Example:

```
# Create list with Alphabet from A-Z

# without comprehension

list1=[]
for n in range(65,91):
    list1.append(chr(n))

print(list1)

# with comprehension
list2=[chr(n) for n in range(65,91)]
print(list2)
```

Output:

```
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T',
'U', 'V', 'W', 'X', 'Y', 'Z']
['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T',
'U', 'V', 'W', 'X', 'Y', 'Z']
```

Example:

```
# Write a program to create 2x2 matrix

# without comprehension
matrix=[]
for i in range(2):
    row=[]
    for j in range(2):
        value=int(input("Enter value"))
        row.append(value)
    matrix.append(row)

print(matrix)

# with comprehension
matrix=[[int(input("Enter Value")) for j in range(2)] for i in range(2)]
print(matrix)
```

Output:

```
Enter value1
Enter value2
Enter value3
Enter value4
[[1, 2], [3, 4]]
Enter Value1
Enter Value2
Enter Value3
Enter Value4
[[1, 2], [3, 4]]
```

Syntax2: [expression/value for variable in iterable if test]
Syntax-2 is used for filtering data
This syntax allows create list based on condition or test.

```
salesList=[10000,20000,30000,40000,15000,25000,22000,35000,50000,60
000]

print(salesList)
#without comprehension
salesList1=[]
for s in salesList:
    if s<50000:
        salesList1.append(s)

print(salesList1)

#with comprehension
salesList2=[s for s in salesList if s<50000]
print(salesList2)
```

Output:

```
[10000, 20000, 30000, 40000, 15000, 25000, 22000, 35000, 50000, 60000]
[10000, 20000, 30000, 40000, 15000, 25000, 22000, 35000]
[10000, 20000, 30000, 40000, 15000, 25000, 22000, 35000]
```

Example:

```
# Example of List comprehension
gradeList=[["naresh","A"],
          ["suresh","B"],
```

```
        ["ramesh","A"],
        ["kishore","C"],
        ["rajesh","C"]]

print(gradeList)

#without comprehension
gradeListA=[]
for stud in gradeList:
    if stud[1]=='A':
        gradeListA.append(stud)

print(gradeListA)

gradeListB=[]
for stud in gradeList:
    if stud[1]=='B':
        gradeListB.append(stud)
print(gradeListB)

gradeListC=[]
for stud in gradeList:
    if stud[1]=='C':
        gradeListC.append(stud)
print(gradeListC)


# with comprehension
gradeListA=[stud for stud in gradeList if stud[1]=='A']
gradeListB=[stud for stud in gradeList if stud[1]=='B']
gradeListC=[stud for stud in gradeList if stud[1]=='C']

print(gradeListA)
print(gradeListB)
print(gradeListC)
```

Output:

```
[['naresh', 'A'], ['suresh', 'B'], ['ramesh', 'A'], ['kishore', 'C'], ['rajesh', 'C']]
[['naresh', 'A'], ['ramesh', 'A']]
[['suresh', 'B']]
```

[['kishore', 'C'], ['rajesh', 'C']]
[['naresh', 'A'], ['ramesh', 'A']]
[['suresh', 'B']]
[['kishore', 'C'], ['rajesh', 'C']]

Creating copy of list

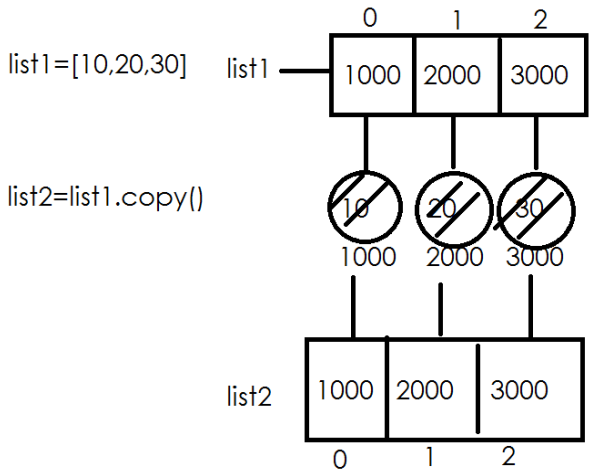
Python allows creating copy of the list in two ways.

1. Shallow Copy
2. Deep Copy

List data type provides a method called copy(), which create shallow copy of the list.

What is shallow copy?

In shallow copy, a new list is created by coping addresses/references of objects found in existing list. Shallow copy is called reference copy.

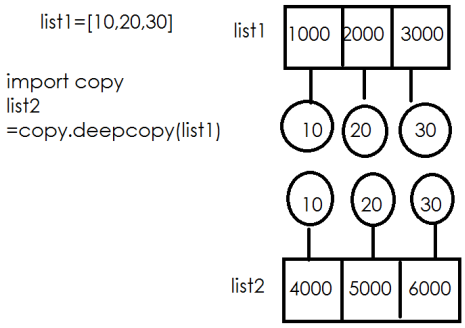



```
>>> list1=[10,20,30]
>>> list2=list1.copy()
>>> print(list1)
[10, 20, 30]
>>> print(list2)
[10, 20, 30]
>>> list3=[[11,22],[33,44]]
>>> list4=list3.copy()
>>> print(list3)
[[11, 22], [33, 44]]
>>> print(list4)
```

[[11, 22], [33, 44]]
>>> list3[0].append(99)
>>> print(list3)
[[11, 22, 99], [33, 44]]
>>> print(list4)
[[11, 22, 99], [33, 44]]
>>> list4[0][0]=88
>>> print(list4)
[[88, 22, 99], [33, 44]]
>>> print(list3)
[[88, 22, 99], [33, 44]]

Deep Copy

In Deep Copy, new list created by coping objects found in existing list. Deep copy is done using deepcopy(), this function exists in one predefined module “copy”.



>>> import copy
>>> list1=[[10,20],[30,40]]
>>> list2=copy.deepcopy(list1)
>>> print(list1)
[[10, 20], [30, 40]]
>>> print(list2)
[[10, 20], [30, 40]]
>>> list1[0].append(99)
>>> print(list1)
[[10, 20, 99], [30, 40]]
>>> print(list2)
[[10, 20], [30, 40]]
>>> list2[0][0]=77
>>> print(list2)

```
[[77, 20], [30, 40]]
>>> print(list1)
[[10, 20, 99], [30, 40]]
```

What is difference between shallow copy and deep copy?
A shallow copy constructs a new compound object and then (to the extent possible) inserts references into it to the objects found in the original. A deep copy constructs a new compound object and then, recursively, inserts copies into it of the objects found in the original.

extend() method
This method is used to append or add more than one value.

Syntax: list-name.extend(iterable)

extend method read each value generated by iterable and append to list.

```
>>>list1=[]
>>> list1.append(10)
>>> print(list1)
[10]
>>> list1.append(20)
>>> print(list1)
[10, 20]
>>> list1.append(30,40)
Traceback (most recent call last):
  File "<pyshell#37>", line 1, in <module>
    list1.append(30,40)
TypeError: list.append() takes exactly one argument (2 given)
>>> list1.extend([30,40])
>>> print(list1)
[10, 20, 30, 40]
>>> list1.extend(range(50,110,10))
>>> print(list1)
[10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

tuple

Tuples are immutable sequences, typically used to store collections of heterogeneous data (such as the 2-tuples produced by the enumerate() built-in). Tuples are also used for cases where an immutable sequence of homogeneous data is needed (such as allowing storage in a set or dict instance).