Python Chapter 2: Input, Processing, and Output

This guide has been written to reinforce your learning with the actual chapter lessons and not to serve as alternative way of learning python. I have tried my best to explain and provide explains, but if you find anything that needs to be corrected, please information me.

- Maruthi Basava

BEFORE YOU START WRITING CODE, PLEASE BE AWARE THAT WHITESPACES MATTER IN PYTHON. LEAVING STRAY SPACES IN VARIOUS AREAS COULD CRASH YOUR PROGRAM. PLEASE MAKE SURE THAT THE SYNTAX IS CORRECT.

Displaying Screen Output:

Python has many **functions** which are pieces of pre-written code that help programmers save time by not re-inventing the wheel.

One of the most commonly used function in python is the **print** function because it displays the value in the terminal.

Print function:

Correlates to Assignments (Program 2-1 to 2-3)

To use the print function, you need to type print and then a **string** (basically something that stores letters and numbers inside of a single or a double quotations.) after it.

For example:

Python 2.X *Syntax*: **print** “fish are friends not food!”

Result In the terminal: fish are friends not food!

Keep in mind that python is **case-sensitive**, that means that **print** is **not** the same as **Print**.
All python functions are lowercase and multiple words are separated by an underscore ‘_’. This rule is also followed by **variables** as well.

For example:

DO : this_is_the_right_way()
DON'T : ThisIsNotTheRightWay()

Variables:

Correlates to Assignments (Program 2-4 to 2-6)

Variables in python are basically the same thing as variables that you would use in your math class, they just store things whether it be strings or numbers for later use.

For example:

x = 10
y = 12
dog_breed = 'Husky'
current_president = 'Donald J. Trump'
your_username = 'eatfast123'

As you can see we made a
variable **x** and set it to 10,
variable **y** and set it to 12,
variable **dog_breed** and set it to a string 'Husky',
variable **current_president** and set it to a string 'Donald J. Trump',
variable **your_username** and set it to a string 'eatfast123' that has both letters and numbers.

Now in python, it's literally the same thing!

Just make a variable name and use the assignment operator (a fancy word for saying the equal sign "=") to set it to a value of a string or a number.

For example:

my_name = 'Maruthi Basava'
my_age = 17

my_name is the variable name,
= is the assignment operator
'Maruthi Basava' is the string that we are setting the variable to.

my_age is the variable name,
= is the assignment operator

17 is the number or **integer** (a number that cannot be a decimal) that we are setting the variable to,

Cool Tip!: you can name your variables anything as long it has a significant meaning.

Using the print function and variables

Here I will show you how to use variables and the print function together:

Python:

```
my_favorite_food = 'Potbelly Sandwiches'
print my_favorite_food
```

Result in the terminal:

Potbelly Sandwiches

Here, we made a variable **my_favorite_food**, then set it to a string 'Potbelly Sandwiches', then called the **print** function to display the value of **my_favorite_food** in the terminal.

One more thing:

The print function can also display numbers too!

Python:

```
super_secret_code = 12345
print super_secret_code
```

Result:

12345

Easy right?
Now let's get a little more advanced!

What if you want to display multiple variables in a single print function?
We can do that by following this syntax:

Python:

```
full_name = 'Emma Watson'
age = 21
occupation = 'actor'

print "Hello I am" + full_name + ", I am  " + str(age) + "years old, and I have been an " + occupation + "all my life."
```

Result:

```
Hello I am Emma Watson, I'm 21 years old, and I have been an actor all my life.
```

Don't get scared if you don't understand what is happening above, I will break it down for you.

First we started out by creating three variables:

```
full_name = 'Emma Watson'
age = 21
occupation = 'actor'
```

Easy!

Then we called the print statement

```
print "Hello I am " + full_name + ", I am " + str(age) + "years old, and I have been an " + occupation + " all my life."
```

Basically we are just adding the variables to the string in the print function.

If you look at it closely, when are adding **age** to the string, we used the **str** function to convert it from a number to a string. Since python cannot allow numbers to be added to a string, we must convert it in order to use. So by doing str(age), we are converting a number to a string.

Also this line of code is super long, how about we break it down by separating it among different lines of code?

We can do just that by placing the entire line into a parenthesis and pressing enter just like this

```
print ("Hello I am "
      + full_name + ", I am "
      + str(age) + "years old, and I have been an "
      + occupation + " all my life.")
```

See how cleaner it looks in contrast to this?

```
print "Hello I am " + full_name + ", I am " + str(age) + "years old, and I have been an " + occupation + " all my life."
```

Yikes!

Getting Input:

Since we learned how to store stuff in things called variables and print them, why not learn how to get information from the terminal.

We can do just that by using the **raw_input** function in Python 2.x.

Take a look at the code below:

```
your_age = raw_input("What is your name? ")
print your_age
```

Here is the break down:

We made a variable your_age equal to a function raw_input that takes in a string “What is your name? ” as a parameter. Then your_age variable is printed onto the terminal.

What this does is that the function raw_input(string) when executed, displays the string in between the parenthesis (also known a parameter) on the terminal and allows you the user to enter in text that will be collected then stored into the variable. Then we just print the variable to see what we have entered.

When this code is executed:

```
your_age = raw_input("What is your name? ")
print your_age
```

This is what happens in the terminal:

```
What is your name?
```

We are shown a gray box next to our string, this means that the terminal is expecting something from us.

When I type in my age and press [ENTER]:

```
What is your name? 17
17
```

It nicely prints out the value that I have just entered.

Ok, now let's try what we have learned from the previous lessons,

Lets try to print a sentence by only collecting your name, age, and favorite food.

We can do that with the following code:

```
your_name = raw_input("What is your name? ")
your_age = raw_input("What is your age? ")
your_favorite_food = raw_input("What is your favorite food? ")

print ("Hi, I'm "
    + your_name + ", I'm "
    + your_age + " years old and my favorite food is "
    + your_favorite_food)
```

Don't be intimidated by it, we are just creating multiple variables that store multiple inputs from the terminal then printing it, it's nothing new.

When this code runs we get the following:

```
What is your name?
```

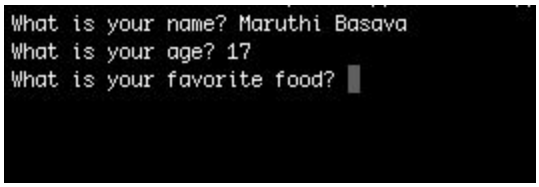
See how it only shows us only one question rather than the three we asked it to?

Python asks the question first, then collects it from the user to store it in a variable, then asks the next question.

So if we were to enter in a name and press [ENTER],

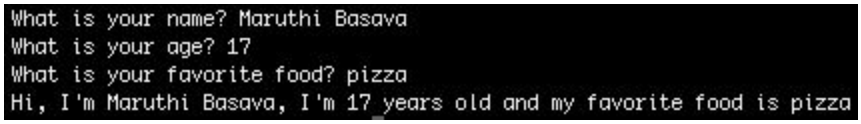
It proceeds to ask the next question,

and if were to enter in the age,



It asks us the next question.

Once we enter in information for all the questions,



The program prints everything that we just entered.

Take a look at the code again:

```
your_name = raw_input("What is your name? ")
your_age = raw_input("What is your age? ")
your_favorite_food = raw_input("What is your favorite food? ")

print ("Hi, I'm "
    + your_name + ", I'm "
    + your_age + " years old and my favorite food is "
    + your_favorite_food)
```

Do you see how the terminal asked us for input in an order?

First it asked us “What is your name?”

Second “What is your age?”

Last “What is your favorite food?”

Remember this next time you code in python.

What if you want to make a program that adds two numbers together?

We can do that with this code here:

```
num_a = int(raw_input("Enter the first number. "))
num_b = int(raw_input("Enter the second number. "))
num_c = num_a + num_b

print ("When you add "
    + str(num_a) + " and "
    + str(num_b) + ", you get "
    + str(num_c))
```

If you take a closer look, you will see the only new thing here is the int function.
But is it really that new to you?
Remember when I introduced you to the str function, where it converts a number into a string?
It’s literally the same thing, but this time you are converting a string to an integer (a number that is not a decimal).

```
num_a = int(raw_input("Enter the first number. "))
num_b = int(raw_input("Enter the second number. "))
num_c = num_a + num_b
```

We need to convert the string from the input to an integer because we need to be able to perform mathematical operations such as adding, subtracting, dividing, and multiplying.

Math in python:

```
addition = 10 + 20
subtraction = 10 - 20
division =  10 / 20
multiplication =  10 * 20
```

```
#also very important

remainder = 10 % 20 # gets the remainder when 10/20
exponent = 10 ** 20 # raises 10 to the power of 20, 10^20
```

```
print ("When you add "
    + str(num_a) + " and "
    + str(num_b) + ", you get "
    + str(num_c))
```

Once we have finished adding our inputs, we then print it, but when printing it, we must convert that integer into a string again, and we can do this by using the str function.

```
num_a = int(raw_input("Enter the first number. "))
num_b = int(raw_input("Enter the second number. "))
num_c = num_a + num_b

print ("When you add "
    + str(num_a) + " and "
    + str(num_b) + ", you get "
    + str(num_c))
```

When we run this code and enter in all the information, we get the following:

```
Enter the first number. 10
Enter the second number. 20
When you add 10 and 20 , you get 30
```

Cool.