While Loops

October 8

Yet another iterative control structure

We already have ‘for’ loops, why do we need another one?

‘For’ loops are good if we have a pretty good idea of how many times a loop is going to execute

- Five times
- “The value of some variable”

For loops will automatically keep track of how many times they’ve executed and even increment the index variable for us

While loops

But what if we need an iterator that runs a completely unknown number of times?

- We have no idea how many times it must run - maybe 0, maybe a few thousand, maybe infinite
- Example: keep asking for input until a human user actually gives us sensible input. We have no idea how long it's going to take before any given human stops playing games.

That's what a "while" loop is for. It runs as long as a boolean condition is "true."

While loops

Age = input(“please enter your age in years”)

#any number <= 0 is nonsensical; it can’t be correct.
#depending on context, maybe any number < 18 is invalid, because minors
#can’t use your program. So check:
While (age < 18):
    Print (“I’m sorry, but that’s not a valid input.”)
    Age = input(“Please enter a value greater than or equal to 18.”)

This loop will run anywhere from 0 to infinite times, and that’s what we want

Priming read

Note - that initial

Age = input(“please enter your age in years”)

Is called a “priming read.” It executes before the while is attempted for the first time. So it “primes the pump” by getting an initial value to test the first time through the loop.

Note that if the user enters a valid value - say, 35 - for this priming read, the boolean in the while statement evaluates to False immediately, and the loop NEVER executes. It executes 0 times.!!!

Literals, magic numbers, and constants

A “literal” is a defined string that shows up in your code, usually with no explanation. A “magic number” is an integer or float that shows up in your code, usually with no explanation about why it’s there or what happens if it changes.

```
If (tests_graded < 593):  # warning - magic number!!!
    # a bunch of code
    Tests_graded += 1
```

what’s the meaning of 593? Is it the total number of students in CMSC201? The total number of people taking the SAT? The point at which we’ll stop and break for pizza, then continue grading when we’re not hungry?

String literals

Suppose you want to print out a message like “Sorry but that is an error” in your program, but it has to be printed in a couple dozen places.

That gets repetitive and difficult to change.

So instead you declare a constant

Constants in Python

Constants are things that are declared and assigned a value once during a program run, and then not changed.

Note: unlike some other languages, Python doesn’t have a true “constant” type that CAN’T be changed somewhere in the program. So we enforce this ourselves with programming practices.

Declare a constant with ALL_CAPS:

```
PI = 3.14159
NUM_OF_201_STUDENTS = 593
```

Constants

Then make sure you don’t change the value anywhere in your code. (Hint: if you do, the Python interpreter will allow the change and then you’re going to be in big trouble the first time something goes wrong.)

You *CAN* change the value of a constant between runs of the program, and it’s quite easy: change
PI = 3.14159

To

PI = 3 #there’s a Purdue U. story in there

Constants

- Which of these are fine as just literal values?

```
count  = 0
count += 1
while count < 100:
if choice == 1:
if age < 0:
percent = num / 100
perimSquare = 4 * sideLen
print("Hello!")
while ans != "yes":
```

Sentinels

Sentinels are constants that hold a value that signals to a while loop that it should end:

```
EXIT = “exit”
if __name__ == "__main__":
  line = input(“Tell me something good.”)
  while line != EXIT:
    line = input(“Tell me something good.”)
```