**Assignment\_18**

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:**

Here's how you can create the `zoo.py` file with the `hours()` function:

```python

# zoo.py

def hours():

print('Open 9-5 daily')

```

Then, you can use the interactive interpreter to import the `zoo` module and call its `hours()` function:

```python

>>> import zoo

>>> zoo.hours()

Open 9-5 daily

```

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

Ans:

To import the zoo module as menagerie and call its hours() function in the interactive interpreter, follow these steps:

1. Launch the interactive interpreter by opening the terminal (or command prompt on Windows) and typing "python" (without quotes).

2. Once the interpreter has launched, type "import zoo as menagerie" (without quotes) to import the zoo module and assign it the alias "menagerie".

3. Type "menagerie.hours()" (without quotes) to call the hours() function from the menagerie module.

Here's what the session would look like:

```

Python 3.9.2 (default, Feb 19 2021, 17:03:19)

[GCC 10.2.0] on linux

Type "help", "copyright", "credits" or "license" for more information.

>>> import zoo as menagerie

>>> menagerie.hours()

Open 9-5 daily

```

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

Ans:

The `zoo` module only contains one `hours()` function, so you can simply import and call it like this:

```

from zoo import hours

hours()

```

This will print the string to the console.

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

Ans:

>>> from zoo import hours as info

>>> info()

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

Ans:

Here's how you can create a plain dictionary and print it out:

```

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

print(my\_dict)

```

This will output:

```

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

```

1. 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:

Yes, an `OrderedDict` will maintain the order in which the key-value pairs were added to the dictionary.

Here's an example:

```python

from collections import OrderedDict

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

print(plain)

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

print(fancy)

```

Output:

```

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

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

```

1. 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:

Here's how you can make a default dictionary called `dict\_of\_lists` and append a value to the list corresponding to 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'])

```

This will output:

```

['something for a']

```