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 a mathematical theory that is (so far) unproven.

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:

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

Caution: Remember to use **==** *when writing a condition, not* **=**. *Using the wrong symbol will result in a syntax error. Write* ***if x==5:****, not* ***if x=5:****. Using the* **[test]** *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 **<Fns…> Type**.

   input(is found on **<Fns…> I/O**.

2. **if** statements come in three flavors: **if..**, **if.. else..**, and **if.. elif.. else..**. They are all found on **<Fns…> Ctl**. Note that there is no "then" in Python.

(They are on the **Ctl** 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** *(either do this or do that). (You 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* ***requires*** *a condition just like* **if**. *You can add as many* **elif***s as your algorithm requires.*

(This structure is used in the application for this unit.)

There *must* be a colon (:) at the end of *each* **if**, **elif** and **else**. These colons are *required* and indicate that what follows are the actions to be taken when the *condition* is **True** or **False**.

3. Insert the **if.. else** statement from **<Fns…> Ctl**.

4. The condition (*written between* ***if*** *and the colon*) 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 **<a A #>** screen or in the **catalog [2nd] [0]**.

The statement now reads: "If the *remainder* when **num** is divided by 2 is zero," which means "if **num** is *even*…"

Note the two equal signs!

For **==** just use any of the following:

- press the **[sto ->]** key twice or
- select it from **<Fns…> Ops** or

- select from **<a A #>** or
- press **[test]**

5. **if** *(the number is even)* **:**
   ♦♦ *(the "True" block)*
   is:

   ♦♦num = num // 2

   Just use two **/** signs (the **[÷]** key).

   // (two division signs) is called *floor division* (no decimal and truncates to the integer just *below* the decimal value). If you use **/** you will see a decimal point even if all the numbers are integers.

6. **else:** *(when the number is odd — no condition here)* **:**
   (the "False" block)
   is:

   ♦♦num = 3 * num + 1

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

   print(num)

8. **Running the program:**
   Select **<Run>** to run the program. Enter a positive integer. An answer appears. Remember it!

   Select **<Tools> [enter]** to run the program again and this time enter *the last answer*.

   Repeat running the program, each time entering the previous answer. Eventually.…

   But wait! Let's 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.

10. On this *blank* line add the **while** statement found on
    <Fns…> Ctl

    You will see:

    while :

    ♦♦

    pasted into your program and your cursor is blinking on the colon.

11. Indent each line below **while :** (the **if.. else** structure and the **print** statement) by:
    - Place your cursor anywhere on a line
    - Select **<Tools> Indent►**

    Blank lines and #comments do not need to be indented since Python ignores both.

    Indenting causes the statements to become the block of code inside the **while** *structure. You still need to write the* **while** *condition.*

12. Now write the *condition* after the word **while** (leave a space after **while** and do not erase the colon).

    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 :**

13. Run the program now. 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 **one line of code** 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 (**[2nd] [uparrow]**) to examine the numbers.*