Module 1: Count the number of words

String:

A *string* is simply a list of characters in order. A *character* is anything you can type on the keyboard in one keystroke, like a letter, a number, or a backslash. For example, "hello" is a string. It is five characters long — h, e, l, l, o. Strings can also have spaces: "hello world" contains 11 characters, including the space between "hello" and "world".

There are no limits to the number of characters you can have in a string — you can have anywhere from one to a million or more. You can even have a string that has 0 characters, which is usually called "the empty string."

There are three ways you can declare a string in Python: single quotes ('), double quotes ("), and triple quotes ("""). In all cases, you start and end the string with your chosen string declaration. For example:

```
print ('I am a single quoted string')
I am a single quoted string
print ("I am a double quoted string")
I am a double quoted string
print ("""I am a triple quoted string""")
I am a triple quoted string
```

You can use quotation marks within strings by placing a backslash directly before them, so that Python knows you want to include the quotation marks in the string, instead of ending the string there. Placing a backslash directly before another symbol like this is known as *escaping* the symbol. Note that if you want to put a backslash into the string, you also have to escape the backslash, to tell Python that you want to include the backslash, rather than using it as an escape character.

```
print ("So I said, \"You don't know me! You'll never understand me!\"")
So I said, "You don't know me! You'll never understand me!"
print ('So I said, "You don\'t know me! You\'ll never understand me!"')
So I said, "You don't know me! You'll never understand me!"
print ("This will result in only three backslashes: \\ \\ \\")
This will result in only three backslashes: \ \ \
print ("""The double quotation mark (") is used to indicate direct quotations.""")
The double quotation mark (") is used to indicate direct quotations.
```

As you can see from the above examples, only the specific character used to quote the string needs to be escaped. This makes for more readable code.

To see how to use strings, let's go back for a moment to an old, familiar program:

```
print("Hello, world!")
Hello, world!
```

Strings and Variables

Now that you've learned about variables and strings separately, lets see how they work together.

Variables can store much more than just numbers. You can also use them to store strings!

Here's Example:

```
question = "What did you have for lunch?"
   ↑              ↑

 Variable       value
print (question)
```

In this program, we are creating a variable called question, and storing the string "What did you have for lunch?" in it. Then, we just tell Python to print out whatever is inside the question variable.

Notice that when we tell Python to print out question, there are **no quotation marks** around the word question: this is to signify that we are using a variable, instead of a string. If we put in quotation marks around question, Python would treat it as a string, and simply print out question instead of What did you have for lunch?.

Let's try something different. Sure, it's all fine and dandy to ask the user what they had for lunch, but it doesn't make much difference if they can't respond! Let's edit this program so that the user can type in what they ate.

```
question = "What did you have for lunch?"
print (question)
answer = raw_input()

print ("You had " + answer + "! That sounds delicious!")
```

To ask the user to write something, we used a function called *raw_input()*, which waits until the user writes something and presses enter, and then returns what the user wrote. Don't forget the parentheses! Even though there's nothing inside of them, they're still important, and Python will give you an error if you don't put them in.

You can also use a different function called input(), which works in nearly the same way. We will learn the differences between these two functions later.

What is a word?

A word is a string without a whitespace or tab or newline. i.e., words are separated by whitespace, tab or new line. For example, “hello world” is a string, which has two words “hello” and “world”.

Basic operations

String Concatenation:

Look at that! You've been using strings since the beginning! You can also add two strings together using the + operator: this is called *concatenating* them.

Example:

```
print ("Hello, " + "world!")
```

```
Hello, world!
```

Notice that there is a space at the end of the first string. If you don't put that in, the two words will run together, and you'll end up with Hello,world!

String Multiplication:

The * operation repeates the string n times.
Example:

```
print ("bouncy, " * n)
```

```
 bouncy, bouncy, bouncy, bouncy, bouncy, bouncy, bouncy, bouncy, bouncy, bouncy,
```

If you want to find out how long a string is, we use the len() function, which simply takes a string and counts the number of characters in it. (len stands for "length.") Just put the string that you want to find the length of, inside the parentheses of the function.

For example:

```
print (len("Hello, world!"))
```

```
13
```

Len():

We can use the **len()** function to calculate the length of the string in characters.

Example:

```
var = 'eagle'

print var, "has", len(var), "characters"

O/P: eagle has 5 characters
```

Int(), float(), str():

We use a built-in **int()** function to convert a string to integer. And there is also a built-in **str()** function to convert a number to a string. And we use the **float()** function to convert a string to a floating point number.

Example:

```
print int("12") + 12
print "There are " + str(22) + " oranges."
print float('22.33') + 22.55
```

Split() :

Is a function splits given string into words.

Example:

sentence =’It is raining cats and dogs’
splitwords = sentence.split()

here sentence splits into words where space encounters. Variable splitwords is a list contains all words.

See here how it looks
print words

['It', 'is', 'raining', 'cats', 'and', 'dogs']

Exercise:

1. Count the number of words in a given string.

Input:

“This is my first program in python”

Output:

7

Note: Explore all string functions