**Boolean Comparisons**

1. Comparison Operators

**Table 2-1:** Comparison Operators

| **Operator** | **Meaning** |
| --- | --- |
| == | Equal to |
| != | Not equal to |
| < | Less than |
| > | Greater than |
| <= | Less than or equal to |
| >= | Greater than or equal to |

1. Activate your Python Virtual Environment and open your Interactive Shell.
   1. What two commands did you use?
   2. **source ~/py3/bin/activate**
   3. **python**
2. Boolean operations evaluate to True or False. The values for true and false and represented by True and False respectively and are case sensitive.
3. Type the following:
   1. True<ENTER>
      1. True will always evaluate to True
      2. Similarly False will always evaluate to False.
   2. not True<ENTER>
      1. not negates the Boolean value
      2. What is the result of: not not True?
      3. True
   3. Equality is tested using double equal signs.
      1. 'cat' == 'cat'<ENTER>
         1. This produces True because the two strings are equal
      2. 42 == 42<ENTER>
         1. This produces True because the two numerical values are equal.
      3. 1 > 2<ENTER>
         1. This produces False because numerically 1 is not greater than 2.
   4. 1 = 1<ENTER>
      1. Explain the results.
      2. Error because you need to use ==
   5. 1 == 1
      1. Explain the results.
      2. 1 is equal to 1 so True
   6. 1 == '1'
      1. Explain the results.
      2. False because int 1 doesn’t equal str 1
   7. 1 == int('1')
      1. Explain the results.
      2. True because int() command switch str 1 to int 1 and int 1 is equal to int 1
   8. a =5  
      (1 == a) == True
      1. This evaluates to false. First the parentheses are evaluated. The numerical value 1 is tested against the value of variable "a" which is currently 5. As 1 does not equal 5 this returns False. Our Boolean comparison is now "False == True" which returns False as false does not equal true.
4. The binary Boolean operators "and" and "or" are used to perform logical comparisons of two binary values. Consult the tables below evaluations.
5. **Table 2-2:** The and Operator’s Truth Table

| **Expression** | **Evaluates to . . .** |
| --- | --- |
| True and True | True |
| True and False | False |
| False and True | False |
| False and False | False |

**Table 2-3:** The or Operator’s Truth Table

| **Expression** | **Evaluates to . . .** |
| --- | --- |
| True or True | True |
| True or False | True |
| False or True | True |
| False or False | False |

1. Given the above evaluate the following:
   1. (42 == 42) and ('cat' == 'dog')
      1. Explain the results.
      2. False because the while 42 is equal to 42, cat doesn’t equal to dog. ‘And’ needs both side to be true
   2. (42 == 42) or ('cat' == 'dog')
      1. Explain the results.
      2. True, because ‘or’ just need one side to be true. 42 is equal to 42
   3. (42 == 42) and not ('cat' == 'dog')
      1. Explain the results.
      2. **True, because the ‘not’ reverse the answer for Boolean operation so when it is false for cat to equal to dog. ‘not’ reverse it to true.**