Unit 3: Conditions, if and while | **Skill Builder 1: Collatz**

In this lesson, you will 'branch' out into the world of conditional programming. You will also investigate an as yet unproven conjecture.

Objectives:

- Learn the relational and logical operators
- Learn the integer operators (% and //)
- Write programs using if statements and while loops

Choices, choices, choices. Our life is one long series of decisions: Are you hungry? What to eat? Is it cold? What to wear? Are we there yet? This decision-making process in programming is handled with **if** statements and **while** loops, which both depend on conditions. Examples of **if** and **while** in action are seen in in this image.

A condition is an expression that results in the value **True** or **False**:

X > Y **A+B <= C** **Qty > 0** **5 != 3**

are all examples of conditions. *(does not equal)*

A condition includes one or more of these relational operators:

== **>** **<** **!=** **>=** **<=**

Compound conditions are made using the logical operators:

and **or** **not**

Conditions are used in **if** structures and **while** loops.

Caution: Remember to use ***==*** *when writing a condition, not* ***=****. Using the wrong symbol will result in a syntax error.* ***if x==5:****, not* ***if x=5:****. Using the* ***ctrl+=*** *menu can help.*

This lesson introduces you to these powerful programming tools.

1. **The Collatz Conjecture**

   Algorithm: *Take a positive integer: if it is even, divide it by 2,*

   otherwise multiply it by 3 and add 1.

   Repeat with the result.

   What happens to the sequence?

   Begin with a blank Python file (we called it **Collatz**).

   Input an integer to the variable **num** using the statement:

   num = int(input("Enter a positive integer: "))

   int(is found on **menu > Built-ins > Type**.

   input(is found on **menu > Built-ins > I/O**.

2. **if** statements come in three flavors: **if..**, **if..else..**, and **if..elif..else..**. They are all found on **menu > Built-ins > Control**. Note that there is no 'then' in Python.
(They are on the '**Control**' menu because these statements *control* the flow of your program.)

If.. *Use when there is no 'otherwise' action.*

if..else.. *Use when there are exactly two alternative actions for* **True** *and* **False**.
(We will use this one soon.)

if..elif..else.. *Use these when there are three or more actions to be taken based on several conditions.*
elif *is short for 'else if…' and* <u>*requires*</u> *a condition like* **if**.
You can add as many **elif**s *as your algorithm requires.*
(This one is used in the Application for this unit.)

3. Insert the **if..else** statement from **menu > Built-ins > Control**.
Press **tab** or right-arrow to move from prompt to prompt.

4. The condition (*BooleanExpr*) is…

if **num % 2 == 0**:

% is called 'mod' and is the mathematical operator (like +, - *, and /) that gives the *remainder* when the first number is divided by the second.
'mod' is short for 'modulus'.
% is found on the punctuation key (next to the letter 'G').
If the *remainder* when num is divided by 2 is zero, then the num is *even*.
Note the two equal signs!
For **==** just press the = key twice. All the relational operators are on **ctrl+=**.

5. The **if** : (when the number is even)
   block (the top one)
   becomes:

   num = num // 2

   // (two division signs) is *integer* division (no decimal and truncates to just the whole number). If you use */* you will see a decimal point even if the number is an integer. Just use two */* signs (the ÷ key).

6. The **else:** (when the number is odd)
   block is:

   num = 3 * num + 1

7. After the **if..else**: statement block, backspace to the beginning of a line (erase the indent characters) and write the **print** statement:

   print(num)

8. Running the program:
   Press **ctrl+R** to run the program. Enter a positive integer. An answer appears.
   Press **ctrl+R** again and this time enter *the last answer*.
   Repeat running the program, each time entering the previous answer.

   Next you will add a loop to the program so that the process runs repeatedly by itself instead of having to run the program over and over…

9. Place your cursor right below the **input** statement and above the **if** statement as shown in this image. (Do not type: '<<< *cursor here*')

10. On this *blank* line add the **while** statement found on

menu > Built-ins > Control.

You will see

while BooleanExpr:
block

pasted into your program.

11. Erase the entire line that says 'block'.

Select the *entire* **if** structure and **print** statement using **shift+down arrow**. Press **tab** to indent all these lines two spaces to become the **while** *block*.

12. Now write the condition by replacing *BooleanExpression*.

The Collatz Conjecture states that all sequences will eventually become 1.

As long as the number is greater than 1, continue processing; so write:

while **num > 1 :**

Be sure to leave the colon (:) at the end of this line.

13. Run the program. Enter 20 as the number. Follow the logic of the program: Odd numbers get larger and even numbers get smaller.

    20 is even → 10
    10 is even → 5
    5 is odd → 16
    16 is even → 8
    and so on…
    … and the program ends when the number reaches 1.

Note that it only took <u>one line of code</u> to create a loop!

Can you find a number that causes the program to NEVER end? Try a large number. Notice how fast the numbers fly by! When the program ends you can scroll upward through the Shell history to examine the numbers.