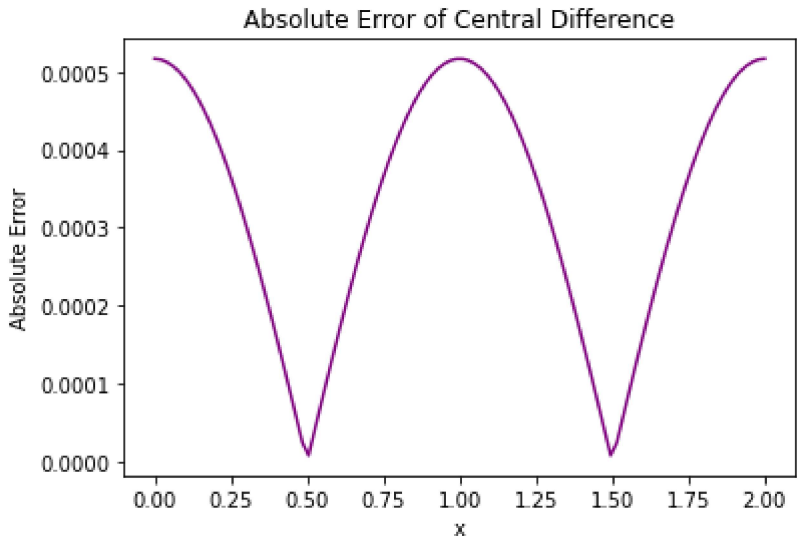
Computational Physics: Assignment 2 - Derivatives

In [4]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Problem 1:

In [8]:
```
def f(x):
    return np.sin(np.pi * x)

def true_dx(x):
    return np.pi * np.cos(np.pi * x)

def central_difference(x, h):
    return (f(x + h) - f(x - h)) / (2 * h)

x_vals = np.linspace(0, 2, 100)
h = 0.01

# call both funcs
approx_derivative = central_difference(x_vals, h)
true_derivative_values = true_dx(x_vals)

# get err
error = np.abs(approx_derivative - true_derivative_values)

#plotting error
plt.plot(x_vals, error, color = 'purple')
plt.xlabel('x')
plt.ylabel('Absolute Error')
plt.title('Absolute Error of Central Difference')

max_error = np.max(error)
max_error_x = x_vals[np.argmax(error)]
min_error = np.min(error)
min_error_x = x_vals[np.argmin(error)]
print(f" Maximum error = {max_error} ")
print(f" Minimum error = {min_error} ")

plt.show()
```

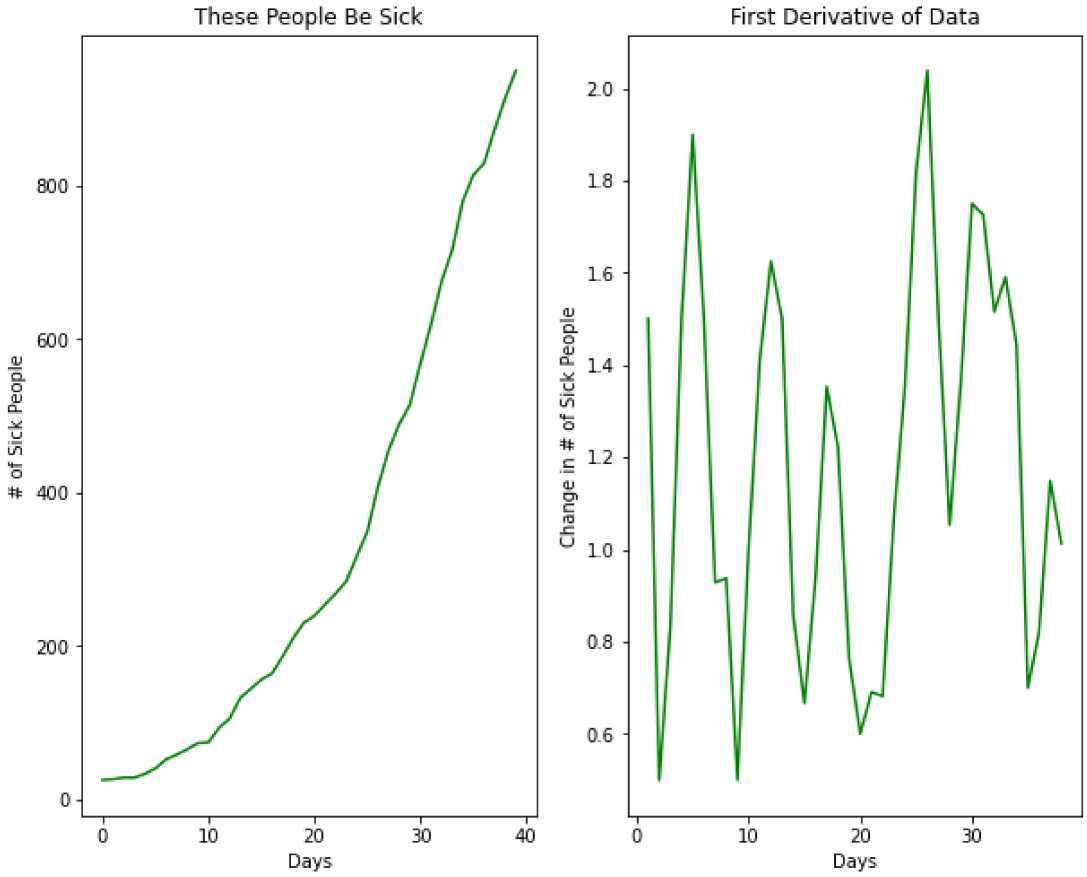
```
Maximum error = 0.0005167457770265393 at x = 2.0
Minimum error = 8.198669804795888e-06 at x = 1.494949494949495
```



Problem 2:

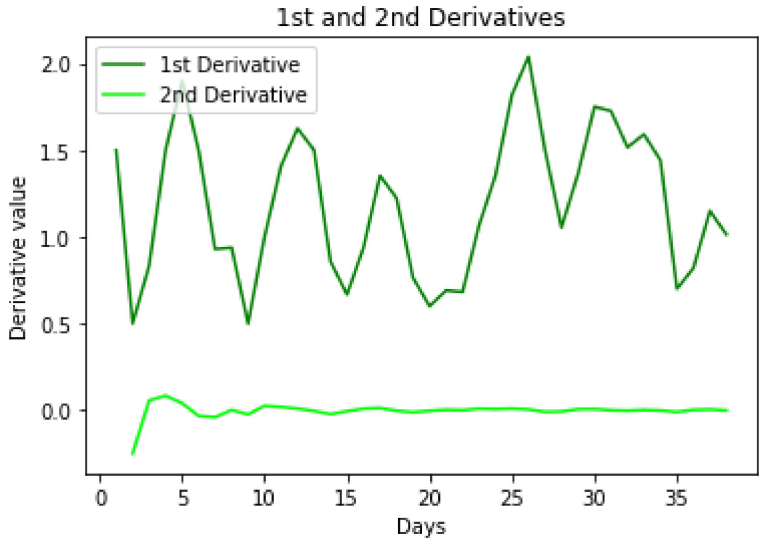
In [17]:
```
# part a
sick = np.array([25, 26, 28, 28, 33, 40, 52, 58, 65, 73, 74, 93, 105, 13
                 156, 164, 186, 210, 230, 239, 254, 268, 284, 317, 349,
                 489, 514, 568, 619, 675, 716, 780, 814, 829, 873, 914,

days = np.array(range(0, 40))

del_days = days[1:-1]
del_sick = sick[2:] - sick[:-2]
first_derivative = del_sick / (2*del_days)

plt.figure(figsize=(10, 8))
plt.subplot(1, 2, 1)
plt.plot(days, sick, color ='green')
plt.xlabel('Days')
plt.ylabel('# of Sick People')
plt.title('These People Be Sick')

plt.subplot(1, 2, 2)
plt.plot(del_days, first_derivative, label='1st dx : Central Difference'
plt.xlabel('Days')
plt.ylabel('Change in # of Sick People')
plt.title('First Derivative of Data')
```

Out[17]: Text(0.5, 1.0, 'First Derivative of Data')



In [16]:
```
# part b
#second derivative using the central difference method
delta_first_derivative = first_derivative[1:] - first_derivative[:-1]
second_derivative = delta_first_derivative / (2 * del_days[1:])

#plotting second derivative
plt.plot(del_days, first_derivative, label='1st Derivative', color = 'gr
plt.plot(del_days[1:], second_derivative, label='2nd Derivative', color
plt.xlabel('Days')
plt.ylabel('Derivative value')
plt.legend()
plt.title('1st and 2nd Derivatives')
```

Out[16]:
```
Text(0.5, 1.0, '1st and 2nd Derivatives')
```

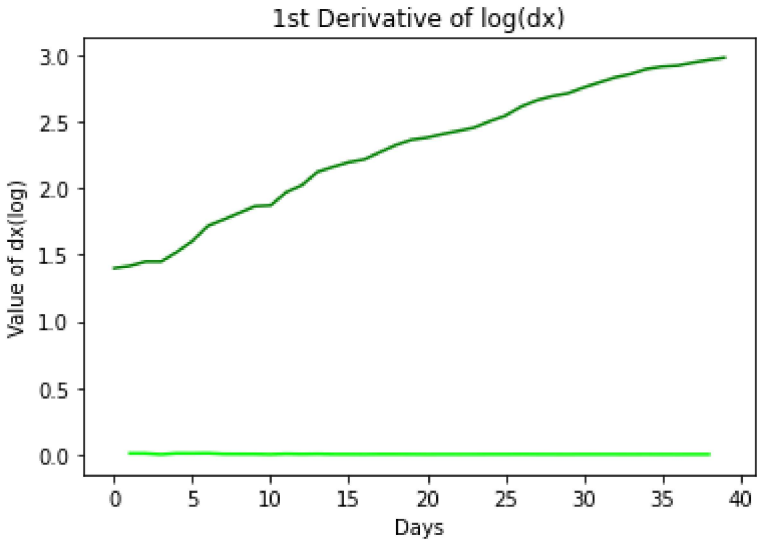


In [20]:
```
# part c
day_14_dx = first_derivative[14]
estimated_sick = sick[14] + day_14_dx * (36 - 14)
print("Sick people on day 36 = ", estimated_sick)
```

```
Sick people on day 36 =  158.66666666666666
```

In [22]:
```
# part d
log_sick = np.log10(sick)

# using the central difference method
dx_log_sick = log_sick[1:-1] - log_sick[:-2]
log_first_dx = dx_log_sick / (2 * del_days)

# plot log and dx log
plt.plot(days, log_sick, label='Log', color = 'green')
plt.xlabel('Days')
plt.ylabel('Log(# of Sick People)')
plt.title('Log(These People Be Sick)')

plt.plot(del_days, log_first_dx, label='dx(log)', color = 'lime')
plt.xlabel('Days')
plt.ylabel('Value of dx(log)')
plt.title('1st Derivative of log(dx)')
```

Out[22]:
```
Text(0.5, 1.0, '1st Derivative of log(dx)')
```



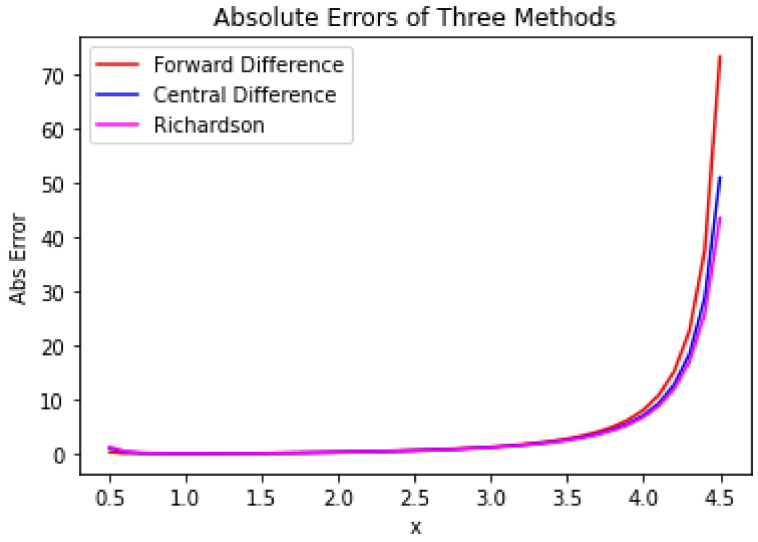
In [23]:
```
# part e
# same thing on day 14 but with log
log_day_14 = log_first_dx[14]
day_36 = log_sick[14] + log_day_14 * (36 - 14)
sick_day_36_log = 10 ** day_36
print("Sick people = ", sick_day_36_log)
```

```
Sick people =  152.70550844319595
```

Problem 3:

In [27]:
```
def f(x):
    return 1 / np.cos(np.log(x))

def df(x):
    return np.sin(np.log(x)) / x

x_vals = np.arange(0.5, 4.6, 0.1)
h = 0.1

# errors
fwd_err = []
central_err = []
rich_err = []

# numerical bois
for x in x_vals:
    forward_diff = (f(x + h) - f(x)) / h
    central_diff = (f(x + h / 2) - f(x - h / 2)) / h
    richardson_diff = (4 * central_diff - forward_diff) / 3  # Richardso

    # Calculate absolute errors
    analytical = df(x)
    fwd_err.append(abs(analytical - forward_diff))
    central_err.append(abs(analytical - central_diff))
    rich_err.append(abs(analytical - richardson_diff))

# plot absolute errors
plt.plot(x_vals, fwd_err, label="Forward Difference", color = 'red')
plt.plot(x_vals, central_err, label="Central Difference", color = 'blue'
plt.plot(x_vals, rich_err, label="Richardson", color = 'magenta')
plt.xlabel("x")
plt.ylabel("Abs Error")
plt.legend()
plt.title('Absolute Errors of Three Methods')
```

Out[27]: Text(0.5, 1.0, 'Absolute Errors of Three Methods')



Problem 4:

In [28]:
```
def three_point(x0, x1, x2, f0, f1, f2):
    # for getting 2nd dx
    h1 = x1 - x0
    h2 = x2 - x1
    second_dx = (f0 - 2*f1 + f2) / (h1 * h2 * (h1 + h2))
    return second_dx

def f(x):
    return x**3 -2

# three points to get func val
x0 = 1.0
x1 = 2.0
x2 = 3.0

# get func vals
f0 = f(x0)
f1 = f(x1)
f2 = f(x2)

# get 2nd derivative val
approx_val = three_point(x0, x1, x2, f0, f1, f2)
print("Second Derivative = ", approx_val)
```

```
Second Derivative =  6.0
```

Problem 5:

In [30]:

```
y_vals = [0.0, 0.25, 0.5, 0.75, 1.0]
x_vals = [[0.0, 1.25, 1.3125, 1.5, 1.8125, 2.25],
    [0.25, 0.8125, 0.875, 1.0625, 1.375, 1.8125],
    [0.5, 0.5, 0.5625, 0.75, 1.0625, 1.5],
    [0.75, 0.3125, 0.375, 0.5625, 0.875, 1.3125],
    [1.0, 0.25, 0.3125, 0.5, 0.8125, 1.25]]

x_goal = 0.25
y_goal = 0.75

# getting vals close to goal
x_i = min(range(len(x_vals)), key=lambda i: abs(x_vals[i][0] - x_goal))
y_i = min(range(len(y_vals)), key=lambda i: abs(y_vals[i] - y_goal))

# 1st deriv with cd
del_x = x_vals[x_i][1] - x_vals[x_i][0]
df_dx = (x_vals[x_i + 1][y_i] - x_vals[x_i - 1][y_i]) / (2 * del_x)

# 2nd deriv with cd
del_y = y_vals[y_i + 1] - y_vals[y_i]
d2f_dy2 = (x_vals[x_i][y_i + 1] - 2 * x_vals[x_i][y_i] + x_vals[x_i][y_i

# 2rd deriv wrt xy with cd
d2f_dxdy = (
    (x_vals[x_i + 1][y_i + 1] - x_vals[x_i - 1][y_i + 1] - x_vals[x_i +
    / (4 * del_x * del_y)
)

# results
print(" df/dx at xgoal, ygoal = ", df_dx)
print(" d^2f/dy^2 at goal = ", d2f_dy2)
print(" d^2f/dxdy at goal = ", d2f_dxdy)
```

```
 df/dx at xgoal, ygoal =  -0.6666666666666666
 d^2f/dy^2 at goal =  2.0
 d^2f/dxdy at goal =  0.0
```