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.**

As an AI language model, I can't directly create files on your computer, but I can provide you with the code to create the `zoo.py` file. Follow these steps:

Step 1: Create a file named `zoo.py` and add the following code to it:

```python

# File: zoo.py

def hours():

print('Open 9-5 daily')

```

Step 2: Save the `zoo.py` file in the desired location on your computer.

Step 3: Open the Python interactive interpreter (usually running `python` or `python3` in your terminal).

Step 4: Import the `zoo` module and call its `hours()` function:

```python

>>> import zoo

>>> zoo.hours()

```

The output will be:

```

Open 9-5 daily

```

Note: Make sure that the `zoo.py` file is in the same directory or is in a location where Python can find it (e.g., in a directory listed in the `PYTHONPATH` environment variable).

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

Sure! You can import the `zoo` module as `menagerie` in the Python interactive interpreter and then call its `hours()` function. Here's how you can do it:

Step 1: Open the Python interactive interpreter (usually running `python` or `python3` in your terminal).

Step 2: Import the `zoo` module as `menagerie`:

```python

>>> import zoo as menagerie

```

Step 3: Call the `hours()` function from the `menagerie` module:

```python

>>> menagerie.hours()

```

The output will be:

```

Open 9-5 daily

```

By importing the `zoo` module as `menagerie`, you give it an alias so that you can refer to it by the new name `menagerie` in the current Python session.

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

Sure! To explicitly import and call the `hours()` function from the `zoo` module in the Python interactive interpreter, follow these steps:

Step 1: Open the Python interactive interpreter (usually running `python` or `python3` in your terminal).

Step 2: Explicitly import the `hours()` function from the `zoo` module:

```python

>>> from zoo import hours

```

Step 3: Call the `hours()` function directly:

```python

>>> hours()

```

The output will be:

```

Open 9-5 daily

```

By using the `from zoo import hours` statement, you bring the `hours()` function into the current namespace, allowing you to call it directly without referencing the module name.

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

Certainly! To import the `hours()` function from the `zoo` module as `info` and then call it, follow these steps:

Step 1: Open the Python interactive interpreter (usually running `python` or `python3` in your terminal).

Step 2: Import the `hours()` function from the `zoo` module as `info`:

```python

>>> from zoo import hours as info

```

Step 3: Call the `info()` function directly:

```python

>>> info()

```

The output will be:

```

Open 9-5 daily

```

By using the `from zoo import hours as info` statement, you import the `hours()` function with an alias `info`. This allows you to call the function using the new name `info` in the current Python session.

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

Sure! You can create a plain dictionary with the given key-value pairs and print it out using the following Python code:

```python

# Create the dictionary

my\_dict = {'a': 1, 'b': 2, 'c': 3}

# Print the dictionary

print(my\_dict)

```

When you run this code, it will output:

```

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

```

This shows the plain dictionary with the specified key-value pairs. The keys are `'a'`, `'b'`, and `'c'`, and their corresponding values are `1`, `2`, and `3`, respectively.

**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?**

To create an `OrderedDict` named `fancy` with the same key-value pairs as in Question 5 and print it, you can use the `collections` module from the Python standard library. The `OrderedDict` retains the order in which items are inserted, unlike the standard `dict`.

Here's the code to create the `OrderedDict` and print it:

```python

from collections import OrderedDict

# Create the OrderedDict

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

# Print the OrderedDict

print(fancy)

```

When you run this code, it will output:

```

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

```

As you can see, the `OrderedDict` maintains the order of insertion, so the output order matches the order in which the items were added. In this case, the order is the same as the original plain dictionary.

**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’].**

To create a `defaultdict` called `dict\_of\_lists` with the argument `list` and append the value `'something for a'` to `dict\_of\_lists['a']` in one assignment, you can use the `collections` module from the Python standard library. The `defaultdict` automatically creates a new list if a key is accessed that doesn't exist yet.

Here's the code to create the `defaultdict` and perform the assignment:

```python

from collections import defaultdict

# Create the defaultdict with default value as an empty list

dict\_of\_lists = defaultdict(list)

# Append 'something for a' to dict\_of\_lists['a'] in one assignment

dict\_of\_lists['a'].append('something for a')

# Print dict\_of\_lists['a']

print(dict\_of\_lists['a'])

```

When you run this code, it will output:

```

['something for a']

```

The `defaultdict` automatically adds a new key `'a'` with an empty list as its value when trying to access `dict\_of\_lists['a']`. Then, the value `'something for a'` is appended to this newly created list.