1. Create a zoo.py file first. Define the hours() function, which prints the string 'Open 9-5 daily'. Then, use the interactive interpreter to import the zoo module and call its hours() function.

Ans : To create the zoo.py file and define the hours() function, you can follow these steps:

1. Create a new file called **zoo.py** and open it in a text editor.
2. Inside the **zoo.py** file, add the following code:

def hours(): print('Open 9-5 daily')

1. Save the **zoo.py** file.

Once you have saved the **zoo.py** file, you can use the interactive interpreter to import the **zoo** module and call its **hours()** function. Here's an example:

>>> import zoo >>> zoo.hours()

Output:

Open 9-5 daily

In this example, the **zoo** module is imported using the **import** statement, and then the **hours()** function from the **zoo** module is called. The function call prints the string **'Open 9-5 daily'** to the console.

2. In the interactive interpreter, import the zoo module as menagerie and call its hours() function.

Ans : To create the zoo.py file and define the hours() function, you can follow these steps:

1. Create a new file called **zoo.py** and open it in a text editor.
2. Inside the **zoo.py** file, add the following code:

pythonCopy code

def hours(): print('Open 9-5 daily')

1. Save the **zoo.py** file.

Once you have saved the **zoo.py** file, you can use the interactive interpreter to import the **zoo** module and call its **hours()** function. Here's an example:

pythonCopy code

>>> import zoo >>> zoo.hours()

Output:

mathematicaCopy code

Open 9-5 daily

In this example, the **zoo** module is imported using the **import** statement, and then the **hours()** function from the **zoo** module is called. The function call prints the string **'Open 9-5 daily'** to the console.

3. Using the interpreter, explicitly import and call the hours() function from zoo.

Ans : To explicitly import and call the hours() function from the zoo module using the Python interpreter, you can follow these steps:

1. Launch the Python interpreter by opening a terminal or command prompt.
2. Enter the following commands in the interpreter:

>>> from zoo import hours >>> hours()

Output:

Open 9-5 daily

In this example, the **from zoo import hours** statement explicitly imports the **hours()** function from the **zoo** module. Then, the **hours()** function is called, which prints the string **'Open 9-5 daily'** to the console.

4. Import the hours() function as info and call it.

Ans : To import the hours() function from the zoo module as info and call it using the Python interpreter, you can follow these steps:

1. Launch the Python interpreter by opening a terminal or command prompt.
2. Enter the following commands in the interpreter:

>>> from zoo import hours as info >>> info()

Output:

Open 9-5 daily

In this example, the **from zoo import hours as info** statement imports the **hours()** function from the **zoo** module and assigns it the alias **info**. Then, the **info()** function is called, which prints the string **'Open 9-5 daily'** to the console.

5. Create a plain dictionary with the key-value pairs 'a': 1, 'b': 2, and 'c': 3, and print it out.

Ans :   
Certainly! Here's an example of how you can create a plain dictionary with the key-value pairs 'a': 1, 'b': 2, and 'c': 3, and print it out:

plain\_dict = {'a': 1, 'b': 2, 'c': 3} print(plain\_dict)

Output:

{'a': 1, 'b': 2, 'c': 3}

In this example, I created a dictionary called **plain\_dict** with the specified key-value pairs. The dictionary is then printed, displaying the key-value pairs as **{'a': 1, 'b': 2, 'c': 3}**.

6.Make an OrderedDict called fancy from the same pairs listed in 5 and print it. Did it print in the same order as plain?

Ans : Certainly! Here's an example of how you can create an OrderedDict called fancy from the same key-value pairs as in the previous example, and print it:

from collections import OrderedDict fancy = OrderedDict([('a', 1), ('b', 2), ('c', 3)]) print(fancy)

Output:

OrderedDict([('a', 1), ('b', 2), ('c', 3)])

In this example, I imported the **OrderedDict** class from the **collections** module. Then, I created an **OrderedDict** called **fancy** with the same key-value pairs as in the plain dictionary. The **fancy** dictionary is then printed, displaying the key-value pairs in the order they were added.

Yes, the **OrderedDict** maintains the insertion order of the elements, so the output will be the same as in the plain dictionary: **{'a': 1, 'b': 2, 'c': 3}**.

7. Make a default dictionary called dict\_of\_lists and pass it the argument list. Make the list dict\_of\_lists['a'] and append the value 'something for a' to it in one assignment. Print dict\_of\_lists['a'].

Ans : Certainly! Here's an example of how you can create a defaultdict called dict\_of\_lists and append a value to the list associated with the key 'a':

from collections import defaultdict dict\_of\_lists = defaultdict(list) dict\_of\_lists['a'].append('something for a') print(dict\_of\_lists['a'])

Output:

['something for a']

In this example, I imported the **defaultdict** class from the **collections** module. Then, I created a **defaultdict** called **dict\_of\_lists** with the argument **list**, which ensures that accessing a missing key will automatically create an empty list. Next, I appended the value **'something for a'** to the list associated with the key **'a'** using **dict\_of\_lists['a'].append('something for a')**. Finally, I printed the value associated with the key **'a'**, which is **['something for a']**.