Condition Controlled Loops

Introduction to Programming - Python

Repetition Structures

- Programmers commonly find that they need to write code that performs the same task over and over again

Example: Commission calculator for a sales team

- Write a program that allows the user to calculate sales commission earned by each member of a sales team.
- Currently there are 3 people on the sales team, but there may be more in the future.
- Input
  - Gross sales (float)
  - Commission Rate (float)
- Process
  - Commission = gross sales * commission rate
- Output
  - Commission earned

+
Repetition Structures

- In the previous example our code ended up being one long sequence structure which contained a lot of duplicate code
- There are several disadvantages to this approach
  - Your programs will tend to get very large
  - Writing this kind of program can be extremely time consuming
  - If part of the duplicated code needs to be corrected then the correction must be implemented many times

Repetition Structures

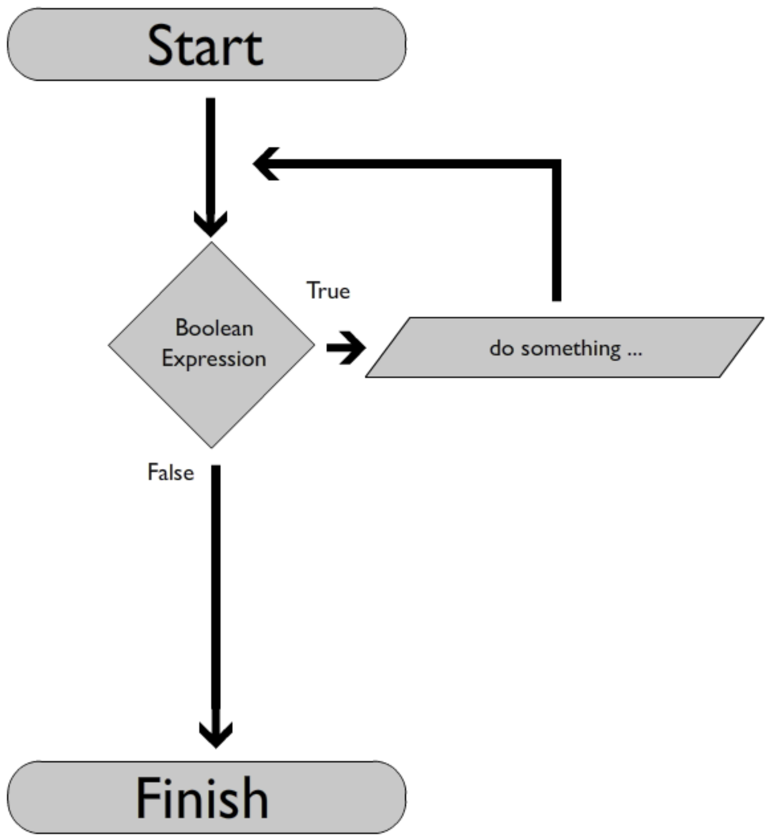
- One solution to this kind of problem is to use a repetition structure, which involves the following:
  - Write the code for the operation one time
  - Place the code into a special structure that causes Python to repeat it as many times as necessary
- We call this a "repetition structure" or, more commonly, a "loop"
- There are a variety of different repetition structures that can be used in Python

Condition Controlled Loops

Condition Controlled Loops

- A condition controlled loop is programming structure that causes a statement or set of statements to repeat as long as a condition evaluates to True

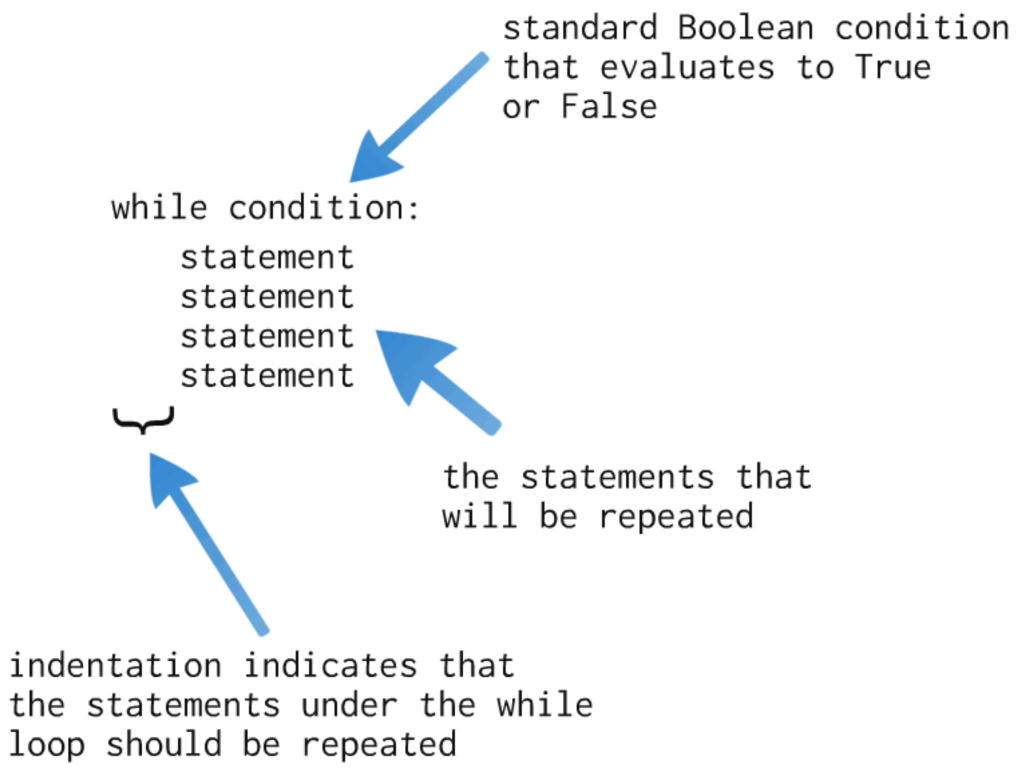
Condition Controlled Loops



The “While” Loop

- In Python we can implement a condition controlled loop by writing a “while” loop
- “while” loops work as follows:
  - Evaluate a Boolean expression.
  - If it is False, skip the block of statements associated with the while loop and condition the program as normal
  - If it is True
    - Execute a series of statements.
    - At the end of the statement block re-evaluate the condition
    - If it is True, repeat the block of statements
    - If it is False, skip the block of statements associated with the while loop and continue the program as normal

+ The "While" Loop



```
while condition:
    statement
    statement
    statement
    statement
```

standard Boolean condition that evaluates to True or False
the statements that will be repeated
indentation indicates that the statements under the while loop should be repeated

Programming Challenge: Commission Calculator

- Write a program that allows the user to calculate sales commission earned by each member of a sales team.
- Input
  - Gross sales (float)
  - Commission Rate (float)
- Process
  - Commission = gross sales * commission rate
- Output
  - Commission earned

Some notes of "while" loops

- We refer to the process of going through a loop as an "iteration"
- If a loop cycles through 5 times then we say we have "iterated" through it 5 times
- The "while" loop is considered a "pre-test" loop, meaning that it only iterates upon the successful evaluation of a condition
- This means that you always need to "set up" your loop prior to Python being able to work with it (i.e. setting up a control variable)

Warning!

- When working with a "while" loop there is nothing to prevent you from writing a Boolean condition that will never evaluate to False
- If this happens your loop will continue executing forever, or until you send an "interrupt" to IDLE using the CTRL-C key combination
- We call this an "infinite loop" since it never stops executing
- With the exception of a few special cases you want to try and avoid writing infinite loops

Trace the Output

```
a = 5

while a < 10:

    print ("A is less than 10!")
```

Programming Challenge: Temperature Conversion

- Write a program that allows the user to convert a temperature in Fahrenheit into Celsius using the following formula
  - Celsius = (Fahrenheit – 32) * 5/9
- After calculating the temperature ask the user if they wish to continue. If so, repeat the conversion with a new number. Otherwise end the program.

Divisibility Tester

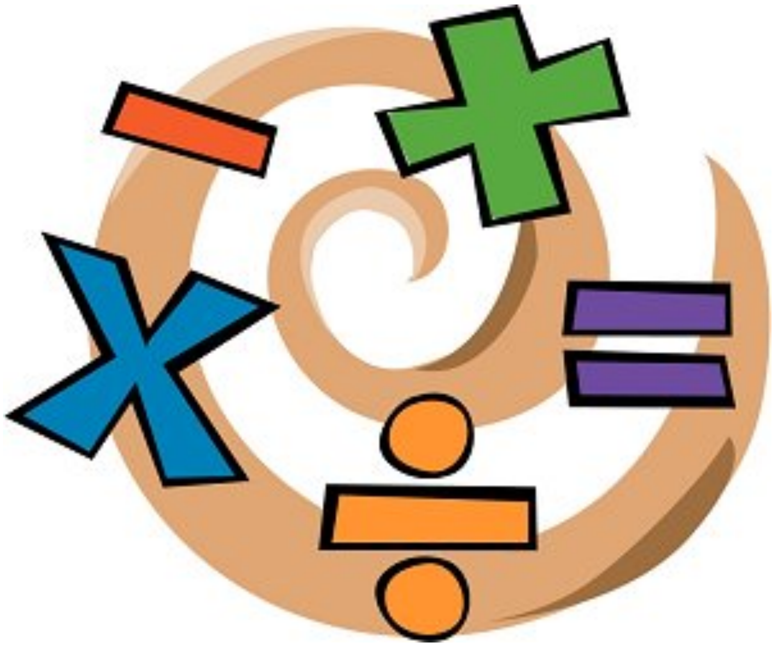
- Write a program that lets the user test to see if a series of numbers are evenly divisible by another number (3). If they are, print out a status message telling the user.
- Extension: Start off by asking the user to enter in the number that should be used during the test (i.e. enter 5 if you want to test to see if a range of numbers is evenly divisible by 5)

+
Programming Challenge: Combo Lock

- Write a program that asks the user for three numbers
- Test those numbers against three "secret" numbers that represent the combination to a virtual padlock
- If the user gets the numbers right you should let them know that they have gained access to your program
- If not, allow them to continue to enter combinations until they guess correctly

Programming Challenge: Arithmetic Quiz

- Write a program that asks the user to answer a simple math problem (5 + 6)
- Continually prompt the user for the correct answer. If they answer correctly, congratulate them and end the program. If they answer incorrectly you should re-prompt them for the answer a second time.



+ Accumulator Variables and Augmented Assignment Operators

Accumulator Variables

- Many programming tasks require you to calculate the total of a series of numbers or the number of times you iterate through a loop (kind of like your homework!)
- We can utilize an "accumulator" variable to do this.

+
Using Accumulator Variables

- Set up your accumulator variables outside of your loops. I generally initialize my accumulator variables right before I enter a repetition structure.
- Decide on a value you want to start your accumulator values at. 0 or 0.0 is generally a good starting point depending on whether you are counting whole numbers or numbers with fractional values.
- Use a self-referential assignment statement when incrementing an accumulator variable. Example:
  - counter = counter + 1

+

Self-referential assignment statements

```
# default x to 5
x = 5
```

$$x = \underbrace{x + 1}_{5 + 1 = 6}$$

Programming Challenge: Average Commission

- Write a program that asks the user to continually enter in the following information for a group of employees:
  - Sales
  - Commission Rate
- Calculate the commission earned by multiplying sales * commission rate
- Keep track of the following information and print out a summary document at the end of your loop
  - Number of employees
  - Total sales
  - Average sales
  - Total commission due

Augmented Assignment Operators

- The self-referential assignment statement that we just used is extremely useful, and can be extended to use any of the other math operations we have covered in class so far.
  - a = a + 1
  - b = b * 2
  - c = c / 3
  - d = d - 4

+

Augmented Assignment Operators

- However, Python (and most other programming languages) contains a series of "shortcuts" that can be used to cut down the amount of typing when working with self-referential assignment statements.
- We call these shortcuts the "augmented assignment operators"

Augmented Assignment Operators

Operator	Usage	Equal to
+=	c += 5	c = c + 5
-=	c -= 2	c = c – 2
*=	c *= 3	c = c * 2
/=	c /= 3	c = c / 3
%=	c %= 3	c = c % 3

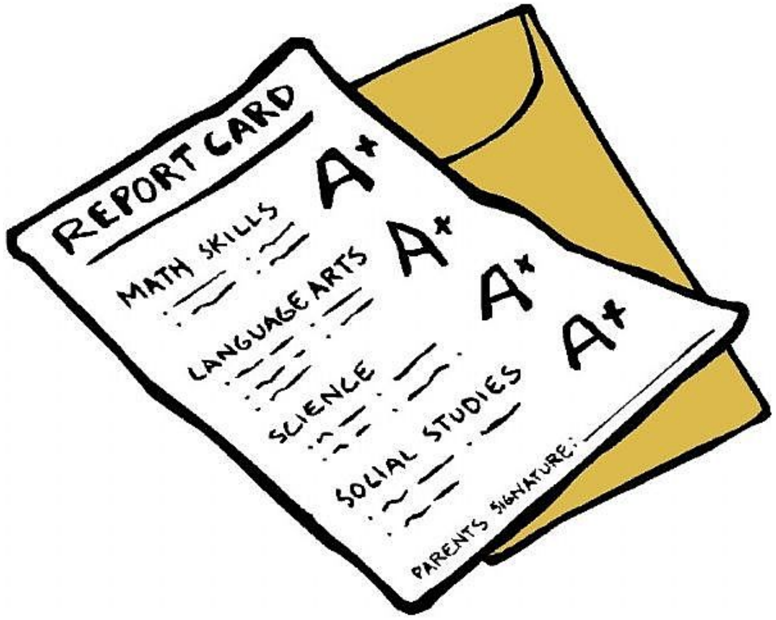
+

Programming Challenge: Grocery Checkout Calculator

- Write a program that asks the user to enter in a series of price values
- Calculate a running total of these values
- Calculate sales tax (7%) on the total bill and display the result to the user at the end of the program

Programming Challenge: My Grades

- Write a program that asks the user to enter in a test score along with the total points possible for the test
- Allow the user to enter in as many scores as he or she wishes
- When finished, calculate the user's average score in the class



+

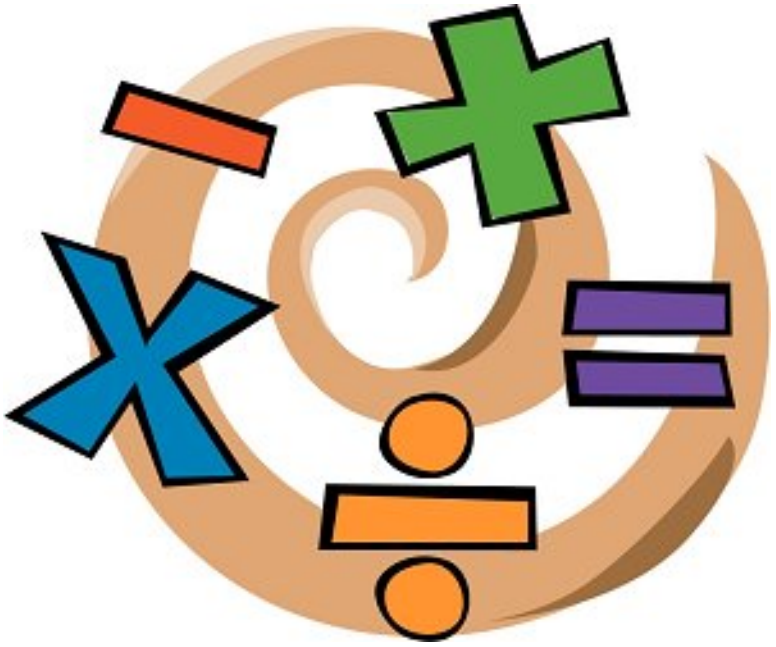
Programming Challenge: Coin Flips

- Write a program that simulates a coin flipping 1 million times
- Count the # of heads and tails that result, and display the result to the user after you have finished running the simulation

+

Programming Challenge: Math Quiz Part II

- Write a program that asks the user 5 simple math problems
- Each problem should utilize random numbers, but you can standardize on a single operation (i.e. subtraction)
- Ask the user a question. If they answer correctly, they earn a point. If not, they do not earn a point.
- At the end of the program present the user with their score.



+

Sentinels

+
Sentinels

- Imagine that you want to ask your users to enter in a large number of items that need to be calculated in a certain way.
- You don't know how many values the user will be entering.
- Given our current toolset we really only have ways to handle this kind of scenario:
  - Ask the user at the end of each iteration if they want to continue. This can be annoying and make your program cumbersome if you will be entering in hundreds or thousands of values.
  - Ask the user ahead of time how many items they will be entering. This can be difficult since the user may not know at the beginning of the loop how many items they will be working with.

+
Sentinels

- A sentinel value is a pre-defined value that the user can type in to indicate that they are finished entering data

- Example:
  - >> Enter a test score (type -1 to end): 100
  - >> Enter a test score (type -1 to end): 80
  - >> Enter a test score (type -1 to end): -1
  - >> Your test average is: 90 %

- In the example above the value -1 is considered a sentinel -- it indicates to the program that the user is finished entering data.

- Sentinels must be distinctive enough that they will not be mistaken for regular data (in the previous example the value -1 was used – there is no way that a "real" test value could be -1)

Programming Challenge: Adding Machine

- Write a program that continually asks the user for an integer
- Add the supplied integer to a total variable
- When the user enters a 0 value end the program and display the sum for the user

Programming Challenge: Weight Loss Log

- Write a program that asks the user to enter in a series of weight measurements taken over a period of days
- The user can enter as many or as few weight values as they would like. Entering the value "0" should indicate that the user has finished entering data.
- Calculate the user's average weight during this period
- Also calculate their weight change from the beginning of their weight loss program to the end of the program

+

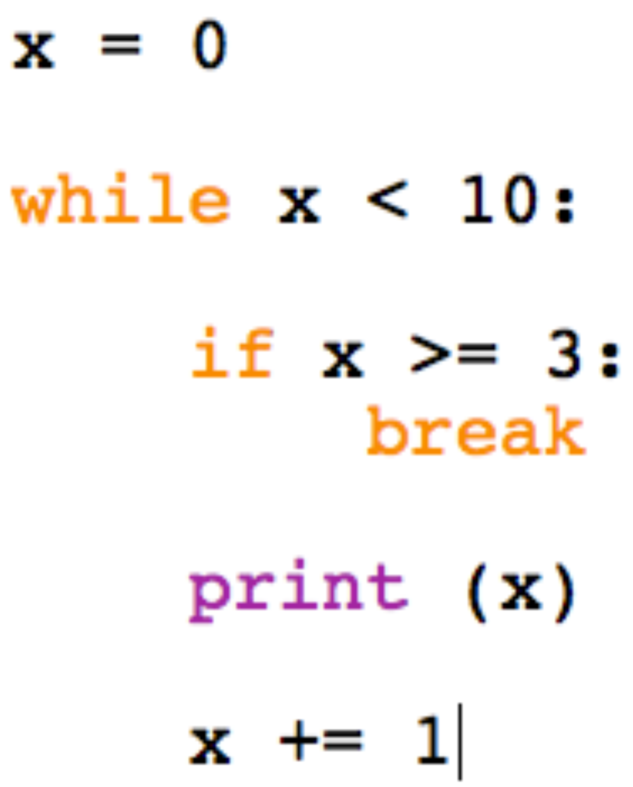
Repetition Flow Control

+

The "break" command

- The "break" command is a special Python command that can be used to immediately end a loop.
- It will not, however, end your program – it simply ends the current repetition structure and allows the program to pick up from the line directly after the end of your loop
- Note that when the break command runs it will immediately terminate the current loop, which prevents any commands after the break command from running

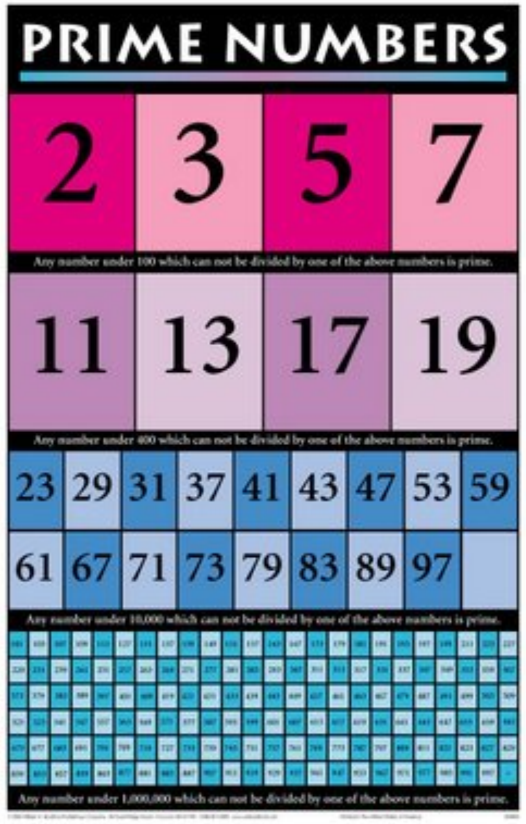
Trace the Output



```
x = 0

while x < 10:

    if x >= 3:
        break

    print (x)

    x += 1
```

Prime Number Tester

- Write a program that asks the user for an integer
- Test to see if the number is prime. A prime number is any number that is evenly divisible by 1 and itself.


PRIME NUMBERS
2 3 5 7
Any number under 100 which can not be divided by one of the above numbers is prime.
11 13 17 19
Any number under 400 which can not be divided by one of the above numbers is prime.
23 29 31 37 41 43 47 53 59
61 67 71 73 79 83 89 97
Any number under 10,000 which can not be divided by one of the above numbers is prime.
Any number under 1,000,000 which can not be divided by one of the above numbers is prime.

+

Simple Data Validation

+
Simple Data Validation

- Often we need to ask the user to supply a value in our programs
- But as you know you can't always trust the user to supply you with usable data!
- One strategy you can use to ensure that you get "good" data is to "validate" the user's input. This involves asking the user for a value – if it meets our criteria we can continue. If not we will need to ask the user to re-supply the value.

Programming Challenge

- Write a program that asks the user for a positive integer
- Do not accept a negative value (or zero) – if the user supplies an invalid value you should re-prompt them
- Once you have a positive integer you can print that number of stars to the screen. For example:

```
Enter a positive integer:  -5
Invalid, try again!
Enter a positive integer:  0
Invalid, try again!
Enter a positive integer:  5

*****
```

+

Infinite Loops

Infinite Loops

- What happens when the condition is never false?
- The loop will run forever! (Or at least … as long as the computer is powered up …)
- In general, we want to be avoid infinite loops.

+

Infinite Loops: An Artistic View

- M.C. Escher (1898-1972) was a Dutch graphic artist who used mathematical concepts in some of his drawings. (http://en.wikipedia.org/wiki/M._C._Escher)
- Here is one vision of an infinite loop: