Text

Variables

Section 21.7

Completed

< back to module

What are Variables?

That last section wasn't so interesting. That's because you can't really do anything cool in any programming language without using variables. A variable is a way of storing data in programs.

Think of a variable like a box with a label on it. Let's say your box has a label that says "Box A". Then you put a whole bunch of socks in the box. If someone asks you for some socks, you can just say "Oh, you'll want to look in Box A". But if one day you decide to remove all the socks and store some shoes in that box instead, you would still refer to that box as "Box A". The contents may have changed but the label or the way of referring to the box remains the same.

Assigning values to a variable

In most programming languages, to assign a value to a variable we use the equals (=) operator. Python follows this standard, so in Python assigning a message to our variable looks like this:

```
user_text = "Agent Q was here."
print(user_text)
```

At any time later in my program, I can change the value of the `user_text` variable again like so:

```
user_text = "Agent Q was here."
print(user_text)

user_text = "Have you seen Agent J lately?"
print(user_text)
```

The above code will create the below output.

```
Agent Q was here.
Have you seen Agent J lately?
```

Naming Variables

You can name a variable almost anything you want, but there are a few guidelines you should follow to prevent errors, and to make your program easier for humans to read and understand. After all, programming languages are for people, not for computers, so we should always be thinking about the people trying to understand our code when we create variables.

- Variable names can contain only letters, numbers, and underscores.
- You can start a variable name with a letter or an underscore, but not with a number. So `variable_1` and `_variable1` are both ok, but `1_variable` won't work.
- Variable names are case sensitive: `AgentQ` and `agentq` are two different variables to a computer.
- You shouldn't use Python's built-in keywords or functions as variable names, such as `print` or `break`.
- Keep your variable names short but descriptive. A variable called `agent_Js_purple_cowboy_boots` is probably a bit long in most contexts, but likewise a variable called `a` doesn't tell us much about what the variable's purpose is. Something short but also descriptive like `agent_boots` is much better.

Variable Types

Every variable has it's own "type" that decides what kind of data it can store. Some of the most common data types are:

- **string:** A series of characters (basically, text). Note that these are always surrounded by quotes. If it isn't in quotes, it isn't a string. (In Python, it doesn't matter if you use single or double quotes, but in some programming languages it matters! You'll see more of this later when we talk about C.)
- **integer:** A whole number (could be positive or negative), eg: -2, -1, 0, 1, 2, 3...
- **float:** A number with a decimal point, eg: 3.14, 56.99998
- **boolean:** A True or False value. In some programming languages, booleans can be represented as 1 (True) and 0 (False).

Python is a dynamically typed programming language. This means you don't have to specify what type a variable is when you create it: it will shift its type depending on the context of the program. Other programming languages, like C, are statically typed. Statically typed languages require you to tell the program what type of data you want to store in the variable when you create it.

Here it an example of storing an integer variable in Python:

```
solved_cases = 42
```

In Python, if you're not sure what type your variable is at any given moment in your program, you can always check using the `type()` method.

```
variable_1 = '42'
variable_2 = 42
variable_3 = 4.2
variable_4 = True

print(type(variable_1))
print(type(variable_2))
print(type(variable_3))
print(type(variable_4))
```

The above should output:

```
<type 'str'>
<type 'int'>
<type 'float'>
<type 'bool'>
```

Notice the capital "T" in "True" for the boolean we have assigned as `variable_4`? If you don't use an uppercase "T", the program will error because in Python boolean values are case sensitive. You must use "True" or "False", not "true" or "false".

Completed

< back to module