Object Initialization ¶

Introduction

Now that you've begun to see OOP and class structures, it's time to investigate the `__init__` method more. The `__init__` method allows classes to have default behaviors and attributes.

Objectives

You will be able to:

- Create instance variables in the `__init__` method
- Use default arguments in the `__init__` method

Introducing `__init__`

By using the `__init__` method, you can initialize instances of objects with defined attributes. Without this, attributes are not defined until other methods are called to populate these fields, or you set attributes manually. This can be problematic. For example, if you had tried to call the `greet_passeneger()` method from the previous lab without first setting the driver's first and last attributes, you would have encountered an error. Here's another example to demonstrate:

```
In [1]: class Person:
            def set_name(self, name):
                self.name = name
            def set_job(self, job):
                self.job = job
```

```
In [2]: bob = Person()
```

If we try to access an attribute before setting it we'll get an error.

```
In [3]: bob.name
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-3-b123a67a06c2> in <module>()
----> 1 bob.name

AttributeError: 'Person' object has no attribute 'name'
```

```
In [4]: bob.set_name('Bob')
```

```
In [5]: bob.name
```

```
Out[5]: 'Bob'
```

To avoid errors such as this, you can use the `__init__` method to set attributes on instantiation.

```
In [6]: class Person:
            def __init__(self, name, job):
                self.name = name
                self.job = job
```

```
In [7]: bob = Person('Bob', 'Carpenter')
        print(bob.name)
        print(bob.job)
```

```
Bob
Carpenter
```

Written like this, these arguments then become required:

In [8]:
```
someone = Person()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-8-1ac56b0c183e> in <module>()
----> 1 someone = Person()

TypeError: __init__() missing 2 required positional arguments: 'name' and 'job'
```

Setting default arguments in the `__init__` method

To circumvent this, we can also define `__init__` to have default arguments. This allows parameters to be specified if desired but are not required.

In [9]:
```
class Person:
    def __init__(self, name=None, job=None):
        self.name = name
        self.job = job
```

In [10]:
```
someone = Person()
print(someone.name)
print(someone.job)

print('\n')
governer = Person(job = 'Carpenter')
print(governer.name)
print(governer.job)

print('\n')
bob = Person('Bob', 'Carpenter')
print(bob.name)
print(bob.job)
```

```
None
None


None
Carpenter


Bob
Carpenter
```

Summary

In this lesson, you got a brief introduction to the `__init__` method and how you can use it to set attributes when objects are initialized, including default parameters.