Exercises chapter 8: Functions

Saul SL

June 2023

1 Exercise 8-1 Message

Write a function called display_message() that prints one sen- tence telling everyone what you are learning about in this chapter. Call the function, and make sure the message displays correctly.

```
def display_message(message):
    print(message)


display_message("I'm learing functions")
```

2 Exercise 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.

```
def favorite_book(title):
    print(f"One of my favorite books is {title.title()}")


favorite_book('The unwinding of a miracle')
```

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

```
def make_shirt(size, text):
    print(f"A shirt of size {size} was made")
    print(f"The message, '{text}' was printed")


make_shirt('small', 'what is up?')
make_shirt(text='Yo!', size='large')
```

4 Exercise 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.

```python
def make_shirt(size='large', text='I love Python'):
    print(f"A shirt of size {size} was made")
    print(f"The message, '{text}' was printed")


make_shirt()
```

5 Exercise 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.

```python
def describe_city(city, country='Bolivia'):
    print(f'{city.title()} is in {country.title()}')


describe_city('la paz')
describe_city(city='tokyo', country='japan')
describe_city('cochabamba')
```

6 Exercise 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.

```python
def city_country(city, country):
    return (f'"{city.title()}, {country.title()}"')


cc1 = city_country('manila', 'philipines')
cc2 = city_country('athens', 'greece')
cc3 = city_country('washington', 'usa')
cities = [cc1, cc2, cc3]
for places in cities:
    print(places)
```

7 Exercise 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 allows you to store the number of songs on an album. If the calling line includes a value 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.

```python
def make_album(artist, album, tracks=None):
    if tracks:
        return {'artist': artist.title(),
                'album': album.title(),
                'N tracks': tracks}
    else:
        return {'artist': artist.title(),
```

```
                'album': album.title()}


alb1 = make_album('bob marley', 'exodus')
alb2 = make_album('placebo', 'black market music', 13)
alb3 = make_album('silvio rodriguez', 'dominguez', 10)
albums = [alb1, alb2, alb3]

for album in albums:
    print(album)
```

8 Exercise 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 user's input and print the dictionary that's created. Be sure to include a quit value in the while loop.

```
import re
print("Please enter the name of a band/artist and an album")
print("Optionally, enter the number of tracks")
print("type 'q' to exit")

while True:
    artist = input('Artist: ')
    album = input('Album: ')
    tracks = input('N tracks: ')

    if artist == 'q' or album == 'q' or tracks == 'q':
        break

    if tracks is not None:
        pattern = r'\b[0-9]+\b'
        match = re.search(tracks, pattern)
        if match:
            print(make_album(artist, album, tracks))
        else:
            print(make_album(artist, album))
    else:
        print(make_album(artist, album))
```

9 Exercise 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.

```
def show_messages(msg_list):
    for msg in msg_list:
        print(msg)


my_msgs = ['Hello good morning',
           'Welcome back',
           'Would you like a report']

show_messages(my_msgs)
```

10 Exercise 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.

```
def send_messages(msg_list):
    sent_msgs = []
    while msg_list:
        msg = msg_list.pop()
        sent_msgs.append(msg)

    if sent_msgs:
        return sent_msgs


sent_msgs = send_messages(my_msgs)
print(my_msgs)
print("----------")
print(sent_msgs)
```

11 Exercise 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.

```
my_msgs = ['Hello good morning',
           'Welcome back',
           'Would you like a report']

sent_msgs = send_messages(my_msgs[:])  # A copy of the list is passed
print(my_msgs)
print("----------")
print(sent_msgs)
```

12 Exercise 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 num- ber of arguments each time.

```
def sandwich_items(*items):
    if items:
        if len(items) == 1:
            print("This is the sandwich ingredient:")
        elif len(items) > 1:
            print("These are the sandwich ingredients:")

        for item in items:
            print(f"- {item.title()}")


ingredients = ['cheese', 'tomato', 'ham', 'onions', 'lettuce']
sandwich_items('cheese', 'tomato', 'ham', 'onions', 'lettuce')
sandwich_items('tomato')
sandwich_items(ingredients)  # Doesn't work with a list
```

13 Exercise 8-13 User Profile

Start with a copy of user_profile.py from page 149. Build a profile of yourself by calling build_profile(), using your first and last names and three other key-value pairs that describe you.

```
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('saul', 'sotomayor',
                             location='la paz',
                             field='plant biology',
                             interest='bioinformatics')
print(user_profile)
```

14 Exercise 8-14 Cars

Write a function that stores information about a car in a diction- ary. The function should always receive a manufacturer and a model name. It should then accept an arbitrary number of keyword arguments. Call the func- tion 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.

```
def make_car(brand, model, **car_info):
    """Returns a dictionary with car info"""
    car_info['brand'] = brand
    car_info['model'] = model
    return car_info


icar = make_car('bmw', 'space', year=2019, millage=25_000)
print(icar)
```

15 Exercise 8-15 Printing Models

Put the functions for the example printing_models.py in a separate file called printing_functions.py. Write an import statement at the top of printing_models.py, and modify the file to use the imported functions.

```
from printing_functions import print_models, show_completed_models

unprinted_designs = ['phone case', 'robot pendant', 'dodecahedron']
completed_models = []
print_models(unprinted_designs, completed_models)
show_completed_models(completed_models)
```

16 Exercise 8-16 Imports

Using a program you wrote that has one function in it, store that function in a separate file. Import the function into your main program file, and call the function using each of these approaches

```
import module_name
from module_name import function_name
from module_name import function_name as fn
import module_name as mn
from module_name import *
```

```
import send_messages

to_send = ['Hi', 'Bye', 'Como esta?', 'Ciao']
sent_msgs = send_messages.send_messages(to_send)

# -------------------------
from send_messages import send_messages as sm

to_send = ['Hola', 'Y Adios', 'Welcome back', 'Shutting down']
sent_msgs_2 = sm(to_send[:])

# -------------------------
import send_messages as sms

sent_msgs_3 = sms.send_messages(to_send[:])
```