第一題執行結果

main.py

```python
import math

def lagrange_interpolation(xs, ys):
    """
    使用 Lagrange 插值法生成插值多項式 P(x)。

    參數:
        xs: 已知 x 值的列表
        ys: 對應的 y 值列表

    回傳:
        P(x): 一個函數，輸入 x 可以計算對應的插值結果
    """
    if len(xs) != len(ys):
        raise ValueError("xs 跟 ys 長度不一致")

    def P(x):
        total = 0.0
        n = len(xs)
        for j in range(n):
            Lj = 1.0
            for m in range(n):
                if m != j:
                    Lj *= (x - xs[m]) / (xs[j] - xs[m])
            total += ys[j] * Lj
```

Ln: 20, Col: 27

Run | Share | $ | Command Line Arguments

```
==== Lagrange Interpolation for f(0.750) ====
Degree 1 approximation: 0.731591, error = 0.000109
Degree 2 approximation: 0.731716, error = 0.000016
Degree 3 approximation: 0.731704, error = 0.000004
Degree 4 approximation: 0.731700, error = 0.000000


** Process exited - Return Code: 0 **
Press Enter to exit terminal
```

第二題執行結果

main.py

```python
import math

def f(x):
    """ 定義函數 f(x) = x - e^(-x) """
    return x - math.exp(-x)

def inverse_quadratic_interp_3points(x0, x1, x2):
    """
    使用三點逆二次插值法 (Inverse Quadratic Interpolation) 找 f(x) = 0 的近似解。

    公式：
      x3 = x0 * (f1 * f2) / ((f0 - f1) * (f0 - f2))
         + x1 * (f0 * f2) / ((f1 - f0) * (f1 - f2))
         + x2 * (f0 * f1) / ((f2 - f0) * (f2 - f1))

    其中：
      f0 = f(x0), f1 = f(x1), f2 = f(x2)

    參數:
        x0, x1, x2: 目前的三個近似根

    回傳:
        x3: 新的近似根
    """
    f0, f1, f2 = f(x0), f(x1), f(x2)
```

Ln: 58, Col: 1

Run | Share | $ | Command Line Arguments

```
==== Inverse Quadratic Interpolation for f(x) = x - e^(-x) ====
Iter |    x0     |    x1     |    x2     |    x_new    |   f(x_new)
-----+-----------+-----------+-----------+-------------+-------------
  1  |  0.400000 |  0.500000 |  0.600000 |  0.56714602 | 4.278589e-06
  2  |  0.500000 |  0.600000 |  0.567146 |  0.56714329 | -5.233713e-11

收斂於 x = 0.56714329

最終近似解 x = 0.56714329, f(x) = -0.00000000
```

第三題執行結果

main.py

```python
import numpy as np
from scipy.optimize import fsolve

# 給定的數據點
t_points = [0, 3, 5, 8, 13]  # 時間 T (秒)
d_points = [0, 200, 375, 620, 990]  # 距離 D (英尺)
v_points = [75, 77, 80, 74, 72]  # 速度 V (英尺/秒)

# Hermite 插值的基礎：計算除法差分表
def divided_differences(t_points, d_points, v_points):
    n = len(t_points)
    z = np.zeros(2 * n)
    Q = np.zeros((2 * n, 2 * n))

    for i in range(n):
        z[2 * i] = t_points[i]
        z[2 * i + 1] = t_points[i]

    for i in range(n):
        Q[2 * i, 0] = d_points[i]
        Q[2 * i + 1, 0] = d_points[i]
        Q[2 * i + 1, 1] = v_points[i]

    for i in range(2, 2 * n):
        for j in range(2, i + 1):
            if j == 2 and i % 2 == 1:
                continue
            Q[i, j] = (Q[i, j-1] - Q[i-1, j-1]) / (z[i] - z[i-j])
```

input

```
a. At t = 10 seconds:
Position: 19.25 feet
Speed: 809.24 feet/second

b. Speed limit: 55 mi/h = 80.67 feet/second
The car never exceeds the speed limit of 55 mi/h.
/usr/lib/python3/dist-packages/scipy/optimize/_minpack_py.py:177: RuntimeWarning: The iteration
is not making good progress, as measured by the
  improvement from the last ten iterations.
  warnings.warn(msg, RuntimeWarning)

c. Predicted maximum speed: 822.37 feet/second = 560.71 mi/h
Critical points for speed: ['1.95', '4.13', '6.50', '10.15', '0.00', '13.00']


...Program finished with exit code 0
Press ENTER to exit console.
```

上個網站不能安裝 scipy，所以換一個