In [1]:
```
from math import sin,exp
import matplotlib.pyplot as  plt
```

In [2]:
```
# Defining factorial function
def factorial(N) :
    error_msg = "Please input a positive integer"
    if N >= 0 and isinstance(N,int):   # Checking whether N is natural number
        product = 1
        for i in range(1,N+1):
            product*=i
        return product
    else:
        return error_msg
```

In [3]:
```
# Defining the sin function at x that is accurate up till input digits
def approx_sin(digits,x,Plot = False):
    sin_at_x = 0
    n = 0
    list_iter = []
    list_sine = []
    while abs(sin(x) - sin_at_x) >= 10**(-(digits + 1)): # Ensures that output accur
        sin_at_x += ((-1)**n)*(x**(2*n + 1)/factorial(2*n + 1)) # Recalling the fact
        n += 1
        list_iter.append(n) # List of Iteration Numbers
        list_sine.append(abs(sin(x) - sin_at_x)) # List of Modulus of Error
    dict_sin = {
        'Aboslute value of sin(x)' : sin(x),
        'Caluclated value of sin(x)' : sin_at_x,
        'List of Modulus of Error' : list_sine
    }
    if Plot is True:
        plt.xlabel('Iteration Number (n)')
        plt.ylabel('Modulus of Error')
        plt.plot(list_iter,list_sine, '-h')
        plt.show()
    else:
        return dict_sin
```

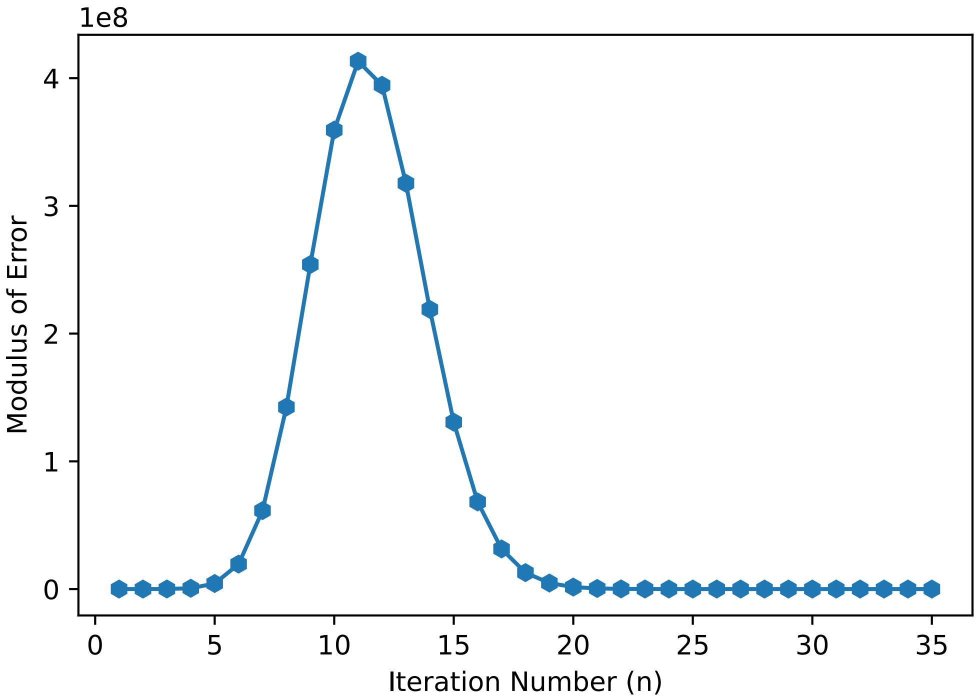
In [4]:
```
approx_sin(4,23)
```

Out[4]:
```
{'Aboslute value of sin(x)': -0.8462204041751706,
 'Caluclated value of sin(x)': -0.8462153313173763,
 'List of Modulus of Error': [23.84622040417517,
  2003.987112929158,
  51632.2045537375,
  623928.4000097547,
  4339565.486297013,
  19530327.839669168,
  61413092.477228984,
  142487237.74962398,
  254068919.19892463,
  359317651.0519005,
  413257338.62116265,
  394434696.03703976,
  317680447.8532753,
  218941932.65666303,
  130655652.67555043,
  68201468.44352585,
  31415403.783738688,
```

```
 12868062.945860423,
 4718989.53158547,
 1558710.0369495347,
 466230.8604620614,
 126899.66818175602,
 31568.033662981066,
 7205.9784905184315,
 1514.8771341419579,
 294.27291505229556,
 52.984115435280586,
 8.867389321597464,
 1.3830636910555703,
 0.20153293188231847,
 0.027497563618813503,
 0.003520647369435026,
 0.0004237357105609796,
 4.812739421466983e-05,
 5.072857794385932e-06]}
```

In [5]:

```
approx_sin(4,23, True)
```



In [6]:

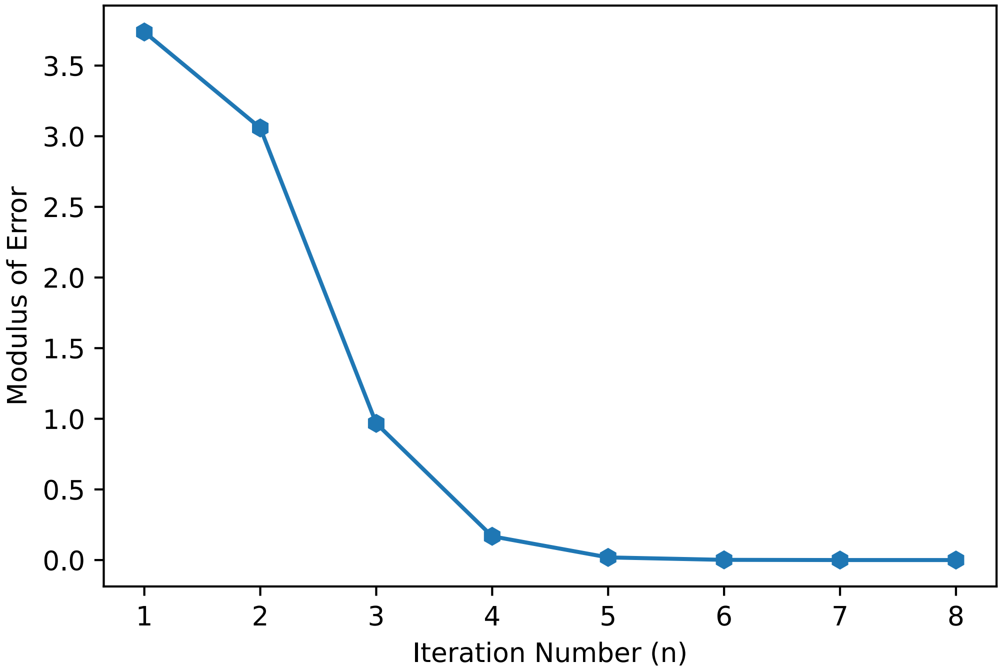
```
approx_sin(4,3.442)
```

Out[6]:

```
{'Aboslute value of sin(x)': -0.29590933502196665,
 'Caluclated value of sin(x)': -0.2959129644164154,
 'List of Modulus of Error': [3.7379093350219668,
 3.0585284796447008,
 0.9674651550413271,
 0.16818751285887296,
 0.01868041122702685,
 0.001445882559973355,
 8.260159685696822e-05,
 3.629394448756429e-06]}
```

In [7]:

```
approx_sin(4,3.442, True)
```



In [8]:

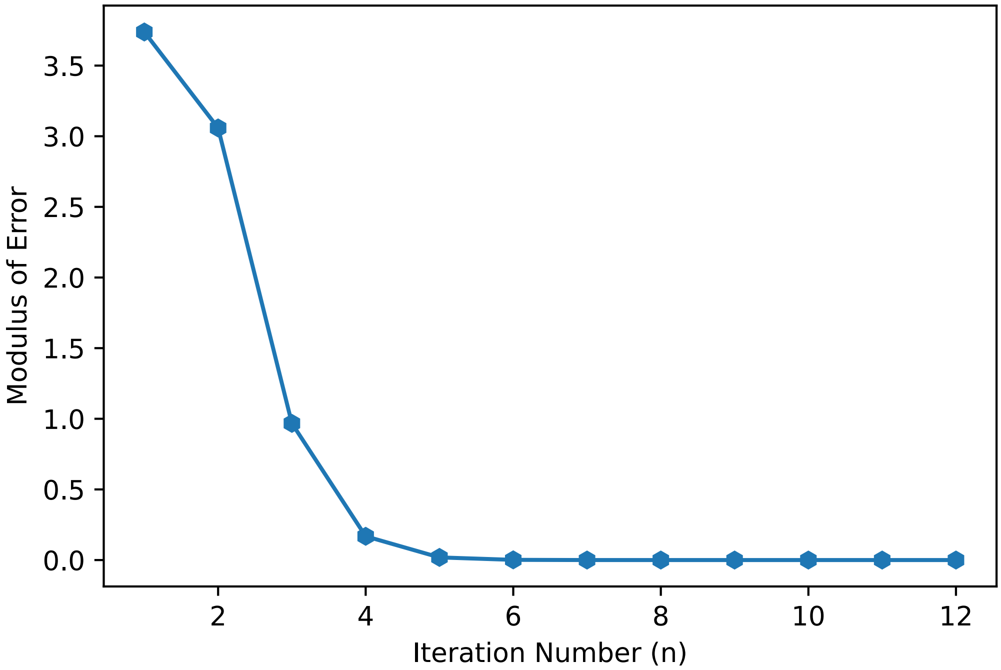
```
approx_sin(10,3.442)
```

Out[8]:

```
{'Aboslute value of sin(x)': -0.29590933502196665,
 'Caluclated value of sin(x)': -0.29590933502363514,
 'List of Modulus of Error': [3.7379093350219668,
  3.0585284796447008,
  0.9674651550413271,
  0.16818751285887296,
  0.01868041122702685,
  0.001445882559973355,
  8.260159685696822e-05,
  3.629394448756429e-06,
  1.265244559678358e-07,
  3.5858903069119208e-09,
  8.426359610069767e-11,
  1.6684986725579165e-12]}
```

In [9]:

```
approx_sin(10,3.442,True)
```



In [10]:

```
# Defining the exp function at -x that is accurate up till input digits
def approx_neg_exp(digits,x,Plot = False):
    exp_at_neg_x = 0
    n = 0
    list_iter = []
    list_exp = []
    while abs(exp(-x) - exp_at_neg_x) >= 10**(-(digits + 1)): # Ensures that output
        exp_at_neg_x += ((-x)**n)/factorial(n) # Recalling the factorial function
        n += 1
        list_iter.append(n) # List of Iteration Numbers
        list_exp.append(abs(exp(-x) - exp_at_neg_x)) # List of Modulus of Error
    dict_exp = {
        'Aboslute value of exp(x)' : exp(-x),
        'Caluclated value of exp(x)' : exp_at_neg_x,
        'List of Modulus of Error' : list_exp
    }
    if Plot is True:
        plt.xlabel('Iteration Number (n)')
        plt.ylabel('Modulus of Error')
        plt.plot(list_iter,list_exp, '-h')
        plt.show()
    else:
        return dict_exp
```

In [11]:

```
approx_neg_exp(4,7)
```

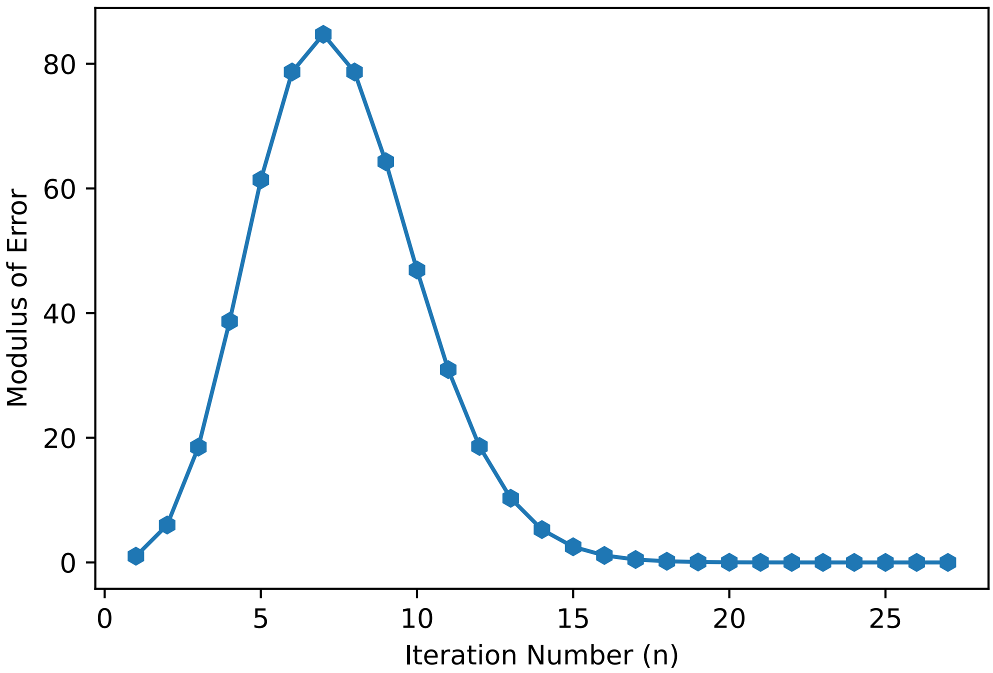
Out[11]:

```
{'Aboslute value of exp(x)': 0.0009118819655545162,
 'Caluclated value of exp(x)': 0.0009167030052464855,
 'List of Modulus of Error': [0.9990881180344455,
  6.000911881965554,
  18.499088118034447,
  38.66757854863222,
  61.37408811803445,
  78.68424521529889,
  84.71714367359002,
  78.68424521529889,
  64.2919700624789,
  46.91175293134827,
  30.930853164330752,
  18.605350714737707,
```

```
 10.290768214718893,
 5.268680439603892,
 2.5110438875575007,
 1.119494131784482 6,
 0.4688662516776351,
 0.18516449445382516,
 0.06918079570840942,
 0.024525363825045415,
 0.008271792011663772,
 0.0026605932672392908,
 0.0008178929578662298,
 0.00024077676281805 91,
 6.800190571485851e-05,
 1.8456121474358378e-05,
 4.8210396919692346e-06]}
```

In [12]:

```
approx_neg_exp(4,7, True)
```



In [13]:

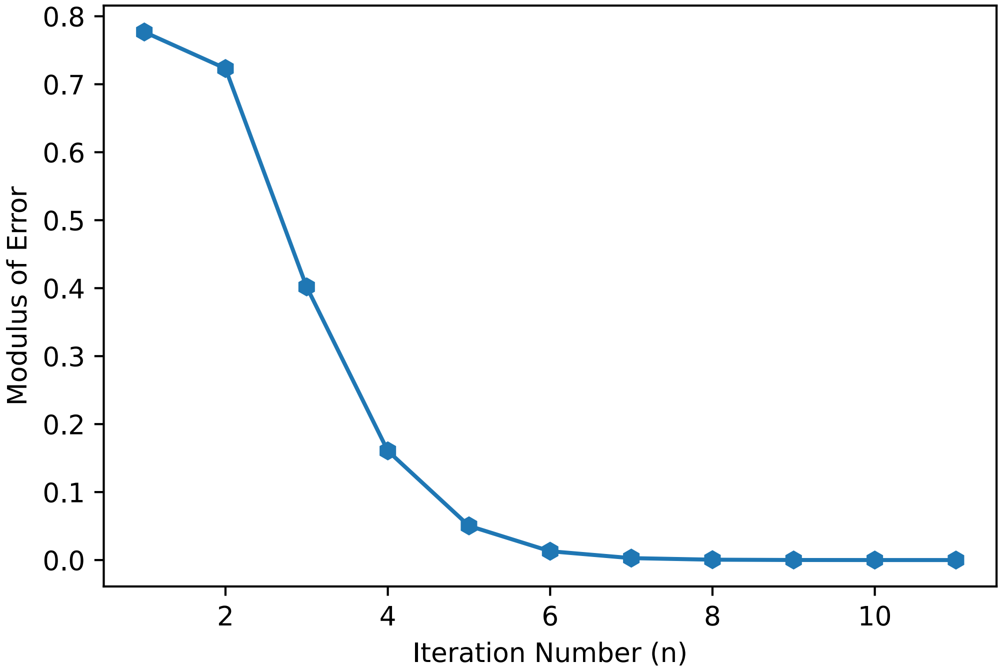
```
approx_neg_exp(4,1.5)
```

Out[13]:

```
{'Aboslute value of exp(x)': 0.22313016014842982,
 'Caluclated value of exp(x)': 0.223132084437779,
 'List of Modulus of Error': [0.7768698398515702,
  0.7231301601484298,
  0.4018698398515702,
  0.16063016014842982,
  0.05030733985157018,
  0.01297391014842983,
  0.0028464023515701597,
  0.0005436646127155442,
  9.197294308802006e-05,
  1.3966649545921195e-05,
  1.924289349169994e-06]}
```

In [14]:

```
approx_neg_exp(4,1.5,True)
```



In [15]:

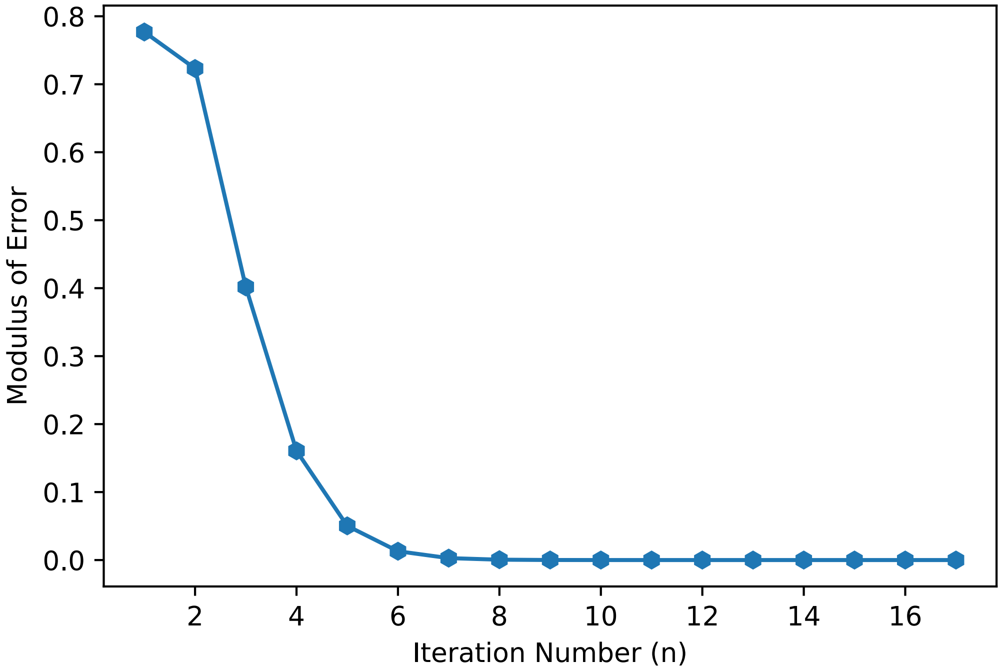
```
approx_neg_exp(10,1.5)
```

Out[15]:

```
{'Aboslute value of exp(x)': 0.22313016014842982,
 'Caluclated value of exp(x)': 0.2231301601509859,
 'List of Modulus of Error': [0.7768698398515702,
  0.7231301601484298,
  0.4018698398515702,
  0.16063016014842982,
  0.05030733985157018,
  0.012973910148429832,
  0.0028464023515701597,
  0.0005436646127155442,
  9.197294308802006e-05,
  1.3966649545921195e-05,
  1.924289349169994e-06,
  2.4265686379698614e-07,
  2.82114128169475e-08,
  3.042619101112365e-09,
  3.0602717582262073e-10,
  2.8837460197550513e-11,
  2.5560942251701135e-12]}
```

In [16]:

```
approx_neg_exp(10,1.5,True)
```


Modulus of Error
Iteration Number (n)
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
2
4
6
8
10
12
14
16