While Loops

February 17, 2020

Administrative notes

- Course calendar now available on the github site.
- Homework 2 is due before midnight tonight!!
- Homework 3 is released. We'll talk about it today.
- Exam 1 is next Wednesday, the 26th, in class. You will only need a pencil; no laptop, phone, etc. will be permitted
    - Next, Monday, February 24, will be a review for the exam.
    - The exam will cover everything up through Wednesday, the 19th - that is, while loops are the last topic covered on this exam
-

if __name__ == “__main__”:

I originally told you to use this on homework #2, but then dropped that requirements because we hadn’t covered it in class. Now we’re covering it

There are two consecutive underscore characters (__) before and after name and main

__name__ is a system, or built-in variable whose value is the name of the module that’s running right now.

A Python program can contain code from many modules. You can import a module with pre-written Python by using the “import” statement

Importing modules:

The following statement in your program will import code that calculates a bunch of widely-needed math values:

import math

See https://docs.python.org/3/library/math.html for details

When that code is executing, the value of “__name__” is “math”

But when the code you just run is executing, the value of “__name__” is “__main__”

if __name__ == “__main__”

Using this statement ensures that the code contained in it - the code indented under the if statement - is ONLY executed when your module is directly executing and this is the main program.

If you write Python code, and somebody imports it into another program, your main program will *not* run.

while loops

“while” loops are the most general type of iteration in Python - anything you can solve with a “for” loop can also be solved with a “while” loop, but not everything that can be solved with a “while” loop can be solved with a “for” loop

- “for” loops are used when you know how many times you want to iterate through some code
  - You know the specific number of times
  - You know you want to iterate a number of times that is the value of a specific variable, such as the length of a list
- “while” loops are the only loop used when you don’t know how many times you will iterate through code

Syntax of a while loop

```
while boolean-condition-is-true:

    Code to be executed
```

Remember indentation. All the code that is indented underneath the “while” statement is executed as part of the loop. When you unindent, that code is no longer part of the loop.

Set the value of your boolean condition. Unlike with “for” loops, Python does not automatically set a value for a new variable used in the boolean condition of a “while.”

Examples:

```
age = 0;
while (age < 18):
    age = input(“enter your age in years: “)
    print (“If you’re 18 or older you should vote”)
print (“that’s the end of our story”)
```

What happens if we leave out the initial age = 0 statement?

How many times will this loop be executed?

Example: factorial

```
# compute 10! Using a while loop

product = 1

factor = 1

while factor <= 10:

    product *= factor   #or product = product * factor

    factor += 1
```

Common programming errors 1: Loop is never executed

Suppose we want to print out all of the even numbers between 2 and 100 inclusive. Why won’t this loop work?

```
num = 1

while num % 2 == 0:

    print(num)

    num += 2
```

It is perfectly acceptable to write a loop that may never be executed, due to other conditions in your code.

But be sure that that’s really what you want

A loop that never executes:

```
age = int(input(“please enter your age in
years: “))
while (age < 0) or (age > 100):
    print(“Age must be between 0 and 100
inclusive “)
    age = int(input(“please enter your age
in years: “))
```

If the user enters, say, 21 at the first prompt, the boolean condition is false at the start, and the code under the while is never executed.

That’s perfectly fine. Just make sure that it’s really what you wanted.

Common programming errors II: infinite loops

An infinite loop is one that never stops executing, because the boolean condition never becomes false.

Common causes:

- You never change a variable used in the boolean condition
- You plan to stop when a variable takes on a value that it will never take on

The code will never stop on its own. The program is only stopped by external action - you shut off the program or run out of resources. On gl.umbc.edu and similar machines, you can hit “Control” and “c”.

Note: infinite loops can occur with “for” loops, but you have to really work hard to make that happen. They’re rare.

Infinite loop examples

```
grade = ""
name  = ""
while name != "Hrabowski":
    # get the user's grade
    grade = input("What is your
grade? ")
print("You passed!")
```

```
cookiesLeft = 50

while cookiesLeft > 0:
    # eat a cookie
    cookiesLeft = cookiesLeft + 1
```

```
#print all the positive odd numbers
#less than 100

num = 1
while num != 100:
    print(num)
    num = num + 2
```