```
def f(x):
    return x**2


a = 0  3
b = 0


n = 20


exact_integral = (b*3 - a*3) / 3


def midpoint_riemann_sum(f, a, b, n):
    h = (b - a) / n
    integral = 0
    for i in range(n):
        xi = a + (i + 0.5) * h
        integral += f(xi)
    integral *= h
    return integral

midpoint_riemann_result = midpoint_riemann_sum(f, a, b, n)

# Trapezoidal Rule problem
def trapezoidal_rule(f, a, b, n):
    h = (b - a) / n
    integral = 0.5 * (f(a) + f(b))
    for i in range(1, n):
        xi = a + i * h
        integral += f(xi)
    integral *= h
    return integral

trapezoidal_result = trapezoidal_rule(f, a, b, n)

# Simpson's Rule problem
def simpsons_rule(f, a, b, n):
    h = (b - a) / n
    integral = f(a) + f(b)
    for i in range(1, n):
        xi = a + i * h
        if i % 2 == 0:
            integral += 2 * f(xi)
        else:
            integral += 4 * f(xi)
    integral *= h / 3
    return integral

simpsons_result = simpsons_rule(f, a, b, n)

# Print the results
print(f"Exact Integral: {exact_integral:.4f}")
print(f"Mid-point Riemann Sum: {midpoint_riemann_result:.4f}")
print(f"Trapezoidal Rule: {trapezoidal_result:.4f}")
print(f"Simpson's Rule: {simpsons_result:.4f}")
```

IDLE Shell 3.11.5

```
Python 3.11.5 (v3.11.5:cce6ba91b3, Aug 24 2023, 10:50:31) [Clang 13.0.0 (clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
========================= RESTART: /Users/lakshman/Documents/hvc.py =========================
Exact Integral: 0.0000
Mid-point Riemann Sum: 0.0000
Trapezoidal Rule: 0.0000
Simpson's Rule: 0.0000
========================= RESTART: /Users/lakshman/Documents/jfbjkfweb.py =========================
Traceback (most recent call last):
  File "/Users/lakshman/Documents/jfbjkfweb.py", line 1, in <module>
    import numpy as np
ModuleNotFoundError: No module named 'numpy'
========================= RESTART: /Users/lakshman/Documents/jfbjkfweb.py =========================
Traceback (most recent call last):
  File "/Users/lakshman/Documents/jfbjkfweb.py", line 1, in <module>
    import num as np
ModuleNotFoundError: No module named 'num'
========================= RESTART: /Users/lakshman/Documents/vgsvhhsv.py =========================
Exact Integral: 0.0000
Mid-point Riemann Sum: 0.0000
Trapezoidal Rule: 0.0000
Simpson's Rule: 0.0000
```

```python
import numpy as np

# Define the sigmoid function
def sigmoid(x):
    return np.tanh(x)

# Define the exact derivative of the sigmoid function
def exact_derivative(x):
    return 1 - np.tanh(x)**2

# Define the three numerical differentiation formulas
def forward_difference(x, h):
    return (sigmoid(x + h) - sigmoid(x)) / h

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

def backward_difference(x, h):
```

Ln: 35, Col: 128

Run Share Command Line Arguments

```
|   h       |   Exact f'(x) | Forward Approximation | Central Approximation | Backward Approximation |
|  0.10000  |   1.000000 |   0.996680 |   0.996680 |   0.996680 |
|  0.01000  |   1.000000 |   0.999967 |   0.999967 |   0.999967 |
|  0.00100  |   1.000000 |   1.000000 |   1.000000 |   1.000000 |
|  0.00010  |   1.000000 |   1.000000 |   1.000000 |   1.000000 |
```

main.py

```python

def backward_difference(x, h):
    return (sigmoid(x) - sigmoid(x - h)) / h

# Values of h to be tested
h_values = [0.1, 0.01, 0.001, 0.0001, 0.00001]

# Create a table header
print(f"|   h      |   Exact f'(x) | Forward Approximation | Central Approximation | Backward Approximation |")

# Iterate through each value of h and print the results
for h in h_values:
    exact_result = exact_derivative(0)
    forward_result = forward_difference(0, h)
    central_result = central_difference(0, h)
    backward_result = backward_difference(0, h)

    # Print the results in a formatted table
```

Ln: 35, Col: 128

Run Share Command Line Arguments

```
|   0.10000   |   1.000000 |   0.996680 |   0.996680 |   0.996680 |
|   0.01000   |   1.000000 |   0.999967 |   0.999967 |   0.999967 |
|   0.00100   |   1.000000 |   1.000000 |   1.000000 |   1.000000 |
|   0.00010   |   1.000000 |   1.000000 |   1.000000 |   1.000000 |
|   0.00001   |   1.000000 |   1.000000 |   1.000000 |   1.000000 |
```

```python
ef backward_difference(x, h):
    return (sigmoid(x) - sigmoid(x - h)) / h

# Values of h to be tested
h_values = [0.1, 0.01, 0.001, 0.0001, 0.00001]

# Create a table header
print(f"|   h       |   Exact f'(x) | Forward Approximation | Central Approximation | Backward Approximation |")

# Iterate through each value of h and print the results
for h in h_values:
    exact_result = exact_derivative(0)
    forward_result = forward_difference(0, h)
    central_result = central_difference(0, h)
    backward_result = backward_difference(0, h)

    # Print the results in a formatted table
    print(f"|   {h:.5f}  |   {exact_result:.6f} |   {forward_result:.6f} |   {central_result:.6f} |   {backward_r
```

5, Col: 128

n | Share | Command Line Arguments

```
|  0.10000  |  1.000000 |  0.996680 |  0.996680 |  0.996680 |
|  0.01000  |  1.000000 |  0.999967 |  0.999967 |  0.999967 |
|  0.00100  |  1.000000 |  1.000000 |  1.000000 |  1.000000 |
|  0.00010  |  1.000000 |  1.000000 |  1.000000 |  1.000000 |
|  0.00001  |  1.000000 |  1.000000 |  1.000000 |  1.000000 |
```

, Col: 128

Share

Command Line Arguments

h	Exact f'(x)	Forward Approximation	Central Approximation	Backward Approximation
0.10000	1.000000	0.996680	0.996680	0.996680
0.01000	1.000000	0.999967	0.999967	0.999967
0.00100	1.000000	1.000000	1.000000	1.000000
0.00010	1.000000	1.000000	1.000000	1.000000
0.00001	1.000000	1.000000	1.000000	1.000000

```
** Process exited - Return Code: 0 **
```


Online Python Compiler