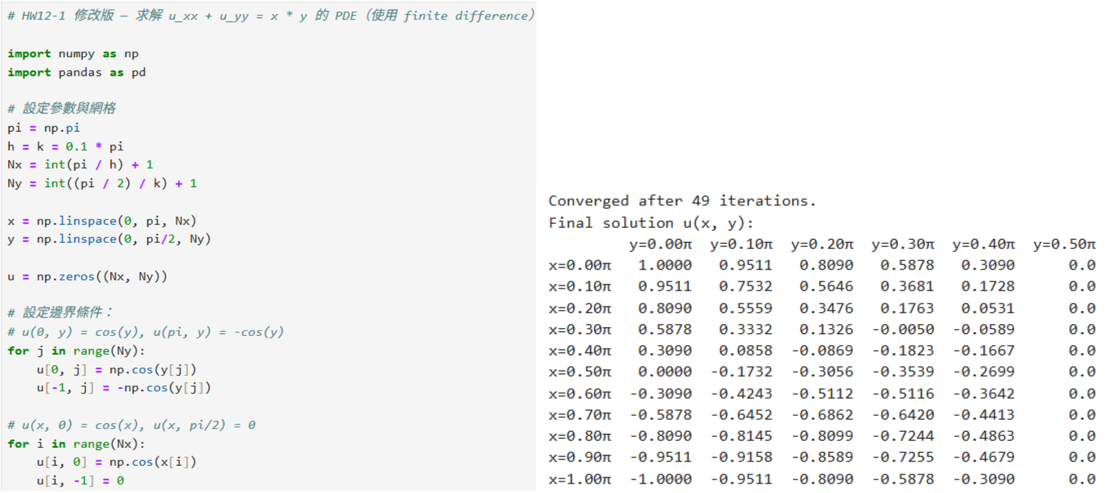
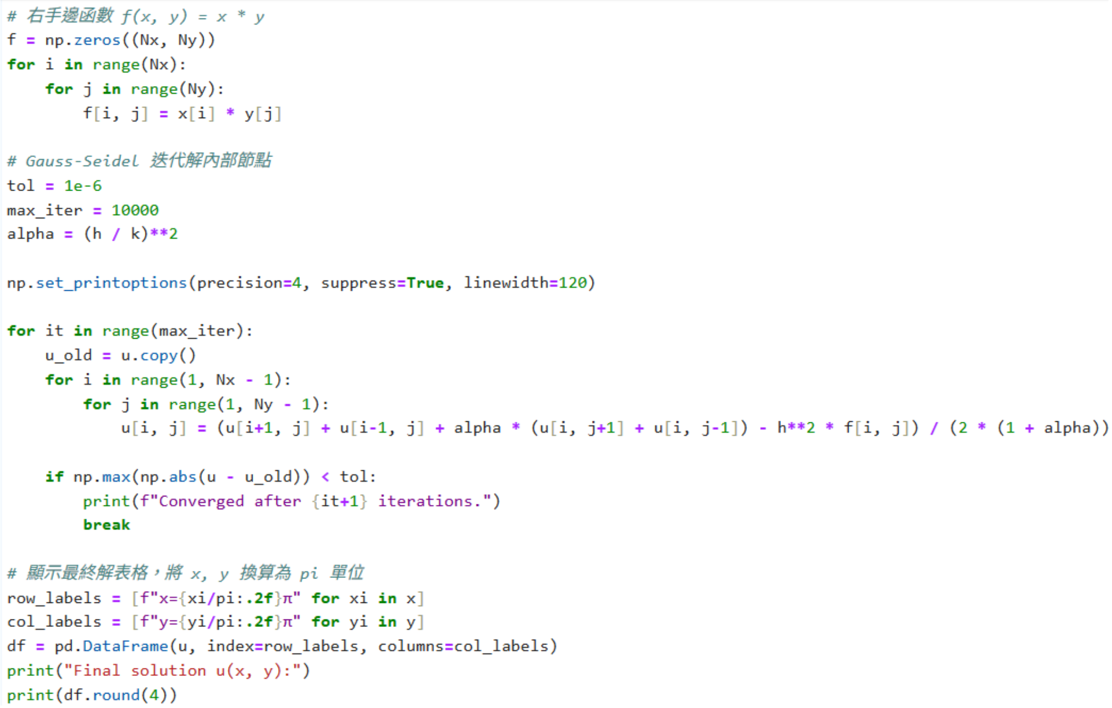
數值 hw12 E94106169 李柏臻

第一題



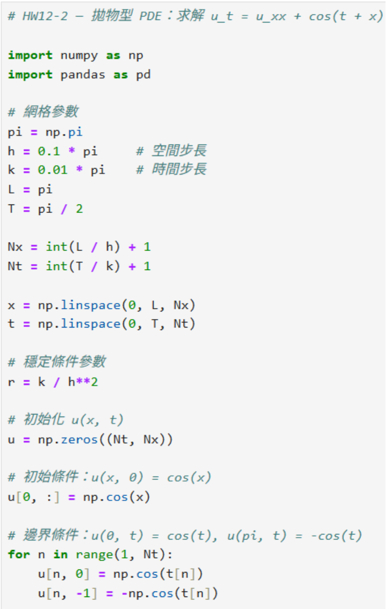
```python
# HW12-1 修改版 — 求解 u_xx + u_yy = x * y 的 PDE (使用 finite difference)

import numpy as np
import pandas as pd

# 設定參數與網格
pi = np.pi
h = k = 0.1 * pi
Nx = int(pi / h) + 1
Ny = int((pi / 2) / k) + 1

x = np.linspace(0, pi, Nx)
y = np.linspace(0, pi/2, Ny)

u = np.zeros((Nx, Ny))

# 設定邊界條件：
# u(0, y) = cos(y), u(pi, y) = -cos(y)
for j in range(Ny):
    u[0, j] = np.cos(y[j])
    u[-1, j] = -np.cos(y[j])

# u(x, 0) = cos(x), u(x, pi/2) = 0
for i in range(Nx):
    u[i, 0] = np.cos(x[i])
    u[i, -1] = 0
```

```
Converged after 49 iterations.
Final solution u(x, y):
         y=0.00π  y=0.10π  y=0.20π  y=0.30π  y=0.40π  y=0.50π
x=0.00π   1.0000   0.9511   0.8090   0.5878   0.3090      0.0
x=0.10π   0.9511   0.7532   0.5646   0.3681   0.1728      0.0
x=0.20π   0.8090   0.5559   0.3476   0.1763   0.0531      0.0
x=0.30π   0.5878   0.3332   0.1326  -0.0050  -0.0589      0.0
x=0.40π   0.3090   0.0858  -0.0869  -0.1823  -0.1667      0.0
x=0.50π   0.0000  -0.1732  -0.3056  -0.3539  -0.2699      0.0
x=0.60π  -0.3090  -0.4243  -0.5112  -0.5116  -0.3642      0.0
x=0.70π  -0.5878  -0.6452  -0.6862  -0.6420  -0.4413      0.0
x=0.80π  -0.8090  -0.8145  -0.8099  -0.7244  -0.4863      0.0
x=0.90π  -0.9511  -0.9158  -0.8589  -0.7255  -0.4679      0.0
x=1.00π  -1.0000  -0.9511  -0.8090  -0.5878  -0.3090      0.0
```

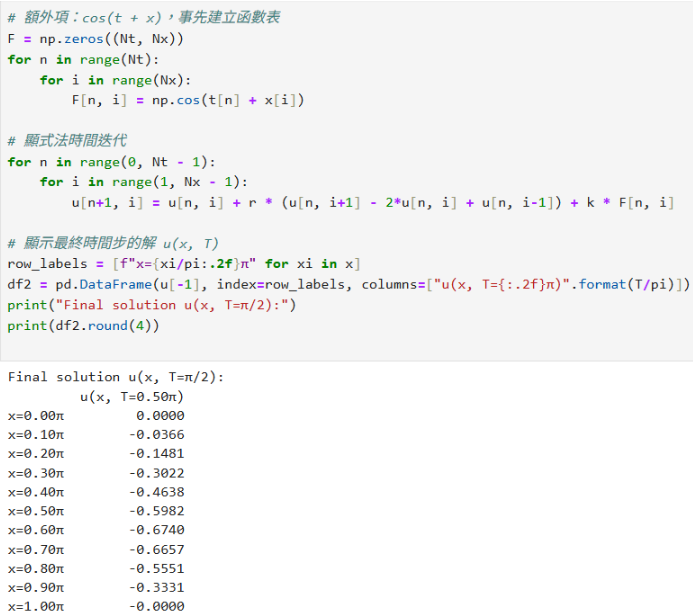


```python
# 右手邊函數 f(x, y) = x * y
f = np.zeros((Nx, Ny))
for i in range(Nx):
    for j in range(Ny):
        f[i, j] = x[i] * y[j]

# Gauss-Seidel 迭代解內部節點
tol = 1e-6
max_iter = 10000
alpha = (h / k)**2

np.set_printoptions(precision=4, suppress=True, linewidth=120)

for it in range(max_iter):
    u_old = u.copy()
    for i in range(1, Nx - 1):
        for j in range(1, Ny - 1):
            u[i, j] = (u[i+1, j] + u[i-1, j] + alpha * (u[i, j+1] + u[i, j-1]) - h**2 * f[i, j]) / (2 * (1 + alpha))

    if np.max(np.abs(u - u_old)) < tol:
        print(f"Converged after {it+1} iterations.")
        break

# 顯示最終解表格，將 x, y 換算為 pi 單位
row_labels = [f"x={xi/pi:.2f}π" for xi in x]
col_labels = [f"y={yi/pi:.2f}π" for yi in y]
df = pd.DataFrame(u, index=row_labels, columns=col_labels)
print("Final solution u(x, y):")
print(df.round(4))
```

第二題



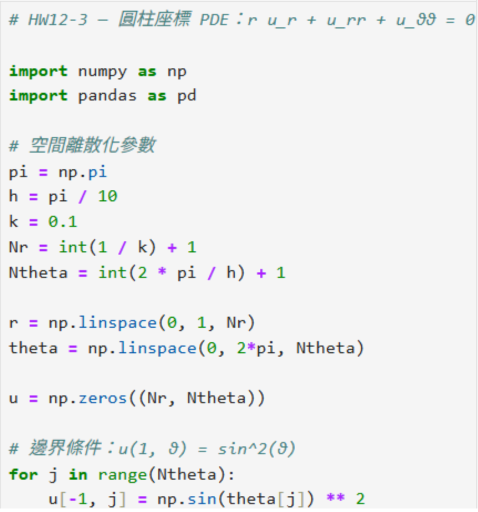
```python
# HW12-2 — 拋物型 PDE：求解 u_t = u_xx + cos(t + x)

import numpy as np
import pandas as pd

# 網格參數
pi = np.pi
h = 0.1 * pi     # 空間步長
k = 0.01 * pi    # 時間步長
L = pi
T = pi / 2

Nx = int(L / h) + 1
Nt = int(T / k) + 1

x = np.linspace(0, L, Nx)
t = np.linspace(0, T, Nt)

# 穩定條件參數
r = k / h**2

# 初始化 u(x, t)
u = np.zeros((Nt, Nx))

# 初始條件：u(x, 0) = cos(x)
u[0, :] = np.cos(x)

# 邊界條件：u(0, t) = cos(t), u(pi, t) = -cos(t)
for n in range(1, Nt):
    u[n, 0] = np.cos(t[n])
    u[n, -1] = -np.cos(t[n])
```



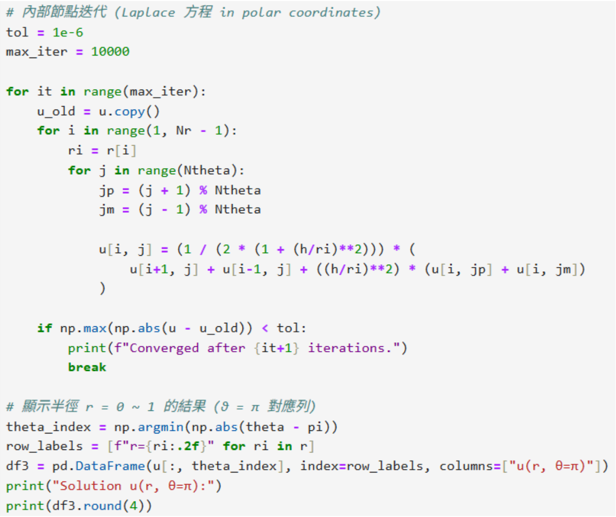
```python
# 額外項：cos(t + x)，事先建立函數表
F = np.zeros((Nt, Nx))
for n in range(Nt):
    for i in range(Nx):
        F[n, i] = np.cos(t[n] + x[i])

# 顯式法時間迭代
for n in range(0, Nt - 1):
    for i in range(1, Nx - 1):
        u[n+1, i] = u[n, i] + r * (u[n, i+1] - 2*u[n, i] + u[n, i-1]) + k * F[n, i]

# 顯示最終時間步的解 u(x, T)
row_labels = [f"x={xi/pi:.2f}π" for xi in x]
df2 = pd.DataFrame(u[-1], index=row_labels, columns=["u(x, T={:.2f}π)".format(T/pi)])
print("Final solution u(x, T=π/2):")
print(df2.round(4))
```

```
Final solution u(x, T=π/2):
         u(x, T=0.50π)
x=0.00π         0.0000
x=0.10π        -0.0366
x=0.20π        -0.1481
x=0.30π        -0.3022
x=0.40π        -0.4638
x=0.50π        -0.5982
x=0.60π        -0.6740
x=0.70π        -0.6657
x=0.80π        -0.5551
x=0.90π        -0.3331
x=1.00π        -0.0000
```

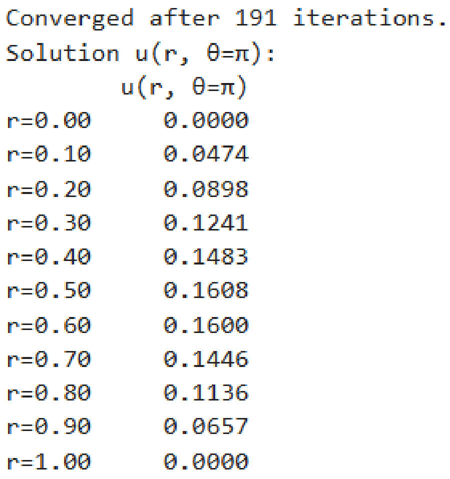
第三題



```python
# HW12-3 — 圓柱座標 PDE：r u_r + u_rr + u_ϑϑ = 0

import numpy as np
import pandas as pd

# 空間離散化參數
pi = np.pi
h = pi / 10
k = 0.1
Nr = int(1 / k) + 1
Ntheta = int(2 * pi / h) + 1

r = np.linspace(0, 1, Nr)
theta = np.linspace(0, 2*pi, Ntheta)

u = np.zeros((Nr, Ntheta))

# 邊界條件：u(1, ϑ) = sin^2(ϑ)
for j in range(Ntheta):
    u[-1, j] = np.sin(theta[j]) ** 2
```

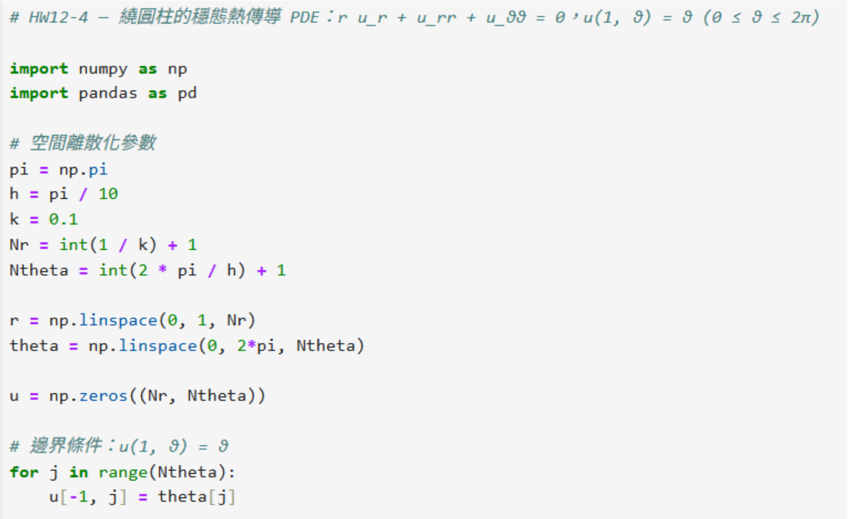


```python
# 內部節點迭代 (Laplace 方程 in polar coordinates)
tol = 1e-6
max_iter = 10000

for it in range(max_iter):
    u_old = u.copy()
    for i in range(1, Nr - 1):
        ri = r[i]
        for j in range(Ntheta):
            jp = (j + 1) % Ntheta
            jm = (j - 1) % Ntheta

            u[i, j] = (1 / (2 * (1 + (h/ri)**2))) * (
                u[i+1, j] + u[i-1, j] + ((h/ri)**2) * (u[i, jp] + u[i, jm])
            )

    if np.max(np.abs(u - u_old)) < tol:
        print(f"Converged after {it+1} iterations.")
        break

# 顯示半徑 r = 0 ~ 1 的結果 (ϑ = π 對應列)
theta_index = np.argmin(np.abs(theta - pi))
row_labels = [f"r={ri:.2f}" for ri in r]
df3 = pd.DataFrame(u[:, theta_index], index=row_labels, columns=["u(r, θ=π)"])
print("Solution u(r, θ=π):")
print(df3.round(4))
```

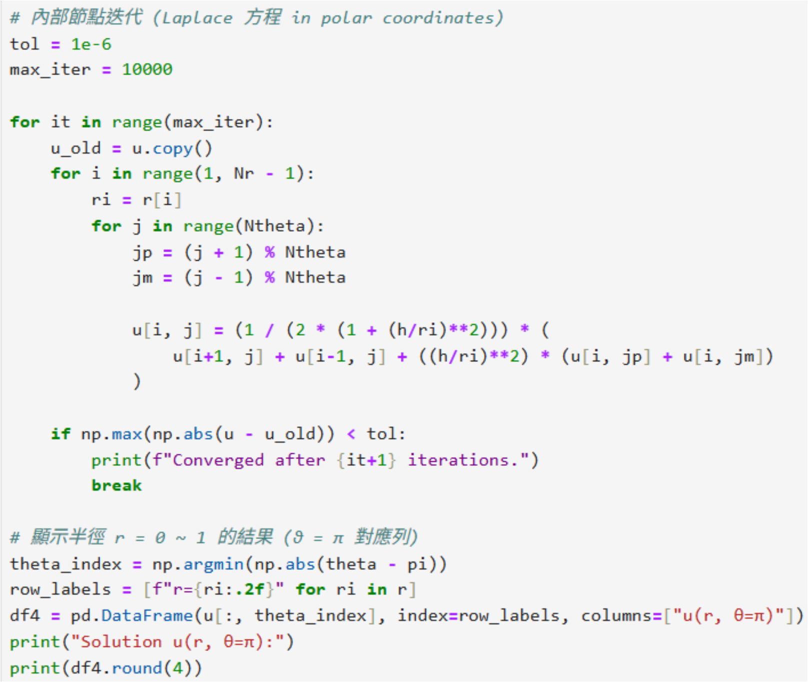


```
Converged after 191 iterations.
Solution u(r, θ=π):
        u(r, θ=π)
r=0.00     0.0000
r=0.10     0.0474
r=0.20     0.0898
r=0.30     0.1241
r=0.40     0.1483
r=0.50     0.1608
r=0.60     0.1600
r=0.70     0.1446
r=0.80     0.1136
r=0.90     0.0657
r=1.00     0.0000
```

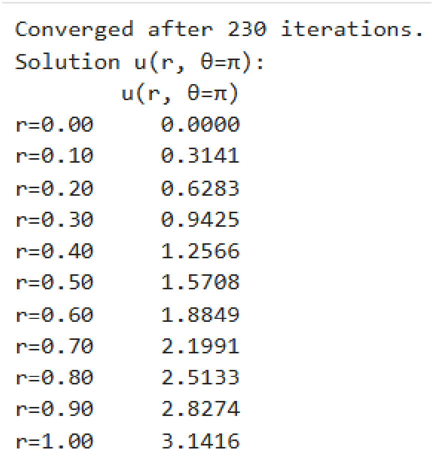
第四題



```python
# HW12-4 — 繞圓柱的穩態熱傳導 PDE：r u_r + u_rr + u_ϑϑ = 0，u(1, ϑ) = ϑ (0 ≤ ϑ ≤ 2π)

import numpy as np
import pandas as pd

# 空間離散化參數
pi = np.pi
h = pi / 10
k = 0.1
Nr = int(1 / k) + 1
Ntheta = int(2 * pi / h) + 1

r = np.linspace(0, 1, Nr)
theta = np.linspace(0, 2*pi, Ntheta)

u = np.zeros((Nr, Ntheta))

# 邊界條件：u(1, ϑ) = ϑ
for j in range(Ntheta):
    u[-1, j] = theta[j]
```



```python
# 內部節點迭代 (Laplace 方程 in polar coordinates)
tol = 1e-6
max_iter = 10000

for it in range(max_iter):
    u_old = u.copy()
    for i in range(1, Nr - 1):
        ri = r[i]
        for j in range(Ntheta):
            jp = (j + 1) % Ntheta
            jm = (j - 1) % Ntheta

            u[i, j] = (1 / (2 * (1 + (h/ri)**2))) * (
                u[i+1, j] + u[i-1, j] + ((h/ri)**2) * (u[i, jp] + u[i, jm])
            )

    if np.max(np.abs(u - u_old)) < tol:
        print(f"Converged after {it+1} iterations.")
        break

# 顯示半徑 r = 0 ~ 1 的結果 (ϑ = π 對應列)
theta_index = np.argmin(np.abs(theta - pi))
row_labels = [f"r={ri:.2f}" for ri in r]
df4 = pd.DataFrame(u[:, theta_index], index=row_labels, columns=["u(r, θ=π)"])
print("Solution u(r, θ=π):")
print(df4.round(4))
```



```
 Converged after 230 iterations.
 Solution u(r, θ=π):
          u(r, θ=π)
r=0.00      0.0000
r=0.10      0.3141
r=0.20      0.6283
r=0.30      0.9425
r=0.40      1.2566
r=0.50      1.5708
r=0.60      1.8849
r=0.70      2.1991
r=0.80      2.5133
r=0.90      2.8274
r=1.00      3.1416
```