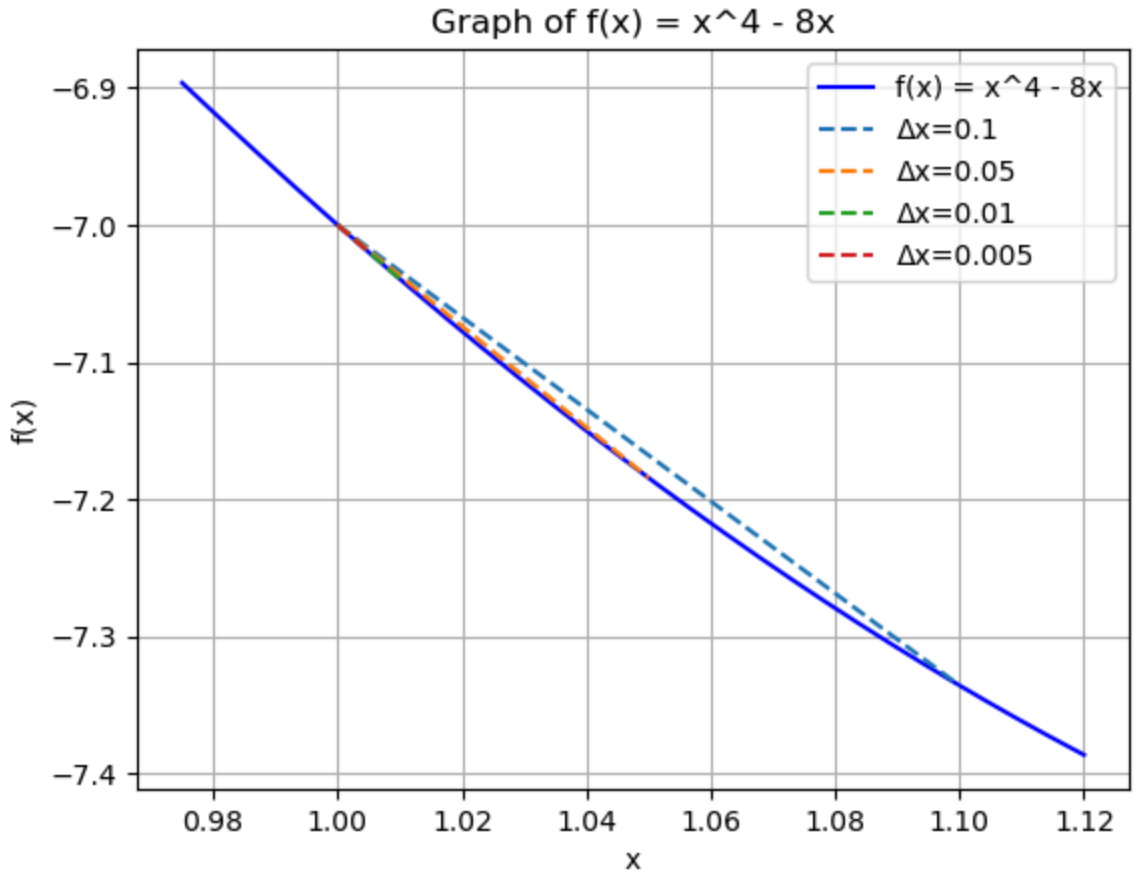
Chapter 3 Activities

pg 115 Problem 2 example part a

In [21]:
```
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return x**4 - 8*x

# Use difference quotient to approximate the derivative
def derivative_approx(x, delta_x):
    return (f(x + delta_x) - f(x)) / delta_x

# Sequence of delta_x values
delta_x_vals = [0.1, 0.05, 0.01, 0.005]
x = 1

approximations = []

# Zoomed in so the secant lines are more visible
x_range = np.linspace(0.975, 1.12, 400)
y_range = f(x_range)

plt.plot(x_range, y_range, label="f(x) = x^4 - 8x", color='blue')

for delta_x in delta_x_vals:
    derivative = derivative_approx(x, delta_x)
    approximations.append(derivative)
    secant_x = [x, x + delta_x]
    secant_y = [f(x), f(x + delta_x)]
    plt.plot(secant_x, secant_y, label=f'Δx={delta_x}', linestyle='--')

plt.title("Graph of f(x) = x^4 - 8x")
plt.xlabel("x")
plt.ylabel("f(x)")
plt.legend()
plt.grid(True)
plt.show()

for i, delta_x in enumerate(delta_x_values):
    print(f"Delta_x = {delta_x}: Derivative at x = {x} is {approximations[i]}")
```




```
Delta_x = 0.1: Derivative at x = 1 is -3.3590000000000053
Delta_x = 0.01: Derivative at x = 1 is -3.6898749999999936
Delta_x = 0.001: Derivative at x = 1 is -3.9395990000000047
Delta_x = 0.0001: Derivative at x = 1 is -3.9698998749999603
```

Problem 10 pg 118/119

a) If $g'(t)$ is positive for all t, we can conclude that $g(214)$ is positive. This is false. Just because the derivative is positive doesn't mean g(t) is positive at a specific value of t. It just means the function is increasing. Counterexample: If $g(t) = t - 220$, the derivative is 1 which is positve. However, $g(214) = -6$.

b) If $g′(t)$ is positive for all t, we can conclude that $g(214) > g(17)$. This is true. A positive derivative means an increasing function for all values of t. Therefore, $g(t_{2})$ will always be greater than $g(t_{1})$. In this case, $g(214) > g(17)$.

c) This is true. If Bill and Samantha have the same speed at every moment (constant speed), the distance between them cannot change. Samantha will be 1 mile ahead for the duration of travel.

d) This is true because of the Mean Value Theorem. The Mean Value Theorem states This theorem states that "if f is continuous function on a closed interval $[a,b]$ and differentiable on the open interval (a,b) then there is at least one point c in (a,b) where the instantaneous rate of change (the derivative) is equal to the average rate of change over the interval"(https://library.fiveable.me/ap-calc/unit-5/using-mean-value-theorem/study-

guide/79sP2PXcyvRvBsjb3HRq). Bill and Samantha's position is both continuous and differentiable over time. Since they both start at the same point and time and end at the same point and time, the total distance traveled is the same. Therefore, at some point of time between their start and finsih they were going the same speed.

Problem 2 pg 130/131

In [26]:
```
# Part a

def f(x):
    return 1 / x

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

h_vals = [0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001]
a = 2

for h in h_vals:
    derivative = central_difference(f, a, h)
    print(f"h = {h}, Derivative at x = {a}: {derivative}")
```

```
h = 0.1, Derivative at x = 2: -0.2506265664160401
h = 0.01, Derivative at x = 2: -0.25000625015624833
h = 0.001, Derivative at x = 2: -0.25000006249997764
h = 0.0001, Derivative at x = 2: -0.25000000062558314
h = 1e-05, Derivative at x = 2: -0.2500000000099645
h = 1e-06, Derivative at x = 2: -0.24999999997943334
```

$f'(a)$ is approximately -0.25

In [28]:
```
# Part b

def f(x):
    return np.sin(7 * x)

a = 3

for h in h_vals:
    derivative = central_difference(f, a, h)
    print(f"h = {h}, Derivative at x = {a}: {derivative}")
```

```
h = 0.1, Derivative at x = 3: -3.528568772540893
h = 0.01, Derivative at x = 3: -3.830974403016585
h = 0.001, Derivative at x = 3: -3.834073509789371
h = 0.0001, Derivative at x = 3: -3.8341045084616665
h = 1e-05, Derivative at x = 3: -3.8341048184842292
h = 1e-06, Derivative at x = 3: -3.834104821631712
```

$f'(a)$ is approximately -3.83

In [29]:
```
# Part c

def f(x):
```

```
    return x**3

a = 200

for h in h_vals:
    derivative = central_difference(f, a, h)
    print(f"h = {h}, Derivative at x = {a}: {derivative}")
```

```
h = 0.1, Derivative at x = 200: 120000.00999999233
h = 0.01, Derivative at x = 200: 120000.00009988435
h = 0.001, Derivative at x = 200: 120000.00000186265
h = 0.0001, Derivative at x = 200: 120000.00000465661
h = 1e-05, Derivative at x = 200: 120000.00006519257
h = 1e-06, Derivative at x = 200: 119999.99964609742
```

$f'(a)$ is approximately 120000.00

In [30]:
```
# Part d

def f(x):
    return 2**x

a = 5

for h in h_vals:
    derivative = central_difference(f, a, h)
    print(f"h = {h}, Approximate derivative at x = {a}: {derivative}")
```

```
h = 0.1, Approximate derivative at x = 5: 22.198475359917644
h = 0.01, Approximate derivative at x = 5: 22.1808873914922
h = 0.001, Approximate derivative at x = 5: 22.180711554058874
h = 0.0001, Approximate derivative at x = 5: 22.180709795645015
h = 1e-05, Approximate derivative at x = 5: 22.180709777153137
h = 1e-06, Approximate derivative at x = 5: 22.180709779107133
```

$f'(a)$ is approximately 22.18

Problem 3 pg 131

In [32]:
```
# Part a

a = 1

print(f"{'h':>10} {'Q1':>20} {'Q2':>20}")

for k in range(9):
    h = 1 / (2 ** k)
    q1 = ((a + h) ** 3 - (a - h) ** 3) / (2 * h)
    q2 = ((a + h) ** 3 - a ** 3) / h
    print(f"{h:>10.8f} {q1:>20.12f} {q2:>20.12f}")
```

```
         h                  Q1                  Q2
1.00000000       4.000000000000       7.000000000000
0.50000000       3.250000000000       4.750000000000
0.25000000       3.062500000000       3.812500000000
0.12500000       3.015625000000       3.390625000000
0.06250000       3.003906250000       3.191406250000
0.03125000       3.000976562500       3.094726562500
0.01562500       3.000244140625       3.047119140625
0.00781250       3.000061035156       3.023498535156
0.00390625       3.000015258789       3.011734008789
```

b) The true derivative is 3. By looking at the table, for Q1 we can that it stabilizes up to 4 decimal places. For Q2 it only stabilizes to 1 decimal place.

c) Q1 is the better estimator becuase it converges to the true value of 3 much faster than Q2.

Problem 4 pg 131

In [33]:
```
# Part a

a = 9

print(f"{'h':>10} {'Q1':>20} {'Q2':>20}")

for k in range(9):
    h = 1 / (2 ** k)
    q1 = ((a + h)**0.5 - (a - h)**0.5) / (2 * h)
    q2 = ((a + h)**0.5 - a**0.5) / h

    print(f"{h:>10.8f} {q1:>20.12f} {q2:>20.12f}")
```

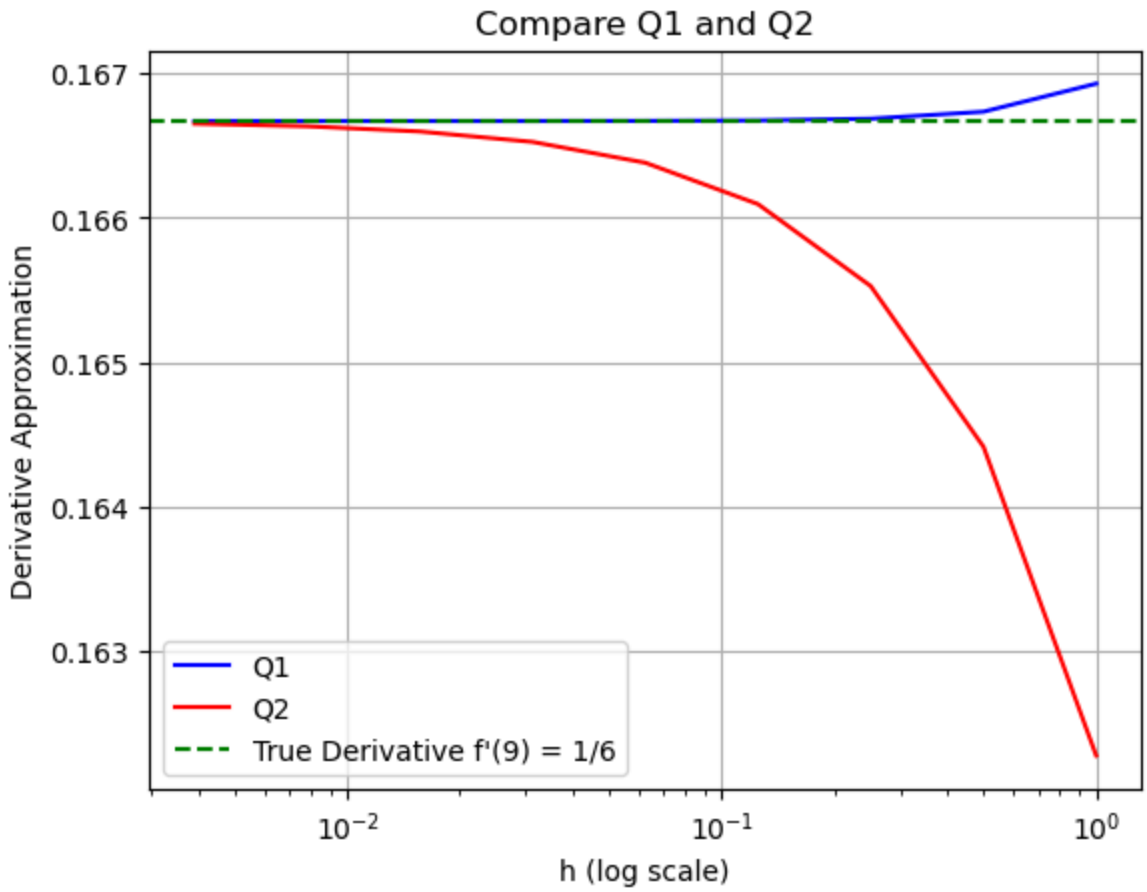
```
         h                  Q1                  Q2
1.00000000       0.166925267711       0.162277660168
0.50000000       0.166731054062       0.164414002969
0.25000000       0.166682747199       0.165525060596
0.12500000       0.166670685782       0.166091947189
0.06250000       0.166667671382       0.166378315169
0.03125000       0.166666917841       0.166522241370
0.01562500       0.166666729460       0.166594391429
0.00781250       0.166666682365       0.166630513375
0.00390625       0.166666670591       0.166648586099
```

b) Q1 stabilizes at 7 decimal places. Q2 stabilizes at 4.

c) Q1 is a better estimater as it converges on the true derivative faster.

In [36]:
```
# Part d

h_vals = [1 / (2 ** k) for k in range(9)]

q1 = [( (a + h)**0.5 - (a - h)**0.5 ) / (2 * h) for h in h_vals]
q2 = [( (a + h)**0.5 - a**0.5 ) / h for h in h_vals]

plt.plot(h_vals, q1, label="Q1", color='blue')
plt.plot(h_vals, q2, label="Q2", color='red')
```

```
plt.axhline(y=1/6, color='green', linestyle='--', label="True Derivative f'(9) = 1/
plt.xscale('log')
plt.xlabel('h (log scale)')
plt.ylabel('Derivative Approximation')
plt.title('Compare Q1 and Q2')
plt.legend()
plt.grid(True)
plt.show()
```



Problem 6 pg 132

In [40]:
```
# Using the central difference function

def f_2x(x):
    return 2**x

def f_3x(x):
    return 3**x

def f_10x(x):
    return 10**x

def f_halfx(x):
    return (1/2)**x

h_vals = [0.1, 0.01, 0.001, 0.0001, 0.00001, 0.000001]
x = 0
```

```
# Exact derivative values
exact_values = {
    "2^x": np.log(2),
    "3^x": np.log(3),
    "10^x": np.log(10),
    "(1/2)^x": -np.log(2)
}

functions = {
    "2^x": f_2x,
    "3^x": f_3x,
    "10^x": f_10x,
    "(1/2)^x": f_halfx
}

for name, func in functions.items():
    print(f"\nEstimating the derivative for {name} at x = {x}:")
    exact_derivative = exact_values[name]
    print(f"Exact derivative: {exact_derivative}")

    for h in h_vals:
        approx_derivative = central_difference(func, x, h)
        error = abs(approx_derivative - exact_derivative)
        print(f"h = {h}, Estimated: {approx_derivative}, Error: {error}")
```

```
Estimating the derivative for 2^x at x = 0:
Exact derivative: 0.6931471805599453
h = 0.1, Estimated: 0.6937023549974286, Error: 0.000555174437483319
h = 0.01, Estimated: 0.6931527309841479, Error: 5.550424202582782e-06
h = 0.001, Estimated: 0.6931472360641178, Error: 5.5504172480347336e-08
h = 0.0001, Estimated: 0.6931471811144618, Error: 5.545165437936816e-10
h = 1e-05, Estimated: 0.69314718055824, Error: 1.7053025658242404e-12
h = 1e-06, Estimated: 0.6931471805415867, Error: 1.8358536912899126e-11

Estimating the derivative for 3^x at x = 0:
Exact derivative: 1.0986122886681096
h = 0.1, Estimated: 1.100823570965711, Error: 0.00221128229760148
h = 0.01, Estimated: 1.098634388284142, Error: 2.2099616032500435e-05
h = 0.001, Estimated: 1.0986125096629218, Error: 2.2099481222781492e-07
h = 0.0001, Estimated: 1.0986122908784868, Error: 2.2103772145953826e-09
h = 1e-05, Estimated: 1.0986122886857963, Error: 1.7686740960698444e-11
h = 1e-06, Estimated: 1.0986122886968985, Error: 2.878897120695001e-11

Estimating the derivative for 10^x at x = 0:
Exact derivative: 2.3025850929940455
h = 0.1, Estimated: 2.322985885349429, Error: 0.0204007923553835
h = 0.01, Estimated: 2.3027885662471714, Error: 0.0002034732531259742
h = 0.001, Estimated: 2.3025871276731724, Error: 2.034679126960981e-06
h = 0.0001, Estimated: 2.302585113340694, Error: 2.0346648366142972e-08
h = 1e-05, Estimated: 2.302585093194587, Error: 2.0054136129488143e-10
h = 1e-06, Estimated: 2.3025850930391556, Error: 4.511013784735951e-11

Estimating the derivative for (1/2)^x at x = 0:
Exact derivative: -0.6931471805599453
h = 0.1, Estimated: -0.6937023549974286, Error: 0.000555174437483319
h = 0.01, Estimated: -0.6931527309841479, Error: 5.550424202582782e-06
h = 0.001, Estimated: -0.6931472360641178, Error: 5.5504172480347336e-08
h = 0.0001, Estimated: -0.6931471811144618, Error: 5.545165437936816e-10
h = 1e-05, Estimated: -0.69314718055824, Error: 1.7053025658242404e-12
h = 1e-06, Estimated: -0.6931471805415867, Error: 1.8358536912899126e-11
```

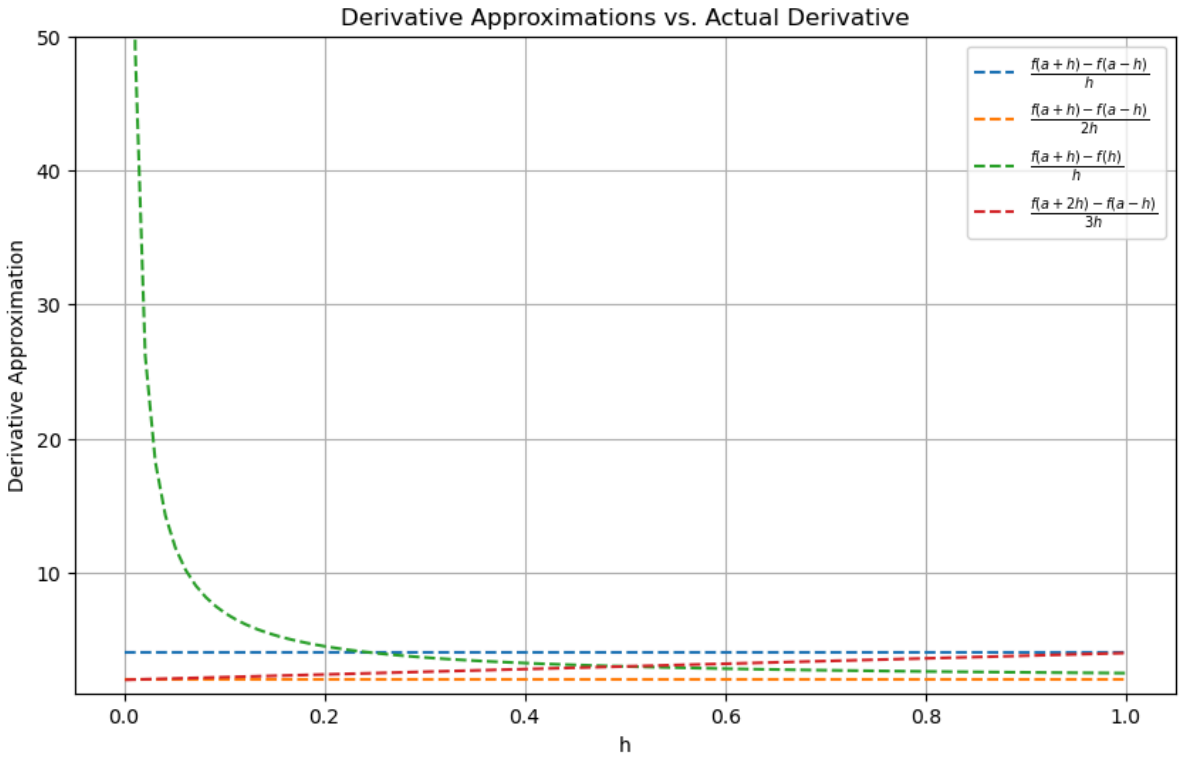
Problem 15 pg 134

If f(a) = b and f'(a) = -3 we can use linear approximation and plug in the values. Linear approximation means $f(a + k) = b + f'(a) * k$. Plugging in the known values we get $b - 3k$. THerefore the best estimate for $f(a + k)$ is $b - 3k$.

Problem 16 pg 134

In [63]:
```python
import numpy as np
import matplotlib.pyplot as plt

def f(x):
    return x**2

def f_prime(x):
    return 2 * x

a = 1
```

```
h_vals = np.linspace(0.001, 2, 100)

actual_derivative = f_prime(a)

formula_1 = (f(a + h_vals) - f(a - h_vals)) / h_vals
formula_2 = (f(a + h_vals) - f(a - h_vals)) / (2 * h_vals)
formula_3 = (f(a + h_vals) - f(h_vals)) / h_vals
formula_4 = (f(a + 2 * h_vals) - f(a - h_vals)) / (3 * h_vals)

plt.figure(figsize=(10, 6))
plt.plot(h_values, formula_1, label=r"$\frac{f(a+h) - f(a-h)}{h}$", linestyle='--')
plt.plot(h_values, formula_2, label=r"$\frac{f(a+h) - f(a-h)}{2h}$", linestyle='--'
plt.plot(h_values, formula_3, label=r"$\frac{f(a+h) - f(h)}{h}$", linestyle='--')
plt.plot(h_values, formula_4, label=r"$\frac{f(a+2h) - f(a-h)}{3h}$", linestyle='--
#plt.axhline(y=actual_derivative, color='r', linestyle='-', label="Actual Derivativ

# Add labels and title
plt.xlabel("h")
plt.ylabel("Derivative Approximation")
plt.title("Derivative Approximations vs. Actual Derivative")
plt.ylim(1, 50)
plt.legend()
plt.grid(True)
plt.show()
```



From the equations listed, I found $\frac{f(a + h) - f(a - h)}{2h}$ (formula_2) to be the most reasonable estimate. THis equation also happens to be the central difference formula. I believe it is the most reasonable because when all four formulas are graphed, the central difference function is the one that most closely falls the actual derivative line. It follows so closely that in order to see the CDF I had to remove the actual derivative line. The next

closest derivative approximation is formula_3. It starts off very far from the actual derivative but eventually grows very close to the actual derivative.

In [ ]: