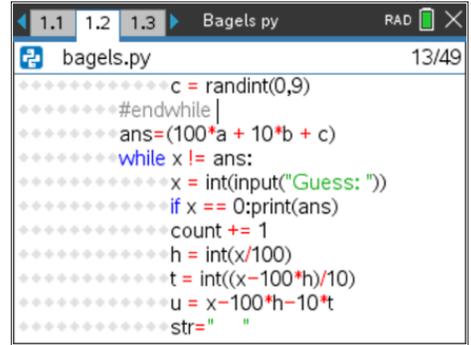


Unit 3: Conditions, if and while **Skill Builder 1: Collatz**

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 this image.



```

1.1 1.2 1.3 Bagels.py RAD 13/49
bagels.py
..... c = randint(0,9)
..... #endwhile |
..... ans=(100*a + 10*b + c)
..... while x != ans:
.....     x = int(input("Guess: "))
.....     if x == 0:print(ans)
.....     count += 1
.....     h = int(x/100)
.....     t = int((x-100*h)/10)
.....     u = x-100*h-10*t
.....     str=" "

```

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.

Teacher Tip: Tricky: Using == for equality and != for 'does not equal'. This lesson introduces two new Python arithmetic operators: % and //. **a % b** gives the integer remainder when a is divided by b **a // b** gives the integer quotient of a divided by b and is equivalent to int(a/b)

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**.



```

1.1 1.2 *U3SB1 C...atz RAD 9/28
*Collatz.py
num=int(input('Enter a positive integer: '))

```

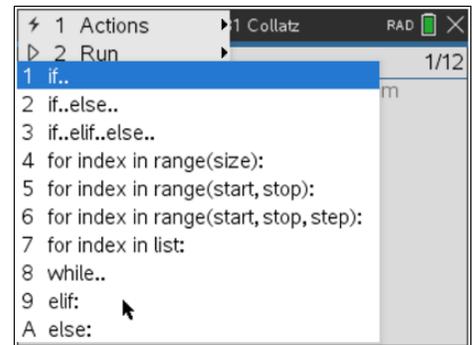
2. **if** statements come in three flavors: **if..**, **if..else..**, and **if..elif..else..**. They are all found on **menu > Built-ins > Control**. Note that there is no ‘then’ in Python.

(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.)

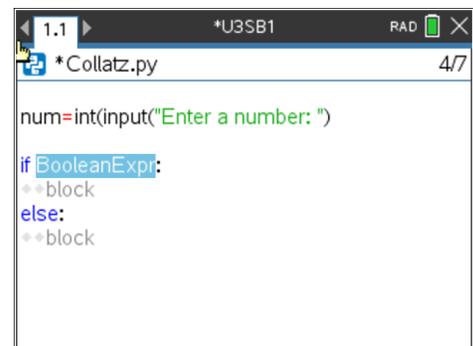
if..elif..else.. *Use these when there are three or more actions to be taken based on several conditions.*
***elif** is short for ‘else if...’ and requires a condition like **if**.*
*You can add as many **elif**s as your algorithm requires.*
(This one is used in the Application for this unit.)



Teacher Tip: Notice the colon(:) at the end of the **if**, **elif** and **else**. These are required and indicate that what follows are the actions taken when the *condition* is **True** or **False**. The *BooleanExpr* and each indented ‘block’ will be replaced with your actions.
else: does not take a condition.
else: and **elif:** are also available as ‘standalones’ on this menu but they must be used in conjunction with an **if..**

True and **False** (yes, they are Capitalized) are Python reserved words. You could write **while True:** to create an infinite loop. To break (stop) a program that it stuck in an infinite loop:
 Handheld: ON Windows: [F12] or Fn+[F12] Mac: [F5]

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



Teacher Tip: If the statement is typed in by hand, then the inline prompts (*BooleanExpr*, *block*) will not appear.
 Press **tab** to move from prompt to prompt.



10 Minutes of Code - Python

TI-NSPIRE™ CX II TECHNOLOGY

UNIT 3: SKILL BUILDER 1

TEACHER NOTES

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+=.

```
1.1 *U3SB1 RAD 9/9
*Collatz.py
num=int(input("Enter a number: "))
if num % 2 == 0:
    ++block
else:
    ++block
    ++
```

Teacher Tip: Notice that there's no 'then' in Python. Waste of space!

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).

```
1.1 1.2 *U3SB1 RAD 8/16
*Collatz.py
num=int(input("Enter a number: "))
if num % 2 == 0:
    ++num=num // 2
else:
    ++
```

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

block is:

num = 3 * num + 1

```
1.1 1.2 *U3SB1 RAD 9/18
*Collatz.py
num=int(input("Enter a number: "))
if num % 2 == 0:
    ++num=num // 2
else:
    ++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)

```
1.1 1.2 *U3SB1 RAD 10/19
*Collatz.py
num=int(input("Enter a number: "))
if num % 2 == 0:
    ++num=num // 2
else:
    ++num = 3 * num + 1
print(num)
```



10 Minutes of Code - Python

TI-NSPIRE™ CX II TECHNOLOGY

UNIT 3: SKILL BUILDER 1

TEACHER NOTES

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...

```

Python Shell 1/13
>>>#Running Collatz.py
>>>from Collatz import *
Enter a number: 5
16
>>>#Running Collatz.py
>>>from Collatz import *
Enter a number: 16
8
>>>#Running Collatz.py
>>>from Collatz import *
Enter a number: 8
  
```

Teacher Tip: What happens? The number eventually becomes 1 where the pattern will just repeat 1,4,2,1,4,2,1...

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

```

*Collatz.py 3/17
num=int(input("Enter a number: "))
<<< cursor here
if num % 2 == 0:
    num=num // 2
else:
    num = 3 * num + 1
print(num)
  
```

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.

```

*Collatz.py 3/18
num=int(input("Enter a number: "))
while BooleanExpr:
    block
if num % 2 == 0:
    num=num // 2
else:
    num = 3 * num + 1
print(num)
  
```

Teacher Tip: The 'block' is indented but the actual block is already in the program: the **if** structure and the **print** statement. The next section shows how to indent an entire section of text.

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**.

```

*Collatz.py 9/18
num=int(input("Enter a number: "))
while BooleanExpr:
    if num % 2 == 0:
        num=num // 2
    else:
        num = 3 * num + 1
    print(num)
  
```

Teacher Tip: Another option for pressing **tab** is menu > **Edit** > **Indent**. **Dedent** is the opposite of indent and its shortcut is **shift+tab**.

12. Now write the condition by replacing *BooleanExpression*.
The Collatz Conjecture states that all sequences will eventually become 1.

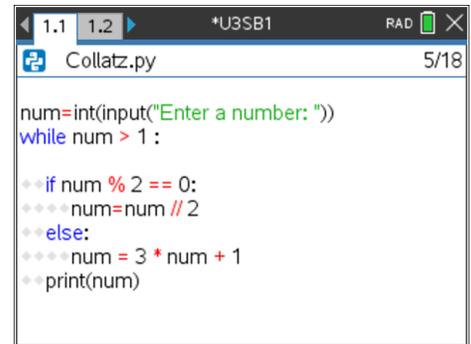
As long as the number is greater than 1, continue processing; so write:

while num > 1 :

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

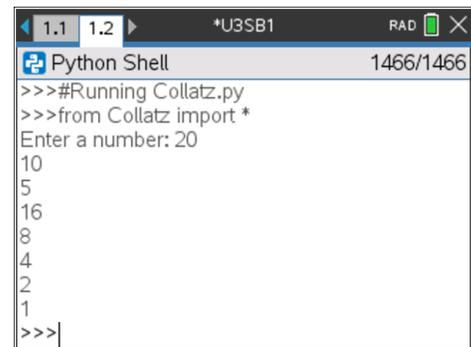
13. Run the program. Enter 20 as the number. Follow the logic of the program: Odd numbers get larger and even numbers get smaller.

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.



```

1.1 1.2 *U3SB1 RAD 5/18
Collatz.py
num=int(input("Enter a number: "))
while num > 1 :
    if num % 2 == 0:
        num=num // 2
    else:
        num = 3 * num + 1
    print(num)
    
```



```

1.1 1.2 *U3SB1 RAD 1466/1466
Python Shell
>>>#Running Collatz.py
>>>from Collatz import *
Enter a number: 20
10
5
16
8
4
2
1
>>>|
    
```

Note that it only took one line of code to create a loop!

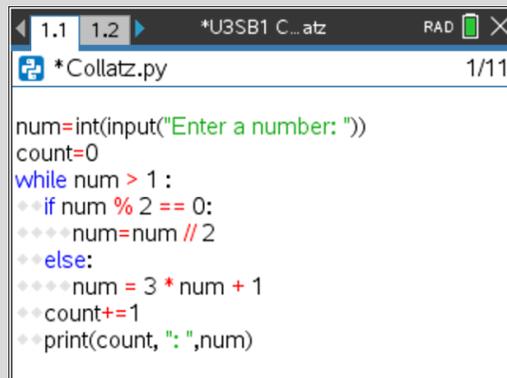
Can you find a number that causes the program to NEVER end? Try a large number. Notice how fast the numbers fly by! When the program ends you can scroll upward through the Shell history to examine the numbers.

Teacher Tip: The Shell history is plain text and is not saved with the TI-Nspire document. If you want to keep some of the Shell history, select, copy, and paste it into a Notes app. To clear the Shell history use **menu > Tools > Clear History**.

There is also a programming command to clear the Shell history (similar to a clear screen function).

The Collatz Conjecture (presented in 1937) is still unproven. Is there something special about the numbers 3, 1, and 2 in the algorithm? Try using different numbers.

One more option to add to the program is to count the number of steps to reach 1.



```

1.1 1.2 *U3SB1 C...atz RAD 1/11
*Collatz.py
num=int(input("Enter a number: "))
count=0
while num > 1 :
    if num % 2 == 0:
        num=num // 2
    else:
        num = 3 * num + 1
    count+=1
    print(count, ":", num)
    
```