**Python Conditionals**

Conditional statements in Python are very similar to those in JavaScript. Remember indentation is important here because Python doesn’t use **{}** to denote code blocks. The colon **:** denotes the end of the expression being checked whereas in JavaScript brackets **()** were used to encapsulate expressions.

**if num > 100:**

**print( "num is greater than 100" )**

**elif num <= 100 and num > 0: # *elif*** *is used and not* ***else if***

**print( "num is 100 or less but greater than 0" )**

**else: #** *do not use expressions for* ***else***

**print( "num is equal to or less than 0" )**

Remember when explicitly checking to see if something is **true** or **false** you need to use the correct boolean **True** with the capital T or **False** with the capital F. Also, triple equals **===** does not exist in Python. Use double equals **==** to test. Double equals **==** is strict and checks to see if the datatypes are the same.

**if hasPassword == True:**

**print( "Person has Password" )**

**Nested Conditional Statements**

Just like other programming languages, **if else** statements can be contained inside other **if else** statements forming nested **if else** statements.

**country = "Canada"**

**province = "BC"**

**if  country ==  "Canada":  
   if  province ==  "BC" or province == "Alberta":  
     print( "Western Canada" )  
   else:  
     print( "Not West Canada" )**

**Useful Operators for Comparisons**

Python has the same **logical operators** as JavaScript however the syntax is different.

**And Or**

**and or**

**colour = "green"**

**if colour == "green" or colour == "orange": # True**

**if colour == "green" and 10 < 0: # False**

Python has the same **comparison operators** as JavaScript except for triple equals **===** which is not needed in the language.

**Equal Not Equal Less than or Equal**

**== != <=**

Python has **membership operators** which can tell whether a string or collection variable contains a certain element.

**In Not In**

**in not in**

**if "m" in "James": # True**

**if 10 not in [5, 10, 15]: # False**

**Truthy and Falsy**

Conditional statements such as **if** and loops that use **while** check to see if an expression equates to either **True** or **False**. However you may have noticed that these expressions can run under less obvious circumstances. This behaviour isn’t just connected to Python as both examples below, even the JavaScript one, will run and print “TRUE”

**if 5: if ( 5 ) {**

**print( “TRUE” ) console.log( “TRUE” )**

**}**

This is because individual expressions and values can equate to true or false. Values that equate to False are called Falsy and values that equate to True are called Truthy. Below you will see how zero values or empty strings and collection variables are considered falsy when checked while any number that isn’t zero nor a non-empty string or collection equates to **True**.

**Falsy Values Truthy Values**

**0 == False 1 == True**

**[] == False [ 2 ] == True**

**() == False ( 3 ) == True**

**{} == False { a: "1" } == True**

**"" == False "A" == True**

**False == False True == True**

**Shorthand way**

Thanks to the way that Python reads like English, there are ways to write **if** statements and **if else** statements in a single line. In the below example we can assign a value to a variable depending on a condition. This is quite similar to JavaScript’s ternary operator. The string variable **maturity** is assigned the value of “Adult” if **age** is greater than 18 but if it isn’t, **maturity** is assigned the value of “Child”.

**age = 30**

**maturity = "Adult" if age > 18 else "Child"**

The above code might seem confusing until you try to read it out loud. “*Maturity is ‘Adult’ if age is greater than 18 else it is ‘Child’*”

Here the code is again except this time I’ve highlighted the conditional part. Note that this shorthand one liner does not use the colon **:**

**maturity = "Adult" if age > 18 else "Child"**