Library Imports

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

Newtons Law of Cooling

$dT/dt = -k * (T - Ta)$

In [ ]:
```
# Parameters
k = 0.1  # Cooling constant
Ta = 25  # Ambient temperature (degrees Celsius)

# Initial conditions
T0 = 100  # Initial temperature (degrees Celsius)
t0 = 0    # Initial time
tf = 10   # Final time

# Time step and number of steps
dt = 0.1
num_steps = int((tf - t0) / dt)

# Arrays to store results
time_euler = np.zeros(num_steps + 1)
temp_euler = np.zeros(num_steps + 1)
time_predictor_corrector = np.zeros(num_steps + 1)
temp_predictor_corrector = np.zeros(num_steps + 1)
```

In [ ]:
```
# Euler's method
time_euler[0] = t0
temp_euler[0] = T0
for i in range(num_steps):
    time_euler[i + 1] = time_euler[i] + dt
    temp_euler[i + 1] = temp_euler[i] - k * (temp_euler[i] - Ta) * dt

# Predictor-Corrector (Improved Euler) method
time_predictor_corrector[0] = t0
temp_predictor_corrector[0] = T0
for i in range(num_steps):
    time_predictor_corrector[i + 1] = time_predictor_corrector[i] + dt
    # Predictor step
    predictor_temp = temp_predictor_corrector[i] - k * (temp_predictor_corrector[i]
    # Corrector step
    temp_predictor_corrector[i + 1] = temp_predictor_corrector[i] - 0.5 * k * ((tem
```

In [ ]:
```
# Plot results
plt.figure(figsize=(10, 6))
plt.plot(time_euler, temp_euler, label="Euler's Method")
plt.plot(time_predictor_corrector, temp_predictor_corrector, label="Predictor-Corre
```

```
plt.xlabel("Time")
plt.ylabel("Temperature (°C)")
plt.title("Cooling of an Object")
plt.legend()
plt.grid(True)
plt.show()
```