1. Goals
    a. Learn what programming is through Python
    b. Learn about the tools a programmer needs to write code
    c. Learn the basic programming constructs that apply across all languages
    d. Combine all these constructs to build a calculator

2. What is programming?
    a. Programming is the process communicating with computers.
    b. But computers don't understand english, so we talk with them using what's called a programming language. In this class we will use a language called Python.

3. Why Python
    a. You are going to hear A LOT of debate over languages and frameworks and what tools are better than others. Should I pick Python or Ruby or Django or Ruby on Rails. Ignore all of it.
    b. Learn one thing and become a master of that domain.
        i. Despite the fact that many employers screen based on experience with a specific language, the concepts you will learn in python apply across the board to almost any language and are highly transferrable.
    c. So why did I pick python?
        i. It reads like English, making it very easy for beginners.
        ii. It's general purpose and can used to solve problems related from systems administration to scientific data analysis to web programming.
        iii. It's popular. 5th of the TIOBE index
            1. I don't mean to contradict my previous statement
            2. Although your skills are transferrable, there is always a learning curve to starting with a new language. Why not start with one lots of people use?
    d. There are actually two popular editions of python: python 2 and python 3. We are going to use python 3 since it is becoming the standard for new projects in the python community.

4. Cloud9 service http://c9.io
    a. Make sure to sign up.
    b. Create a workspace, give it a name, and select custom.

5. Setup
    a. Every programmer needs three tools: a text editor, a terminal, and a programming language.
    b. Cloud 9 covers all of these for us. It comes with Python preinstalled and a built-in text editor and terminal. This way we can avoid installing of these separately for each of you and ensure that everyone is using the same tools because it will make it far easier to follow along.

6. Terminal
    a. The Terminal is a special program that comes with your computer in which you can issue commands to your computer. You can tell it to do things like create files and folders, run programs, and many other things.
    b. While a graphical interfaces on your computer, such as Finder in Mac and File Explorer in Windows, can accomplish many of the same tasks, the Terminal affords programmers more flexibility and power. It can execute commands that graphical interfaces cannot. In particular, we can use the terminal to run Python on a particular file.

7. Python Interpreter
    a. Python is actually a computer program, and this program is installed on the cloud9 machine. We call this program the interpreter, and it's job is to read in Python code and according to the grammar of Python
    b. We can run the interpreter in two ways. Let's look at the first
    c. Type python3 into your terminal, and you will be greeted with ">>>"
    d. This is what is called a REPL, a Read, Evaluate, Print, Loop.
    e. The interpreter wants us to enter python code. it will then read it in, evaluate it, print out a result, and then ask for more input. We can execute code in this manner line by line.
    f. This is not how Python programs are normally run, but you will see me use it frequently today for demonstrations.

8. A Built-In Calculator Demo
    a. If you haven't already, type python3 into your terminal.
        i. If you omit the 3, you will actually run python 2
    b. 1 + 1
    c. 7 * 947
    d. 27 / 3
    e. 5 ** 3 (It even handles exponentiation)
    f. 5 1 (Our first error! This does not conform to Python syntax. Notice that Python does not spit out a response)

9. Our First Program Demo
    a. Click File > New File
    b. Click File > Save File As > helloworld.py
    c. print ("hello world")
    d. python3 helloworld.py
    e. If you get an error like SyntaxError: EOL while scanning string literal
        i. you probably forgot a quotation mark or parenthesis.
    f. We can actually run this program at the interpreter as well

10. Modify program to print multiple statements… Notice that there is a distinct order of operations.
    i. print ("Let’s print some more complicated statements")
    ii. print ("How many friends do you have?", 25 + 30)
    iii. print ("How many are girls?", 25 + 30 / 2)
    iv. print ("That doesn’t look right...")
    v. print ((25 + 30) / 2)
    vi. print ("better")
    vii. print ("Is it true that (25 + 30) / 2 < 25 + 30 /2?")
    viii. print ((25 + 30) / 2 < 25 + 30 / 2)
    ix. print ("I don't believe you...")
    x. print ((25 + 30) / 2, " < ", 25 + 30 / 2)
    xi. print ("Perhaps you're right, Python")
    xii.
    xiii. print ("Is 5 greater?", 5 > -2)
    xiv. print ("Is 5 greater than or equal?", 5 >= -2)
    xv. print ("Is 5 less than or equal?", 5 <= -2)
    xvi.
    xvii. print ("is 5 equal to -2", 5 == -2)
    xviii. print ("but 5 is not equal to -2", 5 != -2)
    xix. print ("but it is equal to 5?", 5 == 5)
11. Strings Demo
    a. So we’ve seen both numerical values, called integers, and strings values. What is the difference between the two? In the simplest terms, strings are surrounded by quotes and integers are not, plus integers are always numbers.
    b. “cat” # Notice that strings are always surrounded by quotes
    c. ‘fish’ # Single quotes work too as long as you’re consistent on each side
    d. fish # But we get an error if we forget quotes
    e. “cat” + “fish” # We can even add strings (called concatenation)
    f. “cat” - “fish” # But we can’t subtract them like we can with integers
    g. 5 - 5 # These are integers because there are no quotes
    h. “5” - “5” # Why does this error out?

    i. Why do some operators work on integers but not strings? This demonstrates that programming languages all have rules. In some languages, it might be illegal to multiply a string by a number. In others, it might be illegal to add two strings. There are many many arbitrary things about every language, and it will be your job to learn these peculiarities.
    j. The important part to understand is that python treats strings and integers differently when interacting with operators.
    k. Generally we use strings when we are looking to make something human-readable, such as when we print data, and we use integers for calculations.

l. What happens if we try execute an operator over a string and an integer at the same time?
m. “cat” + 6   # It turns out it’s simply a rule that you cannot add strings and integers
n. In general, in Python these rules will be very intuitive. Ask yourself if there is an obvious result for a specific operation. If there is, Python likely works that way; if not, it’s probably not allowed.

o. It turns out we can convert integers to strings using the “str” command
p. “cat” + str(6)
q. str(6) + str(6)   # Notice that the answer is not 12, but 66. The strings were concatenated, just cat “cat” and “fish”

r. We can also convert strings to integers
s. int(“23”) + int(“23”)
t. “23” + “23”
u. int (“23”) + int(“23”) == 23 + 23
v. int(“23”) + int(“23”) != “23” + “23”

w. But we can’t just convert any string… It needs to actually be a number
x. int(“cat”)   # Will crash
y. int(“fish”)   # Will crash

12. Variables Demo:
    a. Variables will allow us to store an integer, a string, a calculation, or really anything (this is not an exaggeration).
    b. We are basically naming the integer or string and defining a new way to reference it (by it’s name).
    c. x = 42   # This is not algebra. This means that we are setting x to be 42.
    d. print (x)   # Python remembered that x is equal to 42!
    e. y = 2
    f. print (x / y)
    g. x = 10   # Let’s change the value of x
    h. print (x * y)   # x was overwritten!

    i. Setting variables has nothing to do with algebra.
    j. a + 1 = 4 # This will never work
    k. a = 1 + 4 # But this will
    l. a = 5   # The variable must always be alone on the lefthand side of the equals sign.

    m. my_name = “nick”
    n. your_name = “student”

o. print(my_name + “ “ + your_name) # Variables behave just like the strings they contain! Notice that the strings were properly added (also called concatenation)

p. We can convert the type of a variable just like you’d expect
q. my_number = 55
r. my_name = “nick”
s. print (my_name + my_number)   # Fails as expected
t. my_number = str(my_number)
u. print (my_name + my_number)   # Python remembered that my_number is a string!

13. Reading Input
    a. A program is not much use if it does the same thing every time. Let’s make a program that changes based on user input. We will use the input command for this.

    b. number_to_square = input("Please enter a number to square: ")
    c. print (number_to_square * number_to_square)

    d. TypeError: can't multiply sequence by non-int of type 'str'
    e. Remember when I said strings are used for human readable tasks? That includes asking for input. This means we are trying to use a string in a calculation, but I said integers are for calculations.
    f. So let’s turn our string into an integer.

    g. number_to_square = input("Please enter a number to square: ")
    h. number_to_square = int(number_to_square)
    i. print (number_to_square * number_to_square)

14. Make Our Own Calculator Demo
    a. Program a calculator (first_calculator.py)
    b. show bug with string operand (cannot fix without try)
    c. show bug with operator that is not specified >>> lead into elif and else introduction (if_elif_else.py)
    d. show bug with divide by 0 (fix_divide_by_zero.py)
    e. Show bug with null operator and operands. Explain truthyness (fix_null_inputs.py)
    f. Show necessity of AND statement (actually_fix_null_inputs.py)
    g. Explain functions and add all option (functions.py)
    h. Explain looping (looping.py)
    i. Explain lists (store_history.py)