CISC101 Reminders & Notes

- Assignment 2 is posted
  - Due on Sunday, February 20th at 11:59AM
- Tests are being marked …
  - Grades will be entered in Moodle as soon as possible
  - Tests will be returned in tutorial next week
  - See me in my office hours if you wish to discuss your grade

Slides courtesy of Dr. Alan McLeod

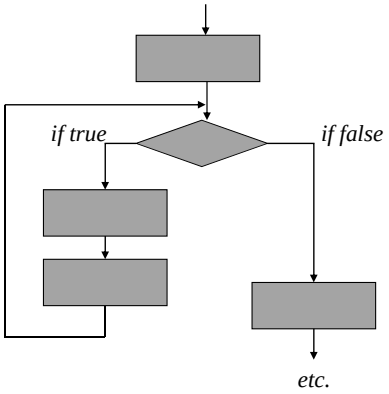
Today

- Finish up chained `if` statements
  - Continue from slide 23 last time
- Introduce loops
  - `while` loops today
  - Another loop next time …
- Look at "looping accessories"
  - `break`, `continue`, `pass` and `else`

Slides courtesy of Dr. Alan McLeod

Repetition or "Loops"

- What if we combine a conditional test with a branch back up to an earlier piece of code?



Slides courtesy of Dr. Alan McLeod

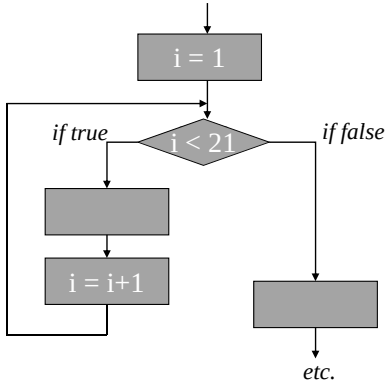
Repetition - Cont.

- The conditional test determines when to stop the repetition
  - As long as the condition is `True`, the loop keeps going
- Something inside the looping part must affect what is tested in the condition
  - What if it did not? What would happen?

Slides courtesy of Dr. Alan McLeod

Repetition - Cont.

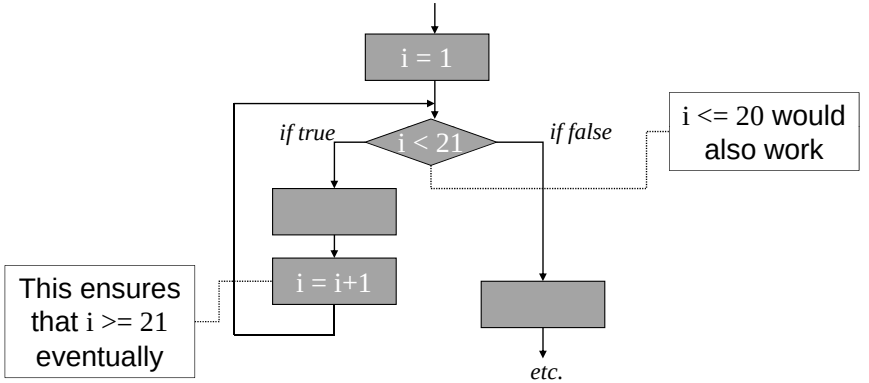
- Suppose we wanted this loop to execute only 20 times …



Slides courtesy of Dr. Alan McLeod

Repetition - Cont.

- Suppose we wanted this loop to execute only 20 times …



Slides courtesy of Dr. Alan McLeod

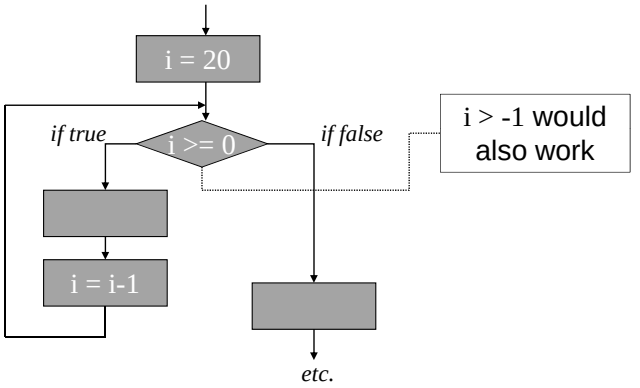
Repetition - Cont.

- The number of repetitions is controlled by changing the limit value for the *loop counter*
  - i in the example on the previous slide
- That example had i increasing by one each time
  - The loop counter was being *incremented* by one
- It could be incremented by any other value
  - "i = i + 2" would increment the loop counter by 2
  - This is the case for any value
- If the counter is decreased in value each time, it is being *decremented*

Slides courtesy of Dr. Alan McLeod

Repetition - Cont.

- Another way for the loop to execute 20 times …



Slides courtesy of Dr. Alan McLeod

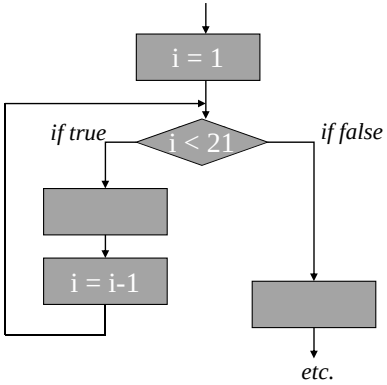
Repetition - Cont.

- There are lots of other ways of stopping a loop
  - Sometimes you don't need to count iterations at all
  - Condition might be based on …
    - user input (*e.g.*, looking for a specific values)
    - value returned from a function
    - *etc.*
- **But** you still need a conditional expression that is affected by something inside the loop
  - Asking for user input
  - Invoking a function
  - *etc.*

Slides courtesy of Dr. Alan McLeod

Repetition - Cont.

- Consider this example where i is decremented by one but the condition is i < 21 – what will happen?



Slides courtesy of Dr. Alan McLeod

Repetition - Cont.

- The dreaded "*infinite loop*"!
  - Caused by a logic error, not a syntax error
- The interpreter will not prevent you from coding a loop like the one shown - it will run!
- And run, and run, and run, and run, and run, and run, and run, and run, and run, and run...
- As a programmer, you must be "on guard" for such logic errors in your code

Slides courtesy of Dr. Alan McLeod

Python Loops

- Python has two kinds of looping syntax
  - **`while`** loops
  - **`for`** loops
- The loops are somewhat interchangeable
  - We'll use just the `while` loop for a while …
- The `while` loop is easier to use at first
  - … but the `for` loop is more powerful and versatile
- A `while` loop tests a condition
  - ... but the `for` loop iterates through elements

Slides courtesy of Dr. Alan McLeod

`while` loop

- A `while` loop can be used to code the structure shown in the flowchart shown previously
  - The "increment" one on slide 6

```
i = 1
while i < 21 :
    print(i)
    i = i + 1
```

- Let's try this out
  - Also try versions on slides 7, 9, and 11
- How do you stop an infinite loop in IDLE?

Slides courtesy of Dr. Alan McLeod

`while` loop - Cont.

- `while` loop syntax:

```
while boolean_expression :
    line1
    line2
    …
```

- As long as *`boolean_expression`* is `True`, the statements in the loop continue to execute

Slides courtesy of Dr. Alan McLeod

Demo - CalculateAverage.py

- Obtain any number of numbers from the user, sum them up and then display the average of the numbers
  - Ignore the possibility of non-numeric input
- How do we stop such a process?
- What about "special cases", like when no numbers are supplied?

Slides courtesy of Dr. Alan McLeod

Demo – Factorial.py

- Write a program that displays all the values of n! (or "n factorial") for 2! up to the user supplied value of n

$$n! = \prod_{i=1}^{n} i$$

- For example, 5! = 5 * 4 * 3 * 2, which is 120

Slides courtesy of Dr. Alan McLeod

Large Demo - DragonSlayer.py

- Game where a knight fights dragons
  - Knight has several properties
    - Current and maximum hit points
    - Maximum and minimum attack damage
  - Each dragon has the same
    - Values are generated randomly for each fight
- Knight can choose to fight a dragon or rest
  - Resting recovers hit points
- During a fight hits are exchanged
  - Knight can choose to fight or run away

Slides courtesy of Dr. Alan McLeod

Demo If Time - BoxDrawing.py

- Write a program that prints a square box outline of stars to the screen of a size provided by the user
  - The size must be at least 3, but no more than 80
- Note use of loop to ensure that a legal number is obtained
  - It still crashes with text input …
- Consider other ways to use a loop to simplify the middle part of the box
- You'll see this in the next lab

Slides courtesy of Dr. Alan McLeod

Use of `break` and `continue` With Loops

- Don't use these keywords unless you feel it makes your code easier to read and debug!
- **`break`** exits a loop immediately
- **`continue`** jumps to the next iteration immediately
- Demo: BreakContinue.py

Slides courtesy of Dr. Alan McLeod

Use of `else` and `pass` With Loops

- **`else`** with a loop gives you a chance to see how the loop exited
  - If it was a "normal" exit, the **`else`** is executed
    - Using `break` does not qualify as normal
- Demo: LoopElse.py
- **`pass`** is used when a statement is required, but you don't have anything you want to do!
  - Or you have not yet written the code
- Demo: LoopPass.py

Slides courtesy of Dr. Alan McLeod