**Announcements**

None!

---

**COMP 141**

CS1: Programming Fundamentals

October 3, 2014

---

**break and continue Keywords**

**break** – immediately terminate a loop (breaks out of loop)

**continue** – ends the current iteration and goes to the end of the loop body (breaks out of iteration)

**Note:** Typically used with while loops, but can be used in a for loop as well.

**Do not overuse!**

Can make code difficult to read and debug!

---

**Using break**

```
#This program will add integers from #1 to 20 in this order to total, until #total is greater than or equal to 100.

total = 0
number = 0

while number < 20:
    number += 1
    total += number
    if total >= 100:
        break

print("The number is", number)
print("The total is", total)
```
Using continue

This program will add integers from 1 to 20 in this order to total, except for 10 and 11.

```python
total = 0
number = 0
while number < 20:
    number += 1
    if number == 10 or number == 11:
        continue
    total += number
print("The total is", total)
```

Using Boolean Variables in Loops

Given a positive integer n, assign True to `prime` if n has no factors other than 1 and itself.

If at any time during the loop, you find a factor of n other than 1 or n, set value of prime to False and break out of the loop.

(Remember, m is a factor of n if m divides n evenly.)

Sentinels

- **Sentinel**: special value that marks the end of a sequence of items
  - When program reaches a sentinel, it knows that the end of the sequence of items was reached, and the loop terminates
  - Must be distinctive enough so as not to be mistaken for a regular value in the sequence
  - Example: when reading an input file, empty line can be used as a sentinel

Using a Sentinel

For the following input, what is the value of temp?

```
4 7 5 9 -1
```

```python
def main():
    input1 = int(input("Please enter an integer: "))
    temp = 0
    while(input1 > 0):
        temp += input1
        input1 = int(input("Please enter another integer: "))
    print("Temp: ", temp)
main()
```
Input Validation Loops

• Computer cannot tell the difference between good data and bad data
  – If user provides bad input, program will produce bad output
  – GIGO: garbage in, garbage out
  – It is important to design program such that bad input is never accepted

Input Validation Loops

• **Input validation**: inspecting input before it is processed by the program
  – If input is invalid, prompt user to enter correct data
  – Commonly accomplished using a while loop which repeats as long as the input is bad
    • If input is bad, display error message and receive another set of data
    • If input is good, continue to process the input

Input Validation Loops (cont’d.)

The following code will ask for a test score and check to see if it’s a valid score. If not, it will re-prompt the user to enter the score.

```python
score = int(input("what is your score? "))
while score < 0 or score > 100:
    print("Not a valid score!")
    score = int(input("What is your score? "))
```
You can also do this with string inputs. (We will learn more useful ways to compare strings (lowercase to lowercase) later this semester.)

```python
shape = input("What is your shape? circle or square?")
while shape != "circle" and shape != "square":
    print("Please enter a valid shape.")
    shape = input("What is your shape? circle or square?")
print("We have a valid shape: ", shape)
```

### Equivalent Code

```python
Total_scores = 0
score = int(input("Score? "))
while score != 0:
    Total_scores = Total_scores + score
    score = int(input("Score? "))
print("the total is", Total_scores)
```

---

**Practice (x2)**

1. Write a program that simulates flipping a coin 1000 times, and displays the number of heads and tails.
   - You can write it as a while or for loop.
   - You’ll need to use the random number generation function we learned
     ```python
     number = random.randint(lowest, highest)
     ```

2. Write a program that prompts the user to enter a number between 50 and 100. If they don’t follow instructions and enter a number outside that range, re-prompt them. Continue to re-prompt them to enter a number until the number they enter is between 50 and 100. Print out their legal input.
   - Note: Test your code by inputting values outside the range (at least 2) to make sure it’s working properly.

**Next Time**

Nested Loops
Read Section 4.7