



# PICKLE MODULE

# Serialization / Pickling

- ▶ Serialization is a process of converting Python object or data structures hierarchy into **byte streams** so that it can be written into a file.
  - ▶ A byte stream is, a stream of bytes – one byte is composed of 8 bits of zeros and ones.
  - ▶ These byte streams can then be stored or transferred easily.
  - ▶ Video games might be the most intuitive example of serialization's usefulness.
  - ▶ Serialization is the process of converting object state into a format that can be transmitted or stored. The serialization changes the object state into series of bits.
- 

# De-Serialization / Unpickling

- ▶ Is the inverse of Pickling where a byte stream is converted into an object hierarchy.
  - ▶ Unpickling produces the exact replica of the original object.
  - ▶ The object state could be reconstructed later in the opposite process, called deserialization.
- 


# pickle module

- ▶ Python's module to achieve serialization/de-serialization.
- ▶ In order to work with the pickle module, you must first import it in your program.

*import pickle*

- ▶ Then you may use **dump()** and **load()** methods to write and read from an open binary file respectively.

# Process of working with binary file

- ▶ Import *pickle* module.
  - ▶ Open binary file in the required mode.
  - ▶ Process binary file by writing/reading objects using appropriate methods.
  - ▶ Once done, close the file
- 

# Creating/Opening/Closing Binary Files

- ▶ A binary file is opened in the same way as any other file, but make sure to use “b” with file modes to open a file in *binary mode*.

- ▶ `file=open("stu.dat","wb")`

OR

- ▶ `file=open("stu.dat","rb")`



Notice 'b' is used with the file modes

- ▶ `file.close()`

# Writing onto a Binary file – Pickling

- ▶ In order to write an object on to a binary file, use **dump()** function of *pickle module*.
- ▶ **pickle.dump(<object-to-be-written>, <file handle-of-open-file>)**

```
import pickle
```

```
with open('record1.txt', 'wb') as fileobj:  
    eno = int(input('Enrollment No: '))  
    name = input('Name: ')  
    student = [eno, name]  
    pickle.dump(student, fileobj)
```

```
€ ª +
```

```
] "(K &-aiysha"e.
```

# Reading from a Binary file – UnPickling

- ▶ In order to write an object on to a binary file, use `load()` function of *pickle module*.
- ▶ `<object>=pickle.load(<file handle-of-open-file>)`

---

```
import pickle
with open('record1.txt', 'rb') as fileobj:
    data = pickle.load(fileobj)
    print('Roll: %d' %(data[0]))
    print('Name: %s' %(data[1]))
```

```
>>>
=====
=====
Roll: 1
Name: aiysha
>>> |
```



# Binary File Opening Modes

Mode	Description
rb	Open file in binary mode for reading only. The file pointer stands at the beginning of the file.
rb+	Open file in binary mode for both reading and writing. The file pointer stands at the beginning of the file.
wb	Open file in binary mode for writing only. It creates the file if it does not exist. If the file exists, then it erases all the contents of the file. The file pointer stands at the beginning of the file.
wb+	Open file in binary mode for both reading and writing. It creates the file if it does not exist. If the file exists, then it erases all the contents of the file. The file pointer stands at the beginning of the file.
ab	Open file in binary mode for appending data. Data is added to the end of the file. It creates the file if it does not exist. The file pointer stands at the end of the file.
ab+	Open a file in binary mode for reading and appending data. Data is added to the end of the file. It creates the file if it does not exist. The file pointer stands at the end of the file.