CIS 122

Throwing you for a loop
Definitively Speaking

- How could we solve the collatz problem using loops?
  - How many times do we need to apply HOTPO before we reach 1?

```
even x → x/2
odd x → 3*x+1
```
Definitively Speaking

- How could we solve the Collatz problem using loops?
  - How many times do we need to apply HOTPO before we reach 1?

- Uh oh
  - Don't know how many times we'll need to loop
How could we solve the collatz problem using loops?
- How many times do we need to apply HOTPO before we reach 1?

Uh oh
- Don't know how many times we'll need to loop

For loop are **definite** loops
- We know exactly how long they will last
- One loop for every element in our sequence

The collatz problem requires an **indefinite** loop
- We don't know how many loops it will require beforehand
While Loops

• We need a new type of loop
  ○ While loops

• While some condition is true
  ○ Keep running block of code

• Very similar to if statement
  ○ If statement runs block once if condition is true
  ○ While loop runs block repeatedly while condition is true
Anatomy of a while loop

```
x = 0
while x < 10:
    print x
    x = x + 1
```

- **Initialization**: $x = 0$
- **Loop Condition**: `while x < 10:`
- **Loop body**:
  - `print x`
  - `x = x + 1`
While Loops

- While condition is True, keep running body
  - What if condition is always true?
- Infinite loop
  - Similar to infinite recursion
  - But no limit on number of loops
- Sometimes an infinite loop is a good thing
  - IDLE shell
  - Operating systems

```
x = 0
while x >= 0:
    print x
    x = x + 1
```

```
x = 0
while True:
    print x
    x = x + 1
```
While Loops

● What if you need to break out of a loop early?
  ○ Use the break keyword
  ○ Stop running whatever loop you're in

```python
x = 0
while True:
    print(x)
    x = x + 1
    if x == 10:
        break
```
While Loops

● Avoid using break statements when you can
  ○ Tend to make code less clear
  ○ A good loop condition is far more readable

● If you use break statements, comment them well

```python
x = 0
while x < 10:
    print x
    x = x + 1
```

```python
x = 0
while True:
    print x
    x = x + 1
    if x == 10:
        break
```
While Loop Practice

- Implement \texttt{collatz(x)} using a while loop
  - How many times do we need to perform HOTPO on \( x \) before it reaches 1?

- How could we use a while loop to solve this problem?
While Loop Practice

- Implement \texttt{collatz(x)} using a while loop
  - How many times do we need to perform HOTPO on \( x \) before it reaches 1?

- How could we use a while loop to solve this problem?
  - Initialize a counter to 0
  - While \( x \) hasn't reached 1...
    - Apply HOTPO to \( x \)
    - Increment counter

\[ \begin{align*}
\text{even } x & \rightarrow x/2 \\
\text{odd } x & \rightarrow 3\times x + 1
\end{align*} \]
While Loop Practice

● Implement \texttt{collatz(x)} using a while loop
  ○ How many times do we need to perform HOTPO on \( x \) before it reaches 1?

```python
def collatz(x):
    steps = 0  # Initialize a counter to 0
    while x != 1:
        x = HOTPO(x)  # Apply HOTPO to \( x \)
        steps = steps+1  # Increment counter
    return steps
```

\[
\text{even } x \rightarrow x/2 \\
\text{odd } x \rightarrow 3 \times x + 1
\]
So many Choices

- We've seen two types of loops
  - for loops
    - Repeat some task *for* each element in a sequence
    - Definite loops
    - Good for specific tasks
  - while loops
    - Repeat some task *while* a condition is true
    - Indefinite loops
    - General purpose
So many Choices

● Which loop should I choose?

● Do have a sequence you want to iterator over?
  ○ for element in sequence

● Do you know how many times you want to loop?
  ○ for x in range(n)

● None of the above?
  ○ while <some condition>
Homework Preview

- Part 0 - Summing Things Up
- Part 1 - Circular Reasoning
- Part 2 - Password Checker
- Part 3 - Guessing Game
• Write a function `mySum(numbers)`
  ○ Takes a list of numbers
  ○ Returns their sum

• What loop should we use?

• For inspiration, look over our max function from yesterday
Part 1 - Circular Reasoning

- Turtle graphics are back!

- Write a function circle(radius)
  - Draw circle of the given radius
  - This isn't an easy task
  - But what if we approximate our circle as a polygon

- Write a function polygon(sides, sideLength)
  - Draw a polygon with the given number of sides
  - Repeatedly move forward and turn
  - What loop should we use?
Part 2 - Password Checker

● Make sure passwords are sufficiently secure
  ○ At least 8 characters long
  ○ At least 1 letter
  ○ At least 2 numbers
  ○ Don't contain 'E' or 'e' (those letters are far too common)

● Write a function passwordChecker(password)
  ○ Returns False if password fails any tests
  ○ Returns True if password passes all tests
Part 2 - Password Checker

- Write helper functions to test individual cases
  - Does this string contain a letter?
  - Does this string contain two numbers?

- Call helper functions from main password checker

- What loops should we use?
Part 2 - Password Checker

● Special string methods
  ○ dot notation

>>> 'a'.isalpha()
True

>>> 'b'.isdigit()
False

>>> myChar.isupper()
???
Part 3 - Guessing Game

- Write a function `guessingGame()`

- When called, Python should play a guessing game
  - Pick a random number
  - Ask the user to guess a number
  - If they guess wrong, give them a hint (too high, too low)
  - If they guess right, congratulate them
    - And tell them how many guesses they took

- What needs to loop?
  - And loop should we use?