CSCI-UA-0002

Intro to Computer Programming (No Prior Experience)

Module 4: Condition-Controlled (While) Loops

Professor Emily Zhao

Section 008
T/R 12:30-1:45PM

Section 012
T/R 4:55-6:10PM

Agenda

- Review Ed Questions
- Module 4 Review
- Practice Problems

Module 4

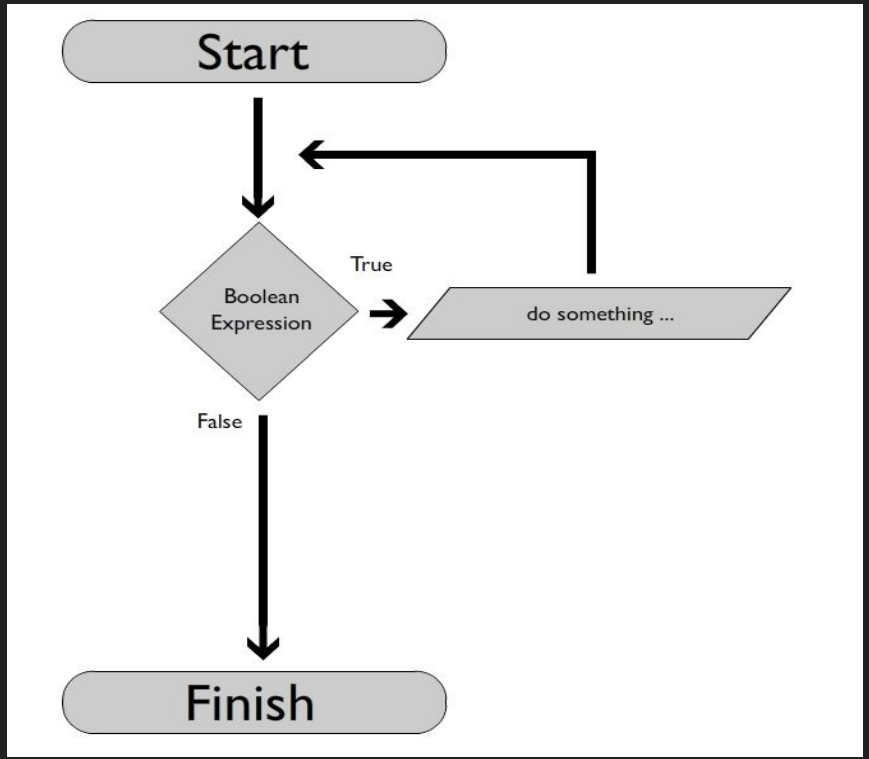
- Repetition Structures + `while` loops
- Self-referential assignment statements
- Accumulator variables
- “While” loop control
  - Infinite loops
  - `break` and `continue`
  - Sentinels
- Data Validation
- Color in Turtle Graphics

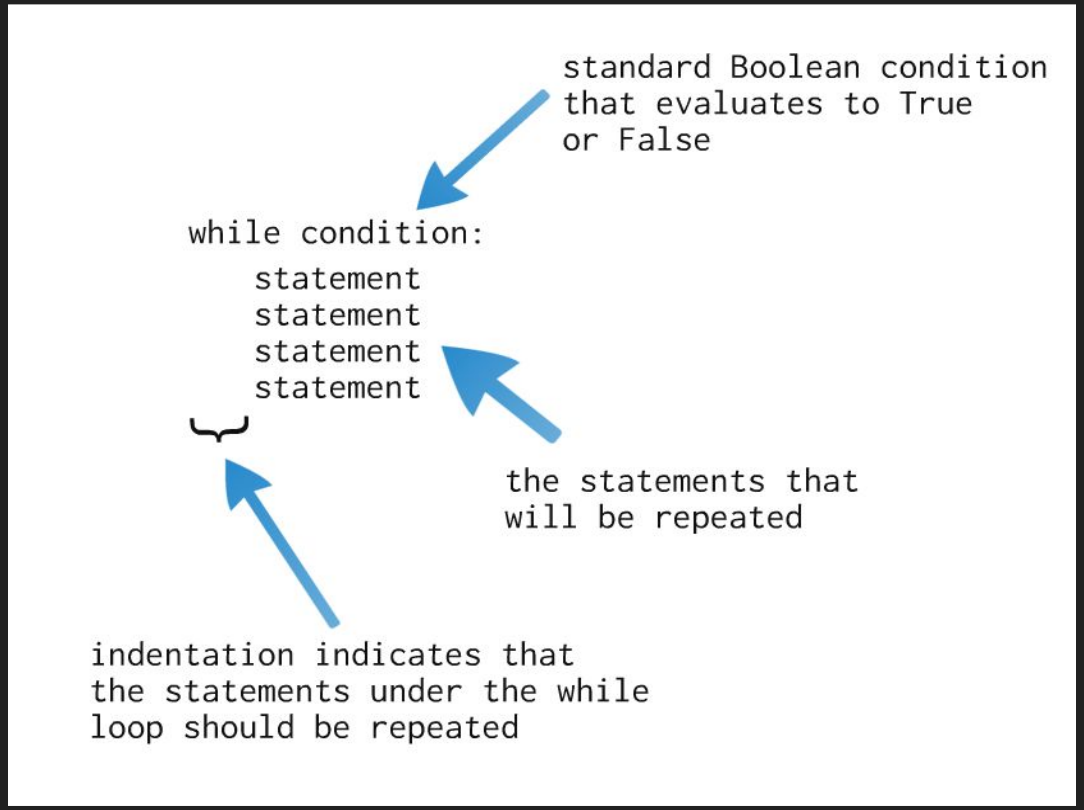
Your Questions

→ How do you best count how many times a loop runs?
→ What does it mean to “iterate”?
→ Can you review `+=` and `-=`?
→ What is an accumulator variable?
→ Can you go over `continue`?

“While” loops

A **condition-controlled loop** is a programming structure that causes a statement or set of statements to repeat as long as a condition evaluates to true




standard Boolean condition
that evaluates to True
or False
while condition:
statement
statement
statement
statement
the statements that
will be repeated
indentation indicates that
the statements under the while
loop should be repeated

Reasons to use a while loop

Repetition
Repeat a task an uncertain amount of times until a specific condition is met

Data Validation
Repeatedly prompt the user for input until they provide valid data

User Interaction
Require user input or interaction until the user chooses to exit

Reasons to use a while loop

Repetition
You want to keep rolling a dice until you get a 6

Data Validation
You want to ensure the user enters a positive number, and keep prompting them until they do

User Interaction
You are creating a simple menu-driven program that continues to display options to the user until they choose to exit

Repetition with Uncertain Iteration Count

Scenario: You want to keep rolling a dice until you get a 6.

```
1 import random
2
3 roll = 0
4 while roll != 6:
5     roll = random.randint(1, 6)
6     print("You rolled a", roll)
```

Data Validation

Scenario: You want to ensure the user enters a positive number, and keep prompting them until they do

```
num = -1

while num <= 0:
    num = int(input("Enter a positive number: "))
    if num <= 0:
        print("Invalid input. Please enter a positive number.")
```

User Interaction

Scenario: You are creating a simple menu-driven program that continues to display options to the user until they choose to exit

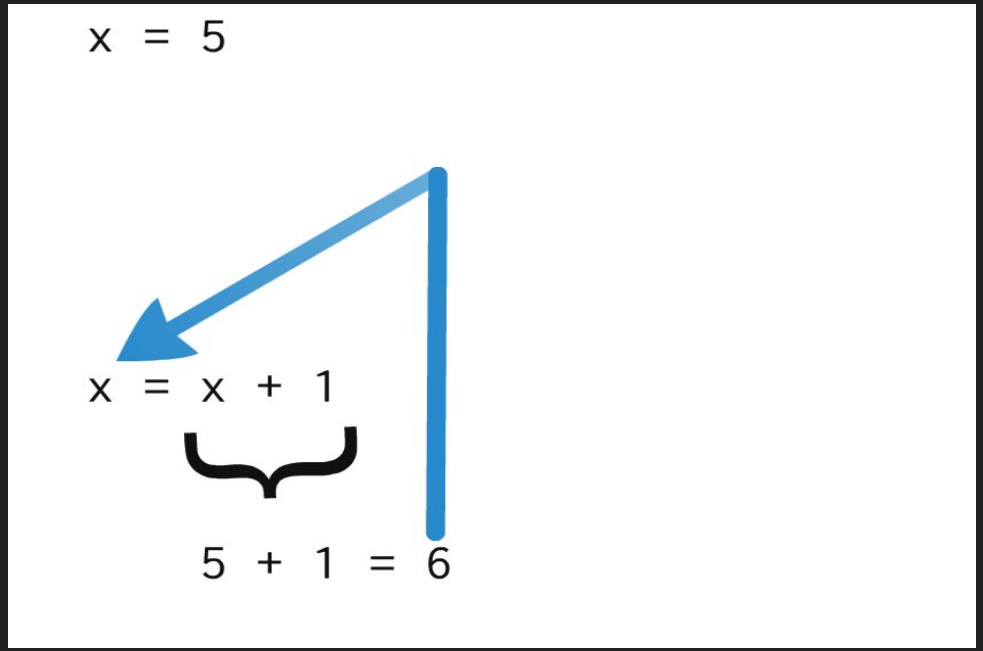
```
choice = ""
while choice != "4":
    print("Menu:")
    print("1. View profile")
    print("2. Edit settings")
    print("3. Help")
    print("4. Exit")

    choice = input("Enter your choice: ")

    if choice == "1":
        # View profile logic
    elif choice == "2":
        # Edit settings logic
    elif choice == "3":
        # Help logic
    elif choice == "4":
        print("Goodbye!")
    else:
        print("Invalid choice. Please try again.")
```

Let’s practice counting

Self-referential assignment statements




```
x = x + 1
```

```
x += 1
```

```
a = 0
b = 4

while a < b:
    a += 1  # Enters the loop: a = 0,
            # Run 1: a -> 1,
            # Run 2: a -> 2,
            # Run 3: a -> 3,
            # Run 4: a -> 4,
            # -> will not enter loop a 5th time
```

Python Tutor: Visualize code in Python, JavaScript, C, C++, and Java

Write code in Python 3.6 [reliable stable version, select 3.11 for newest]

```
1
```

Visualize Execution | NEW: Get AI Help | If you use ChatGPT or other AI, take this survey

hide exited frames [default] | inline primitives, don't nest objects [default] | draw pointers as arrows [default]

Show code examples

```
# counter variable
count = 5

while count < 10:
    print(count)
    count -= 1
```

Infinite loop

(control + c to end)

```
# counter variable
count = 0

while count < 10:
    count += 1
    print(count)
```

> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10

`break`

The break keyword is used to break out a loop.

`continue`

Used to end the current iteration in a loop
and continues to the next iteration.

```
# counter variable
count = 0

while count < 10:
    count += 1
    if count % 2 == 0:
        continue
    print(count)
```

> 1, 3, 5, 7, 9

```
# counter variable
count = 0

while count < 10:
    count += 1
    if count % 2 == 0:
        break
    print(count)
```

```
> 1
```

Two ways to set up your while loops

while condition == True:

Keeps looping until the condition is no longer true (either arithmetic happens so that it is no longer true, or we set the condition to false)

while True:

Immediately enters and continues to loop indefinitely until the program encounters a `break`

```
# set up control variable
keepAsking = True

if keepAsking == True: # enters the loop

    # will keep asking user to give them an integer
    num = int(input("Give me an integer: "))

    # until the condition is no longer true
    if num > 0:
        print("Thank you for inputting a valid integer")
        keepAsking = False
    else:
        print("Keep trying")
        # go back to the top of the loop and ask for input again
        # because keepAsking is still true
```

```
while True: # enter loop immediately

    # will keep asking user to give them an integer
    num = int(input("Give me an integer: "))

    # until the program reaches a BREAK
    if num > 0:
        print("Thank you for inputting a valid integer")
        break
    else:
        print("Keep trying")
        # go back to the top of the loop and ask for input again
```

Programming Challenge: Divisibility Tester

Write a program that lets the user test to see if a series of numbers are evenly divisible by 3. If they are, print out a status message telling the user.

Extension: Start off by asking the user to enter in the number that should be used during the test (i.e. enter 5 if you want to test to see if a range of numbers is evenly divisible by 5)

Accumulator Variable

- a container that keeps track of a running total
- typically initialized to an initial value and then updated or modified
- used in loops to add up or collect information as the loop progresses
- a “virtual piggy bank”

Sentinels

A sentinel value is a predefined value that the user can type in to indicate that they are finished entering data

Example:

```
>> Enter a test score (type -1 to end): 100
>> Enter a test score (type -1 to end): 80
>> Enter a test score (type -1 to end): -1
>> Your test average is:  90 %
```

Programming Challenge: Adding Machine

- Write a program that continually asks the user for an integer
- Add the supplied integer to a total variable
- When the user enters a 0 value end the program and display the sum for the user

Programming Challenge: Marbles

- Assume you have a jar that contains 5 marbles. The jar can hold 10 marbles total.
- Continually ask the user if they want to add or remove a marble
- If they add a marble you should increase the total # of marbles in the jar. If the jar is full tell the user and end the program.
- If they remove a marble you should decrease the # of marbles in the jar. If the jar is empty you should tell the user and end the program.

Homework

- Assignment #3 (due Thurs @ 11:59PM)
- Self-Paced Learning Module #5 (due next Tues)
- Quiz #5 (due next Tues)