AggieSTAAR Python Bootcamp

Tutorial 2: Conditions and loops

TEXAS A&M UNIVERSITY

To-do:

Open up Jupyter lab through Anaconda, or by typing “jupyter lab” into a terminal, and open up tutorial2_conditions&loops.ipynb.

To complete both exercises in the tutorial, you will need to have numpy and astropy installed, as well as rv2015.txt downloaded in the same folder as tutorial2_conditions&loops.ipynb.

Conditions

Conditions - Python can apply basic logical mathematical conditions:

- `==` (equals),
- `!=` (does not equal),
- `<` (less than),
- `<=` (less than or equal to),
- `>` (greater than),
- `>=` (greater than or equal to).

When you apply a condition to two variables, you will get a Boolean. Booleans take on one of two values: `True` or `False`.

if statements

if - IF a condition is found to be True, code within the if statement will execute.

Example:

```
X = 5

Y = 4

if X > Y:

    print(“X is greater than Y”).
```

Indentation here is VERY important! Use the TAB key to indent.

elif and else statements

elif - if the previous condition is found to be False, then code within the elif statement will execute.

elif is short for “else, if”.

else - if all conditions are found to be False, then code within the else statement will execute.

Think of else statements as a catch-all statement.

More logical operators…

and - 2 conditions must be satisfied to be True.

or - 1 if 2 conditions can be satisfied to be True.

in - checks if elements are in a string or a list.

Like more traditional mathematical conditions, these logical operators are usually applied within if statements.

Loops

Loops are used to repeat actions efficiently.

Print out the numbers 1 - 5.

You can probably do that within a few seconds by copying and pasting code.

Print out the numbers 1 - 10,000.

Loops can do that faster than you can copy and paste 5 lines of code.

Two different kinds of loops: while and for loops.

while loops

while loops - WHILE a condition is True, execute the code within the while loop.

```
A = 0

while A < 5:

    print(A)

    A = A + 1
```

This while loop will keep executing the indented lines until the condition A < 5 becomes False.

Make sure that the condition will eventually become False! Or else you will have an infinite loop.

for loops

for loops - FOR every element in an object, execute code.

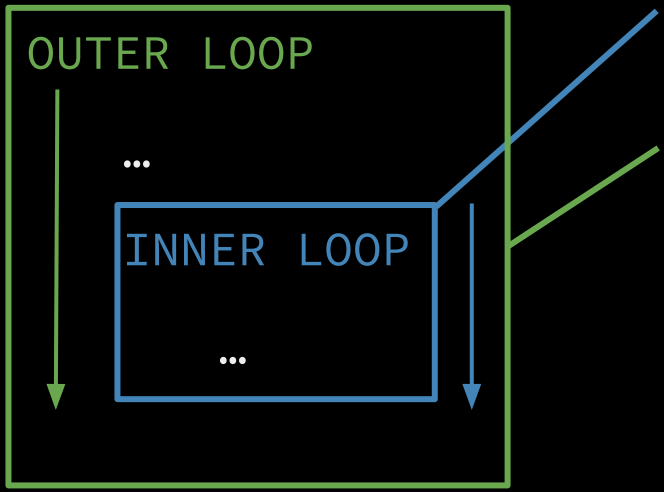
```
Y = [1, 2, 3]

for x in Y:

    print(x)

> 1, 2, 3.
```

Dummy variables: here, x is a dummy variable.

Be careful with dummy variables: they do not really ‘exist’ outside of the for loop.

Nested loops

It can get more complicated - you can put loops within loops. A visual demonstration:



Inner loop executes completely.

Inner loop continues to execute until conditions are met to break the outer loop.

Combining statements

while and for loops can be combined with if statements!

This can be very helpful to sort through data. You will get a practical example in the last exercise of this tutorial.

Example: you can quickly find all numbers that fall below a certain threshold in a list.