4. **[9 marks] Running-time analysis.**

Note: for all parts of this question, you may go from a step count expression (e.g., $2n + 3$) to a final Big-O/Omega/Theta expression (e.g., $\mathcal{O}(n)$, $\Theta(n)$) without proof. Your final Big-O/Omega/Theta expressions should be fully simplified (e.g., $\Theta(n)$, not $\Theta(2n + 3)$).

(a) **[5 marks]** Consider the following function.

```
def f1(n: int) -> None:
    """Precondition: n >= 0"""
    for i in range(0, n * n):        # Loop 1
        for j in range(0, i * i):    # Loop 2
            print(j)
```

Analyse the running time of this function in terms of its input n. Your final step count (before concluding a Theta bound) should not contain any summations. You will find the following formula useful:

$$\forall m \in \mathbb{N},\ \sum_{i=0}^{m} i^2 = \frac{m(m+1)(2m+1)}{6}$$

Loop 2 runs for i^2 iterations and takes one step each time, thus it runs i^2 steps for each i in range$(0, n*n)$.

Loop 1 runs while i goes from 0 to n^2-1, thus the total number of steps is:

$$RT_{f_1} = \sum_{i=0}^{n^2-1} i^2(1) = \frac{(n^2-1)(n^2)(2n^2-1)}{6}$$

$$= \frac{n^6}{3} - \frac{n^4}{2} + \frac{n^2}{6} \in \Theta(n^6)$$

Thus the running time of f_1 is $RT_{f_1} \in \Theta(n^6)$