In [3]:

```
#  Question 1
#Code for calculating the sum of first N natural numbers and Odd natural numbers

def Nat_Sum(n) :#defined a function
    sum = 0
    j = 1
    i = 0
    if n < 0 :
        print("Enter a valid input for n i.e. put n > 0")#Check for valid input
    else:
        sum = 0
    while i < n:
        sum = sum + j
        j = j + 1
        i = i + 1
    return sum

#Driver function
n = 10
print (" Sum of first" , n, "natural Numbers is: ",
                                Nat_Sum(n) )

n = 15
print (" Sum of first" , n, "natural Numbers is: ",
                                Nat_Sum(n) )
n = 20
print (" Sum of first" , n, "natural Numbers is: ",
                                Nat_Sum(n) )
n = 9
print (" Sum of first" , n, "natural Numbers is: ",
                                Nat_Sum(n) )

n = -3
print (" Sum of first" , n, "natural Numbers is: ",
                                Nat_Sum(n) )
```

```
 Sum of first 10 natural Numbers is:  55
 Sum of first 15 natural Numbers is:  120
 Sum of first 20 natural Numbers is:  210
 Sum of first 9 natural Numbers is:  45
Enter a valid input for n i.e. put n > 0
 Sum of first -3 natural Numbers is:  0
```

In [2]:

```
def odd_Sum(n) :#defining a function for calculating the odd sum
    sum = 0
    j = 1
    i = 0
    if n < 0 :# condition for valid input
        print("Enter a valid input for n i.e. put n > 0")
    else:
        sum = 0
    while i < n:
        sum = sum + j
        j = j + 2
        i = i + 1
    return sum

# Driver Code
n = 20
print (" Sum of first" , n, "Odd Numbers is: ",
                                odd_Sum(n) )
n = 5
print (" Sum of first" , n, "Odd Numbers is: ",
                                odd_Sum(n) )
n = 6
print (" Sum of first" , n, "Odd Numbers is: ",
                                odd_Sum(n) )
n = 7
print (" Sum of first" , n, "Odd Numbers is: ",
                                odd_Sum(n) )
n = -3
print (" Sum of first" , n, "Odd Numbers is: ",
                                odd_Sum(n))
```

```
 Sum of first 20 Odd Numbers is:  400
 Sum of first 5 Odd Numbers is:  25
 Sum of first 6 Odd Numbers is:  36
 Sum of first 7 Odd Numbers is:  49
Enter a valid input for n i.e. put n > 0
 Sum of first -3 Odd Numbers is:  0
```

In [2]:

```
# Question 2
#Calculating the sum of N terms of an AP, GP and HP series for common difference d=1.5
 and common ratio 0.5.


def sumOfAP( a, d, N) : #defining a function for AP
    sum = 0
    i = 0
    if N < 0:
        print("Invalid input of N, put N > 0")
    else:
        while i < N :
            sum = sum + a
            a = a + d
            i = i + 1
    return sum

# Driver function

N = -3
a = 1
d = 1.5
sumAp = sumOfAP(a,d,N)
if sumAp!=0 :
    print ("Sum of the ap series is ",sumOfAP(a, d, N))


N = 7
a = 5
d = 1.5
print ("Sum of the ap series is ",sumOfAP(a, d, N))


N = 10
a = 6
d = 1.5
print ("Sum of the ap series is ",sumOfAP(a, d, N))


N = 20
a = 5
d = 1.5
print ("Sum of the ap series is ",sumOfAP(a, d, N))
```

```
Invalid input of N, put N > 0
Sum of the ap series is  66.5
Sum of the ap series is  127.5
Sum of the ap series is  385.0
```

In [3]:

```
#Calculating the sum of N terms of a GP series for  common ratio 0.5.

def sumOfgp( a, R, N) : #defining a function of GP
    sum = 0
    i = 0

    while i < N :
        sum = sum + a
        a = a * R
        i = i + 1
    return sum

# Driver function
N = 10
a = 1
R = 0.5
print ("Sum of the GP series is ",sumOfgp(a, R, N))

N = 10
a = 5
R = 0.5
print ("Sum of the GP series is ",sumOfgp(a, R, N))

N = 15
a = 10
R = 0.5
print ("Sum of the GP series is ",sumOfgp(a, R, N))

N = 5
a = 9
R = 0.5
print ("Sum of the GP series is ",sumOfgp(a, R, N))

N = 30
a = 12
R = 0.5
print ("Sum of the GP series is ",sumOfgp(a, R, N))
```

```
Sum of the GP series is  1.998046875
Sum of the GP series is  9.990234375
Sum of the GP series is  19.9993896484375
Sum of the GP series is  17.4375
Sum of the GP series is  23.999999977648258
```

In [5]:

```
#Calculating the sum of N terms of a HP series for  common difference d=1.5

def sumOfHP( a, d, N) :
    sum = 0
    i = 0
    if a == 0:
        print("Enter a valid input for 'a', 'a' can't be 0")#Checking validity of input
'a'
    else:
        sum = 0
        while i < N :
            sum = sum + 1/a
            a = a + d
            i = i + 1
        return sum


sumhp=sumOfHP(a,d,N)
if sumhp!= None:
    print ("Sum of the HP series is ",sumOfHP(a, d, N))


# Driver function
N = 2
a = 1
d = 1.5
sumhp=sumOfHP(a,d,N)
if sumhp!= None:
    print ("Sum of the HP series is ",sumOfHP(a, d, N))

N = 6
a = 0
sumhp=sumOfHP(a,d,N)
if sumhp!= None:
    print ("Sum of the HP series is ",sumOfHP(a, d, N))

N = 8
a = 16
sumhp=sumOfHP(a,d,N)
if sumhp!= None:
    print ("Sum of the HP series is ",sumOfHP(a, d, N))

N = 9
a = 150
sumhp=sumOfHP(a,d,N)
if sumhp!= None:
    print ("Sum of the HP series is ",sumOfHP(a, d, N))

N = 2
a = 15

sumhp=sumOfHP(a,d,N)
if sumhp!= None:
    print ("Sum of the HP series is ",sumOfHP(a, d, N))
```

```
Sum of the HP series is  0.12727272727272726
Sum of the HP series is  1.4
Enter a valid input for 'a', 'a' can't be 0
Sum of the HP series is  0.38679850989561454
Sum of the HP series is  0.05772790639122848
Sum of the HP series is  0.12727272727272726
```

In [1]:

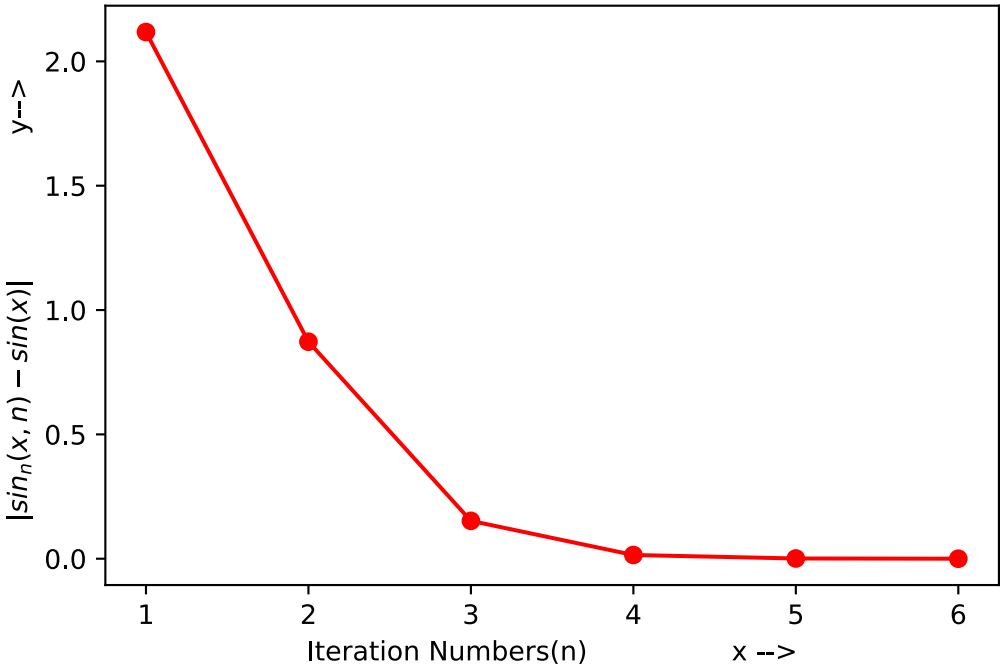
```
#Question 3
#Calculation factotial of a natural number
def fact_rial(num) :#defined a function
   if num < 0:
      print("Sorry,Invalid Input as factorial does not exist for negative integers")#ch
ecking valid input
   else:
      i = 0
      prod = 1
      j = 1
      while i < num :#Executing while loop
         prod = prod * j
         j = j + 1
         i = i + 1
      return prod

#Driver function
num = 6
#specifying output for any invalid input
fact=fact_rial(num)
if fact!=None :
   print (" Factorial of " , num , " is: ",
                              fact_rial(num) )

num = 0

fact=fact_rial(num)
if fact!=None :
   print (" Factorial of " , num , " is: ",
                              fact_rial(num) )

num = -3

fact=fact_rial(num)
if fact!=None :
   print (" Factorial of " , num , " is: ",
                              fact_rial(num) )

num = 5

fact=fact_rial(num)
if fact!=None :
   print (" Factorial of " , num , " is: ",
                              fact_rial(num) )
```

```
 Factorial of  6  is:  720
 Factorial of  0  is:  1
Sorry,Invalid Input as factorial does not exist for negative integers
 Factorial of  5  is:  120
```

In [9]:

```
#Question 4 part (a)
#Approximating sine function upto 4 decimal places and plotting a graph between Modulus
of errors and iteration number

import math
import matplotlib.pyplot  as plt

def fact_rial(num) :#Running the factorial statement
   if num < 0:
      print("Sorry, Invalid Input as factorial does not exist for negative numbers")
   else:
      i = 0
      prod = 1
      j = 1
      while i < num :
         prod = prod * j
         j = j + 1
         i = i + 1
      return prod

def sin_n( x, n):#defining sine function
    add = 0
    for j in range(n):
        y = x*(math.pi/180)#Converting degrees to radians
        add += (-1)**j*((y**(2*j+1))/int((fact_rial(2*j+1))))#calling the predefined fa
ctorial function
    return add


x=120 # x in degrees
n=40

print("The value of sin(",x,") degrees according to defined Taylor series is ", sin_n(x
,n))#value of defined sine function
print("Original value of sin(",x,") degrees is",math.sin(math.pi*x/180)) #original valu
e of sine function


# desired value of 'x' in 'degrees'
x=150
list1=[]
list2=[]
#making a list of plotting variables
i=1
while abs(math.sin((math.pi*x)/180)-sin_n(x,i))>10**(-5):
    list1.append(i)
    list2.append(abs(math.sin((math.pi*x)/180)-sin_n(x,i)))
    i=i+1

print("The Plot of Modulus of errors vs. Iteration Numbers(n) for x=150 degrees :")
#plot
plt.xlabel("Iteration Numbers(n)                x -->")
plt.ylabel("$|sin_n(x,n)-sin(x)|$                y--> ")
plt.plot(list1,list2,'-ro')


plt.show()
```

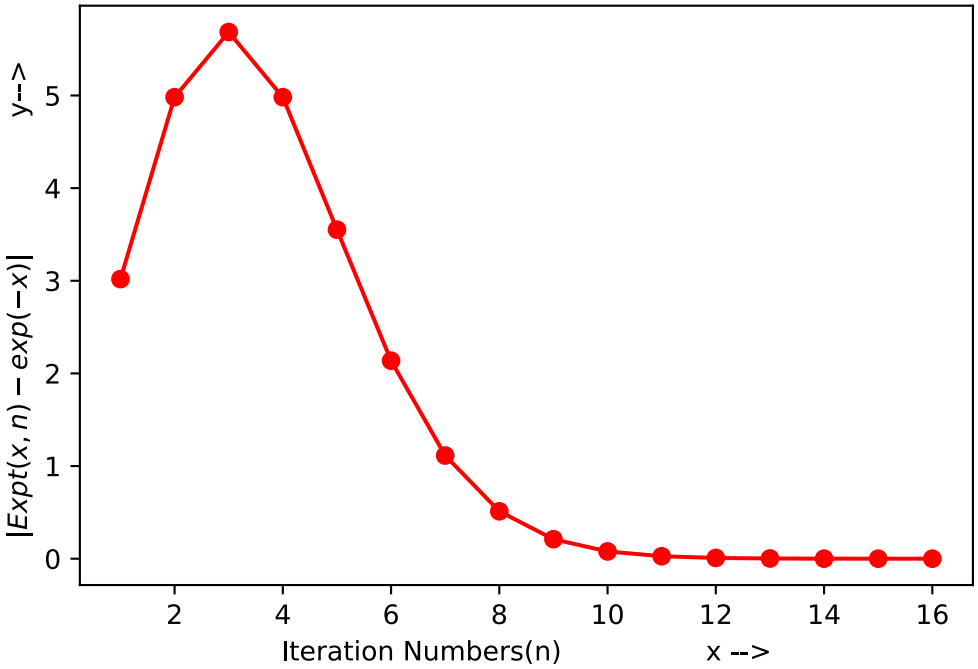
```
The value of sin( 120 ) degrees according to defined Taylor series is  0.8
660254037844389
Original value of sin( 120 ) degrees is 0.8660254037844387
The Plot of Modulus of errors vs. Iteration Numbers(n) for x=150 degrees :
```



In [6]:

```
#Question 4 part (b)
#Approximating exp(-x) function upto 4 decimal places and plotting a graph between Modu
lus of errors and iteration numbers


import math
import matplotlib.pyplot  as plt
def fact_rial(num) :#Running the factorial statement
   if num < 0:
      print("Sorry, Invalid Input as factorial does not exist for negative numbers")
   else:
      i = 0
      prod = 1
      j = 1
      while i < num :
         prod = prod * j
         j = j + 1
         i = i + 1
      return prod

def Expt( x, n):#defining Exponential function
    add = 0
    for j in range(n+1):

        add += (-1)**j*(x)**(j)/int(fact_rial(j)) #calling the predefined factorial fun
ction
    return add

x=1.4 # value of x
n=15
print("The value of exp(",-x,")according to defined Taylor series is ", Expt(x,n))#valu
e of defined exp(-x) function
print("Original value of exp(",-x,") is",math.exp(-x)) #original value of exp(-x) funct
ion


x=4# desired value of 'x'


# making lists to collect data points
list1=[]
list2=[]

i=1
while abs(math.exp(-x)-Expt(x,i))>10**(-5) :#Taking Error level accuracy upto 10^{-5}
    list1.append(i)#iteration number
    list2.append(abs(math.exp(-x)-Expt(x,i)))
    i=i+1



print("The Plot of Modulus of errors vs. Iteration Numbers(n) :")
#plot
plt.xlabel("Iteration Numbers(n)                x -->")
plt.ylabel("$|Expt(x,n)-exp(-x)|$               y--> ")
plt.plot(list1,list2,'-ro')

plt.show()
```

```
The value of exp( -1.4 )according to defined Taylor series is  0.246596963
93199226
Original value of exp( -1.4 ) is 0.2465969639416065
The Plot of Modulus of errors vs. Iteration Numbers(n) :
```



In [ ]: