

Name : Ubaid Ali

ID No: 6840

Assignment: Lab 01

Submitted To: Mr Fahim Ullah khattak

CHAPTER: 03

Introduction to Lists

Task 01

Code:

```
def print_invitation(guest_list):  
    for guest in guest_list:  
        print("Hello " +guest+ ". I would like to  
invite you to dinner.")  
  
guest_list = ['Artistotle', 'Sartre', 'Epictetus']  
print_invitation(guest_list)  
  
print("Oh no! Epictetus was enslaved by the  
Epaphroditus. He can't come to diner!")
```

```
guest_list.remove('Epictetus')
guest_list.append('Senecea')
print_invitation(guest_list)
```

Output:

```
Hello Artistotle. I would like to invite you to dinner.
Hello Sartre. I would like to invite you to dinner.
Hello Epictetus. I would like to invite you to dinner.
Oh no! Epictetus was enslaved by the Epaphroditus. He can't come to diner!
Hello Artistotle. I would like to invite you to dinner.
Hello Sartre. I would like to invite you to dinner.
Hello Senecea. I would like to invite you to dinner.

In [2]:
```

Task 02

Code:

```
def p(guest):
    print("Hello " +guest+ ". I would like to invite you to
dinner.")

def sorry(guest):
    print("Sorry "+guest+ ". Sadly you can't come since we won't
have enough room for all of you.")

guest_list = ['Aristotle', 'Sartre', 'Epictetus']

list(map(p, guest_list))

print(len(guest_list))

print("Oh no! Epictetus was enslaved by the Epaphroditus. He
can't come to diner!")

guest_list.remove('Epictetus')
```

```
guest_list.append('Senecea')
list(map(p, guest_list))
print(len(guest_list))
print("Ordered a bigger table! More guests!")
guest_list.insert(0, 'Kant')
guest_list.insert(2, 'Gautama')
guest_list.append('Confucius')
list(map(p, guest_list))
print(len(guest_list))
print("Oh no! The table won't arrive in time!")
white_list = ['Aristotle', 'Confucius']

#Side effects and parenthesis almost as bad as in lisp...
print(len(list(map(p, filter(lambda guest: guest in
white_list, guest_list)))))

print(len(list(map(sorry, filter(lambda guest: guest not in
white_list, guest_list)))))
```

Output:

```

Hello Aristotle. I would like to invite you to dinner.
Hello Sartre. I would like to invite you to dinner.
Hello Epictetus. I would like to invite you to dinner.
3
Oh no! Epictetus was enslaved by the Epaphroditus. He can't come to diner!
Hello Aristotle. I would like to invite you to dinner.
Hello Sartre. I would like to invite you to dinner.
Hello Seneca. I would like to invite you to dinner.
3
Ordered a bigger table! More guests!
Hello Kant. I would like to invite you to dinner.
Hello Aristotle. I would like to invite you to dinner.
Hello Gautama. I would like to invite you to dinner.
Hello Sartre. I would like to invite you to dinner.
Hello Seneca. I would like to invite you to dinner.
Hello Confucius. I would like to invite you to dinner.
6
Oh no! The table won't arrive in time!
Hello Aristotle. I would like to invite you to dinner.
Hello Confucius. I would like to invite you to dinner.
2
Sorry Kant. Sadly you can't come since we won't have enough room for all of you.
Sorry Gautama. Sadly you can't come since we won't have enough room for all of you.
Sorry Sartre. Sadly you can't come since we won't have enough room for all of you.
Sorry Seneca. Sadly you can't come since we won't have enough room for all of you.
4
None

```

Task 03

Code:

Phillip Ryan Alfred Odling Cohan Gardner

Bradley Fisher Jenny Marshall Meredith Connolly Cassia
Simons

Maegan Terrell Fox Branch Nichola Hale Saskia Allison

Calista Brennan Petra Naylor Anita Lane Patrik Guy

Lillia Salgado Hadiya McBride Ashton McDonnell Rex Chambers
Sama Rios Jarrad Sweet Jokubas Davenport Kathy Cooke Alison
Duke

Aneesa Garner Dalia Bartlett Yisroel Griffiths

Yara Redfern Wiktorina Kearney Alex Cooley Sylvie Todd Eren
Moon

Aya Pitt Lynda Banks Lola Lucas Eve Shah

```
"".split()

for i in range(int(len(friends)/2)):

print("Hello " + friends[2*i] + " " + friends[2*i+1]+ "!"
How are you?")
```

Output:

```
Programming/chapter_03')
Hello Coral Alfaro! How are you?
Hello Phillip Ryan! How are you?
Hello Alfred Odling! How are you?
Hello Cohan Gardner! How are you?
Hello Bradley Fisher! How are you?
Hello Jenny Marshall! How are you?
Hello Meredith Connolly! How are you?
Hello Cassia Simons! How are you?
Hello Maegan Terrell! How are you?
Hello Fox Branch! How are you?
Hello Nichola Hale! How are you?
Hello Saskia Allison! How are you?
Hello Calista Brennan! How are you?
Hello Petra Naylor! How are you?
Hello Anita Lane! How are you?
Hello Patrik Guy! How are you?
Hello Lillia Salgado! How are you?
Hello Hadiya Mcbride! How are you?
Hello Ashton Mcdonnell! How are you?
Hello Rex Chambers! How are you?
Hello Sama Rios! How are you?
Hello Jarrad Sweet! How are you?
Hello Jokubas Davenport! How are you?
Hello Kathy Cooke! How are you?
Hello Alison Duke! How are you?
Hello Aneesa Garner! How are you?
```

Task 04

Code:

```
guest_list = ['Artistotle', 'Sartre', 'Epictetus']

for guest in guest_list:

    print("Hello " +guest+ ". I would like to invite you to
dinner.")
```

Output:

```
Programming/chapter_03')
Hello Artistotle. I would like to invite you to dinner.
Hello Sartre. I would like to invite you to dinner.
Hello Epictetus. I would like to invite you to dinner.

In [9]:
```

Task 05

Code:

```
def print_invitation(guest_list):
    for guest in guest_list:
        print("Hello " +guest+ ". I would like to
invite you to dinner.")

guest_list = ['Artistotle', 'Sartre', 'Epictetus']

print_invitation(guest_list)

print("Oh no! Epictetus was enslaved by the
Epaphroditus. He can't come to diner!")

guest_list.remove('Epictetus')
```

```
guest_list.append('Senecea')
print_invitation(guest_list)
print("Found a bigger table! More guests!")
guest_list.insert(0, 'Kant')
guest_list.insert(2, 'Gautama')
guest_list.append('Confucius')
print_invitation(guest_list)
```

Output:

```
Programming/chapter_03')
Hello Artistotle. I would like to invite you to dinner.
Hello Sartre. I would like to invite you to dinner.
Hello Epictetus. I would like to invite you to dinner.
Oh no! Epictetus was enslaved by the Epaphroditus. He can't come to diner!
Hello Artistotle. I would like to invite you to dinner.
Hello Sartre. I would like to invite you to dinner.
Hello Senecea. I would like to invite you to dinner.
Found a bigger table! More guests!
Hello Kant. I would like to invite you to dinner.|
Hello Artistotle. I would like to invite you to dinner.
Hello Gautama. I would like to invite you to dinner.
Hello Sartre. I would like to invite you to dinner.
Hello Senecea. I would like to invite you to dinner.
Hello Confucius. I would like to invite you to dinner.
In [10]:
```

Task 06

Code:

```
Patrik Guy Lillia Salgado Hadiya Mcbride Ashton
Mcdonnell Rex Chambers Sama Rios Jarrad Sweet
Jokubas Davenport Kathy Cooke Alison Duke
```

```
Aneesa Garner Dalia Bartlett Yisroel Griffiths  
Yara Redfern Wiktorina Kearney Alex Cooley Sylvie  
Todd Eren Moon Aya
```

```
"".split()  
  
for i in range(int(len(friends)/2)):  
    print(friends[2*i] + " " + friends[2*i+1])
```

```
Programming/chapter_03')
```

```
Coral Alfaro  
Phillip Ryan  
Alfred Odling  
Cohan Gardner  
Bradley Fisher  
Jenny Marshall  
Meredith Connolly  
Cassia Simons  
Maegan Terrell  
Fox Branch  
Nichola Hale  
Saskia Allison  
Calista Brennan  
Petra Naylor  
Anita Lane  
Patrik Guy  
Lillia Salgado  
Hadiya McBride  
Ashton McDonnell  
Rex Chambers  
Sama Rios  
Jarrad Sweet  
Jokubas Davenport  
Kathy Cooke  
Alison Duke  
Aneesa Garner
```

Output:

Task 07

Code:

```
transportation = ['car', 'train', 'bike']

print("If it's hot and sunny, I like to ride the "
      + transportation[2] + ".")

print("If I need to transport something heavy and
big I use the" + transportation[0] + ".")

print("Using the " + transportation[1] + " instead of
```

```
the " +transportation[0] + " saves the
environment.")
```

Output:

```
Programming/chapter_03')
If it's hot and sunny, I like to ride the bike.
If I need to transport something heavy and big I use the car.
Using the train instead of the car saves the environment.

In [12]:
```

Task 08

Code:

```
nice_places =
["NYC", "Seattle", "London", "Berlin", "Rome", "Shanghai
"]

print(nice_places)

print(sorted(nice_places))

print(nice_places)

print(sorted(nice_places, reverse=True))

print(nice_places)

nice_places.reverse()

print(nice_places)

nice_places.reverse()

print(nice_places)
```

```
nice_places.sort()
print(nice_places)
nice_places.sort(reverse=True)
print(nice_places)
```

Output:

```
Programming/chapter_03')
['NYC', 'Seattle', 'London', 'Berlin', 'Rome', 'Shanghai']
['Berlin', 'London', 'NYC', 'Rome', 'Seattle', 'Shanghai']
['NYC', 'Seattle', 'London', 'Berlin', 'Rome', 'Shanghai']
['Shanghai', 'Seattle', 'Rome', 'NYC', 'London', 'Berlin']
['NYC', 'Seattle', 'London', 'Berlin', 'Rome', 'Shanghai']
['Shanghai', 'Rome', 'Berlin', 'London', 'Seattle', 'NYC']
['NYC', 'Seattle', 'London', 'Berlin', 'Rome', 'Shanghai']
['Berlin', 'London', 'NYC', 'Rome', 'Seattle', 'Shanghai']
['Shanghai', 'Seattle', 'Rome', 'NYC', 'London', 'Berlin']
In [13]:
```

Task 09

Code:

```
def p(guest):
    print("Hello " +guest+ ". I would like to
invite you to dinner.")

def sorry(guest):
    print("Sorry "+guest+ ". Sadly you can't come
since we won't have enough room for all of you.")

guest_list = ['Aristotle', 'Sartre', 'Epictetus']

list(map(p, guest_list))
```

```
print("Oh no! Epictetus was enslaved by the
Epaphroditus. He can't come to diner!")

guest_list.remove('Epictetus')

guest_list.append('Senecea')

list(map(p, guest_list))

print("Ordered a bigger table! More guests!")

guest_list.insert(0, 'Kant')

guest_list.insert(2, 'Gautama')

guest_list.append('Confucius')

list(map(p, guest_list))

print("Oh no! The table won't arrive in time!")

white_list = ['Aristotle', 'Confucius']

list(map(p, filter(lambda guest: guest in
white_list, guest_list)))

list(map(sorry, filter(lambda guest: guest not in
white_list, guest_list)))

print(guest_list.clear())
```

Output:

```
Introduction to Programming/chapter_03')
Hello Aristotle. I would like to invite you to dinner.
Hello Sartre. I would like to invite you to dinner.
Hello Epictetus. I would like to invite you to dinner.
Oh no! Epictetus was enslaved by the Epaphroditus. He can't come to diner!
Hello Aristotle. I would like to invite you to dinner.
Hello Sartre. I would like to invite you to dinner.
Hello Senecea. I would like to invite you to dinner.
Ordered a bigger table! More guests!
Hello Kant. I would like to invite you to dinner.
Hello Aristotle. I would like to invite you to dinner.
Hello Gautama. I would like to invite you to dinner.
Hello Sartre. I would like to invite you to dinner.
Hello Senecea. I would like to invite you to dinner.
Hello Confucius. I would like to invite you to dinner.
Oh no! The table won't arrive in time!
Hello Aristotle. I would like to invite you to dinner.
Hello Confucius. I would like to invite you to dinner.
Sorry Kant. Sadly you can't come since we won't have enough room for all of you.
Sorry Gautama. Sadly you can't come since we won't have enough room for all of you.
Sorry Sartre. Sadly you can't come since we won't have enough room for all of you.
Sorry Senecea. Sadly you can't come since we won't have enough room for all of you.
None

In [14]:
```

CHAPTER: 04

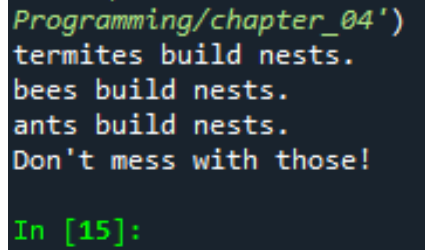
Tasks: Working on Lists

Task 01

Code:

```
animals = ['termites', 'bees', 'ants']
for animal in animals:
    print(animal + " build nests.")
print("Don't mess with those!")
```

Output:



```
Programming/chapter_04')
termites build nests.
bees build nests.
ants build nests.
Don't mess with those!
In [15]:
```

Task 02

Code:

```
food = ('potatoe salad', 'noodles', 'turkey', 'caesar
salad', 'pork')

for f in food:
    print(f)

food = ('potatoe salad', 'soup', 'lasagna', 'caesar
salad', 'pork')

for f in food:
    print(f)
```

Output:

```
Programming/chapter_04')
potatoe salad
noodles
turkey
caesar salad
pork
potatoe salad
soup
lasagna
caesar salad
pork
In [16]:
```

Task 03

Code:

```
numbers = [i for i in range(1,1000001)]

for x in numbers:

    print(x)
```

Output:

```
73951
73952
73953
73954
73955
73956
73957
73958
73959
73960
73961
73962
73963
73964
73965
73966
73967
73968
73969
73970
73971
73972
73973
73974
73975
73976
73977
73978
73979
73980
73981
```

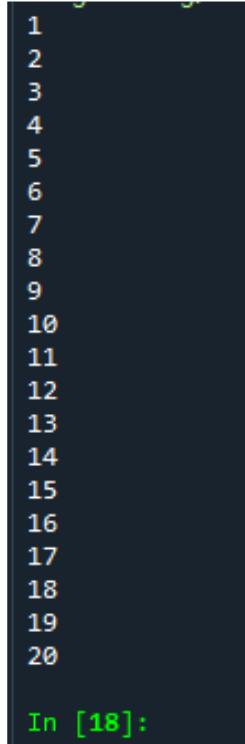
```
492923
492924
492925
492926
492927
492928
492929
492930
492931
492932
492933
492934
492935
492936
492937
492938
492939
492940
492941
492942
492943
492944
492945
492946
492947
492948
492949
492950
492951
492952
492953
```

Task 04

Code:

```
numbers = [i for i in range(1,21)]  
for number in numbers:  
    print(number)
```

Output:



```
1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
  
In [18]:
```

Task 05

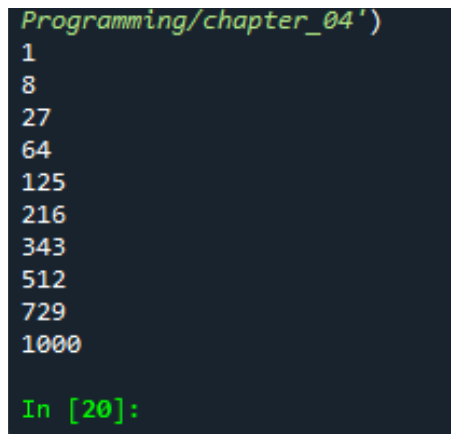
Code:


```
cubes = list(map(lambda x:
x*x*x,list(range(1,11))))

for cube in cubes:

    print(cube)
```

Output:



```
Programming/chapter_04')
1
8
27
64
125
216
343
512
729
1000
In [20]:
```

Task 06

Code:

```
pizzas = ['salami','peperoni','pineapple']

other_pizzas = pizzas[:]

other_pizzas.append('tuna')

print("My favourite pizzas are:")

list(map(print,pizzas))

print("My friends favourite pizzas are:")

list(map(print,other_pizzas))
```

Output:

```
Programming/chapter_04')
My favourite pizzas are:
salami
peperoni
pineapple
My friends favourite pizzas are:
salami
peperoni
pineapple
tuna
In [21]:
```

Task 07

Code:

```
my_foods = ['pizza', 'falafel', 'carrot cake']
friend_foods = my_foods[:]
friend_foods.append('beef')
print("My favorite foods are:")
list(map(print,my_foods))
print("\nMy friend's favorite foods are:")
list(map(print,friend_foods))
```

Output:

```
Programming/chapter_04')
My favorite foods are:
pizza
falafel
carrot cake

My friend's favorite foods are:
pizza
falafel
carrot cake
beef

In [23]:
```

Task 08

Code:

```
multitple_three = [i*3 for i in range(1,11)]
for number in multitple_three:
    print(number)
```

Output:

```
Programming/chapter_04')
3
6
9
12
15
18
21
24
27
30

In [24]:
```

Task 09

Code:

```
numbers = list(filter(lambda x: x % 2
==1,list(range(1,21))))

#numbers = list(range(1,21,2))

for number in numbers:

    print(number)
```

Output:

```
Programming/chapter_04')
1
3
5
7
9
11
13
15
17
19
```

Task 10

Code:

```
pizzas = ['salami', 'peperoni', 'pineapple']
for pizza in pizzas:
    print(pizza)
```

Output:

```
Programming/chapter_04')
salami
peperoni
pineapple
```

Task 11

Code:

```
numbers = [i for i in range(1,120001)]  
print(min(numbers))  
print(max(numbers))  
print(sum(numbers))
```

Output:

```
Programming/chapter_04')  
1  
120000  
7200060000  
  
In [31]:
```

CHAPTER: 06

Task 1

Code:

```
person = {  
  
    'first_name': 'eric',  
    'last_name': 'matthes',  
    'age': 43,  
    'city': 'sitka',
```

```
}  
  
print(person['first_name'])  
print(person['last_name'])  
    print(person['age'])  
    print(person['city'])
```

Output:

eric

matthes

43

sitka

Task 2

Code:

```
favorite_numbers = {  
    'mandy': 42,  
    'micah': 23,
```

```
'gus': 7,  
'hank': 1000000,  
'maggie': 0,  
}
```

```
num = favorite_numbers['mandy']  
print("Mandy's favorite number is " + str(num) + ".")
```

```
num = favorite_numbers['micah']  
print("Micah's favorite number is " + str(num) + ".")
```

```
num = favorite_numbers['gus']  
print("Gus's favorite number is " + str(num) + ".")
```

```
num = favorite_numbers['hank']  
print("Hank's favorite number is " + str(num) + ".")
```

```
num = favorite_numbers['maggie']  
print("Maggie's favorite number is " + str(num) + ".")
```

Output:

Mandy's favorite number is 42.

Micah's favorite number is 23.

Gus's favorite number is 7.

Hank's favorite number is 1000000.

Maggie's favorite number is 0.

Task 3

Code:

```
glossary = {  
    '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.",  
}  
  
word = 'string'
```

```
print("\n" + word.title() + ": " + glossary[word])
```

```
word = 'comment'
```

```
print("\n" + word.title() + ": " + glossary[word])
```

```
word = 'list'
```

```
print("\n" + word.title() + ": " + glossary[word])
```

```
word = 'loop'
```

```
print("\n" + word.title() + ": " + glossary[word])
```

```
word = 'dictionary'
```

```
print("\n" + word.title() + ": " + glossary[word])
```

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.

Task 4

Code:

```
glossary = {  
    '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.',  
}
```

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

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

Output:

Dictionary: A collection of key-value pairs.

String: A series of characters.

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

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

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

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

List: A collection of items in a particular order.

Conditional Test: A comparison between two values.

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

Float: A numerical value with a decimal component.

Task 5

Code:

```
rivers = {  
    'nile': 'egypt',  
    'mississippi': 'united states',  
    'fraser': 'canada',  
    'kuskokwim': 'alaska',  
    'yangtze': 'china',  
}  
  
for river, country in rivers.items():  
    print("The " + river.title() + " flows through " + country.title() + ".")  
  
print("\nThe following rivers are included in this data set:")
```

```
for river in rivers.keys():  
    print("- " + river.title())
```

```
print("\n\nThe following countries are included in this data set:")
```

```
for country in rivers.values():  
    print("- " + country.title())
```

Output*:

The Mississippi flows through United States.

The Yangtze flows through China.

The Fraser flows through Canada.

The Nile flows through Egypt.

The Kuskokwim flows through Alaska.

The following rivers are included in this data set:

- Mississippi
- Yangtze
- Fraser
- Nile
- Kuskokwim

The following countries are included in this data set:

- United States
- China
- Canada
- Egypt
- Alaska

*Sometimes we like to think of Alaska as our own separate country.

Task 6

Code:

```
favorite_languages = {  
    'jen': 'python',  
    'sarah': 'c',  
    'edward': 'ruby',  
    'phil': 'python',  
}
```

```
for name, language in favorite_languages.items():
    print(name.title() + "'s favorite language is " +
          language.title() + ".")

    print("\n")

coders = ['phil', 'josh', 'david', 'becca', 'sarah', 'matt', 'danielle']

    for coder in coders:
        if coder in favorite_languages.keys():
            print("Thank you for taking the poll, " + coder.title() + "!")
        else:
            print(coder.title() + ", what's your favorite programming
                  language?")
```

Output:

Jen's favorite language is Python.

Sarah's favorite language is C.

Phil's favorite language is Python.

Edward's favorite language is Ruby.

Thank you for taking the poll, Phil!

Josh, what's your favorite programming language?

David, what's your favorite programming language?

Becca, what's your favorite programming language?

Thank you for taking the poll, Sarah!

Matt, what's your favorite programming language?

Danielle, what's your favorite programming language?

Task 6

Code:

```
# Make an empty list to store people in.
```

```
people = []
```

```
# Define some people, and add them to the list.
```

```
person = {
```

```
    'first_name': 'eric',
```

```
    'last_name': 'matthes',
```

```
        'age': 43,  
        'city': 'sitka',  
    }  
people.append(person)
```

```
    person = {  
        'first_name': 'ever',  
        'last_name': 'matthes',  
        'age': 5,  
        'city': 'sitka',  
    }  
people.append(person)
```

```
    person = {  
        'first_name': 'willie',  
        'last_name': 'matthes',  
        'age': 8,  
        'city': 'sitka',  
    }  
people.append(person)
```

```
# Display all of the information in the dictionary.

    for person in people:
name = person['first_name'].title() + " " + person['last_name'].title()
        age = str(person['age'])
        city = person['city'].title()

print(name + ", of " + city + ", is " + age + " years old.")
```

Output:

Eric Matthes, of Sitka, is 43 years old.
Ever Matthes, of Sitka, is 5 years old.
Willie Matthes, of Sitka, is 8 years old.

Task 7

Code:

```
# Make an empty list to store the pets in.

pets = []
```

Make individual pets, and store each one in the list.

```
    pet = {  
        'animal type': 'python',  
        'name': 'john',  
        'owner': 'guido',  
        'weight': 43,  
        'eats': 'bugs',  
    }  
    pets.append(pet)
```

```
    pet = {  
        'animal type': 'chicken',  
        'name': 'clarence',  
        'owner': 'tiffany',  
        'weight': 2,  
        'eats': 'seeds',  
    }  
    pets.append(pet)
```

```
    pet = {
```

```
'animal type': 'dog',
    'name': 'peso',
    'owner': 'eric',
    'weight': 37,
    'eats': 'shoes',
    }
pets.append(pet)

# Display information about each pet.
for pet in pets:
    print("\nHere's what I know about " + pet['name'].title() + ":")
        for key, value in pet.items():
            print("\t" + key + ": " + str(value))
```

Output:

Here's what I know about John:

weight: 43

animal type: python

name: john

owner: guido

eats: bugs

Here's what I know about Clarence:

weight: 2

animal type: chicken

name: clarence

owner: tiffany

eats: seeds

Here's what I know about Peso:

weight: 37

animal type: dog

name: peso

owner: eric

eats: shoes

Task 8

Code:

```
favorite_places = {  
    'eric': ['bear mountain', 'death valley', 'tierra del fuego'],  
    'erin': ['hawaii', 'iceland'],  
    'ever': ['mt. verstovia', 'the playground', 'south carolina']  
}  
  
for name, places in favorite_places.items():  
    print("\n" + name.title() + " likes the following places:")  
        for place in places:  
            print("- " + place.title())
```

Output:

Ever likes the following places:

- Mt. Verstovia
- The Playground
- South Carolina

Erin likes the following places:

- Hawaii
- Iceland

Eric likes the following places:

- Bear Mountain
- Death Valley
- Tierra Del Fuego

Task 8

Code:

```
favorite_numbers = {  
    'mandy': [42, 17],  
    'micah': [42, 39, 56],  
    'gus': [7, 12],  
}
```

```
for name, numbers in favorite_numbers.items():
```



```
print("\n" + name.title() + " likes the following numbers:")  
  
    for number in numbers:  
  
        print(" " + str(number))
```

Output:

Micah likes the following numbers:

42

39

56

Mandy likes the following numbers:

42

17

Gus likes the following numbers:

7

12

Task 9

Code:

```
cities = {  
    'santiago': {  
        'country': 'chile',  
        'population': 6158080,  
        'nearby mountains': 'andes',  
    },  
    'talkeetna': {  
        'country': 'alaska',  
        'population': 876,  
        'nearby mountains': 'alaska range',  
    },  
    'kathmandu': {  
        'country': 'nepal',  
        'population': 1003285,  
        'nearby mountains': 'himilaya',  
    }  
}
```

```
for city, city_info in cities.items():  
    country = city_info['country'].title()
```

```
population = city_info['population']
mountains = city_info['nearby mountains'].title()

print("\n" + city.title() + " is in " + country + ".")
print(" It has a population of about " + str(population) + ".")
print(" The " + mountains + " mountains are nearby.")
```

Output:

Santiago is in Chile.

It has a population of about 6158080.

The Andes mountains are nearby.

Kathmandu is in Nepal.

It has a population of about 1003285.

The Himilaya mountains are nearby.

Talkeetna is in Alaska.

It has a population of about 876.

The Alaska Range mountains are nearb

CHAPTER: 07

Task 1

Code:

```
car = input("What kind of car would you like? ")  
  
print("Let me see if I can find you a " + car.title() + ".")
```

Output:

```
What kind of car would you like? Toyota Tacoma  
Let me see if I can find you a Toyota Tacoma.
```

Task 2

Code:

```
party_size = input("How many people are in your dinner party tonight?
")
party_size = int(party_size)

if party_size > 8:
print("I'm sorry, you'll have to wait for a table.")
else:
print("Your table is ready.")
```

Output:

How many people are in your dinner party tonight? 12

I'm sorry, you'll have to wait for a table.

or:

How many people are in your dinner party tonight? 6

Your table is ready.

Task 3

Code:

```
number = input("Give me a number, please: ")
number = int(number)

if number % 10 == 0:
    print(str(number) + " is a multiple of 10.")
else:
    print(str(number) + " is not a multiple of 10.")
```

Output:

Give me a number, please: 23

23 is not a multiple of 10.

or:

Give me a number, please: 90

90 is a multiple of 10.

Task 4

Code:

```
prompt = "\nWhat topping would you like on your pizza?"  
prompt += "\nEnter 'quit' when you are finished: "  
  
while True:  
    topping = input(prompt)  
    if topping != 'quit':  
        print(" I'll add " + topping + " to your pizza.")  
    else:  
        break
```

Output:

```
What topping would you like on your pizza?  
Enter 'quit' when you are finished: pepperoni  
I'll add pepperoni to your pizza.
```

```
What topping would you like on your pizza?
```

Enter 'quit' when you are finished: sausage

I'll add sausage to your pizza.

What topping would you like on your pizza?

Enter 'quit' when you are finished: bacon

I'll add bacon to your pizza.

What topping would you like on your pizza?

Enter 'quit' when you are finished: quit

Task 5

Code:

```
prompt = "How old are you?"  
prompt += "\nEnter 'quit' when you are finished. "
```

```
while True:  
    age = input(prompt)  
    if age == 'quit':
```



```
        break
    age = int(age)

    if age < 3:
        print(" You get in free!")
    elif age < 13:
        print(" Your ticket is $10.")
    else:
        print(" Your ticket is $15.")
```

Output:

```
How old are you?
Enter 'quit' when you are finished. 2
    You get in free!
How old are you?
Enter 'quit' when you are finished. 3
    Your ticket is $10.
How old are you?
Enter 'quit' when you are finished. 12
```

Your ticket is \$10.

How old are you?

Enter 'quit' when you are finished. 18

Your ticket is \$15.

How old are you?

Enter 'quit' when you are finished. quit

Task 7

Code:

```
sandwich_orders = ['veggie', 'grilled cheese', 'turkey', 'roast beef']
```

```
finished_sandwiches = []
```

```
while sandwich_orders:
```

```
    current_sandwich = sandwich_orders.pop()
```

```
    print("I'm working on your " + current_sandwich + " sandwich.")
```

```
    finished_sandwiches.append(current_sandwich)
```

```
print("\n")
```

```
for sandwich in finished_sandwiches:  
print("I made a " + sandwich + " sandwich.")
```

Output:

I'm working on your roast beef sandwich.

I'm working on your turkey sandwich.

I'm working on your grilled cheese sandwich.

I'm working on your veggie sandwich.

I made a roast beef sandwich.

I made a turkey sandwich.

I made a grilled cheese sandwich.

I made a veggie sandwich.

Task 8

Code:

```
sandwich_orders = [
```

```
'pastrami', 'veggie', 'grilled cheese', 'pastrami',
'turkey', 'roast beef', 'pastrami']
finished_sandwiches = []

print("I'm sorry, we're all out of pastrami today.")

while 'pastrami' in sandwich_orders:
    sandwich_orders.remove('pastrami')

    print("\n")

    while sandwich_orders:
        current_sandwich = sandwich_orders.pop()
        print("I'm working on your " + current_sandwich + " sandwich.")
        finished_sandwiches.append(current_sandwich)

    print("\n")

    for sandwich in finished_sandwiches:
        print("I made a " + sandwich + " sandwich.")
```

Output:

I'm sorry, we're all out of pastrami today.

I'm working on your roast beef sandwich.

I'm working on your turkey sandwich.

I'm working on your grilled cheese sandwich.

I'm working on your veggie sandwich.

I made a roast beef sandwich.

I made a turkey sandwich.

I made a grilled cheese sandwich.

I made a veggie sandwich.

Task 10

Code:

```
name_prompt = "\nWhat's your name? "  
place_prompt = "If you could visit one place in the world, where would  
it be? "  
continue_prompt = "\nWould you like to let someone else respond?  
(yes/no) "
```

```
# Responses will be stored in the form {name: place}.

responses = {}

while True:

    # Ask the user where they'd like to go.

    name = input(name_prompt)

    place = input(place_prompt)

    # Store the response.

    responses[name] = place

    # Ask if there's anyone else responding.

    repeat = input(continue_prompt)

    if repeat != 'yes':

        break

    # Show results of the survey.

    print("\n--- Results ---")

    for name, place in responses.items():

        print(name.title() + " would like to visit " + place.title() + ".")
```

Output:

What's your name? eric

If you could visit one place in the world, where would it be? tierra del
fuego

Would you like to let someone else respond? (yes/no) yes

What's your name? erin

If you could visit one place in the world, where would it be? iceland

Would you like to let someone else respond? (yes/no) yes

What's your name? ever

If you could visit one place in the world, where would it be? death valley

Would you like to let someone else respond? (yes/no) no

CHAPTER: 08

Task 1

Code:

```
def display_message():  
    """Display a message about what I'm learning."""  
    msg = "I'm learning to store code in functions."  
    print(msg)  
  
display_message()
```

Output:

I'm learning to store code in functions.

[top](#)

Task 2

Code:

.


```
def favorite_book(title):  
    """Display a message about someone's favorite book."""  
    print(title + " is one of my favorite books.")  
  
favorite_book('The Abstract Wild')
```

Output:

The Abstract Wild is one of my favorite books.

Task 3

Code:

Call the function once using positional arguments to make a shirt. Call the function a second time using keyword arguments.

```
def make_shirt(size, message):  
    """Summarize the shirt that's going to be made."""  
    print("\nI'm going to make a " + size + " t-shirt.")  
    print('It will say, "' + message + "'')
```

```
make_shirt('large', 'I love Python!')
make_shirt(message="Readability counts.", size='medium')
```

Output:

I'm going to make a large t-shirt.

It will say, "I love Python!"

I'm going to make a medium t-shirt.

It will say, "Readability counts."

Task 4

Code:

```
def make_shirt(size='large', message='I love Python!'):
    """Summarize the shirt that's going to be made."""
    print("\nI'm going to make a " + size + " t-shirt.")
    print('It will say, "' + message + "'")

make_shirt()
```

```
make_shirt(size='medium')
make_shirt('small', 'Programmers are loopy.')
```

Output:

I'm going to make a large t-shirt.

It will say, "I love Python!"

I'm going to make a medium t-shirt.

It will say, "I love Python!"

I'm going to make a small t-shirt.

It will say, "Programmers are loopy."

Task 5

Code:

```
def describe_city(city, country='chile'):
    """Describe a city."""
    msg = city.title() + " is in " + country.title() + "."
```

```
print(msg)
```

```
describe_city('santiago')
```

```
describe_city('reykjavik', 'iceland')
```

```
describe_city('punta arenas')
```

Output:

Santiago is in Chile.

Reykjavik is in Iceland.

Punta Arenas is in Chile.

Task 6

Code:

```
“Santiago, Chile”
```

Call your function with at least three city-country pairs, and print the value that’s returned.

```
def city_country(city, country):
```

```
"""Return a string like 'Santiago, Chile'."""
```

```
return(city.title() + ", " + country.title())
```

```
city = city_country('santiago', 'chile')
```

```
print(city)
```

```
city = city_country('ushuaia', 'argentina')
```

```
print(city)
```

```
city = city_country('longyearbyen', 'svalbard')
```

```
print(city)
```

Output:

Santiago, Chile

Ushuaia, Argentina

Longyearbyen, Svalbard

Task 7

Code:

Simple version:

```
def make_album(artist, title):  
    """Build a dictionary containing information about an album."""  
    album_dict = {  
        'artist': artist.title(),  
        'title': title.title(),  
    }  
    return album_dict
```

```
album = make_album('metallica', 'ride the lightning')  
print(album)
```

```
album = make_album('beethoven', 'ninth symphony')  
print(album)
```

```
album = make_album('willie nelson', 'red-headed stranger')  
print(album)
```

Output:

```
{'title': 'Ride The Lightning', 'artist': 'Metallica'}
```

```
{'title': 'Ninth Symphony', 'artist': 'Beethoven'}
```

```
{'title': 'Red-Headed Stranger', 'artist': 'Willie Nelson'}
```

With tracks:

```
def make_album(artist, title, tracks=0):  
    """Build a dictionary containing information about an album."""  
    album_dict = {  
        'artist': artist.title(),  
        'title': title.title(),  
    }  
    if tracks:  
        album_dict['tracks'] = tracks  
    return album_dict  
  
album = make_album('metallica', 'ride the lightning')  
print(album)  
  
album = make_album('beethoven', 'ninth symphony')  
print(album)
```

```
album = make_album('willie nelson', 'red-headed stranger')
print(album)
```

```
album = make_album('iron maiden', 'piece of mind', tracks=8)
print(album)
```

Output:

```
{'artist': 'Metallica', 'title': 'Ride The Lightning'}
{'artist': 'Beethoven', 'title': 'Ninth Symphony'}
{'artist': 'Willie Nelson', 'title': 'Red-Headed Stranger'}
{'tracks': 8, 'artist': 'Iron Maiden', 'title': 'Piece Of Mind'}
```

top

Task 8

Code:

```
def make_album(artist, title, tracks=0):
    """Build a dictionary containing information about an album."""
```



```
album_dict = {
    'artist': artist.title(),
    'title': title.title(),
}
if tracks:
    album_dict['tracks'] = tracks
return album_dict

# Prepare the prompts.
title_prompt = "\nWhat album are you thinking of? "
artist_prompt = "Who's the artist? "

# Let the user know how to quit.
print("Enter 'quit' at any time to stop.")

while True:
    title = input(title_prompt)
    if title == 'quit':
        break

    artist = input(artist_prompt)
```

```
if artist == 'quit':
```

```
    break
```

```
album = make_album(artist, title)
```

```
print(album)
```

```
print("\nThanks for responding!")
```

Output:

Enter 'quit' at any time to stop.

What album are you thinking of? number of the beast

Who's the artist? iron maiden

```
{'artist': 'Iron Maiden', 'title': 'Number Of The Beast'}
```

What album are you thinking of? touch of class

Who's the artist? angel romero

```
{'artist': 'Angel Romero', 'title': 'Touch Of Class'}
```

What album are you thinking of? rust in peace

Who's the artist? megadeth

```
{'artist': 'Megadeth', 'title': 'Rust In Peace'}
```

What album are you thinking of? quit

Thanks for responding!

Task 9

Code:

```
def show_magicians(magicians):  
    """Print the name of each magician in the list."""  
    for magician in magicians:  
        print(magician)  
  
def make_great(magicians):  
    """Add 'the Great!' to each magician's name."""  
    # Build a new list to hold the great musicians.  
    great_magicians = []
```

```
# Make each magician great, and add it to great_magicians.
```

```
    while magicians:
```

```
        magician = magicians.pop()
```

```
        great_magician = magician + ' the Great'
```

```
        great_magicians.append(great_magician)
```

```
# Add the great magicians back into magicians.
```

```
    for great_magician in great_magicians:
```

```
        magicians.append(great_magician)
```

```
magicians = ['Harry Houdini', 'David Blaine', 'Teller']
```

```
    show_magicians(magicians)
```

```
        print("\n")
```

```
        make_great(magicians)
```

```
    show_magicians(magicians)
```

Output:

Harry Houdini

David Blaine

Teller

Teller the Great
David Blaine the Great
Harry Houdini the Great

CHAPTER: 09

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!

top

9-2: Three Restaurants

Start with your class from Exercise 9-1. Create three different instances from the class, and call describe_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.

top

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 called describe_user() that prints a summary of the user's information. Make another method called greet_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!

top

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

top

9-5: Login Attempts

Add an attribute called login_attempts to your User class from Exercise 9-3 (page 166). Write a method 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 call reset_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

top

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). Either 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 these flavors. Create an instance of IceCreamStand, 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

top

9-7: Admin

An administrator is a special kind of user. Write a class called Admin that inherits from the User class 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

top

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 a Privileges 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',
```

1

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

top

9-9: Battery Upgrade

Use the final version of electric car.py from this section. Add a method to the Battery class called upgrade 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 call get 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 battery'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.

top

9-10: Imported Restaurant

Using your latest Restaurant class, store it in a module. Make a separate file that imports Restaurant. Make a Restaurant instance, and call one of Restaurant's methods to show that the import 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!

top

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_privileges() 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

top

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

top

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.

top

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]