Student Name: Rosmit Gyawali
Student ID: 22069030

Library import

In [43]:
```
import numpy as np
import matplotlib.pyplot as plt
import sympy
```

Question 1

For what values of 1 ≤ n ≤ 100, the formula 2^n + 3^n produces a number which is divisible by 7?

In [47]:
```
numbers_divisible_by_7 = []
for i in range(1, 101):
    eqn = 2**i + 3**i
    if eqn % 7 == 0:
        numbers_divisible_by_7.append(i)
numbers_divisible_by_7
```

Out[47]:
```
[3, 9, 15, 21, 27, 33, 39, 45, 51, 57, 63, 69, 75, 81, 87, 93, 99]
```

So the numbers in above list are the numbers whuch are divisible by 7 in the given range.

Question 2

Find all numbers between 1 and 500 which have the following property: if n = d1d2 · · · dk then n = d1^3 + d2^3+....+dk^3 (e.g. 153 = 1^3 + 5^3 + 3^3).

In [45]:
```
numbers_with_property = []
for i in range(1, 501):
    str_i = str(i) #changing number to string
    sum=0
    for n in str_i:
        a = int(n) #changing back to integer
        sum+= a**3
    if sum == i:
        numbers_with_property.append(i)
```

In [46]:
```
numbers_with_property
```

Out[46]:
```
[1, 153, 370, 371, 407]
```

So the following numbers above have the asked property within the given range.

Question 3

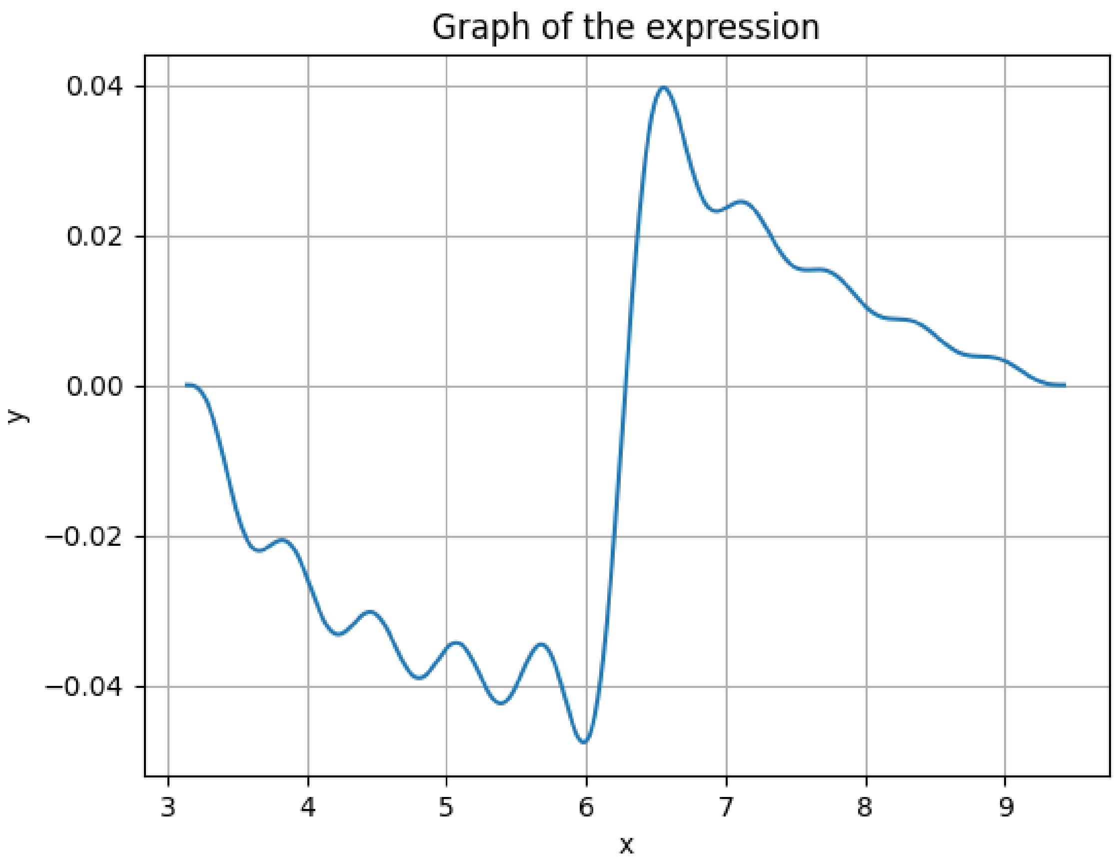
Plot the graph of the expression: sumation(upper 10, lower(n)=1) sin(n*x)/n*x^2 for π ≤ x ≤ 3π.

In [16]:

```
# Define the range for x values
x = np.linspace(np.pi, 3*np.pi, 1000)

# Calculate the corresponding y values using the given expression
y = np.array([sum(np.sin(n*x)/(n*x**2) for n in range(1, 11)) for x in

# Plot the graph
plt.plot(x, y)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Graph of the expression')
plt.grid(True)
plt.show()
```



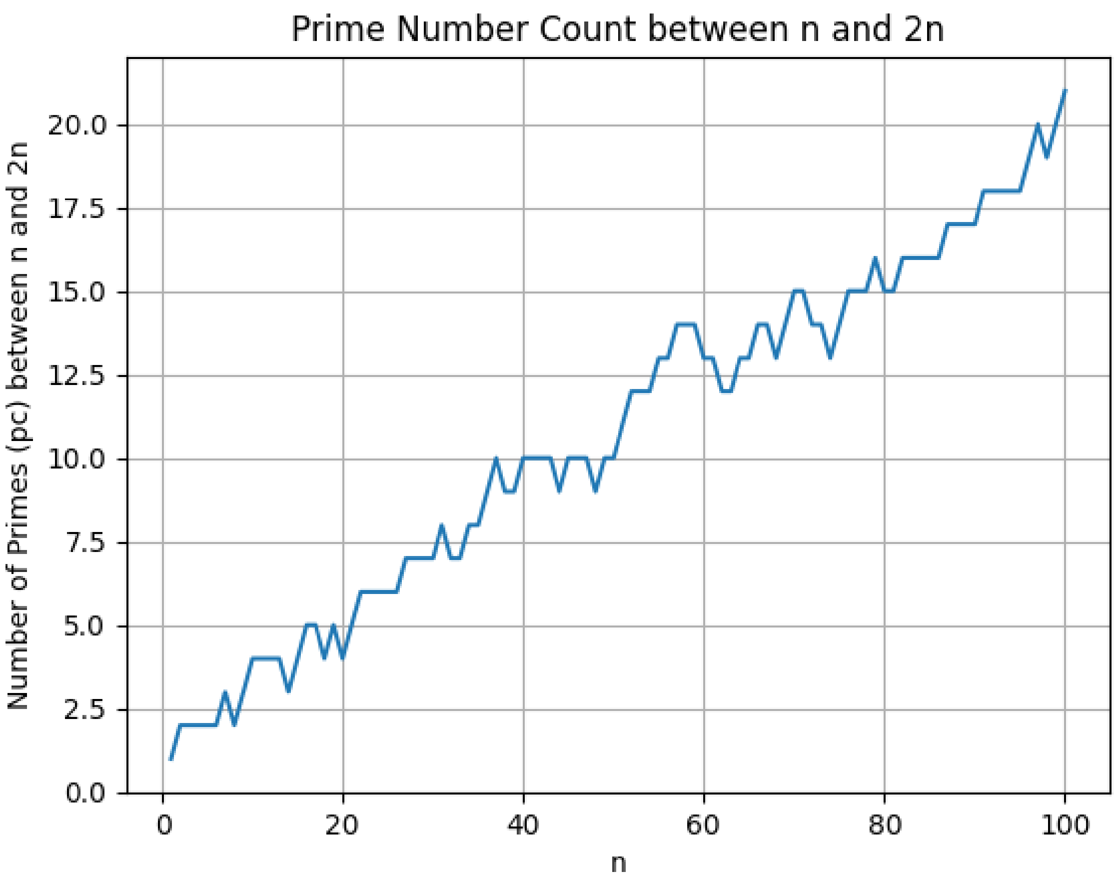
Question 4

It is a conjecture that there is a prime number between n and 2n, where n is a positive number. For example, between 5 and 10 there is at least one prime number i.e., 5 or 7. For n between 1 and 100, plot the graph of (n, pc) where pc is the number of primes between n and 2n (count n if it is prime itself).

In [42]:
```python
def primes_count_between_n_and_2n(n):
    count = 0
    for i in range(n, 2 * n + 1):
        if sympy.isprime(i):
            count += 1
    return count

n_values = list(range(1, 101))
prime_counts = [primes_count_between_n_and_2n(n) for n in n_values]

plt.plot(n_values, prime_counts)
plt.xlabel('n')
plt.ylabel('Number of Primes (pc) between n and 2n')
plt.title('Prime Number Count between n and 2n')
plt.grid(True)
plt.show()
```



Question 5

Let A = (aij) denote the 4 × 4 matrix with aij = x^i + x^j + x^ij What is the determinant of A?

In [28]:
```
#function to calc determinant of A
a = np.zeros((4, 4))
def determinant_of_A(x):
    for i in range(4):
        for j in range(4):
            a[i, j] = x**i + x**j + x**(i*j)

    det_A = np.linalg.det(a)
    return det_A
```

In [29]:
```
# if x = sqrt(2) the determinant:
determinant_A = determinant_of_A(math.sqrt(2))
```

In [30]:
```
#if if x = sqrt(2) the inverse of matrix A:
if determinant_A != 0:  # Check if the determinant is non-zero before f
    # Calculate the inverse of A
    inverse_A = np.linalg.inv(a)
else:
    inverse_A = None  # If the determinant is zero, the inverse doesn't
```

In [31]:
```
inverse_A
```

Out[31]:
```
array([[ 11.10993609, -17.25874045,   8.24264069,  -1.32037724],
       [-17.25874045,  30.94307873, -16.98528137,   2.91421356],
       [  8.24264069, -16.98528137,  10.80330086,  -2.06066017],
       [ -1.32037724,   2.91421356,  -2.06066017,   0.46682385]])
```

In [32]:
```
determinant_A
```

Out[32]:
```
1.3440123022848764
```

So for the matrix A, we've calculated its inverse and determinants

Question 6

Let Bn denote that n × n matrix with (i, j)'th entry equal to

Define a function B(n) in Python to generate this matrix for any n. Look at the numerical values of the determinant of Bn for 3 ≤ n ≤ 30 and display these values graphically. You should be able to see a pattern emerging from the graph. From the graph, can you predict what would be the det(B(100000))?

In [51]:
```
def B(n):
    # Create an empty n by n matrix
    mat = np.empty([n, n])
    # Loop through the rows and columns of the matrix
    for i in range(1, n + 1):
        for j in range(1, n + 1):
            # Check if the row and column indices are not equal
            if i > j:
                # Calculate the off-diagonal elements
                mat[i - 1, j - 1] = 1 / ((2*j)-i**2)
            elif j>i:
                mat[i - 1, j - 1] =  (1/(i-j)) + (1/((n**2)-j-i))
    return mat
```

In [52]:
```
#checking for B(3)
B(3)
```

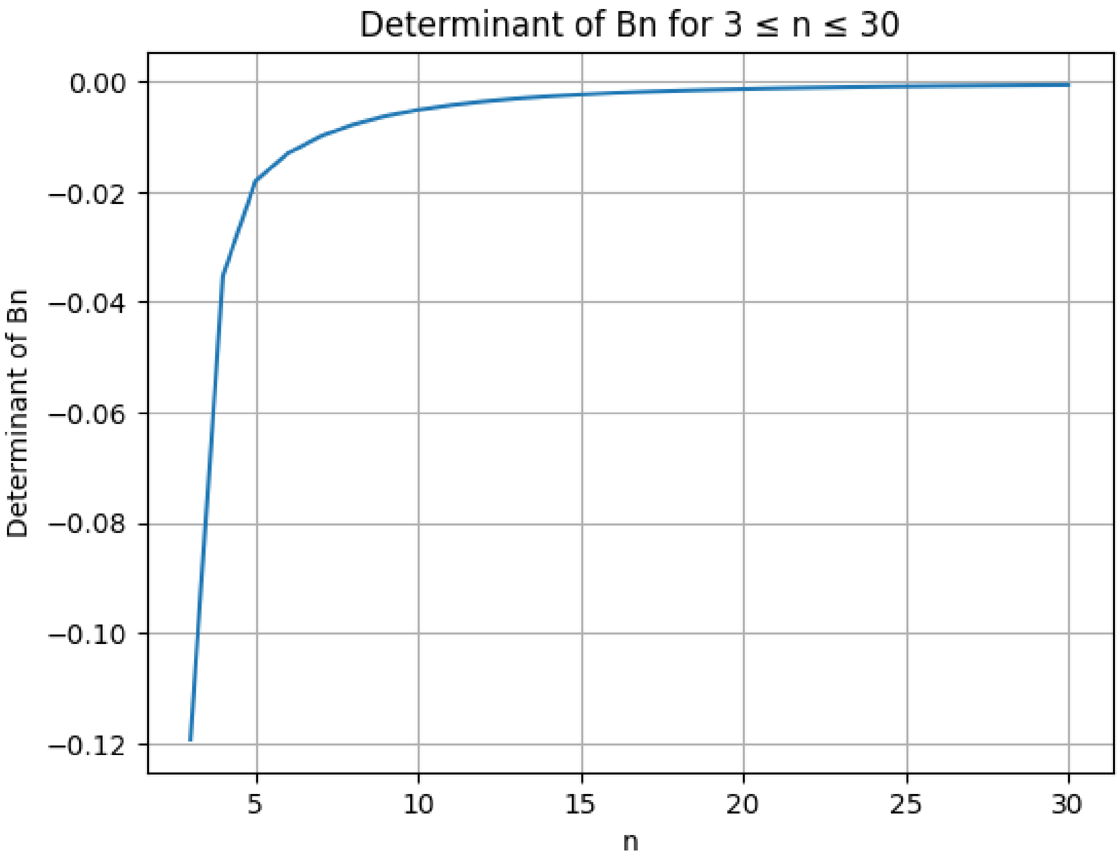
Out[52]:
```
array([[ 0.01      , -0.83333333, -0.3       ],
       [-0.5       ,  0.01      , -0.75      ],
       [-0.14285714, -0.2       ,  1.        ]])
```

In [41]:
```
determinants = [np.linalg.det(B(n)) for n in range(3, 31)]
determinants
```

Out[41]:
```
[-0.11928571428571424,
 -0.03525757631818237,
 -0.018062205431347532,
 -0.013023479578175417,
 -0.009958843179948986,
 -0.007821845590148313,
 -0.0062884030416988125,
 -0.005158398365984695,
 -0.004304481161370978,
 -0.0036446279221737836,
 -0.0031247089104408155,
 -0.0027080421624650643,
 -0.0023691349450837445,
 -0.0020898586035864126,
 -0.0018570517180940616,
 -0.0016609799979487628,
 -0.0014943228129644334,
 -0.0013514914085016741,
 -0.0012281608848641343,
 -0.0011209429257580908,
 -0.0010271530498695712,
 -0.0009446425065400807,
 -0.0008716751273363096,
 -0.0008068359271461349,
 -0.0007489624490510942,
 -0.0006970926172696754,
 -0.0006504247189354632,
 -0.0006082863985421253]
```

In [42]:

```
#plotting the graph
n_values = list(range(3, 31)) # Values for n and x
# Plot the determinants
plt.plot(n_values, determinants)
plt.xlabel('n')
plt.ylabel('n_values')
plt.title('Determinant of Bn for 3 ≤ n ≤ 30')
plt.grid(True)
plt.show()
```



Observing the pattern of the graph it looks like as the value of n increase the determinant of Bn tends to zero. Thus, we can say the determinant of det(B(100000)) would be zero.