YAML Bootcamp

- **Intro**
- **Syntax**
- **Properties / datatypes**
- **YAML Toos**

What is Markup Language :-

Definition : A markup language is a system for annotating a document in a way that is syntactically distinguishable from the text. It provides a way to structure and format text in a document. –

Purpose : Used to store and format documents, define the structure of data, and separate content from presentation.

Child-Parent Relationship**: Markup languages often define a hierarchical structure where elements are nested within one another, creating a parent-child relationship.

What is YAML :-

- Yet Another Markup Language
- Now Yaml stands for **YAML aint markup Language**
- **Yaml** is Data format used to exchange Data like JSON.
- Human Readable Language , Used to Represent Data.
- We Can store only data, not commands
- It's a use for Data Serielisation.
- Use to represent data into code.
- Similar to XML , JSON .

Data Serielisation :-

- **Object => Stream of Bytes => DB , YAML file , Memory**
- Serielisation is a process of converting Data Object (Code + Data) into series of Bytes that save state of object.
- Data serielisation languages XML, YAML, JSON.

Uses :-

- Used in Files Configuration (docker , Kubernates) .
- Used in Cache and Logs

Benefits :-

- Simple and Easy to read (Human Readbale)
- Has Strict Syntax
- Indentation is Important
- Easily convertable to json and XML or
- Most Programing Languges uses YAML
- More powerfull when Representing Complex Data.
- Various tools Available : parsers etc
- Parsing is Easy

## YAML vs XML vs JSON ###

## YAML :-

Human Readability: YAML is designed to be easy to read and write for humans. It uses indentation to represent nested structures and is more readable than XML or JSON.

Syntax: Uses indentation, dashes, and colons to represent data structures. It supports complex data types and has a strict syntax for indentation.

-**Usage**: Commonly used for configuration files (e.g., Docker, Kubernetes), data exchange, and in applications where readability is crucial.

XML (EXtensible Markup Language) –

Human Readability: XML is also readable but often considered more verbose compared to YAML.

Syntax : Uses tags (e.g., `<tag>value</tag>`) to represent data structures. XML is more rigid and verbose, with every element needing opening and closing tags. –

Usage:-Frequently used in web services, document formatting, and data exchange where a strict schema is required.

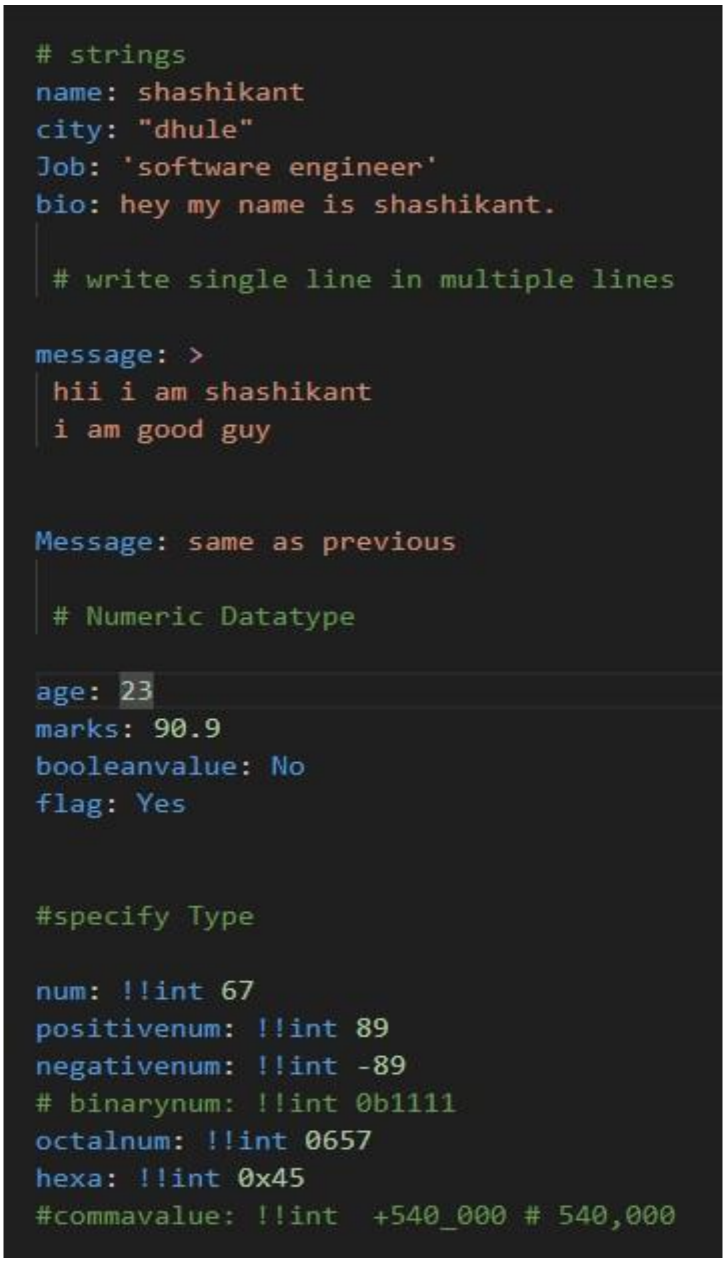
JSON (JavaScript Object Notation) –

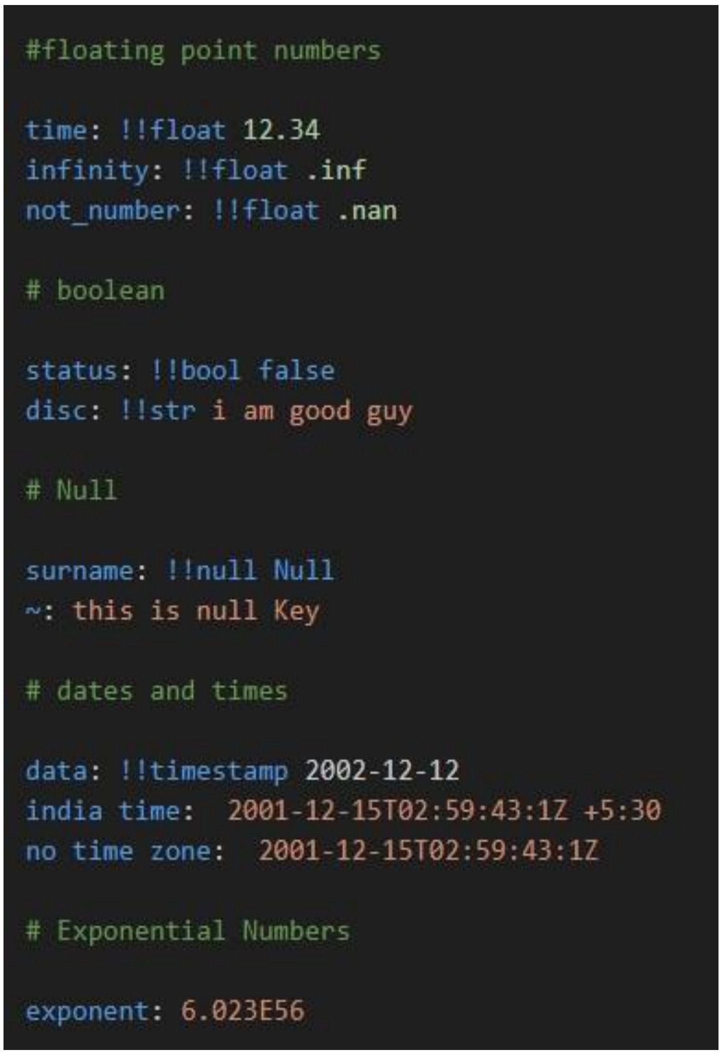
Human Readability: JSON is less verbose than XML and more compact but can be harder to read than YAML due to its lack of comments and reliance on brackets and commas.

Syntax:- Uses curly braces for objects and square brackets for arrays. JSON data structures are represented with key-value pairs and arrays.

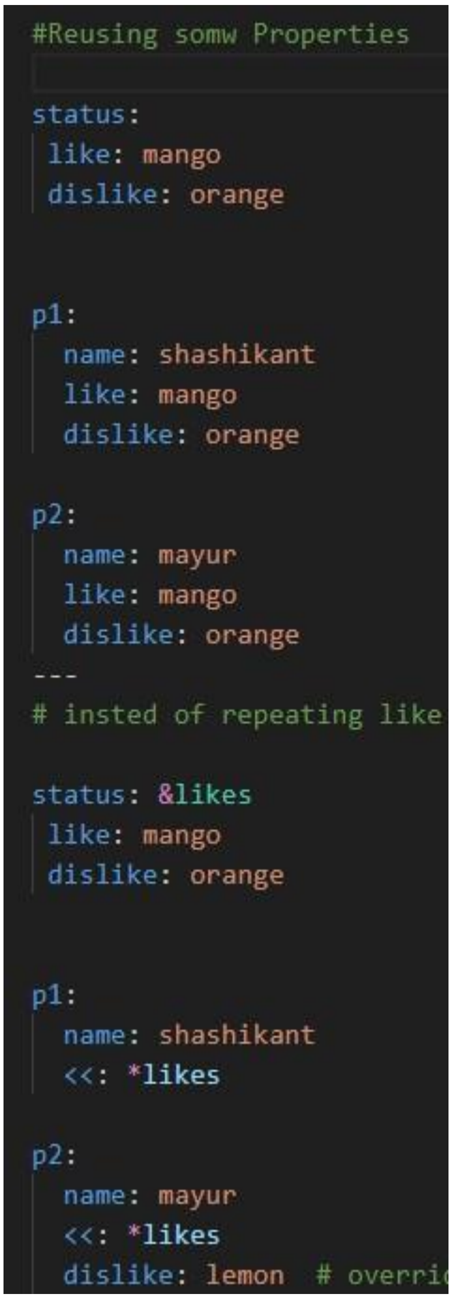
Usage: Widely used for data interchange between web services and applications, particularly in REST APIs and JavaScript environments

Datatypes

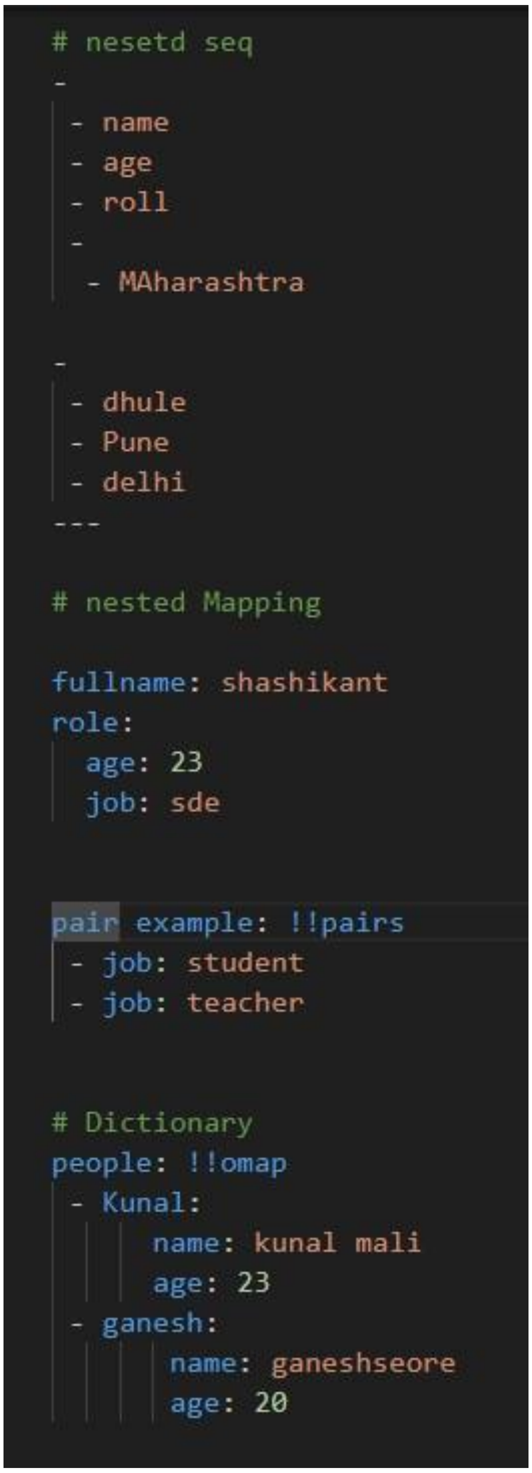

```yaml
# strings
name: shashikant
city: "dhule"
Job: 'software engineer'
bio: hey my name is shashikant.

 # write single line in multiple lines

message: >
 hii i am shashikant
 i am good guy


Message: same as previous

 # Numeric Datatype

age: 23
marks: 90.9
booleanvalue: No
flag: Yes


#specify Type

num: !!int 67
positivenum: !!int 89
negativenum: !!int -89
# binarynum: !!int 0b1111
octalnum: !!int 0657
hexa: !!int 0x45
#commavalue: !!int  +540_000 # 540,000
```


```yaml
#floating point numbers

time: !!float 12.34
infinity: !!float .inf
not_number: !!float .nan

# boolean

status: !!bool false
disc: !!str i am good guy

# Null

surname: !!null Null
~: this is null Key

# dates and times

data: !!timestamp 2002-12-12
india time:  2001-12-15T02:59:43:1Z +5:30
no time zone:  2001-12-15T02:59:43:1Z

# Exponential Numbers

exponent: 6.023E56
```

Advanced DataType

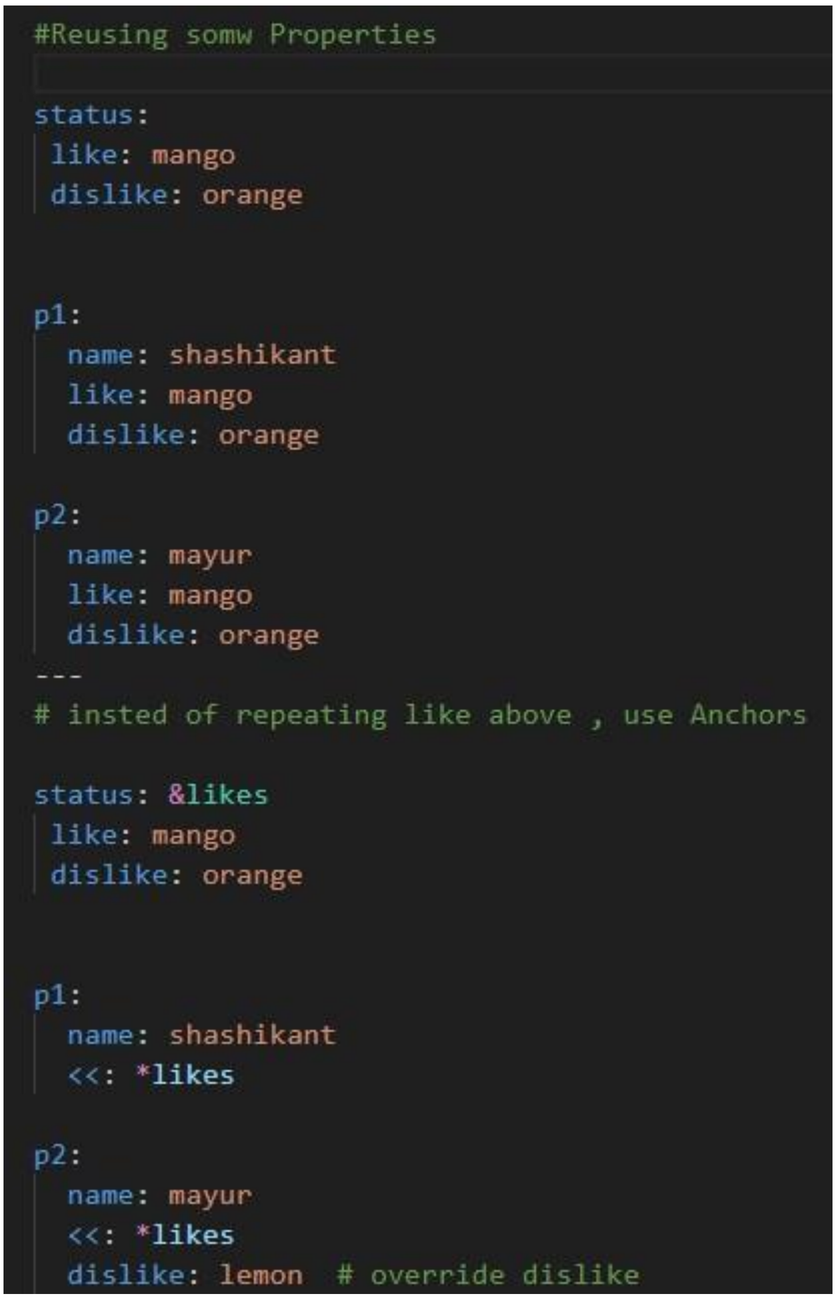


```
#Reusing somw Properties

status:
 like: mango
 dislike: orange


p1:
  name: shashikant
  like: mango
  dislike: orange

p2:
  name: mayur
  like: mango
  dislike: orange
---
# insted of repeating like

status: &likes
 like: mango
 dislike: orange


p1:
  name: shashikant
  <<: *likes

p2:
  name: mayur
  <<: *likes
  dislike: lemon  # overri
```

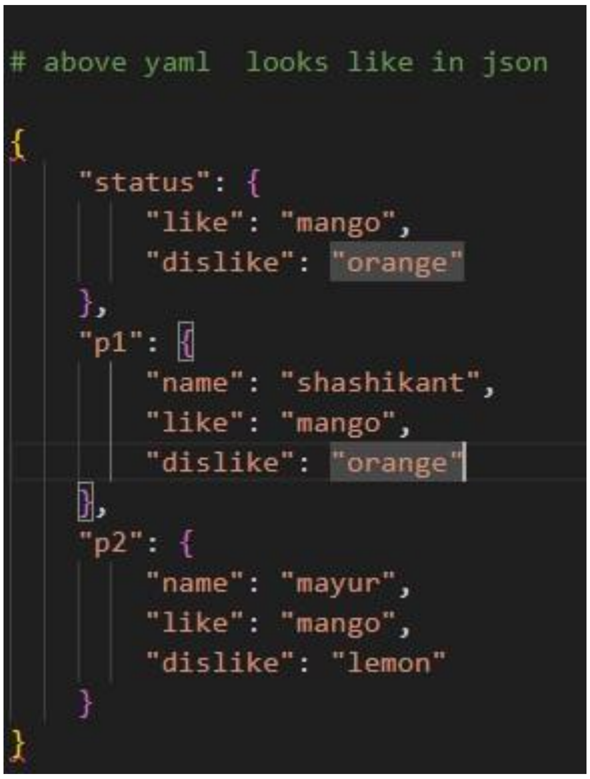


```
# nesetd seq
-
 - name
 - age
 - roll
 -
  - MAharashtra

-
 - dhule
 - Pune
 - delhi
---

# nested Mapping

fullname: shashikant
role:
  age: 23
  job: sde


pair example: !!pairs
 - job: student
 - job: teacher


# Dictionary
people: !!omap
 - Kunal:
      name: kunal mali
      age: 23
 - ganesh:
       name: ganeshseore
       age: 20
```

# Reusing Properties

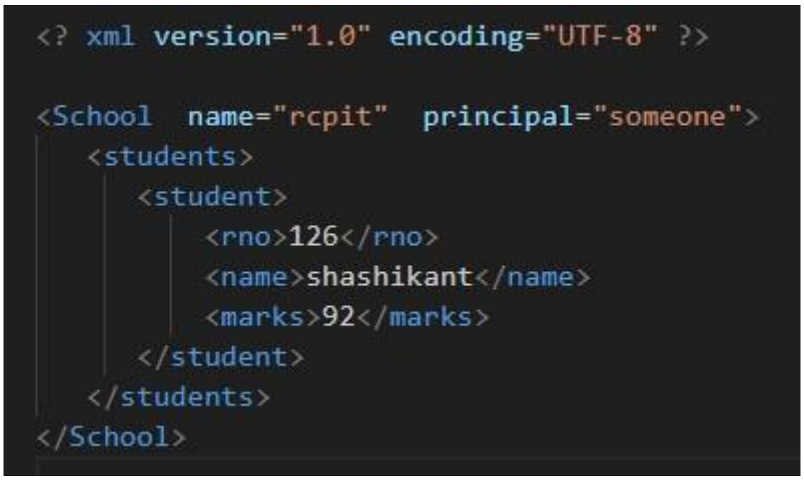


```yaml
#Reusing somw Properties

status:
 like: mango
 dislike: orange


p1:
  name: shashikant
  like: mango
  dislike: orange

p2:
  name: mayur
  like: mango
  dislike: orange
---
# insted of repeating like above , use Anchors

status: &likes
 like: mango
 dislike: orange


p1:
  name: shashikant
  <<: *likes

p2:
  name: mayur
  <<: *likes
  dislike: lemon  # override dislike
```

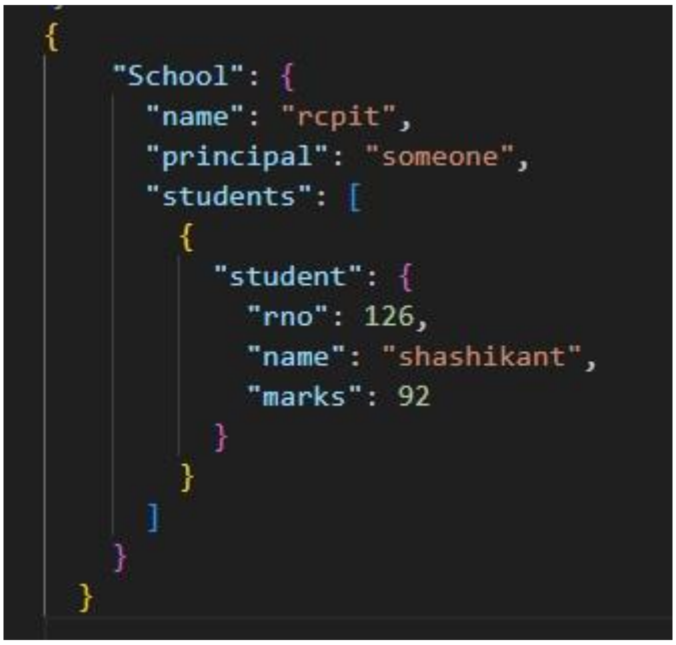


```
# above yaml  looks like in json

{
    "status": {
        "like": "mango",
        "dislike": "orange"
    },
    "p1": {
        "name": "shashikant",
        "like": "mango",
        "dislike": "orange"
    },
    "p2": {
        "name": "mayur",
        "like": "mango",
        "dislike": "lemon"
    }
}
```

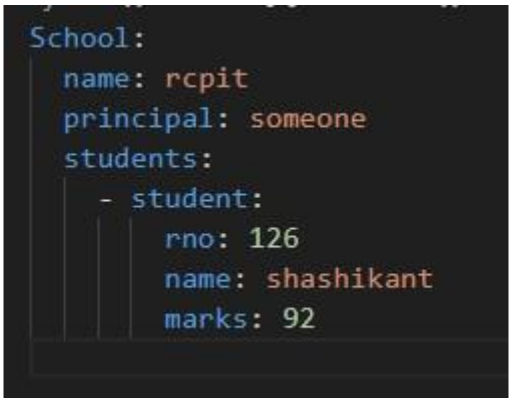
# XML | JSON |YAML



```xml
<? xml version="1.0" encoding="UTF-8" ?>

<School  name="rcpit"  principal="someone">
   <students>
      <student>
          <rno>126</rno>
          <name>shashikant</name>
          <marks>92</marks>
      </student>
   </students>
</School>
```



```json
{
    "School": {
      "name": "rcpit",
      "principal": "someone",
      "students": [
        {
          "student": {
            "rno": 126,
            "name": "shashikant",
            "marks": 92
          }
        }
      ]
    }
  }
```



```yaml
School:
  name: rcpit
  principal: someone
  students:
    - student:
        rno: 126
        name: shashikant
        marks: 92
```