ID:11757

NAME: SALMAN KHAN

SUBJECT: MODREN PROGRAMMING LANGUAGE LAB

CHAPTER#3 TASK:

NAME:

```
names = ['ron', 'tyler', 'dani']

print(names[0])

print(names[1])

print(names[2])
```

Output:

```
ron

tyler

dani
```

GREETING:

```
names = ['ron', 'tyler', 'dani']

msg = "Hello, " + names[0].title() + "!"

print(msg)

msg = "Hello, " + names[1].title() + "!"

print(msg)

msg = "Hello, " + names[2].title() + "!"

print(msg)
```

Output:

Hello, Ron!

Hello, Tyler!

Hello, Dani!

YOUR OWN LIST: traffic_tools = ['bicycle' , 'car' , 'motorcycle']

print('I would like to own a ' + traffic_tools[0])

print('I would like to own a ' + traffic_tools[1])

print('I would like to own a ' + traffic_tools[2])

GUEST LIST: guests = ['guido van rossum', 'jack turner', 'lynn hill']

name = guests[0].title()

print(name + ", please come to dinner.")

name = guests[1].title()

print(name + ", please come to dinner.")

name = guests[2].title()

print(name + ", please come to dinner.")

Output:

Guido Van Rossum, please come to dinner.

Jack Turner, please come to dinner.

Lynn Hill, please come to dinner.

CHANGING GUEST LIST: # Invite some people to dinner.

guests = ['guido van rossum', 'jack turner', 'lynn hill']

name = guests[0].title()

print(name + ", please come to dinner.")

name = guests[1].title()

```
print(name + ", please come to dinner.")

name = guests[2].title()

print(name + ", please come to dinner.")

name = guests[1].title()

print("\nSorry, " + name + " can't make it to dinner.")

# Jack can't make it! Let's invite Gary instead.

del(guests[1])

guests.insert(1, 'gary snyder')

# Print the invitations again.

name = guests[0].title()

print("\n" + name + ", please come to dinner.")

name = guests[1].title()

print(name + ", please come to dinner.")

name = guests[2].title()

print(name + ", please come to dinner.")
```

Output:

```
Guido Van Rossum, please come to dinner.

Jack Turner, please come to dinner.

Lynn Hill, please come to dinner.

Sorry, Jack Turner can't make it to dinner.

Guido Van Rossum, please come to dinner.

Gary Snyder, please come to dinner.

Lynn Hill, please come to dinner.
```

MORE GUESTS: # Invite some people to dinner.

```
guests = ['guido van rossum', 'jack turner', 'lynn hill']

name = guests[0].title()

print(name + ", please come to dinner.")

name = guests[1].title()

print(name + ", please come to dinner.")

name = guests[2].title()

print(name + ", please come to dinner.")

name = guests[1].title()

print("\nSorry, " + name + " can't make it to dinner.")

# Jack can't make it! Let's invite Gary instead.

del(guests[1])

guests.insert(1, 'gary snyder')

# Print the invitations again.

name = guests[0].title()

print("\n" + name + ", please come to dinner.")

name = guests[1].title()

print(name + ", please come to dinner.")

name = guests[2].title()

print(name + ", please come to dinner.")

# We got a bigger table, so let's add some more people to the list.

print("\nWe got a bigger table!")

guests.insert(0, 'frida kahlo')

guests.insert(2, 'reinhold messner')

guests.append('elizabeth peratrovich')
```

```
name = guests[0].title()
print(name + ", please come to dinner.")
name = guests[1].title()
print(name + ", please come to dinner.")
name = guests[2].title()
print(name + ", please come to dinner.")
name = guests[3].title()
print(name + ", please come to dinner.")
name = guests[4].title()
print(name + ", please come to dinner.")
name = guests[5].title()
print(name + ", please come to dinner.")
```

Output:

```
Guido Van Rossum, please come to dinner.
Jack Turner, please come to dinner.
Lynn Hill, please come to dinner.
Sorry, Jack Turner can't make it to dinner.
Guido Van Rossum, please come to dinner.
Gary Snyder, please come to dinner.
Lynn Hill, please come to dinner.
We got a bigger table!
Frida Kahlo, please come to dinner.
Guido Van Rossum, please come to dinner.
Reinhold Messner, please come to dinner.
```

Gary Snyder, please come to dinner.

Lynn Hill, please come to dinner.

Elizabeth Peratrovich, please come to dinner.

SHRINKING GUEST LIST: # Invite some people to dinner.

```
guests = ['guido van rossum', 'jack turner', 'lynn hill']

name = guests[0].title()

print(name + ", please come to dinner.")

name = guests[1].title()

print(name + ", please come to dinner.")

name = guests[2].title()

print(name + ", please come to dinner.")

name = guests[1].title()

print("\nSorry, " + name + " can't make it to dinner.")

# Jack can't make it! Let's invite Gary instead.

del(guests[1])

guests.insert(1, 'gary snyder')

# Print the invitations again.

name = guests[0].title()

print("\n" + name + ", please come to dinner.")

name = guests[1].title()

print(name + ", please come to dinner.")

name = guests[2].title()

print(name + ", please come to dinner.")

# We got a bigger table, so let's add some more people to the list.
```

```
print("\nWe got a bigger table!")

guests.insert(0, 'frida kahlo')

guests.insert(2, 'reinhold messner')

guests.append('elizabeth peratrovich')

name = guests[0].title()

print(name + ", please come to dinner.")

name = guests[1].title()

print(name + ", please come to dinner.")

name = guests[2].title()

print(name + ", please come to dinner.")

name = guests[3].title()

print(name + ", please come to dinner.")

name = guests[4].title()

print(name + ", please come to dinner.")

name = guests[5].title()

print(name + ", please come to dinner.")

# Oh no, the table won't arrive on time!

print("\nSorry, we can only invite two people to dinner.")

name = guests.pop()

print("Sorry, " + name.title() + " there's no room at the table.")

name = guests.pop()

print("Sorry, " + name.title() + " there's no room at the table.")

name = guests.pop()

print("Sorry, " + name.title() + " there's no room at the table.")
```

```
name = guests.pop()

print("Sorry, " + name.title() + " there's no room at the table.")

# There should be two people left. Let's invite them.

name = guests[0].title()

print(name + ", please come to dinner.")

name = guests[1].title()

print(name + ", please come to dinner.")

# Empty out the list.

del(guests[0])

del(guests[0])

# Prove the list is empty.

print(guests)
```

Output:

```
Guido Van Rossum, please come to dinner.

Jack Turner, please come to dinner.

Lynn Hill, please come to dinner.

Sorry, Jack Turner can't make it to dinner.

Guido Van Rossum, please come to dinner.

Gary Snyder, please come to dinner.

Lynn Hill, please come to dinner.

We got a bigger table!

Frida Kahlo, please come to dinner.

Guido Van Rossum, please come to dinner.

Reinhold Messner, please come to dinner.
```

Gary Snyder, please come to dinner.

Lynn Hill, please come to dinner.

Elizabeth Peratrovich, please come to dinner.

Sorry, we can only invite two people to dinner.

Sorry, Elizabeth Peratrovich there's no room at the table.

Sorry, Lynn Hill there's no room at the table.

Sorry, Gary Snyder there's no room at the table.

Sorry, Reinhold Messner there's no room at the table.

Frida Kahlo, please come to dinner.

Guido Van Rossum, please come to dinner.

[]

SEEING THE WORLD: locations = ['himalaya', 'andes', 'tierra del fuego', 'labrador', 'guam']

print("Original order:")

print(locations)

print("\nAlphabetical:")

print(sorted(locations))

print("\nOriginal order:")

print(locations)

print("\nReverse alphabetical:")

print(sorted(locations, reverse=True))

print("\nOriginal order:")

print(locations)

print("\nReversed:")

```
locations.reverse()

print(locations)

print("\nOriginal order:")

locations.reverse()

print(locations)

print("\nAlphabetical")

locations.sort()

print(locations)

print("\nReverse alphabetical")

locations.sort(reverse=True)

print(locations)
```

OUTPUT:

```
Original order:

['himalaya', 'andes', 'tierra del fuego', 'labrador', 'guam']

Alphabetical:

['andes', 'guam', 'himalaya', 'labrador', 'tierra del fuego']

Original order:

['himalaya', 'andes', 'tierra del fuego', 'labrador', 'guam']

Reverse alphabetical:

['tierra del fuego', 'labrador', 'himalaya', 'guam', 'andes']

Original order:

['himalaya', 'andes', 'tierra del fuego', 'labrador', 'guam']

Reversed:

['guam', 'labrador', 'tierra del fuego', 'andes', 'himalaya']
```

Original order:

['himalaya', 'andes', 'tierra del fuego', 'labrador', 'guam']

Alphabetical

['andes', 'guam', 'himalaya', 'labrador', 'tierra del fuego']

Reverse alphabetical

['tierra del fuego', 'labrador', 'himalaya', 'guam', 'andes']

DINNER GUESTS: persons = ['guido van rossum', 'jack turner', 'lynn hill']

```
for person in persons:
    print(person + ", I want to invite you to have dinner with me.")
print('I have invited '+str(len(persons))+' persons.')
```

OUTPUT: guido van rossum, I want to invite you to have dinner with me.

jack turner, I want to invite you to have dinner with me.

lynn hill, I want to invite you to have dinner with me.

I have invited 3 persons.

EVERY FUNCTION:

```
elements = ['mountain' , 'river' , 'country' , 'city' , 'language']
elements.append('sport')
elements.insert(0,'comic')
print(elements)
del elements[6]
popped = elements.pop()
print(popped)
elements.remove('country')
```

```
print(elements)

temp = sorted(elements)

print(temp)

print(elements)

elements.sort()

print(elements)

elements.reverse()

print(elements)

print(len(elements))
```

INTENTINAL ERROR: #list=[65,"alex",'dog']

```
print ("Intentional Error calling for del element at -4 which is out of range hehehehehe.....")

#del list[-4]
```

CHAPTER#4 TASK:

PIZZAS: favorite_pizzas = ['pepperoni', 'hawaiian', 'veggie']

```
# Print the names of all the pizzas.

for pizza in favorite_pizzas:

    print(pizza)

print("\n")

# Print a sentence about each pizza.

for pizza in favorite_pizzas:

    print("I really love " + pizza + " pizza!")

print("\nI really love pizza!")
```

Output:

pepperoni

hawaiian

veggie

I really love pepperoni pizza!

I really love hawaiian pizza!

I really love veggie pizza!

I really love pizza!

ANIMALS: commonAnimals = ["Dog", "Cat", "Crocodile"]

```
for animal in commonAnimals:
    print(animal)
print("\n")
# Statement about each animal
for animal in commonAnimals:
    print("A " + animal + "has four legs.")
print("\n")
# Final statement out side of loop
for animal in commonAnimals:
    print("A " + animal + "has four legs.")
print("All of these animals have four legs!")
```

COUNTING TO TWENTY: numbers = list(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

ONE MILLION: millions= list(range(1,1000001))

for million in range:

```
    print(million)
```

SUMMING A MILLION:

```
numbers = list(range(1, 1000001))

print(min(numbers))

print(max(numbers))

print(sum(numbers))
```

Output:

```
1

1000000

500000500000
```

ODD NUMBERS:

```
odd_numbers = []

for value in range(1,11):

    number = value % 2 = 1

    odd_numbers.append(number)

print(odd_numbers)
```

THREES:

```
threes = list(range(3, 31, 3))

for number in threes:

    print(number)
```

Output:

```
3

6

9

12

15
```

18

21

24

27

30

CUBE: cubes = []

```
for number in range(1, 11):
    cube = number**3
    cubes.append(cube)
for cube in cubes:
    print(cube)
```

Output:

1

8

27

64

125

216

343

512

729

1000

CUBE COMPREHENSION: cubes = [number**3 for number in range(1,11)]

```
for cube in cubes:
    print(cube)
```

Output:

```
1
8
27
64
125
216
343
512
729
1000
```

SLICE:

```
cube = [value**3 for value in range(1,11)]
cube_1 = cubes[0:3]
cube_2 = cubes[3:6]
cube_3 = cubes[-3:]
print('The first three items in the list are:'+ str(cube_1))
print('Three items from the middle of the list are:'+ str(cube_2))
print('The last three items in the list are:'+ str(cube_3))
```

MY PIZZAS ,YOUR PIZZAS:

```
favorite_pizzas = ['pepperoni', 'hawaiian', 'veggie']
friend_pizzas = favorite_pizzas[:]
favorite_pizzas.append("meat lover's")
```

```
friend_pizzas.append('pesto')

print("My favorite pizzas are:")

for pizza in favorite_pizzas:

    print("- " + pizza)

print("\nMy friend's favorite pizzas are:")

for pizza in friend_pizzas:

    print("- " + pizza)
```

Output:

```
My favorite pizzas are:

- pepperoni

- hawaiian

- veggie

- meat lover's

My friend's favorite pizzas are:

- pepperoni

- hawaiian

- veggie

- pesto
```

MORE LOOPS:

```
#set list type variable and initialize the elements

myPizza = ['Margarita', 'Capsicum and onion', 'Chicken']

#set variable and initialize the elements

frndPizzas = myPizza[:]

#append value in list variable
```

```
myPizza.append('Corn')

#append value in list variable

frndPizzas.append('paperica')

#print message

print("My pizzas are:")

#set the for loop and print elements of 1st list

for pizza in myPizza:

    print(pizza)

#print message

print("\nFriend's pizzas are:")

#set the for loop and print elements of 2st list

for frndsPizza in frndPizzas:

    print(frndsPizza)
```

Output:

```
My pizzas are:

Margarita
Capsicum and onion
Chicken corn

Friend's pizzas are:

Margarita
Capsicum and onion
Chicken
paperica
```

BUFFET:

```
menu_items = (

    'rockfish sandwich', 'halibut nuggets', 'smoked salmon chowder',
```

```
    'salmon burger', 'crab cakes',
    )
print("You can choose from the following menu items:")
for item in menu_items:
    print("- " + item)
menu_items = (
    'rockfish sandwich', 'halibut nuggets', 'smoked salmon chowder',
    'black cod tips', 'king crab legs',
    )
print("\nOur menu has been updated.")
print("You can now choose from the following items:")
for item in menu_items:
    print("- " + item)
```

Output:

```
You can choose from the following menu items:
- rockfish sandwich
- halibut nuggets
- smoked salmon chowder
- salmon burger
- crab cakes
Our menu has been updated.
You can now choose from the following items:
- rockfish sandwich
- halibut nuggets
```

- smoked salmon chowder

- black cod tips

- king crab legs

CHAPTER#6 TASKS:

PERSON: 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

FAVORITE NUMBERS: 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.
```

GLOSSERY 1:

```
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.
```

GLOSSERY 2:

```
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

RIVERS:

```
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("\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

POLLING: 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?

PEOPLE: # 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.
```

PETS:

```
# 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

FAVORTIE PLACES: 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

FAVORITE NUMBERS: 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

CITIES: 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 nearby.

EXTENSIONS:]

},

{

"cell_type": "code",

"execution_count": 37,

"metadata": {},

"outputs": [],

"source": [

"# update the cities dictionary "

]

},

{

"cell_type": "code",

"execution_count": 38,

"metadata": {},

```
    "outputs": [],
    "source": [
     "cities.update(Los_angles = {\"Country\": \"United States\", \"Population\": \"3,792,621\", \n",
     "                                                \"Fact\" : \" Home to Los Angles Lakers\"})"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 39,
    "metadata": {},
    "outputs": [
     {
      "name": "stdout",
      "output_type": "stream",
      "text": [
       "{'New York City': {'Country': 'United States', 'Population': '8,175,133', 'Fact': 'city that never sleeps'}, 'Chicago': {'Country': 'United States', 'Population': '2,695,598', 'Fact': 'Home to the Bears'}, 'Houston': {'Country': 'United States', 'Population': '2,099,451', 'Fact': 'Oil Hub of North America'}, 'Los_angles': {'Country': 'United States', 'Population': '3,792,621', 'Fact': ' Home to Los Angles Lakers'}}\n"
      ]
     }
    ],
    "source": [
     "print(cities)"
```

```
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
```

```
      "nbconvert_exporter": "python",
      "pygments_lexer": "ipython3",
      "version": "3.7.3"
    }
  },
  "nbformat": 4,
  "nbformat_minor": 2
}
```


CHAPTER#7 TASKS:

RENTAL CAR:

```
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.
```

RESAURENTING SEATING:

```
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.

MULTIPLES OF TEN: 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.

PIZZA TOPPING: 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
```

MOVIE TICKETS:

```
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
```

THREE EXIT:

```
toppings = '\nPlease add your pizza ingredients.'
toppings += "\n(Enter 'quit' when you are finished.) "
#active = True
#while active:
```

```
#       topping = input(toppings)
#       if topping == 'quit':
#               active = False
#       else:
#               print('I will add '+topping.title()+' in your pizza.')
topping = ""
while topping != 'quit':
        topping = input(toppings)
        if topping != 'quit':
                print('I will add ' + topping.title() + ' in your pizza.')
```

INFINITY:

```
x = 1
while x <= 5:
        print(x)
```

DELI:

```
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.

NO PASTRAMI: 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.

DREAM VACATION: Fuegname_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

--- Results ---

Ever would like to visit Death Valley.

Erin would like to visit Iceland.

Eric would like to visit Tierra Del Fuego.
```

CHAPTER#8 TASKS:

MESSAGE:

```
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.
```

FAVORITE BOOK:

```
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.
```

T- SHIRT:

```
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."
```

LARGE SHIRT:

```
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."
```

CITIES:

```
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.
```

CITY NAMES: 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
```

ALBUM: 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'}
```

USER ALBUMS:

```
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!
```

MAGICIANS:

```
def show_magicians(magicians):
    """Print the name of each magician in the list."""
```

```
    for magician in magicians:
        print(magician.title())
magicians = ['harry houdini', 'david blaine', 'teller']
show_magicians(magicians)
```

Output:

Harry Houdini

David Blaine

Teller

GREAT MAGICAINS:

```
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
```

UNCHANGED MAGICIANS:

```
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)
    return magicians
magicians = ['Harry Houdini', 'David Blaine', 'Teller']
show_magicians(magicians)
print("\nGreat magicians:")
great_magicians = make_great(magicians[:])
show_magicians(great_magicians)
print("\nOriginal magicians:")
show_magicians(magicians)
```

Output:

```
Harry Houdini
David Blaine
Teller
Great magicians:
Teller the Great
David Blaine the Great
Harry Houdini the Great
Original magicians:
Harry Houdini
```

David Blaine

Teller

SANDWICHES: def make_sandwich(*items):

```
    """Make a sandwich with the given items."""
    print("\nI'll make you a great sandwich:")
    for item in items:
        print("    ...adding " + item + " to your sandwich.")
    print("Your sandwich is ready!")
make_sandwich('roast beef', 'cheddar cheese', 'lettuce', 'honey dijon')
make_sandwich('turkey', 'apple slices', 'honey mustard')
make_sandwich('peanut butter', 'strawberry jam')
```

Output:

```
I'll make you a great sandwich:
   ...adding roast beef to your sandwich.
   ...adding cheddar cheese to your sandwich.
   ...adding lettuce to your sandwich.
   ...adding honey dijon to your sandwich.
Your sandwich is ready!
I'll make you a great sandwich:
   ...adding turkey to your sandwich.
   ...adding apple slices to your sandwich.
   ...adding honey mustard to your sandwich.
Your sandwich is ready!
```

I'll make you a great sandwich:

...adding peanut butter to your sandwich.

...adding strawberry jam to your sandwich.

Your sandwich is ready!

USER PROFILE:

```
def build_profile(first, last, **user_informa):
        profile = {}
        profile['first_name'] = first
        profile['last_name'] = last
        for k, v in user_informa.items():
                profile[k] = v
        return profile
user_profile = build_profile('zhang','yong',location = 'guang zhou',sex = 'man',height = 175)
print(user_profile)
```

CARS:

```
def make_car(manufacturer, model, **options):
    """Make a dictionary representing a car."""
    car_dict = {
        'manufacturer': manufacturer.title(),
        'model': model.title(),
        }
    for option, value in options.items():
        car_dict[option] = value
    return car_dict
my_outback = make_car('subaru', 'outback', color='blue', tow_package=True)
```

```
print(my_outback)

my_accord = make_car('honda', 'accord', year=1991, color='white',
        headlights='popup')

print(my_accord)
```

Output:

```
{'manufacturer': 'Subaru', 'color': 'blue', 'tow_package': True, 'model': 'Outback'}

{'year': 1991, 'manufacturer': 'Honda', 'color': 'white', 'headlights': 'popup', 'model': 'Accord'}
```

PRINTING MODELS:

```
"""Functions related to printing 3d models."""

def print_models(unprinted_designs, completed_models):
    """
    Simulate printing each design, until there are none left.
    Move each design to completed_models after printing.
    """
    while unprinted_designs:
        current_design = unprinted_designs.pop()
        # Simulate creating a 3d print from the design.
        print("Printing model: " + current_design)
        completed_models.append(current_design)

def show_completed_models(completed_models):
    """Show all the models that were printed."""
    print("\nThe following models have been printed:")
    for completed_model in completed_models:
        print(completed_model)
```

printing_models.py:

```
import printing_functions as pf

unprinted_designs = ['iphone case', 'robot pendant', 'dodecahedron']

completed_models = []

pf.print_models(unprinted_designs, completed_models)

pf.show_completed_models(completed_models)
```

Output:

```
Printing model: dodecahedron

Printing model: robot pendant

Printing model: iphone case

The following models have been printed:

dodecahedron

robot pendant

iphon case
```

IMPORTS: print('')

```
import magician

magicians = ['ab','cd','ef','gh']

magician.show_magicians(magicians)

print('')

from magician import show_magicians

magicians = ['ab','cd','ef','gh']

show_magicians(magicians)

print('')
```

```
from magician import show_magicians as mg

magicians = ['ab','cd','ef','gh']

mg(magicians)

print('')

import magician as m

magicians = ['ab','cd','ef','gh']

m.show_magicians(magicians)

print('')

from magician import *

magicians = ['ab','cd','ef','gh']

show_magicians(magicians)
```

CHAPTER#9 TASKS:

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
    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!
```

THREE RESTAURANTS: 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.
```

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()
```

```
    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!

NUMBER SAVRED: 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
```

LOGIN ATTEMPTS: 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
```

ICE CREAM STAND: 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
```

ADMIN: 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
```

PRIVILEGE: 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

BATTERY UPGRADE: 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.
```

IMPORTED RESTAURANT: 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!
```

IMPORTED ADMIN: 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
```

MULTIPLE MODULES: 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
```

ORDEREDDICT REWRITE: 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.

DICE:

```
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]