The **load( )** function is used as per the following syntax :

```
<object> = pickle.load(<filehandle>)
```

For instance, to read an object in **nemp** from a file open in file-handle **fout**, you would write :

```
nemp = pickle.load( fout )
```

*Read from the file opened with file handle as **fout** and store the read data in an object namely **nemp***

Following program 5.11 does the same for you. It reads the objects written by program 5.8 from the file **Emp.dat** and displays them. But before the program 5.11, read the following box. **(Important)**

IMPORTANT

But before you move onto the program code, it is important to know that **pickle.load( )** function would raise ***EOFError*** (a run time exception) when you reach end-of-file while reading from the file. You can handle this by following one of the below given two methods.

- Use **try** and **except** blocks
- Using with statement

(i) Use try and except blocks

Thus, you must write **pickle.load( )** enclosed in **try** and **except** statements as shown below. The **try** and **except** statements together, can handle runtime exceptions. In the **try** block, *i.e.*, between the **try** and **except** keywords, you write the code that can generate an exception and in the **except** block, *i.e.*, below the **except** keyword, you write what to do when the exception (**EOF – end of file** in our case) has occurred. (See below)

```
<filehandle> = open (<filename>, <readmode>)
try :
    <object> = pickle.load(<filehandle>)
    # other processing statements
except EOFError :
    <filehandle>.close()
```

*In the **try** block, write the **pickle.load( )** statement and other processing statements. In order to read all the records, read inside a loop as shown in the following program.*

Use this keyword with except keyword for checking EOF (end of file)

*In the **except** block, write code for what to do when EOF exception has occurred.*

Here, you just need to just concentrate on the syntax; you need not go in further details of **try** and **except** as it is beyond the scope of the book.

(ii) Using with statement

The **with** statement is a compact statement which combines the opening of file and processing of file along with inbuilt exception handling. (Refer to Info box 5.3 given earlier where we have talked about the **with** statement.) The **with** statement will also close the file automatically after **with block** is over. You can use the **with** statement as :

```
with open(<filename>, <mode>) as <file handle>:
    # use pickle.load here in this with block
    # perform other file manipulation task in this with block
```

Notice that you need not mention any exception with the **with** statement, explicitly.

Please note that while writing onto file, the exceptions like "**File does not exist**" or the EOF error do not arise as most write modes create the file if it does not exist already and you can write onto them as long as you want, *i.e.*, there is no restricting EOF marker for writing.

In the program below, we have used both the **try..except** block (in *programs 5.11* and 5.12) and the **with** statement (in programs *5.13* and *5.14*) for working with the files. Now consider the following program that is reading from the file you created in the previous program.

5.11 Write a program to open the file **Emp.dat** (created in program 5.8), read the objects written in it and display them.

Program

```
import pickle
#declare empty dictionary object; it will hold the read record
emp = { }
empfile = open('Emp.dat', 'rb')          # open binary file in read mode
#read from the file
try:
```

*Read repeatedly; when no more records, it will give EOF and take you to **except** block where file is closed.*

```
  while True :  # it will become False when the end of file is reached (EOF exception).
    emp = pickle.load(empfile)
    print(emp)
except EOFError:
  empfile.close()                        # close file
```

This statement would unpickle the object being read from file

You can now use the object in usual way (e.g., we printed its contents)

The statements in this block will get executed when EOF has occurred, i.e., the file will be closed on reaching EOF

The output produced by the above program will be :

```
{'Empno': 1201, 'Name': 'Anushree', 'Age': 25, 'Salary': 47000}
{'Empno': 1211, 'Name': 'Zoya', 'Age': 30, 'Salary': 48000}
{'Empno': 1251, 'Name': 'Simarjeet', 'Age': 27, 'Salary': 49000}
{'Empno': 1266, 'Name': 'Alex', 'Age': 29, 'Salary': 50000}
```

It is returning the same data as we wrote in the previous program (Compare with the data of the previous program)

Now that you have an idea of how to read and write in binary files, let us write a few more programs before we talk about searching in and updating the binary files.

5.12 Write a program to open file created and used in programs 5.9 and 5.10 and display the student records stored in it.

gram

```
import pickle
stu = {}                          # declare empty dictionary object to hold read records
fin = open('Stu.dat', 'rb')       # open binary file in read mode
# read from the file
try:
    print("File Stu.dat stores these records")
    while True :                  # it will become False upon EOF
        stu = pickle.load(fin)    # read record in stu dictionary from fin file handle
        print(stu)                # print the read record
except EOFError:
    fin.close()                   # close file
```