# Class 2: Inputs and Variables

Date: 01/27/2019

## Warm-Up

Let’s create a new program where we’ll keep our class warm-ups.

1. At the top-left of your screen, click on the Raspberry Pi icon and hover over Programming. From there, select Python 3 (IDLE).
2. This will open the Python 3 **terminal**, called “Python 3.5.3 Shell.” In this window, click on File>New File. This will open up a new **text editor** window called “Untitled.”
3. Save this text editor document to your desktop. On the window called “Untitled,” click on File>Save As… and **double-click** on “Desktop.”
4. Name your file “WarmUp” then click Save.

Now, as your warm-up, create a program that produces the following output:

```
She sells seashells by the seashore.
```

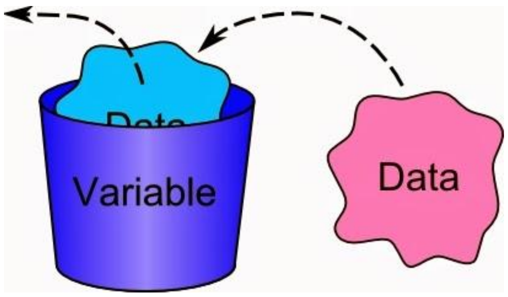
## How to Open Up Your Adventure Project

Back in the terminal (the “Python 3.5.3 Shell”):

1. Click on File>Open…
2. Double-click on Desktop
3. Double-click on MyAdventure.py

## Variables

*****Even if you already did this exercise for homework, please read through this section again!*****



One very important feature in computer programming is the variable. We can think of a **variable** as a sort of "container" where a computer can hold a number, word, phrase, or other piece of important information. Let's say I'd like to create and use a variable in my program called `cool_gal` (you can name your variable *anything you want!* But, it's recommended that you name it something you can easily remember).

Try this:

1. In the text editor, at the very top of your program, write the following code:

```
cool_gal = "Bella"
```

2. This tells the computer to get the container called "cool_gal," and put the word "Bella" in it. On the next line, to print the value of the variable to the terminal, you can then write the following code:

```
print(cool_gal + " is the coolest.")
```

3. Notice that this print statement looks a little different than the others that we've seen before. However, it works much the same: first, the computer will print the thing *inside* the container for `cool_gal`, then simply print the phrase "` is the coolest`" on that same line. Run the program in the terminal, and you'll now see:

```
Bella is the coolest.
```

4. Try going back into your program and changing the value of the variable to your own name. And, if you're not a girl, try changing the name of the variable to something like `cool_dude`, `neatkid`, `thisismy_name`, or anything you like! Remember to change the name of the variable both in the first line as well as in the `print()` parentheses.

*Here are a few important things to keep in mind when working with variables:*

- When we program, we only use variables *when we need to represent any sort of information that could change*. So, for example, we might use variables to represent a video game character's name, the video game character's health, or the list of items that the video game character is carrying (because all of these things *could change!*), but we would NOT use variables to represent something like the description of a room, or intro to our story—those things will always stay the same.
- Additionally, you should keep in mind that when using the **`print()`** statement to print the value that is held inside a variable, *you do NOT use quotations (" ")*.

## Inputs

Think you have a pretty solid idea of how a variable works? Awesome! For the next half of the class, we'll take a look at how to allow a person who is using your program to change the value of a variable. To start, let's just try typing out the following code:

```
print("Adventurer, what's your name?")

cool_gal = input()

print("Hi, " + cool_gal + "!")
```

Now we'll try to understand what is really happening here. Clearly, the first thing that the computer does is simply print out the sentence, "**`Adventurer, what's your name?`**" However, next, we start to use a variable. The program creates a variable (again, just a "container") called **`cool_gal`**. Unlike last time where we set **`cool_gal = "Bella"`**, this time we will do something slightly different. In programming, the **`input()`** function tells the computer to stop what it's doing, and wait for the person running the program to type in a number, word, or phrase. Then, the computer takes that number, word, or phrase and *stores that input inside the variable*. At the end, I used one last print statement to make the computer say "**`Hi, `**" then tell me what value was stored in the variable **`cool_gal`**. Let's try running it yourself:

```
Adventurer, what's your name?

Bella

Hi, Bella!
```

Notice that the text in **BLUE** is stuff that the computer prints out to you, while the stuff in **BLACK** is what you type back in to the computer, using the terminal (the “Python 3.5.3 Shell” window). Of course, you can type in whatever you want instead of “`Bella`”. Very cool!

## Arithmetic

Did you know that you can use your Raspberry Pi as a calculator? It’s true! Try creating a variable, then put some kind of number/math expression into the variable “container.”

```
i_wanna_add = 4 + 3

print(i_wanna_add)
```

Run it, and you’ll get:

```
7
```

This works for addition (+), subtraction (-), multiplication (*) and division(/) !

## Homework + Extra Learning

For homework, your goal is to, in *your* project, have the computer ask the user for some kind of information using the `input()` function, store that information by using a variable, and then recite that information back to them. Be creative! You can have the program ask the user for anything you want. For example, in my project I made the computer ask the user for their favorite food:

```
print(“Please tell me your favorite food:”)

favFood = input()

print(“On your adventures, you find a plate of ” + favFood
+ “ on the ground. You pick it up and eat it.”)
```

And when you run the program, you get:

```
Please tell me your favorite food:

chocolate

On your adventures, you find a plate of chocolate on the
ground. You pick it up and eat it.
```

## Next-Week Snapshot: If/Then Relationships

*Don't worry if you don't have the time to get to this step; we'll cover it next week!*

Next week, we're going to learn how to use If/Then Relationships that allow a computer program to make decisions. In programming, **decisions** are actions that the computer will do if certain conditions are met (and of course, you get to choose what these conditions are!). Let's just start out by typing the following code:

```
print("How are you today?")
myMood = input()
if ("good" in myMood):
    print("How nice!")
```

Now, we'll take some time to think through what exactly this means. In the first line, the computer prints the phrase, "`How are you today?`" On the line after that, it creates a variable "container" called `myMood`, waits for the person using the computer to type in some kind of input, and then stores that input back in the `myMood` variable. Next, in the `if` statement, the computer checks to see if the phrase "`good`" can be found in the variable `myMood`, and *IF* this is true, *THEN* the program prints "`How nice!`" (that's why we call it an IF/THEN relationship). Easy as that!

So, if we were to run it:

```
How are you today?
good
How nice!
```

However, if we respond to the computer saying something other than "good," you'll find that this will happen:

```
How are you today?
bad
```

...And, because the computer can't find the phrase "good" in the variable `myMood`, the computer won't execute the "THEN" part at all, and the program will simply end right there.