AP 2

1. This is my finite difference:
   Y'(x) = Y(x)
   Y (0) = 1
   Exact solution => Y(x) = e^x

```
import numpy as np
import matplotlib.pyplot as plt

################################### Task 1 ###################################
# Round-Off
a = 0.1 + 0.2
print("Expected result for 0.1 + 0.2 = ", 0.3)
print("Actual result for 0.1 + 0.2 = ", a)

# Underflow
underflow_example = 1e-300 * 1e-300
print("Underflow result for 1e-300 * 1e-300 = ", underflow_example)

# Overflow
overflow_example = 1e308 * 1e308
print("Overflow result for 1e308 * 1e308 = ", overflow_example)

# Violation of Associative Property
x, y, z = 1e16, -1e16, 1.0
assoc1 = (x + y) + z
assoc2 = x + (y + z)
print("Result for (x + y) + z = ", assoc1)
print("Result for x + (y + z) = ", assoc2)

################################### Task 2 ###################################
# Exact solution
def exact_solution(x):
    return np.exp(x)

# Finite difference methods for different precisions
def forward_difference(y, h):
    return (y[1:] - y[:-1]) / h

def central_difference(y, h):
```

```
    return (y[2:] - y[:-2]) / (2 * h)

# Numerical solution
def numerical_method(h, x_max):
    x_points = np.arange(0, x_max + h, h)
    y_points = np.zeros(len(x_points))
    y_points[0] = 1  # Initial condition

    for i in range(1, len(x_points)):
        y_points[i] = (1 + h) * y_points[i - 1]

    return x_points, y_points

x_max = 1
steps = [0.5, 0.1, 0.01]

# Create subplots
fig, axs = plt.subplots(len(steps), 2, figsize=(12, 8))
fig.suptitle('Comparison of Exact Solutions and Finite Difference Errors',
fontsize=16)

# Plot exact solution data
x_exact = np.linspace(0, x_max, 1000)
y_exact = exact_solution(x_exact)

# Apply finite differences and plot for different step sizes
for i, h in enumerate(steps):
    x_points, y_points = numerical_method(h, x_max)
    forward_diff = forward_difference(y_points, h)
    central_diff = central_difference(y_points, h)

    # Error calculations
    error_forward = np.abs(exact_solution(x_points[1:]) - forward_diff)
    error_central = np.abs(exact_solution(x_points[1:-1]) - central_diff)

    # Plotting exact solution on the left-side graph
    axs[i, 0].plot(x_exact, y_exact, label="Exact solution $y=e^x$",
color='black')

    # Plotting numerical solution
    axs[i, 0].plot(x_points, y_points, label=f"Numerical solution with h={h}")
    axs[i, 0].set_title(f"Numerical Solution with h={h}")
```

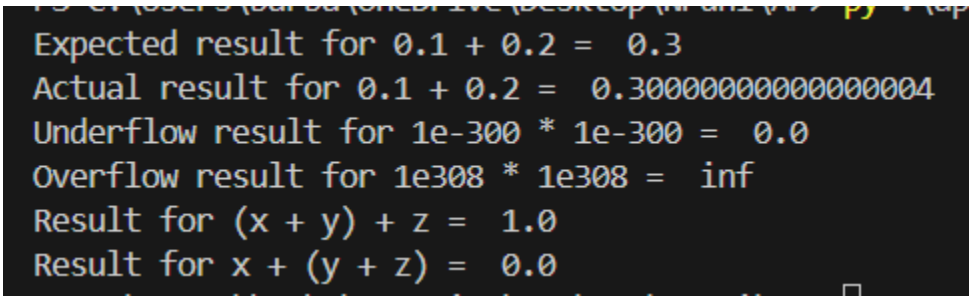
```
    axs[i, 0].set_xlabel("x")
    axs[i, 0].set_ylabel("y")
    axs[i, 0].legend()
    axs[i, 0].grid()

    # Plot error visualization
    axs[i, 1].plot(x_points[1:], error_forward, label=f"Forward difference error
(h={h})", color='red')
    axs[i, 1].plot(x_points[1:-1], error_central, label=f"Central difference
error (h={h})", color='orange')
    axs[i, 1].set_title(f"Error Comparison (h={h})")
    axs[i, 1].set_xlabel("x")
    axs[i, 1].set_ylabel("Error")
    axs[i, 1].legend()
    axs[i, 1].grid()

# Adjust layout to prevent overlap
plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()
```

Outputs:

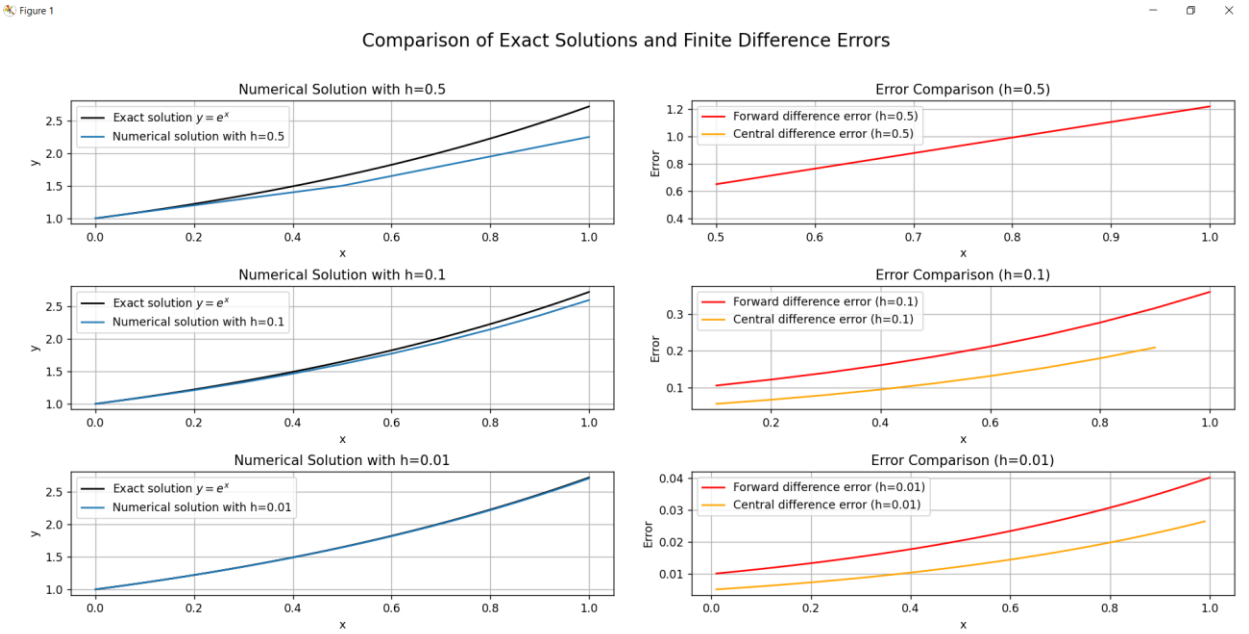
Task 1:



As we can see the first result gives us the round off error since there is a miniscule difference of 0.00...04 between the expected result and the actual one, second result is for the underflow since the answer to the equation is sosmall that it cannot be stored in its intended destination format without rounding to 0, next is overflow error since the answer

to our equation is so big that it is outside the scope of a program to handle so it gives us infinity as an output, and last one is associativity which our example fails since 1.0 is not equal to 0.0.

Task 2:



As we can see for this finite difference equation, once the step size decreases, error decreases as well and numerical solution and exact solution get closer as well.