

While Loops

A loop allows you to execute the same statements multiple times. **while** loops continue running as long as a condition is true.

Manager:

Recorder:

Presenter:

Reflector:

Content Learning Objectives

After completing this activity, students should be able to:

- Identify the three main components of a **while** loop.
- Write a **while** loop to ask for and validate user input.
- Use the `random` module to generate random numbers.

Process Skill Goals

During the activity, students should make progress toward:

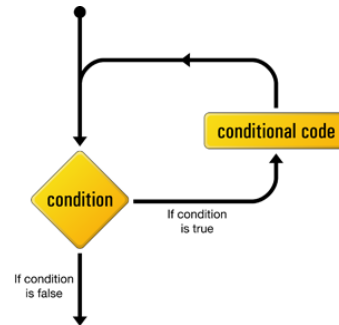
- Tracing the execution of while loops and predicting the output. (Critical Thinking)



Model 1 while Statements

A **while** statement is a general loop that continues to run while the condition is true. Run the following code, and record the output.

```
i = 0
while i < 3:
    print("the number is", i)
    i = i + 1
print("goodbye")
```



Questions (25 min)

Start time:

1. What must the value of the Boolean expression (after the **while**) be in order for the first print statement to execute?
2. Circle the statement that changes the variable *i* in the above code.
3. What happens to the value of the variable *i* during the execution of the loop?
4. Explain why the loop body does not execute again after “the number is 2” is output.
5. Reverse the order of the statements in the loop body:

```
while i < 3:
    i = i + 1
    print("the number is", i)
```

- a) How does the order impact the output displayed by the **print** function?
- b) Does the order impact the total number of lines that are output?

6. Identify three different ways to modify the code so that the loop only executes twice.
7. Describe the three parts of a `while` loop that control the number of times the loop executes.
8. Comment out the statement `i = i + 1`, and run the code. Then press Ctrl-C (hold down the Ctrl key and press C). Describe the behavior you see, and explain why it happened.
9. Consider the following program:

```
1 total = 0
2 count = 0
3
4 n = None
5 while n != 0:
6     n = float(input("Next number: "))
7     total += n
8     count += 1
9
10 print()
11 print("Total:", total)
12 print("Count:", count)
```

- a) Which line initializes the loop variable?
- b) Which line updates the loop variable?
- c) Can you predict how many times the loop will run? Explain why or why not.
- d) Does the code have the potential to loop forever? Explain why or why not.

10. Consider the following program output (with user input underlined):

```
Enter a number between 1 and 100: 0  
Too low!  
Enter a number between 1 and 100: 101  
To high!  
Enter a number between 1 and 100: 50  
You entered 50
```

Before writing the code, answer these questions:

- a) What variable must be initialized before the loop starts?
- b) What condition should control whether the loop continues?
- c) What must change inside the loop so that it eventually ends?

Now write the code for the program. The user may enter invalid input more than twice.

Model 2 Random Numbers

You can generate a seemingly random sequence of numbers using the Python random module. A mathematical function is used to produce the sequence based on a *seed* value. If no seed is given, the current system time is used.

Python code	Shell output
import random	
randint(1, 10)	NameError: name 'randint' is not defined
random.randint(1, 10)	5
random.randint(1, 10)	2
seed(100)	NameError: name 'seed' is not defined
random.seed(100)	
random.random()	0.1456692551041303
random.random()	0.45492700451402135
random.seed(100)	
random.random()	0.1456692551041303
random.random()	0.45492700451402135

Questions (20 min)

Start time:

11. What is the name of the module that must be imported to generate random numbers?
12. Based on Model 2, what are the names of three functions defined in the random module?
13. Explain the reason for the name errors in the table above.
14. What is the effect on the “random” numbers generated after calling the seed method?

15. Describe one reason to set the same seed each time a program is run, and one reason *not* to use the seed method.

16. Run `random.random()` in a shell multiple times. Based on the results, describe:

- a) the range of numbers returned by the random function
- b) the nature of the distribution of numbers generated. (Do they appear clustered around a particular value, or are they spread out uniformly over the range?)
- c) Type `help(random.random)` in the shell to check your answers.

17. Run `random.randint(1, 10)` in a shell multiple times. Based on the results, describe:

- a) the range of numbers returned by the randint function
- b) the nature of the distribution of numbers generated. (Do they appear clustered around a particular value, or are they spread out uniformly over the range?)
- c) Type `help(random.randint)` in the shell to check your answers.

18. Write an if statement to simulate a coin toss. The code should print "Heads!" or "Tails!", with 50% probability. *Hint:* `random.random()` is less than 0.5 about 50% of the time.

19. Write a statement that generates a random integer between 80 and 100, inclusive, and assigns the integer to a variable named `score`.