

Objectives

- Indefinite Loops

1

Reflection on Lab

- Solving problems with programming is important!
- **Articulating** what your program is doing and the **tradeoffs** of how you wrote your code is also important!
 - Tradeoffs could be measured in efficiency, readability, reusability, how easily changed, ...

2

Review

- Problem: We are judging a science fair. There is different criteria for winning a first place ribbon, depending on what grade the student is in. Given the variables `scienceScore` and `grade`
 - Write a condition that will evaluate to True if (and only if) the student's score is above the first place threshold of 60 points and the student's grade is 8.
 - Otherwise, the condition should evaluate to False
- Synthesis: What questions should you ask to solve problems once you realize that you need a conditional?
 - How do the answers to these questions inform your solution?

Feb 15, 2023

Sprenkle - CSCI111

3

3

Review

- Problem: We are judging a science fair. There is different criteria for winning a first place ribbon, depending on what grade the student is in. Given the variables `scienceScore` and `grade`
 - Write a condition that will evaluate to True if (and only if) the student's score is above the first place threshold of 60 points and the student's grade is 8.
 - Otherwise, the condition should evaluate to False

```
scienceScore > 60 and grade == 8
```

Feb 15, 2023

Sprenkle - CSCI111

4

4

Solving Problems with Conditionals

- Broadly: What are the special cases? (You know you need a conditional)
- What code needs to execute in certain circumstances?
 - This is the *body* of your if (or elif or else)
- Under what conditions does that code execute?
 - This is the *condition* of your if (or elif)
- Are there multiple conditions? Are they mutually exclusive?
 - Informs you about nesting/if/else
- There are other questions, but this is a good start

Feb 15, 2023

Sprenkle - CSCI111

5

5

INDEFINITE LOOPS

Feb 15, 2023

Sprenkle - CSCI111

6

6

Definite vs Indefinite Loops

- **for** loops are **definite** loops
 - Execute a *fixed* number of times
- **Indefinite** loops: keep iterating until certain conditions are met
 - Depending on condition, no guarantee in advance of how many times the loop body will be executed

Feb 15, 2023

Sprenkle - CSC1111

7

7

While Loop Syntax

```
keyword → while condition :  
        statement1  
        statement2  
        ...  
        statementn } body of while loop
```

- Like a *looped if* statement
 - Execute statements **only** when condition is true
 - Stop executing when condition is false

Feb 15, 2023

Sprenkle - CSC1111

8

8

While Loop

```
i = 0
while i < 5 :
    print("i equals", i)
    i+=1
print("Done", i)
```

Feb 15, 2023

Sprenkle - CSC1111

while.py

9

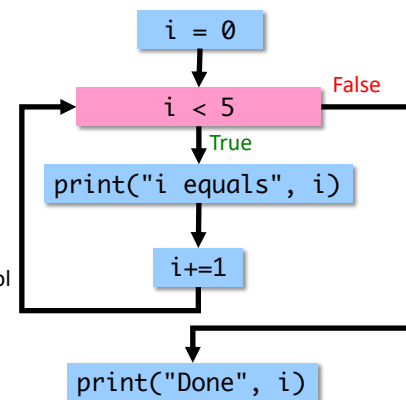
9

While Loop

```
i = 0
while i < 5 :
    print("i equals", i)
    i+=1
print("Done", i)
```

Questions:

- Explain how the code maps to the control flow diagram
- How many times will `i` get printed out?
- How many times is the condition evaluated?
- What is the value of `i` after the loop?



Feb 15, 2023

Sprenkle - CSC1111

while.py

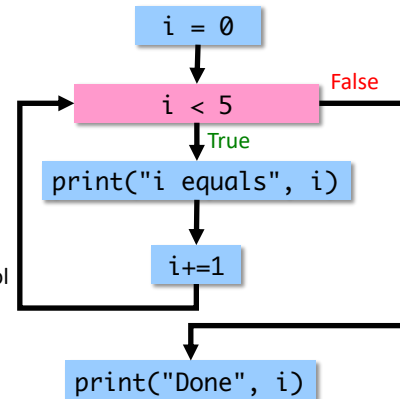
10

10

While Loop

```
i = 0  
while i < 5 :  
    print("i equals", i)  
    i+=1  
print("Done", i)
```

Initialize i before using in condition



Questions:

- Explain how the code maps to the control flow diagram
- How many times will `i` get printed out?
- How many times is the condition evaluated?
- What is the value of `i` after the loop?

Feb 15, 2023

Sprenkle - CSC1111

while.py

11

11

While vs. For Loops

- **Any** `for` loop can be translated into a `while` loop
- But **NOT** vice versa
 - Only *some* `while` loops can be converted into `for` loops
 - `while` loops are more *powerful* than `for` loops

Feb 15, 2023

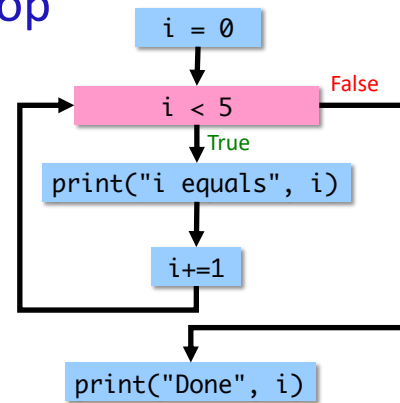
Sprenkle - CSC1111

12

12

Convert to a for loop

```
i = 0
while i < 5:
    print("i equals", i)
    i+=1
print("Done", i)
```



We *can* convert this `while` loop into a `for` loop because it executes a *fixed* number of times.

Feb 15, 2023

Sprenkle - CSC1111

while.py

13

13

Comparing while and for

- What are the main differences between these loops?
- What are the advantages and disadvantages of each?

```
i = 0
while i < 5:
    print("i equals", i)
    i+=1
print("Done", i)
```

```
for i in range(5):
    print("i equals", i)
print("Done", i+1)
```

Feb 15, 2023

Sprenkle - CSC1111

whilevsfor.py

14

14

What Does This Loop Output?

```
count = 1
while count > 0:
    print(count)
    count += 1
```

Feb 15, 2023

Sprenkle - CSC1111

loop.py

15

15

Infinite Loop

- Condition will never be **False** so keeps executing

```
count = 1
while count > 0:
    print(count)
    count += 1
```

- To stop an executing program in Linux use
➤ **Control-C**

Feb 15, 2023

Sprenkle - CSC1111

16

16

Infinite Loop Discussion

- Is there ever a time that an infinite loop is wanted?
 - Yes! For example in web servers, we have something like

```
while True:  
    listenForRequest()  
    handleRequest()
```

- Can a computer automatically detect infinite loops?
 - No, that is an **undecidable** problem
 - Best to **prevent** infinite loops (more later)
 - Benefit of **for** loops: *definite* loops

Feb 15, 2023

Sprenkle - CSC1111

17

17

while Loops

```
x=eval(input("Enter number:"))  
while x % 2 != 0 :  
    print("Error!")  
    x = eval(input("Enter number: "))  
print(x, "is an even number.")
```

What does this code do?

Feb 15, 2023

Sprenkle - CSC1111

18

18

while Loops

```
x=eval(input("Enter number:"))
while x % 2 != 0 :
    print("Error!")
    x = eval(input("Enter number: "))
print(x, "is an even number.")
```

Example of a **while** loop that cannot be transformed into a **for** loop (Why not?)

Feb 15, 2023

Sprenkle - CSCI111

19

19

A Very Simple Therapist

- Whenever a user tells the computer/program what they think, the program asks, "How does that make you feel?"
- Ends when user enters nothing ("")

Feb 15, 2023

Sprenkle - CSCI111

[therapist.py](#)

20

20

A Very Simple Therapist

- Whenever a user tells the computer/program what they think, the program asks, "How does that make you feel?"
- Ends when user enters nothing ("")
- Partial example output:

```
Tell me what is bothering you.  
There is too much going on in my life.  
How does that make you feel?  
I feel like I am out of control and can't juggle it all.  
How does that make you feel?  
Really stressed and tired.  
How does that make you feel?  
Thank you! Come again!
```

What questions should you ask to inform your solution?

therapist.py

Feb 15, 20

21

21

Solving Indefinite Loop Problems

- What needs to be repeated?
 - That tells you what is in the *body* of your loop
- When/under what circumstances should it be repeated?
 - That informs the loop's *condition*

Feb 15, 2023

Sprenkle - CSCI111

22

22

Design Pattern: Sentinel Loop

- Sentinel: when to stop

➤ “guard” to the loop

```
value = set value
while value != sentinel :
    process value
    value = set value
```

- “Keep going until you see the sentinel”

- Options for “set value”

➤ From a simple assignment, user input, calling a function, reading from a file, ...

Feb 15, 2023

Sprenkle - CSC1111

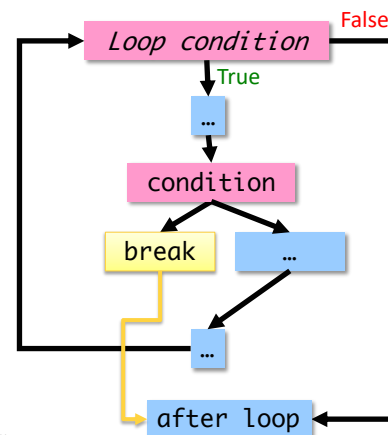
23

23

break Statement

- **break** statement “breaks” out of the current loop

Example Control Flow:



Feb 15, 2023

Sprenkle - CSC1111

24

24

while Loops using break

```
while True :  
    x = eval(input("Enter number:"))  
    if x % 2 == 0 :  
        break  
    print("Error!")  
print(x, "is an even number.")
```

What does this code do?
Think about the control flow

Feb 15, 2023

Sprenkle - CSCI111

25

25

while Loops using break

```
while True : Infinite loop!?!  
    x = eval(input("Enter number:"))  
    if x % 2 == 0 :  
        break "breaks" out of a loop  
    print("Error!")  
print(x, "is an even number.")
```

Internal condition says
when to stop

Feb 15, 2023

Sprenkle - CSCI111

26

26

while Loops: comparing use of break

```
x=eval(input("Enter number:"))
while x % 2 != 0 :
    print("Error!")
    x = eval(input("Enter number: "))
print(x, "is an even number.")
```

Loop condition says when to keep going

```
while True :
    x = eval(input("Enter number:"))
    if x % 2 == 0 :
        break
    print("Error!")
print(x, "is an even number.")
```

Internal condition says when to stop

Feb 15, 2023

Sprenkle - CSCI111

27

27

while Loops: comparing use of break

```
x=eval(input("Enter number:"))
while x % 2 != 0 :
    print("Error!")
    x = eval(input("Enter number: "))
print(x, "is an even number.")
```

Loop condition says when to keep going

```
while True :
    x = eval(input("Enter number:"))
    if x % 2 == 0 :
        break
    print("Error!")
print(x, "is an even number.")
```

Internal condition says when to stop

Using break statements:
Best when body of loop *has* to execute at least once.

Feb 15, 2023

28

28

Summary

- **while** vs. **for** Loops

- Any **for** loop can be translated into a **while** loop

- But NOT vice versa

- Only some **while** loops can be converted into **for** loops

- **while** loops are more powerful than **for** loops

- When you recognize you need a while loop, ask

- What needs to be repeated? (Loop body)

- Under what conditions does it need to be repeated/does it stop?

Feb 15, 2023

Sprenkle - CSCI111

30

30

Looking Ahead

- Lab 5 due Friday

- Broader Issue: Autonomous Vehicles

Feb 15, 2023

Sprenkle - CSCI111

31

31