Lesson Plan

Warmup: Variable Swap

```
a=1
b=2
#Find a way to exchange the values of a and b.
#You cannot just assign a=2 and b=1, your code should work for any values of a and b
#Hint: You can use a 3rd new variable
```

Review booleans, strings AND input/output:

Input

Python uses the input() function to receive information from the user.

Let's look at the following program:

```
print("What's your age?")
age = input()
```

When you run this program, Python will print out "What's your age?" in the terminal. It then waits for the user to enter in something from the keyboard. If we type in '16' and press enter, the program finishes running. Now, if we print out the age variable, it outputs '16' - what we entered in.

```
print(age) # Will print out whatever you typed in earlier
```

So whatever you type in gets stored in the variable.

We can simplify our code by putting the statement we want to print out first inside the parentheses of the input function.

```
age=input("What's your age?") #Does the same thing as before
print(age)
```

Let's say you have a birthday. We try to increase the age variable by one:

```
age=input("What's your age?") #We input in '16'.
print(age)
age = age + 1 #Birthday!
print(age) #Should print out '17'.
```

However, now it gives us an error!

The issue is that Python automatically thinks that what you entered in is a text string. It doesn't make sense to add a number and a string together, hence the error.

In order to fix this, we can convert the string input into a number using the int() function.

```
age=int(input("What's your age?")) #We input in '16'.
#The int function should convert the string into an integer
age = age + 1 #Birthday! #This will work now
print(age) #Prints out '17'.
```

You can also convert inputs into decimal numbers.

Exercise: #Write a program that takes in two numbers as input from the user, multiplies them and prints it out.

String Concatenation and Variables

In Python, we can join strings together using the plus ('+') symbol.

```
name = "Richard"
myString = "My name is" + name
print(myString)
```

If we run this, it outputs "My name isRichard", with no space in between the two strings. Python simply pushes the two strings together.

If we don't want this, simply put a space after "My name is". (“My name is ”)

```
myString = "My name is " + name #note the space in the string
```

Often, we will want to print out some variable along with some text string.

```
temperature = 80
print() #We want to print out "The temperature is 80".
```

We could do it like this:

```
print("The temperature is " + temperature)
```

The temperature variable is an integer, but Python converts it into a string and then joins those two strings together.

An alternative way:

```
print("The temperature is", temperature)
```

When we use the comma, Python automatically inserts a space in between the two strings. Exercise:

#Introduce the user - ask them their name and age, then output an introduction

#Example input: Flora, 16

#Example output: Your name is Flora and you are 16 years old.

Comparison Operators Let's take two numbers a and b.

```
a=1
b=2
```

Many times in programming we'll check if two numbers (and other kinds of data) are equal to each other. How we do this is with the "==" operator.

```
print(a==b)
```

When we run this, it outputs "False". What Python has done is check if a equals b. If their values are the same, then it is true that they are equal. Otherwise, it is

false. So Python evaluates and stores "a==b" as a boolean.

Note: a common mistake is to confuse "==" with '='. '=' is used for assigning values to variables, '==' is used for comparison.

We can do other comparisons as well.

```
greaterthan = (a>b)
lessthan = (a<b)
geq = (a>=b) #checks if a is greater than or equal to b.
leq =(a<=b) #checks if a is less than or equal to b.
```

In addition, we can do operations on booleans themselves.

```
raining = True
sunday = True
rainySunday = raining and sunday
```

What the 'and' operator does is check if both of the booleans are true. Only then will rainySunday also be true.

```
pizza = True
sushi = False
goodLunch = pizza or sushi
```

What the 'or' operator does is check if either of the booleans are true. If so, goodLunch will be true. Only if both of the booleans are false will it return false.

Exercise:

```
#Largest Number
a=
b=
c=
isCtheLargest = #
print(isCtheLargest)
#Write an expression using comparisons that will output true if c is the largest out of a, b, c and false otherwise.
```