 4], [4, 12]]
```

Multidimensional Array

- A Two dimensional array is called matrix
- M = [[1, 2, 3], *# A 3 × 3 matrix, as nested lists*
  [4, 5, 6], *# Code can span lines if bracketed*
  [7, 8, 9]]

Shared References

num-simple.py - /home/nowzari/Desktop/python/pyth

File Edit Format Run Options Window Help

```
X=42
Y=42
print(X==Y)
print(X is Y)

print("id X is", id(X))
print("id Y is", id(Y))

X=40
Y=X
print(X==Y)
print(X is Y)

print("id X is", id(X))
print("id Y is", id(Y))
X=40
Y=5
print(X==Y)
print(X is Y)

print("id X is", id(X))
print("id Y is", id(Y))
```

```
True
True
id X is 10915680
id Y is 10915680
True
True
id X is 10915616
id Y is 10915616
False
False
id X is 10915616
id Y is 10914496
```

Shared References

simple.py - /home/nowzari/Desktop/python/python

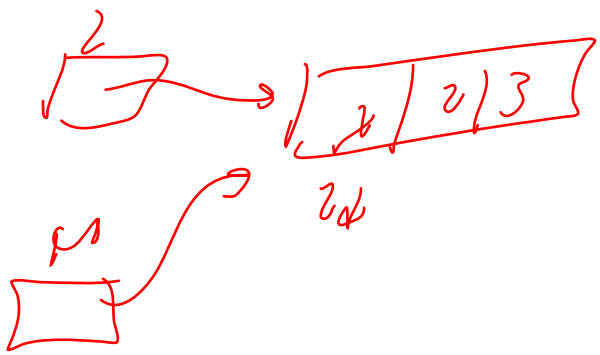
File Edit Format Run Options Window Help

```
L='banana'
M='banana'
print(L==M)
print(L is M)
print(id(L), id(M))   # Same values

L='banana'
M=L
print(L==M)
print(L is M)
print(id(L), id(M))   # Same values

L='banana'
M=L[:]
print(L==M)
print(L is M)
print(id(L), id(M))    # Same values
```

```
True
True
140495238079072 140495238079072
True
True
140495238079072 140495238079072
True
True
140495238079072 140495238079072
```

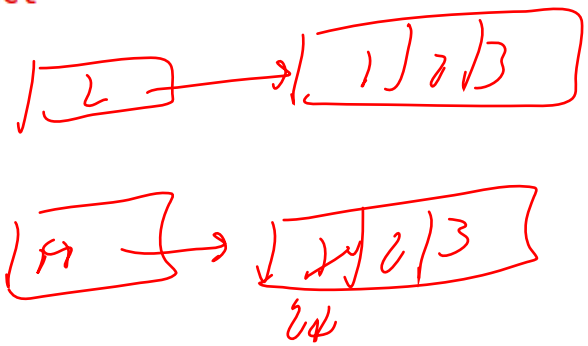
Shared References

```
L = [1, 2, 3]
M = L
print(id(L), id(M))          # M and L reference the same object
print(L == M)                # same values
print(L is M)                # same object
M[0]=24
print("List L:" , L)
print("List M:" , M)

#output

140058938699144 140058938699144
True
True
List L: [24, 2, 3]
List M: [24, 2, 3]
```



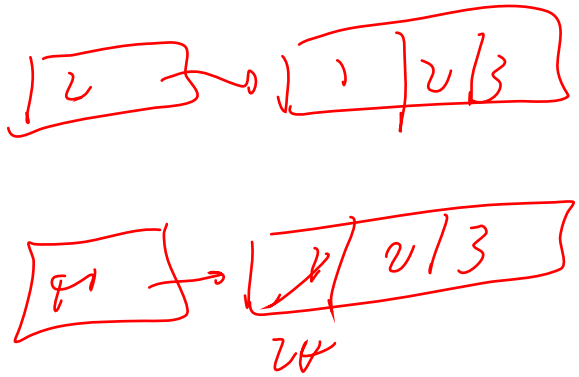
Shared References

```
L = [1, 2, 3]
M = [1, 2, 3]
print(id(L), id(M))          # M and L reference the different object
print(L == M)                # same values
print(L is M)                # different object
M[0]=24
print("List L:" , L)
print("List M:" , M)

#output
140058938695880 140058938762120
True
False
List L: [1, 2, 3]
List M: [24, 2, 3]
```



Shared References

```
L = [1, 2, 3]
M = L[:]
print(id(L), id(M))        # M and L reference the different object
print(L == M)              # same values
print(L is M)              # different object
M[0]=24
print("List L:" , L)
print("List M:" , M)

#output

140058914089800 140058938695880
True
False
List L: [1, 2, 3]
List M: [24, 2, 3]
```



Shared References

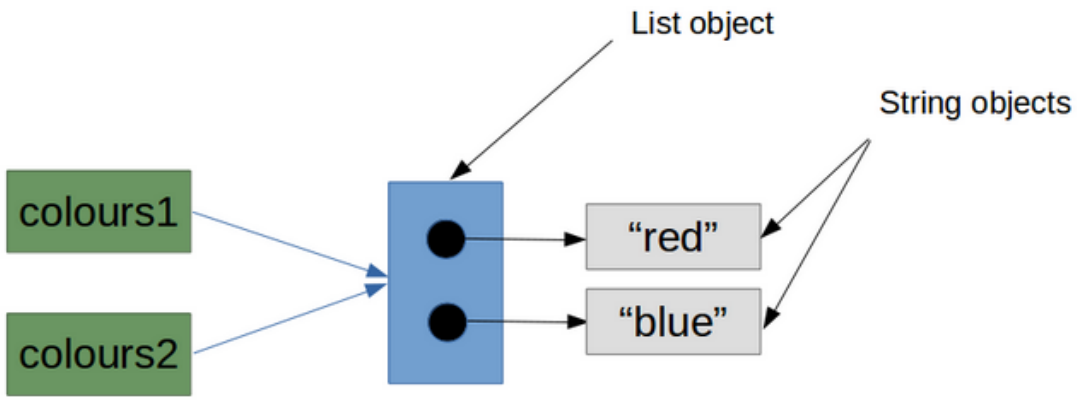
listchange.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/16/

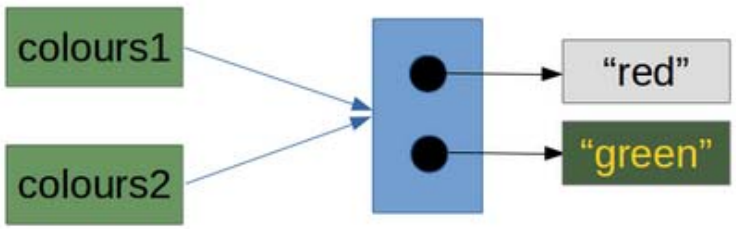
File Edit Format Run Options Window Help

```
colours1 = ["red", "blue"]
colours2 = colours1
print(colours1)
print(colours2)
print(id(colours1),id(colours2))
colours2[1] = "green"
print(colours1)
print(colours2)
print(id(colours1),id(colours2))
```

```
['red', 'blue']
['red', 'blue']
24837656 24837656
['red', 'green']
['red', 'green']
24837656 24837656
```

Shared References





Shared References

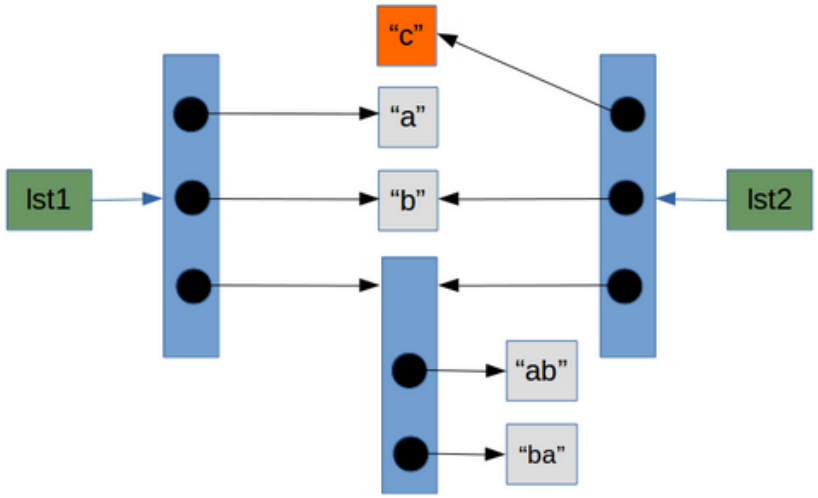
listchange.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/16/listchange.py

File Edit Format Run Options Window Help

```
import copy
lst1 = ['a','b',['ab','ba']]
lst2 = lst1[:]
lst2[0] = 'c'
print(lst1)
print(lst2)
print()

lst1 = ['a','b',['ab','ba']]
lst2 = copy.copy(lst1)
lst2[0] = 'c'
print(lst1)
print(lst2)
print()
```

```
['a', 'b', ['ab', 'ba']]
['c', 'b', ['ab', 'ba']]

['a', 'b', ['ab', 'ba']]
['c', 'b', ['ab', 'ba']]
```

Shared References



Shared References

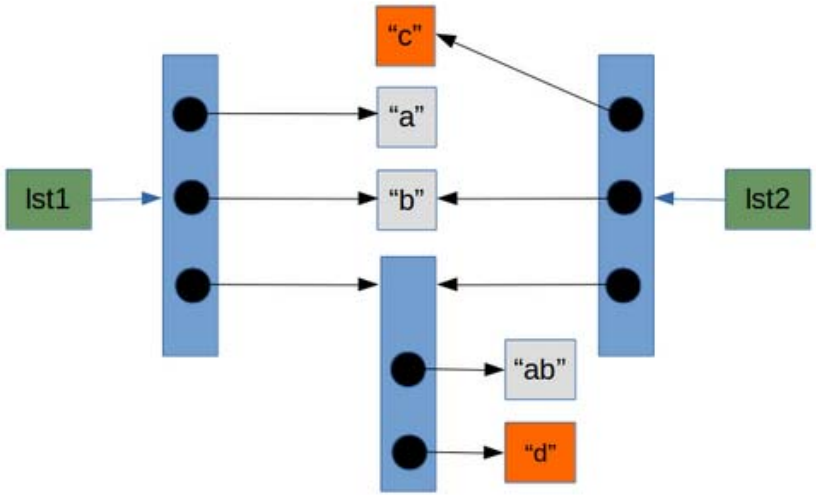
listchange.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/16/listchange.py (3.4.4)

File Edit Format Run Options Window Help

```
import copy
lst1 = ['a','b',['ab','ba']]
lst2 = copy.copy(lst1)
lst2[0] = 'c'
print(lst1)
print(lst2)
print()
lst2[2][1] = 'd'
print(lst1)
print(lst2)
print()
```

```
['a', 'b', ['ab', 'ba']]
['c', 'b', ['ab', 'ba']]

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

Shared References



Shared References

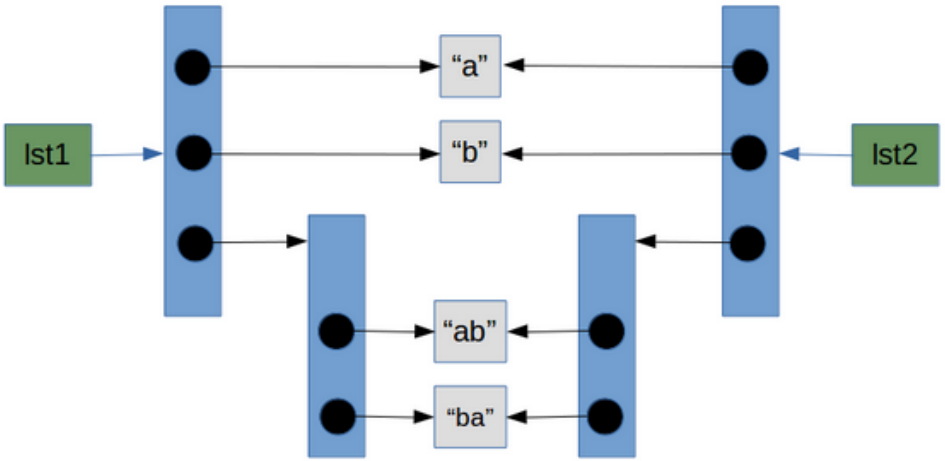
listchange.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/1

File Edit Format Run Options Window Help

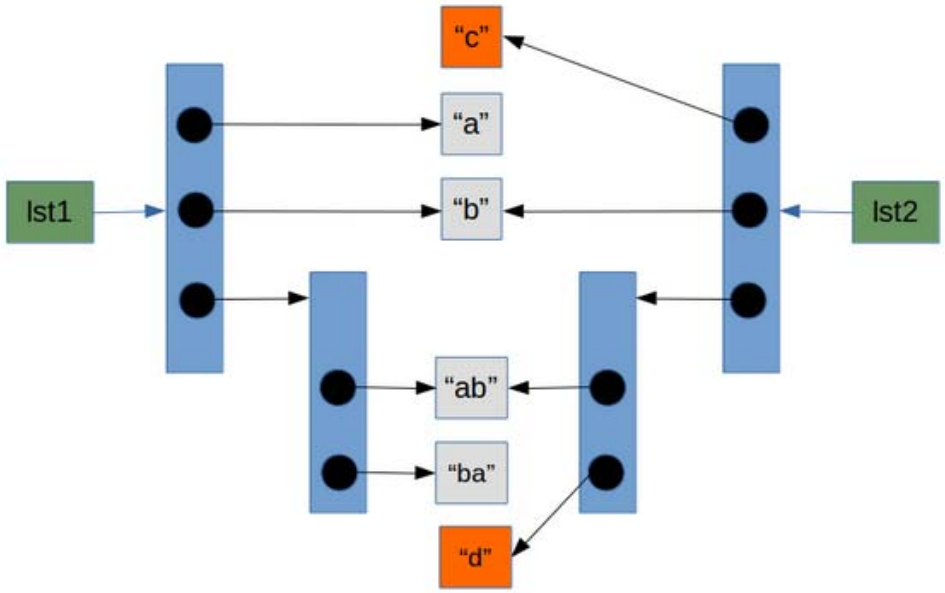
```
from  copy import *
lst1 = ['a','b',['ab','ba']]
lst2 = deepcopy(lst1)
lst2[0] = 'c'
print(lst1)
print(lst2)
print()
lst2[2][1] = 'd'
print(lst1)
print(lst2)
print()
```

```
['a', 'b', ['ab', 'ba']]
['c', 'b', ['ab', 'ba']]

['a', 'b', ['ab', 'ba']]
['c', 'b', ['ab', 'd']]
```

Shared References



Shared References



List as a function parameter

listprint.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/16/listprint.py (3.4.4)

File Edit Format Run Options Window Help

```
def printelm(x, n):
    i=0
    while(i<n):
        print(x[i].ljust(8), repr(x[i+1]).rjust(4))
        i=i+2
    return

lst1=["ali", 12, "hasan", 15, "akbar", 2]
print(lst1)
print()
printelm(lst1, 6)
```

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015,
tel)] on win32
Type "copyright", "credits" or "license()" for
>>>
 RESTART: C:/Documents and Settings/admin/Deskt
rint.py
['ali', 12, 'hasan', 15, 'akbar', 2]

ali         12
hasan       15
akbar        2
```

Parameter passing

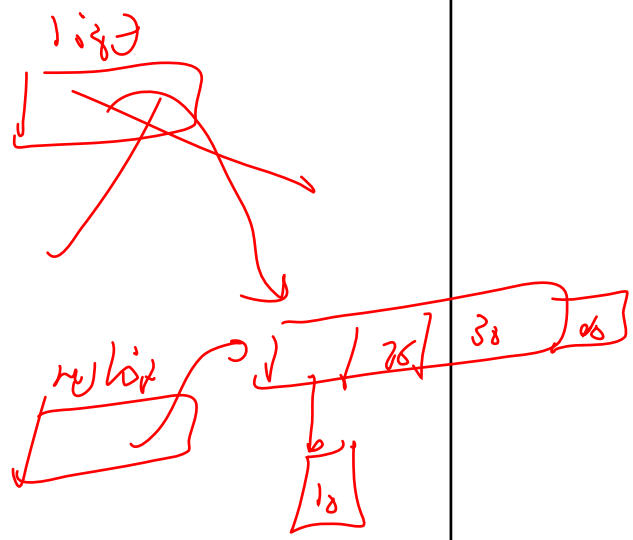
If you pass immutable arguments like integers, strings or tuples to a function, the passing acts like call-by-value. The object reference is passed to the function parameters. They can't be changed within the function, because they can't be changed at all, i.e. they are immutable. It's different, if we pass mutable arguments. They are also passed by object reference, but they can be changed in place in the function. If we pass a list to a function, we have to consider two cases: Elements of a list can be changed in place, i.e. the list will be changed even in the caller's scope. If a new list is assigned to the name, the old list will not be affected, i.e. the list in the caller's scope will remain untouched.

Parameter passing

*listprint.py - /home/nowzari/Desktop/python/python-my/python/examples/14-list/listprin

File Edit Format Run Options Window Help

```
# Function definition is here
def changeme( list1 ):
   "This changes a passed list into this function"
   list1.append(40);
   print ("Values inside the function: ", list1)
   return

# Now you can call changeme function
mylist = [10,20,30];
print ("Values outside the function: ", mylist)
changeme( mylist );
print ("Values outside the function: ", mylist)

#output
Values outside the function:  [10, 20, 30]
Values inside the function:   [10, 20, 30, 40]
Values outside the function:  [10, 20, 30, 40]
```

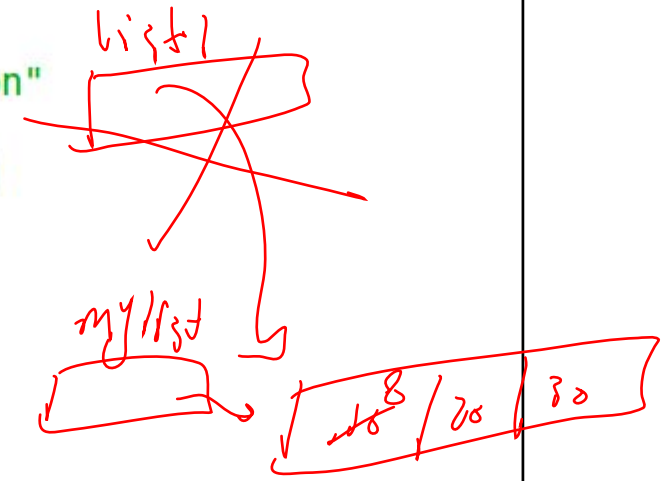


Parameter passing

*listprint.py - /home/nowzari/Desktop/python/python-my/python/examples/14-list/listp

File Edit Format Run Options Window Help

```
# Function definition is here
def changeme( list1 ):
   "This changes a passed list into this function"
   list1[0]=8
   print ("Values inside the function: ", list1)
   return

# Now you can call changeme function
mylist = [10,20,30];
print ("Values outside the function: ", mylist)
changeme( mylist );
print ("Values outside the function: ", mylist)

#output
Values outside the function:  [10, 20, 30]
Values inside the function:   [8, 20, 30]
Values outside the function:  [8, 20, 30]
```

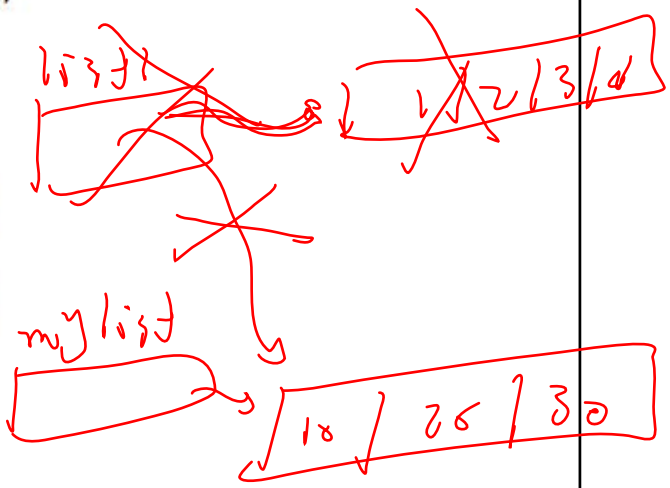


Parameter passing

*listprint.py - /home/nowzari/Desktop/python/python-my/python/examples/14-list/listprint.py (

File Edit Format Run Options Window Help

```
# Function definition is here
def changeme( list1 ):
   "This changes a passed list into this function"
   list1 = [1,2,3,4]; # This would assig new reference in mylist
   print ("Values inside the function: ", list1)
   return

# Now you can call changeme function
mylist = [10,20,30];
print ("Values outside the function: ", mylist)
changeme( mylist );
print ("Values outside the function: ", mylist)

#output
Values outside the function:  [10, 20, 30]
Values inside the function:   [1, 2, 3, 4]
Values outside the function:  [10, 20, 30]
```



Parameter passing

listprint.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/16/listprint.py (3.4.4)

File Edit Format Run Options Window Help

```
# Function definition is here
def changeme( mylist ):
   "This changes a passed list into this function"
   mylist = [1,2,3,4]; # This would assig new reference in mylist
   print ("Values inside the function: ", mylist)
   return

# Now you can call changeme function
mylist = [10,20,30];
changeme( mylist );
print ("Values outside the function: ", mylist)
```

Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

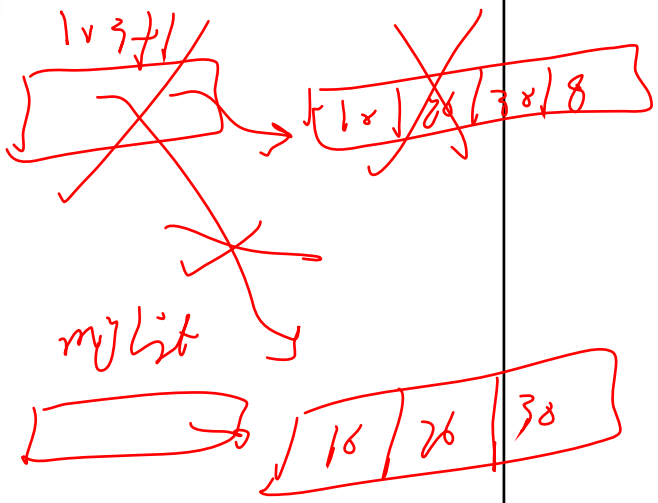
```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:
tel)] on win32
Type "copyright", "credits" or "license()" for more
>>>
 RESTART: C:/Documents and Settings/admin/Desktop/:
rint.py
Values inside the function:  [1, 2, 3, 4]
Values outside the function:  [10, 20, 30]
```

Parameter passing

*listprint.py - /home/nowzari/Desktop/python/python-my/python/examples/14-list/listprin

File Edit Format Run Options Window Help

```
def changeme( list1 ):
    "This changes a passed list into this function"
    list1=list1 + [8]
    print ("Values inside the function: ", list1)
    return

# Now you can call changeme function
mylist = [10,20,30];
print ("Values outside the function: ", mylist)
changeme( mylist );
print ("Values outside the function: ", mylist)

#output
Values outside the function:  [10, 20, 30]
Values inside the function:   [10, 20, 30, 8]
Values outside the function:  [10, 20, 30]
```



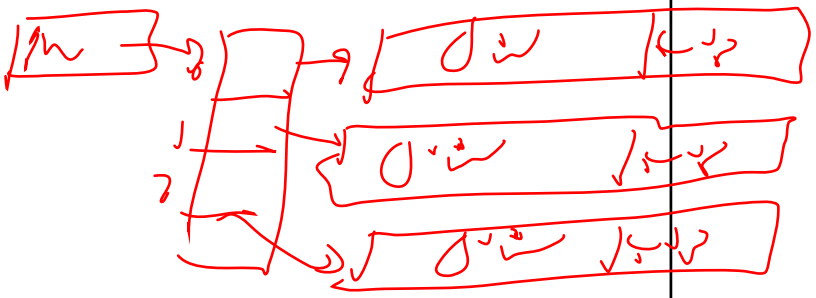
listexp.py - C:/Documents and Settings/admin/Desktop/intro-python/examples/16/listexp.py (3.4.4)

File Edit Format Run Options Window Help

```python
## This program runs a test of knowledge
# First get the test questions
# Later this will be modified to use file io.
def get_questions():
    # notice how the data is stored as a list of lists
    return [["What color is the daytime sky on a clear day? ", "blue"],
            ["What is the answer to life, the universe and everything? ", "42"],
            ["What is a three letter word for mouse trap? ", "cat"]]

# This will test a single question
# it takes a single question in
# it returns True if the user typed the correct answer, otherwise False

def check_question(question_and_answer):
    # extract the question and the answer from the list
    # This function takes a list with two elements, a question and an answer.
    question = question_and_answer[0]
    answer = question_and_answer[1]
    # give the question to the user
    given_answer = input(question)
    # compare the user's answer to the tester's answer
    if answer == given_answer:
        print("Correct")
        return True
    else:
        print("Incorrect, correct was:", answer)
        return False
```

Parameter passing

```
# This will run through all the questions
def run_test(questions):
    if len(questions) == 0:
        print("No questions were given.")
        # the return exits the function
        return
    index = 0
    right = 0
    while index < len(questions):
        # Check the question
        # Note that this is extracting a question and answer list from
        # the list of lists.
        if check_question(questions[index]):
            right = right + 1
        # go to the next question
        index = index + 1
    # notice the order of the computation, first multiply, then divide
    print("You got", right * 100 / len(questions),\
            "% right out of", len(questions))

# now let's get the questions from the get_questions function, and
# send the returned list of lists as an argument to the run_test function.

run_test(get_questions())
```



Python 3.4.4 Shell

File Edit Shell Debug Options Window Help

```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:18) [MS
tel)] on win32
Type "copyright", "credits" or "license()" for more informati
>>>
 RESTART: C:/Documents and Settings/admin/Desktop/intro-pytho
xp.py
What color is the daytime sky on a clear day? green
Incorrect, correct was: blue
What is the answer to life, the universe and everything? 42
Correct
What is a three letter word for mouse trap? cat
Correct
You got 66.66666666666667 % right out of 3
```

End