Importing my library notebook

In [15]:
```
%run my_functions_library.ipynb

import math
import matplotlib.pyplot as plt       #to be used for plotting
plt.figure(figsize=(9,6))
```

Out[15]:
```
<Figure size 648x432 with 0 Axes>

<Figure size 648x432 with 0 Axes>
```

Question 1

Sum of natural no.s

In [16]:
```
# Sum of first N natural numbers


n=input("Enter a natural number: ")                                        # taking input from the user
a=is_natural(n)
if a != 1:
    print("The sum of first "+ str(n) + " natural numbers is " + str(sum_natural_numbers(a)))
```

```
Enter a natural number: 5
The sum of first 5 natural numbers is 15
```

In [17]:
```
n=['5','7.2','dibya','45','-32']

for i in range(len(n)):
    print("\nThe input number is " + n[i])
    a = is_natural(n[i])
    if a!=1:
        print("The sum of first "+ str(n[i]) + " natural numbers is " + str(sum_natural_numbers(a)))
```

```
The input number is 5
The sum of first 5 natural numbers is 15

The input number is 7.2
Input is not a Natural number.

The input number is dibya
Input is not a Natural number.

The input number is 45
The sum of first 45 natural numbers is 1035

The input number is -32
Input is not a Natural number.
```

Sum of odd no.s

In [18]:
```
# Sum of first N odd natural numbers


n=input("Enter a natural number: ")                                         # taking input from the user
a=is_natural(n)
if a != 1:
    sum_odd_numbers(a)
```

```
Enter a natural number: 7
The sum of first 7 odd numbers is 49
```

In [19]:
```
n=['5','7.2','dibya','45','-32']

for i in range(len(n)):
    print("\nThe input number is " + n[i])
    a = is_natural(n[i])
    if a!=1:
        sum_odd_numbers(a)
```

```
The input number is 5
The sum of first 5 odd numbers is 25

The input number is 7.2
Input is not a Natural number.

The input number is dibya
Input is not a Natural number.

The input number is 45
The sum of first 45 odd numbers is 2025

The input number is -32
Input is not a Natural number.
```

Question 2

AP

In [20]:
```
# Sum of first N entries of an AP


print("common difference of the AP is 1.5")
a=input("Enter first term of AP: ")
n=input("Enter the number of terms: ")                                    # taking input from the user
p=is_float(a)
q=is_natural(n)
if (p!=1 and q!=1):
    sum_numbers_AP(p,q)
```

```
common difference of the AP is 1.5
Enter first term of AP: 3
Enter the number of terms: 4

The sum of first 4 terms of an AP is 21.0
```

In [22]:
```
a=['8','7.9','5','3.0','dib']                              #array of first term of AP
n=['5','7.2','dibya','45','-32']                           #array of no. of terms for AP

for i in range(len(n)):
    print("\nThe input number is " + n[i])
    print("The first term of AP is "+ a[i])
    print("common difference of the AP is 1.5")
    p=is_float(a[i])
    q=is_natural(n[i])
    if (p!=1 and q!=1):
        sum_numbers_AP(p,q)
```

```
The input number is 5
The first term of AP is 8
common difference of the AP is 1.5

The sum of first 5 terms of an AP is 55.0

The input number is 7.2
The first term of AP is 7.9
common difference of the AP is 1.5
Input is not a Natural number.

The input number is dibya
The first term of AP is 5
common difference of the AP is 1.5
Input is not a Natural number.

The input number is 45
The first term of AP is 3.0
common difference of the AP is 1.5

The sum of first 45 terms of an AP is 1620.0

The input number is -32
The first term of AP is dib
common difference of the AP is 1.5
The input is not as desired.
Input is not a Natural number.
```

GP

In [23]:
```
#  Sum of first N terms of a GP


print("common ratio of the GP is 0.5")                                 # taking input from the user
a=input("Enter first term of GP: ")
n=input("Enter number of terms: ")
p=is_float(a)
q=is_natural(n)
if (p!=1 and q!=1):
    sum_numbers_GP(p,q)
```

```
common ratio of the GP is 0.5
Enter first term of GP: 6
Enter number of terms: 6

The sum of first 6 terms of an GP is 11.8125
```

In [26]:
```
a=['8','7.9','5','3.0','dib']                                #array of first term of GP
n=['5','7.2','dibya','45','-32']                             #array of no. of term of GP

for i in range(len(n)):
    print("\nThe input number is " + n[i])
    print("The first term of GP is "+ a[i])
    print("common ratio of the GP is 1.5")
    p=is_float(a[i])
    q=is_natural(n[i])
    if (p!=1 and q!=1):
        sum_numbers_GP(p,q)
```

```
The input number is 5
The first term of GP is 8
common ratio of the GP is 1.5

The sum of first 5 terms of an GP is 15.5

The input number is 7.2
The first term of GP is 7.9
common ratio of the GP is 1.5
Input is not a Natural number.

The input number is dibya
The first term of GP is 5
common ratio of the GP is 1.5
Input is not a Natural number.

The input number is 45
The first term of GP is 3.0
common ratio of the GP is 1.5

The sum of first 45 terms of an GP is 5.9999999999998295

The input number is -32
The first term of GP is dib
common ratio of the GP is 1.5
The input is not as desired.
Input is not a Natural number.
```

HP

In [27]:
```
#  Sum of first N terms of a HP


print("common difference of the HP is 1.5")                              # taking input from the user
a=input("Enter first term of HP: ")
n=input("Enter number of terms: ")
p=is_float(a)
q=is_natural(n)
if (p!=1 and q!=1):
    sum_numbers_HP(p,q)
```

```
common difference of the HP is 1.5
Enter first term of HP: 5.6
Enter number of terms: 7

The sum of first 7 terms of an HP is 0.7657413948507557
```

In [28]:
```
a=['8','7.9','5','3.0','dib']                                  #array first term of HP
n=['5','7.2','dibya','45','-32']                               #array of no. of terms of HP

for i in range(len(n)):
    print("\nThe input number is " + n[i])
    print("The first term of HP is "+ a[i])
    print("common difference of the HP is 1.5")
    p=is_float(a[i])
    q=is_natural(n[i])
    if (p!=1 and q!=1):
        sum_numbers_HP(p,q)
```

```
The input number is 5
The first term of HP is 8
common difference of the HP is 1.5

The sum of first 5 terms of an HP is 0.47260082023239924

The input number is 7.2
The first term of HP is 7.9
common difference of the HP is 1.5
Input is not a Natural number.

The input number is dibya
The first term of HP is 5
common difference of the HP is 1.5
Input is not a Natural number.

The input number is 45
The first term of HP is 3.0
common difference of the HP is 1.5

The sum of first 45 terms of an HP is 2.2777914973240696

The input number is -32
The first term of HP is dib
common difference of the HP is 1.5
The input is not as desired.
Input is not a Natural number.
```

Question 3 (Factorial)

In [29]:
```
# Factorial of a number


n=input("Enter a natural number")                                  # taking input from the user

if int(n)==0:                                    # finding factorial of 0 separately and other natural numbers separately
    factorial(0)
else:
    n=is_natural(n)                        # n has datatype integer now
    if n!=1:
        print("Factorial of " + str(n) + " is : " + str(factorial(n)))
```

```
Enter a natural number8
Factorial of 8 is : 40320
```

In [30]:
```
n=['5','7.2','dibya','45','-32']

for i in range(len(n)):
    print("\nThe input number is " + n[i])
    a = is_natural(n[i])
    if a!=1:
        print("Factorial of " + str(a) + " is : " + str(factorial(a)))
```

```
The input number is 5
Factorial of 5 is : 120

The input number is 7.2
Input is not a Natural number.

The input number is dibya
Input is not a Natural number.

The input number is 45
Factorial of 45 is : 119622220865480194561963161495657715064383733760000000000

The input number is -32
Input is not a Natural number.
```

Question 4

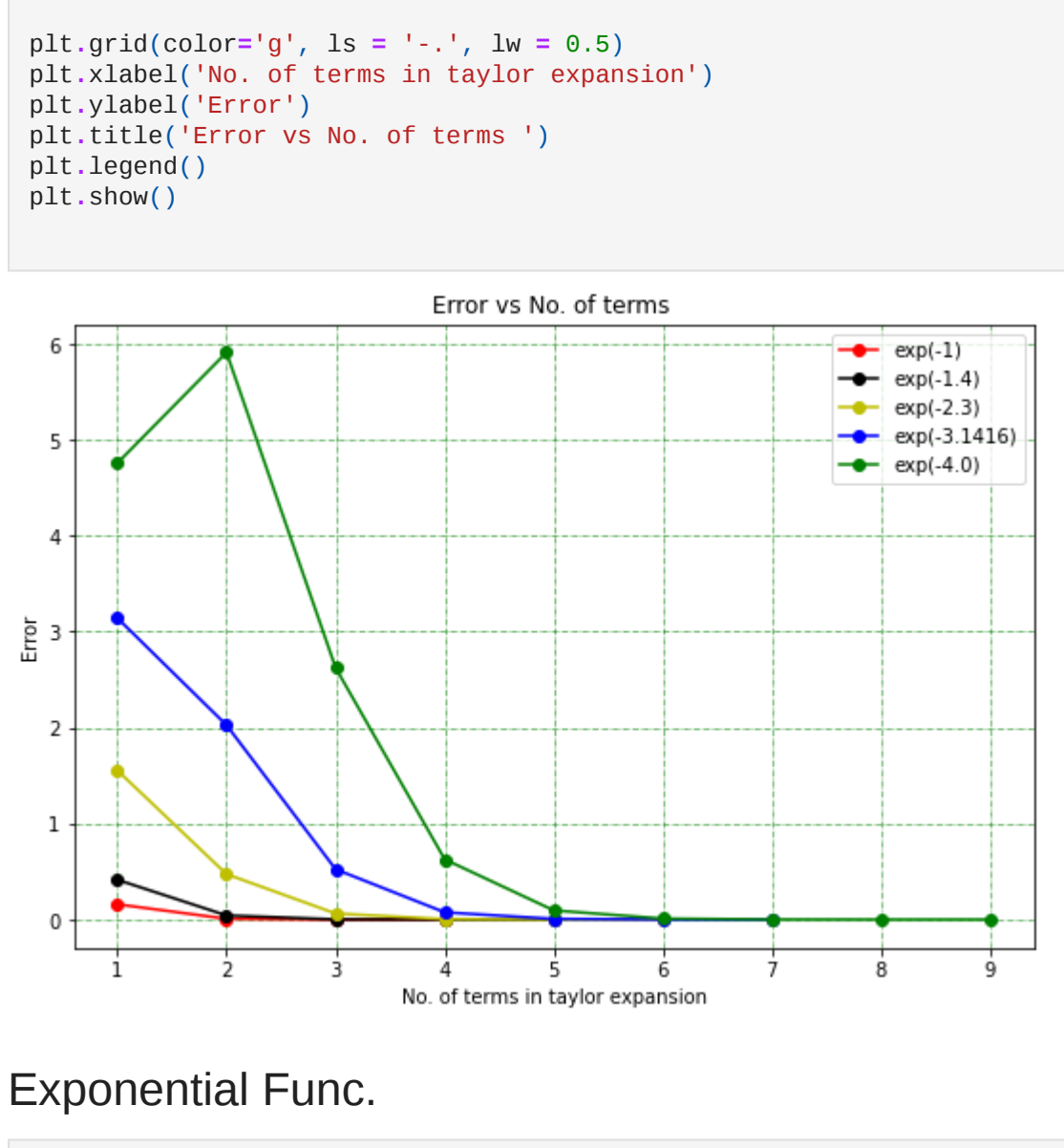
Sine Func.

In [31]:
```
# Question 4(a): Sine function

n=input("Enter the argument for sin function")
n=is_float(n)
if int(n)!=1:
    exp=10**-6
    i=1                                            # The loop runs till the value matches with the actual valueof sin(x)
    while abs(sin_func(n,i)-math.sin(n))>exp:
        i+=1
    print("\nSin( " +str(n)+ " ) = "+ str(sin_func(n,i)))
    print("It is accurate atleast upto 4 decimal places.")
```

```
Enter the argument for sin function3.1417

Sin( 3.1417 ) = -0.00010811964341245458
It is accurate atleast upto 4 decimal places.
```

Errors and Plots (sin_func)

In [33]:
```
%run my_functions_library.ipynb

import math
import matplotlib.pyplot as plt       #to be used for plotting
plt.figure(figsize=(9,6))

eps=10**-6 # value of epsilon - decimal places upto which accuracy is desired

argument=[1, 1.4, 2.3, 3.1416, 4.0] # array of arguments given for comparison
color=['r-o', 'k-o', 'y-o','b-o','g-o'] # array of colors for plotting

for j in range(len(argument)):
    # initializing two arrays to store indices and errors
    index=[]
    error=[]

    x=argument[j] # argument if sine function
    i=1

    # the loop runs till the value doesn't match with
    # the actual value of sine and terminates as it matches
    # upto desired decimal places
    while abs(sin_func(x,i)-math.sin(x))>eps:
        index.append(i)
        error.append(abs(sin_func(x,i)-math.sin(x)))
        i+=1
    plt.plot(index, error, color[j],label='exp(-'+str(x)+')')

plt.grid(color='g', ls = '-.', lw = 0.5)
plt.xlabel('No. of terms in taylor expansion')
plt.ylabel('Error')
plt.title('Error vs No. of terms ')
plt.legend()
plt.show()
```



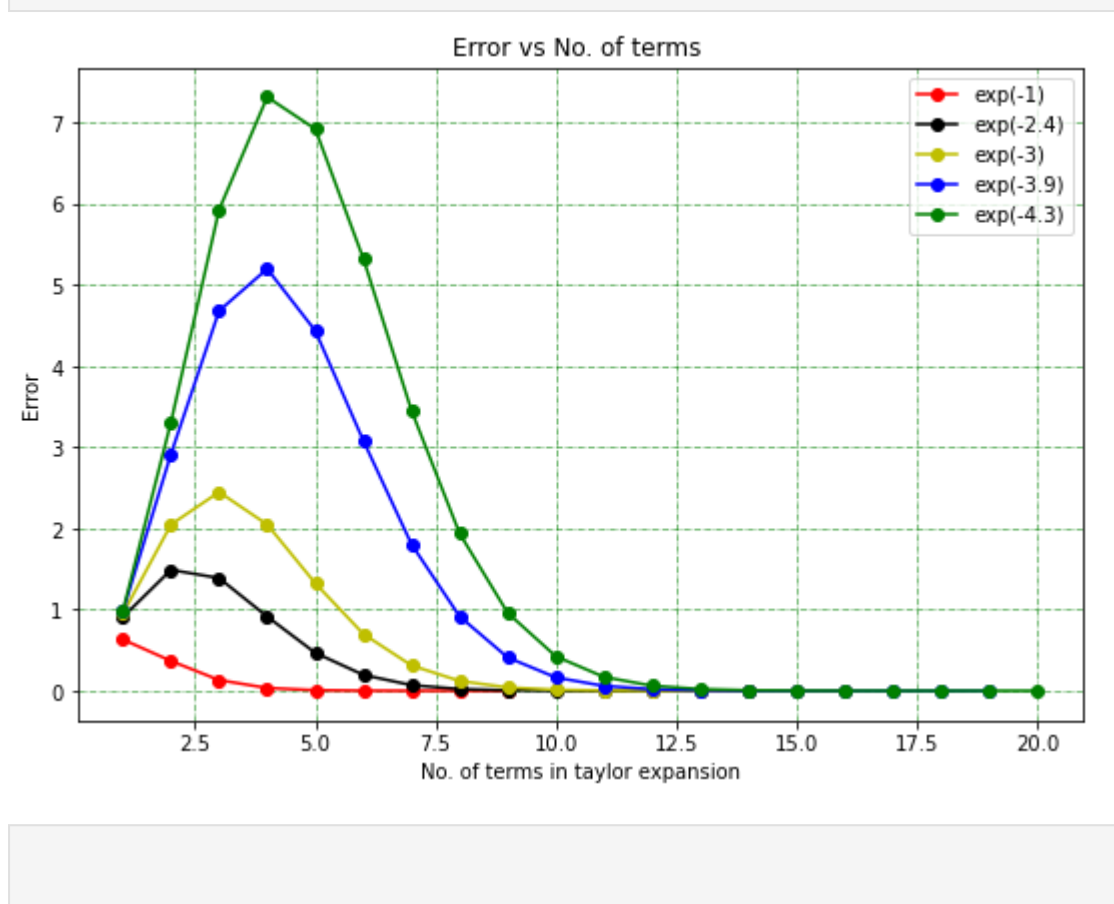
Exponential Func.

In [34]:
```
# Question 4(b): Exp function

n=input("Enter the argument for exp function: ")
n=is_natural(n)
if int(n)!=1:
    eps=10**-6
    i=1
    while abs(exp_func(x,i)-math.exp(-x))>eps:             #The loop runs till the value matches with the actual valueof exp(-x)
        i+=1
    print("Exp(-"+ str(n) + ") =" + str(exp_func(n,i)))
    print("It is accurate upto atleast 4 decimal places. ")
```

```
Enter the argument for exp function: 3
Exp(-3) =0.04978706711474102
It is accurate upto atleast 4 decimal places.
```

Errors and Plots (exp_func)

In [37]:
```
%run my_functions_library.ipynb

import math
import matplotlib.pyplot as plt                                        #to be used for plotting
plt.figure(figsize=(9,6))

eps=10**-6                                               # value of epsilon - decimal places upto which accuracy is desired

argument=[1, 2.4, 3, 3.9, 4.3]                           # array of arguments given for comparison
color=['r-o', 'k-o', 'y-o','b-o','g-o']                   # array of colors for plotting

for j in range(len(argument)):
    # initializing two arrays to store indices and errors
    index=[]
    error=[]

    x=argument[j] # argument if sine function
    i=1

    # the loop runs till the value doesn't match with
    # the actual value of sine and terminates as it matches
    # upto desired decimal places
    while abs(exp_func(x,i)-math.exp(-x))>eps:
        index.append(i)
        error.append(abs(exp_func(x,i)-math.exp(-x)))
        i+=1
    plt.plot(index, error, color[j], label='exp(-'+ str(x)+')')

plt.grid(color='g', ls = '-.', lw = 0.5)
plt.xlabel('No. of terms in taylor expansion')
plt.ylabel('Error')
plt.title('Error vs No. of terms ')
plt.legend()
plt.show()
```



In [ ]:

In [ ]: