Hello, World and Variables

“Hello, World!”

The most basic program in any language (Python included) is often considered to be the “Hello, world!” statement. As it’s name would suggest, the program simply returns the phrase “Hello, World!”. In Python, this is done with the following line of code:

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

Which will return the following:

```
Hello, World!
```

`print()` will return the text encased in quotation marks in the parentheses. The item in quotation marks is referred to as a **string**, which is just a sequence of written characters. For example, “Hello” is a string of “H”, “e”, “l”, “l”, and “o”. But, **`print()`** can output numbers as well:

```
print(10)
10
```

We can also print out two strings “added” together:

```
print("spam" + "eggs")

spameggs
```

It may seem like it would be easier to put **`“spameggs”`** as the string to print, and in this example it would be, but once we start working with variables, it will become very useful.

This is one of the primary examples of program output. It is incredibly valuable for the vast majority of programs.

`Try it yourself`: print out your name.

Variables

Another valuable tool for programming is the **variable**. A variable is a representation of a number, string, or other piece of data. The representation is a letter or word that is **assigned** the given data with the **=** operator:

```
x = 10
y = "Hello, World!"
```

Variables can be used in place of the data for the same effect:

```
print(x)

10

print(y)

Hello, World!
```

Some variables do not contain a string or a number, but instead contain either **`True`** or **`False`**. These are known as **boolean values**:

```
x = True
print(x)

True
```

Booleans are particularly useful as **Flags** which can be used with **Conditionals** and **Loops**, which will be covered in detail later.

There are also rules for variables in Python:

- The name must start with a letter or an underscore:
  - Both **`hello`** and **`_world`** are valid names for a variable
  - **`10hello`** is not, since it starts with the number 1
- After that, the name may contain letters, numbers and underscores, but only letters, numbers and underscores:
  - **`h3llo_w0rld`** is a valid name
  - **`Hello-world`** is not, since " - " is not a legal character!
- Variable names are case sensitive:
  - **`helloworld`** and **`HelloWorld`** will be treated as two different variables
- Variables are assigned left to right:
  - **`x = 10`** works
  - **`10 = x`** will give you an error
- Variables can be reassigned at any time with the **`=`** operator:
  - If you say **`x = 5`** and then **`x = 10`**, **`x`** will be considered to be 10 until you reassign it
- Some words can't be used as variable names, since Python uses them for other purposes.

- As a general rule, if the word changes color, such as `print`, don't use it as a variable name.

There are also some common conventions for variable names in most programming languages:

- **Constants**, which are variables that never change value while the program is running, are usually written in all caps: GRAVITY = -9.8.
- Other variables are usually written in **camelcase** or with underscores. Camelcase uses all lowercase letters except when there would be a space, we skip the space, and capitalize the next letter: Player one score would be stored in playerOneScore or player_one_score.

These conventions make it easier for programmers to read each other's code, but they are not rules of the language. You can break them any time you want, and your code will still run; it is just a good idea to follow them to make your code readable for yourself and others.

Variables can also be assigned as answers to equations:

```
x = 1+1
print(x)

2
```

While addition, subtraction work the same way that you are likely already familiar with, there are some other different operands that it you should be familiar with:

- To divide, we use 18/6 rather than 18÷6 or 6⌐18

```
print(18/6)

3.0
```

- Multiplication works the same way in Python as you are familiar with, but Python uses the * symbol.

```
print(5*4)

20
```

- Exponents use ** ie. 2**3 = 2^3 = 8.

```
print(2**3)
```

```
8
```

- The Modulus operator, also known simply as mod, is represented as x % y. Modulus is basically a remainder operator. It gives the remainder of x / y, while dropping the quotient.
    - 10 % 5 = 0, since 10/5 = 2 R0
    - 11 % 5 = 1, since 11/4 = 2 R1

```
print(10%5)
```

```
0
```

```
print(11%5)
```

```
1
```

- Addition and subtraction work as you would expect: 2+3 = 5, 4-7 = -3

```
print(2+3)
```

```
5
```

```
print(4-7)
```

```
-3
```

- Order of operations works too (good old PEMDAS), but it is a good idea to use parentheses to make things explicit; 2+3*4 is 14, but it is easier to read if you write it as 2+(3*4)

```
print(2+3*4)
```

```
print(2+(3*4))
```

```
14
```

`Try it yourself`: use variables and math to convert 98.6 degrees Fahrenheit to Celsius. The equation is C = $(F - 32) \cdot \frac{5}{9}$.

Exercises

1. Consider the following code snippets and give the results for each:

```
>>> x = 5 % 3
>>> x = x + 5
>>> x = x - 6
>>> x = x**2
>>> print(x)
```

```
>>> x = 7
>>> y = 9
>>> x = y % x
>>> y = y + x
>>> z = y / x
>>> print(z)
```

2. Use variables to store each of your names (first, middle, and last) and print them out in the following orders:
    a. First, middle, last
    b. Last, first, middle
    c. Middle, first, last
3. Print the equations for the lines created by each pair of coordinates (it may be easiest to write a program and change the variable values):
    a. (1, 4), (2, 6)
    b. (5, 10), (3, 12)
    c. (34.5, 65.2), (81.6, 19.1)