Lesson 06 – Notes

**While Loops, Break Statements, & Continue Statements**:

Loops –

While loop:

* executes the block of code over and over again as long as the condition remains true
* last statement should be a incrementation statement
* each run through the loop = 1 iteration

Infinite Loop 🡪 press CTRL+C

Break Statement:

* immediately jumps out of current block without reevaluating the loop’s condition
* useful for when you might have several cases in which the block should terminate

Continue Statement:

* immediately jumps back to the start of the loop & reevaluates the loop’s condition

\*\*\*

1. All loops have 3 parts to them, initialization, condition and increment.
2. In many ways, Python harkens back to the ‘good ole days’ of programming when coders used a console. For example, the use of the break statement (kind of a go-to statement) is used in Python, and it exists in most languages. However, over the years, the design of programs has implemented break statements less and less, at least in the context of infinite loops. The example is a good one to demonstrate what a break statement can do – but perhaps not in a practical way. Specifically, if you have multiple break statements in a loop, the code becomes very difficult to debug. I am not saying that you should not use break statements – just be careful.
3. Even though continue statements still effect the flow of the code, they are often easier to debug than a break statement – as long as you do not use the control statement to jump around too much**. If designed properly, a continue statement will maintain the flow of the code – while a break statement often ‘changes it’**.
4. Note that in Python, **you need to convert the integer to a string;** many languages do this automatically in a print statement.
5. Note that you can increment with something other than 1.