


Python Code Reading


Recitation B

Functions [1/7]

```
1 # 8-1 greeter.py
2
3 def greet_user(username):
4     """Display a simple greeting."""
5     print("Hello, " + username.title() + "!")
6
7 greet_user('jesse')
```

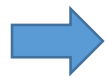


```
1 # 8-2 pets.py
2
3 def describe_pet(pet_name, animal_type='dog'):
4     """Display information about a pet."""
5     print("\nI have a " + animal_type + ".")
6     print("My " + animal_type + "'s name is " + pet_name.title() + ".")
7
8 # A dog named Willie.
9 describe_pet('willie')
10 describe_pet(pet_name='willie')
11
12 # A hamster named Harry.
13 describe_pet('harry', 'hamster')
14 describe_pet(pet_name='harry', animal_type='hamster')
15 describe_pet(animal_type='hamster', pet_name='harry')
```




Functions [2/7]

```
1  # 8-3 formatted_name.py
2
3  def get_formatted_name(first_name, last_name, middle_name=''):
4      """Return a full name, neatly formatted."""
5      if middle_name:
6          full_name = first_name + ' ' + middle_name + ' ' + last_name
7      else:
8          full_name = first_name + ' ' + last_name
9      return full_name.title()
10
11 musician = get_formatted_name('jimi', 'hendrix')
12 print(musician)
13
14 musician = get_formatted_name('john', 'hooker', 'lee')
15 print(musician)
16
```




Functions [3/7]

```
1 # 8-4 peson.py
2
3
4 def build_person(first_name, last_name, age=''):
5     """Return a dictionary of information about a person."""
6     person = {'first': first_name, 'last': last_name}
7     if age:
8         person['age'] = age
9     return person
10
11 musician = build_person('jimi', 'hendrix', age=27)
12 print(musician)
```



```
1 # 8-5 greet_users.py
2
3 def greet_users(names):
4     """Print a simple greeting to each user in the list."""
5     for name in names:
6         msg = "Hello, " + name.title() + "!"
7         print(msg)
8
9 usernames = ['hannah', 'ty', 'margot']
10 greet_users(usernames)
```



Functions [4/7]

```
1  # 8-6 printing_models.py
2
3
4  def print_models(unprinted_designs, completed_models):
5      """
6      Simulate printing each design, until there are none left.
7      Move each design to completed_models after printing.
8      """
9      while unprinted_designs:
10         current_design = unprinted_designs.pop()
11
12         # Simulate creating a 3d print from the design.
13         print("Printing model: " + current_design)
14         completed_models.append(current_design)
15
16  def show_completed_models(completed_models):
17      """Show all the models that were printed."""
18      print("\nThe following models have been printed:")
19      for completed_model in completed_models:
20         print(completed_model)
21
22
23  unprinted_designs = ['iphone case', 'robot pendant', 'dodecahedron']
24  completed_models = []
25
26  print_models(unprinted_designs, completed_models)
27  show_completed_models(completed_models)
```



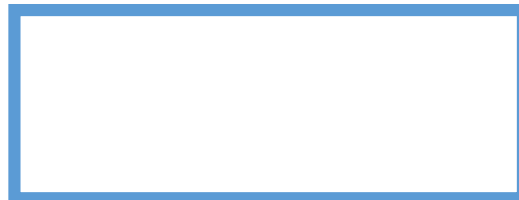
Functions [5/7]

```
1  # 8-7 pizza.py
2
3  def make_pizza(size, *toppings):
4      """Summarize the pizza we are about to make."""
5      print("\nMaking a " + str(size) +
6            "-inch pizza with the following toppings:")
7      for topping in toppings:
8          print("- " + topping)
9
10 make_pizza(16, 'pepperoni')
11 make_pizza(12, 'mushrooms', 'green peppers', 'extra cheese')
```



Functions [6/7]

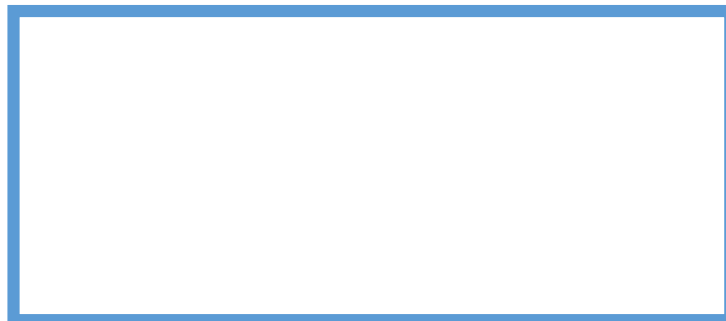
```
1  # 8-8 user_profile.py
2
3  def build_profile(first, last, **user_info):
4      """Build a dictionary containing everything we know about a user."""
5      profile = {}
6      profile['first_name'] = first
7      profile['last_name'] = last
8      for key, value in user_info.items():
9          profile[key] = value
10     return profile
11
12     user_profile = build_profile('albert', 'einstein',
13                                 location='princeton',
14                                 field='physics')
15     print(user_profile)
```



Functions [7/7]

```
3 def make_pizza(size, *toppings):  
4     """Summarize the pizza we are about to make."""  
5     print("\nMaking a " + str(size) +  
6           "-inch pizza with the following toppings:")  
7     for topping in toppings:  
8         print("- " + topping)
```

```
1 # 8-9 making_pizzas.py  
2  
3 import pizza as p  
4  
5 p.make_pizza(16, 'pepperoni')  
6 p.make_pizza(12, 'mushrooms', 'green peppers', 'extra cheese')
```




```
1  # 9-1 dog.py
2
3  class Dog():
4      """A simple attempt to model a dog."""
5
6      def __init__(self, name, age):
7          """Initialize name and age attributes."""
8          self.name = name
9          self.age = age
10
11     def sit(self):
12         """Simulate a dog sitting in response to a command."""
13         print(self.name.title() + " is now sitting.")
14
15     def roll_over(self):
16         """Simulate rolling over in response to a command."""
17         print(self.name.title() + " rolled over!")
18
19
20 my_dog = Dog('willie', 6)
21 your_dog = Dog('lucy', 3)
22
23 print("My dog's name is " + my_dog.name.title() + ".")
24 print("My dog is " + str(my_dog.age) + " years old.")
25 my_dog.sit()
26
27 print("\nMy dog's name is " + your_dog.name.title() + ".")
28 print("My dog is " + str(your_dog.age) + " years old.")
29 your_dog.sit()
```



```

1  # 9-2 car.py
2
3  """A class that can be used to represent a car."""
4
5  class Car():
6      """A simple attempt to represent a car."""
7
8      def __init__(self, manufacturer, model, year):
9          """Initialize attributes to describe a car."""
10         self.manufacturer = manufacturer
11         self.model = model
12         self.year = year
13         self.odometer_reading = 0
14
15         def get_descriptive_name(self):
16             """Return a neatly formatted descriptive name."""
17             long_name = str(self.year) + ' ' + self.manufacturer + ' ' + self.model
18             return long_name.title()
19
20         def read_odometer(self):
21             """Print a statement showing the car's mileage."""
22             print("This car has " + str(self.odometer_reading) + " miles on it.")
23
24         def update_odometer(self, mileage):
25             """
26             Set the odometer reading to the given value.
27             Reject the change if it attempts to roll the odometer back.
28             """
29             if mileage >= self.odometer_reading:
30                 self.odometer_reading = mileage
31             else:
32                 print("You can't roll back an odometer!")
33
34         def increment_odometer(self, miles):
35             """Add the given amount to the odometer reading."""
36             self.odometer_reading += miles

```

```
my_used_car = Car('subaru', 'outback', 2013)
print(my_used_car.get_descriptive_name())

my_used_car.update_odometer(23500)
my_used_car.read_odometer()

my_used_car.increment_odometer(100)
my_used_car.read_odometer()
```



```

1  # 9-3 electric_car.py
2
3
4  """A set of classes that can be used to represent electric cars."""
5
6  from car import Car
7
8  class Battery():
9      """A simple attempt to model a battery for an electric car."""
10
11     def __init__(self, battery_size=60):
12         """Initialize the batteery's attributes."""
13         self.battery_size = battery_size
14
15     def describe_battery(self):
16         """Print a statement describing the battery size."""
17         print("This car has a " + str(self.battery_size) + "-kWh battery.")
18
19     def get_range(self):
20         """Print a statement about the range this battery provides."""
21         if self.battery_size == 60:
22             range = 140
23         elif self.battery_size == 85:
24             range = 185
25
26         message = "This car can go approximately " + str(range)
27         message += " miles on a full charge."
28         print(message)
29
30
31     class ElectricCar(Car):
32         """Models aspects of a car, specific to electric vehicles."""
33
34         def __init__(self, manufacturer, model, year):
35             """
36             Initialize attributes of the parent class.
37             Then initialize attributes specific to an electric car.
38             """
39             super().__init__(manufacturer, model, year)
40             self.battery = Battery()
41

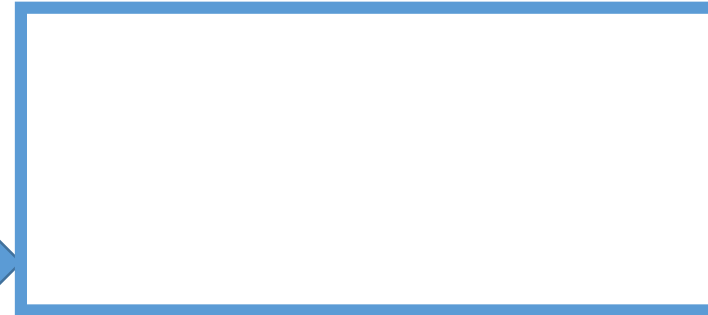

```

```
my_tesla = ElectricCar('tesla', 'model s', 2016)

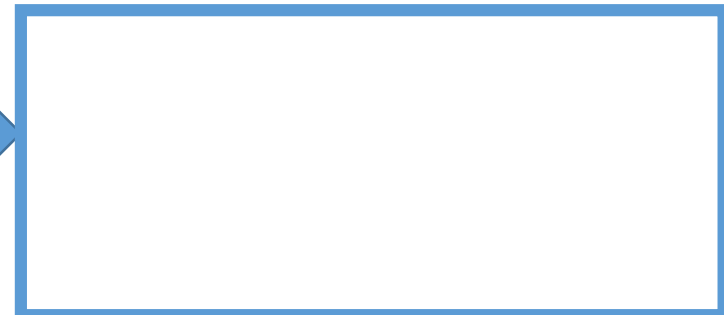

print(my_tesla.get_descriptive_name())
my_tesla.battery.describe_battery()
```




```
1  # 9-4 my_car.py
2
3  from car import Car
4
5  my_new_car = Car('audi', 'a4', 2015)
6  print(my_new_car.get_descriptive_name())
7
8  my_new_car.odometer_reading = 23
9  my_new_car.read_odometer()
10
```



```
1  # 9-5 my_cars.py
2
3  from car import Car
4  from electric_car import ElectricCar
5
6  my_beetle = Car('volkswagen', 'beetle', 2015)
7  print(my_beetle.get_descriptive_name())
8
9  my_tesla = ElectricCar('tesla', 'roadster', 2015)
10 print(my_tesla.get_descriptive_name())
11
```



```

1  # 9-6 favorite_languages.py
2
3  from collections import OrderedDict
4
5  favorite_languages = OrderedDict()
6
7  favorite_languages['jen'] = 'python'
8  favorite_languages['sarah'] = 'c'
9  favorite_languages['edward'] = 'ruby'
10 favorite_languages['phil'] = 'python'
11
12 for name, language in favorite_languages.items():
13     print(name.title() + "'s favorite language is " +
14           language.title() + ".")
15

```

