If statements

If statements are used to have options when something happens. We have if statements everywhere in our lives: if the time is after 10:30 and I am hungry, then I will go eat lunch; if I get 10 points in a game before you, then I win.

In Python, we have three different keywords that allow us to do this: `if`, `elif`, and `else`. `Elif` is a contraction of `else if`.

If

```
color = raw_input("What is your favorite color? ")
color = color.lower()
print "Your favorite color is", color

if color == "blue":
  print color, "is my favorite color too!"
print "Thank you for telling me."
```

Here is what the program does:

The program asks the user `"What is your favorite color?"` and takes in the value the user gave it and saves it to the variable `color`. It then uses the function `.lower()` to change all uppercase letters to lowercase letters in the variable `color`. It will print the user's color in the line `"Your favorite color is,"` `color`, to show the user the program saved it. Then the program compares the string to the value `"blue"` in the `if` statement on line 5. If the strings are equal, then it will do the lines that are indented. There is only one line, which is `print color, "is my favorite color too!"`. After this line, the program will print `"Thank you for telling me."`

In this example, the `if` statement only works if the value after the word `if` have the value `True`. If you look at the section about comparisons, we see that the symbol (operator) `==` checks to see if two values are equal. If they are equal, then it returns the value `True`. If they are not equal, then it returns the value `False`.

If you enter any variation of the string `"blue"`, it will compare it to the string `"blue"` on line 5 and return `True` to the `if` statement.

Let's pretend I ran this program, here are the two possible outputs:

```
What is your favorite color? RED

Your favorite color is red

Thank you for telling me.
```

And

```
What is your favorite color? BlUe

Your favorite color is blue

blue is my favorite color too!

Thank you for telling me.
```

Notice that no matter the `if` statement being `True` or `False`, our last line of output is still the same.

if and elif

We can use `if` statements one after each other to check different cases:

```
number = int(raw_input("Give me a number: "))

if number % 2 != 0:
  print "Your number is not even."
if number % 3 == 0:
  print "Your number is divisible by 3."
```

This works, however we can use something else to get more out of it. Here is a different way:

```
number = int(raw_input("Give me a number: "))

if number % 2 !=0:
  print "Your number is not even."
elif number % 3 == 0:
  print "Your number is even and divisible by 3."
```

For the first program, if your number is odd, it will print `Your number is not even.`, and if it is a multiple of 3 (e.g. 3,6,9...) the program will print out `Your number is divisible by 3`. If I say my number is `3`, the program will print out both statements about my number. If I say my number is '`4`, the program will print out neither of the statements. If I say my number is `5`, the program will print out just the first statement.

For the second program, we are using an `elif` statement. They are used with with `if` statements and the knowledge gained from them. Let's pretend I started the second program and give it the number `3`. the program will do the `3 % 2 != 0` which simplifies to `1 != 0`, which is `True`. It will then print the line `Your number is not even.` and the program will end. Now let's pretend I started the second program and gave it the number `6`. The program will do the first comparison, `6 % 2 != 0`, and return `False`. It then moves to the next comparison,`number % 3 ==0`, and returns `True` and prints out `Your number is even and divisible by 3.`

Think for a moment about how these two programs are different and their different outcomes.

Doing elif without elif

I have written a different programs to give the same results as the second program above without using the `elif` statement to explain what is happening:

“‘py number = int(raw_input(“Give me a number:”))

if number % 2 !=0: print “Your number is not even.” if number % 3 == 0 and number % 2 ==0: print “Your number is even and divisible by 3.” “‘

Else

This statement is meant to be a “catch all”. For example, I could say that colors are either my favorite color, blue, or not my favorite color, every other color.

```
color = raw_input("What is your favorite color? ")
color = color.lower()
if color == "blue":
  print "Blue is my favorite color too!"
else:
  print "That's cool!"
print "Thanks for telling me."
```

If your color is `blue`, it will print `Blue is my favorite color too!`. If it is not blue, we will print `That's cool!`. Knowing that the first statement, `color == "blue"`, is `False`, we do what is in the `else` statement.

Putting everything together

You can put all of these together if you want:

```
town = raw_input("Where were you born?")
town = town.lower()
if (town=="toledo"):
  print "I went to MV for school there, did you?"
elif (town=="findlay"):
  print "I was born there too!"
elif (town =="bowling green" or town == "bg"):
  print "I know people there."
else:
  print "I don't know where that is."
```

Notice that this has all three (`if`, `elif`, and `else`) and two `elif`'s. The idea here is that the `else` will only be printed if all of the other if-statement conditions are `False`.

Nesting if statements

Read this program:

```
print "I'll tell you if the year you enter is a leap year."
year = int(raw_input("Enter a year: "))
print
print year,
if(year %4==0):
  if (year %100 == 0):
    if (year % 400 ==0):
      print "is a leap year."
    else:
      print "is not a leap year."
  else:
    print "is a leap year."
else:
  print "is not a leap year."
```

Take your time. Try to understand it before moving on.

When you have another if statement inside an if statement like above, it means that it will be testing those if the one outside of it is true. E.a. the statement `year % 100 ==0` will only be tested if `year %4 ==0` is `True`. This can be very useful for cases like finding a year is a leap year.

However, we could rearrange the tests to get a more efficient program that gives us the same answer:

```
print "I'll tell you if the year you enter is a leap year."
year = int(raw_input("Enter a year: "))
print
print year,
if year % 400 ==0:
  print "is a leap year"
elif year % 100 ==0:
  print "is not a leap year"
elif year % 4 == 0:
  print "is a leap year"
else:
  print "is not a leap year"
```

This program gives us the same result as the previous one. In my eyes, neither program is better. The first one is better because it reads better to understand the rules of a leap year, this is known as readability. The second one is better because it is less complex, this is known as Occam's Razor. In programming, the less lines the better. Which approach depends on which you want to go with.