Python Project:

Make a two-player Rock-Paper-Scissors game. (Hint: Ask for player plays (using input), compare them, print out a message of congratulations to the winner, and ask if the players want to start a new game)

Remember the rules:

* Rock beats scissors
* Scissors beats paper
* Paper beats rock

**While loops**

We have already discussed for loops, or loops that look sequentially (one by one) at elements in a list. There is a second type of loop that works in a slightly different way called a while loop.

The idea is simple: while a certain condition is True, keep doing something. For example:

a **=** 5

**while** (a **>** 0):

**print**(a)

a **-=** 1

The output of this code segment is:

5

4

3

2

1

A particularly useful way to use while loops is checking user input for correctness. For example:

quit **=** input('Type "enter" to quit:' )

**while** quit **!=** "enter":

quit **=** input('Type "enter" to quit:' )

The uses for this are infinite and can (and should!) be combined with conditionals to yield the most efficient results.

**Infinite loops**

An infinite loop is a loop that never stops. This means that the condition in the beginning of the while loop will always be true.

For example:

i **=** 5

**while** i **>** 0:

**print**("Inside the loop")

What will happen is the loop will print out the phrase “Inside the loop” forever and ever. If you are running your computer, you will have to “kill the program” to stop it. Each operating system has a different way of “killing a program” to get out of an infinite loop.

On Linux: in the terminal, type “CTRL-C” to kill the program that is currently running in the terminal. If you are using the IDLE Python IDE, then you must press “CTRL-D” to exit your running program. When in doubt, do a Google search before you start programming!

On Windows: type “CTRL-ALT-DEL” and open the task manager to kill the program.

On Mac: right-click on the task, and kill the program that is running forever.

If you find yourself in an infinite loop, your program will never end.

**Break statements**

A break statement stops the execution of a loop before the original condition is met. While the use of a breakstatement will often start an argument about good coding practices, sometimes it is useful.

For example:

**while** True:

usr\_command **=** input("Enter your command: ")

**if** usr\_command **==** "quit":

**break**

**else**:

**print**("You typed " **+** usr\_command)

In this case, the break statement is used to break off the “infinite while loop” that we have constructed with the while True statement.