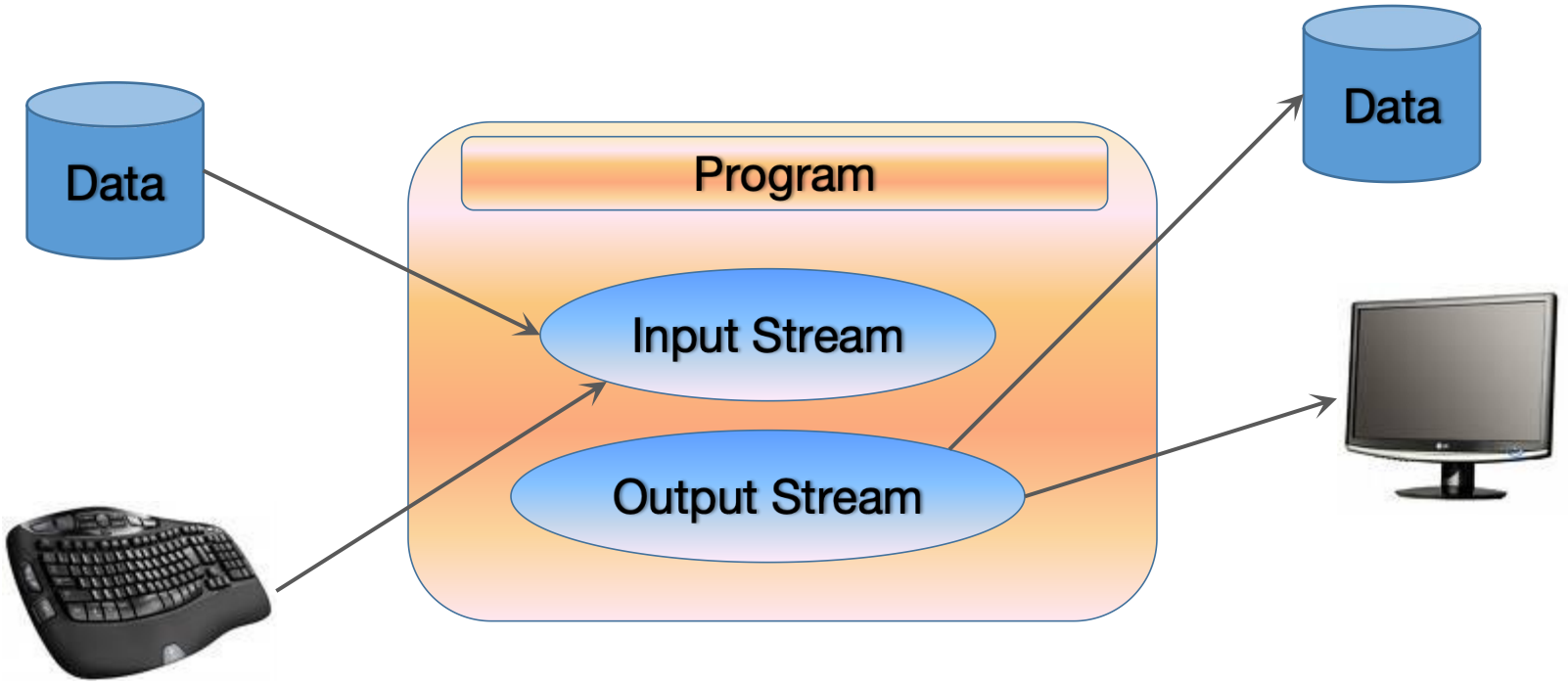
File Handling

Stream

- A stream is simply a sequence of bytes that flows into or out of our program.
- It's an abstract representation of an input or output device.



Input & Output Streams

Input Stream : Any stream, the data can be read from.

Output stream : Any stream, the data can be written to.

Types of Streams :

Binary or Byte Streams: While writing data to a Binary Stream, the data is written as a series of bytes, exactly as it appears in the memory. No data transformation takes place.

Character Streams: With character streams, program reads and writes Unicode characters, but the stream will contain characters in the equivalent character encoding used by the local computer.

Working with File

- ✓ Files are storage compartments on your computer that are managed by operating system
- ✓ Python built-in `open()` function creates a Python file object, which serves as a link to a file residing on your machine

```
fileObj = open(file_name, access_mode,encoding)
```

```
>>> # Creating a new file and opening it in writing mode
>>> newFile = open("test.txt" , "w",)
```

```
>>> # Opening a file in a read mode
>>> newFile = open("test.txt","r")
```

Mode of Opening a File

Mode	Meaning
r	Opens a file for reading only.
w	Create a file for writing only.
a	Append to a file.
rb	Open a binary file for reading.
wb	Create a binary file for writing.
ab	Append to a binary file.
r+	Opens a file for read/write.
w+	Create a file for write/read.
a+	Append or create a file for read/write.
rb+	Open a binary file for read/write.
wb+	Create a binary file for read/write.
ab+	Append or create a binary file for read/write.

File Handling

Attribute	Description
file.closed	Returns true if file is closed, false otherwise
file.mode	Returns access mode with which file was opened
file.name	Returns name of the file

Common File Methods

✓ write()

- The write() method writes any string to an open file. It is important to note that Python strings can have binary data and not just text.

  Syntax: fileObj.write(string);

✓ read()

- The read() method reads a string from an open file.

  Syntax: fileObj.read([count]);

✓ close()

- The close() method of a file object flushes any unwritten information and closes the file object, after which no more writing can be done.
- Python automatically closes a file when the reference object of a file is reassigned to another file.

  Syntax: fileObj.close();

Python pickle Package

- Pickle is used to serialize and deserialize a python object structure. Any object on python can be pickled so that it can be saved on disk.
- Pickle first serialize the object and then converts the object into a character stream so that this character stream contains all the information necessary to reconstruct(deserialize) the object in another python script.

- Use `pickle.dump(object,file,protocol)` function to store the object data to the file.
- Use `pickle.load(file)` to retrieve pickled data.

Python pickle Example (pickling)

```
import pickle

data = {
          'a': [1, 2.5, 3, 4 + 6j],
          'b': ("character string"),
          'c': {None, True, False}
       }

file=open('data.pickle', ‘wb')

pickle.dump(data, file, pickle.HIGHEST_PROTOCOL)

file.close()

print("data written")
```

Python pickle Example (unpickling)

```
import pickle

file=open('data.pickle', 'rb')

print("data from file:")

data = pickle.load(file)

file.close()

print(data)
```

Thank You :)