Introduction to Variables: Variable Assignment

Introduction

> "There are only two hard things in Computer Science: cache invalidation and naming things."

> -- Phil Karlton

> "...But ordinary language is all right."

> -- Ludwig Wittgenstein

Objectives

You will be able to:

- Assign and declare a Python variable

Declaring and Assigning Variables

In this lesson, we'll learn how to use variables to assign names to data. For example, the name `"art vandelay"` .

In [1]:
```
"Art Vandelay"
```

Out[1]:
```
'Art Vandelay'
```

Now months later, if we see that string in some code, we may be confused as to what it is, and with even more data, this only becomes more difficult.

So, let's use variables to indicate what each of the following strings mean.

In [2]:
```
email = "art.vandelay@vandelay.com"
```

> **Note:** For this, and all of the subsequent code in gray boxes, you should press shift + enter to ensure that the code executes. If you do not do so with the line above for example, then when we reference `email` in the lines that follow, Jupyter will throw an error indicating that the variable is undefined. So, it is not enough to just type the correct code, we need to run shift + enter on our gray boxes to run this code.

In programming terms, we say that we just declared a variable, `email`, and assigned it to the string, `"art.vandelay@vandelay.com"`. To do so, we'll follow the procedure below:

```
variable = data
```

Now that we have assigned a variable `email` to a string, we just type the word `email` to see the string again.

In [3]:
```
email
```

Out[3]:
```
'art.vandelay@vandelay.com'
```

> *remember to press shift + enter on the gray box above to see the value of our variable, `email`.*

Now let's try this with the website:

In [4]:
```
website = "vandelay.com"
website
```

Out[4]:
```
'vandelay.com'
```

Note that if you introduce a new variable, (declare it), but do not also assign it in the same line, Python will raise an error.

In [5]:
```
full_name
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call
 last)
<ipython-input-5-e1e889fa3b78> in <module>()
----> 1 full_name

NameError: name 'full_name' is not defined
```

That error tells us that `full_name` is not defined. We just fix this by declaring `full_name`, storing data inside that variable and returning that variable in the same line.

In [6]:
```
full_name = 'Art Vandelay'
full_name
```

Out[6]:
```
'Art Vandelay'
```

So this is assigning and reading a variable. And when we want to see some information again, we can easily find out.

In [7]:
```
email
```

Out[7]:
```
'art.vandelay@vandelay.com'
```

Declaring variables without assignment

We have seen that we can have data without assigning it to variables.

In [8]:
```
"Unassigned data"
```

Out[8]:
```
'Unassigned data'
```

Sometimes we wish to declare a variable without assigning it to data. In Python, that's a little tricky to do. As we just saw with `name`, declaring variables without assignment throws an error. Thankfully, Python has a special type for us that represents nothing at all.

In [9]:
```
None
```

In [10]:
```
type(None)
```

Out[10]:
```
NoneType
```

None is a data type in Python that represents nothing. So, if we do not know the type of a variable and want to have the data to the variable be assigned later, we can assign that variable to `None`.

In [11]:
```
address = None
```

Notice that `address` is now assigned, but it is assigned to `None`.

In [12]:
```
address
```

Note: *when variables are assigned to `None`, pressing shift + enter on the cell block will not output anything.*

Reassigning variables

Now that we have this data, we can imagine using it for some kind of instruction. For example,

Now that we have this data, we can imagine using it for some kind of instruction. For example, say we want to write ourself a memo on how to reach out to someone we just met. Here's the message:

In [13]:
```
"Send an email to Art Vandelay at 'art.vandelay@vandelay.com' to say hov
```

Out[13]:
```
"Send an email to Art Vandelay at 'art.vandelay@vandelay.com' to say h
ow nice it was meeting yesterday."
```

If we construct this message with variables, we can write the following:

In [14]:
```
"Send an email to " + full_name + " at " + email +  " to say how nice it
```

Out[14]:
```
'Send an email to Art Vandelay at art.vandelay@vandelay.com to say how
nice it was meeting yesterday.'
```

Now you meet someone else, "Liz Kaplan" with the email of "liz@ka-plan.com (mailto:liz@ka-plan.com)" and want to write a memo with the same instructions. Only two things vary in the memo: name and email. This should be easy enough given the way we set up our code above. First we need to change the variables, `full_name` and `email`, by setting them to our new data.

In [15]:
```
full_name = 'Liz Kaplan'
email = 'liz@ka-plan.com'
```

So as you can see, we reassign our variables by just setting `variable = 'new data'`. Presto, our variable is then updated.

In [16]:
```
full_name # 'Liz Kaplan'
```

Out[16]:
```
'Liz Kaplan'
```

In [17]:
```
email # 'liz@ka-plan.com'
```

Out[17]:
```
'liz@ka-plan.com'
```

Now, if we copy and re-run our previous code, we will see it is automatically updated.

In [18]:
```
"Send an email to " + full_name + " at " + email +  " to say how nice it
```

Out[18]:
```
'Send an email to Liz Kaplan at liz@ka-plan.com to say how nice it was
meeting yesterday.'
```

In the line above, we are getting to some of the real power of programming. By choosing the correct variable name, we can begin to change the values of `full_name` or `email` and operate on their underlying values in the same ways.

Summary

In this lesson, we got a taste for what makes computer programs so powerful. By using variables, we can write programs that know how to combine data. This can save us time by avoiding boring, repetitive tasks. We declare and assign a variable with the pattern of `variable = data`, and reassign a variable with the same pattern. To reference a variable, we simply type the variable's name.

We also saw that one of the things to pay attention to when working with variables is that they are sometimes different from what we expect. Just type the name of the variable to see what it really is and make any necessary changes.