

## Learning Objectives

By the end of this worksheet, you will:

- Determine the exact number of iterations of loops with a variety of loop counter behaviours.
- Find the asymptotic running time of programs containing loops.

- 
1. **Loop variations.** Each of the following functions takes as input a *non-negative integer* and performs at least one loop. For each loop, determine the *exact* number of iterations that will occur (in terms of the function's input  $n$ ), and then use this to determine the simplest Theta expression<sup>1</sup> for the running time of each function. You do *not* need to prove any " $g \in \Theta(f)$ " statements here.

Note: each loop body runs in  $\Theta(1)$  time in this question. While this won't always be the case, such examples allow you to focus on just counting loop iterations here.

```
1 def f1(n: int) -> None:
2     i = 0
3     while i < n:
4         print(i)
5         i = i + 5
```

```
1 def f2(n: int) -> None:
2     i = 4
3     while i < n:
4         print(i)
5         i = i + 1
```

---

<sup>1</sup>By "simplest," we mean ignoring constants and slower-growth terms. For example, write  $\Theta(n)$  instead of  $\Theta(2n + 0.3)$ .

```
1 def f3(n: int) -> None:
2     """Precondition: n > 0."""
3     i = 0
4     while i < n:
5         print(i)
6         i = i + (n / 10)
```

```
1 def f4(n: int) -> None:
2     i = 20
3     while i < n * n:
4         print(i)
5         i = i + 3
```

```
1 def f5(n: int) -> None:
2     i = 20
3     while i < n * n:
4         print(i)
5         i = i + 3
6
7     j = 0
8     while j < n:
9         print(j)
10        j = j + 0.01
```

2. **Multiplicative increments.** Consider the following function:

```
1 def f(n: int) -> None:
2     """Precondition: n > 0."""
3     i = 1
4     while i < n:
5         print(i)
6         i = i * 2
```

Even though this looks similar to previous examples, the fact that the loop variable `i` changes by a multiplicative rather than additive factor requires a more principled approach in determining the number of loop iterations.

- (a) Let  $i_0$  be the value of variable `i` when 0 loop iterations have occurred,  $i_1$  be the value of `i` immediately after 1 loop iteration has occurred, and in general  $i_k$  be the value of `i` immediately after  $k$  loop iterations have occurred. For example,  $i_0 = 1$  (the initial value of `i`),  $i_1 = 2$ , and  $i_2 = 4$ . Determine the values of  $i_3$ ,  $i_4$ , and a general formula for  $i_k$ .<sup>2</sup>

- (b) Use your formula from part (a) to determine the exact number of loop iterations that occur, in terms of  $n$ .  
HINT: Find the *smallest* value of  $k$  that makes the loop condition false.

- (c) Determine the Theta running time for the function `f`.

- (d) Why did we not initialize `i = 0` in this function?

---

<sup>2</sup>Of course, if  $n$  is small then not a lot of loop iterations occur. More formally,  $i_k$  represents the value of `i` after  $k$  loop iterations, *if*  $k$  iterations occur.

3. **A more unusual increment.** Consider the following function:

```
1 def f(n: int) -> None:
2     """Precondition:  $n \geq 2$ ."""
3     i = 2
4     while i < n:
5         print(i)
6         i = i * i
```

Analyse the running time of this function using the same technique as the previous question.