Boolean expressions (type bool)

These notes are just a description of the rules. See the many examples posted along with them for how they are applied.

- Boolean expressions have just two possible values, True and False. These are reserved words in Python.
- You form boolean expressions by comparing two expressions (with numerical or string type) using one of the comparison operators <, <=, >, >=, ==, !=. Note that ‘==’ means ‘is equal to’, ‘!’ means ‘is not equal to’.
- Examples: Check what happens when you type things like 4<=5.2, 4==4.7, ‘cat’==’Cat’, ‘dog’<’cat’, ‘Dog’<’cat’ in the Python shell. The last might be a surprise. Also check ‘32’<32. This is illegal (by the way, if you ever run into Python 2, you’ll find that it wasn’t always illegal).
- You further build boolean expressions by combining them with the logical operators and, or and not.
- The if statement: Syntactically, an if statement has the form

```
if A:
  B
```

where A is a boolean expression and B is a sequence of statements (possibly only a single statement) indented. (In this respect it looks just like a function definition.
- When the statement is executed, the boolean expression A is evaluated. If the value is True, the sequence B is executed. Otherwise nothing happens and execution continues with the statement following the if statement.

A variant is the if...else construct;
Here C, like B, is a statement or a sequence of statements, indented. The behavior is given in the diagram below.

![Decision Tree Diagram](image)

- Often, you have to make a multi-way decision. For example, an improved version of our calendar program would print the name of the day of the week corresponding to the integer computed by the algorithm. A decision-tree diagram of the process looks like this:

![Decision Tree Diagram](image)

and continues down for four more levels. If you coded this in Python you would get:
It's not a good idea to nest statements within one another so deeply. This sort of multiway decision comes up so often that Python provides a much more readable alternative, the if..elif... statement. Here's what it looks like.

```python
if d==0:
    print('Sunday')
elif d==1:
    print('Monday')
elif d==2:
    print('Tuesday')
elif d==3:
    print('Wednesday')
elif d==4:
    print('Thursday')
elif d==5:
    print('Friday')
else:
    print('Saturday')
```