In [1]:

```
#write a function mymap that takes in a callable and iterable-and returns a square of odd numbers from 1 to n
def num_square(n):
    return n**2

def mymap(my_function, my_list):
    output_list=[]
    for ele in my_list:
        output_list.append(my_function(ele))
    return output_list

n = int(input("Enter value for upper range: "))
l = mymap(num_square, range(1, n+1, 2))

print(f'\nThe squares of odd numbers from 1 to {n} are:')
print(l)
```

```
Enter value for upper range: 10

The squares of odd numbers from 1 to 10 are:
[1, 9, 25, 49, 81]
```

In [2]:

```
#write a function mymap that takes in a callable and iterable-takes a list of words and returns a word with "ing" appended to it
def append_ing(word):
    return word + 'ing'

def mymap(my_function, my_list):
    output_list=[]
    for word in my_list:
        output_list.append(my_function(word))
    return output_list

words_list=['study','work','help','enjoy','walk']
l= mymap(append_ing, words_list)

print(f"List of words with 'ing' appended to it are:\n")
print(l)
```

```
List of words with 'ing' appended to it are:

['studying', 'working', 'helping', 'enjoying', 'walking']
```

In [3]:

```
#write a function mymap that takes in a callable and iterable-return a list of tuples that has the word and its length
def word_wordlength(name):
    return name, len(name)

def mymap(my_function, my_list):
    output_list=[]
    for word in my_list:
        output_list.append(my_function(word))
    return output_list

city_list=['Bangalore','Mysore','Chickmagalur','Shivamogga','Belgaum']
l= mymap(word_wordlength, city_list)
print(l)
```

```
[('Bangalore', 9), ('Mysore', 6), ('Chickmagalur', 12), ('Shivamogga', 10), ('Belgaum', 7)]
```

In [ ]:

```
# ###### 2.Write a function to mimic filter - called myfilter. Test this with the following.
#     a) Given a list of strings, remove all strings having first character as digit.
#     b) Pickup all words whose length exceeds 6.
#     c) Find all strings in a given line of text ending with a given suffix.
```

In [4]:

```
#write a function mymap that takes in a callable and iterable-and returns a list of words having a word and its length
def str_not_isdigit(s):
    return not s[0].isdigit()

def myfilter(my_function, my_list):
    result = []
    for ele in my_list:
        if my_function(ele):
            result.append(ele)
    return result

str_list=['2oclock','PESU','5star','M-Section','24barseven']
print(myfilter(str_not_isdigit,str_list))
```

```
['PESU', 'M-Section']
```

In [5]:

```
def myfilter(my_function, my_list):
    result = []
    for ele in my_list:
        if my_function(ele):
            result.append(ele)
    return result

city_list=['Bangalore','Mysore','Chickmagalur','Shivamogga','Belgaum']
print(myfilter(lambda s: len(s)>6,city_list))
```

```
['Bangalore', 'Chickmagalur', 'Shivamogga', 'Belgaum']
```

In [7]:

```
def ends_with(word):
    return word.endswith(suffix)

def myfilter(my_function, my_list):
    result = []
    for ele in my_list:
        if my_function(ele):
            result.append(ele)
    return result

words_list=input('Enter a line of text: ').split()
print(words_list)
suffix=input('\nEnter a suffix:')
print(myfilter(ends_with,words_list))
```

```
Enter a line of text: India is my country
['India', 'is', 'my', 'country']

Enter a suffix:India
[]
```

In [4]:

```
# ###### 3.Write a function to mimic reduce - called myreduce. Test this with the following.
#     a) Given a multiword name, print its first letters
#     b) Find the sum of first n natural numbers
#     c) Find factorial of n
```

In [5]:

```
def capitalize_word(s1,s2):
    return s1+s2[0]

def myreduce(my_function, my_list):
    result=''
    for name in my_list:
        result=my_function(result,name)
    return result

name_list=['Punith','Raj','Kumar']
print(myreduce(capitalize_word,name_list))
```

```
PRK
```

In [9]:

```
def add2(n1,n2):
    return n1+n2

def myreduce(my_function, my_list):
    result=0
    for num in my_list:
        result=my_function(result,num)
    return result

n=10
print(f'The sum of first {n} natural numbers is', end=' ')
print(myreduce(add2,range(1,n+1)))
```

```
The sum of first 10 natural numbers is 55
```

In [10]:

```
#
def prod2(n1,n2):
    return n1*n2

def myreduce(my_function, my_list):
    result=1
    for num in my_list:
        result=my_function(result,num)
    return result

n=5
print(f'The factorial of {n} is', end=' ')
print(myreduce(prod2,range(2,n+1)))
```

```
The factorial of 5 is 120
```

In [1]:

```
# write a function that mimics reduce called myreduce- given a list of integrs find the maximum in it.
def my_reduce(my_function,my_list):
    r=my_list[0]
    for next in my_list[1:]:
        r=my_function(r,next)
    return r
l=[23,45,12,47,54]
print(my_reduce(lambda x,y: x if x>y else y,l))
```

```
54
```

In [2]:

```
# write a function that mimics reduce called myreduce- given a list of integrs combine them to form a single integer.
def my_reduce(my_function,my_list):
    r=my_list[0]
    for next in my_list[1:]:
        r=my_function(r,next)
    return r
l=[1,25,32,4]
print(int(my_reduce(lambda x,y:str(x)+str(y),l)))
```

```
125324
```

In [ ]: