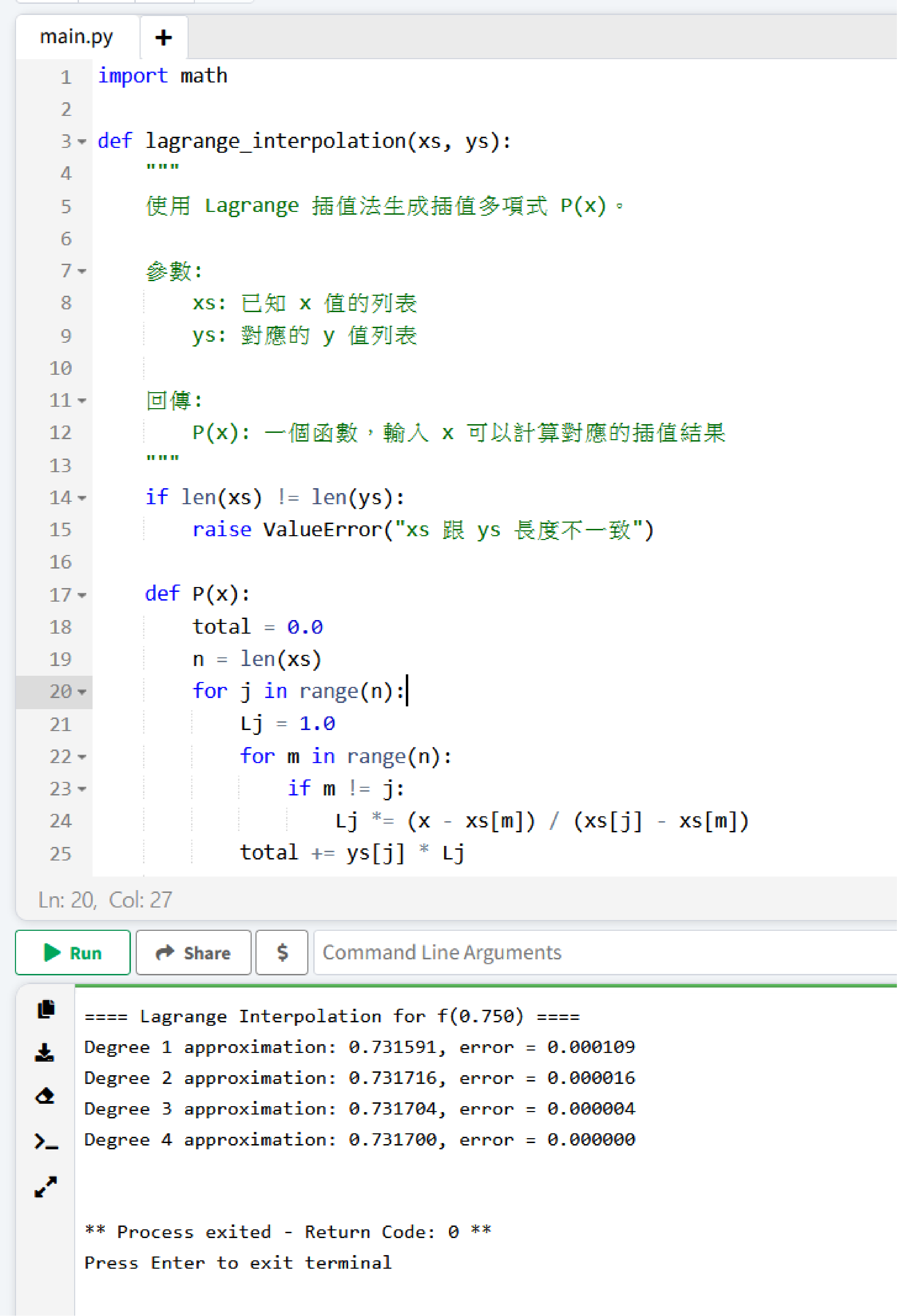
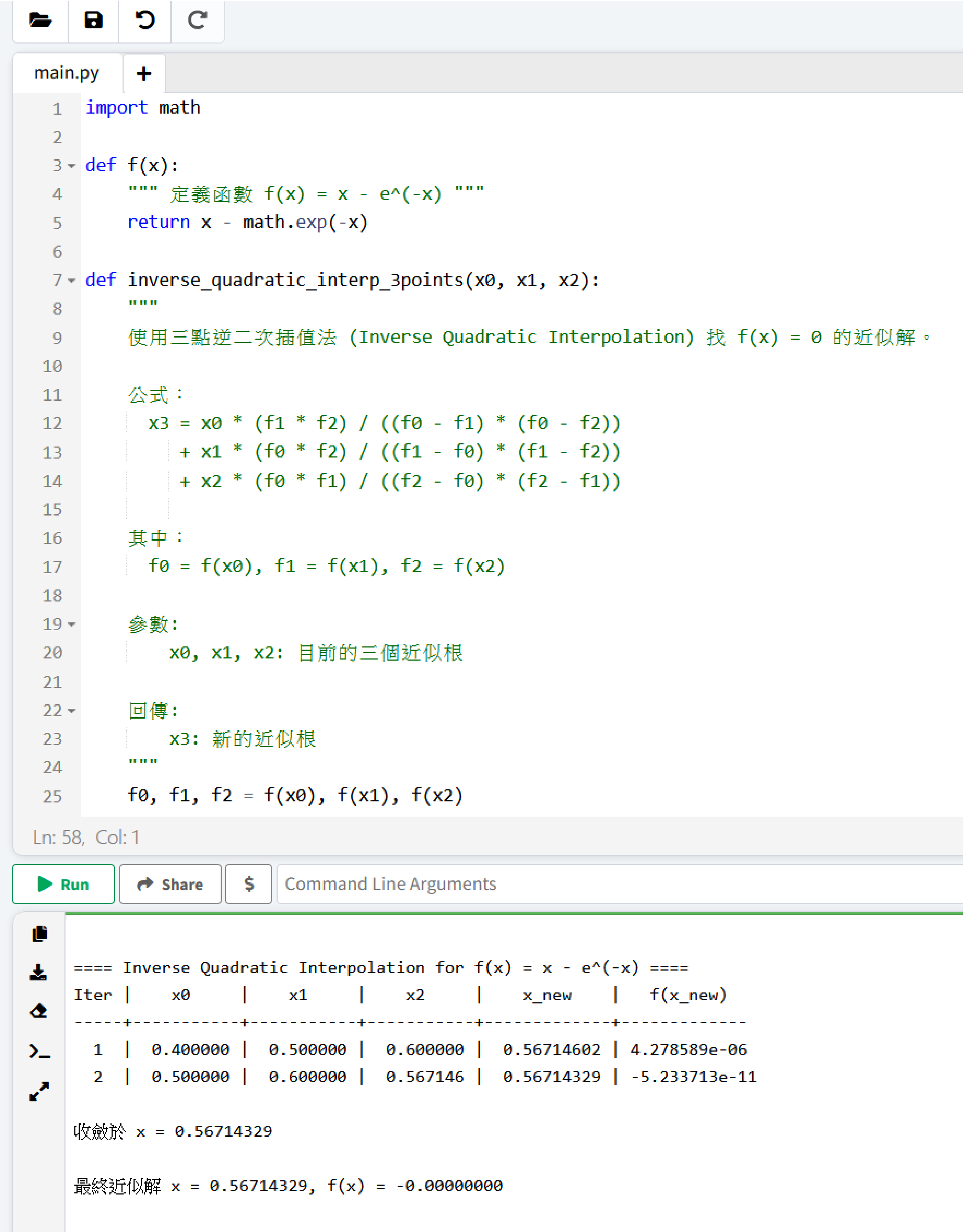
第一題執行結果



```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
```

```
==== 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
```

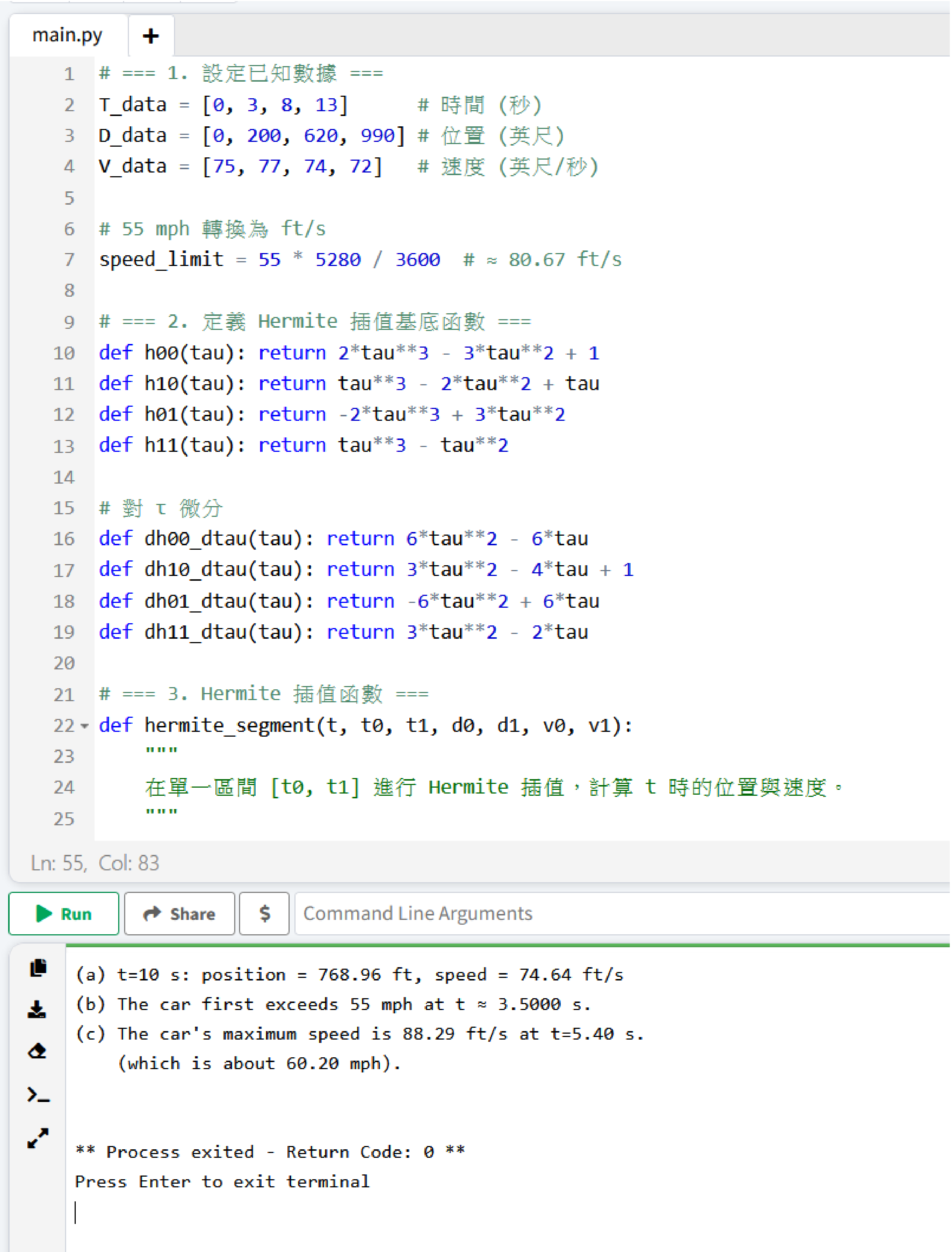
第二題執行結果



```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)
```

```

==== 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



```python
# === 1. 設定已知數據 ===
T_data = [0, 3, 8, 13]       # 時間 (秒)
D_data = [0, 200, 620, 990] # 位置 (英尺)
V_data = [75, 77, 74, 72]   # 速度 (英尺/秒)

# 55 mph 轉換為 ft/s
speed_limit = 55 * 5280 / 3600  # ≈ 80.67 ft/s

# === 2. 定義 Hermite 插值基底函數 ===
def h00(tau): return 2*tau**3 - 3*tau**2 + 1
def h10(tau): return tau**3 - 2*tau**2 + tau
def h01(tau): return -2*tau**3 + 3*tau**2
def h11(tau): return tau**3 - tau**2

# 對 τ 微分
def dh00_dtau(tau): return 6*tau**2 - 6*tau
def dh10_dtau(tau): return 3*tau**2 - 4*tau + 1
def dh01_dtau(tau): return -6*tau**2 + 6*tau
def dh11_dtau(tau): return 3*tau**2 - 2*tau

# === 3. Hermite 插值函數 ===
def hermite_segment(t, t0, t1, d0, d1, v0, v1):
    """
    在單一區間 [t0, t1] 進行 Hermite 插值，計算 t 時的位置與速度。
    """
```

```
(a) t=10 s: position = 768.96 ft, speed = 74.64 ft/s
(b) The car first exceeds 55 mph at t ≈ 3.5000 s.
(c) The car's maximum speed is 88.29 ft/s at t=5.40 s.
    (which is about 60.20 mph).


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