1. Python – Join Tuples if similar initial element

In [1]:
```
test_list = [(5, 6), (5, 7), (6, 8), (6, 10), (7, 13)]

print("The original list is : " + str(test_list))

res = []
for sub in test_list:
    if res and res[-1][0] == sub[0]:
        res[-1].extend(sub[1:])
    else:
        res.append([ele for ele in sub])
res = list(map(tuple, res))

print("The extracted elements : " + str(res))
```

```
The original list is : [(5, 6), (5, 7), (6, 8), (6, 10), (7, 13)]
The extracted elements : [(5, 6, 7), (6, 8, 10), (7, 13)]
```

2. Python – Extract digits from Tuple list

In [2]:
```
from itertools import chain

test_list = [(15, 3), (3, 9), (1, 10), (99, 2)]

print("The original list is : " + str(test_list))

temp = map(lambda ele: str(ele), chain.from_iterable(test_list))
res = set()
for sub in temp:
    for ele in sub:
        res.add(ele)

print("The extracted digits : " + str(res))
```

```
The original list is : [(15, 3), (3, 9), (1, 10), (99, 2)]
The extracted digits : {'0', '9', '1', '3', '5', '2'}
```

3. Python – All pair combinations of 2 tuples

In [3]:
```
test_tuple1 = (4, 5)
test_tuple2 = (7, 8)

print("The original tuple 1 : " + str(test_tuple1))
print("The original tuple 2 : " + str(test_tuple2))

res =  [(a, b) for a in test_tuple1 for b in test_tuple2]
res = res +  [(a, b) for a in test_tuple2 for b in test_tuple1]

print("The filtered tuple : " + str(res))
```

```
The original tuple 1 : (4, 5)
The original tuple 2 : (7, 8)
The filtered tuple : [(4, 7), (4, 8), (5, 7), (5, 8), (7, 4), (7, 5), (8, 4), (8, 5)]
```

4. Python – Remove Tuples of Length K

In [4]:
```
test_list = [(4, 5), (4, ), (8, 6, 7), (1, ), (3, 4, 6, 7)]

print("The original list : " + str(test_list))

K = 1

res = [ele for ele in test_list if len(ele) != K]

print("Filtered list : " + str(res))
```

```
The original list : [(4, 5), (4,), (8, 6, 7), (1,), (3, 4, 6, 7)]
Filtered list : [(4, 5), (8, 6, 7), (3, 4, 6, 7)]
```

5. Sort a list of tuples by second Item.

In [5]:
```
def Sort_Tuple(tup):

    lst = len(tup)
    for i in range(0, lst):

        for j in range(0, lst-i-1):
            if (tup[j][1] > tup[j + 1][1]):
                temp = tup[j]
                tup[j]= tup[j + 1]
                tup[j + 1]= temp
    return tup

tup =[('for', 24), ('is', 10), ('Geeks', 28),
      ('Geeksforgeeks', 5), ('portal', 20), ('a', 15)]

print(Sort_Tuple(tup))
```

```
[('Geeksforgeeks', 5), ('is', 10), ('a', 15), ('portal', 20), ('for', 24), ('Geeks', 28)]
```

6. Python program to Order Tuples using external List.

In [7]:
```
test_list = [('god', 3), ('best', 9), ('CS', 10), ('Goods', 2)]

print("The original list is : " + str(test_list))

ord_list = ['Goods', 'best', 'CS', 'god']

temp = dict(test_list)
res = [(key, temp[key]) for key in ord_list]

print("The ordered tuple list : " + str(res))
```

```
The original list is : [('god', 3), ('best', 9), ('CS', 10), ('Goods', 2)]
The ordered tuple list : [('Goods', 2), ('best', 9), ('CS', 10), ('god', 3)]
```

7. Python – Flatten tuple of List to tuple

In [8]:
```
test_tuple = ([5, 6], [6, 7, 8, 9], [3])

print("The original tuple : " + str(test_tuple))

res = tuple(sum(test_tuple, []))

print("The flattened tuple : " + str(res))
```

```
The original tuple : ([5, 6], [6, 7, 8, 9], [3])
The flattened tuple : (5, 6, 6, 7, 8, 9, 3)
```

8. Python – Convert Nested Tuple to Custom Key Dictionary

In [9]:
```
test_tuple = ((4, 'Gfg', 10), (3, 'is', 8), (6, 'Best', 10))

print("The original tuple : " + str(test_tuple))

res = [{'key': sub[0], 'value': sub[1], 'id': sub[2]}
                              for sub in test_tuple]

print("The converted dictionary : " + str(res))
```

```
The original tuple : ((4, 'Gfg', 10), (3, 'is', 8), (6, 'Best', 10))
The converted dictionary : [{'key': 4, 'value': 'Gfg', 'id': 10}, {'key': 3, 'value': 'is', 'id': 8}, {'key': 6, 'value': 'Best', 'id': 10}]
```

9. Python Program for Binary Search (Recursive and Iterative)

In [10]:
```
def binary_search(arr, low, high, x):

    if high >= low:

        mid = (high + low) // 2

        if arr[mid] == x:
            return mid

        elif arr[mid] > x:
            return binary_search(arr, low, mid - 1, x)

        else:
            return binary_search(arr, mid + 1, high, x)

    else:

        return -1

arr = [ 2, 3, 4, 10, 40 ]
x = 10

result = binary_search(arr, 0, len(arr)-1, x)

if result != -1:
    print("Element is present at index", str(result))
else:
    print("Element is not present in array")
```

```
Element is present at index 3
```

10. Python Program for Linear Search

In [13]:
```
def linear_Search(list1, n, key):

    for i in range(0, n):
        if (list1[i] == key):
            return i
    return -1


list1 = [1 ,3, 5, 4, 7, 9]
key = 7

n = len(list1)
res = linear_Search(list1, n, key)
if(res == -1):
    print("Element not found")
else:
    print("Element found at index: ", res)
```

```
Element found at index:  4
```

11. Python Program for Insertion Sort

In [14]:
```
def insertionSort(arr):

    for i in range(1, len(arr)):

        key = arr[i]

        j = i-1
        while j >=0 and key < arr[j] :
                arr[j+1] = arr[j]
                j -= 1
        arr[j+1] = key

arr = [12, 11, 13, 5, 6]
insertionSort(arr)
print ("Sorted array is:")
for i in range(len(arr)):
    print ("%d" %arr[i])
```

```
Sorted array is:
5
6
11
12
13
```

In [ ]:
```

```