Control Flow: The Collatz Conjecture

Background:

The Collatz Conjecture is a famous, unsolved problem in mathematics. We'll be exploring it today to build some skills with Python's control flow features. The idea is as follows:

- Take a positive, whole number x and
  - If it is even, divide it by two.
  - If it is odd, triple it and add one.
- The result is now a new x.
- Repeat until x = 1.

Every positive, whole number we have tried always ends up reaching one. There may be some number out there that ends up stuck in a loop and never reaches one, but so far, we have not found any instances of this. Today we'll build a python program that explores the Collatz Conjecture for various inputs.

Here's an example calculation: 10 → 5 → 16 → 8 → 4 → 2 → 1.

In words: 10 is even, so we divide it by 2 and get 5. But 5 is odd so we triple it and add one to get 16. 16 is even, so we divide by 2 to get 8. 8 is even, so we divide by 2 to get 4. Likewise, 4 is even so we divide by two and get 2. Two is even, and 2/2 = 1, so 10 satisfies the Collatz Conjecture.

Skills:

First, we'll need to learn a few new things about python syntax and Kaggle:

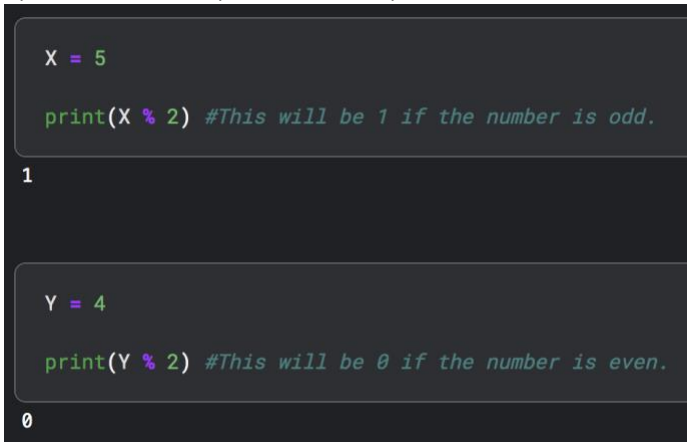
- if, elif, else conditions:

```
y = 10

if y < 10:
    print("y is less than ten.")
elif y < 100:
    print("y is less than one hundred but greater than or equal to ten.")
else:
    print("y is 100 or greater")
```

```
y is less than one hundred but greater than or equal to ten.
```

- You can check if a number is even by using the % operator if a number X % 2 == 0, then it is even, if X % 2 == 1, then it is odd:



```
X = 5

print(X % 2) #This will be 1 if the number is odd.
```

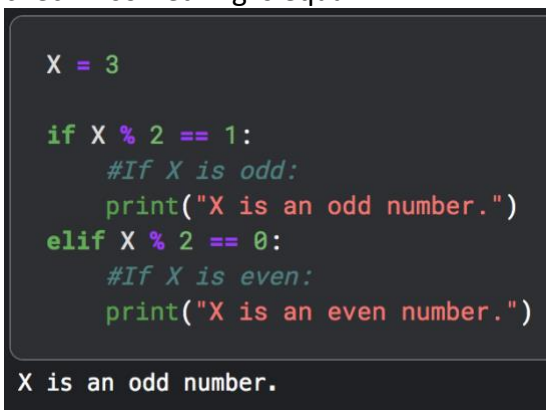
```
1
```

```
Y = 4

print(Y % 2) #This will be 0 if the number is even.
```

```
0
```

- These can be used as conditions in an if statement. Note the == syntax is used to check if something is equal:

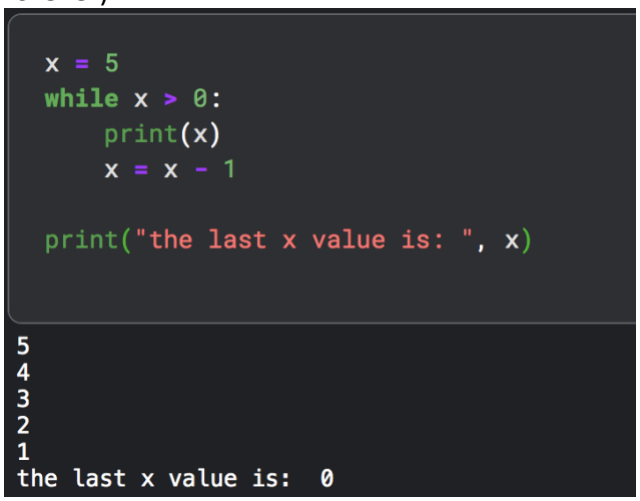


```
X = 3

if X % 2 == 1:
    #If X is odd:
    print("X is an odd number.")
elif X % 2 == 0:
    #If X is even:
    print("X is an even number.")
```

```
X is an odd number.
```

- while loop – loops *until a condition is no longer met* (be careful, it may go on forever).



```
x = 5
while x > 0:
    print(x)
    x = x - 1

print("the last x value is: ", x)
```

```
5
4
3
2
1
the last x value is:  0
```

Problems:

1. In Kaggle, go to code and create a new notebook.

2. The Collatz Conjecture is essentially a "while" loop. That is, we can tell python: while x is not equal to one, do some operations. Then, if x eventually equals one, the program stops and the number we have chosen does not break the Collatz Conjecture. Write a while loop to perform the calculations for some value x – that is: compute x//2 or 3*x+1 depending on if the number x is even or odd respectively and repeat this until x = 1. Print out each number along the way. Try it yourself first, but if you get stuck, here's a step-by-step guide:

    a. Create a variable called x and set it equal to some number.

    b. Write a while loop that will loop over and over again as long as x does not equal one (x != 1). An exclamation mark followed by the equal sign is the python syntax for 'is not equal to.'

    c. Inside of the while loop, write an if statement that checks if x is an odd number. If it is odd, set x to be your old value of x times three plus one (x = 3*x + 1). Python interprets this as three times x plus one *and then* assigns the result to x, overwriting the previous number stored in x.

    d. Still inside the while loop but outside of the if statement you just wrote, write an elif statement that checks if x is an even number. If it is even you can reassign x to be x // 2. Note we must use x // 2 for integer division so that python knows you want a positive, whole number from your calculation.

    e. After both of your if statements but still inside your while loop, print out x.

    f. …and that's it! Your while loop will then repeat over and over again until x = 1.

3. Let's make another version of the above code. Copy and paste your code from question 2 into a new cell.

4. For this new version, get rid of the print statement you wrote in step 2-e so that your program does not print out a bunch of x values. Instead let's try something different.

    a. Before your while loop, define a new variable called counter and set it equal to zero. Make sure you are still defining an x value as well.

    b. Inside your while loop, after the if statements increase the counter by one every time the while loop is entered. You can do this by writing counter = counter + 1.

    c. As a check, set x = 42. Your counter should be 8 in this case. We call this number the stopping time.

5. Try several different numbers out for your starting x value. How high of a stopping time can you get?

6. Why are while loops potentially dangerous?