+
Warm Up: Loan Qualification

- You're working for a small bank that wants to write a program to allow its customers to pre-qualify themselves for a personal loan

- Rules for qualification are as follows:
  - Borrower must make more than $50,000 per year and be at his or her job for at least 2 years
  - The 2 year job requirement can be waived, however, for borrowers making more than $100,000 per year

- Write a program to ask the user for their yearly salary as well as the # of years they have been at their current company. Use the rules above to output the string 'You qualify' or 'You do not qualify'

+

Condition Controlled Loops

Introduction to Programming - Python

+

Repetition Structures

Programming Challenge: Even or Odd

- Write a program that asks the user to enter a number and reports whether is even or odd.

+
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

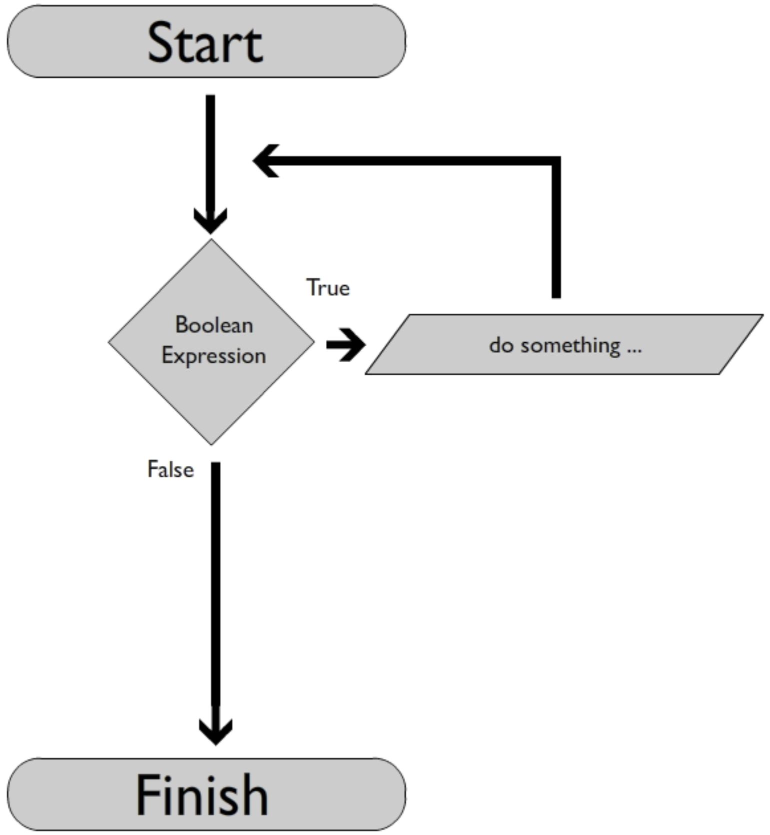
+

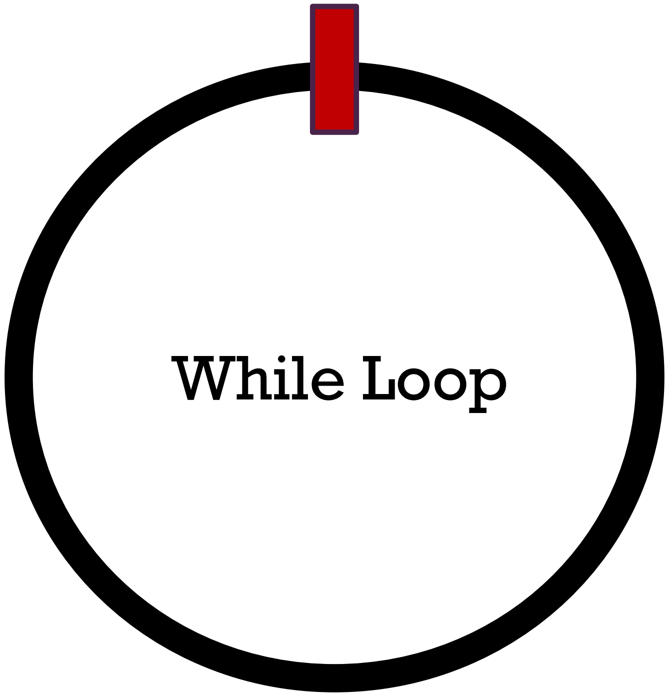
Condition Controlled Loops

+
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

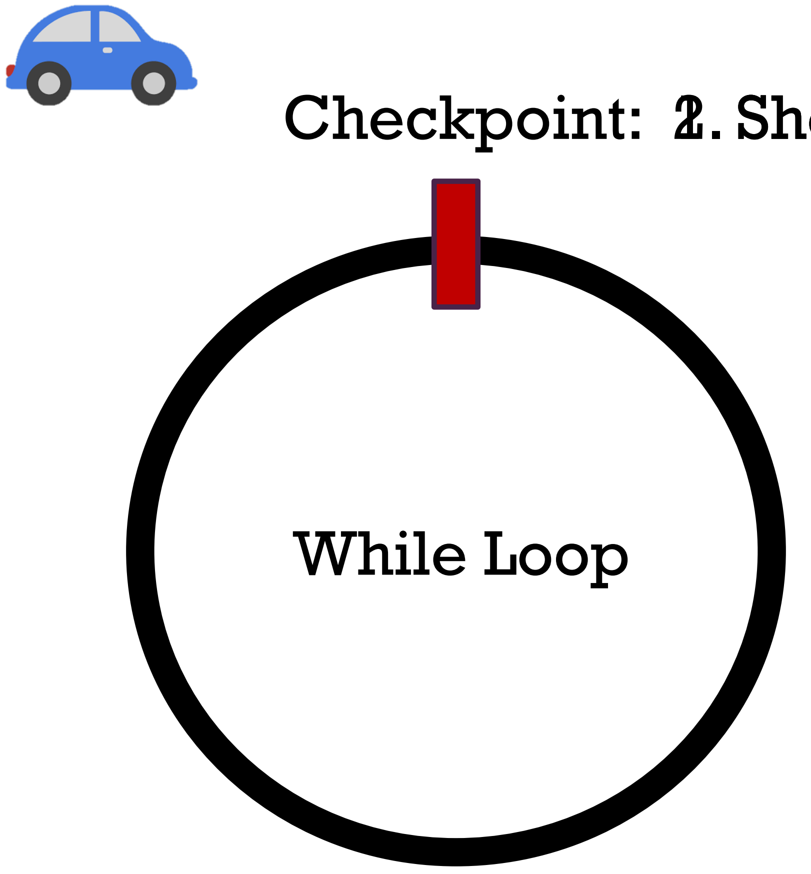




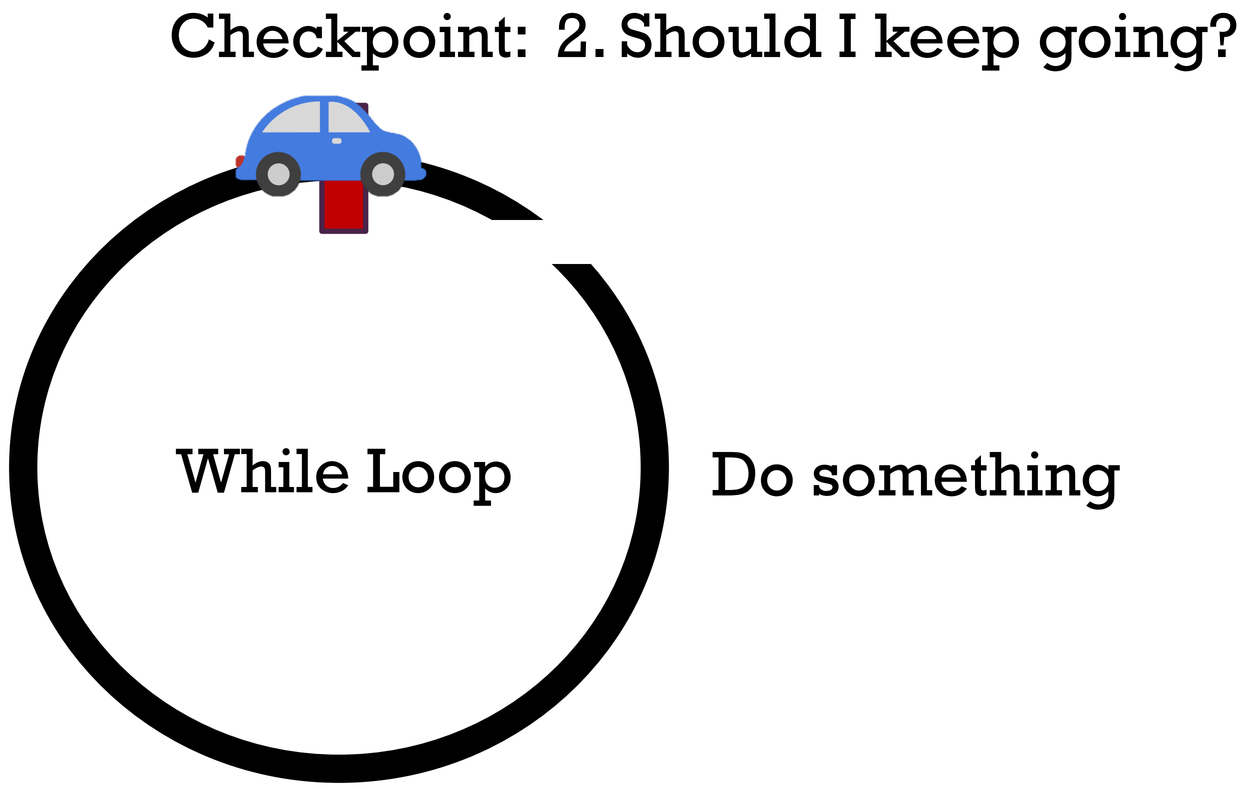
Do something

Checkpoint: 1. Should I enter?

2. Should I keep going?



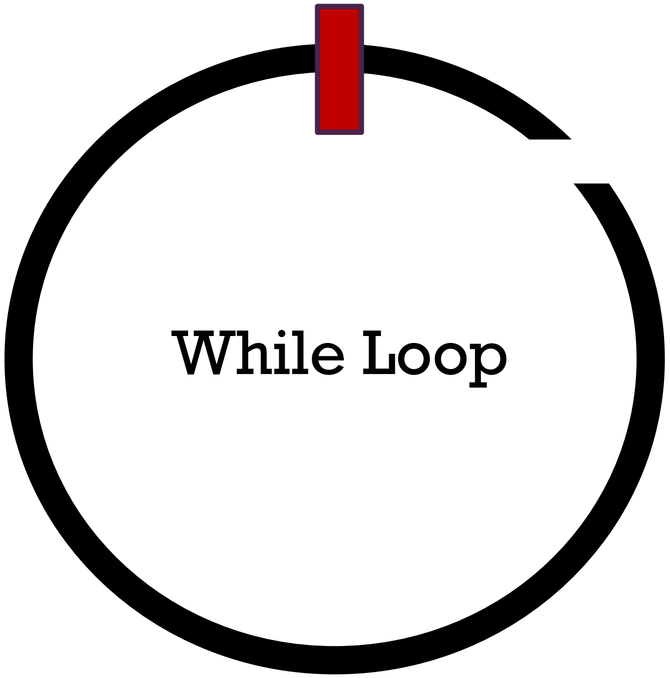
Do something


Checkpoint: 2. Should I keep going?
While Loop
Do something

+

laps=0

Checkpoint: laps<2



Output:

Going around the track

Going around the track

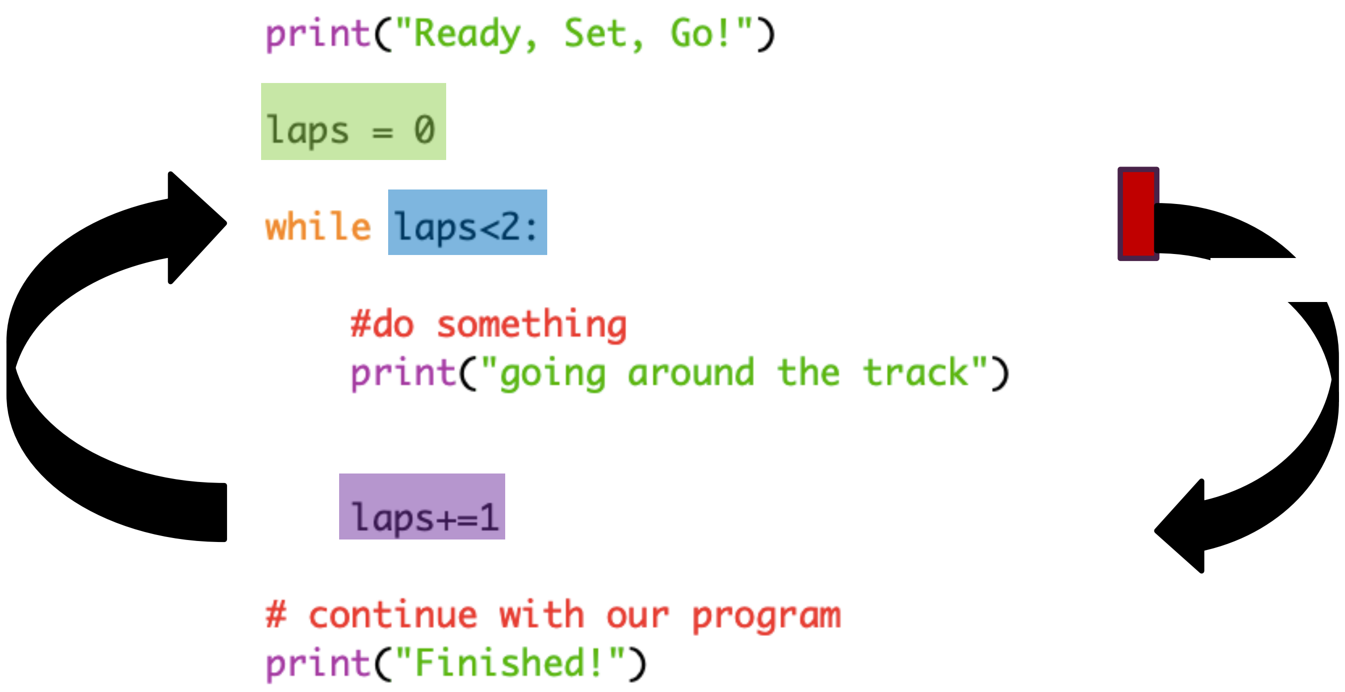
Do something

Print("Going around the track")

laps= laps + 1

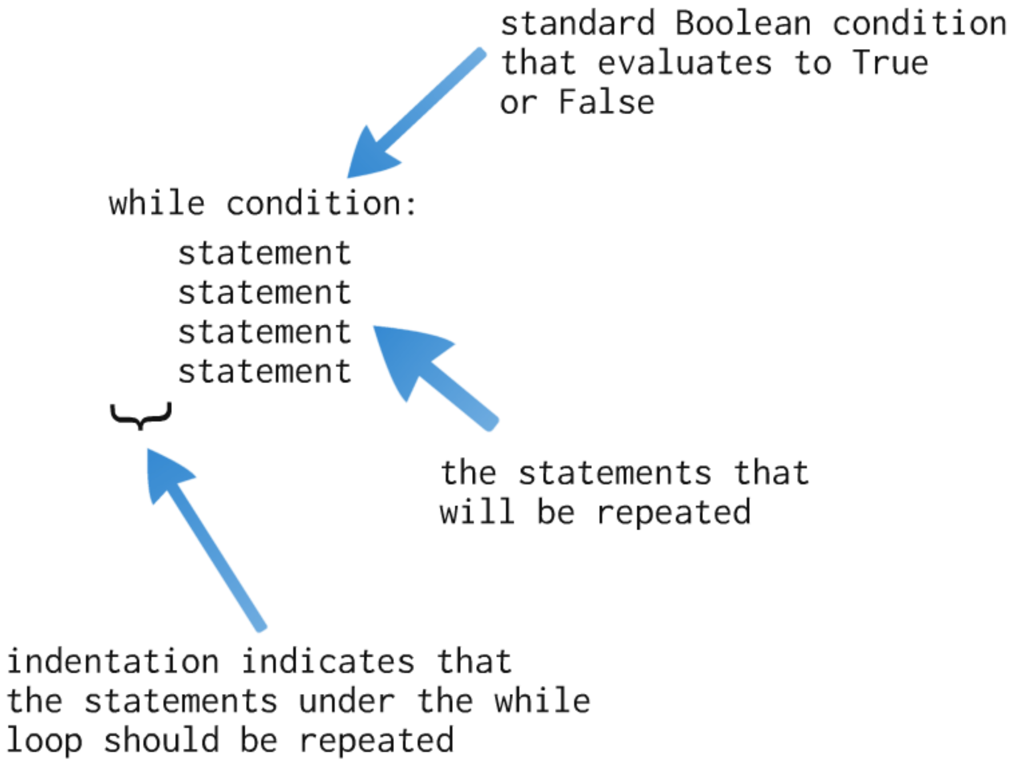
laps+=1

What does this look like in Python Code?



```
print("Ready, Set, Go!")

laps = 0

while laps<2:

    #do something
    print("going around the track")


    laps+=1

# continue with our program
print("Finished!")
```

The "While" Loop



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

standard Boolean condition that evaluates to True or False

the statements that will be repeated

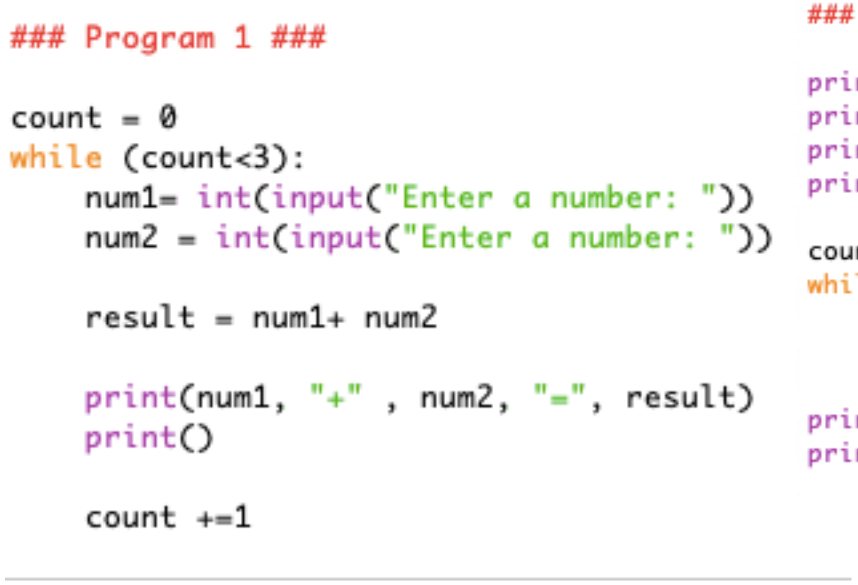
indentation indicates that the statements under the while loop should be repeated

3 Requirements for FINITE while loops

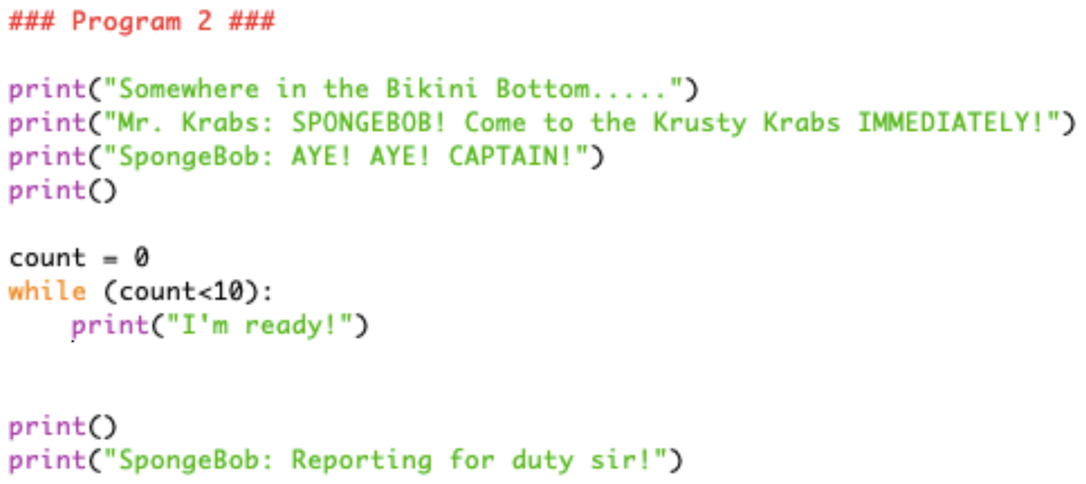
- 1. Define our accumulator variable OUTSIDE the while loop
- 2. Define our condition (checkpoint)
- 3. Update our accumulator variable during every lap (iteration) of our while loop.

Let's Practice!

For each program below, identify the three requirements of while loops and how many times code inside the while loop will be repeated.

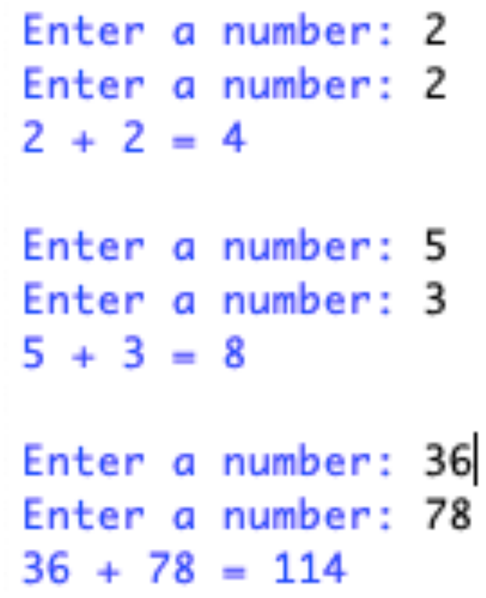


```
### Program 1 ###

count = 0
while (count<3):
    num1= int(input("Enter a number: "))
    num2 = int(input("Enter a number: "))

    result = num1+ num2

    print(num1, "+" , num2, "=", result)
    print()

    count +=1
```

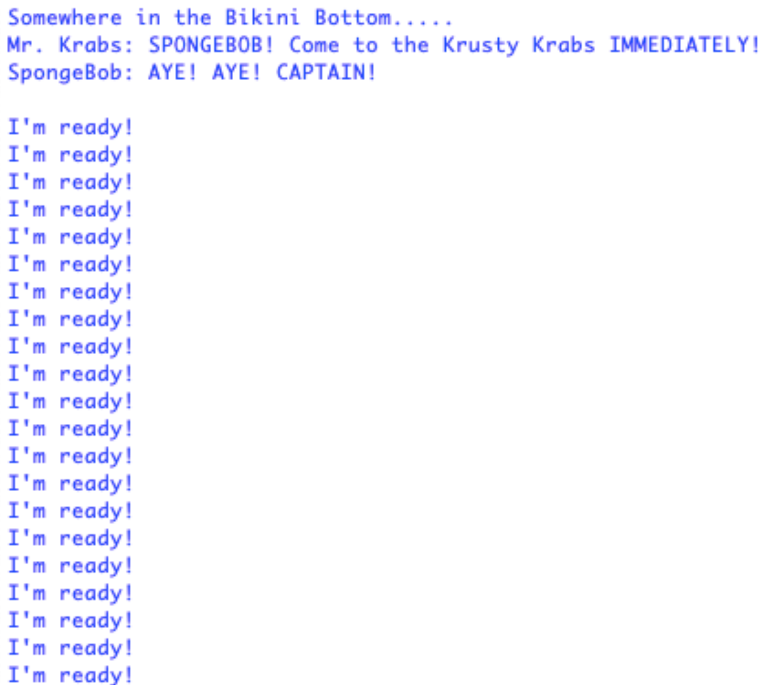


```
### Program 2 ###

print("Somewhere in the Bikini Bottom.....")
print("Mr. Krabs: SPONGEBOB! Come to the Krusty Krabs IMMEDIATELY!")
print("SpongeBob: AYE! AYE! CAPTAIN!")
print()

count = 0
while (count<10):
    print("I'm ready!")


print()
print("SpongeBob: Reporting for duty sir!")
```

Answers



```
Enter a number: 2
Enter a number: 2
2 + 2 = 4

Enter a number: 5
Enter a number: 3
5 + 3 = 8

Enter a number: 36
Enter a number: 78
36 + 78 = 114
```



```
Somewhere in the Bikini Bottom.....
Mr. Krabs: SPONGEBOB! Come to the Krusty Krabs IMMEDIATELY!
SpongeBob: AYE! AYE! CAPTAIN!

I'm ready!
I'm ready!
I'm ready!
I'm ready!
I'm ready!
I'm ready!
I'm ready!
I'm ready!
I'm ready!
I'm ready!
I'm ready!
I'm ready!
I'm ready!
I'm ready!
I'm ready!
I'm ready!
I'm ready!
I'm ready!
I'm ready!
I'm ready!
I'm ready!
```

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.

+

Programming Challenge: Guess the Number

- Rewrite the "guess the number" game we wrote back in the selection statement unit to use a "while" loop

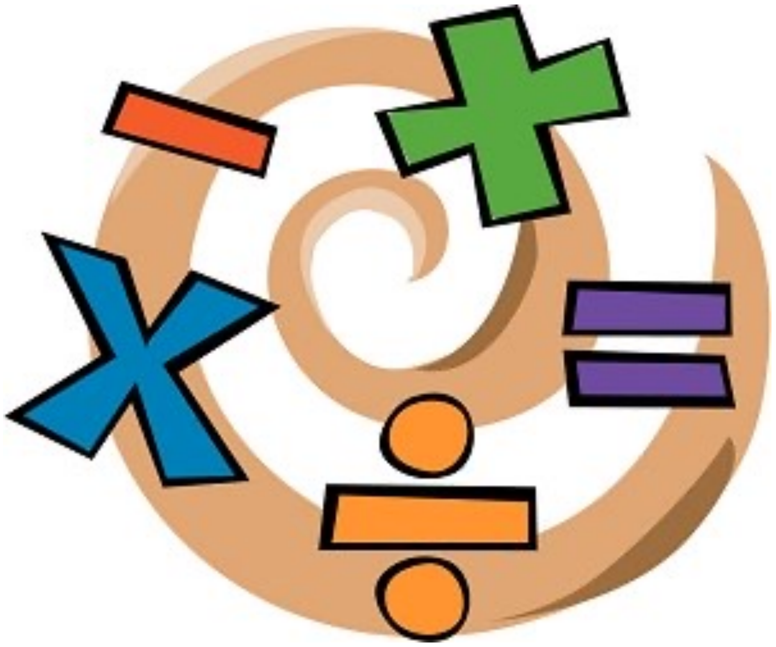
- Allow the user to continually guess a number until they eventually guess the correct number

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.
- Extension: Randomize the numbers used in the math problem
- Extension: Randomize the type of problem the user is presented with (i.e. addition, subtraction)



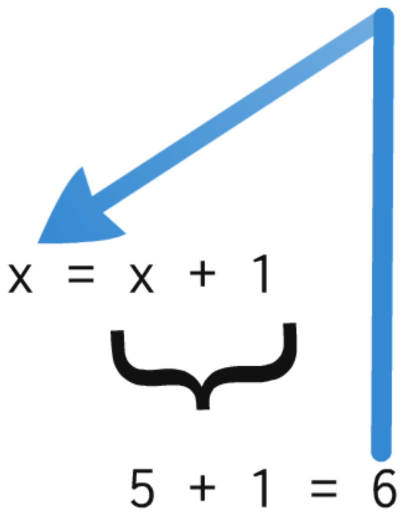
Accumulator Variables and Augmented Assignment Operators

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 = x + 1
```

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

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

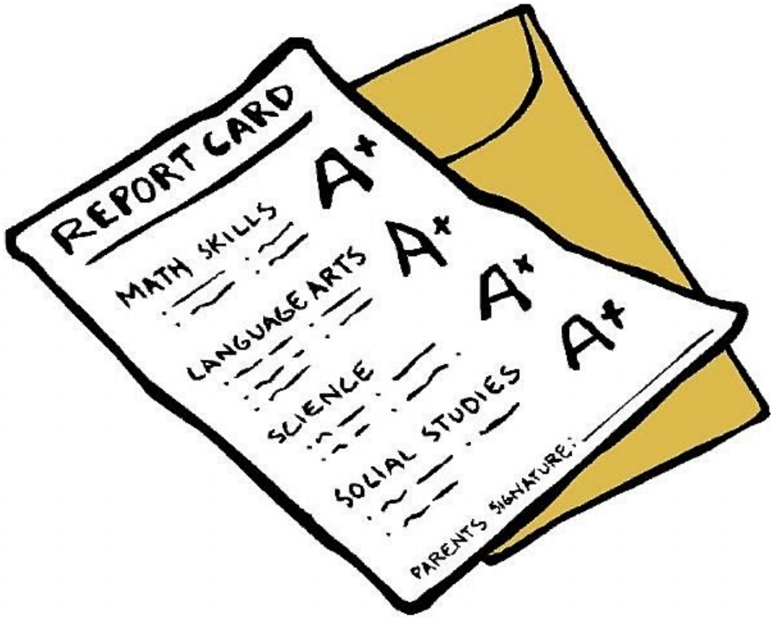
Operator	Usage	Equal to
`+=`	`c += 5`	`c = c + 5`
`-=`	`c -= 2`	`c = c – 2`
`*=`	`c *= 3`	`c = c * 2`
`/=`	`c /= 3`	`c = c / 3`
`%=`	`c %= 3`	`c = c % 3`

Programming Challenge: 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: My Grades

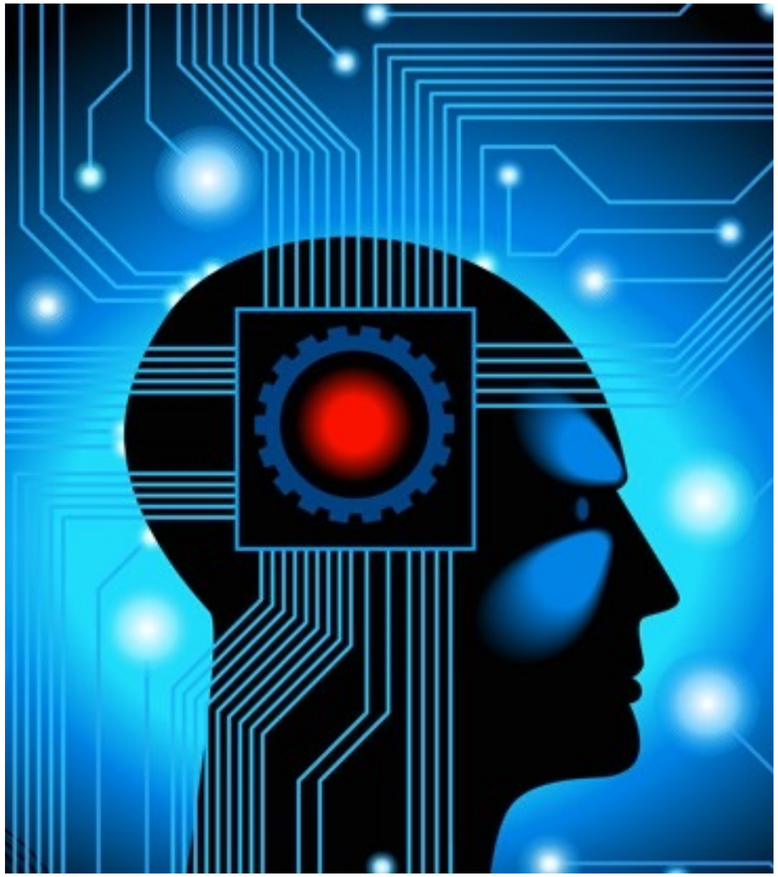
- Write a program that asks the user to enter in a test score (out of 100%)
- Allow the user to enter in as many scores as he or she wishes
- When finished, calculate the user's average score in the class



+

Programming Challenge: AI "Guess a Number" game

- Write a program that asks the user to supply a secret number between 1 and 1,000,000
- Then have the computer continually guess until they find the secret number
- Keep track of the number of attempts
- Extension: How can this be optimized?



+

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

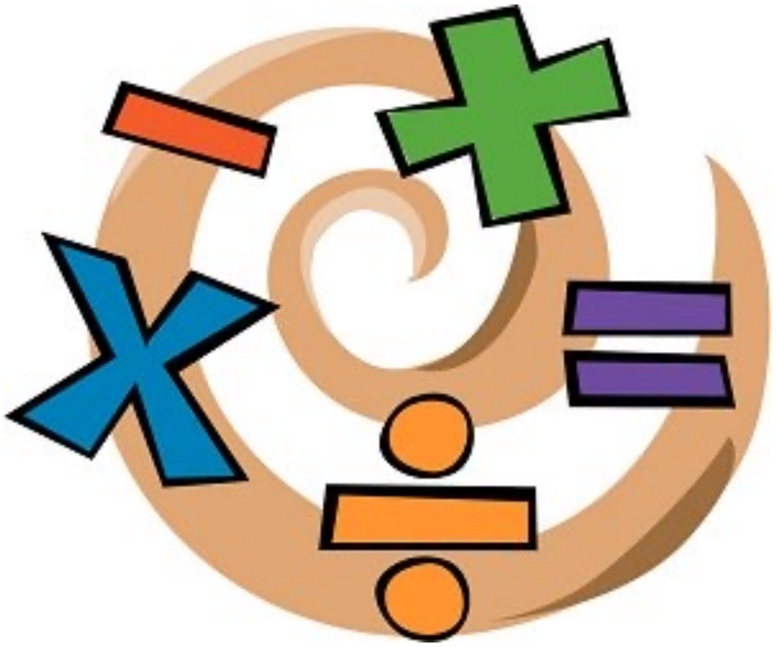
+

Programming Challenge: 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

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.



Programming Challenge: Rock, Paper, Scissors Tournament

- Write a program that lets the user play a game of Rock, Paper, Scissors against the computer
- End the game when either the player or the computer earns 3 points

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

+

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.

+

Trace the output

```
while 4 > 5:
    print("Hello")
```

Trace the output

```
while 4 > 5:
    print("Hello")
```

```
while False:
    print("Hello")
```

Both programs do the same thing!

+
Trace the output

```python
while 10 > 3:
    print("Hello")
```

Trace the output

```
while 10 > 3:
    print("Hello")
```

```
while True:
    print("Hello")
```

Both programs do the same thing!

+

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 loop 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

```
count = 0

while True:

    if count == 5:
        break

    print("I'm ready!")

    count += 1
```

+

Trace the output

Using break	Without break

Using break:

```
count = 0

while True:

    if count == 5:
        break

    print("I'm ready!")

    count += 1
```

Without break:

```
count = 0

while count < 5:
    print("I'm ready!")

    count += 1
```

Trace the Output

```
x = 0

while x < 10:

    if x >= 3:
        break

    print (x)

    x += 1
```

The “continue” command

- The “continue” command is a special Python command that causes the loop to immediately go to the next iteration.
- 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 continue command runs it will immediately stop current iteration and recheck the condition, which prevents any commands after the continue command from running

+
Trace the output

```
x = 0

while x < 10:

    x += 1

    if x % 2 == 0:
        continue

    print(x)
```

+

Programming Challenge

- Write a program that asks the user for the item name and the item price
- When the user enters the word "end" you should stop asking the user for item names and price. Then, display the following:
  - The list of items they purchased
  - Their total bill
  - The highest priced item
  - The lowest priced item