Library Imports

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

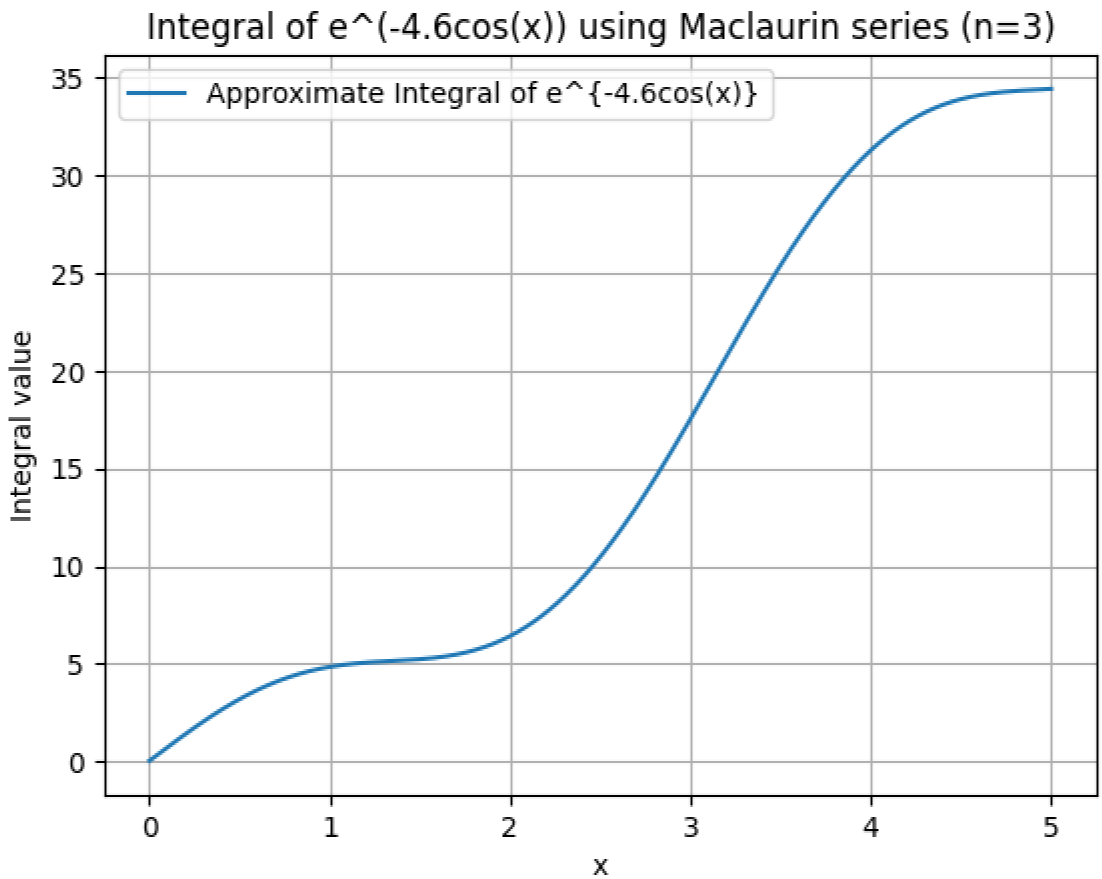
Maclaurin Series

$$e^{-4.6\cos(x)} \approx 1 - 4.6\sin(x) + \frac{(-4.6\cos(x))^2}{2!} + \frac{(-4.6\cos(x))^3}{3!} + \dots$$

let first try an 'n' value close to my hand calculation

In [ ]:
```
n = 3
# Maclaurin series approximation up to n = 10 for e^f(x)
def maclaurin_approximation(x):
    terms = [(-4.6 * np.cos(x))**i / math.factorial(i) for i in range(n)]
    return sum(terms)

x_values = np.linspace(0, 5, 1000)
y_values = maclaurin_approximation(x_values)

#rather than storing the outputs in an array and adding them we can just use a cumu
integral_values = np.cumsum(y_values) * (x_values[1] - x_values[0])

# Print the final value of the integral over the interval [0, 5]
print(f"Approximate value of the integral from 0 to 5 using {n} terms: {integral_va

# Plot the integral
plt.plot(x_values, integral_values, label="Approximate Integral of e^{-4.6cos(x)}")
plt.title(f"Integral of e^(-4.6cos(x)) using Maclaurin series (n={3})")
plt.xlabel("x")
plt.ylabel("Integral value")
plt.legend()
plt.grid(True)
plt.show()

print(integral_values[-1])
```

```
Approximate value of the integral from 0 to 5 using 3 terms: 34.44095363111059
```




```
34.44095363111059
```

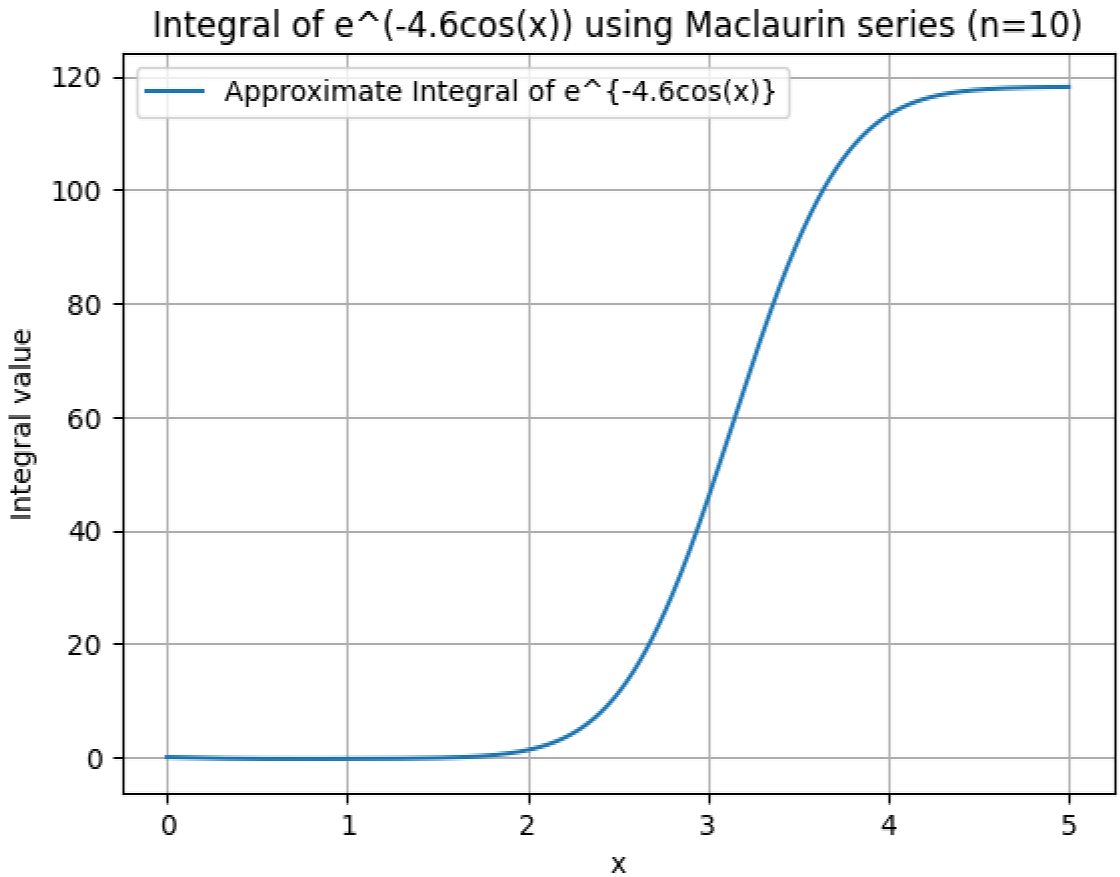
This is close to what I got in my hand calculation. Now lets step up the number of intervals!

Alt text

In [ ]:
```
n = 10
# Maclaurin series approximation up to n = 10 for e^f(x)
def maclaurin_approximation(x):
    terms = [(-4.6 * np.cos(x))**i / math.factorial(i) for i in range(n)]
    return sum(terms)

# Integrate using numerical integration (trapezoid method)
x_values = np.linspace(0, 5, 1000)
y_values = maclaurin_approximation(x_values)

integral_values = np.cumsum(y_values) * (x_values[1] - x_values[0])

# Print the final value of the integral over the interval [0, 5]
print(f"Approximate value of the integral from 0 to 5 using 10 terms: {integral_val

# Plot the integral
plt.plot(x_values, integral_values, label="Approximate Integral of e^{-4.6cos(x)}")
plt.title(f"Integral of e^(-4.6cos(x)) using Maclaurin series (n={n})")
plt.xlabel("x")
plt.ylabel("Integral value")
plt.legend()
```

```
plt.grid(True)
plt.show()

print(integral_values[-1])
```

```
Approximate value of the integral from 0 to 5 using 10 terms: 118.10686276153056
```




```
118.10686276153056
```

In [ ]:
```
# Define the function to integrate
def integrand(t, b):
    return np.exp(-b * np.cos(t))

# Calculate I(4.6)
num_intervals = 100  # Using 100 intervals for a good approximation
b_value_to_shade = 4.6
x_values = np.linspace(0, 5, num_intervals + 1)
h = x_values[1] - x_values[0]
y_values = integrand(x_values, b_value_to_shade)
I_4_6 = (h / 3) * (y_values[0] + 4 * np.sum(y_values[1:-1:2]) + 2 * np.sum(y_values
print(f"I(4.6) = {I_4_6:.4f}")

# Calculate I(b) for each b using Simpson's rule
b_values = np.arange(0, 5.1, 0.1)
I_values = []

for b in b_values:
    y_values = integrand(np.linspace(0, 5, num_intervals + 1), b)
    I = (h / 3) * (y_values[0] + 4 * np.sum(y_values[1:-1:2]) + 2 * np.sum(y_values
    I_values.append(I)
```

```
# Plot the results
plt.plot(b_values, I_values)
plt.xlabel('b')
plt.ylabel('I(b)')
plt.title('Plot of I(b) using Simpson\'s rule')
plt.grid(True)
plt.show()
```

```
I(4.6) = 119.8920
```