Add your name here: [Dennis Bazan]

Background Information----------

What is Python?

Python is an interpreted, general-purpose, high-level programming language.

In the world of programming, there are **high-level**, **mid-level**, and **low-level languages**. What makes a language higher than the other?

> **Abstraction...** I like to think of abstraction as how "presentable" the code is. Is it possible for a person with no programming experience to read code in this programming language and be able to interpret it with no guidance? If the answer is yes (or almost yes), this language is very abstract. Another way to think of abstraction is in terms of how much work does the program does for you. High-level languages can often accomplish a task in less commands than a low-level one.

> **Processing time...** You would think a high-level language takes less time to process, right? Unfortunately, the abstraction present in high-level languages demands more processing time. Basically, it takes the computer more time to understand what a high-level language is saying versus a low-level language. This is because low-level languages work more on the "computer's terms" than high-level ones do.

Abstraction Diagram

Compilers

For a computer to process a high-level language, the code written in the high-level language must be translated so that the computer will understand it. Like stated earlier, high-level languages abstract to the degree where the language is readable to us but not necessarily a computer.

Therefore, to process a high-level language, its code must go through **interpreters** or **compilers**.

> An interpreter reads the code *line by line*. At each line, it translates and performs the command indicated by the contents of the line.

> A compiler reads the whole code and creates a translated version called an **executable**, which is just the code you wrote but translated so that the computer can understand it.

Python uses an interpreter.

Now for the Fun Stuff! ----------

Data Types and Variables

There are several basic data types in python: integers, floats, booleans, and strings. We use these data types to store pieces of data. For instance, the number of dogs you own, a phone number, the temperature outside, etc...

```
dogs            = 2
phone_number    = "815386xxxx" # Yes, that is part of my phone number.
temperature     = 95.4
did_I_eat_lunch = True
```

- **Integers (aka int) store numeric values**, so a phone number wouldn't work here because your phone number is more like an address and doesn't really posses a significant numerical value. However, something like the amount of amount
- **Floats store numerical values as a decimal**. Keep this in mind when choosing your data type. Integers do not represent whatever comes after the decimal. Floats do.
- **Booleans (aka bool) store true or false values; yes or no; 1 or 0**. In python, to assign a boolean a value of true, you must type it as 'True'. Capital T and all. Same for False. These are used in if-statements, which you will learn soon.
- **Strings hold text and characters**. String use double quotes. "**String**" is a string, but **string** is not a string. Examples of strings include an address, a color of a fruit, the name of a city, etc...

The way we keep track of data is by assigning data to a variable. We do this by first creating a variable and then assigning it to a piece of data.

> First, to create a variable we need to create a variable name. Coding etiquette states this name has to make sense with the data we are storing the variable in. So if we wanted to store my dog's name, a variable called 'catName' wouldn't make sense now would it? And a variable name like 'x' falls a bit short. 'dogName' is perfect. Below is the syntax for creating an assigning a variable:

```
dogName           = "Mojo"     # Note how strings use the double quotes
catName           = "Sir Gato" # Yes, a space will show up in a string. Even if the string is
isItBESTSummerYet = True       # Remember: capitals T or F for booleans!
GPA               = 3.6        # Is this an int?
daysInWeek        = 7          # No, but this is an int.
```

Expressions and Operators

What if you want to write an equation? Or what if you want to add two strings together to form one big string (called concatenation)? What if you want to tally up a vote? **Operators are how you do this.**

Operators take values and 'operate on' them. The plus sign '+' is an operator which adds two values.

```
# Numerical operators: works with numbers and produces a number
sumOfTwoNumbers  = 3 + 4
diffOfTwoNumbers = 3 - 4  # Yes, python can handle negative numbers!
prodOfTwoNumbers = 3 * 4
quotOfTwoNumbers = 3 / 4
```

```
# Boolean operators: works with just about any data type and produces a Boolean
andOfTwoVals = True and False
orOfTwoVals  = True or False
notOfOneVal  = not True
```

If you have never worked with the AND, OR, and NOT operators before, here is a truth table which explains how they are evaluated: [Truth Tables for AND, OR, and NOT]

```
# Comparison operators: works with numbers and produces a Boolean
isGreaterThan     = 4 > 3
isGreaterThanOrEq = 4 >= 3
isLessThan        = 4 < 3
isLessThanOrEq    = 4 <= 3
isEqualTo         = 4 == 3        # if they are equal, evaluates to True
isNotEqualTo      = 4 != 3        # if they aren't equal, evaluates to True
```

Important Notes----------

Make sure to run the code cells below to see what they do! You can do that by pressing the play button

- If you haven't figured out yet, the "print()" function prints out whatever you provide it. Typically we place a variable name inside the parentheses but you can also just straightup put a number or expression.

```
# Example of print statements
var = "hi!"
print(4)
print(True and False)
print("Hello, World!")
print(var)
```

```
4
False
Hello, World!
hi!
```

- We typically do not use numbers 1, 2, 3, etc... in variables names and we also don't use special characters like '!' or '*'.
- You can use variables in an expression, as long as they have been pre-defined. For instance...

```
x = 44
y = x / 11
print(y)
```

```
4.0
```

- In the above, note how 'x' was defined and assigned before I created 'y'. I then used 'x' when assigning 'y' to a value. It looks like 'y' evaluates to 4 (which is 44/11 or x/11).
- You can add two strings together using the '+' operator. This is called concatenation. Example below...

```
stringOne = "Hello,"
stringTwo = " World!"
stringSum = stringOne + stringTwo
print(stringSum)
```

```
Hello, World!
```

- DO NOT THINK OF THE '=' AS SETTING THE VARIABLE *EQUAL* TO A PIECE OF DATA. Instead, think of the '=' as *assigning* the piece of data to a variable name. This is important to remember because....
- You can give a variable a new value! A variable can be constantly changing in its value! Below, see how 'example' is changing to be equal to 14, then 11, then 12.

```
example = 14
example = 11
example = example + 1
print(example)
```

```
12
```

Practice Problems----------

1. **Create some variables!** Create at least one variable for each data type we just went over. Make sure they follow coding etiquette rules.
2. **Create a short code.** Create a short code which takes the lengths and widths of two rectangles and determines if one rectangle's area is greater or equal than the other's. However you choose to do this is fine. Use print to print: the area of Rectangle One, the area of Rectangle Two, and the result of the comparison.
3. **Make a variable that takes the square of another variable.** Print the result.

```
# Write your solutions below
```

```
daysInYear = 365
money      = 10.60
isItJuly   = True
birthplace = "Lima"
```

```
lengthOfRectangle1 = 10
widthOfRectangle1  = 5
lengthOfRectangle2 = 11
widthOfRectangle2  = 6
areaOfRectangle1   =  lengthOfRectangle1 * widthOfRectangle1
areaOfRectangle2   =  lengthOfRectangle2 * widthOfRectangle2
isGreaterThanOrEq  =   areaOfRectangle2 >= areaOfRectangle1
print(areaOfRectangle1)
print(areaOfRectangle2)
print(isGreaterThanOrEq)
```

```
50
66
True
```

```
x = 5
y = x*x
print(y)
```

```
25
```