YAML

Full form of YAML :-

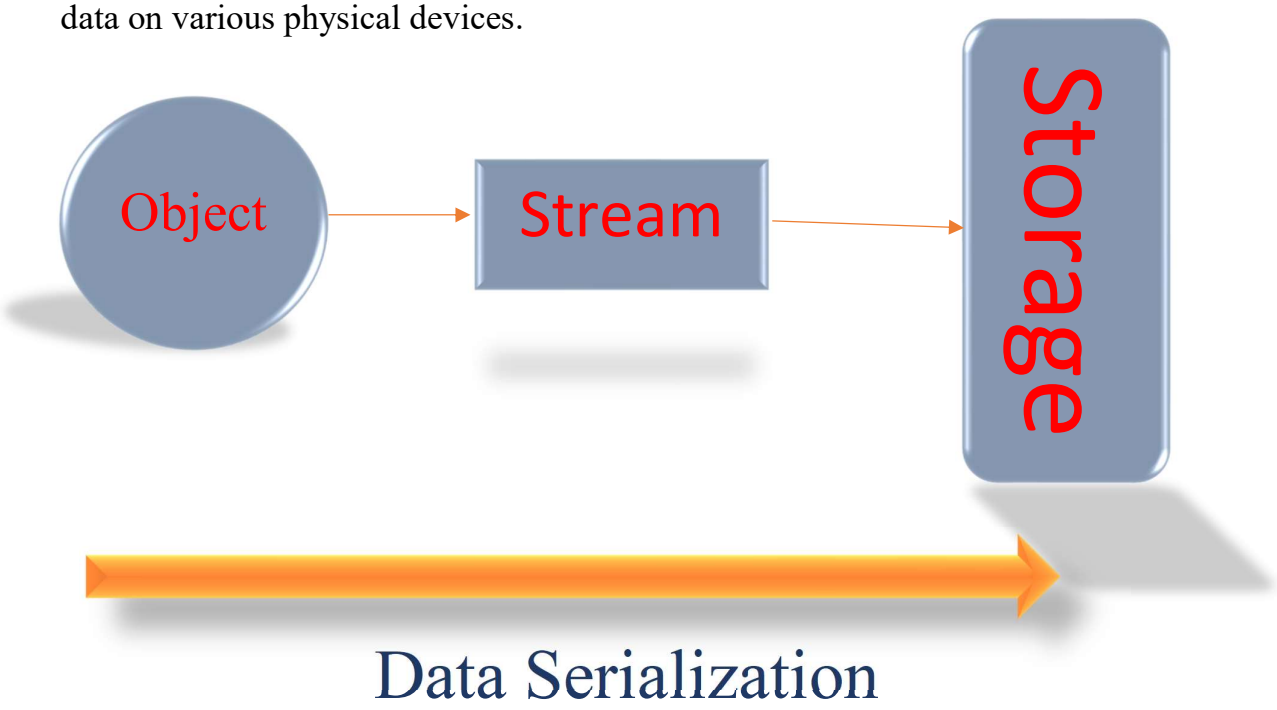
Previously :- It stands for yet another markup language.

Now :- It stands for YAML ain't markup language.

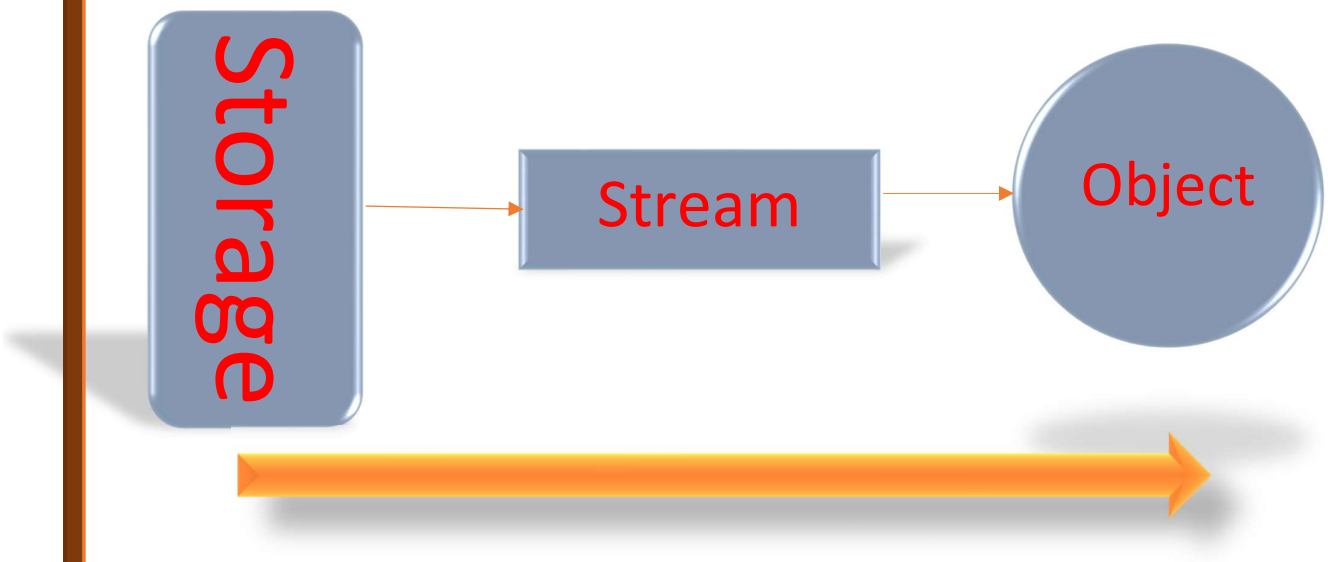
YAML is simple human readable language that can be used to represent data.

1. YAML is not a programming language.
2. It is actually a data format used to exchange data.
3. It is similar to XML and JSON.
4. In YAML you can store only data, and not commands.

Data Serialization :- It is a process of converting the data objects present in complex data structure into a byte stream for storage. It is used to transfer the data on various physical devices.



Data Deserialization :- Deserialization is the reverse process Data serialization, reconstructing the object from taking data from storage and stream (series of bytes).



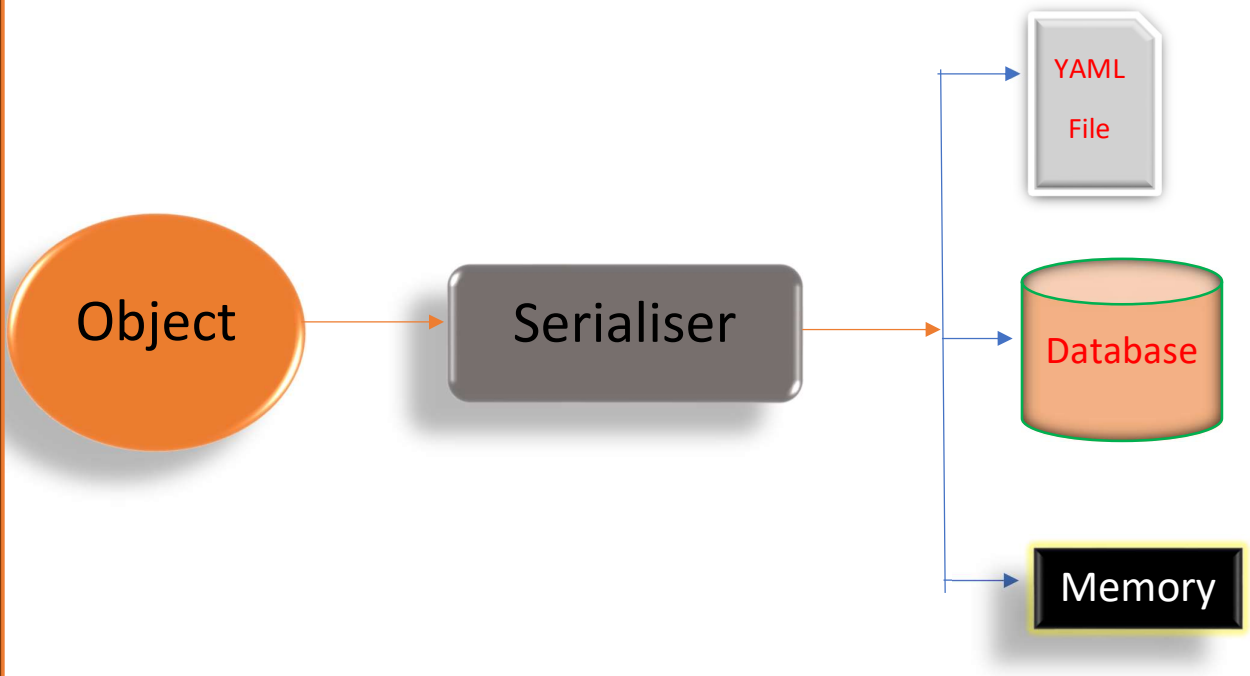
Deserialization

Object = code + data

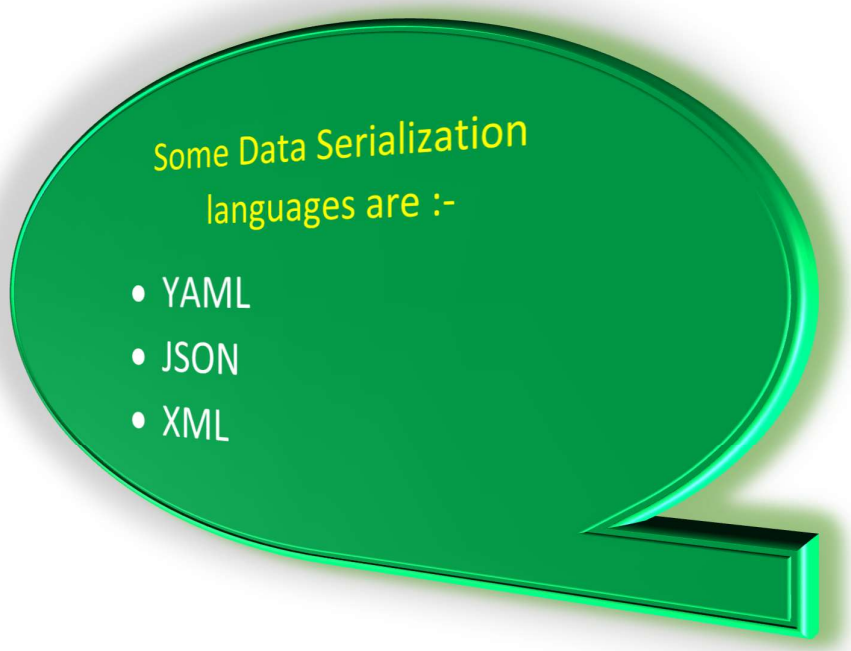
In simple language :-

We have an object we can put it into serialiser, serialiser convert the object into stream of bytes then we can store that in our YAML file or database or memory.

"Serialization is basically a process of converting the data object into series of bytes that saves the state of the object in a form that is easily transmittable"



- If we want to represent the object in a file that we can read, code or modify those files are known as data serialization files.
- Language used in data serialization files is known as data serialization language.
- Language used to represent the data in text format is called data serialization language.



YAML

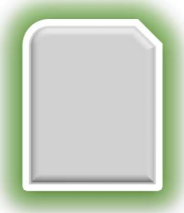
- YAML is Data serialization language
- Used to store data
- Used in configuration files like docker, Kubernetes etc.
- It is also used for logs and caches etc.

Benefits of YAML

- Simple and easy to read
- It has a strict syntax – Indentation is important.
- Easily convertible to JSON, XML
- Most languages use YAML
- More powerful when representing complex data
- Various tools available like parser etc.
- Parsing is easy (parsing means reading the data)

Creating a YAML File

➔ Extension of YAML file is either **.yaml** or **.yml**



hello.yaml

Or

hello.yml

In YAML we can store key value pairs

- It means a key is pointing to some value.
- It is just a textual representation.
- We can use this file and convert a HashMap out of it.

Representation of key value pair :-

Key is pointing to some value

Example:-

```
"apple" : "I am a red fruit"
1: "this is gautam's roll number."
```

In above example apple is a key which is pointing to value “I am a red fruit” and 1 is a key which is pointing to value “this is gautam’s roll number”.

Comments in YAML

- # is used to comment any line in YAML
- YAML does not provide multiline comments

Example :-

```
# lists
```

Lists in YAML

- We can represent list in YAML using –

Example :-

```
# lists
- apple
- mango
- banana
- Apple
```

- YAML is case sensitive so apple and Apple is two different things.

Block style in YAML

- Blocks style in YMAL are indented by spaces.
- If we do not indent every line by spaces it may give error
- Spaces or indentation are extremely important in YAML

Representation of block style:-

Cities is pointing to some lists

```yaml
cities:
 - new delhi
 - mumbai
 - gujrat
```

Checking YAML Syntax

- We check YAML syntax using yamllint website

Differentiate between documents

- We can differentiate between different document using ---

Example :-

```yaml
"apple" : "I am a red fruit"
1: "this is gautam's roll number."

---
# lists
- apple
- mango
- banana
- Apple
---
# block style
cities:
 - new delhi
 - mumbai
 - gujrat
```

"YAML is a collection of 0 or more documents"

- To end Document we can use …

Storing data in a single line

- As we know YMAL files has indentation problem so we can store data in single line to avoid indentation problem.

For Example :-

```
# for block style
cities: [new delhi, mumbai, gujrat]
---
# for key value pairs
{mango: "Yellow fruit", age: 56}
---
```

Data types in YAML

- As we all know datatype means type of data we are using like string, integer, float Boolean etc.
- In YAML we do not have to specify datatype to the key, YAML automatically detect the data type of value we are storing in key.
- But if we want We can explicitly tells YAML the data type of value using double explanation mark with data type

1. **Strings** :- We can store strings in key by 3 ways :-
   - **i.** Without using quotes
   - **ii.** With using double quotes
   - **iii.** With using single quotes

Example:-

```
# String datatype
myself: Gautam Jha
fruit: "apple"
job: 'Devops Engineer'
```

- In YAML one line is one entry and second line is another entry it means we can not put two different lines value in one key directly. So if we want to store two lines value in single key we have to use vertical bar (|).

Example:-

```
bio: |
 Hello my name is Gautam Jha.
 I am learning Devops with kunal kushwaha.
```

- If we want to write a single line in multiple lines we can use angle bracket/ greater than sign (>), this case may arise

Example :-

```
# write a single line in multiple lines
message: >
 this will
 be treated as
 one single line.
```

The above code is same as

```
# same as
message: this will be treated as one single line.
```

Specifying type explicitly :-

```
explicit string: !!str here string is explicity defined data type fot this key
```

2. **Integer :-** we can store integer in key as :-

```
number: 5478
```

For example:-

```
zero: !!int 0
PositiveNum: !!int 45
negetiveNum: !!int -96
binaryNum: !!int 0b1101
octalNum: !!int 06547
hexa: !!int 0x45
commaValue: !!int +540_000 #540,000
```

3. **Float :-** we can store float as :-

```
marks: 95.74
```

Specifying type :-

```
#floating point numbers
marks: !!float 56.89
infinite: !!float .inf
not a num: .nan
```

4. **Boolean:-** we can store Boolean value as :-

```
booleanValue: No # n, N, false, False, FALSE
# same for true --> yes, y, Y
```

Specifying type :-

```
booleanValue: !!bool No
```

5. **Null :-** some time we have to define null then we can use :-

```
# null
surname: !!null Null #or null, NULL, ~
# sometimes keys may also be null
~: this is a null key.
```

6. **Dates and time :-** we can also specify date an time in YAML file, if no time stamp is given then YAML automatically assume UTC time zone.

```
# date and time
Date: 2022-01-28
India time: 2022-01-28T02:59:43.10 +5:30
No time zone: 2022-12-15T02:59:43.10
```

Advance Datatypes

1. **Lists/sequence :-** List datatype in yaml is known as sequence.

- Items in sequence must be indented
- Different item contain minus sign (-) to differentiate between different item.
- If we want to explicitly define datatype of sequence we have to use !!seq
- We can put item of sequence in different lines as well as in same line.

Example :-

```
# sequence item in different line
student: !!seq
 - marks
 - name
 - roll_no

# sequence item in same line
cities: [new delhi,mumbai]
```

Sparse Sequence :- when some of the values of sequence is empty that sequence is called sparse sequence.

Example :-

```
#sparse sequence
sparse seq:
 - hey
 - how
 -
 - Null
 - sup
```

Nested Sequence :- when items are present inside an iten in a sequence is called Nested sequence

Example :-

```
# nested sequence
nested_seq:
 - Fruit
   - Apple
   - banana
   - Grapes
 - Names
   - Kunal kushwaha
   - Gautam Jha
   - Rohit
 - Vegitables
   - Capsicum
   - Carrot
   - Cabbage
```

2. **Maps :-** key value pairs are called maps
   Example:-

```
"apple" : "I am a red fruit"
1: "this is gautam's roll number."
```

Nested Mapping :- map within a map is called nested map

Example :-

```yaml
# nested maping
name: Gautam Jha
role:
  age: 78
  job: student

# same as
name: Gautam Jha
role: { age: 78, job: student}
```

pairs :- when a single key has different values is called pairs, we can explicitly define pairs by (!!pairs)

example :-

```yaml
# pairs
who am i: !!pairs
 - job: student
 - job: teacher
# same as
who am i: !!pairs [job: student, job: teacher]
```

- This will be an array of hashtables.

Set :- set allow us to have unique values, we can't have duplicate names. We can explicitly define set as (!!set)

Example:-

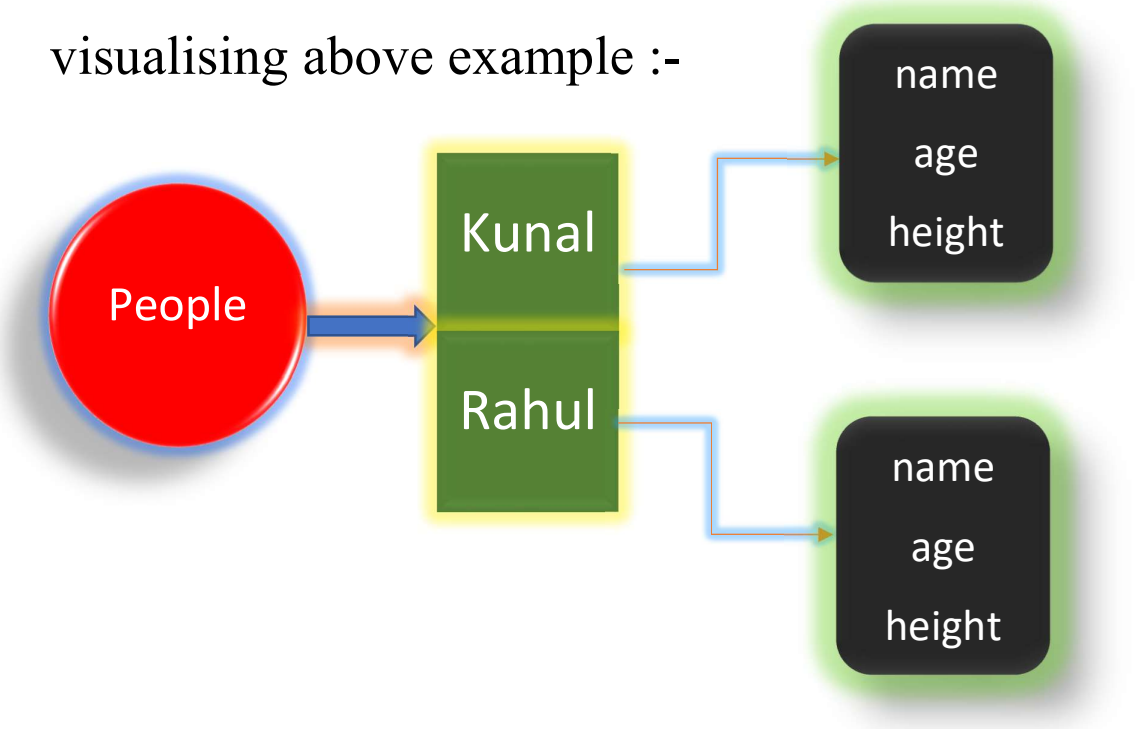
```yaml
names: !!set
 ? Kunal
 ? Gautam
 ? Rahul
 ? Apoorv
```

Dictionary :- When a particular key has a value as sequence, the entire sequence is represented as value of a key is called dictionary. We can explicitly define dictionary as (!!omap)

Example :-

```
# dictionary !!omap
people: !!omap
 - kunal:
   name: kunal kushwaha
   age: 78
   height: 678
 - rahul:
   name:  Rahul Op
   age: 50
   height: 456
```

visualising above example :-



Anchors :- Anchors are used to reuse some properties at different places.

- & symbol with the name of property are used on the properties which you want to reuse.
- << symbol with star * and properties name are used where you want to reuse the property.

Example :-

```
likings: &likes
 fav fruit: mango
 dislikes: Grapes

person1:
 name: Kunal kushwaha
 <<: *likes


person2:
 name: Gautam Jha
 <<: *likes


person3:
 name: Rahul
 <<: *likes
```

➔ fav fruit and dislikes are same for all 3 person as mango and grapes respectively.

We can also override the property of any person

For example :- I want to override the dislike of gautam jha as apple.

```
likings: &likes
 fav fruit: mango
 dislikes: Grapes

person1:
 name: Kunal kushwaha
 <<: *likes


person2:
 name: Gautam Jha
 <<: *likes
 dislikes: apple


person3:
```

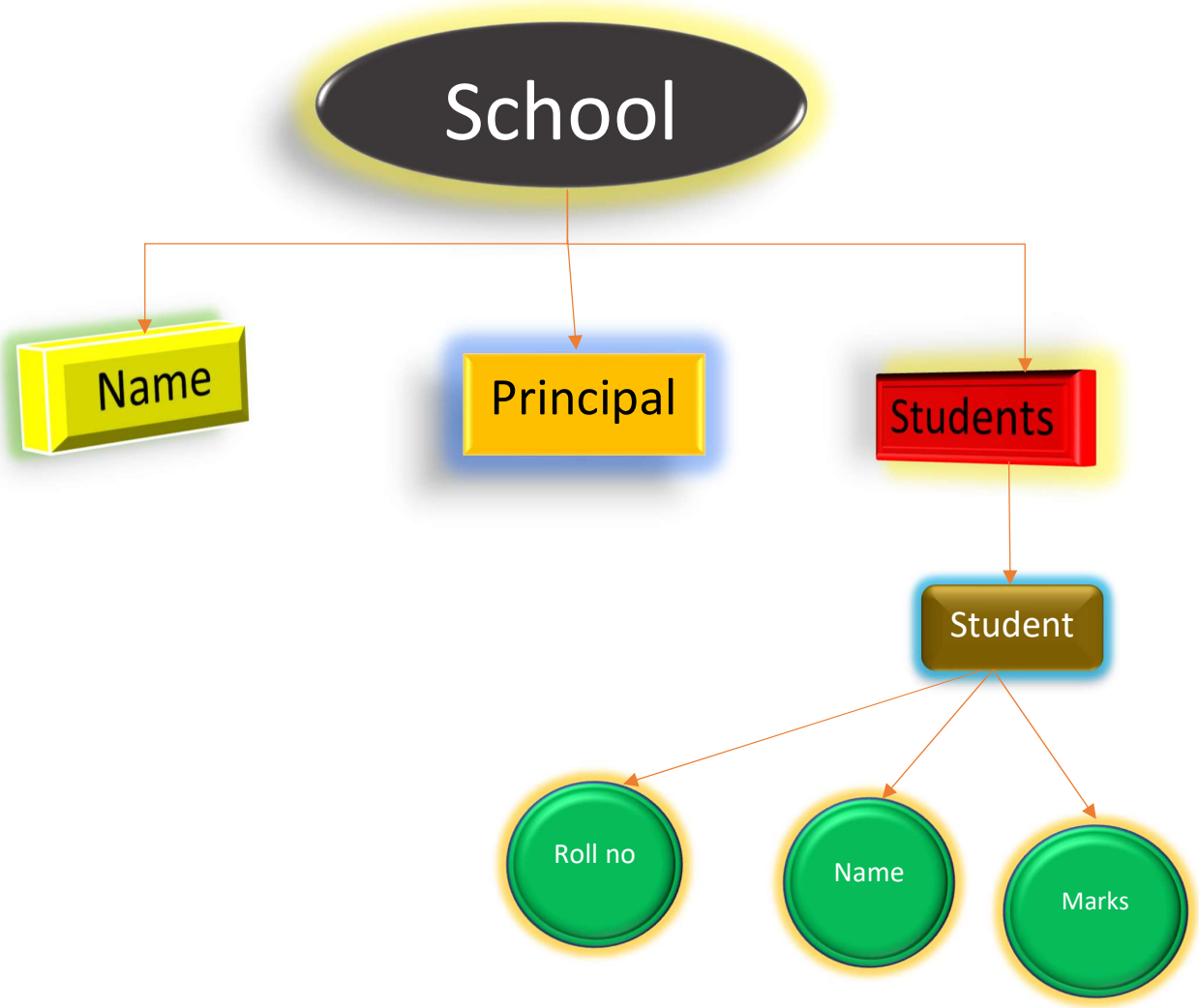
```
name: Rahul
<<: *likes
```

Real World Example

Let say we want to store data in various files like XML, JSON, YAML.

For example we have school and school has name, who is the principle and list of students.

Every single students has roll no., name, and marks.



Above diagram is a data we have to store it in a form of code.

➔ **Store in XML (extensible markup language) file:-**

- XML is not readable by humans
- Used to store and data
- XML has various uses

```xml
<?xml version="1.0" encoding="UTF-8"?>
<School name="DPS" principal = "Someone">
    <Students>
        <Student>
            <rno>11</rno>
            <name>"Gautam"</name>
            <marks>84</marks>
        </Student>
    </Students>
</School>
```

➔ **Store in JSON (JavaScript Object Notation) file.**

- It is heavily used in JavaScript.
- It is most popular serialization language.
- JSON is much human readable than XML

```json
{
    "School": [
        {
        "name": "DPS",
        "principal": "Someone",
        "Students":[
            {
                "rno": 11,
                "name": "Gautam Jha",
                "marks": 53
            }
        ]
        }
    ]
}
```

➔ **Store in YAML (YAML ain’t markup language) file**

```yaml
---
School:
- name: DPS
  principal: Someone
  Students:
  - rno: 11
    name: Gautam Jha
    marks: 53
```