# While-Loops

## Table of contents



See csse1001[1] for course logistics — this note covers Lecture 4A's technical content.

### Today's outline

- Why we need to repeat code, possibly indefinitely
- The while-loop
- Simulating a do-while, and reading user input

### Learning objectives

1. Loops are a control structure for repeating code.
2. A while-loop repeats code while a condition holds.
3. `input()` reads from the keyboard; `random.randint` generates random integers.

[1] courses/csse1001/csse1001.pdf

## Motivation

There are (at least) two scenarios where repeating code, possibly indefinitely, is necessary:

1. prompting the user for valid input, and
2. playing a random game (e.g. guessing dice rolls).

See python-while-loops[2] for the full reference on loops, while-loops, `break`, augmented assignment operators (`+=`, `*=`, `/=`, `%=`), simulating a do-while, `input()`, and `random.randint`.

## Accumulator pattern

```
>>> x = 0
>>> while x < 10:
...     x = x + 1
>>> x
10
```

Using an augmented assignment operator (see python-while-loops[3]) makes the accumulation more concise:

```
>>> x = 0
>>> while x < 10:
...     x += 1
>>> x
10
```

There is usually a key-stroke (typically `ctrl+c`) that terminates a runaway loop (e.g. `while True: print(x); x += 1`) — it is a good idea to learn what it is in your IDE.

## Exercise: guess the number

Write a function `foo(target: int) -> int` that prompts the user to input a number until the user guesses some (secret) target. For each guess the function should print `Too high`, `Too low`, or `That's it!` depending on the guess, and return the number of guesses required.

[2] courses/csse1001/python-while-loops.pdf
[3] courses/csse1001/python-while-loops.pdf

```
def foo(target: int) -> int:
    number_of_guesses = 0
    while True:
        guess = int(input("Guess: "))
        number_of_guesses += 1
        if guess > target:
            print("Too high")
        elif guess < target:
            print("Too low")
        else:
            print("That's it!")
            return number_of_guesses
```

### Exercise: guess the dice roll

Write a function `bar() -> None` that prompts the user to predict the result of a six-sided dice roll, repeating the prompt until the user predicts correctly:

```
from random import randint

def bar() -> None:
    """
    Prompt user to predict the result of a six sided
    dice roll until the guess is correct.
    """
    while True:
        guess = int(input("Guess in [1..6]: "))
        roll = randint(1, 6)
        if roll == guess:
            break
    return
```

### Further exercises

The source poses these without a worked solution — left here as open exercises rather than guessed at:

- Write a function that prints the result of a six-sided dice roll until a six is rolled, and returns the number of rolls required.
- Write a function that accumulates the result of dice rolls until some threshold is met/passed, and returns the number of rolls required.

- Develop functions for adding and multiplying positive integers together, restricting yourself to exactly two arithmetic operations — `succ` (adding one) and `pred` (subtracting one). The source only gives the two building blocks:

```
def succ(x: int) -> int:
    return x+1

def pred(x: int) -> int:
    return x-1
```

(The add/multiply functions built from `succ`/`pred` are not solved in the source.)

## Summary

The while-loop is a control structure for repeating code a possibly infinite number of times.

## Next lecture

2025-08-18-non-primitive-data[4] — lists and dictionaries.

[4] courses/csse1001/2025-08-18-non-primitive-data.pdf