IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

Chapter 3

Task 3-1: Store the names of a few of your friends in a list called names. Print each person’s name by accessing each element in the list, one at a time.

Solution:

```
Friend_Names=['hameed','salman','karim','sauood']

print(Friend_Names)
```

Output:

```
['hameed', 'salman', 'karim', 'sauood']
```

Task 3-2: Start with the list you used in Exercise 3-1, but instead of just printing each person’s name, print a message to them. The text of each message should be the same, but each message should be personalized with the person’s name.

Solution:

```
Friend_Names=['hameed','salman','karim','sauood']

message="My room mate is " + Friend_Names[0].title() + "." + Friend_Names[1].title() + " and " + Friend_Names[2].title() + " are my Classmates." + Friend_Names[3].title() + " is my best Friend."

print(message)
```

Output:

```
My room mate is Hameed.Salman and Karim are my Classmates.Sauood is my best Friend.
```

Task 3-3: Think of your favorite mode of transportation, such as a

motorcycle or a car, and make a list that stores several examples. Use your list

to print a series of statements about these items, such as “I would like to own a

Honda motorcycle.

Solution:

```
Subjects=['DLD','DS','OB','MPL']

MS=Subjects[3] + " is my best  subject and this subject teaches us Sir Faheem." + Subjects[0] + "," + Subjects[1] + " and " + Subjects[2] + " abbreviate, ' Digital and Logical Data', 'Data Structure' and 'Organizational Behaviour' respectively."
```

```
print(MS)
```

Output:

```
MPL is my best  subject and this subject teaches us Sir Faheem.DLD,DS and OB abbreviate, ' Digital and Logical Data', 'Data Structure' and 'Organizational Behaviour' respectively.
```

Task 3-4: If you could invite anyone, living or deceased, to dinner, who

would you invite? Make a list that includes at least three people you’d like to

invite to dinner. Then use your list to print a message to each person, inviting

them to dinner.

Solution:

```
list=['Doctor','Teacher','Engineer']

MSG=" I invite you " + list[0] + " to Dinner at 2:00PM in Peshawar Thanks." + "\n I invite you " + list[1] + " to Dinner at 2:00PM in Peshawar Thanks." + "\n I invite you " + list[2] + " to Dinner at 2:00PM in Peshawar Thanks."

print(MSG)
```

Output:

```
I invite you Doctor to Dinner at 2:00PM in Peshawar Thanks.

 I invite you Teacher to Dinner at 2:00PM in Peshawar Thanks.

 I invite you Engineer to Dinner at 2:00PM in Peshawar Thanks.
```

Task 3-5: You just heard that one of your guests can't make the dinner, so you need to send out a new set of invitations. You'll have to think of someone else to invite.

- Start with your program from Exercise 3-4. Add a print statement at the end of your program stating the name of the guest who can't make it.
- Modify your list, replacing the name of the guest who can't make it with the name of the new person you are inviting.
- Print a second set of invitation messages, one for each person who is still in your list.

Solution:

```
list=['Doctor','Teacher','Engineer']

list[1]='Police'

print(list)

MSG=" I invite you " + list[0] + " to Dinner at 2:00PM in Peshawar Thanks." + "\n I invite you " + list[1] + " to Dinner at 2:00PM in PeshawarThanks." + "\n I invite you " + list[2] + " to Dinner at 2:00PM in Peshawar Thanks."

print(MSG)
```

Output:

```
['Doctor', 'Police', 'Engineer']

 I invite you Doctor to Dinner at 2:00PM in Peshawar Thanks.

 I invite you Police to Dinner at 2:00PM in Peshawar Thanks.

 I invite you Engineer to Dinner at 2:00PM in Peshawar Thanks.
```

Task 3-6: You just found a bigger dinner table, so now more space is

available. Think of three more guests to invite to dinner.

• Start with your program from Exercise 3-4 or Exercise 3-5. Add a print statement to the end of your program informing people that you found a

bigger dinner table.

- Use insert() to add one new guest to the beginning of your list.
- Use insert() to add one new guest to the middle of your list.
- Use append() to add one new guest to the end of your list.
- Print a new set of invitation messages, one for each person in your list.

Solution:

```
list=['Doctor','Teacher','Engineer']

list.insert(0,'Businessman')

print(list)

list.insert(2,'Watchman')

print(list)

list.append('Gardener')

print(list)
```

```
MSG=" I invite you " + list[0] + " to Dinner at 2:00PM in Peshawar Thanks." + "\n I invite you " + list[1] + " to Dinner at 2:00PM in Peshawar Thanks." + "\n I invite you " + list[2] + " to Dinner at 2:00PM in Peshawar Thanks." + "\n I invite you " + list[3] + " to Dinner at 2:00PM in Peshawar Thanks." + "\n I invite you " + list[4] + " to Dinner at 2:00PM in Peshawar Thanks." + "\n I invite you " + list[5] + " to Dinner at 2:00PM in Peshawar Thanks."

print(MSG)
```

Output:

```
['Businessman', 'Doctor', 'Teacher', 'Engineer']

['Businessman', 'Doctor', 'Watchman', 'Teacher', 'Engineer']

['Businessman', 'Doctor', 'Watchman', 'Teacher', 'Engineer', 'Gardener']

 I invite you Businessman to Dinner at 2:00PM in Peshawar Thanks.

 I invite you Doctor to Dinner at 2:00PM in Peshawar Thanks.

 I invite you Watchman to Dinner at 2:00PM in Peshawar Thanks.

 I invite you Teacher to Dinner at 2:00PM in Peshawar Thanks.

 I invite you Engineer to Dinner at 2:00PM in Peshawar Thanks.

 I invite you Gardener to Dinner at 2:00PM in Peshawar Thanks.
```

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan   ID# 14486

Dept: BS (CS)   Final Term lab Exam   Subject: Modren programming language (MPL)

Task 3-7: You just found out that your new dinner table won't arrive in time for the dinner, and you have space for only two guests.

- Start with your program from Exercise 3-6. Add a new line that prints a message saying that you can invite only two people for dinner.
- Use pop() to remove guests from your list one at a time until only two names remain in your list. Each time you pop a name from your list, print a message to that person letting them know you're sorry you can't invite them to dinner.
- Print a message to each of the two people still on your list, letting them know they're still invited.
- Use del to remove the last two names from your list, so you have an empty list. Print your list to make sure you actually have an empty list at the end of your program.

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan

ID# 14486

Dept: BS (CS)

Final Term lab Exam

Subject: Modren programming language (MPL)

Solution:

```
list=['Doctor','Teacher','Engineer']
list.insert(0,'Businessman')
print(list)
list.insert(2,'Watchman')
print(list)
list.append('Gardener')
print(list)
line="I can invite only two people for Dinner."
print(line)
print(list[5] +   " I am sorry, I can't invite you to Dinner.")
popped_list =list.pop()
print(list)
print(popped_list)
print(list[4] +   " I am sorry, I can't invite you to Dinner.")

popped_list =list.pop()

print(list)

print(popped_list)

print(list[3] +   " I am sorry, I can't invite you to Dinner.")

popped_list=list.pop()

print(list)

print(popped_list)

print(list[2] +   " I am sorry, I can't invite you to Dinner.")

popped_list=list.pop()

print(list)

print(popped_list)

MSG=" I invite you " + list[0] + " to Dinner at 2:00PM in Peshawar Thanks." + "\n I invite you " + list[1] + " to Dinner at 2:00PM in Peshawar Thanks."

print(MSG)
```

Output:

```
['Businessman', 'Doctor', 'Teacher', 'Engineer']

['Businessman', 'Doctor', 'Watchman', 'Teacher', 'Engineer']

['Businessman', 'Doctor', 'Watchman', 'Teacher', 'Engineer', 'Gardener']

I can invite only two people for Dinner.

Gardener I am sorry, I can't invite you to Dinner.

['Businessman', 'Doctor', 'Watchman', 'Teacher', 'Engineer']

Gardener

Engineer I am sorry, I can't invite you to Dinner.

['Businessman', 'Doctor', 'Watchman', 'Teacher']

Engineer

Teacher I am sorry, I can't invite you to Dinner.

['Businessman', 'Doctor', 'Watchman']

Teacher
```

```
Watchman I am sorry, I can't invite you to Dinner.

['Businessman', 'Doctor']

Watchman

 I invite you Businessman to Dinner at 2:00PM in Peshawar Thanks.

 I invite you Doctor to Dinner at 2:00PM in Peshawar Thanks.
```

Task 3-8: Think of at least five places in the world you'd like to visit.

- Store the locations in a list. Make sure the list is not in alphabetical order.
- Print your list in its original order. Don't worry about printing the list neatly, just print it as a raw Python list.
- Use sorted() to print your list in alphabetical order without modifying the actual list.
- Show that your list is still in its original order by printing it.
- Use sorted() to print your list in reverse alphabetical order without changing

the order of the original list.

• Show that your list is still in its original order by printing it again.

• Use reverse() to change the order of your list. Print the list to show that its order has changed.

• Use reverse() to change the order of your list again. Print the list to show it’s back to its original order.

• Use sort() to change your list so it’s stored in alphabetical order. Print the list to show that its order has been changed.

• Use sort() to change your list so it’s stored in reverse alphabetical order. Print the list to show that its order has changed.

Solution:

```
Places=['Japan','India','Pakistan','America']

print(Places)

print("\nHere is the sorted list.")
print(sorted(Places))

print("\nHere is the original list.")

print(Places)

print("\nHere is the sorted list.")

print(sorted(Places,reverse=True)

print("\nHere is the original list again.")

print(Places)

print("\nHere the list in reverse order.")

Places.reverse()

print(Places)

print("\nHere the list in reverse order again.")

Places.reverse()

print(Places)

print("\nHere is the sort list.")
```

```
Places.sort()

print(Places)

print("\nHere is the sort list in reverse.")

Places.sort(reverse=True)

print(Places)
```

Output:

```
['Japan', 'India', 'Pakistan', 'America']


Here is the sorted list.

['America', 'India', 'Japan', 'Pakistan']


Here is the original list.

['Japan', 'India', 'Pakistan', 'America']
```

```
Here is the sorted list.

['Pakistan', 'Japan', 'India', 'America']


Here is the original list again.

['Japan', 'India', 'Pakistan', 'America']


Here the list in reverse order.

['America', 'Pakistan', 'India', 'Japan']


Here the list in reverse order again.

['Japan', 'India', 'Pakistan', 'America']

Here is the sort list.

['America', 'India', 'Japan', 'Pakistan']

Here is the sort list in reverse.
```

```
['Pakistan', 'Japan', 'India', 'America']
```

Task 3-9: Working with one of the programs from Exercises 3-4 through 3-7 (page 46), use len() to print a message indicating the number of people you are inviting to dinner.

Solution:

```
list=['Businessman', 'Doctor', 'Watchman', 'Teacher', 'Engineer', 'Gardener']

print(len(list))

msge="The number of people I invited to Dinner are " + str(len(list)) + "."

print(msge)
```

Output:

```
6

The number of people I invited to Dinner are 6.
```

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

Task 3-10: If you haven't received an index error in one of your programs yet, try to make one happen. Change an index in one of your Programs to produce an index error. Make sure you correct the error before closing the program.

Solution:

```
list=['Businessman', 'Doctor', 'Watchman']

print(list[3])
```

Correction of error:

```
list=['Businessman', 'Doctor', 'Watchman']

print(list[2])
```

Output:

```
Traceback (most recent call last):

  File "C:/Users/X61s/AppData/Local/Programs/Python/8.py", line 2, in <module>

    print(list[3])

IndexError: list index out of range
```

Correction of error:

```
Watchman
```

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

Chapter 4

Task 4-1: Think of at least three kinds of your favorite pizza. Store these pizza names in a list, and then use a for loop to print the name of each pizza.

• Modify your for loop to print a sentence using the name of the pizza instead of printing just the name of the pizza. For each pizza you should have one line of output containing a simple statement like I like pepperoni pizza.

• Add a line at the end of your program, outside the for loop, that states how much you like pizza. The output should consist of three or more lines about the kinds of pizza you like and then an additional sentence, such as I really love pizza!

Solution:

```
Pizzas=[‘california’,’broadway’,’penny’]

for pizza in pizzas:
    print(pizza)

    msg=pizza.title() + " is very sweet."

    print(msg)

print("I like to go to eat pizza with my friends.")
```

Output:

```
california

California is very sweet.

broadway

Broadway is very sweet.

penny

Penny is very sweet.

I like to go to eat pizza with my friends.
```

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

Task 4-2: Think of at least three different animals that have a common characteristic.

Store the names of these animals in a list, and then use a for loop to

print out the name of each animal.

- Modify your program to print a statement about each animal, such as

A dog would make a great pet.

- Add a line at the end of your program stating what these animals have in

common. You could print a sentence such as Any of these animals would

make a great pet!

Solution:

```
animals=['cow','goat','horse']

for animal in animals:

    print(animal)

statement="It is known as the " + animals[0].title() + " is our Mother  in India." + "\nThe horn of the " + animals[1].title() + " is used to make spoons." + "\n" + animals[2].title() + " is a very powerful animal."

print(statement)
print("All these animals have four legs, they eat grass and they give milk.")
```

Output:

```
cow

goat

horse

It is known as the Cow is our Mother  in India.

The horn of the Goat is used to make spoons.

Horse is a very powerful animal.

All these animals have four legs, they eat grass and they give milk.
```

Task 4-3: Use a for loop to print the numbers from 1 to 20,

inclusive.

Solution:

```
for value in range(1,21):
   print(value)
```

Output:

```
1

2

3

4

5

6

7

8

9

10

11

12
```

```
13

14

15

16

17

18

19

20
```

Task 4-4: Make a list of the numbers from one to one million, and then use a for loop to print the numbers. (If the output is taking too long, stop it by pressing ctrl-C or by closing the output window.)

Solution:

```
numbers=list(range(1,1000000))
```

```
print(numbers)
```

Output:

Its output is too long.

Task 4-5: Make a list of the numbers from one to one million,

and then use min() and max() to make sure your list actually starts at one and

ends at one million. Also, use the sum() function to see how quickly Python can

add a million numbers.

Solution:

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

>>> min(numbers)

1

>>> max(numbers)

1000000
```

```
>>> sum(numbers)

500000500000
```

Task 4-6: Use the third argument of the range() function to make a list

of the odd numbers from 1 to 20. Use a for loop to print each number.

Solution:

```
odd_numbers=list(range(1,20,2))

for value in range(1,20,2):

   odd_numbers.append(value+2)

print(odd_numbers)
```

Output:

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21]
```

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

Task 4-7: Make a list of the multiples of 3 from 3 to 30. Use a for loop to print the numbers in your list.

Solution:

```
list=[]
for value in range(1,11):
   list.append(value*3)
print(list)
```

Output:

```
[3, 6, 9, 12, 15, 18, 21, 24, 27, 30]
```

Task 4-8: A number raised to the third power is called a cube. For example, the cube of 2 is written as 2**3 in Python. Make a list of the first 10 cubes (that is, the cube of each integer from 1 through 10), and use a for loop to print out the value of each cube.

Solution:

```
cubes=[]

for value in range(1,11):

    cubes.append(value**3)

print(cubes)
```

Output:

```
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

Task 4-9: Use a list comprehension to generate a list of the first 10 cubes.

Solution:

```
cubes=[value**3 for value in range(1,11)]

print(cubes)
```

Output:

```
[1, 8, 27, 64, 125, 216, 343, 512, 729, 1000]
```

Task 4-10: Using one of the programs you wrote in this chapter, add several lines to the end of the program that do the following:

• Print the message, The first three items in the list are:. Then use a slice to print the first three items from that program's list.

• Print the message, Three items from the middle of the list are:. Use a slice to print three items from the middle of the list.

• Print the message, The last three items in the list are:. Use a slice to print the last three items in the list.

Solution:

```
list=['goat','cow','horse','cat','dog']

print("The first three items in the list are; ")

print(list[0:3])
print("Three items from the middle of the list are; ")

print(list[1:4])

print("The last three items in the list are; ")

print(list[2:])
```

Output:

```
The first three items in the list are;

['goat', 'cow', 'horse']

Three items from the middle of the list are;

['cow', 'horse', 'cat']

The last three items in the list are;

['horse', 'cat', 'dog']
```

Task 4-11: Start with your program from Exercise 4-1

(page 60). Make a copy of the list of pizzas, and call it friend_pizzas.

Then, do the following:

- Add a new pizza to the original list.
- Add a different pizza to the list friend_pizzas.
- Prove that you have two separate lists. Print the message, My favorite pizzas are:, and then use a for loop to print the first list. Print the message, My friend’s favorite pizzas are:, and then use a for loop to print the second list. Make sure each new pizza is stored in the appropriate list.

Solution:

```
pizzas=['california','broadway','penny']

print("My pizzas are; ")

print(pizzas)

friend_pizzas=pizzas[:]

print("My friend pizzas are; ")

print(friend_pizzas)
```

```
pizzas.append('hut')

friend_pizzas.append('telepizza')

print("My pizzas are; ")

print(pizzas)

print("My friend pizzas are; ")

print(friend_pizzas)

print("My favourite pizzas are; ")

for pizza in pizzas:

    print(pizza)

print("My friend's favourite pizzas are; ")

for friend_pizza in friend_pizzas:

    print(friend_pizza)
```

Output:

```
My pizzas are;
```

```
['california', 'broadway', 'penny']
My friend pizzas are;
['california', 'broadway', 'penny']
My pizzas are;
['california', 'broadway', 'penny', 'hut']
My friend pizzas are;
['california', 'broadway', 'penny', 'telepizza']
My favourite pizzas are;
california
broadway
penny
hut
My friend's favourite pizzas are;
california
```

broadway

penny

telepizza

Task 4-12: All versions of foods.py in this section have avoided using for loops when printing to save space. Choose a version of foods.py, and write two for loops to print each list of foods.

Solution:

```
foods = ['pizza', 'falafel', 'carrot cake']
for food in foods:
   print(food)
foods=['cake','biscuit','sweet']
for food in foods:
   print(food)
```

Output:

```
pizza

falafel

carrot cake

cake

biscuit

sweet
```

Task 4-13: A buffet-style restaurant offers only five basic foods. Think of five simple foods, and store them in a tuple.

• Use a for loop to print each food the restaurant offers.

• Try to modify one of the items, and make sure that Python rejects the change.

• The restaurant changes its menu, replacing two of the items with different foods. Add a block of code that rewrites the tuple, and then use a for loop to print each of the items on the revised menu.

Solution:

```
foods=('chiken','berger','pizza','rice','kabab')

for food in foods:

   print(food)

foods[1]='potato'

print("Modified Foods: ")

foods=('chiken','berger','meat','lobia','rice')

for food in foods:

   print(food)
```

Output:

```
chiken

berger

pizza
```

```
rice

kabab

Traceback (most recent call last):

  File "C:/Users/X61s/AppData/Local/Programs/Python/Python37-32/Task 4-13.py", line 4, in <module>

    foods[2]='potato'

TypeError: 'tuple' object does not support item assignment

Modified Foods:

chiken

berger

meat

lobia

rice
```

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan | ID# 14486

Dept: BS (CS) | Final Term lab Exam | Subject: Modren programming language (MPL)

chapter 5

Task 5-1: Write a series of conditional tests. Print a statement describing each test and your prediction for the results of each test. Your code should look something like this:

```
car = 'subaru'
print("Is car == 'subaru'? I predict True.")
print(car == 'subaru')
print("\nIs car == 'audi'? I predict False.")
print(car == 'audi')
```

- Look closely at your results, and make sure you understand why each line evaluates to True or False.
- Create at least 10 tests. Have at least 5 tests evaluate to True and another 5 tests evaluate to False.

Solution:

```
book="mathematics"

print("If book=='mathematics',So its true and I like this.")

print(book=='mathematics')



print("If book=='science',So its false and I don't like this.")
print(book=='science')


print("\n")
food='mango'
print(food=='mango')
print(food=='orange')
car='toyota'
print(car=='toyota')
print(car=='civic')
mobile='samsung'
print(mobile=='samsung')
print(mobile=='oppo')
fan='superasia'
print(fan=='superasia')
print(fan=='millat')

battery='osaka'

print(battery=='osaka')

print(battery=='power')
```

Output:

```
If book=='mathematics',So its true and I like this.

True

If book=='science',So its false and I don't like this.

False

True

False

True

False

True
```

```
False

True

False

True

False
```

Task 5-2: You don’t have to limit the number of tests you create to 10. If you want to try more comparisons, write more tests and add them to *conditional_tests.py*. Have at least one True and one False result for each of the following:

- Tests for equality and inequality with strings
- Tests using the lower() function
- Numerical tests involving equality and inequality, greater than and less than, greater than or equal to, and less than or equal to
- Tests using the and keyword and the or keyword
- Test whether an item is in a list
- Test whether an item is not in a list.

Solution:

```
team='pakistan'

print(team=='pakistan')
print(team=='australia')

college='islamia'

print(college!='icms')

print(college!='islamia')

print("\n")

university='IqRa'

print(university=='iqra')

print(university.lower()=='iqra')

print("\n")

age=20

print(age==20)

print(age==18)

answer=38

print(answer!=30)
print(answer!=38)

counter=140

print(counter>100)

print(counter>150)

number=45

print(number<50)

print(number<40)

id=70

print(id>=60)

print(id>=80)

account=200

print(account<=220)

print(account<=190)

print("\n")
```

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

```
no_1=14

no_2=20

print(no_1<=22 and no_2>=15)

print(no_1>=30 and no_2<=12)

id_0=350

id_00=150

 print(id_0>=400 or id_00<=180)

print(id_0<=300 or id_00>=190)

print("\n")

city=['peshawar','karachi','islamabad','quetta']

print('islamabad' in city)

print('swat' in city)

print("\n")

village=['panjpir','zarobi','anbar','lahore']
```

```
print('gohati' not in village)

print('panjpir' not in village)
```

Output:

```
True

False

True

False

False

True

True

False

True

False

True
```

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan

ID# 14486

Dept: BS (CS)

Final Term lab Exam

Subject: Modren programming language (MPL)

False

True

False

True

False

True

False

True

False

True

False

True

False

```
True

False
```

Task 5-3: Imagine an alien was just shot down in a game. Create a variable called alien_color and assign it a value of 'green', 'yellow', or 'red'.

- Write an if statement to test whether the alien's color is green. If it is, print a message that the player just earned 5 points.
- Write one version of this program that passes the if test and another that fails. (The version that fails will have no output.)

Solution:

```
alien_color='green'

if alien_color=='green':

    print("The player just earned 5 points.")

alien_color='green'

if alien_color=='yellow':

    print("The player just earned 50 points.")
```

Output:

```
The player just earned 5 points.
```

No output.

Task 5-4: Choose a color for an alien as you did in Exercise 5-3, and write an if-else chain.

- If the alien’s color is green, print a statement that the player just earned 5 points for shooting the alien.
- If the alien’s color isn’t green, print a statement that the player just earned 10 points.
- Write one version of this program that runs the if block and another that runs the else block.

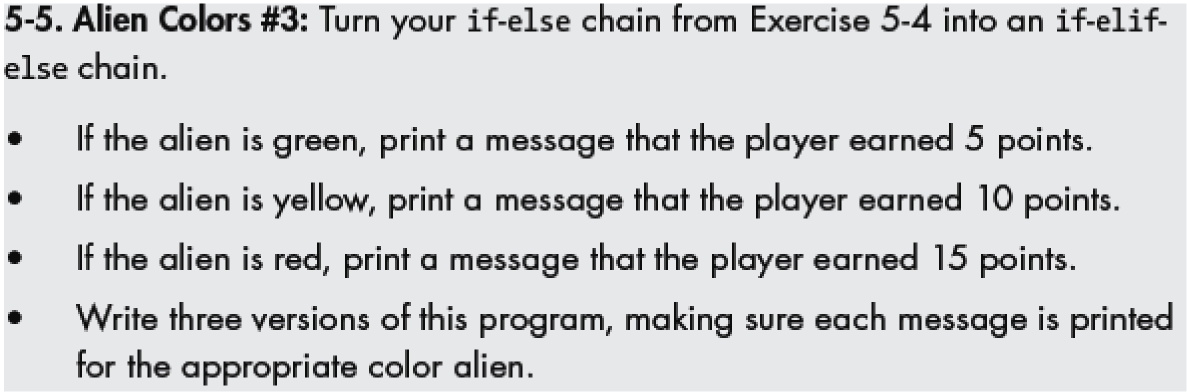
Solution:

```
alien_color='green'

if alien_color=='green':

    print("The player just earned 5 points for shooting the alien.")

else:

    print("The player just earned 10 points for shooting the alien.")

if alien_color=='yellow':

    print("The player just earned 5 points for shooting the alien.")

else:
    print("The player just earned 10 points for shooting the alien.")
```

Output:

```
The player just earned 5 points for shooting the alien.

The player just earned 10 points for shooting the alien.
```

Task:



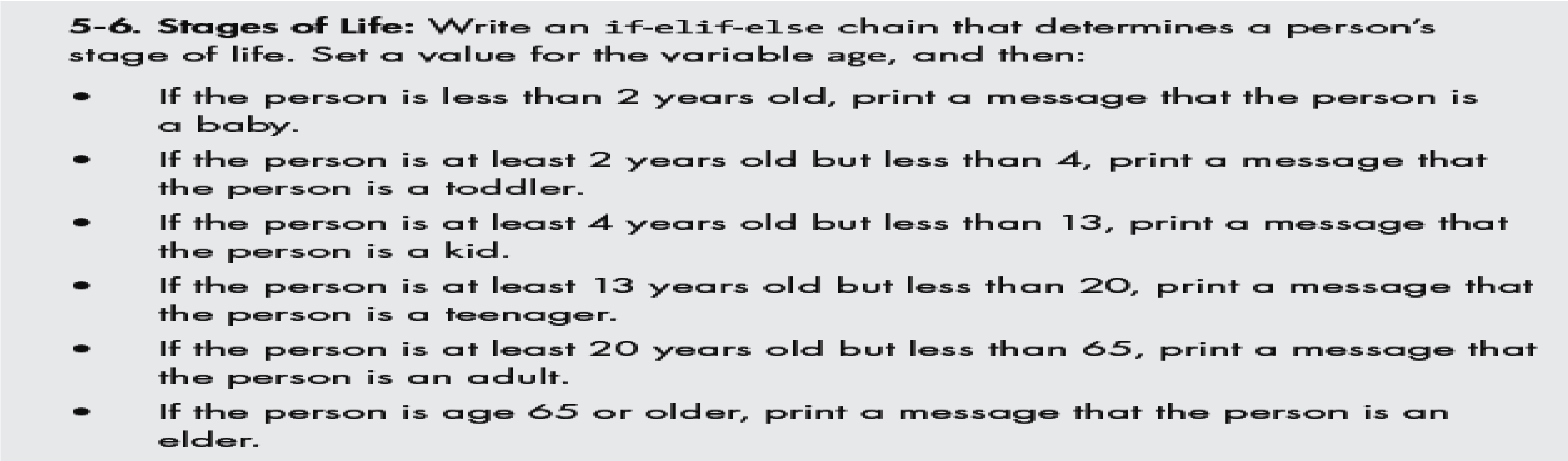
5-5. Alien Colors #3: Turn your if-else chain from Exercise 5-4 into an if-elif-else chain.

- If the alien is green, print a message that the player earned 5 points.
- If the alien is yellow, print a message that the player earned 10 points.
- If the alien is red, print a message that the player earned 15 points.
- Write three versions of this program, making sure each message is printed for the appropriate color alien.

Solution:

```
alien_color='green'

if alien_color=='green':
    print("The player earned 5 points.")
elif alien_color=='yellow':
    print("The player earned 10 points.")
else:
    print("The player earned 15 points.")
alien_color='yellow'
if alien_color=='green':
    print("The player earned 5 points.")
elif alien_color=='yellow':
    print("The player earned 10 points.")
else:
    print("The player earned 15 points.")
alien_color='red'
if alien_color=='green':
    print("The player earned 5 points.")

elif alien_color=='yellow':

    print("The player earned 10 points.")

else:

    print("The player earned 15 points.")
```

Task:



5-6. Stages of Life: Write an if-elif-else chain that determines a person's stage of life. Set a value for the variable age, and then:

- If the person is less than 2 years old, print a message that the person is a baby.
- If the person is at least 2 years old but less than 4, print a message that the person is a toddler.
- If the person is at least 4 years old but less than 13, print a message that the person is a kid.
- If the person is at least 13 years old but less than 20, print a message that the person is a teenager.
- If the person is at least 20 years old but less than 65, print a message that the person is an adult.
- If the person is age 65 or older, print a message that the person is an elder.

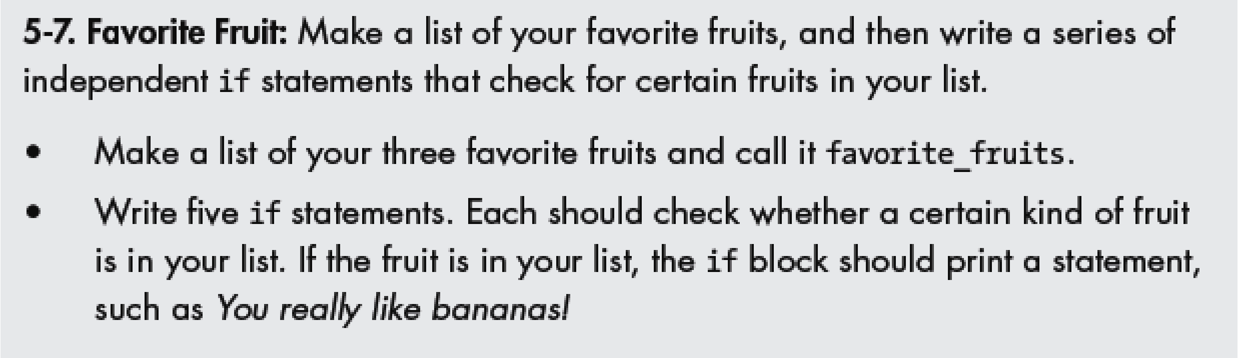
Solution:

```
age=15

if age < 2:

   print("The person is a baby.")

elif age < 4:

   print("The person is a toddler.")

elif age < 13:

   print("The person is a kid.")

elif age < 20:

   print("The person is a teenager.")

elif age < 65:

   print("The person is an adult.")

else:

   print("The person is an elder.")
```

Output:

The person is a teenager.

Task:



5-7. Favorite Fruit: Make a list of your favorite fruits, and then write a series of independent `if` statements that check for certain fruits in your list.

- Make a list of your three favorite fruits and call it `favorite_fruits`.
- Write five `if` statements. Each should check whether a certain kind of fruit is in your list. If the fruit is in your list, the `if` block should print a statement, such as *You really like bananas!*

Solution:

```
favourite_fruits=['banana','orange','apple']

if 'apple' in favourite_fruits:

    print("I like apple juice.")

if 'peach' in favourite_fruits:

    print("peach is very sweet.")

if 'orange' in favourite_fruits:
    print("I like eat orange.")

if 'strawberry' in favourite_fruits:

    print("strawberry is blood like colour.")

if 'banana' in favourite_fruits:

    print("The tree of banana is very weak.")
```

Output:

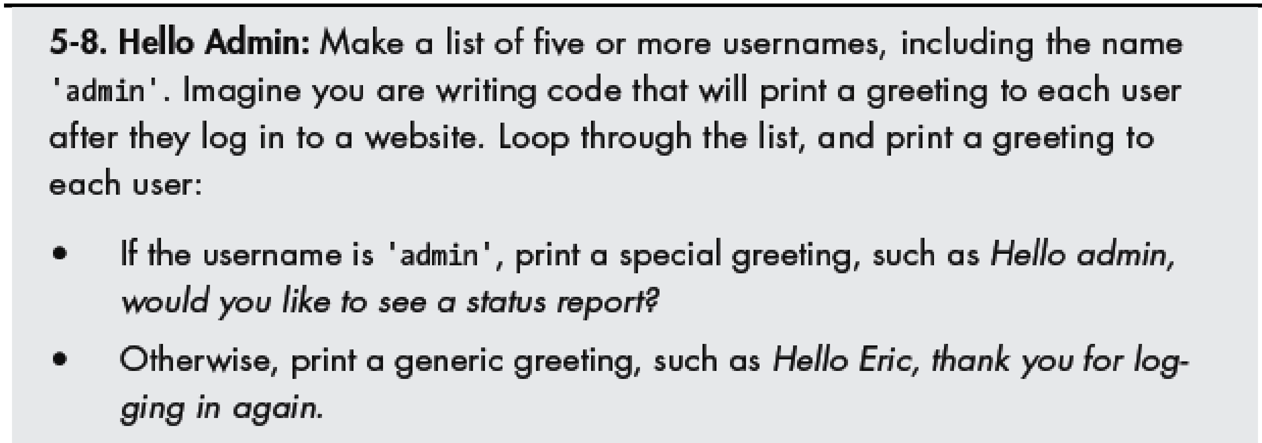
```
I like apple juice.

I like eat orange.

The tree of banana is very weak.
```

Task:

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)



5-8. Hello Admin: Make a list of five or more usernames, including the name `'admin'`. Imagine you are writing code that will print a greeting to each user after they log in to a website. Loop through the list, and print a greeting to each user:

- If the username is `'admin'`, print a special greeting, such as *Hello admin, would you like to see a status report?*
- Otherwise, print a generic greeting, such as *Hello Eric, thank you for logging in again.*

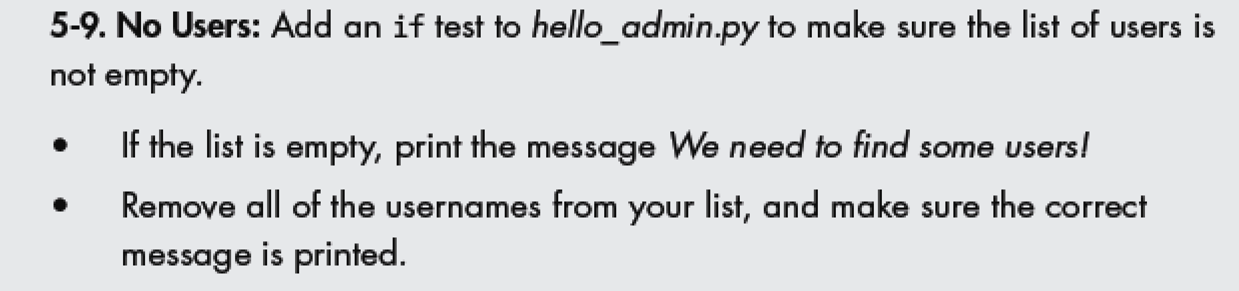
Solution:

```
usernames=['principal','professor','admin','teacher','officer']

for username in usernames:

  if username=='admin':

    print("Hello " + usernames[2].title() + ", would you like to eat mango?")

  else:

    print("Hello " + username.title() + ", thank you for logging in again.")
```

Output:

```
Hello Principal, thank you for logging in again.

Hello Professor, thank you for logging in again.

Hello Admin, would you like to eat mango?

Hello Teacher, thank you for logging in again.

Hello Officer, thank you for logging in again.
```

Task:



5-9. No Users: Add an `if` test to *hello_admin.py* to make sure the list of users is not empty.

- If the list is empty, print the message *We need to find some users!*
- Remove all of the usernames from your list, and make sure the correct message is printed.

Solution:

```
usernames=[]

if usernames:

   for username in usernames:
        print("Adding" + username +".")

else:

    print("We need to find some users.")
```

Output:

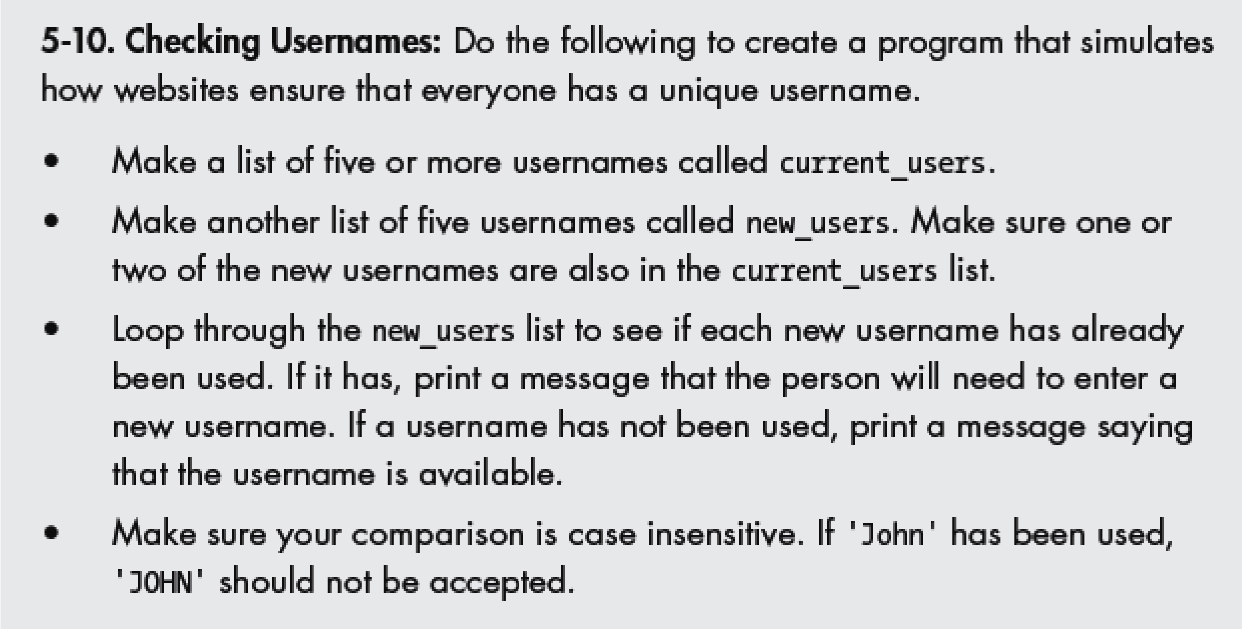
```
We need to find some users.
```

Task:

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)



5-10. Checking Usernames: Do the following to create a program that simulates how websites ensure that everyone has a unique username.

- Make a list of five or more usernames called current_users.
- Make another list of five usernames called new_users. Make sure one or two of the new usernames are also in the current_users list.
- Loop through the new_users list to see if each new username has already been used. If it has, print a message that the person will need to enter a new username. If a username has not been used, print a message saying that the username is available.
- Make sure your comparison is case insensitive. If 'John' has been used, 'JOHN' should not be accepted.

Solution:

```
current_users=['akmal','baber','azam','umar','zaman']

new_users=['haris','akmal','ahmad','umar','shadab']

for new_user in new_users:

   if new_user in current_users:
        print("The " + new_user + " will need to enter a new username.")
    else:
        print("The " + new_user + " is available.")
```

Output:

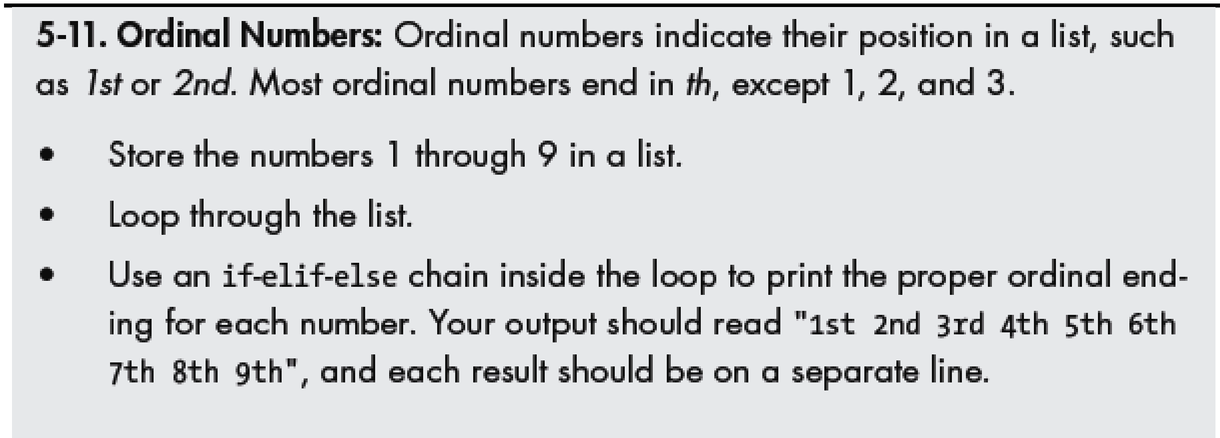
```
The haris is available.
The akmal will need to enter a new username.
The ahmad is available.
The umar will need to enter a new username.
The shadab is available.
```

Task:

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)



5-11. Ordinal Numbers: Ordinal numbers indicate their position in a list, such as *1st* or *2nd*. Most ordinal numbers end in *th*, except 1, 2, and 3.

- Store the numbers 1 through 9 in a list.
- Loop through the list.
- Use an `if-elif-else` chain inside the loop to print the proper ordinal ending for each number. Your output should read `"1st 2nd 3rd 4th 5th 6th 7th 8th 9th"`, and each result should be on a separate line.

Solution:

```
numbers=[1,2,3,4,5,6,7,8,9]

for number in numbers:

   if number==1:

      print(str(number) + "st")

   elif number==2:

      print(str(number) + "nd")

   elif number==3:
        print(str(number) + "rd")
    elif number==4:
        print(str(number) + "th")
    elif number==5:
        print(str(number) + "th")
    elif number==6:
        print(str(number) + "th")
    elif number==7:
        print(str(number) + "th")
    elif number==8:
        print(str(number) + "th")
    else:
        print(str(number) + "th")
```

Output:

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

1st

2nd

3rd

4th

5th

6th

7th

8th

9th

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

Chapter 6

Task 6-1:

```
person={'first_name':'Hameed','last_name':'khan','age':'23','city':'peshawar'}

for key,value in person.items():

    print(key+":"+value)
```

Output:

```
first_name:Hameed

last_name:khan

age:23

city:peshawar
```

Task 6-2:

```
favourite_numbers={'karim':25,'salman':20,'asad':30,'saood':55,'hayat':99}

for k,v in favourite_numbers.items():

    print(k+":"+str(v))
```

Output:

```
karim:25

salman:20

asad:30

saood:55

hayat:99
```

Task 6-3:

```
glossary={'int':'A series of numbers.',

        'string':'A series of characters.',

        'list':'A collection of items in a particular order.',

        'loop':'runs a piece of code again and again.',

        'dictionary':'A collection of key-value pairs.'}

print('int:'+glossary['int'])
print('String:'+glossary['string'])

print('List:'+glossary['list'])

print('Loop:'+glossary['loop'])

print('Dictionary:'+glossary['dictionary'])
```

Output:

```
int:A series of numbers.

String:A series of characters.

List:A collection of items in a particular order.

Loop:runs a piece of code again and again.

Dictionary:A collection of key-value pairs.
```

Task6-4:

```
glossary2={'int':'A series of numbers.',

    'string':'A series of characters.',

    'list':'A collection of items in a particular order.',
          'loop':'runs a piece of code again and again.',

          'dictionary':'A collection of key-value pairs.',

           'key':'The first item 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.',

           'boolean expression':'An expression that is either True or False.'}

for key,value in glossary2.items():

    print(key.title() + ":" + value)
```

Output:

```
Int:A series of numbers.

String:A series of characters.

List:A collection of items in a particular order.

Loop:runs a piece of code again and again.
```

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan   ID# 14486

Dept: BS (CS)   Final Term lab Exam   Subject: Modren programming language (MPL)

Dictionary:A collection of key-value pairs.

Key:The first item 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.

Boolean Expression:An expression that is either True or False.

Task 6-5:

```
rivers={'kabul':'Afghanistan','Indus':'India','chenab':'Pakistan'}

for river,country in rivers.items():

   print("The " + river.title() + " runs through " + country.title() + ".")

print("Rivers:")

for river in rivers.keys():

   print(river)
print("Country:")

for country in rivers.values():

   print(country)
```

Output:

```
The Kabul runs through Afghanistan.

The Indus runs through India.

The Chenab runs through Pakistan.

Rivers:

kabul

Indus

chenab

Country:

Afghanistan

India
```

Task 6-6:

```
favourite_languages={'amir':'c++','hameed':'java','salman':'ruby','karim':'python'}

for n,l in favourite_languages.items():

   print(n.title() + "'s favourite language is " + l.title() + ".")

names=['taimoor','hameed','saood','asad','karim','bilal','salman']

for name in names:

   if name in favourite_languages.keys():

      print("Thank you for responding," + name.title(bv) + ".")

   else:

      print(name.title() + ", inviting you to take the poll.")
```

Output:

```
Amir's favourite language is C++.

Hameed's favourite language is Java.
```

```
Salman's favourite language is Ruby.

Karim's favourite language is Python.

Taimoor, inviting you to take the poll.

Thank you for responding,Hameed.

Saood, inviting you to take the poll.

Asad, inviting you to take the poll.

Thank you for responding,Karim.

Bilal, inviting you to take the poll.

Thank you for responding,Salman.
```

Task 6-7:

```
people=[]

person={'first_name':'Hameed','last_name':'khan','age':'23','city':'peshawar'}

people.append(person)

person1={'first_name':'salman','last_name':'khan','age':'30','city':'swat'}
```

```
people.append(person1)

person2={'first_name':'hayat','last_name':'ahmad','age':'80','city':'mardan'}

people.append(person2)

for person in people:

    name=person['first_name'] + " " + person['last_name']

    age=str(person['age'])

    city=person['city']

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

Output:

```
Hameed khan is 23 years old from peshawar.

salman khan is 30 years old from swat.

hayat ahmad is 80 years old from mardan.
```

Task 6-8:

```
Pet1={'name':'lion','kind of animal':'carnivore',"owner's name":'khayyam'}
```

```
Pet2={'name':'goat','kind of animal':'herbivore',"owner's name":'yaman'}

Pet3={'name':'hen','kind of animal':'omnivore',"owner's name":'zaman'}

pets =[pet1,pet2,pet3]

for pet in pets:

        print(pet)
```

Output:

```
{'name':'lion','kind of animal':'carnivore',"owner's name":'khayyam'}

{'name':'goat','kind of animal':'herbivore',"owner's name":'yaman'}

{'name':'hen','kind of animal':'omnivore',"owner's name":'zaman'}
```

Task 6-9:

```
favourite_places={'mazhar':['naran','lahore','swat'],

          'hameed':['peshawar','murree','karachi'],

          'hayat':['kalam','dir','kohat']}

for name,places in favourite_places.items():
    print("\n" + name.title() + "'s favourite places are:")

    for place in places:

        print(place.title())
```

Output:

```
Mazhar's favourite places are:

Naran

Lahore

Swat


Hameed's favourite places are:

Peshawar

Murree

Karachi
```

```
Hayat's favourite places are:

Kalam

Dir

Kohat
```

Task 6-10:

```
favourite_numbers={'karim':[25,50,70],'salman':[20,34,98],'asad':[30,45,59],'saood':[55,58,67],'hayat':[99,111]}

for k,v in favourite_numbers.items():

   print(k+":"+str(v))
```

Output:

```
karim:[25, 50, 70]

salman:[20, 34, 98]

asad:[30, 45, 59]

saood:[55, 58, 67]

hayat:[99, 111]
```

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

Task 6-11:

```
cities={'Lahore':{'country':'Pakistan','population':'3453677','fav_place':'Shahi Qila'},
    'Mumbai':{'country':'India','population':'1840007','fav_place':'Victoria Terminus'},
    'Macau':{'country':'China','population':'555466','fav_place':'The Venetian'}}
for city,city_info in cities.items():
  cities={'Lahore':{'country':'Pakistan','population':'3453677','fav_place':'Shahi Qila'},
    'Mumbai':{'country':'India','population':'1840007','fav_place':'Victoria Terminus'},
    'Macau':{'country':'China','population':'555466','fav_place':'The Venetian'}}
for city,city_info in cities.items():
  country=city_info['country']
  population=city_info['population']
  favourite_place=city_info['fav_place']
  print(city + " is in " + country + ".")
  print("\tIt has a population of about " + str(population) + ".")
```

```
    print("\tIts favpourite place is " + favourite_place + ".")



    country=city_info['country']

    population=city_info['population']

    favourite_place=city_info['fav_place']

    print(city + " is in " + country + ".")

    print("\tIt has a population of about " + str(population) + ".")

    print("\tIts favpourite place is " + favourite_place + ".")
```

Output:

```
Lahore is in Pakistan.

        It has a population of about 3453677.

        Its favpourite place is Shahi Qila.

Mumbai is in India.

        It has a population of about 1840007.
        Its favpourite place is Victoria Terminus.

Macau is in China.

        It has a population of about 555466.

        Its favpourite place is The Venetian.
```

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

Chapter 7

Task 7-1:

```
rental_car=input("What kind of rental car you would like?")

print("Let me see if I can find you a " + rental_car + ".")
```

Output:

```
What kind of rental car you would like?Suzuki

Let me see if I can find you a Suzuki.
```

Task 7-2:

```
Restaurant_seats=input("How many people are in your dinner group?")

Restaurant_seats=int(Restaurant_seats)

if Restaurant_seats > 8:

   print("you will have to wait for a table, I am sorry!!!" )

else:

   print("your table is ready.")
```

Output:

```
How many people are in your dinner group?8

your table is ready.
```

Task 7-3:

```
multiples_of_ten=input("Enter a number:")

multiples_of_ten=int(multiples_of_ten)

if multiples_of_ten % 10 == 0:

   print(str(multiples_of_ten) + " is a multiple of 10. ")

else:

   print(str(multiples_of_ten) + " is not a multiple of 10.")
```

Output:

```
Enter a number:100

100 is a multiple of 10.
```

Task 7-4:

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

```
pizza="Enter a series of pizza toppings."

pizza += "\nEnter 'quit' when you are finished:"

while True:

    topping=input(pizza)

    if topping != 'quit':

        print("you will add " + topping + " to your pizza.")

    else:

        break
```

Output:

```
Enter a series of pizza toppings.

Enter 'quit' when you are finished:hut

you will add hut to your pizza.

Enter a series of pizza toppings.

Enter 'quit' when you are finished:quit
```

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

Task 7-5:

```
text="Enter you age."
text +="\nEnter 'quit' when you are finished."
while True:
   age=input(text)
   if age == 'quit':
      break
   age=int(age)
   if age < 3:
      print("The ticket is free.")
   elif age < 12:
      print("The ticket is $10.")
   else:
      print("The ticket is $15.")
```

Output:

```
Enter you age.

Enter 'quit' when you are finished.20

The ticket is $15.

Enter you age.

Enter 'quit' when you are finished.quit
```

Task 7-6:

(a)

```
text="Enter you age."

text +="\nEnter 'quit' when you are finished."

while True:

    age=input(text)

    if age == 'quit':

        break
    if age == 'stop':

        break

    age=int(age)

    if age < 3:

        print("The ticket is free.")

    elif age < 12:

        print("The ticket is $10.")

    else:

        print("The ticket is $15.")
```

(b)

```
text="Enter you age."

text +="\nEnter 'quit' when you are finished."

active = True

while active:
    age=input(text)
    if age == 'quit':
        break
    age=int(age)
    if age < 3:
        print("The ticket is free.")
    elif age < 12:
        print("The ticket is $10.")
    else:
        print("The ticket is $15.")
```

(c)

```
text="Enter you age."
text +="\nEnter 'quit' when you are finished."
while True:
    age=input(text)
    if age == 'quit':
        break
    age=int(age)
    if age < 3:
        print("The ticket is free.")
    elif age < 12:
        print("The ticket is $10.")
    else:
        print("The ticket is $15.")
```

Output:

(a)

```
Enter you age.
Enter 'quit' when you are finished.stop
```

(b)

```
Enter you age.

Enter 'quit' when you are finished.50

The ticket is $15.

Enter you age.

Enter 'quit' when you are finished.3

The ticket is $10.

Enter you age.

Enter 'quit' when you are finished.4

The ticket is $10.

Enter you age.

Enter 'quit' when you are finished.2

The ticket is free.
```

```
Enter you age.

Enter 'quit' when you are finished.quit
```

(c)

```
Enter you age.

Enter 'quit' when you are finished.quit
```

Task 7-7:

```
number=30

while number>=29:

   print(number)
```

Output:

```
Infinite Loop
```

Task 7-8:

```
sandwich_orders=['roast','turkey','cheese']

finished_sandwiches=[]
```

```
while sandwich_orders:

    sandwich = sandwich_orders.pop()

    print("I made your " + sandwich + " sandwich.")

    finished_sandwiches.append(sandwich)

print("\n")

for finished_sandwich in finished_sandwiches:

    print("I made a " + finished_sandwich + " sandwich.")
```

Output:

```
I made your cheese sandwich.

I made your turkey sandwich.

I made your roast sandwich.




I made a cheese sandwich.
```

```
I made a turkey sandwich.

I made a roast sandwich.
```

Task 7-9:

```
sandwich_orders=['pastrami','roast','pastrami','turkey','pastrami','cheese']

finished_sandwiches=[]

print("I am sorry, we has run out of pastrami.")

while 'pastrami' in sandwich_orders:

   sandwich_orders.remove('pastrami')

while sandwich_orders:

   sandwich = sandwich_orders.pop()

   print("I made your " + sandwich + " sandwich.")

   finished_sandwiches.append(sandwich)

print("\n")

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

Output:

```
I am sorry, we has run out of pastrami.

I made your cheese sandwich.

I made your turkey sandwich.

I made your roast sandwich.




I made a cheese sandwich.

I made a turkey sandwich.

I made a roast sandwich.
```

Task 7-10:

```
name_prompt="What is your name?"
```

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

```
place_prompt="If you could visit one place in the world, where would you go?"

continue_prompt="\nWould you like to let someone else respond?(yes/no)"

responses={}

while True:

    name=input(name_prompt)

    place=input(place_prompt)

    responses[name]=place

    repeat=input(continue_prompt)

    if repeat != 'yes':

        break

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

for name,place in responses.items():

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

Output:

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

```
What is your name?hameed khan

If you could visit one place in the world, where would you go?london



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



---Results---

Hameed Khan would like to visit London.
```

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

Chapter 8

Task 8-1:

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.

Solution:

```
def display_message():
   print("Hello Hameed, What you are learning about in this chapter?")
display_message()
```

Output:

```
Hello Hameed, What you are learning about in this chapter?
```

Task 8-2:

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.

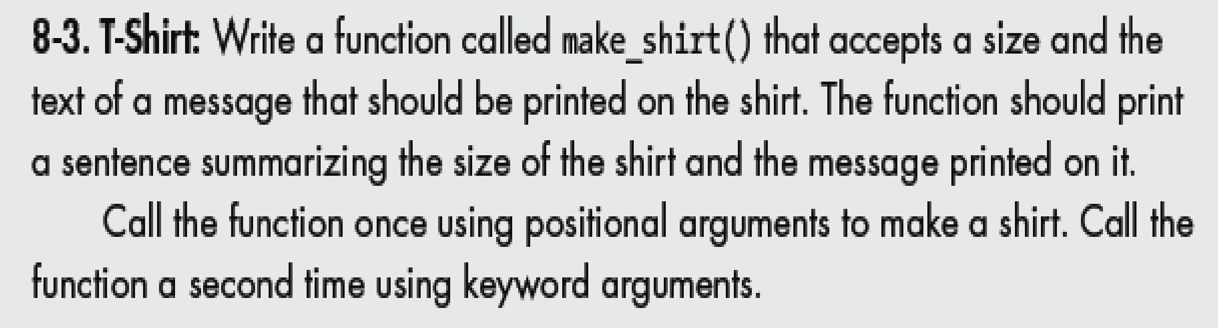
Solution:

```
def favourite_book(title):

   print("One of my favourite book is " + title.title() + " in wonderland.")

favourite_book('alice')
```

Output:

```
One of my favourite book is Alice in wonderland.
```

Task 8-3:



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.

Solution:

```
def make_shirt(size, text):

   print("The size of the shirt is " + str(size) + " inch long.")

   print("The message that should be printed on the shirt is " + text.title() + ".")

make_shirt(15, 'love you')

make_shirt(22, "don't touch me")
```

Output:

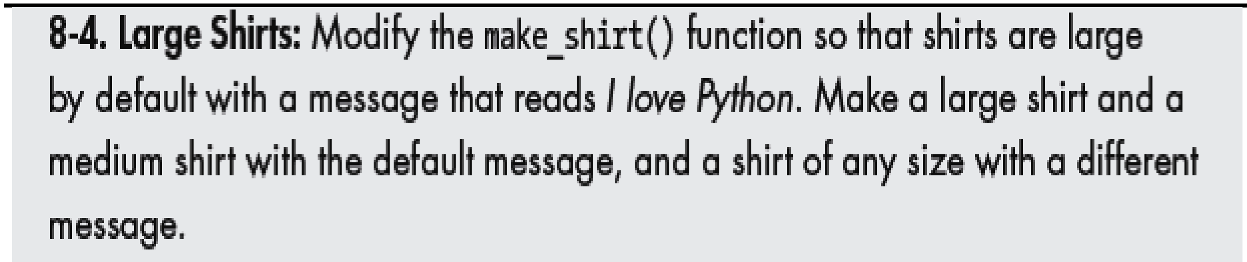
```
The size of the shirt is 15 inch long.

The message that should be printed on the shirt is Love You.

The size of the shirt is 22 inch long.

The message that should be printed on the shirt is Don'T Touch Me.
```

Task 8-4:

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)



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.

Solution:

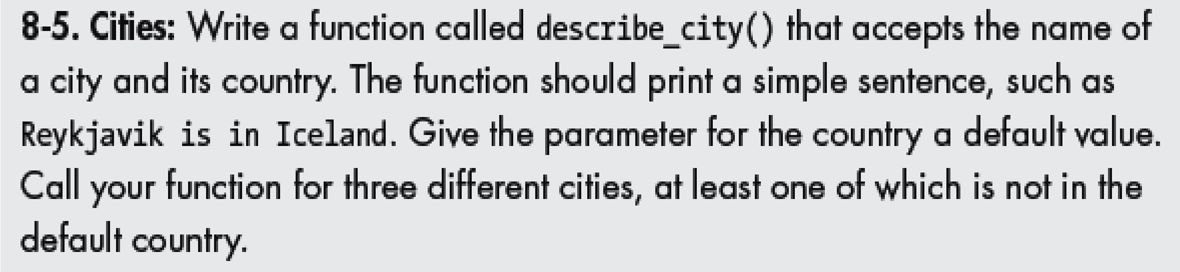
```
def make_shirt (text, size = 'large'):

   print("The size of the shirt is " + size.title() + ".")

   print("The message that should be printed on the shirt is " + text.title() + ".")

make_shirt(text = 'i love python')

make_shirt(text = "don't touch me", size = 'medium')
```

Output:

```
The size of the shirt is Large.

The message that should be printed on the shirt is I Love Python.

The size of the shirt is Medium.
```

The message that should be printed on the shirt is Don'T Touch Me.

Task 8-5:



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.

Solution:

```
def describe_city(city_name, city_country = 'pakistan'):

   print(city_name.title() + " is in " + city_country.title() + ".")

describe_city(city_name ='lahore')

describe_city('reykjavik', 'iceland')

describe_city('mumbai', 'india')
```

Output:

```
Lahore is in Pakistan.
```

```
Reykjavik is in Iceland.

Mumbai is in India.
```

Task 8-6:

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 value

that’s returned.

Solution:

```
def city_country(city_name, country_name):

    city_country = city_name + " , " + country_name

    return city_country.title()

value_1 = city_country('karachi', 'pakistan')

value_2 = city_country('kolkata', 'india')
```

```
value_3 = city_country('dhaka', 'bangladesh')

print(value_1)

print(value_2)

print(value_3)
```

Output:

```
Karachi , Pakistan

Kolkata , India

Dhaka , Bangladesh
```

Task 8-7:

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.

Add an optional parameter to make_album() that allows you to store the

number of tracks on an album. If the calling line includes a value for the number

of tracks, add that value to the album’s dictionary. Make at least one new

function call that includes the number of tracks on an album.

Solution:

```
def make_album(artist_name, album_title, no_of_tracks):

    music_album = {'artist' : artist_name, 'album' : album_title, 'tracks' : no_of_tracks}

    return music_album

dictionary_1 = make_album('atif', 'music', 4)

dictionary_2 = make_album('hameed', 'news', 2)

dictionary_3 = make_album('salman', 'picture', 10)

print(dictionary_1)
print(dictionary_2)

print(dictionary_3)

def number_of_tracks(no_of_tracks):

   print("The number of tracks on an album is " + str(no_of_tracks) + ".")

number_of_tracks(6)
```

Output:

```
{'artist': 'atif', 'album': 'music', 'tracks': 4}

{'artist': 'hameed', 'album': 'news', 'tracks': 2}

{'artist': 'salman', 'album': 'picture', 'tracks': 10}

The number of tracks on an album is 6.
```

Task 8-8:

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.

Solution:

```
def make_album(music_album):

    music_album['artist'] = input("Artist Name: ")

    music_album['album'] = input("Album Title: ")



music_album = {'artist' : '', 'album' : ''}

while True:

    exit_st = input("(Enter 'm' at any time to quit or press any other key to continue): ")

    if exit_st == 'm':

        print(music_album)

        break

    print("\nPlease tell me about album's information:")

    make_album( music_album)
```

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

Task 8-9:

Magicians: Make a list of magician’s names. Pass the list to a function

called show_magicians(), which prints the name of each magician in the list.

Solution:

```
def show_magicians(names):

    for name in names:

        print(name.title())

magicians_names = ['musa','hameed','salman','karim']

show_magicians(magicians_names)
```

Output:

```
Musa

Hameed

Salman

Karim
```

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

Task 8-10:

Great Magicians: Start with a copy of your program from Exercise 8-9.

Write a function called make_great() that modifies the list of magicians by adding

the phrase *the Great* to each magician’s name. Call show_magicians() to

see that the list has actually been modified.

Solution:

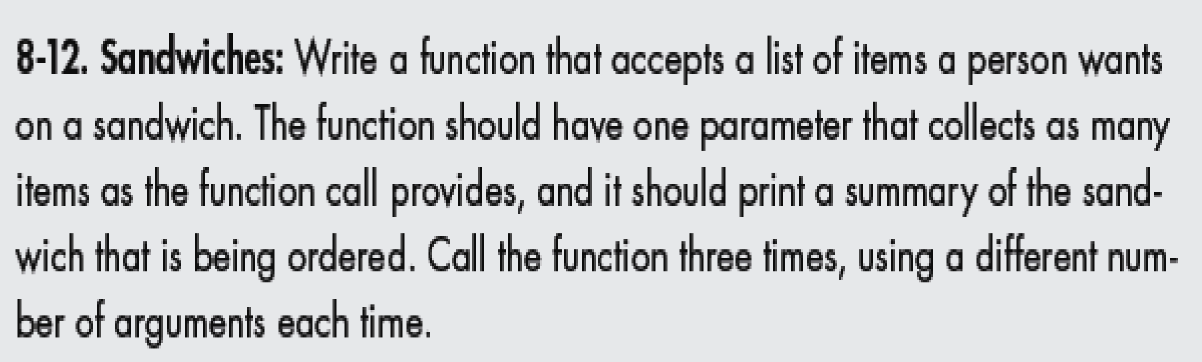
```
def show_magicians(names, modifies_names):
    while names:
        current_name = names.pop()
        modifies_names.append(current_name)
def make_great(modifies_names):
    for modifies_name in modifies_names:
        print("The great " + modifies_name.title())
names = ['musa','hameed','salman','karim']
```

```
modifies_names = []

show_magicians(names, modifies_names)

make_great(modifies_names)
```

Output:

```
The great Karim

The great Salman

The great Hameed

The great Musa
```

Task 8-12:



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 is being ordered. Call the function three times, using a different number of arguments each time.

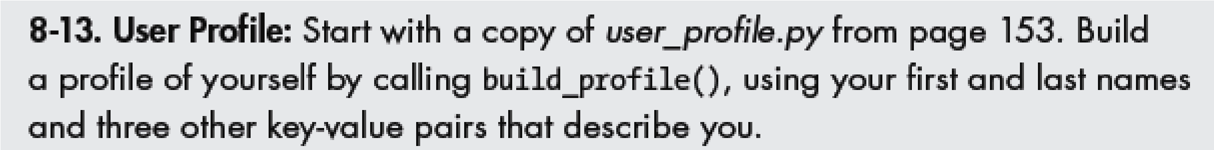
Solution:

```
def sandwiches(*sandwich):

   print("The " + str(sandwich) + " sandwich is very Excellent.")

sandwiches("balochi")

sandwiches('sindhi', 'punjabi')

sandwiches('balochi', 'sindhi', 'afghani', 'mushroom')
```

Output:

```
The ('balochi',) sandwich is very Excellent.

The ('sindhi', 'punjabi') sandwich is very Excellent.

The ('balochi', 'sindhi', 'afghani', 'mushroom') sandwich is very Excellent.
```

Task 8-13:



8-13. User Profile: Start with a copy of *user_profile.py* from page 153. Build a profile of yourself by calling `build_profile()`, using your first and last names and three other key-value pairs that describe you.

Solution:

```
def build_profile(first, last, **user_info):
    """Build a dictionary containing everything we know about a user."""

    profile = {}

    profile['first_name'] = first

    profile['last_name'] = last

    for key, value in user_info.items():

        profile[key] = value

    return profile

my_profile = build_profile('muhammad', 'musa',

                           location='swabi',

                           field='computer science',

                           age= '23')

print(my_profile)
```

Output:

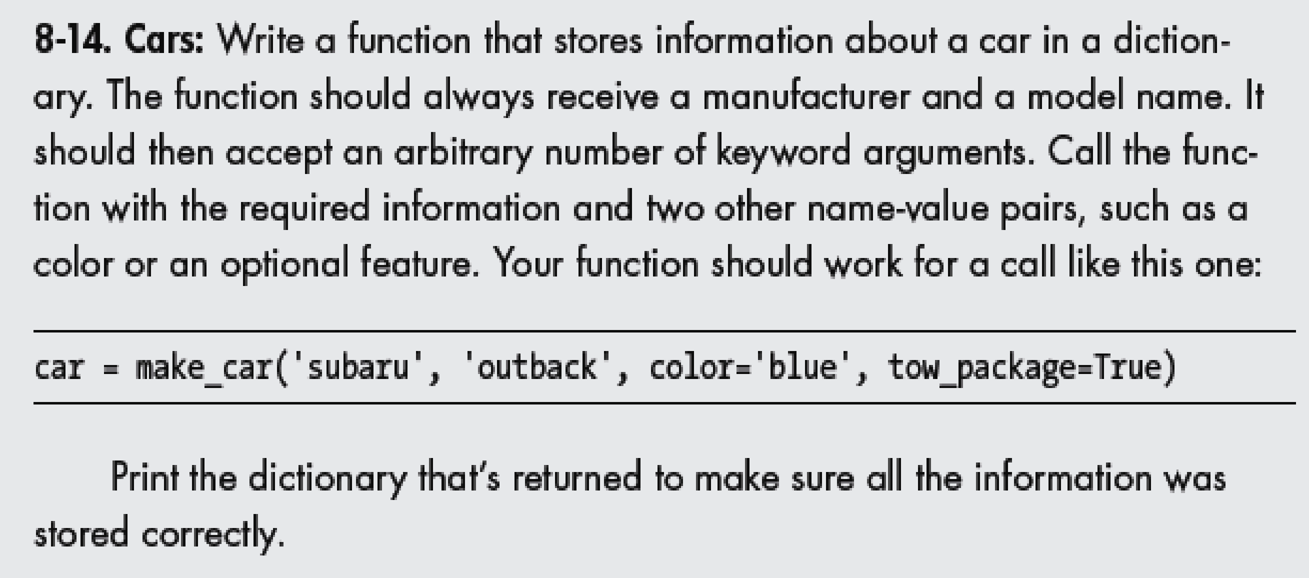
```
{'first_name': 'muhammad', 'last_name': 'musa', 'location': 'swabi', 'field': 'computer science', 'age': '23'}
```

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

Task 8-14:



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.

Solution:

```
def make_car(manufacturer_name, model_name, **car_info):

    profile = {}

    profile['manufacturer_name'] = manufacturer_name
    profile['model_name'] = model_name

    for key, value in car_info.items():

        profile[key] = value

    return profile

car = make_car('suzuki','gli',color = 'red',wheel = '4')

print(car)
```

Output:

```
{'manufacturer_name': 'suzuki', 'model_name': 'gli', 'color': 'red', 'wheel': '4'}
```

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

Chapter 9

Task 9-1:

Make a class called Restaurant. The __init__() method for

Restaurant should store two attributes: a restaurant_name and a cuisine_type.

Make a method called describe_restaurant() that prints these two pieces of

information, and a method called open_restaurant() that prints a message indicating

that the restaurant is open.

Make an instance called restaurant from your class. Print the two attributes

individually, and then call both methods.

Solution:

```
class Restaurant():
    def __init__(self, restaurant_name, cuisine_type):
        self.restaurant_name = restaurant_name
        self.cuisine_type = cuisine_type
    def describe_restaurant(self):

        print("The " + self.restaurant_name.title() + " restaurant is near Swabi.")

        print("The restaurant's food is " + self.cuisine_type.title())

    def open_restaurant(self):

        print("The " + self.restaurant_name.title() + " restaurant is open.")

restaurant = Restaurant('bacha khan' , 'delicious')

print(restaurant.restaurant_name)

print(restaurant.cuisine_type)

restaurant.describe_restaurant()

restaurant.open_restaurant()
```

Output:

```
bacha khan

delicious

The Bacha Khan restaurant is near Swabi.
The restaurant's food is Delicious

The Bacha Khan restaurant is open.
```

Task 9-2:

Start with your class from Exercise 9-1. Create three

different instances from the class, and call describe_restaurant() for each

instance.

Solution:

```
class Restaurant():

    def __init__(self, restaurant_name, cuisine_type):

        self.restaurant_name = restaurant_name

        self.cuisine_type = cuisine_type

    def describe_restaurant(self):

        print("The " + self.restaurant_name.title() + " restaurant is near Swabi.")

        print("The restaurant's food is " + self.cuisine_type.title() + ".")
    def open_restaurant(self):
        print("The " + self.restaurant_name.title() + " restaurant is open.")
restaurant = Restaurant('peshawar', 'delicious')
restaurant_1=Restaurant('islamia', 'good')
restaurant_2=Restaurant('kabul', 'excellent')
restaurant.describe_restaurant()
restaurant_1.describe_restaurant()
restaurant_2.describe_restaurant()
```

Output:

```
The Peshawar restaurant is near Swabi.
The restaurant's food is Delicious.
The Islamia restaurant is near Swabi.
The restaurant's food is Good.
The Kabul restaurant is near Swabi.
```

The restaurant's food is Excellent.

Task 9-3:

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.

Solution:

```
class User():
    def __init__(self, first_name, last_name, age, address):
        self.first_name = first_name
        self.last_name = last_name
        self.age = age
        self.address = address
    def describe_user(self):
        print(self.first_name.title() + " " + self.last_name.title() + " is " + str(self.age) + " years old from " + self.address.title() + ".")
    def greet_user(self):
        print("Assalam O Alaikum " + self.first_name.title() + " " + self.last_name.title() + "; How are you?")
user_1 = User('muhammad', 'musa', '23', 'swabi')
user_2 = User('hameed', 'khan', '40', 'upper dir')
user_3 = User('karim', 'ullah', '32', 'swat')
user_1.describe_user()
user_1.greet_user()
user_2.describe_user()
user_2.greet_user()
user_3.describe_user()
```

```
user_3.greet_user()
```

Output:

```
Muhammad Musa is 23 years old from Swabi.

Assalam O Alaikum Muhammad Musa; How are you?

Hameed Khan is 40 years old from Upper Dir.

Assalam O Alaikum Hameed Khan; How are you?

Karim Ullah is 32 years old from Swat.

Assalam O Alaikum Karim Ullah; How are you?
```

Task 9-4:

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.

Solution:

```
class Restaurant():
    def __init__(self, restaurant_name, cuisine_type):
        self.restaurant_name = restaurant_name
        self.cuisine_type = cuisine_type
        self.number_served = 0
    def describe_restaurant(self):
        print("The " + self.restaurant_name.title() + " restaurant is near Swabi.")
        print("The restaurant's food is " + self.cuisine_type.title() + ".")

    def open_restaurant(self):

        print("The " + self.restaurant_name.title() + " restaurant is open.")

    def set_number_served(self):

        print("The number of customers the restaurant has served are " + str(self.number_served) + ".")

    def increment_number_served(self,customers):

        self.number_served += customers

restaurant = Restaurant('peshawar', 'delicious')

restaurant.set_number_served()

restaurant.number_served = 22

restaurant.set_number_served()

restaurant.increment_number_served(100)

restaurant.set_number_served()
```

Output:

```
The number of customers the restaurant has served are 0.

The number of customers the restaurant has served are 22.

The number of customers the restaurant has served are 122.
```

Task 9-5:

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.

Solution:

IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

```
class User():
    def __init__(self, first_name, last_name, age, address):
        self.first_name = first_name
        self.last_name = last_name
        self.age = age
        self.address = address
        self.login_attempts = 1
    def describe_user(self):
        print(self.first_name.title() + " " + self.last_name.title() + " is " + str(self.age) + " years old from " + self.address.title() + ".")
    def greet_user(self):
        print("Assalam O Alaikum " + self.first_name.title() + " " + self.last_name.title() + "; How are you?")
    def increment_login_attempts(self, attempts):
        self.login_attempts += attempts
        print(self.login_attempts)
    def reset_login_attempts(self):

        self.login_attempts = 0

        print(self.login_attempts)

user_1 = User('muhammad', 'musa', '23', 'swabi')

user_1.increment_login_attempts(22)

user_1.increment_login_attempts(44)

user_1.increment_login_attempts(12)

user_1.increment_login_attempts(20)

user_1.reset_login_attempts()
```

Output:

```
23

67

79

99
```

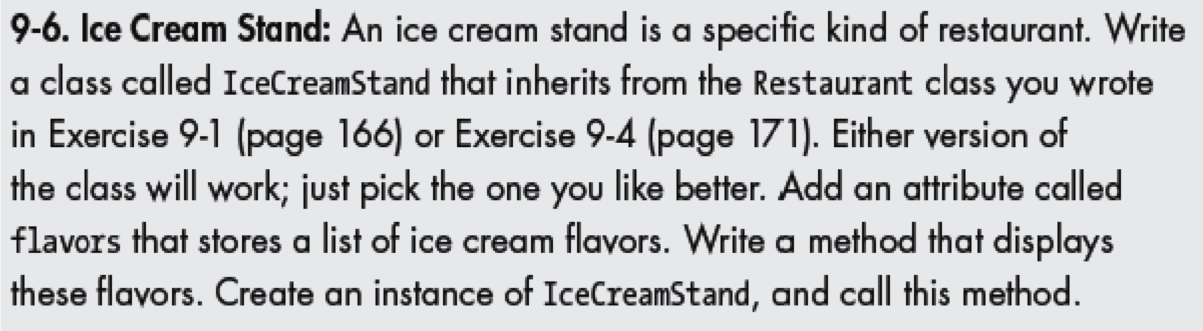
IQRA NATIONAL UNIVERSITY
Phase II,Hayatabad peshawer

Student: Hayat ahmad khan ID# 14486

Dept: BS (CS) Final Term lab Exam Subject: Modren programming language (MPL)

0

Task 9-6:



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.

Solution:

```
class Restaurant():

   def __init__(self, restaurant_name, cuisine_type):

       self.restaurant_name = restaurant_name

       self.cuisine_type = cuisine_type

       self.number_served = 0

       #self.flavors = flavors
```

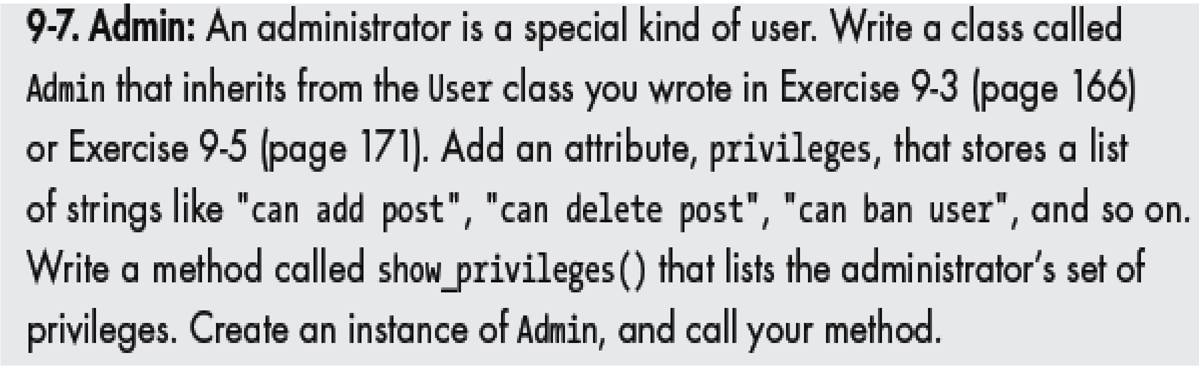
```
    def describe_restaurant(self):

        print("The " + self.restaurant_name.title() + " restaurant is near Swabi.")

        print("The restaurant's food is " + self.cuisine_type.title() + ".")

    def open_restaurant(self):

        print("The " + self.restaurant_name.title() + " restaurant is open.")

    def set_number_served(self):

        print("The number of customers the restaurant has served are " + str(self.number_served) + ".")

    def increment_number_served(self,customers):

        self.number_served += customers

class icecreamstand(Restaurant):

    def __init__(self, restaurant_name, cuisine_type):

        super().__init__(restaurant_name, cuisine_type)

        self.flavors = 'strawberry'

    def display_flavors(self):
```

```
    print("The icecream has " + self.flavors + " flavor.")

icecreamstand_1 = icecreamstand('kabul', 'excellent')

icecreamstand_1.display_flavors()
```

Output:

```
The icecream has strawberry flavor.
```

Task 9-7:



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.

Solution:

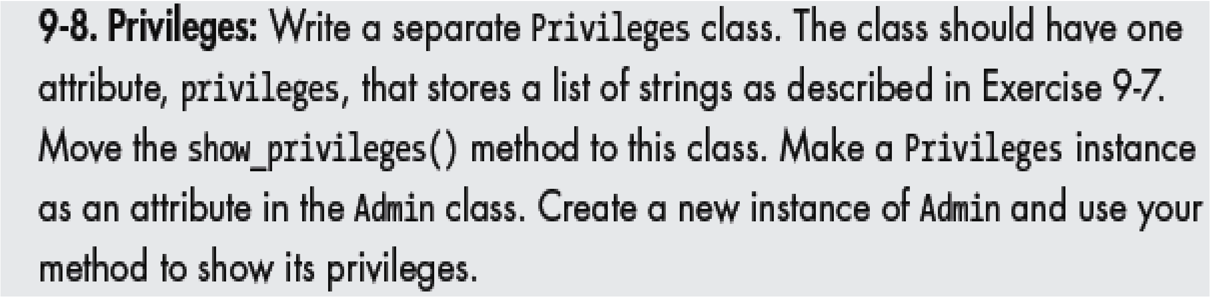
```
class User():

  def __init__(self, first_name, last_name, age, address):
        self.first_name = first_name

        self.last_name = last_name

        self.age = age

        self.address = address

    def describe_user(self):

        print(self.first_name.title() + " " + self.last_name.title() + " is " + str(self.age) + " years old from " + self.address.title() + ".")

    def greet_user(self):

        print("Assalam O Alaikum " + self.first_name.title() + " " + self.last_name.title() + "; How are you?")

class admin(User):

    def __init__(self, first_name, last_name, age, address):

        super().__init__(first_name, last_name, age, address)

        self.privileges =”'can add post' , 'can delete post', 'can ban user'"

    def show_privileges(self):

        print("The administrations set of privileges are " + self.privileges + ".")
```

```
admin_1 = admin('muhammad', 'musa', '23', 'swabi')

admin_1.show_privileges()
```

Output:

```
The administrations set of privileges are 'can add post', 'can delete post', 'can ban user'.
```

Task 9-8:



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.

Solution:

```
class User():

   def __init__(self, first_name, last_name, age, address):

      self.first_name = first_name

      self.last_name = last_name
```

```
        self.age = age

        self.address = address

    def describe_user(self):

        print(self.first_name.title() + " " + self.last_name.title() + " is " + str(self.age) + " years old from " + self.address.title() + ".")

    def greet_user(self):

        print("Assalam O Alaikum " + self.first_name.title() + " " + self.last_name.title() + "; How are you?")

class admin(User):

    def __init__(self, first_name, last_name, age, address):

        super().__init__(first_name, last_name, age, address)

        self.privileges ="'can add post', 'can delete post', 'can ban user'"

    def show_privileges(self):

        print("The administrations set of privileges are " + self.privileges + ".")

class privileges():

    def __init__(self, privileges):
        self.privileges = privileges

    def show_privileges(self):

        print("The administrations set of privileges are " + self.privileges + ".")

privileges_1 = privileges('can add post')

privileges_1.show_privileges()

admin_1 = admin('Hameed', 'Khan', '24', 'dir')

admin_1.show_privileges()
```

Output:

```
The administrations set of privileges are can add post.

The administrations set of privileges are 'can add post', 'can delete post', 'can ban user'.
```