2

Introduction to YAML

Basic YAML Syntax
Brief overview of ESM-Tools Extended YAML Syntax

2 Introduction to YAML

- How to **save** the configuration and **transfer** it across different apps and machines
- **Serialization** & **Deserialization**
- Binary vs **Text** (human-readable)
- **Easy** to read & write
- 3 standards exist: XML, JSON, YAML

XML	JSON	YAML

<Servers>
<Server>
<name>Server1</name>
<owner>John</owner>
<created>123456</created>
<status>active</status>
</Server>
</Servers>
	{
	Servers: [
	{
	name: Server1,
	owner: John,
	created: 123456,
	status: active
	}
	]
	}
		Servers:
		- name: Server1
		owner: John
		created: 123456
		status: active

Use space (eg. 4 spaces) instead of tab

Hands-on introduction

```yaml
# ============================
# This is a sample YAML file
# ============================
age: 18   # integer value

1: one    # integer key

PalMod_Date: 2022-04-20  # date value

another_date: '2020-03-09'   # note the quotes

golden_ratio: 1.6180339887   # float value

3.14159265359: pi  # float key

# list: one item per line
shopping_lists:
- tofu
- beer
- apples

# inline lists
models: ["echam", "fesom", "pism"]

# dictionary, map
foo:
    bar:
        fizz:
            buzz: 666
```

```python
import yaml

fname = "data.yaml"

# choose a loader
loader = yaml.BaseLoader    # basic loader
# loader = yaml.FullLoader  # more advanced loader

# Opend and deserialize the YAML file
with open(fname, "rt") as yaml_file:
    yaml_data = yaml.load(yaml_file, Loader=loader)


# Print the information about the deserialized data
print("Type of the loaded data is: ", type(yaml_data), "\n")
print("Contents of the YAML file:")
print("--------------------------")
print( yaml.dump(yaml_data, indent=4) )
print("\n")


# iterate over the items and print them
width = 40
for key, value in yaml_data.items():
    print(f"{str(key):40}  {type(key)}")
    print(f"{str(value):40}  {type(value)}")
    print()

print()
print(yaml_data["PalMod_Date"])
print(yaml_data["foo"]["bar"]["fizz"]["buzz"])
print(yaml_data["models"][-1])
```

Hands-on introduction

Output

```
Type of the loaded data is:  <class 'dict'>

Contents of the YAML file:
--------------------------
'1': one
'3.14159265359': pi
PalMod_Date: '2022-04-20'
age: '18'
another_date: '2020-03-09'
foo:
    bar:
        fizz:
            buzz: '666'
golden_ratio: '1.6180339887'
models:
- echam
- fesom
- pism
shopping_lists:
- tofu
- beer
- apples
```

Output (contd.)

```
age                                     <class 'str'>
18                                      <class 'str'>

1                                       <class 'str'>
one                                     <class 'str'>

PalMod_Date                             <class 'str'>
2022-04-20                              <class 'str'>

another_date                            <class 'str'>
2020-03-09                              <class 'str'>

golden_ratio                            <class 'str'>
1.6180339887                            <class 'str'>

3.14159265359                           <class 'str'>
pi                                      <class 'str'>

shopping_lists                          <class 'str'>
['tofu', 'beer', 'apples']              <class 'list'>

models                                  <class 'str'>
['echam', 'fesom', 'pism']              <class 'list'>

foo                                     <class 'str'>
{'bar': {'fizz': {'buzz': '666'}}}      <class 'dict'>

2022-04-20    # yaml_data["PalMod_Date"]
666           # yaml_data["foo"]["bar"]["fizz"]["buzz"]
pism          # yaml_data["models"][-1]
```

data.yaml:

```yaml
age: 18

1: one

PalMod_Date: 2022-04-20

another_date: '2020-03-09'

golden_ratio: 1.6180339887

3.14159265359: pi

# list: one item per line
shopping_lists:
- tofu
- beer
- apples

# inline lists
models: ["echam", "fesom", "pism"]

# dictionary, map
foo:
    bar:
        fizz:
            buzz: 666
```

Exercise: Change the loader to **`FullLoader`** and try again

2 ESM-Tools Extended YAML Syntax: overview

- We can only store data and configuration in a YAML file.
- ESM-Tools extends this functionality by adding **operations** and **commands**.
- ESM-Tools programs (eg. `esm_master`, `esm_runscripts`) parse these commands.
- Language for Earth System Modelling
- **One language** for all (supported) models and HPCs.
- Model agnostic. High-level abstraction
- It is always possible to access the model internals (eg. namelists, configuration files).

❷ ESM-Tools Extended YAML Syntax: overview

```yaml
# creating and accessing variables from different sections
ini_restart_dir: "${general.ini_restart_dir}/fesom/"
```

```yaml
# maths and calendar operations
runtime: $(( ${end_date} - ${time_step}seconds ))
```

```yaml
# adding and removing elements from lists and dicts

list1:
    - element1
    - element2

add_list1:
    - element3
    - element4
```

```yaml
# Changing Fortran namelists
namelist_changes:
    namelist.echam:
        runctl:
            out_expname: ${general.expid}
            dt_start:
                - ${pseudo_start_date!year}
                - ${pseudo_start_date!month}
```

```yaml
# choose_ blocks allow select-case (aka switch) statements
resolution: CORE2

choose_resolution:
    CORE2:
        nx: 126858
        mesh_dir: "${pool_dir}/meshes/mesh_CORE2_final/"
        nproc: 288
        time_step: 450
    GLOB:
        nx: 830305
```

... and many more to come