

Conditional

Conditional statements are ones that you want to execute only if some condition is true. Here's a boring example:

```
x = 5
if x < 6:
    print 'less than 6'
```

Note the form of the 'if' statement:

```
if <condition>:
    <statement>
    <statement>
```

An **else** clause says what to do if the 'if' condition is false, e.g.,

```
if x < 6:
    print 'less than 6'
else:
    print 'more or equal to 6'
```

If you have more than two alternatives, use the keyword **elif**, which is short for else-if.

```
if x < 6:
    print 'less than 6'
elif x < 10:
    print 'between 6 and 10'
else:
    print 'ten or bigger'
```

You can have as many 'elif's as you want.

Comparison and Logical Operators

Here are the comparison operators that you can use in conditional statements (and iteration statements):

```
x < 3
x > 3
```

```
x == 3
x != 3
x <= 3
x >= 3
```

Note that `==` is used for equality comparison as opposed to `=` (the assignment operator). This causes a lot of errors of course.

You can also use the logical operators `and`, `or`, and `not`, to create more complex conditions, e.g.,

```
if (x<3 and y != 7):
```

Statement Blocks

A block of code is one or more statements that are grouped together, e.g., if a condition is true, execute this block of code.

In many languages (e.g., C, C++, Java), a programmer adds curly braces around statements to define a block:

```
if <condition> {
    statement 1
    statement 2
}
```

In such languages, indentation is ignored by the compiler.

Python is different. It uses indentation to define blocks of code and doesn't use curly brackets. Tabs are used to indent. With this rule, can you figure out the value of `x` for the following snippet:

```
x=5
if x<4:
    x=x+1
x=x+1
```

What if `x` began as 3?

In-Class Assignment

Modify your code which reads in a mastermind guess to state whether each color input is a valid color or not.