YAML

* YAML

→ Previously it was known as "Yet Another Markup language".

→ Now it is known as "YAML Ain't Markup language".

→ YAML is not a programming language. It is basically a data format used to exchange data.

→ Similar to XML and JSON datatypes.

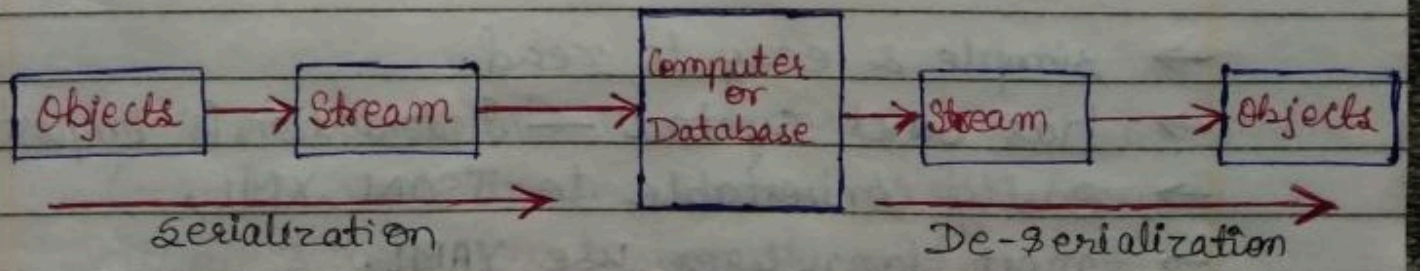
→ Basically, it is a simple human readable language that can be used to represent data.

→ used to store some information about configuration

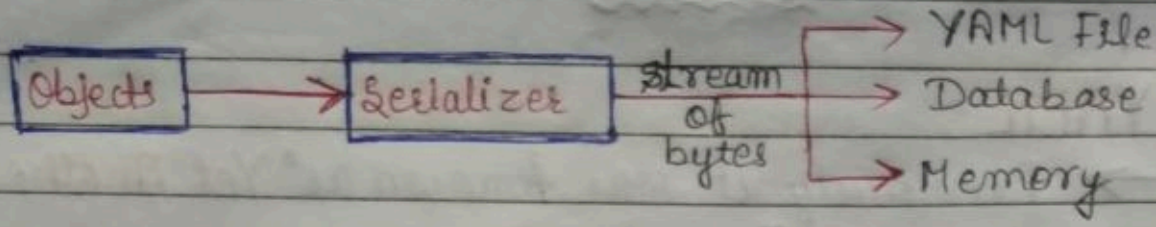
→ In YAML, you can store only data, not commands.

* Data Serialization

→ It is a process of converting the data objects (that is present in some complex data structure) into a stream of storage that can be used to transfer this data on your physical devices.



- Objects — combination of code & data.
  It is basically a data storage unit.
- Stream — chunks of data



→ Serialization is basically a process of converting the data object (which is a combination of code & data) into series of bytes that saves the state of the object in a form that is easily transmittable. And the reverse of this is known as De-serialization

Data - Serialization language :- YAML, JSON, XML.

** NOTE: Markup languages are used to store only documents. But in YAML, you can store document as well as object data. That's why, YAML is known as YAML Ain't Markup Language.

→ YAML is used in configuration files in Docker, K8s.. in logs, caches, etc.

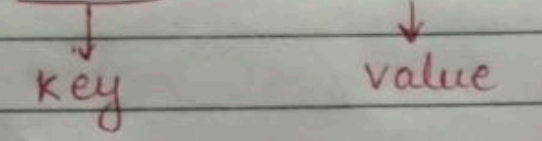
** Benefits of YAML

→ simple & easy to read.
→ has strict syntax — Indentation is important.
→ easily convertable to JSON, XML.
→ most languages use YAML.
→ more powerful when representing complex data.
→ various tools available like Parsers
→ Parsing is easy (Parsing means reading the data).

→ The YAML file should have .yaml or .yml as the extension.

→ In YAML, we store in key-value pairs.
i.e, key: value

Ex → "apple": "Red Fruit"



→ List : (hyphen + space) used to begin a new item in a list.

Ex →

```
- apple
- mango
- banana
```

→ YAML is case-sensitive.

→ In YAML, we use spaces for indentation, not tabs.

Ex →

```
cities:
    - New Delhi
    - Mumbai
    - Gujarat
```

→ Multiple documents are separated with 3 hyphens (---) and to mark end of the document use 3 dots (...) in the end.

Ex →

```
"apple": "Red fruit"
---
- apple
- mango
- banana
---
cities:
  - New Delhi
  - Mumbai
...
```

← Now, new type of document starts

← represent end of the document

** cities:
- New Delhi
- Mumbai

OR cities: [New Delhi, Mumbai]

** apple: "red fruit"
mango: "yellow fruit"

OR {apple: "red fruit", mango: "yellow fruit"}

→ YAML supports single line comments.
this is single line comment.

→ YAML does not support multi line comments.
this is
a multi
line comment

→ ways to represent strings :
① fruit: apple
② fruit: "apple"
③ fruit: 'apple'

* If you want to add a string that contains multiple lines then we use | .

Ex:-

```
lines: |
This is first line.
This is second line
```

* write a single line in multiple line, use > .

Ex:-

```
message: >
this will
all be
in one single line
```

same as

```
message: this will all be in one single line
```

* Integers, float, Boolean

```
* number: 5473        # integer
* marks: 98.76        # float
* booleanValue: No    # boolean
  (No/n/N/false/False/FALSE → for False)
  (YES/yes/Y/y → for True)
```

* Specify the type

```
* zero: !!int 0
* positiveNum: !!int 45
* negativeNum: !!int -45
* binaryNum: !!int 0b11001
```

* octalNum: !!int 06574
* hexadecimalNum: !!int 0x45
* commaValue: !!int +540_000 # 540,000
* exponential numbers: 6.023E56

* Floating Point Numbers

* marks: !!float 56.89
* Infinite: !!float .inf
* not a num: .nan

* Boolean
* booleanValue: !!bool No # for false

* String
* message: !!str this is a string

* Null
* surname: !!null Null # or null /NULL /~
* ~: this is a null key

* Dates and Time
* date: !!timestamp 2022-02-04
* india time: 2022-02-04T02:59:43.10 +5:30
* no time zone: 2022-02-04T02:59:43.10

** List Datatype

Ex:-

```
student: !!seq
  - name
  - marks
  - roll_no
```

* Sometime, some of the keys of the sequence will be emty, this is known as Sparse Sequence.

Ex:

```
sparse seq:
  - hey
  - how
  -
  - Hello
```

** Nested Sequence – list inside list

```
-
  - mango
  - apple
  - banana
-
  - marks
  - roll_no
```

* key: value pairs are called maps. For this we use, !!map

** Nested Mappings – map within map

Ex

```
name: Kunal Kushwaha
role:
  age: 23
  job: student
```

** Pairs – keys may have duplicate values.

Ex:

```
pair example : !!pairs
  - job: Student
  - job: teacher
```

** Set - It allow you to have unique values (!!set)

Ex:
```
names: !!set
  ? Kunal
  ? Apoorv
  ? Rahul
```

** Dictionary — For dictionary we use !!omap

Ex
```
people: !!omap
  - Kunal:
      name: Kunal Kushwaha
      roll-no: 7
  - Rahul:
      name: Rahul Mishra
      roll-no: 56
```

** Reusing some properties using anchors

Ex
```
likings: &likes          # Give a name → Here: likes
  fav fruit: mango                          ↑ anchor
  dislikes: grapes

person1:
  name: Kunal Kushwaha
  <<: *likes
```

SAME AS

```
person1:
  name: Kunal Kushwaha
  fav fruit: mango
  dislikes: grapes
```

```
person2:
  name: Rahul
  <<: *likes
  dislikes: berries      # override the dislikes of Rahul
```

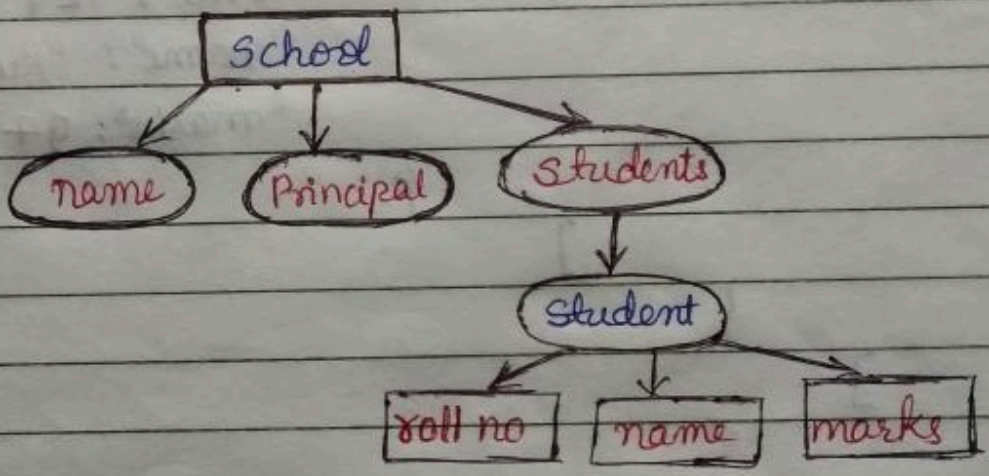
XML

→ XML stands for Extensible Markup language.
→ It is used to store data and share data across various platform.

JSON

→ JSON stands for JavaScript Object Notation
→ heavily used in JavaScript

Ex:-



→ Represention in XML

```
<?xml version="1.0" encoding="UTF-8"?>
<School name="DPS" principal="Someone">
    <students>
        <student>
            <rno>23</rno>
            <name>"Kunal"</name>
            <marks>94</marks>
        </student>
    </students>
</School>
```

→ Representation in JSON

```
{
    "School": [
        {
            "name": "DPS",
            "principal": "Someone",
            "Students": [
                {
                    "rno": 12,
                    "name": "Kunal",
                    "marks": 94
                }
            ]
        }
    ]
}
```

→ Representation in YAML

```
---
School:
- name: DPS
  principal: Someone
  Students:
  - rno: 12
    name: Kunal
    marks: 94
```