In [3]:
```
# seperate the positive and negative number from given list
my_list = [-3,5,-5,-6,-7]
pos_list = []
neg_list = []

for value in my_list:
    if value > 0:
        pos_list.append(value)
    else:
        neg_list.append(value)
```

In [4]:
```
print(f'positive number list :{pos_list}')
print('negative number list:',neg_list)
```

```
positive number list :[5]
negative number list: [-3, -5, -6, -7]
```

In [5]:
```
# syntax for creating the function
# def function_name(param1,param2,etc)
def seperator(my_list):
    # local variable which are only accessed by seperator function itself
    neg =[]
    pos =[]
    for value in my_list:
        if value > 0:
            pos.append(value)
        else:
            neg.append(value)
    print(f'positive number list:{pos}')
    print(f'negative number list:{neg}')
```

In [7]:
```
filtered_list = [2,-1,-2,4,5-6,-8,6]
filtered_list

# calling function
seperator(filtered_list)
```

```
positive number list:[2, 4, 6]
negative number list:[-1, -2, -1, -8]
```

In [8]:
```
second_filtered_list = [2,4,5,-1,-5,3,-1]
seperator(second_filtered_list)
```

```
positive number list:[2, 4, 5, 3]
negative number list:[-1, -5, -1]
```

In [9]:
```
# syntax for creating the function
# def function_name(param1,param2,etc)
def filter_pos_neg_function(my_list):
    # local variable which are only accessed by seperator function itself
    neg =[]
    pos =[]
    for value in my_list:
        if value > 0:
            pos.append(value)
        else:
            neg.append(value)
    # a return is special statements which returns function variable back
    # to function caller
    return pos, neg
```

In [14]:
```
pos_list,neg_list = filter_pos_neg_function(second_filtered_list)
```

In [15]:
```
pos_list
```

Out[15]: [2, 4, 5, 3]

In [16]:
```
neg_list
```

Out[16]: [-1, -5, -1]

One line functins

In [19]:
```
add = lambda x:x+10
```

In [20]:
```
add(10)
```

Out[20]: 20

In [21]:
```
add(30)
```

Out[21]: 40

In [22]:
```
add_two_numb = lambda x,y: x+y
add_two_numb(12,23)
```

Out[22]: 35

In [23]:
```
add_two_numb = lambda x,y: x-y
```

In [24]:
```
add_two_numb(12,34)
```

Out[24]: -22

In [35]:
```
check_greater = lambda x,y:True if x> y else False
```

In [36]:
```
check_greater(20,10)
```

Out[36]:
```
True
```

List comprehension

In [44]:
```
two_table = []
for value in range(1,11):
    two_table.append(value*2)
    print(two_table)
two_table
```

```
[2]
[2, 4]
[2, 4, 6]
[2, 4, 6, 8]
[2, 4, 6, 8, 10]
[2, 4, 6, 8, 10, 12]
[2, 4, 6, 8, 10, 12, 14]
[2, 4, 6, 8, 10, 12, 14, 16]
[2, 4, 6, 8, 10, 12, 14, 16, 18]
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

Out[44]:
```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

In [43]:
```
# syntax [return_value operations1, operations2]
two_table = [value*2 for value in range(1,11)]
two_table
```

```
CPU times: total: 0 ns
Wall time: 0 ns
```

Out[43]:
```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

In [45]:
```
# syntax [return_value operations1, operations2]
two_table = [value*2 for value in range(1,11) if value> 4]
two_table
```

Out[45]:
```
[10, 12, 14, 16, 18, 20]
```

Use of break, continue and pass

In [52]:
```
# continue is special statements to skip the given condition and contiune
# the loop till its end
for value in range(1,10):
    if value in [5,6,7]:
        continue
    print(value)
```

```
1
2
3
4
8
9
```

In [55]:
```
# break is special statements to break the loop continuation as soon as
# given condition matches
for value in range(1,10):
    if value == 6:
        break
    print(value)
```

```
1
2
3
4
5
```

In [60]:
```
# pass is the special statements to be used as placeholder
# for future code
for i in range(1,30):
    pass
```

In [65]:
```
numb_item = int(input('Enter number of items:'))
my_list = []
price_list =[]
for i in range(numb_item):
    order = input('Enter ordered value :')
    my_list.append(order)
    price = int(input('Enter price'))
    price_list.append(price)
```

```
Enter number of items:2
Enter ordered value :tea
Enter price10
Enter ordered value :bread
Enter price20
```

In [66]:
```
price_list
```

Out[66]:
```
[10, 20]
```

In [67]:
```
my_list
```

Out[67]:
```
['tea', 'bread']
```

In [ ]:
```

```