JSON file

JSON stands for Java Script Object Notation.
JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

To work with json file python provides a predefined module called “json”.
Json module provides encoders and decoders.
Encoders are used to convert python data into json format
Decoders are used to convert json data into python format
“json” is a default module which comes with python software.

In json data is stored in key and value format.

`json.dump(obj, fp)`
This method is used to write python object into json format inside file
Supports the following objects and types by default:

Python	JSON
dict	object
list, tuple	array
str	string
int, float, int- & float-derived Enums	number
True	true
False	false
None	null

Creating json file

import json

```
with open("e:\\products.json","w") as jsonfile:
    prod={101:['mouse',200],
          102:['keyboard',1500],
          103:['monitor',5000]}
    json.dump(prod,jsonfile)

print("data is written")
```

Output:
data is written

json.load(fp)
This method or function converts json data into python object

Performs the following translations in decoding by default:

JSON	Python
object	dict
array	list
string	str
number (int)	int
number (real)	float
true	True
false	False
null	None

Example:

```
# Reading data from json file

import json

with open("e:\\products.json","r") as jsonfile:
    prod=json.load(jsonfile)
    print(prod)
    print(prod['101'])
    print(prod['102'])
```

Output:
{'101': ['mouse', 200], '102': ['keyboard', 1500], '103': ['monitor', 5000]}
['mouse', 200]
['keyboard', 1500]

Binary file
Binary file is a collection of bytes.
Binary file allows only bytes data.
Example: image, audio, video,…

Open("file1","w") → w+t
Open("file1","r") → r+t
Open("file1","wb") → w+b
Open("file1","rb") → r+b

Example:
```
# Creating binary file

f=open("e:\\file1","wb")
b1=bytes(range(65,69))
f.write(b1)
f.close()
```

Example:
```
# Reading data from binary file
f=open("e:\\file1","rb")
b=f.read()
print(b)
f.close()
```

Output:
b'ABCD'

Example:
```
# Create copy of image

f1=open("e:\\django.png","rb")
f2=open("e:\\django1.png","wb")
b=f1.read()
```

```
f2.write(b)
f1.close()
f2.close()
print("file copied")
```

Output:
file copied

pickle module

pickle module is a default module which comes with python software. The pickle module implements binary protocols for serializing and de-serializing a Python object structure. “Pickling” is the process whereby a Python object hierarchy is converted into a byte stream, and “unpickling” is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as “serialization”, “marshalling,” or “flattening”; however, to avoid confusion, the terms used here are “pickling” and “unpickling”.

1. dump
2. load

Example:

```
# pickling or wirting python objects into binary file

import pickle

with open("e:\\file2.ser","wb") as file:
    pickle.dump(65,file)
    pickle.dump(1.5,file)
    pickle.dump("PYTHON",file)
    pickle.dump(1+2j,file)
    pickle.dump([10,20,30],file)

print("Data is saved")
```

Output:
Data is saved

Example:

```
# Reading data from binary file

import pickle

with open("e:\\file2.ser","rb") as file:
    a=pickle.load(file)
    b=pickle.load(file)
    c=pickle.load(file)
    d=pickle.load(file)
    e=pickle.load(file)
    print(a,b,c,d,e,sep="\n")
```

Output:

```
65
1.5
PYTHON
(1+2j)
[10, 20, 30]
```

Example:

```
import pickle
class Student:
    def __init__(self):
        self.__rollno=None
        self.__name=None
    def setData(self,r,n):
        self.__rollno=r
        self.__name=n
    def __str__(self):
        return f'{self.__rollno},{self.__name}'


stud1=Student()
stud1.setData(101,"naresh")
print(stud1)
with open("e:\\stud.ser","wb") as file:
    pickle.dump(stud1,file)

with open("e:\\stud.ser","rb") as file:
```

```
    stud2=pickle.load(file)
    print(stud2)
```

Output:

```
101,naresh
101,naresh
```

Regular Expression (re module)