```python
#第一題
import numpy as np
from scipy.integrate import quad

# 定義函數f(x)
f = lambda x: np.exp(x) * np.sin(4*x)

# 設定積分區間 [a, b] 與 步長 h
a, b, h = 1, 2, 0.1

# 生成積分點(包含端點)
x = np.arange(a, b + h, h)

# 計算每個子區間的中點
midpoints = (x[:-1] + x[1:]) / 2

# 複合梯形法則的近似結果
trapezoidal = (h/2)*(f(x[0]) + 2*np.sum(f(x[1:-1])) + f(x[-1]))

# 複合辛普森法則的近似結果(注意：點數須為偶數，n=10)
simpson = (h/3)*(f(x[0]) + 4*np.sum(f(x[1:-1:2])) + 2*np.sum(f(x[2:-2:2])) + f(x[-1]))

# 複合中點法則的近似結果
midpoint = h * np.sum(f(midpoints))

# 使用 scipy 的 quad 計算精確解以便比較
exact, _ = quad(f, a, b)
```

```python
# 輸出結果
print("第1題結果 ：")
print("複合梯形法則 :", trapezoidal)
print("複合辛普森法則 :", simpson)
print("複合中點法則 :", midpoint)
print("精確值 :", exact)
```

```
第1題結果 ：
複合梯形法則 : 0.39614759221490675
複合辛普森法則 : 0.3856635960237503
複合中點法則 : 0.38080479837729914
精確值 : 0.38593572931020736
```

```python
#第二題
from scipy.integrate import quad, fixed_quad
import numpy as np

# 定義積分函數f(x)
f = lambda x: x**2 * np.log(x)

# 使用 Gaussian Quadrature (點數 n=3) 的近似解
gauss_n3, _ = fixed_quad(f, 1, 1.5, n=3)

# 使用 Gaussian Quadrature (點數 n=4) 的近似解
gauss_n4, _ = fixed_quad(f, 1, 1.5, n=4)

# 使用quad取得精確解
exact, _ = quad(f, 1, 1.5)

# 輸出結果
print("第2題結果：")
print("Gaussian Quadrature (n=3):", gauss_n3)
print("Gaussian Quadrature (n=4):", gauss_n4)
print("精確值:", exact)
```

```
第2題結果：
Gaussian Quadrature (n=3): 0.19225937725687903
Gaussian Quadrature (n=4): 0.19225935780486317
精確值: 0.19225935773279604
```

```python
#第三題
from scipy.integrate import dblquad
import numpy as np

# 定義被積分函數f(y, x)
f = lambda y, x: 2*y*np.sin(x) + np.cos(x)**2

# 積分範圍定義: 外層x積分範圍[0, π/4]
x_lower, x_upper = 0, np.pi/4

# 積分範圍定義: 內層y積分範圍[sin(x), cos(x)]
y_lower = lambda x: np.sin(x)
y_upper = lambda x: np.cos(x)

# Gaussian Quadrature近似解 (scipy內建的dblquad使用高精度方法)
gaussian_result, _ = dblquad(f, x_lower, x_upper, y_lower, y_upper)

# 同時以相同方法計算精確解(此處視為精確解)
exact_result, _ = dblquad(f, x_lower, x_upper, y_lower, y_upper)

# 輸出結果
print("第3題結果:")
print("Gaussian Quadrature雙重積分結果:", gaussian_result)
print("精確值:", exact_result)
```

```
第3題結果:
Gaussian Quadrature雙重積分結果: 0.5118446353109126
精確值: 0.5118446353109126
```

```python
#第四題a
from scipy.integrate import quad
import numpy as np

# 定義廣義積分函數
fa = lambda x: x**(-1/4) * np.sin(x)

# 使用quad進行廣義積分近似
result_a, _ = quad(fa, 0, 1, limit=100)

# 輸出結果
print("第4題(a)結果:")
print("積分結果 (0到1):", result_a)
```

```
第4題(a)結果:
積分結果 (0到1): 0.5284080812266488
```

```python
#第四題b使用變數變換t = 1/x 從t=0積到t=1
from scipy.integrate import quad
import numpy as np

# 經變數替換後的函數定義
fb_trans = lambda t: (t**2) * np.sin(1/t)

# 使用quad計算積分 (積分範圍0到1)
result_b, _ = quad(fb_trans, 0, 1, limit=100)

# 輸出結果
print("第4題(b)結果:")
print("積分結果 (1到∞):", result_b)
```

```
第4題(b)結果:
積分結果 (1到∞): 0.28652953744083975
```