In [5]:

```
import math
import matplotlib.pyplot as graph

def fact(p):       #defining factorial function for later use
    u=1
    for i in range(1,p+1):
        u=u*i
    return u
```

In [6]:

```
# Defining our sine function

def sin_val(x,n):
    if n <= 0 and math.floor(n) != n:         #check for undesired input
        print("Please enter a positive natural number.")
    else:
        t = 0
        for k in range(n+1):
            t = t + ((-1)**k)*x**(2*k+1)/(fact(2*k+1))  #taylor series expansion of
        return t
print(sin_val(2,3))
```

```
0.9079365079365079
```

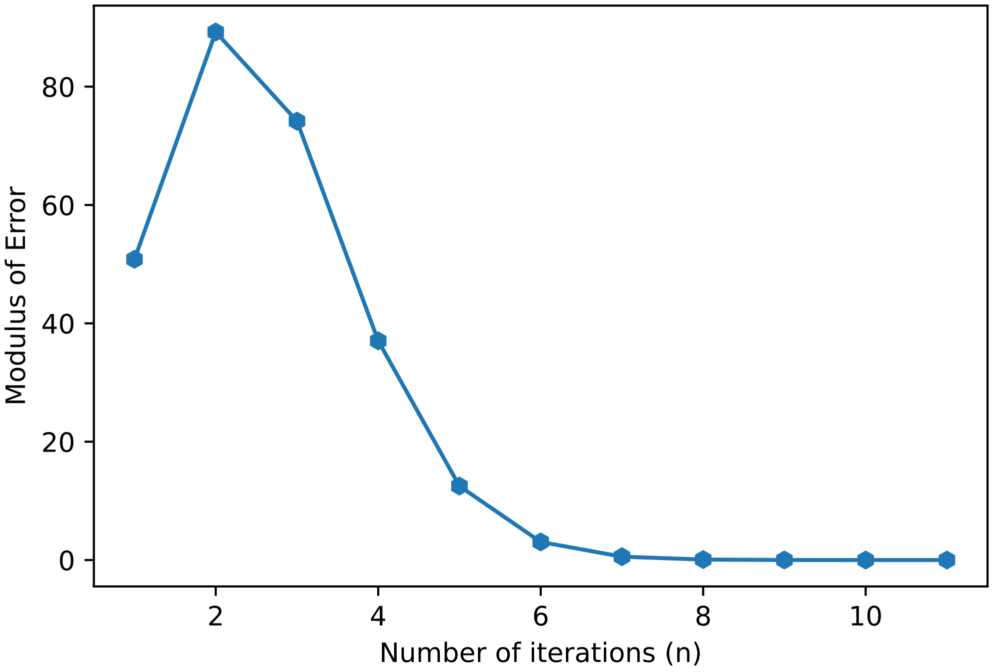
In [21]:

```
# Saving the data into lists
x = 7
l1=[]
l2=[]
b = 1
while abs(math.sin(x) - sin_val(x, b)) > 10**(-5): # condition for accuracy upto 5th
    l1.append(b)
    l2.append(abs(math.sin(x) - sin_val(x, b)))
    b = b + 1
print(l2)

#graphing the values

graph.xlabel("Number of iterations (n)")
graph.ylabel("Modulus of Error")
graph.plot(l1,l2,'-0')
graph.show()
```

```
[50.82365326538545, 89.23468006794789, 74.16670882094101, 37.037014172886146, 12.499
189706182314, 3.060258948140471, 0.5702790712015124, 0.08375167492994784, 0.00995448
4603507008, 0.0009779006753960484, 8.076904528819817e-05]
```



In [42]:

```
print(sin_val(7,100))
print(math.sin(7))
print(abs(math.sin(7)-sin_val(7,100)))
```

```
0.656986598718787
0.6569865987187891
2.1094237467877974e-15
```

In [50]:

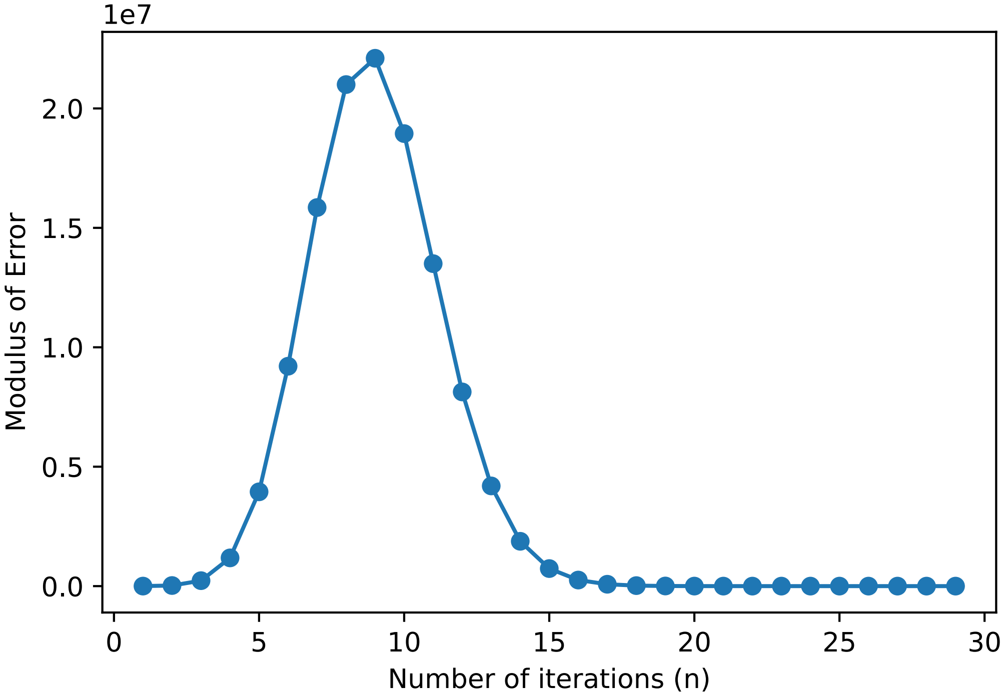
```
# Changing the input value and checking

x = 20
l1=[]
l2=[]
b = 1
while abs(math.sin(x) - sin_val(x, b)) > 10**(-5):
    l1.append(b)
    l2.append(abs(math.sin(x) - sin_val(x, b)))
    b = b + 1
print(l2)

# Plotting as a graph

graph.xlabel("Number of iterations (n)")
graph.ylabel("Modulus of Error")
graph.plot(l1,l2,"-o")
graph.show()
```

```
[1314.246278584061, 25352.42038808261, 228615.83358017134, 1182318.9106879062, 39483
52.886650558, 9207215.82447371, 15851010.291953465, 20999322.23220414, 22100481.8896
1763, 18946950.607355483, 13501612.631358435, 8130762.861117513, 4195377.020635164,
1876613.0688981742, 734995.5717613262, 254250.12545818152, 78269.43663240933, 21586.
28771911945, 5365.324791414636, 1208.2392355448965, 247.6996408581007, 46.4294250818
987, 7.988533463892129, 1.2662213908342022, 0.18550486088757534, 0.025195611060433,
0.003181556878692704, 0.00037447920390454303, 4.118836857902597e-05]
```



In [43]:

```
print(sin_val(20,100))
print(math.sin(20))
print(abs(math.sin(20)-sin_val(20,100)))
```

```
0.9129452484183126
0.9129452507276277
2.309315072501761e-09
```

In [27]:

```
# Defining our exponential of negative x function

def exp_val(x,n):
    if n <= 0 and math.floor(n) != n:
        print("Please enter a positive natural number.")
    else:
        p=0
        for k in range(n+1):
            p = p + (-x)**k/fact(k)
        return p
```

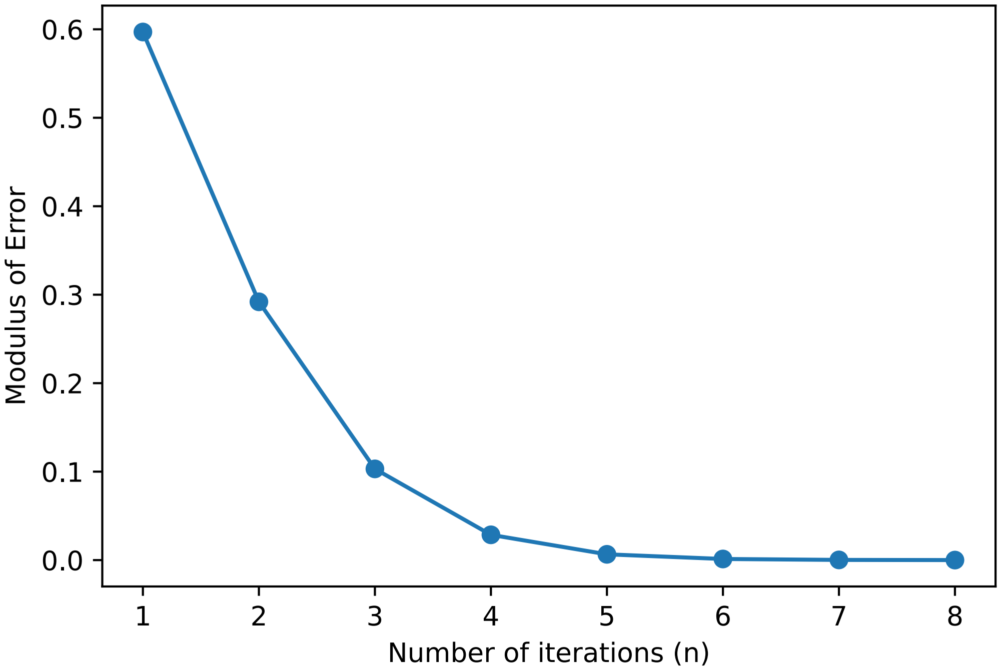
In [36]:

```
# Graphing for one value

x = 4/3
l1=[]
l2=[]
t = 1
while abs(math.exp(-x) - exp_val(x, t))>10**(-5):
    l1.append(t)
    l2.append(abs(math.exp(-x) - exp_val(x, t)))
    t = t + 1
print(l2)

graph.xlabel("Number of iterations (n)")
graph.ylabel("Modulus of Error")
graph.plot(l1,l2,"-o")
graph.show()
```

```
[0.59693047144906, 0.2919584174398288, 0.10310331095523284, 0.02858393184312108, 0.0
065326662364399435, 0.0012710222256847037, 0.00021539462424380318, 3.234151741093871
4e-05]
```



In [48]:

```
print(math.exp(-4/3))
print(exp_val(4/3,10))
print(abs(math.exp(-4/3)-exp_val(4/3,10)))
```

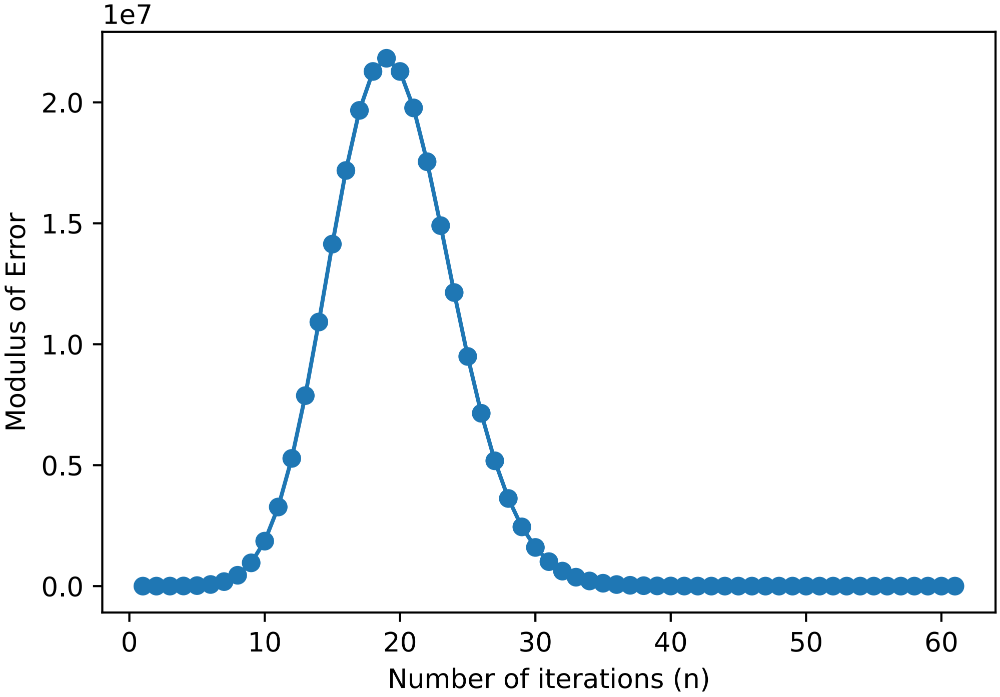
```
0.26359713811572677
0.26359767153593755
5.334202107798447e-07
```

In [40]:

```
# Putting another value
x = 20
l1=[]
l2=[]
t = 1
while abs(math.exp(-x) - exp_val(x, t))>10**(-5):
    l1.append(t)
    l2.append(abs(math.exp(-x) - exp_val(x, t)))
    t = t + 1
print(l2)

graph.xlabel("Number of iterations (n)")
graph.ylabel("Modulus of Error")
graph.plot(l1,l2,"-o")
graph.show()
```

```
[19.000000002061153, 180.99999999793886, 1152.3333333353944, 5514.333333331273, 2115
2.3333333354, 67736.5555555535, 186231.69841270047, 448688.9365079345, 962245.807760
1432, 1859623.6807760121, 3271048.116562452, 5280071.545668321, 7875497.165455947, 1
0918172.421864433, 14140053.694562742, 17182728.950971223, 19667603.573186383, 21277
210.342544295, 21822593.779277474, 21277210.342544295, 19770222.154428817, 17545625.
570092194, 14902937.668621724, 12137531.69697321, 9494843.795502739, 7145445.0448633
75, 5180694.836889302, 3623690.7929340377, 2448299.2965993006, 1599694.096422925, 10
11914.5442365755, 620340.8561756122, 368904.8410438954, 213004.39261463846, 119515.1
6947595237, 65217.92057437587, 34637.8037771529, 17917.840618388556, 9033.7718921455
25, 4442.03436312 1516, 2131.5296638380164, 998.738920428428, 457.199955974 5693, 204.
5904423904294 2, 89.53862354957, 38.34357903303844, 16.07437951275239, 6.599769881327
123, 2.6549849733992077, 1.0469169684913247, 0.4048092832304528, 0.1535469674317694,
0.05715350451623895, 0.020883707316356734, 0.007493460622768969, 0.002641242212633 06
8, 0.0009147938699641378, 0.0003114254688624849, 0.00010424210362111601, 3.431375387
3417625e-05, 1.1114396124790122e-05]
```


1e7
2.0
1.5
1.0
0.5
0.0
Modulus of Error
0
10
20
30
40
50
60
Number of iterations (n)