anant-var /
DocKer

Code Issues Pull requests Actions Projects Wiki Security

DocKer / COA /

anant-var Create readme.md bd79e04 · now

Name	Name	Last commit date
..		
app.py	Add files via upload	5 minutes ago
readme.md	Create readme.md	now
students.yaml	Add files via upload	5 minutes ago

readme.md

YAML Guide

Introduction

YAML (YAML Ain't Markup Language) is a human-readable data serialization format commonly used for configuration files and data exchange. It is designed to be simple, readable, and easy to use compared to formats like JSON or XML.

Features

- Human-readable and easy to understand.
- Supports complex data structures.
- Uses indentation for hierarchy (no brackets or closing tags).
- Widely used in configuration management, CI/CD pipelines, and cloud infrastructure (e.g., Kubernetes, Ansible, Docker Compose).

Syntax Basics

Key-Value Pairs

YAML stores data as key-value pairs:

```yaml
name: John Doe
age: 30
city: New York
```

Lists (Arrays)

Lists are created using a dash (-):

```yaml
fruits:
  - Apple
  - Banana
  - Orange
```

Nested Objects

Indentation is used to define hierarchy:

```yaml
person:
  name: Alice
  age: 25
  address:
    city: San Francisco
    zip: 94107
```

Inline Objects and Lists

Objects and lists can also be written inline:

```yaml
person: { name: Bob, age: 28 }
fruits: [Apple, Banana, Orange]
```

Comments

Use `#` for comments:

```yaml
# This is a comment
name: John Doe  # Inline comment
```

Multi-line Strings

Multi-line strings can be written using `|` (preserves line breaks) or `>` (folds into a single line):

```yaml
message: |
  This is a multi-line
  string in YAML.

description: >
  This is a long description
  that will be folded into a
  single line.
```

Boolean, Null, and Numeric Values

```yaml
is_active: true
has_paid: false
value: null
score: 9.8
```

Anchors and Aliases (References)

To reuse values:

```yaml
defaults: &default_settings
  theme: dark
  font_size: 14

user_settings:
  <<: *default_settings
  font_size: 16  # Override default
```

Step-by-Step Guide

Step 1: Install Required Packages

To handle YAML files in Python, install the PyYAML library:

```
pip install pyyaml
```

Step 2: Create the YAML File

Create a YAML file named `students.yaml` containing student data:

```yaml
students:
  - name: Alice
    age: 21
    major: Computer Science
    gpa: 3.8
  - name: Bob
    age: 22
    major: Mathematics
    gpa: 3.5
  - name: Charlie
    age: 20
    major: Physics
    gpa: 3.9
  - name: David
    age: 23
    major: Chemistry
    gpa: 3.2
  - name: Eva
    age: 21
    major: Computer Science
    gpa: 3.7
```

Step 3: Write the Python Application

Create a Python script `app.py` to read and process student data:

```python
import yaml

def load_data(file_path):
    """Load data from a YAML file."""
    with open(file_path, 'r') as file:
        data = yaml.safe_load(file)
    return data

def display_students(students):
    """Display all student information."""
    print("\nAll Students:")
    for student in students:
        print(f"Name: {student['name']}, Age: {student['age']}, Major: {stu

def filter_students_by_gpa(students, min_gpa):
    """Filter students with a GPA above a certain threshold."""
    filtered_students = [s for s in students if s['gpa'] >= min_gpa]
    print(f"\nStudents with GPA >= {min_gpa}:")
    if filtered_students:
        for student in filtered_students:
            print(f"Name: {student['name']}, Age: {student['age']}, Major:
    else:
        print("No students found.")
```

```python
def main():
    data = load_data('students.yaml')
    students = data['students']
    display_students(students)
    min_gpa = float(input("\nEnter minimum GPA to filter students: "))
    filter_students_by_gpa(students, min_gpa)

if __name__ == "__main__":
    main()
```

Step 4: Run the Application

1. Ensure `app.py` and `students.yaml` are in the same directory.
2. Open the terminal and run:

```
python app.py
```

Expected Output

Example output after running the script:

```
All Students:
Name: Alice, Age: 21, Major: Computer Science, GPA: 3.8
Name: Bob, Age: 22, Major: Mathematics, GPA: 3.5
Name: Charlie, Age: 20, Major: Physics, GPA: 3.9
Name: David, Age: 23, Major: Chemistry, GPA: 3.2
Name: Eva, Age: 21, Major: Computer Science, GPA: 3.7

Enter minimum GPA to filter students: 3.6
Students with GPA >= 3.6:
Name: Alice, Age: 21, Major: Computer Science, GPA: 3.8
Name: Charlie, Age: 20, Major: Physics, GPA: 3.9
Name: Eva, Age: 21, Major: Computer Science, GPA: 3.7
```

Conclusion

This example demonstrates how to use YAML in Python for data storage and retrieval. You can expand the script by adding features such as sorting, updating student information, or saving changes back to the YAML file.