Solutions - Chapter 9

9-1: Restaurant

Make a class called Restaurant. The \_\_init\_\_() method for Restaurant should store two attributes: a restaurant\_name and a cuisine\_type. Make a method called describe\_restaurant() that prints these two pieces of information, and a method called open\_restaurant() that prints a message indicating that the restaurant is open.

Make an instance called restaurant from your class. Print the two attributes individually, and then call both methods.

**class** **Restaurant**():

"""A class representing a restaurant."""

**def** **\_\_init\_\_**(self, name, cuisine\_type):

"""Initialize the restaurant."""

self**.**name **=** name**.**title()

self**.**cuisine\_type **=** cuisine\_type

**def** **describe\_restaurant**(self):

"""Display a summary of the restaurant."""

msg **=** self**.**name **+** " serves wonderful " **+** self**.**cuisine\_type **+** "."

**print**("\n" **+** msg)

**def** **open\_restaurant**(self):

"""Display a message that the restaurant is open."""

msg **=** self**.**name **+** " is open. Come on in!"

**print**("\n" **+** msg)

restaurant **=** Restaurant('the mean queen', 'pizza')

**print**(restaurant**.**name)

**print**(restaurant**.**cuisine\_type)

restaurant**.**describe\_restaurant()

restaurant**.**open\_restaurant()

Output:

The Mean Queen

pizza

The Mean Queen serves wonderful pizza.

The Mean Queen is open. Come on in!

9-2: Three Restaurants

Start with your class from Exercise 9-1. Create three different instances from the class, and calldescribe\_restaurant() for each instance.

**class** **Restaurant**():

"""A class representing a restaurant."""

**def** **\_\_init\_\_**(self, name, cuisine\_type):

"""Initialize the restaurant."""

self**.**name **=** name**.**title()

self**.**cuisine\_type **=** cuisine\_type

**def** **describe\_restaurant**(self):

"""Display a summary of the restaurant."""

msg **=** self**.**name **+** " serves wonderful " **+** self**.**cuisine\_type **+** "."

**print**("\n" **+** msg)

**def** **open\_restaurant**(self):

"""Display a message that the restaurant is open."""

msg **=** self**.**name **+** " is open. Come on in!"

**print**("\n" **+** msg)

mean\_queen **=** Restaurant('the mean queen', 'pizza')

mean\_queen**.**describe\_restaurant()

ludvigs **=** Restaurant("ludvig's bistro", 'seafood')

ludvigs**.**describe\_restaurant()

mango\_thai **=** Restaurant('mango thai', 'thai food')

mango\_thai**.**describe\_restaurant()

Output:

The Mean Queen serves wonderful pizza.

Ludvig'S Bistro serves wonderful seafood.

Mango Thai serves wonderful thai food.

9-3: Users

Make a class called User. Create two attributes called first\_name and last\_name, and then create several other attributes that are typically stored in a user profile. Make a method calleddescribe\_user() that prints a summary of the user’s information. Make another method calledgreet\_user() that prints a personalized greeting to the user.

Create several instances representing different users, and call both methods for each user.

**class** **User**():

"""Represent a simple user profile."""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""Initialize the user."""

self**.**first\_name **=** first\_name**.**title()

self**.**last\_name **=** last\_name**.**title()

self**.**username **=** username

self**.**email **=** email

self**.**location **=** location**.**title()

**def** **describe\_user**(self):

"""Display a summary of the user's information."""

**print**("\n" **+** self**.**first\_name **+** " " **+** self**.**last\_name)

**print**(" Username: " **+** self**.**username)

**print**(" Email: " **+** self**.**email)

**print**(" Location: " **+** self**.**location)

**def** **greet\_user**(self):

"""Display a personalized greeting to the user."""

**print**("\nWelcome back, " **+** self**.**username **+** "!")

eric **=** User('eric', 'matthes', 'e\_matthes', 'e\_matthes@example.com', 'alaska')

eric**.**describe\_user()

eric**.**greet\_user()

willie **=** User('willie', 'burger', 'willieburger', 'wb@example.com', 'alaska')

willie**.**describe\_user()

willie**.**greet\_user()

Output:

Eric Matthes

Username: e\_matthes

Email: e\_matthes@example.com

Location: Alaska

Welcome back, e\_matthes!

Willie Burger

Username: willieburger

Email: wb@example.com

Location: Alaska

Welcome back, willieburger!

9-4: Number Served

Start with your program from Exercise 9-1 (page 166). Add an attribute called number\_served with a default value of 0. Create an instance called restaurant from this class. Print the number of customers the restaurant has served, and then change this value and print it again.

Add a method called set\_number\_served() that lets you set the number of customers that have been served. Call this method with a new number and print the value again.

Add a method called increment\_number\_served() that lets you increment the number of customers who’ve been served. Call this method with any number you like that could represent how many customers were served in, say, a day of business.

**class** **Restaurant**():

"""A class representing a restaurant."""

**def** **\_\_init\_\_**(self, name, cuisine\_type):

"""Initialize the restaurant."""

self**.**name **=** name**.**title()

self**.**cuisine\_type **=** cuisine\_type

self**.**number\_served **=** 0

**def** **describe\_restaurant**(self):

"""Display a summary of the restaurant."""

msg **=** self**.**name **+** " serves wonderful " **+** self**.**cuisine\_type **+** "."

**print**("\n" **+** msg)

**def** **open\_restaurant**(self):

"""Display a message that the restaurant is open."""

msg **=** self**.**name **+** " is open. Come on in!"

**print**("\n" **+** msg)

**def** **set\_number\_served**(self, number\_served):

"""Allow user to set the number of customers that have been served."""

self**.**number\_served **=** number\_served

**def** **increment\_number\_served**(self, additional\_served):

"""Allow user to increment the number of customers served."""

self**.**number\_served **+=** additional\_served

restaurant **=** Restaurant('the mean queen', 'pizza')

restaurant**.**describe\_restaurant()

**print**("\nNumber served: " **+** str(restaurant**.**number\_served))

restaurant**.**number\_served **=** 430

**print**("Number served: " **+** str(restaurant**.**number\_served))

restaurant**.**set\_number\_served(1257)

**print**("Number served: " **+** str(restaurant**.**number\_served))

restaurant**.**increment\_number\_served(239)

**print**("Number served: " **+** str(restaurant**.**number\_served))

Output:

The Mean Queen serves wonderful pizza.

Number served: 0

Number served: 430

Number served: 1257

Number served: 1496

9-5: Login Attempts

Add an attribute called login\_attempts to your User class from Exercise 9-3 (page 166). Write amehtod called increment\_login\_attempts() that increments the value of login\_attempts by 1. Write another method called reset\_login\_attempts() that resets the value of login\_attempts to 0.

Make an instance of the User class and call increment\_login\_attempts() several times. Print the value of login\_attempts to make sure it was incremented properly, and then callreset\_login\_attempts(). Print login\_attempts again to make sure it was reset to 0.

**class** **User**():

"""Represent a simple user profile."""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""Initialize the user."""

self**.**first\_name **=** first\_name**.**title()

self**.**last\_name **=** last\_name**.**title()

self**.**username **=** username

self**.**email **=** email

self**.**location **=** location**.**title()

self**.**login\_attempts **=** 0

**def** **describe\_user**(self):

"""Display a summary of the user's information."""

**print**("\n" **+** self**.**first\_name **+** " " **+** self**.**last\_name)

**print**(" Username: " **+** self**.**username)

**print**(" Email: " **+** self**.**email)

**print**(" Location: " **+** self**.**location)

**def** **greet\_user**(self):

"""Display a personalized greeting to the user."""

**print**("\nWelcome back, " **+** self**.**username **+** "!")

**def** **increment\_login\_attempts**(self):

"""Increment the value of login\_attempts."""

self**.**login\_attempts **+=** 1

**def** **reset\_login\_attempts**(self):

"""Reset login\_attempts to 0."""

self**.**login\_attempts **=** 0

eric **=** User('eric', 'matthes', 'e\_matthes', 'e\_matthes@example.com', 'alaska')

eric**.**describe\_user()

eric**.**greet\_user()

**print**("\nMaking 3 login attempts...")

eric**.**increment\_login\_attempts()

eric**.**increment\_login\_attempts()

eric**.**increment\_login\_attempts()

**print**(" Login attempts: " **+** str(eric**.**login\_attempts))

**print**("Resetting login attempts...")

eric**.**reset\_login\_attempts()

**print**(" Login attempts: " **+** str(eric**.**login\_attempts))

Output:

Eric Matthes

Username: e\_matthes

Email: e\_matthes@example.com

Location: Alaska

Welcome back, e\_matthes!

Making 3 login attempts...

Login attempts: 3

Resetting login attempts...

Login attempts: 0

9-6: Ice Cream Stand

An ice cream stand is a specific kind of restaurant. Write a class called IceCreamStand that inherits from the Restaurant class you wrote in Exercise 9-1 (page 166) or Exercise 9-4 (page 171). Eitehr version of the class will work; just pick the one you like better. Add an attribute called flavors that stores a list of ice cream flavors. Write a method that displays theese flavors. Create an instance ofIceCreamStand, and call this method.

**class** **Restaurant**():

"""A class representing a restaurant."""

**def** **\_\_init\_\_**(self, name, cuisine\_type):

"""Initialize the restaurant."""

self**.**name **=** name**.**title()

self**.**cuisine\_type **=** cuisine\_type

self**.**number\_served **=** 0

**def** **describe\_restaurant**(self):

"""Display a summary of the restaurant."""

msg **=** self**.**name **+** " serves wonderful " **+** self**.**cuisine\_type **+** "."

**print**("\n" **+** msg)

**def** **open\_restaurant**(self):

"""Display a message that the restaurant is open."""

msg **=** self**.**name **+** " is open. Come on in!"

**print**("\n" **+** msg)

**def** **set\_number\_served**(self, number\_served):

"""Allow user to set the number of customers that have been served."""

self**.**number\_served **=** number\_served

**def** **increment\_number\_served**(self, additional\_served):

"""Allow user to increment the number of customers served."""

self**.**number\_served **+=** additional\_served

**class** **IceCreamStand**(Restaurant):

"""Represent an ice cream stand."""

**def** **\_\_init\_\_**(self, name, cuisine\_type**=**'ice\_cream'):

"""Initialize an ice cream stand."""

super()**.**\_\_init\_\_(name, cuisine\_type)

self**.**flavors **=** []

**def** **show\_flavors**(self):

"""Display the flavors available."""

**print**("\nWe have the following flavors available:")

**for** flavor **in** self**.**flavors:

**print**("- " **+** flavor**.**title())

big\_one **=** IceCreamStand('The Big One')

big\_one**.**flavors **=** ['vanilla', 'chocolate', 'black cherry']

big\_one**.**describe\_restaurant()

big\_one**.**show\_flavors()

Output:

The Big One serves wonderful ice\_cream.

We have the following flavors available:

- Vanilla

- Chocolate

- Black Cherry

9-7: Admin

An administrator is a special kind of user. Write a class called Admin that inherits from the Userclass you wrote in Exercise 9-3 (page 166) or Exercise 9-5 (page 171). Add an attribute, privileges, that stores a list of strings like "can add post", "can delete post", "can ban user", and so on. WRite a method called show\_privileges() that lists the administrator’s set of privileges. Create an instance of Admin, and call your method.

**class** **User**():

"""Represent a simple user profile."""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""Initialize the user."""

self**.**first\_name **=** first\_name**.**title()

self**.**last\_name **=** last\_name**.**title()

self**.**username **=** username

self**.**email **=** email

self**.**location **=** location**.**title()

self**.**login\_attempts **=** 0

**def** **describe\_user**(self):

"""Display a summary of the user's information."""

**print**("\n" **+** self**.**first\_name **+** " " **+** self**.**last\_name)

**print**(" Username: " **+** self**.**username)

**print**(" Email: " **+** self**.**email)

**print**(" Location: " **+** self**.**location)

**def** **greet\_user**(self):

"""Display a personalized greeting to the user."""

**print**("\nWelcome back, " **+** self**.**username **+** "!")

**def** **increment\_login\_attempts**(self):

"""Increment the value of login\_attempts."""

self**.**login\_attempts **+=** 1

**def** **reset\_login\_attempts**(self):

"""Reset login\_attempts to 0."""

self**.**login\_attempts **=** 0

**class** **Admin**(User):

"""A user with administrative privileges."""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""Initialize the admin."""

super()**.**\_\_init\_\_(first\_name, last\_name, username, email, location)

self**.**privileges **=** []

**def** **show\_privileges**(self):

"""Display the privileges this administrator has."""

**print**("\nPrivileges:")

**for** privilege **in** self**.**privileges:

**print**("- " **+** privilege)

eric **=** Admin('eric', 'matthes', 'e\_matthes', 'e\_matthes@example.com', 'alaska')

eric**.**describe\_user()

eric**.**privileges **=** [

'can reset passwords',

'can moderate discussions',

'can suspend accounts',

]

eric**.**show\_privileges()

Output:

Eric Matthes

Username: e\_matthes

Email: e\_matthes@example.com

Location: Alaska

Privileges:

- can reset passwords

- can moderate discussions

- can suspend accounts

9-8: Privileges

Write a separate Privileges class. The class should have one attribute, privileges, that stores a list of strings as described in Exercise 9-7. Move the show\_privileges() method to this class. Make aPrivileges instance as an attribute in the Admin class. Create a new instance of Admin and use your method to show its privileges.

**class** **User**():

"""Represent a simple user profile."""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""Initialize the user."""

self**.**first\_name **=** first\_name**.**title()

self**.**last\_name **=** last\_name**.**title()

self**.**username **=** username

self**.**email **=** email

self**.**location **=** location**.**title()

self**.**login\_attempts **=** 0

**def** **describe\_user**(self):

"""Display a summary of the user's information."""

**print**("\n" **+** self**.**first\_name **+** " " **+** self**.**last\_name)

**print**(" Username: " **+** self**.**username)

**print**(" Email: " **+** self**.**email)

**print**(" Location: " **+** self**.**location)

**def** **greet\_user**(self):

"""Display a personalized greeting to the user."""

**print**("\nWelcome back, " **+** self**.**username **+** "!")

**def** **increment\_login\_attempts**(self):

"""Increment the value of login\_attempts."""

self**.**login\_attempts **+=** 1

**def** **reset\_login\_attempts**(self):

"""Reset login\_attempts to 0."""

self**.**login\_attempts **=** 0

**class** **Admin**(User):

"""A user with administrative privileges."""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""Initialize the admin."""

super()**.**\_\_init\_\_(first\_name, last\_name, username, email, location)

*# Initialize an empty set of privileges.*

self**.**privileges **=** Privileges()

**class** **Privileges**():

"""A class to store an admin's privileges."""

**def** **\_\_init\_\_**(self, privileges**=**[]):

self**.**privileges **=** privileges

**def** **show\_privileges**(self):

**print**("\nPrivileges:")

**if** self**.**privileges:

**for** privilege **in** self**.**privileges:

**print**("- " **+** privilege)

**else**:

**print**("- This user has no privileges.")

eric **=** Admin('eric', 'matthes', 'e\_matthes', 'e\_matthes@example.com', 'alaska')

eric**.**describe\_user()

eric**.**privileges**.**show\_privileges()

**print**("\nAdding privileges...")

eric\_privileges **=** [

'can reset passwords',

'can moderate discussions',

'can suspend accounts',

]

eric**.**privileges**.**privileges **=** eric\_privileges

eric**.**privileges**.**show\_privileges()

Output:

Eric Matthes

Username: e\_matthes

Email: e\_matthes@example.com

Location: Alaska

Privileges:

- This user has no privileges.

Adding privileges...

Privileges:

- can reset passwords

- can moderate discussions

- can suspend accounts

9-9: Battery Upgrade

Use the final version of *electric\_car.py* from this section. Add a method to the Battery class calledupgrade\_battery(). This method should check the battery size and set the capacity to 85 if it isn’t already. Make an electric car with a default battery size, call get\_range() once, and then callget\_range() a second time after upgrading the battery. You should see an increase in the car’s range.

**class** **Car**():

"""A simple attempt to represent a car."""

**def** **\_\_init\_\_**(self, manufacturer, model, year):

"""Initialize attributes to describe a car."""

self**.**manufacturer **=** manufacturer

self**.**model **=** model

self**.**year **=** year

self**.**odometer\_reading **=** 0

**def** **get\_descriptive\_name**(self):

"""Return a neatly formatted descriptive name."""

long\_name **=** str(self**.**year) **+** ' ' **+** self**.**manufacturer **+** ' ' **+** self**.**model

**return** long\_name**.**title()

**def** **read\_odometer**(self):

"""Print a statement showing the car's mileage."""

**print**("This car has " **+** str(self**.**odometer\_reading) **+** " miles on it.")

**def** **update\_odometer**(self, mileage):

"""

Set the odometer reading to the given value.

Reject the change if it attempts to roll the odometer back.

"""

**if** mileage **>=** self**.**odometer\_reading:

self**.**odometer\_reading **=** mileage

**else**:

**print**("You can't roll back an odometer!")

**def** **increment\_odometer**(self, miles):

"""Add the given amount to the odometer reading."""

self**.**odometer\_reading **+=** miles

**class** **Battery**():

"""A simple attempt to model a battery for an electric car."""

**def** **\_\_init\_\_**(self, battery\_size**=**60):

"""Initialize the batteery's attributes."""

self**.**battery\_size **=** battery\_size

**def** **describe\_battery**(self):

"""Print a statement describing the battery size."""

**print**("This car has a " **+** str(self**.**battery\_size) **+** "-kWh battery.")

**def** **get\_range**(self):

"""Print a statement about the range this battery provides."""

**if** self**.**battery\_size **==** 60:

range **=** 140

**elif** self**.**battery\_size **==** 85:

range **=** 185

message **=** "This car can go approximately " **+** str(range)

message **+=** " miles on a full charge."

**print**(message)

**def** **upgrade\_battery**(self):

"""Upgrade the battery if possible."""

**if** self**.**battery\_size **==** 60:

self**.**battery\_size **=** 85

**print**("Upgraded the battery to 85 kWh.")

**else**:

**print**("The battery is already upgraded.")

**class** **ElectricCar**(Car):

"""Models aspects of a car, specific to electric vehicles."""

**def** **\_\_init\_\_**(self, manufacturer, model, year):

"""

Initialize attributes of the parent class.

Then initialize attributes specific to an electric car.

"""

super()**.**\_\_init\_\_(manufacturer, model, year)

self**.**battery **=** Battery()

**print**("Make an electric car, and check the battery:")

my\_tesla **=** ElectricCar('tesla', 'model s', 2016)

my\_tesla**.**battery**.**describe\_battery()

**print**("\nUpgrade the battery, and check it again:")

my\_tesla**.**battery**.**upgrade\_battery()

my\_tesla**.**battery**.**describe\_battery()

**print**("\nTry upgrading the battery a second time.")

my\_tesla**.**battery**.**upgrade\_battery()

my\_tesla**.**battery**.**describe\_battery()

Output:

Make an electric car, and check the battery:

This car has a 60-kWh battery.

Upgrade the battery, and check it again:

Upgraded the battery to 85 kWh.

This car has a 85-kWh battery.

Try upgrading the battery a second time.

The battery is already upgraded.

This car has a 85-kWh battery.

9-10: Imported Restaurant

Using your latest Restaurant class, store it in a module. Make a separate file that importsRestaurant. Make a Restaurant instance, and call one of Restaurant’s methods to show that theimport statement is working properly.

*restaurant.py:*

"""A class representing a restaurant."""

**class** **Restaurant**():

"""A class representing a restaurant."""

**def** **\_\_init\_\_**(self, name, cuisine\_type):

"""Initialize the restaurant."""

self**.**name **=** name**.**title()

self**.**cuisine\_type **=** cuisine\_type

self**.**number\_served **=** 0

**def** **describe\_restaurant**(self):

"""Display a summary of the restaurant."""

msg **=** self**.**name **+** " serves wonderful " **+** self**.**cuisine\_type **+** "."

**print**("\n" **+** msg)

**def** **open\_restaurant**(self):

"""Display a message that the restaurant is open."""

msg **=** self**.**name **+** " is open. Come on in!"

**print**("\n" **+** msg)

**def** **set\_number\_served**(self, number\_served):

"""Allow user to set the number of customers that have been served."""

self**.**number\_served **=** number\_served

**def** **increment\_number\_served**(self, additional\_served):

"""Allow user to increment the number of customers served."""

self**.**number\_served **+=** additional\_served

*my\_restaurant.py:*

from restaurant import Restaurant

channel\_club **=** Restaurant('the channel club', 'steak and seafood')

channel\_club**.**describe\_restaurant()

channel\_club**.**open\_restaurant()

Output:

The Channel Club serves wonderful steak and seafood.

The Channel Club is open. Come on in!

9-11: Imported Admin

Start with your work from Exercise 9-8 (page 178). Store the classes User, Privileges and Admin in one module. Create a separate file, make an Admin instance, and call show\_priveleges() to show that everything is working correctly.

*user.py:*

"""A collection of classes for modeling users."""

**class** **User**():

"""Represent a simple user profile."""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""Initialize the user."""

self**.**first\_name **=** first\_name**.**title()

self**.**last\_name **=** last\_name**.**title()

self**.**username **=** username

self**.**email **=** email

self**.**location **=** location**.**title()

self**.**login\_attempts **=** 0

**def** **describe\_user**(self):

"""Display a summary of the user's information."""

**print**("\n" **+** self**.**first\_name **+** " " **+** self**.**last\_name)

**print**(" Username: " **+** self**.**username)

**print**(" Email: " **+** self**.**email)

**print**(" Location: " **+** self**.**location)

**def** **greet\_user**(self):

"""Display a personalized greeting to the user."""

**print**("\nWelcome back, " **+** self**.**username **+** "!")

**def** **increment\_login\_attempts**(self):

"""Increment the value of login\_attempts."""

self**.**login\_attempts **+=** 1

**def** **reset\_login\_attempts**(self):

"""Reset login\_attempts to 0."""

self**.**login\_attempts **=** 0

**class** **Admin**(User):

"""A user with administrative privileges."""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""Initialize the admin."""

super()**.**\_\_init\_\_(first\_name, last\_name, username, email, location)

*# Initialize an empty set of privileges.*

self**.**privileges **=** Privileges([])

**class** **Privileges**():

"""Stores privileges associated with an Admin account."""

**def** **\_\_init\_\_**(self, privileges):

"""Initialize the privileges object."""

self**.**privilege **=** privileges

**def** **show\_privileges**(self):

"""Display the privileges this administrator has."""

**for** privilege **in** self**.**privileges:

**print**("- " **+** privilege)

*my\_user.py:*

from user import Admin

eric **=** Admin('eric', 'matthes', 'e\_matthes', 'e\_matthes@example.com', 'alaska')

eric**.**describe\_user()

eric\_privileges **=** [

'can reset passwords',

'can moderate discussions',

'can suspend accounts',

]

eric**.**privileges**.**privileges **=** eric\_privileges

**print**("\nThe admin " **+** eric**.**username **+** " has these privileges: ")

eric**.**privileges**.**show\_privileges()

Output:

Eric Matthes

Username: e\_matthes

Email: e\_matthes@example.com

Location: Alaska

The admin e\_matthes has these privileges:

- can reset passwords

- can moderate discussions

- can suspend accounts

9-12: Multiple Modules

Store the User class in one module, and store the Privileges and Admin classes in a separate module. In a separate file, create an Admin instance and call show\_privileges() to show that everything is still working correctly.

*user.py:*

"""A class for modeling users."""

**class** **User**():

"""Represent a simple user profile."""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""Initialize the user."""

self**.**first\_name **=** first\_name**.**title()

self**.**last\_name **=** last\_name**.**title()

self**.**username **=** username

self**.**email **=** email

self**.**location **=** location**.**title()

self**.**login\_attempts **=** 0

**def** **describe\_user**(self):

"""Display a summary of the user's information."""

**print**("\n" **+** self**.**first\_name **+** " " **+** self**.**last\_name)

**print**(" Username: " **+** self**.**username)

**print**(" Email: " **+** self**.**email)

**print**(" Location: " **+** self**.**location)

**def** **greet\_user**(self):

"""Display a personalized greeting to the user."""

**print**("\nWelcome back, " **+** self**.**username **+** "!")

**def** **increment\_login\_attempts**(self):

"""Increment the value of login\_attempts."""

self**.**login\_attempts **+=** 1

**def** **reset\_login\_attempts**(self):

"""Reset login\_attempts to 0."""

self**.**login\_attempts **=** 0

*admin.py:*

"""A collection of classes for modeling an admin user account."""

from user import User

**class** **Admin**(User):

"""A user with administrative privileges."""

**def** **\_\_init\_\_**(self, first\_name, last\_name, username, email, location):

"""Initialize the admin."""

super()**.**\_\_init\_\_(first\_name, last\_name, username, email, location)

*# Initialize an empty set of privileges.*

self**.**privileges **=** Privileges([])

**class** **Privileges**():

"""Stores privileges associated with an Admin account."""

**def** **\_\_init\_\_**(self, privileges):

"""Initialize the privileges object."""

self**.**privilege **=** privileges

**def** **show\_privileges**(self):

"""Display the privileges this administrator has."""

**for** privilege **in** self**.**privileges:

**print**("- " **+** privilege)

*my\_admin.py*

from admin import Admin

eric **=** Admin('eric', 'matthes', 'e\_matthes', 'e\_matthes@example.com', 'alaska')

eric**.**describe\_user()

eric\_privileges **=** [

'can reset passwords',

'can moderate discussions',

'can suspend accounts',

]

eric**.**privileges**.**privileges **=** eric\_privileges

**print**("\nThe admin " **+** eric**.**username **+** " has these privileges: ")

eric**.**privileges**.**show\_privileges()

Output:

Eric Matthes

Username: e\_matthes

Email: e\_matthes@example.com

Location: Alaska

The admin e\_matthes has these privileges:

- can reset passwords

- can moderate discussions

- can suspend accounts

9-13: OrderedDict Rewrite

Start with Exercise 6-4 (page 108), where you used a standard dictionary to represent a glossary. Rewrite the program using the OrderedDict class and make sure the order of the output matches the order in which key-value pairs were added to the dictionary.

***Note:****In Python 3.6, dictionaries store keys in order. However, this is not guaranteed to work in all versions of Python, so you should still use an OrderedDict when you need key-value pairs to be stored in a particular order.*

from collections import OrderedDict

glossary **=** OrderedDict()

glossary['string'] **=** 'A series of characters.'

glossary['comment'] **=** 'A note in a program that the Python interpreter ignores.'

glossary['list'] **=** 'A collection of items in a particular order.'

glossary['loop'] **=** 'Work through a collection of items, one at a time.'

glossary['dictionary'] **=** "A collection of key-value pairs."

glossary['key'] **=** 'The first item in a key-value pair in a dictionary.'

glossary['value'] **=** 'An item associated with a key in a dictionary.'

glossary['conditional test'] **=** 'A comparison between two values.'

glossary['float'] **=** 'A numerical value with a decimal component.'

glossary['boolean expression'] **=** 'An expression that evaluates to True or False.'

**for** word, definition **in** glossary**.**items():

**print**("\n" **+** word**.**title() **+** ": " **+** definition)

Output:

String: A series of characters.

Comment: A note in a program that the Python interpreter ignores.

List: A collection of items in a particular order.

Loop: Work through a collection of items, one at a time.

Dictionary: A collection of key-value pairs.

Key: The first item in a key-value pair in a dictionary.

Value: An item associated with a key in a dictionary.

Conditional Test: A comparison between two values.

Float: A numerical value with a decimal component.

Boolean Expression: An expression that evaluates to True or False.

9-14: Dice

The module random contains functions that generate random numbers in a variety of ways. The function randint() returns an integer in the range you provide. the following code returns a number between 1 and 6:

from random import randint

x **=** randint(1, 6)

Make a class Die with one attribute called sides, which has a default value of 6. Write a method called roll\_die() that prints a random number between 1 and the number of sides the die has. Make a 6-sided die and roll it 10 times.

Make a 10-sided die and a 20-sided die. Roll each die 10 times.

from random import randint

**class** **Die**():

"""Represent a die, which can be rolled."""

**def** **\_\_init\_\_**(self, sides**=**6):

"""Initialize the die."""

self**.**sides **=** sides

**def** **roll\_die**(self):

"""Return a number between 1 and the number of sides."""

**return** randint(1, self**.**sides)

*# Make a 6-sided die, and show the results of 10 rolls.*

d6 **=** Die()

results **=** []

**for** roll\_num **in** range(10):

result **=** d6**.**roll\_die()

results**.**append(result)

**print**("10 rolls of a 6-sided die:")

**print**(results)

*# Make a 10-sided die, and show the results of 10 rolls.*

d10 **=** Die(sides**=**10)

results **=** []

**for** roll\_num **in** range(10):

result **=** d10**.**roll\_die()

results**.**append(result)

**print**("\n10 rolls of a 10-sided die:")

**print**(results)

*# Make a 20-sided die, and show the results of 10 rolls.*

d20 **=** Die(sides**=**20)

results **=** []

**for** roll\_num **in** range(10):

result **=** d20**.**roll\_die()

results**.**append(result)

**print**("\n10 rolls of a 20-sided die:")

**print**(results)

Output:

10 rolls of a 6-sided die:

[5, 5, 6, 3, 6, 4, 2, 2, 1, 1]

10 rolls of a 10-sided die:

[8, 9, 8, 10, 7, 1, 3, 5, 3, 4]

10 rolls of a 20-sided die:

[4, 3, 18, 17, 3, 1, 13, 12, 5, 14]