In [162…

```
#To find sum of first N natural numbers and of odd numbers upto N
def sum(n):
    total = 0
    total_odd = 0
    try:
        if n<=0:                        #non-positive integers are eliminated
            print("Please enter a natural number")
        else:
            for i in range(n+1):
                total+=i                #natural number total is calculated upto the spe
                total_odd+=2*i+1        #sum of N odd numbers is calculated
            print("The sum of first " + str(n) + " natural numbers is = " + str(total))
            print("The sum of first " + str(n) + " odd numbers is = " + str(total_odd))
    except TypeError:                   #fractional real numbers (floats) are eliminated
        print("Please enter a natural number")
```

In [163…

```
sum(2.4), sum(10), sum(-10), sum(1000)
```

```
Please enter a natural number
The sum of first 10 natural numbers is = 55
The sum of first 10 odd numbers is = 121
Please enter a natural number
The sum of first 1000 natural numbers is = 500500
The sum of first 1000 odd numbers is = 1002001
```

Out[163…

```
(None, None, None, None)
```

In [164…

```
#To find the sum of N terms of an AP series for common difference 1.5 and a common rati
def AP(ap_init,n):
    try:
        if n<0:
            print("Please enter a non-negative integer for N")
        else:
            current_term = ap_init                                  #the initial term is s
            ap_total = ap_init                                      #the initial term is a
            common_difference = 1.5
            for k in range(1,n):                                    #the loop runs k-1 tim
                current_term = current_term + common_difference     #the common difference
                ap_total = ap_total + current_term                  #(k+1)th term of the A
            print("The sum of the AP upto " + str(n) + " terms is = " + str(ap_total))
    except TypeError:
        print("Please enter a non-negative integer for N")
```

In [165…

```
AP(0,3), AP(-10,5), AP(5,5), AP(2.5,2), AP(3,0), AP(3,-5), AP(3,2.5)
```

```
The sum of the AP upto 3 terms is = 4.5
The sum of the AP upto 5 terms is = -35.0
The sum of the AP upto 5 terms is = 40.0
The sum of the AP upto 2 terms is = 6.5
The sum of the AP upto 0 terms is = 3
Please enter a non-negative integer for N
Please enter a non-negative integer for N
```

Out[165…

```
(None, None, None, None, None, None, None)
```

In [166…

```
#To find the sum of N terms of a GP series for a common ratio of 0.5
```

```
def GP(gp_init,n):
    try:
        if n<0:
            print("Please enter a non-negative integer for N")
        else:
            current_term = gp_init          #the initial term is set at first as curren
            gp_total = gp_init              #the initial term is added before the for l
            common_ratio = 0.5
            for k in range(1,n):            #the loop runs k-1 times when k>=1
                current_term = current_term*common_ratio     #the current term is multi
                gp_total = gp_total + current_term           #(k+1)th term of the GP is
            print("The sum of the GP upto " + str(n) + " terms is = " + str(gp_total))
    except TypeError:
        print("Please enter a non-negative integer for N")
```

In [167…

```
GP(1,0), GP(2,4), GP(1,53), GP(1,-10), GP(1,2.5)
```

```
The sum of the GP upto 0 terms is = 1
The sum of the GP upto 4 terms is = 3.75
The sum of the GP upto 53 terms is = 1.9999999999999998
Please enter a non-negative integer for N
Please enter a non-negative integer for N
```

Out[167…

```
(None, None, None, None, None)
```

In [168…

```
#To find the sum of N terms of an HP series for a common ratio of 0.5
def HP(hp_init,n):
    try:
        if n<0:
            print("Please enter a non-negative integer for N")
        else:
            current_term = hp_init              #the initial term is set at first as cu
            hp_total = hp_init                  #the initial term is added before the f
            common_ratio = 0.5
            for k in range(1,n):                #the loop runs k-1 times when k>=1
                current_term = current_term/common_ratio        #the current term is di
                hp_total = hp_total + current_term              #(k+1)th term of the HP
            print("The sum of the HP upto " + str(n) + " terms is = " + str(hp_total))
    except TypeError:
        print("Please enter a non-negative integer for N")
```

In [169…

```
HP(1,0), HP(3,4), HP(-2,3), HP(2,3), HP(5,2.5), HP(1,-2)
```

```
The sum of the HP upto 0 terms is = 1
The sum of the HP upto 4 terms is = 45.0
The sum of the HP upto 3 terms is = -14.0
The sum of the HP upto 3 terms is = 14.0
Please enter a non-negative integer for N
Please enter a non-negative integer for N
```

Out[169…

```
(None, None, None, None, None, None)
```

In [170…

```
#To find the factorial of N, where N is a natural number
def factorial(F):
    try:
        if F<0:
            print("Please enter a non-negative integer")
        else:
            fact = 1
            for l in range(1,F+1):                  #l runs from 1 to F
                fact = l*fact                       #fact is multiplied by the index, pr
            print("The factorial of " + str(F) + " is " + str(fact))
    except TypeError:
        print("Please enter a non-negative integer")
```

In [171…

```
factorial(3), factorial(5.6), factorial(10), factorial(-1), factorial(0)
```

```
The factorial of 3 is 6
Please enter a non-negative integer
The factorial of 10 is 3628800
Please enter a non-negative integer
The factorial of 0 is 1
```

Out[171…

```
(None, None, None, None, None)
```

In [172…

```
#To calculate sin(x) and exp(-x) accurate up to 5 decimal places
import math
import matplotlib.pyplot as plt

#General note: unit used for sine function calculations and plots is radians

def factorial_no_print(F):   #define a factorial function that returns a value instead
    try:
        if F<0:
            print("Please enter a non-negative integer")
        else:
            fact = 1
            for l in range(1,F+1):
                fact = l*fact
            return(fact)
    except TypeError:
        print("Please enter a non-negative integer")

def sin_func(x,N):         #define a sine function
    if N <= 0:
        print("Enter a non-negative integer for N")
    else:
        y = 0
        for k in range(N+1):
            y = y + ((-1)**k)*x**(2*k+1)/factorial_no_print(2*k+1)     #
        return y

def exp_negative(x, N):    #define a negative exponential function
    if N <= 0:
        print("Enter a non-negative integer for N")
    else:
        y = 0
        for k in range(N+1):
            y = y + (-x)**k/factorial_no_print(k)
        return y
```

In [173…

```
math.sin(3), sin_func(1.047,3)
```

Out[173…

```
(0.1411200080598672, 0.8659224861546546)
```

In [174…

```
math.sin(10), sin_func(10,3)
```

Out[174…

```
(-0.5440211108893698, -1307.4603174603174)
```

In [175…

```
math.exp(-2), exp_negative(2,10)
```

Out[175…

```
(0.1353352832366127, 0.13537918871252214)
```

In [176…

```
math.exp(-2), exp_negative(2,3)
```

Out[176…

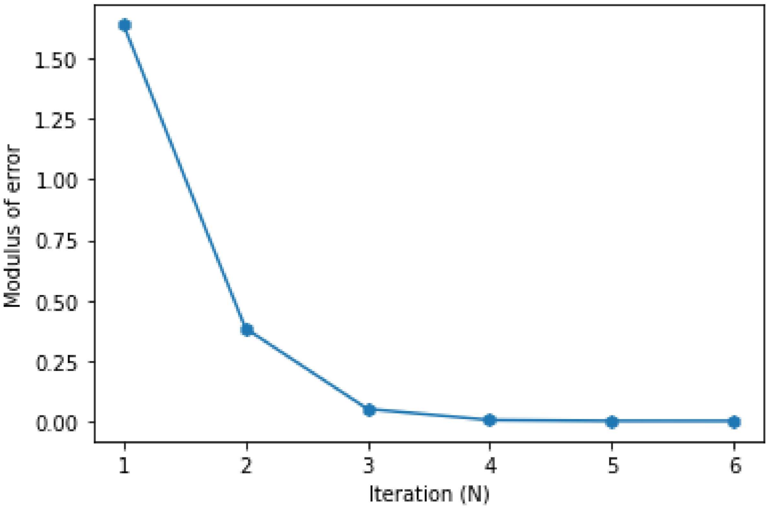
```
(0.1353352832366127, -0.33333333333333326)
```

In [177…

```
#To plot modulus of error versus iteration numbers for sin(x)
x = 3           #as taken above in a test example
list_iter=[]
list_err=[]
digits = 1
while abs(math.sin(x) - sin_func(x, digits))>10**(-5):              #run until 4 decim
    list_iter.append(digits)                                         #generate list for
    list_err.append(abs(math.sin(x) - sin_func(x, digits)))          #generate list for
    digits = digits + 1
print(list_err)

#plot
plt.xlabel("Iteration (N)")
plt.ylabel("Modulus of error")
plt.plot(list_iter,list_err,'-h')
plt.show()
```

```
[1.6411200080598671, 0.38387999194013267, 0.050048579488438744, 0.004192491940132686
6, 0.00024541390402316177, 1.0619125447364208e-05]
```

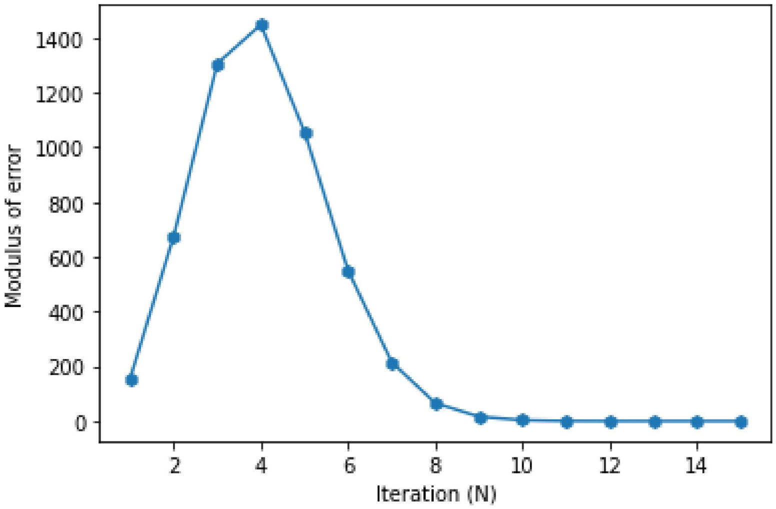


In [178…

```
#To plot modulus of error versus iteration numbers for sin(x)
x = 10           #as taken above in a test example
list_iter=[]
list_err=[]
digits = 1
while abs(math.sin(x) - sin_func(x, digits))>10**(-5):              #run until 4 decim
    list_iter.append(digits)                                         #generate list for
    list_err.append(abs(math.sin(x) - sin_func(x, digits)))          #generate list for
    digits = digits + 1
print(list_err)

#plot
plt.xlabel("Iteration (N)")
plt.ylabel("Modulus of error")
plt.plot(list_iter,list_err,'-h')
plt.show()
```

```
[156.1226455557773, 677.2106877775561, 1306.916296349428, 1448.8156260491612, 1056.39521
24950108, 549.5091711871507, 215.20720199483097, 65.9385234397211, 16.26782902652221, 3.
3051120368690494, 0.5630581337616343, 0.08163689467681312, 0.01020000396114229, 0.001109
9589253054098, 0.00010616611624814087]
```

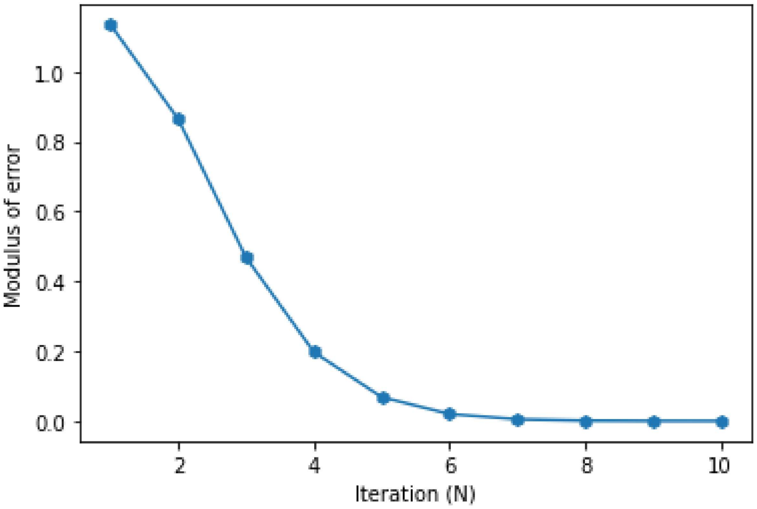


In [179…

```
#To plot modulus of error versus iteration numbers for sin(x)
x = 2            #as taken above in a test example
list_iter=[]
list_err=[]
digits = 1
while abs(math.exp(-x) - exp_negative(x, digits))>10**(-5):              #run until 4
    list_iter.append(digits)                                            #generate list
    list_err.append(abs(math.exp(-x) - exp_negative(x, digits)))         #generate lis
    digits = digits + 1
print(list_err)

#plot
plt.xlabel("Iteration (N)")
plt.ylabel("Modulus of error")
plt.plot(list_iter,list_err,'-h')
plt.show()
```

```
[1.1353352832366128, 0.8646647167633873, 0.46866861656994596, 0.19799805009672067, 0.068
668616569946, 0.02022027231894291, 0.005176553077882479, 0.0011726532713238758, 0.000238
28147294419066, 4.390547590943372e-05]
```

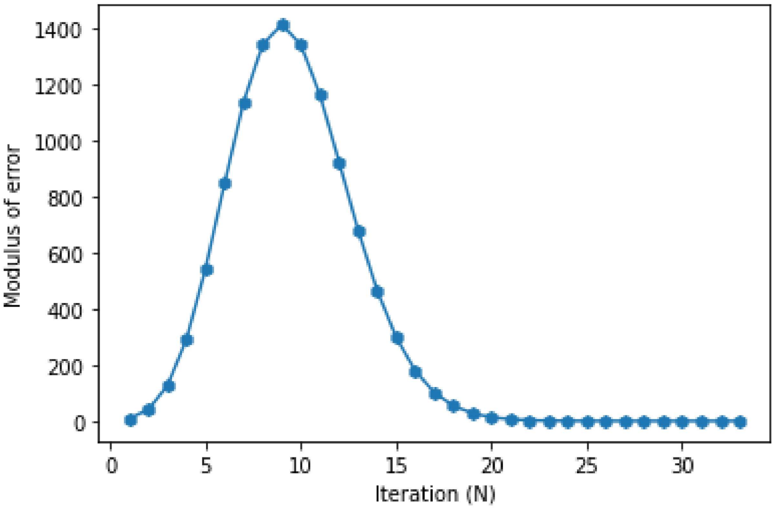


In [180…

```
#To plot modulus of error versus iteration numbers for sin(x)
x = 10          #as taken above in a test example
list_iter=[]
list_err=[]
digits = 1
while abs(math.exp(-x) - exp_negative(x, digits))>10**(-5):           #run until 4
    list_iter.append(digits)                                         #generate list
    list_err.append(abs(math.exp(-x) - exp_negative(x, digits)))       #generate lis
    digits = digits + 1
print(list_err)

#plot
plt.xlabel("Iteration (N)")
plt.ylabel("Modulus of error")
plt.plot(list_iter,list_err,'-h')
plt.show()
```

```
[9.000045399929762, 40.99995460007024, 125.66671206659642, 290.9999546000702, 542.333378
7332631, 846.5555101556258, 1137.5714739713583, 1342.5872561873718, 1413.1446662112173,
1342.5872561873718, 1162.6235823568002, 925.0521164300097, 680.8522672521518, 466.222292
52082053, 298.49408066116115, 179.45365257757737, 101.69207285697466, 54.4999968288876,
27.706355637355703, 13.39682059576595, 6.176120467625312, 2.720670924825262, 1.147499245
8054223, 0.4642383252906961, 0.1804567031477513, 0.06750292317472845, 0.0243339754632270
13, 0.008464916907471363, 0.002845045978976353, 0.0009249416498395525, 0.000291183391713
96544, 8.885568377150888e-05, 2.6307672436210606e-05]
```


Modulus of error
Iteration (N)