3.1 Write a Python Program to implement your own myreduce() function which works exactly like Python's built-in function reduce()

In [7]:
```
def myreduce(anyfunc, sequence):
   # Get first item in sequence and assign to result
   result = sequence[0]

   # iterate over remaining items in sequence and apply reduction function
   for item in sequence[1:]:
      result = anyfunc(result, item)

   return result

def sum(x,y):
    return x + y

print ("Sum on list [1,2,3] using custom reduce function " + str(myreduce(sum, [1,2,3])) )
```

```
Sum on list [1,2,3] using custom reduce function 6
```

In [8]:
```
def myreduce(func, iterable, start = None):
    it = iter(iterable)
    startposition = 0
    if start is None:
        try:
            start = next(it)
            startposition = 1
        except StopIteration:
            raise TypeError('reduce() of empty sequence with no initial value')
    accum_value = start

    for x in iterable[startposition:]:
        accum_value = func(accum_value, x)

    return accum_value

print(myreduce(lambda x,y: x+y, [2,3,5,10]))
print(myreduce(lambda x,y: x+y, [2,3,5,10], 5))
print(myreduce(lambda x,y: x-y, [2,3,5,10], 3))
```

```
20
25
-17
```

3.2 Write a Python program to implement your own myfilter() function which works exactly like Python's built-in function filter()

In [9]:
```
# Custom filter function
def myfilter(anyfunc, sequence):

    # Initialize empty list
    result = []

    # iterate over sequence of items in sequence and apply filter function
    for item in sequence:
      if anyfunc(item):
       result.append(item)

    # return funal output
    return result

# test myfilter function
def ispositive(x):

 if (x <= 0):
   return False
 else:
   return True

print ("Filter only positive Integers on list [0,1,-2,3,4,5] using custom filter function" + str(myfilter(isp
ositive, [0,1,-2,3,4,5])))
```

```
Filter only positive Integers on list [0,1,-2,3,4,5] using custom filter function[1, 3, 4, 5]
```

3.3.Implement List comprehensions to produce the following lists. Write List comprehensions to produce the following Lists

['A', 'C', 'A', 'D', 'G', 'I', ’L’, ‘ D’]

['x', 'xx', 'xxx', 'xxxx', 'y', 'yy', 'yyy', 'yyyy', 'z', 'zz', 'zzz', 'zzzz']

['x', 'y', 'z', 'xx', 'yy', 'zz', 'xxx', 'yyy', 'zzz', 'xxxx', 'yyyy', 'zzzz']

[[2], [3], [4], [3], [4], [5], [4], [5], [6]]

[[2, 3, 4, 5], [3, 4, 5, 6], [4, 5, 6, 7], [5, 6, 7, 8]]

[(1, 1), (2, 1), (3, 1), (1, 2), (2, 2), (3, 2), (1, 3), (2, 3), (3, 3)]

In [10]:
```
my_string ='ACADGILD'
my_list = [letter for letter in my_string]
print(my_list)
```

```
['A', 'C', 'A', 'D', 'G', 'I', 'L', 'D']
```

In [11]:
```
my_string ='xyz'
my_range = [1, 2, 3,4]
my_list = [x * y for x in my_string for y in my_range]
print(my_list)
```

```
['x', 'xx', 'xxx', 'xxxx', 'y', 'yy', 'yyy', 'yyyy', 'z', 'zz', 'zzz', 'zzzz']
```

In [12]:
```
my_string ='xyz'
my_range = [1, 2, 3,4]
my_list = [x *y  for y in my_range for x in my_string]
print(my_list)
```

```
['x', 'y', 'z', 'xx', 'yy', 'zz', 'xxx', 'yyy', 'zzz', 'xxxx', 'yyyy', 'zzzz']
```

In [13]:
```
input_list = [2,3,4]
result = [ [item+num] for item in input_list for num in range(0,3)]
print(str(result))
```

```
[[2], [3], [4], [3], [4], [5], [4], [5], [6]]
```

In [14]:
```
matrix = [[y, y+1, y+2, y+3] for x in range(2,3) for y in range(x, x+4)]
print(matrix)
```

```
[[2, 3, 4, 5], [3, 4, 5, 6], [4, 5, 6, 7], [5, 6, 7, 8]]
```

In [15]:
```
matrix = [(y,x) for x in range(1,4) for y in range(1,4)]
print(matrix)
```

```
[(1, 1), (2, 1), (3, 1), (1, 2), (2, 2), (3, 2), (1, 3), (2, 3), (3, 3)]
```

3.4 Find the longest word from a list of words

In [16]:
```
def longestWord(words_list):
    word_len = []
    for n in words_list:
        word_len.append((len(n), n))
    word_len.sort()
    return word_len[-1][1]

print(longestWord(["python 3.0", "Data Science learning" , "numpy", "matplotlib", "pandas", "Data cleaning",
"Anaconda", "ML"]))
```

```
Data Science learning
```