Homework Problems

8-1. Message: Write a function called `display_message()` that prints one sentence telling everyone what you are learning about in this chapter. Call the function, and make sure the message displays correctly.

In [1]:
```python
def display_message():
    print("I am learning about functinos in this chapter")

display_message()
```

```
I am learning about functinos in this chapter
```

8-2. Favorite Book: Write a function called `favorite_book()` that accepts one parameter, `title` . The function should print a message, such as *One of my favorite books is Alice in Wonderland.* Call the function, making sure to include a book title as an argument in the function call.

In [2]:
```python
def favorite_book(title):
    print(f"One of my favorite books is {str(title).title()}")

favorite_book("alice in wonderland")
```

```
One of my favorite books is Alice In Wonderland
```

8-3. T-Shirt: Write a function called `make_shirt()` that accepts a size and the text of a message that should be printed on the shirt. The function should print a sentence summarizing the size of the shirt and the message printed on it.

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

In [3]:
```python
def make_shirt(size, text):
    print(f'The shirt of size {str(size).lower()} will have the message "{str

make_shirt('small', 'hello world')
make_shirt(text='hello world', size='small')
```

```
The shirt of size small will have the message "hello world" printed on it.
The shirt of size small will have the message "hello world" printed on it.
```

8-4. Large Shirts: Modify the `make_shirt()` function so that shirts are large by default with a message that reads *I love Python*. Make a large shirt and a medium shirt with the default message, and a shirt of any size with a different message.

```
In [4]: def make_shirt(size='large', text='I love Python'):
            print(f'The shirt of size {str(size).lower()} will have the message "{str

        make_shirt()
        make_shirt('medium')
        make_shirt(text='Python is cool')
```

```
The shirt of size large will have the message "I love Python" printed on it.
The shirt of size medium will have the message "I love Python" printed on i
t.
The shirt of size large will have the message "Python is cool" printed on i
t.
```

8-5. Cities: Write a function called `describe_city()` that accepts the name of a city and its country. The function should print a simple sentence, such as *Reykjavik is in Iceland.* Give the parameter for the country a default value. Call your function for three different cities, at least one of which is not in the default country (pass the correct country as a keyword parameter).

```
In [5]: def describe_city(city, country='united states'):
            print(f"{str(city).title()} is in {str(country).title()}")

        describe_city('annapolis')
        describe_city('severna park')
        describe_city(city='paris', country='france')
```

```
Annapolis is in United States
Severna Park is in United States
Paris is in France
```

8-6. City Names: Write a function called `city_country()` that takes in the name of a city and its country. The function should return a string formatted like this:

```
"Santiago, Chile"
```

Call your function with at least three city-country pairs, and print the values that are returned.

```
In [6]: def city_country(city, country):
            formated_string = f"{str(city).title()}, {str(country).title()}"
            return formated_string

        print(city_country('santiago', 'chile'))
        print(city_country('paris', 'france'))
        print(city_country('annapolis', 'united states'))
```

```
Santiago, Chile
Paris, France
Annapolis, United States
```

8-7. Album: Write a function called `make_album()` that builds a dictionary describing a music album. The function should take in an artist name and an album title, and it should return a dictionary containing these two pieces of information. Use the function to make three dictionaries representing different albums. Print each return value to show that the dictionaries are storing the album information correctly.

Use `None` to add an optional parameter to `make_album()` that alows you to store the number of songs on an album. If the calling line includes a values for the number of songs, add that value to the album's dictionary. Make at least one new function call that includes the number of songs on an album.

In [7]:
```
def make_album(name, title, songs=None):
    album = {'artist_name': str(name).title(), 'album_title': str(title).titl
    if songs:
        album['number_of_songs'] = int(songs)
    return album

print(make_album('olivia rodrigo', 'guts'))
print(make_album('michael jackson', 'thriller', 7))
print(make_album('beatles', 'abbey road'))
```

```
{'artist_name': 'Olivia Rodrigo', 'album_title': 'Guts'}
{'artist_name': 'Michael Jackson', 'album_title': 'Thriller', 'number_of_son
gs': 7}
{'artist_name': 'Beatles', 'album_title': 'Abbey Road'}
```

8-8. User Albums: Start with your program from Exercise 8-7. Write a `while` loop that allows users to enter an album's artist and title. Once you have that information, call `make_album()` with the users' input and print the dictionary that's created. Be sure to include a quit value in the while loop.

In [8]:
```
while True:
    name = input("Please enter the name of the artist: ")
    title = input("Please enter the name of the album title: ")
    print(make_album(name, title))

    quit = input("Would you like to enter another album? (yes or no)")
    if quit == 'no':
        break
```

```
Please enter the name of the artist: Taylor Swift
Please enter the name of the album title: Red
{'artist_name': 'Taylor Swift', 'album_title': 'Red'}
Would you like to enter another album? (yes or no)no
```

8-9. Messages: Make a list containing a series of short text messages. Pass the list to a function called `show_messages()`, which prints each text message.

```
In [9]: def show_messages(messages):
            for message in messages:
                print(message)

        messages = ['Hello.', 'How are you?', 'Good how are you?', "I'm doing alright
        show_messages(messages)
```

```
Hello.
How are you?
Good how are you?
I'm doing alright, thanks for asking.
```

8-10. Sending Messages: Start with a copy of your program from Exercise 8-9. Write a function called `send_messages()` that prints each text message and moves each message to a new list called `sent_messages` as it's printed. After calling the function, print both of your lists to make sure the messages were moved correctly.

```
In [10]: def send_messages(messages, sent_messages):
             while messages:
                 message = messages.pop(0)
                 sent_messages.append(message)
                 print(message)

         messages = ['Hello.', 'How are you?', 'Good how are you?', "I'm doing alright
         sent_messages = []
         send_messages(messages, sent_messages)
         print(messages)
         print(sent_messages)
```

```
Hello.
How are you?
Good how are you?
I'm doing alright, thanks for asking.
[]
['Hello.', 'How are you?', 'Good how are you?', "I'm doing alright, thanks f
or asking."]
```

8-11. Archived Messages: Start with your work from Exercise 8-10. Call the function `send_messages()` with a copy of the list of messages. After calling the function, print both of your lists to show that the original list has retained its messages.

```
In [11]: messages = ['Hello.', 'How are you?', 'Good how are you?', "I'm doing alright
         copy_messages = messages[:]
         sent_messages = []
         send_messages(copy_messages, sent_messages)
         print(messages)
         print(copy_messages)
```

```
Hello.
How are you?
Good how are you?
I'm doing alright, thanks for asking.
['Hello.', 'How are you?', 'Good how are you?', "I'm doing alright, thanks f
or asking."]
[]
```

8-12. Sandwiches: Write a function that accepts a list of items a person wants on a sandwich. The function should have one parameter that collects as many items as the function call provides, and it should print a summary of the sandwich that's being ordered. Call the function three times, using a different number of arguments each time.

```
In [12]: def make_sandwich(*items):
             print(f"Your sandwich will have: ")
             for item in items:
                 print(f"- {str(item)} ")

         make_sandwich('ham', 'cheese')
         make_sandwich('pickles', 'beef', 'lettuce')
         make_sandwich('tomatoes', 'onions', 'cheese', 'mayo')
```

```
Your sandwich will have:
- ham
- cheese
Your sandwich will have:
- pickles
- beef
- lettuce
Your sandwich will have:
- tomatoes
- onions
- cheese
- mayo
```

8-13. User Profile: Start with a copy of the user_profile code above (or on page 149). Build a profile of yourself by calling `build_profile()` , using yourfirst and last names and three other key-value pairs that describe you.

In [13]:
```python
def build_profile(first, last, **user_info):
    """Build a dictionary containing everything we know about a user."""
    user_info['first_name'] = first
    user_info['last_name'] = last
    return user_info

user_profile = build_profile('andrew', 'bernas', major='robotics', home='cali

print(user_profile)
```

```
{'major': 'robotics', 'home': 'california', 'company': 26, 'first_name': 'an
drew', 'last_name': 'bernas'}
```

8-14. Cars: Write a function that stores information about a car in a dictionary. The function should always receive a manufacturer and a model name. It should then accept an arbitrary number of keyword arguments. Call the function with the required information and two other name-value pairs, such as a color or an optional feature. Your function should work for a call like this one:

```
car = make_car('subaru', 'outback', color='blue', tow_package=True)
```

Print the dictionary that's returned to make sure all the information was stored correctly.

In [14]:
```python
def make_car(manufacturer, model, **car_info):
    car_info['manufacturer'] = str(manufacturer)
    car_info['model'] = str(model)
    return car_info

car = make_car('subaru', 'outback', color='blue', tow_package=True)
print(car)
```

```
{'color': 'blue', 'tow_package': True, 'manufacturer': 'subaru', 'model': 'o
utback'}
```

8-15. Printing Models: Move the functions below into a separate file called *printing_functions.py* Write an `import` statement at the top of the code below so the code uses the module's functions.

In [15]:
```
#TODO: Move to printing_functions.py
# def print_models(unprinted_designs, completed_models):
#     """
#     Simulate printing each design, until none are left.
#     Move each design to completed_models after printing.
#     """
#     while unprinted_designs:
#         current_design = unprinted_designs.pop()
#         print(f"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)

from printing_functions import *

unprinted_designs = ['phone case', 'robot pendant', 'dodecahedron']
completed_models = []

print_models(unprinted_designs, completed_models)
show_completed_models(completed_models)
```

```
Printing model: dodecahedron
Printing model: robot pendant
Printing model: phone case

The following models have been printed:
dodecahedron
robot pendant
phone case
```

8-16.Imports: Create a file called hello.py with a function that prints a greeting. Import the function using each of these approaches and verify it works by calling the function (you should see 6 copies of the greeting when you run the code)

In [16]:
```
import hello

hello.greeting()

from hello import greeting

greeting()

from hello import greeting as g

g()

import hello as h

h.greeting()

from hello import *

greeting()
```

```
Hello!
Hello!
Hello!
Hello!
Hello!
```