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

In [1]:
```
from functools import reduce
import random
lst = [random.randint(1,100) for i in range(10)]
print("List containing random integers are---> \t",lst)
```

In [2]:
```
def myreduce(myfunc,mylist):
    res = mylist[0]
    for i in range(1,len(mylist)):
        res = myfunc(res,mylist[i])
    return res
```

In [3]:
```
# Checking result of my function
ans = myreduce(lambda x,y:x+y,lst)
print("Result of implemented myfunc = ",ans )
```

```
Result of implemented myfunc =  400
```

In [4]:
```
# Checking result of in-built function
print("Result of in-built reduce function = ",reduce(lambda x,y:x+y,lst))
```

```
Result of in-built reduce function =  400
```

In [5]:
```
# Checking result of my function
ans = myreduce(lambda x,y:x*y,lst)
print("Result of implemented myfunc = ",ans )
# Checking result of in-built function
print("Result of in-built reduce function = ",reduce(lambda x,y:x*y,lst))
```

```
Result of implemented myfunc =  32154523914240
Result of in-built reduce function =  32154523914240
```

In [6]:
```
# Checking result of my function
ans = myreduce(lambda x,y:x/y,lst)
print("Result of implemented myfunc = ",ans )
# Checking result of in-built function
print("Result of in-built reduce function = ",reduce(lambda x,y:x/y,lst))
```

```
Result of implemented myfunc =  1.5238913233698908e-12
Result of in-built reduce function =  1.5238913233698908e-12
```

In [7]:
```
# Checking result of my function
ans = myreduce(lambda x,y:x//y,lst)
print("Result of implemented myfunc = ",ans )
# Checking result of in-built function
print("Result of in-built reduce function = ",reduce(lambda x,y:x//y,lst))
```

```
Result of implemented myfunc =  0
Result of in-built reduce function =  0
```

In [8]:
```
# Checking result of my function
ans = myreduce(lambda x,y:x^y,lst)
print("Result of implemented myfunc = ",ans )
# Checking result of in-built function
print("Result of in-built reduce function = ",reduce(lambda x,y:x^y,lst))
```

```
Result of implemented myfunc =  100
Result of in-built reduce function =  100
```

In [ ]:

In [ ]:

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

In [9]:
```
def myfilter(myfunc, mylist):
    #myfunc has to return boolean value
    res = []
    for elem in mylist:
        if(myfunc(elem)):
            res.append(elem)
    return res
```

In [10]:
```
# Printing list elements
lst =[random.randint(1,100) for i in range(10)]
print("List elements are  = ",lst)
```

```
List elements are  =  [8, 35, 66, 43, 73, 80, 66, 92, 35, 38]
```

In [11]:
```
# Checking result of myfilter() function
print("Condition check is of even numbers\n")
ans = myfilter(lambda x:1 if x%2==0 else 0,lst)
```

```
print("Result of implemented myfilter = ",ans )
# Checking result of in-built filter function
print("Result of in-built filter function = ",list(filter(lambda x:1 if x%2==0 else 0,lst)))
```

```
Condition check is of even numbers

Result of implemented myfilter =  [8, 66, 80, 66, 92, 38]
Result of in-built filter function =  [8, 66, 80, 66, 92, 38]
```

In [12]:
```
# Checking result of myfilter() function
print("Condition check is of multiple of 5 numbers\n")
ans = myfilter(lambda x:1 if x%5==0 else 0,lst)
print("Result of implemented myfilter = ",ans )
# Checking result of in-built filter function
print("Result of in-built filter function = ",list(filter(lambda x:1 if x%5==0 else 0,lst)))
```

```
Condition check is of multiple of 5 numbers

Result of implemented myfilter =  [35, 80, 35]
Result of in-built filter function =  [35, 80, 35]
```

In [13]:
```
lst = "iNeuron is offering best course content"
print("Iterable elements are  = ",lst)
```

```
Iterable elements are  =  iNeuron is offering best course content
```

In [14]:
```
# Checking result of myfilter() function
print("Filetring all vowels in the String \n")
ans = myfilter(lambda x:0 if x in ['a','e','i','o','u'] else 1,lst)
print("Result of implemented myfilter = ",ans )
# Checking result of in-built filter function
print("Result of in-built filter function = ",list(filter(lambda x:0 if x in ['a','e','i','o','u'] else 1,lst)))
```

```
Filetring all vowels in the String

Result of implemented myfilter =  ['N', 'r', 'n', ' ', 's', ' ', 'f', 'f', 'r', 'n', 'g', ' ', 'b', 's', 't', ' ', 'c', 'r', 's',
' ', 'c', 'n', 't', 'n', 't']
Result of in-built filter function =  ['N', 'r', 'n', ' ', 's', ' ', 'f', 'f', 'r', 'n', 'g', ' ', 'b', 's', 't', ' ', 'c', 'r',
's', ' ', 'c', 'n', 't', 'n', 't']
```

In [ ]:
```

```

2. Implement List comprehensions to produce the following lists.

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

In [15]:
```
print([y*x for x in ['x','y','z'] for y in range(1,5)])
```

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

In [16]:
```
#Other way -
print(['x'*i for i in range(1,5)]+['y'*i for i in range(1,5)]+['z'*i for i in range(1,5)])
```

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

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

In [17]:
```
print([y for x in [['x','y','z',i+1] for i in range(4)] for y in [x[0]*x[3],x[1]*x[3],x[2]*x[3]]])
```

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

In [ ]:

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

In [18]:
```
print([[y] for x in [[i,i+1,i+2] for i in range(2,5)] for y in x])
```

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

In [ ]:

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

In [19]:
```
print([x for x in [[i,i+1,i+2,i+3] for i in range(2,6)] ])
```

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

In [ ]:

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

In [20]:
```
print([(y,x) for x in range(1,4) for y in range(1,4)])
```

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

In [ ]:

In [ ]: