Creating copy of the list

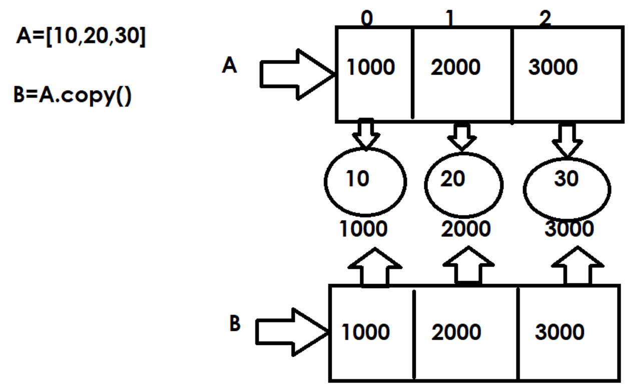
Python allows creating copy using 2 different methods

1. shallow copy
2. deep copy

Shallow copy or Shallow Cloning

Shallow copy creates a new list by copying address of objects found in existing list.
List provides a “copy” method, which returns shallow copy of the list.

Syntax: list-name.copy()



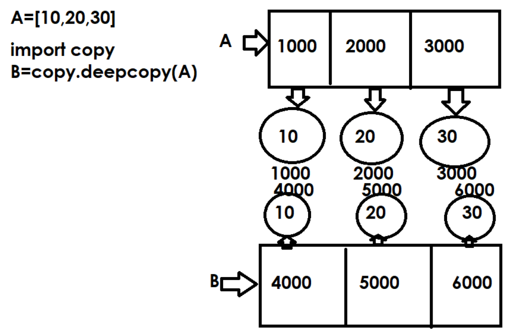
If object exists within list are mutable, changes done using one list is reflected to another list.

```
>>> A=[10,20,30]
>>> B=A.copy()
>>> print(A,B,sep="\n")
[10, 20, 30]
[10, 20, 30]
>>> print(id(A[0]),id(B[0]))
140733845474008 140733845474008
>>> X=[[10,20],30,40]
>>> Y=X.copy()
>>> print(X,Y,sep="\n")
[[10, 20], 30, 40]
[[10, 20], 30, 40]
>>> print(id(X[0]),id(Y[0]))
2266378679360 2266378679360
>>> X[0].append(100)
>>> print(X)
[[10, 20, 100], 30, 40]
>>> print(Y)
[[10, 20, 100], 30, 40]
```

Deep Copy or Deep Cloning
In deep copy a new list object is created by copy objects found in existing list.
Deepcopy is done using a predefined library provided by python.

Syntax:

```
import copy
copy.deepcopy(list-name)
```



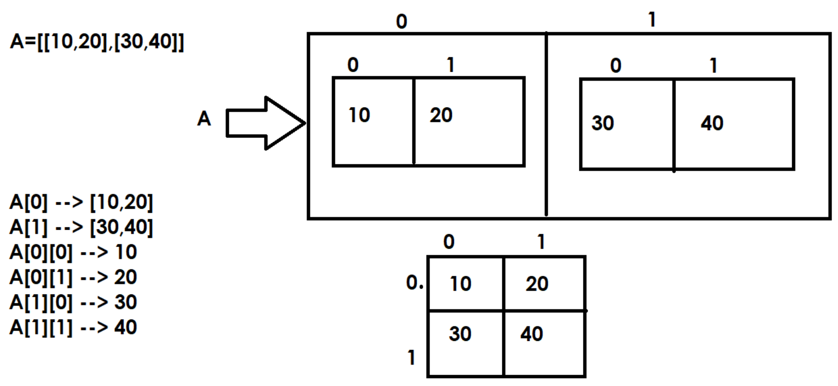
```
>>> import copy
>>> A=[10,20,30]
>>> B=copy.deepcopy(A)
>>> print(A,B,sep="\n")
```

```
[10, 20, 30]
[10, 20, 30]
>>> X=[[10,20]]
>>> import copy
>>> Y=copy.deepcopy(X)
>>> print(id(X[0]),id(Y[0]))
2266378919040 2266378814208
>>> print(Y)
[[10, 20]]
>>> X[0].append(100)
>>> print(X)
[[10, 20, 100]]
>>> print(Y)
[[10, 20]]
>>> Y[0].append(200)
>>> print(Y)
[[10, 20, 200]]
>>> print(X)
[[10, 20, 100]]
```

Nested List

Defining list inside list is called nested list (OR) defining list as element/item within list is called nested list.

Nested list is used to represent data in rows and columns (matrix)



Example:

```
A=[[10,20],[30,40],[50,60],[70,80]]

for i in range(4):
    print(A[i])

for i in range(4): # 0 1 2 3
    for j in range(2): # 0 1
        print(A[i][j],end=' ')
    print()

print()
for x in A:
    print(x)

for x in A:
    for y in x:
        print(y,end=' ')
    print()
```

Output

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

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

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

Example:

```
# Write a program create 3x3 matrix and display

A=[]

for i in range(3):
    row=[]
    for j in range(3):
        value=int(input("Enter Value "))
        row.append(value)
    A.append(row)


print(A)
```

Output

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

Example:

```
# Write a program to add two matrices
# input 2x2 matrix

A=[]
B=[]
C=[]

print("Input Elements of first matrix")
for i in range(2):
    row=[]
    for j in range(2):
        value=int(input("Enter value "))
        row.append(value)
    A.append(row)

print("Input Elements of second matrix")
for i in range(2):
    row=[]
    for j in range(2):
        value=int(input("Enter value "))
        row.append(value)
    B.append(row)

print(A,B,sep="\n")

for i in range(2):
    row=[]
    for j in range(2):
        row.append(A[i][j]+B[i][j])
    C.append(row)

print(C)
```

Output

```
Input Elements of first matrix
Enter value 1
Enter value 2
Enter value 3
Enter value 4
Input Elements of second matrix
Enter value 6
Enter value 7
Enter value 8
Enter value 9
[[1, 2], [3, 4]]
[[6, 7], [8, 9]]
[[7, 9], [11, 13]]
```

Example:

```
# Write a program to print transpose of 3x3 matrix
A=[]
```

```
for i in range(3):
    row=[]
    for j in range(3):
        value=int(input("Enter value "))
        row.append(value)
    A.append(row)

B=[]
for i in range(3):
    row=[]
    for j in range(3):
        row.append(A[j][i])
    B.append(row)

print(A)
print(B)
```

Output

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

Example:

```
# Write a program to read 3x3 matrix and display
# upper triangle and lower traingle
A=[]
print("Input elements of matrix")
for i in range(3):
    row=[]
    for j in range(3):
        value=int(input("Enter value "))
        row.append(value)
    A.append(row)

print("Matrix")
for i in range(3):
    for j in range(3):
        print(A[i][j],end=' ')
    print()

print()
print("Upper Triangle")
for i in range(3):
    for j in range(3):
        if j>=i:
            print(A[i][j],end=' ')
        else:
            print(" ",end=" ")
    print()

print()
print("Lower Triangle")
for i in range(3):
    for j in range(3):
        if i>=j:
            print(A[i][j],end=' ')
    print()
```

Output

```
Input elements of matrix
Enter value 1
Enter value 2
Enter value 3
Enter value 4
Enter value 5
Enter value 6
Enter value 7
Enter value 8
Enter value 9
Matrix
1 2 3
4 5 6
7 8 9

Upper Triangle
1 2 3
  5 6
    9

Lower Triangle
1
4 5
7 8 9
```

Example:

```
# Write a program to read 3 students 3 subject marks
# and calculate total,avg,result(PASS/FAIL)
marks=[]
```

```
print("Input Marks")
for i in range(3):
    row=[]
    for j in range(3):
        m=int(input(f'Student{i+1} Subject{j+1} Marks'))
        row.append(m)
    marks.append(row)

for row in marks:
    total=sum(row)
    avg=total/3
    result="PASS" if row[0]>=40 and row[1]>=40 and row[2]>=40 else
"FAIL"
    print(f'{row}\t{total}\t{avg:.2f}\t{result}')
```

Output

```
Input Marks
Student1 Subject1 Marks40
Student1 Subject2 Marks50
Student1 Subject3 Marks60
Student2 Subject1 Marks70
Student2 Subject2 Marks80
Student2 Subject3 Marks90
Student3 Subject1 Marks60
Student3 Subject2 Marks30
Student3 Subject3 Marks90
[40, 50, 60]150   50.00PASS
[70, 80, 90]240   80.00PASS
[60, 30, 90]180   60.00FAIL
```

Tuple

Tuple is an immutable sequence data type or collection.

After creating tuple changes cannot done (OR) tuple not support mutable operations

1. append()
2. extend()
3. remove()
4. pop()
5. sort()
6. del
7. replace
8. clear()

Where tuple is used?

In application development tuple Is used,

1. for representing immutable list
2. for representing data for other collections (sets, mapping..)

How to create tuple?

Tuple is created is different ways

1. empty tuple is created using empty ()

```
>> A=[]
>>> print(A,type(A))
[] <class 'list'>
>>> B=()
>>> print(B,type(B))
() <class 'tuple'>
>>> A.append(10)
>>> print(A)
[10]
>>> B.append(20)
Traceback (most recent call last):
  File "<pyshell#32>", line 1, in <module>
```

```
    B.append(20)
AttributeError: 'tuple' object has no attribute 'append'
>>> C=[10,20,30]
>>> C[0]=99
>>> print(C)
[99, 20, 30]
>>> D=(10,20,30)
>>> D[0]=99
Traceback (most recent call last):
  File "<pyshell#37>", line 1, in <module>
    D[0]=99
TypeError: 'tuple' object does not support item assignment
```

2. tuple with single element/value is created with single element separated with ,
   A tuple consists of single value/element is called singleton tuple

```
>>> X=(10)
>>> print(type(X))
<class 'int'>
>>> Y=(10,)
>>> print(Y,type(Y))
(10,) <class 'tuple'>
>>> Z=(1.5,)
>>> print(Z,type(Z))
(1.5,) <class 'tuple'>
```

3. tuple with multiple elements are created by separating with ,

```
>>> t1=(10,20,30)
>>> t2=(1.5,2.5,3.5)
>>> t3=(1,"naresh","python",4500.0)
>>> print(t1,t2,t3,sep="\n")
(10, 20, 30)
(1.5, 2.5, 3.5)
(1, 'naresh', 'python', 4500.0)
>>> print(type(t1),type(t2),type(t3),sep="\n")
<class 'tuple'>
<class 'tuple'>
<class 'tuple'>
```

4. tuple can be using tuple() function
   tuple() □ create empty tuple
   tuple(iterable) □ create tuple by converting other collections into tuple type

```
>> t1=tuple()
>>> print(t1,type(t1))
() <class 'tuple'>
>>> t2=tuple(range(10,60,10))
>>> print(t2,type(t2))
(10, 20, 30, 40, 50) <class 'tuple'>
>>> t3=tuple([10,20,30])
>>> print(t3,type(t3))
(10, 20, 30) <class 'tuple'>
>>> t4=tuple("ABC")
>>> print(t4,type(t4))
('A', 'B', 'C') <class 'tuple'>
>>> t5=tuple((10,20,30))
>>> print(t5,type(t5))
(10, 20, 30) <class 'tuple'>
```