Class Troubleshooting

by Sophia

WHAT'S COVERED

In this lesson, we'll look at some troubleshooting techniques that can help identify what the classes consist of. Specifically, this lesson covers:

1. Using the help() Function
2. Using the dict Method

1. Using the help() Function

Getting information about a class using the `help()` function

When we have different classes defined, we're not always able to know what each class contains. Luckily, Python has a built-in **help()** function that you can use with any class to get more information about that class. This will be helpful when it comes to troubleshooting classes and inheritance properties.

Let's take a look at the `Member` class and subclasses that we created in a prior lesson.

EXAMPLE

```
import datetime

class Member:
  expiry_days = 365
  def __init__(self, first, last):
    self.first_name = first
    self.last_name = last

    self.expiry_date = datetime.date.today() + datetime.timedelta(days = self.expiry_days)

  def show_expiry(self):
    return f'{self.first_name} {self.last_name} expires on {self.expiry_date}'

  def show_status(self):
    return f'{self.first_name} {self.last_name} is a Member'

#Subclass for us to use for administrators
class Admin(Member):
  expiry_days = 365 * 100

  def __init__(self, first, last, secret):
    super().__init__(first,last)
    self.secret_code = secret
  def show_status(self):
    return f'{self.first_name} {self.last_name} is an Admin'

#Subclass for us to use for normal users
class User(Member):
  def show_status(self):
    return f'{self.first_name} {self.last_name} is a User'
```

TRY IT

Directions: Enter this program in the IDE (if you don't have it already in there) so you can follow the troubleshooting techniques that Python offers. Notice, we have not added the class instances or `print()` functions, only the base class and subclass definitions.

After the code, we'll run the `help()` function on the `Member` class by entering the following line of code.

EXAMPLE

`help(Member)`
Here is that output.

```
Help on class Member in module __main__:

class Member(builtins.object)
 |  Member(first, last)
 |
 |  Methods defined here:
 |
 |  __init__(self, first, last)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  show_expiry(self)
 |
 |  show_status(self)
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors defined here:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  expiry_days = 365
```

TRY IT

Directions: Go ahead and add that line of code `help(Member)` at the bottom of the program, just under the `User` subclass definition. Then run the program.

With the results, we'll see a few things that look familiar. You don't have to worry about all of the sections but you'll see that the methods of the member are listed at the top and the extra data elements are listed at the bottom.

If we run the `help()` function on `Admin`, we should see the following.

EXAMPLE

`help(Admin)`
Here is that output.

```
Help on class Admin in module __main__:

class Admin(Member)
 |  Admin(first, last, secret)
 |
 |  #Subclass for us to use for administrators
 |
 |  Method resolution order:
 |      Admin
 |      Member
 |      builtins.object
 |
 |  Methods defined here:
 |
 |  __init__(self, first, last, secret)
 |      Initialize self.  See help(type(self)) for accurate signature.
 |
 |  show_status(self)
```

```
 |
 |  ----------------------------------------------------------------------
 |  Data and other attributes defined here:
 |
 |  expiry_days = 36500
 |
 |  ----------------------------------------------------------------------
 |  Methods inherited from Member:
 |
 |  show_expiry(self)
 |
 |  ----------------------------------------------------------------------
 |  Data descriptors inherited from Member:
 |
 |  __dict__
 |      dictionary for instance variables (if defined)
 |
 |  __weakref__
 |      list of weak references to the object (if defined)
```

TRY IT

Directions: Now substitute `Admin` for the `Member` in the `help()` function and run the program again.

There is a new section that we'll see now called the "Method resolution order". This section is quite important. If a class has a method that exists in multiple classes like our `show_status()` method, the **method resolution order** tells us which classes will be looked at first. In our case, it would first check the `Admin` class, then the `Member` class, and then lastly the `builtins.object` class, which is a default one for all classes that are created in Python. It's a reference to the built-in methods that all classes and subclasses share. It is quite important to identify which `show_status()` method would be referenced and in what order.

If we store our data in hierarchies of classes and subclasses, when we call a method in a subclass (`Admin` in this case), Python will use that subclass method if it exists. If it does not exist in the subclass, Python will then try to use the base class (`Member` in this case) if it exists. If that also does not exist, it will then try the built-in methods. And if all else fails, it will throw an error as it cannot find the method that our code is trying to call. Usually, the main reason for this type of error would just be that we misspelled the method name in our code and Python can't find it.

In using the `help()` function, you are able to look up the classes, methods, and functions and understand how to use them without having to look at ancillary resources. This is especially important when you are developing since modules can change between versions. By using the `help()` function, you can ensure that the code that you are developing will work in that version.

TERMS TO KNOW

help()
Python has a built-in `help()` function that you can use with any class to get more information about that class.

Method Resolution Order
If a class has a method that exists in multiple classes, the method resolution order tells us which classes will be looked at first.

---

2. Using the __dict__ Method

One example of a built-in method that we saw in the `__help()__` function's output is the `__dict__` method. The dict is short for dictionary. Although we never defined a method called `__dict__`, there's a built-in method with that name.

The **`__dict__` method** contains a dictionary collection of all of the attributes and methods for that object.

Let's take a look at what happens if we call the `__dict__` method on the `Admin` subclass and output to the screen.

EXAMPLE

```
import datetime

class Member:
  expiry_days = 365
  def __init__(self, first, last):
    self.first_name = first
    self.last_name = last

    self.expiry_date = datetime.date.today() + datetime.timedelta(days = self.expiry_days)
```

```
  def show_expiry(self):
    return f'{self.first_name} {self.last_name} expires on {self.expiry_date}'

  def show_status(self):
    return f'{self.first_name} {self.last_name} is a Member'

#Subclass for us to use for administrators
class Admin(Member):
  expiry_days = 365 * 100

  def __init__(self, first, last, secret):
    super().__init__(first,last)
    self.secret_code = secret
  def show_status(self):
    return f'{self.first_name} {self.last_name} is an Admin'

#Subclass for us to use for normal users
class User(Member):
  def show_status(self):
    return f'{self.first_name} {self.last_name} is a User'

print(Admin.__dict__)
```

TRY IT

Directions: Go ahead and add the `__dict__` method on the `Admin` class at the end of the program and select the Run button.

The results are a dictionary result of the methods for the `Admin` class and that includes both the ones we've defined as well as the built-in ones.

```
{'__module__': '__main__', 'expiry_days': 36500, '__init__': <function Admin.__init__ at 0x7fc2e3ab1b80>, 'show_status': <function Admin.s
```

We can see that the `show_status()` method is part of the `Admin` class.

Although the method resolution order isn't something you have to get too involved in, it's helpful to look at this for classes, methods, and properties to see what they contain, especially when debugging a program. Note that if you try to call a method that doesn't exist at the subclass, base class, or built-in method, you should get an error.

EXAMPLE

```
print(Admin.methodThatDoesNotExist())
```

Here is that output.

```
Trackback (most recent call last):
  File "/home/main.py", line 32, in <module>
   print(Admin.methodThatDoesNotExist())
AttributeError: type object 'Admin' has no attribute 'methodThatDoesNotExist'
```

The error tells us that Python doesn't know what the `methodThatDoesnotExist()` is about. This is either due to us not creating the method or potentially misspelling the method name in our code.

When we use `__dict__` method, we are able to identify where the methods, functions, and variables names originate from. This allows us to troubleshoot errors related to issues that originate from those names. For example, in our call above, we are able to identify that the `show_status()` method is inside of the `Admin` class. If we had an issue with the `show_status()` method, we would look in the `Admin` class first. If a method was showing up as part of another class, we would not look in the `Admin` class but go to the referenced class.

TERM TO KNOW

__dict__

The `__dict__` method contains a dictionary collection of all of the attributes and methods for that object.

SUMMARY

In this lesson, we learned that there are a few troubleshooting techniques that we can utilize to understand what a class contains and if the class contains methods, what the method resolution order would be. This resolution allows us to see the order that the program will take if we are using a method that exists in multiple classes. Classes and subclasses are heavily used in Python since they help speed up coding and make programs more efficient, so it is helpful to know that there are techniques like **using the** `help()` **function** and **the** `__dict__` **method** to help better understand our classes, including method execution order and what is contained (or not contained) in our classes.

Best of luck in your learning!

Source: THIS CONTENT AND SUPPLEMENTAL MATERIAL HAS BEEN ADAPTED FROM “PYTHON FOR EVERYBODY” BY DR. CHARLES R. SEVERANCE ACCESS FOR FREE AT www.py4e.com/html3/ LICENSE: CREATIVE COMMONS ATTRIBUTION 3.0 UNPORTED.

TERMS TO KNOW

Method Resolution Order

If a class has a method that exists in multiple classes, the method resolution order tells us which classes will be looked at first.

__dict__

The `__dict__` method contains a dictionary collection of all of the attributes and methods for that object.

help()

Python has a built-in `help()` function that you can use with any class to get more information about that class.