Objectives

- Indefinite Loops

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, reusabililty, how easily changed, …
- Understanding the tradeoffs in small parts will help as you build larger, more complex programs

Review

- Problem: We are judging a science fair and awarding ribbons. 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 evaluates to `True` if (and only if) the student's grade is 8 and the student's score is above the first place threshold of 60 points. Otherwise, the condition evaluates 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?
- This is not well-written code. Why not? How could it be improved?

```
if check_condition() :
    pass
else:
    print("Does not meet condition")
```

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
```

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 about nesting/if/else
- There are other questions, but this is a good start

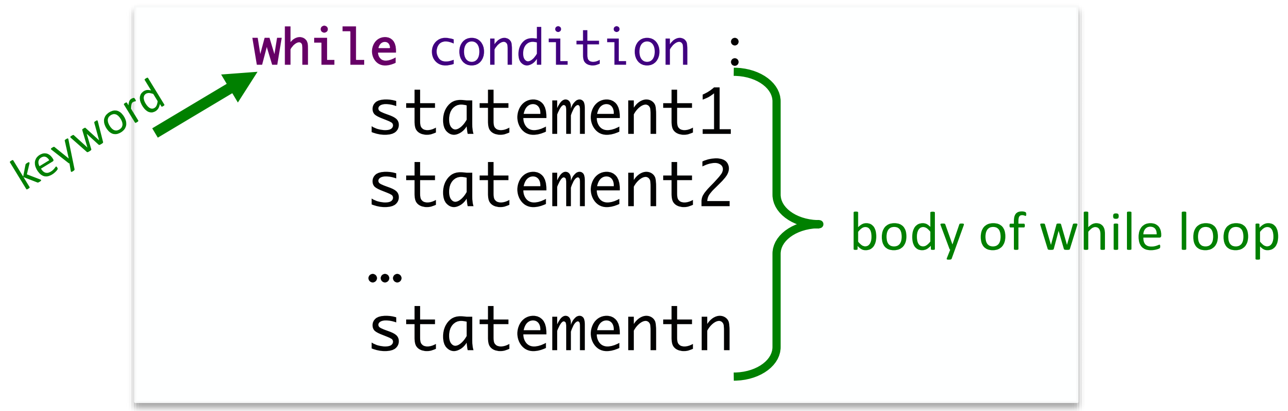
INDEFINITE LOOPS

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

While Loop Syntax



```
while condition :
    statement1
    statement2
    …
    statementn
```

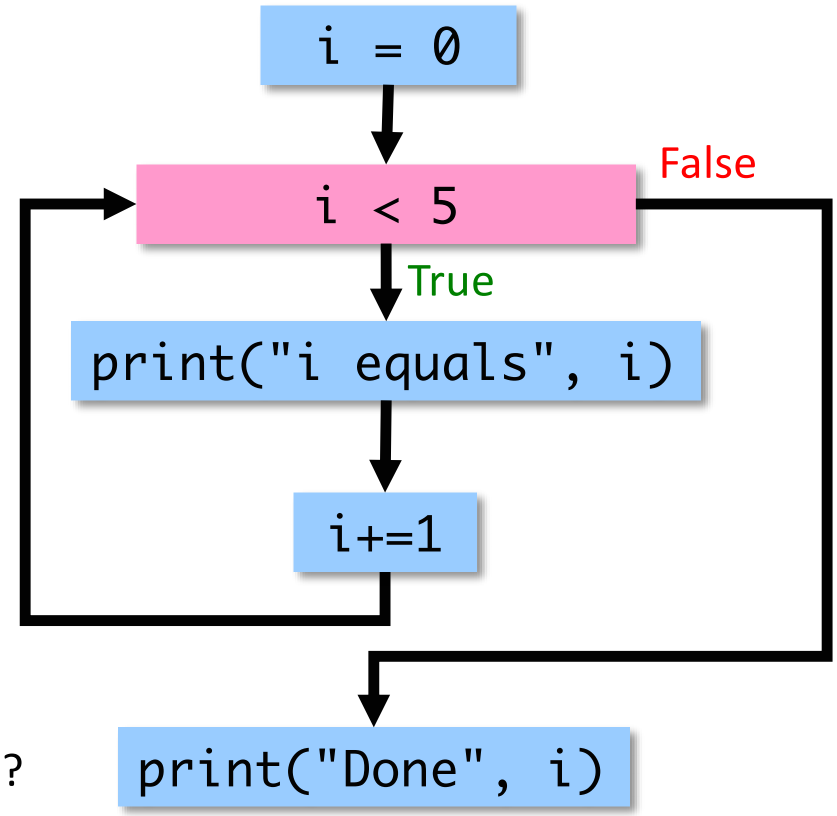
- Like a *looped* `if` statement
  - Execute statements **only** when condition is true
  - Stop executing when condition is false

While Loop

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

While Loop

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



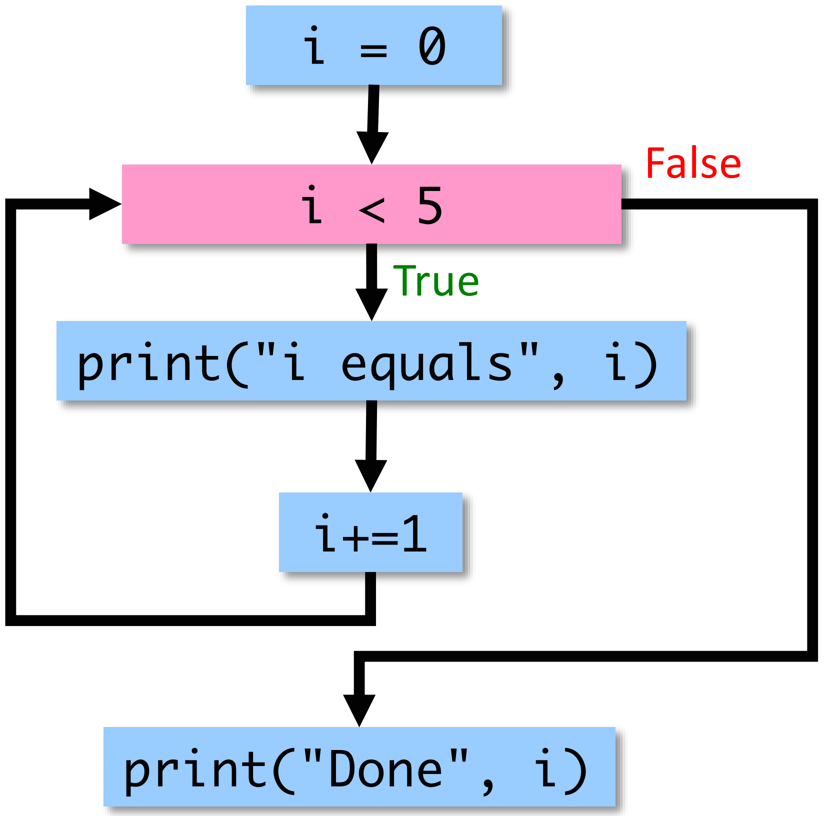
- Explain how the code maps to the control flow diagram
- How many times will `i` get printed out?
- How many times is the condition evaluated?
- What is the value of `i` after the loop?

while.py

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?

while.py

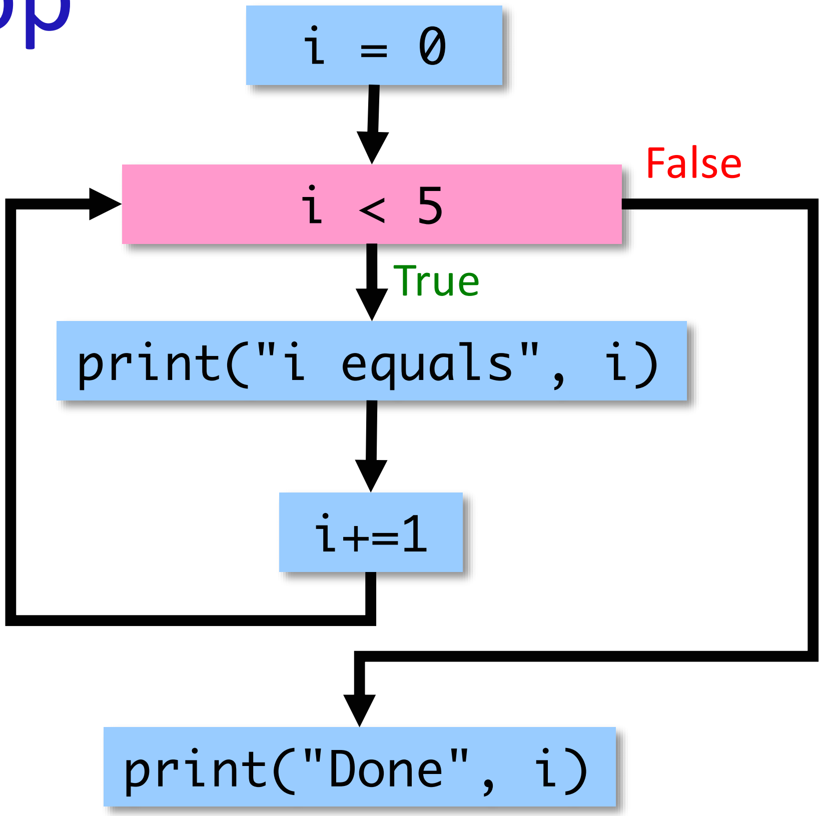
While vs. For Loops

- ***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

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 the loop executes a ***fixed*** number of times.

`while.py`

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)
```

What Does This Loop Output?

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

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

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

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?

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?)

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 ("")

`therapist.py`

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:

What questions should you ask to inform your solution?

```
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!
```

therapist.py

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*

Design Pattern: Sentinel Loop

- Sentinel: when to stop
  - “guard” to the loop

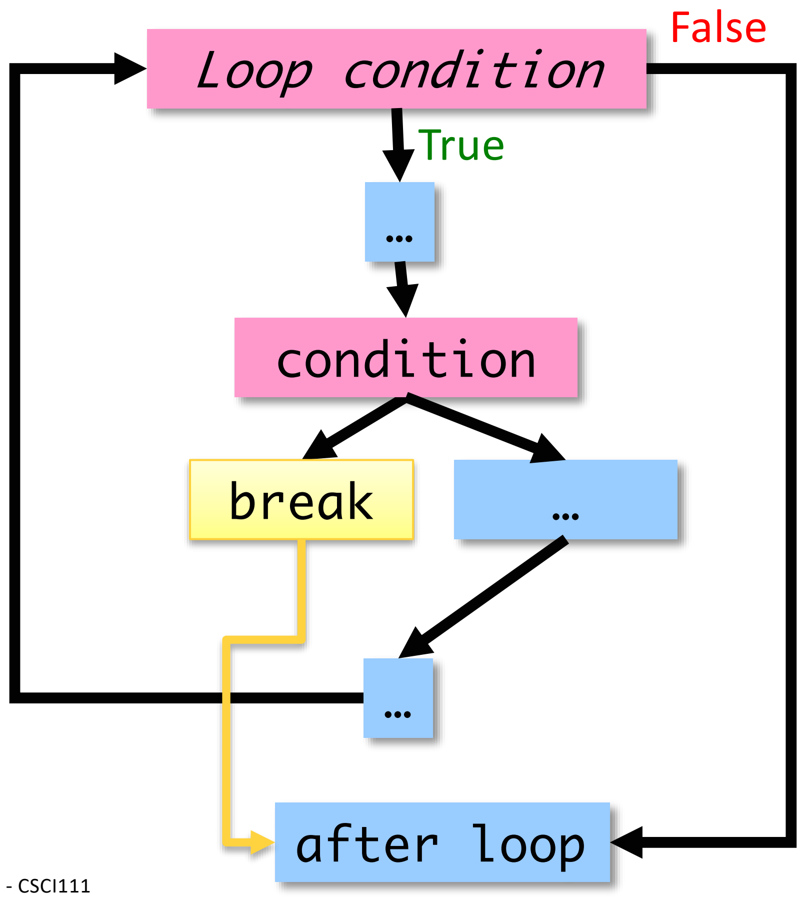
```
value = set value
while value != sentinel :
    process value
    value = set value
```

- “Keep going until you see the sentinel”
- Options for “set value”
  - From a simple assignment, user input, calling a function, reading from a file, …

`break` Statement

- `break` statement "breaks out of" or stops the execution of the current loop

Example Control Flow:



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

while Loops using break

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

Internal condition says when to stop

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.")
```

"breaks" out of a loop

Internal condition says when to stop

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.")
```

"breaks" out of a loop

Internal condition says when to stop

Using break statements:

Best when body of loop must execute at least once.

Transform Therapist

For practice, on your own

- Write using a `break` instead…

Looking Ahead

- Lab 5 due Friday
- Broader Issue due Thursday: Autonomous Vehicles