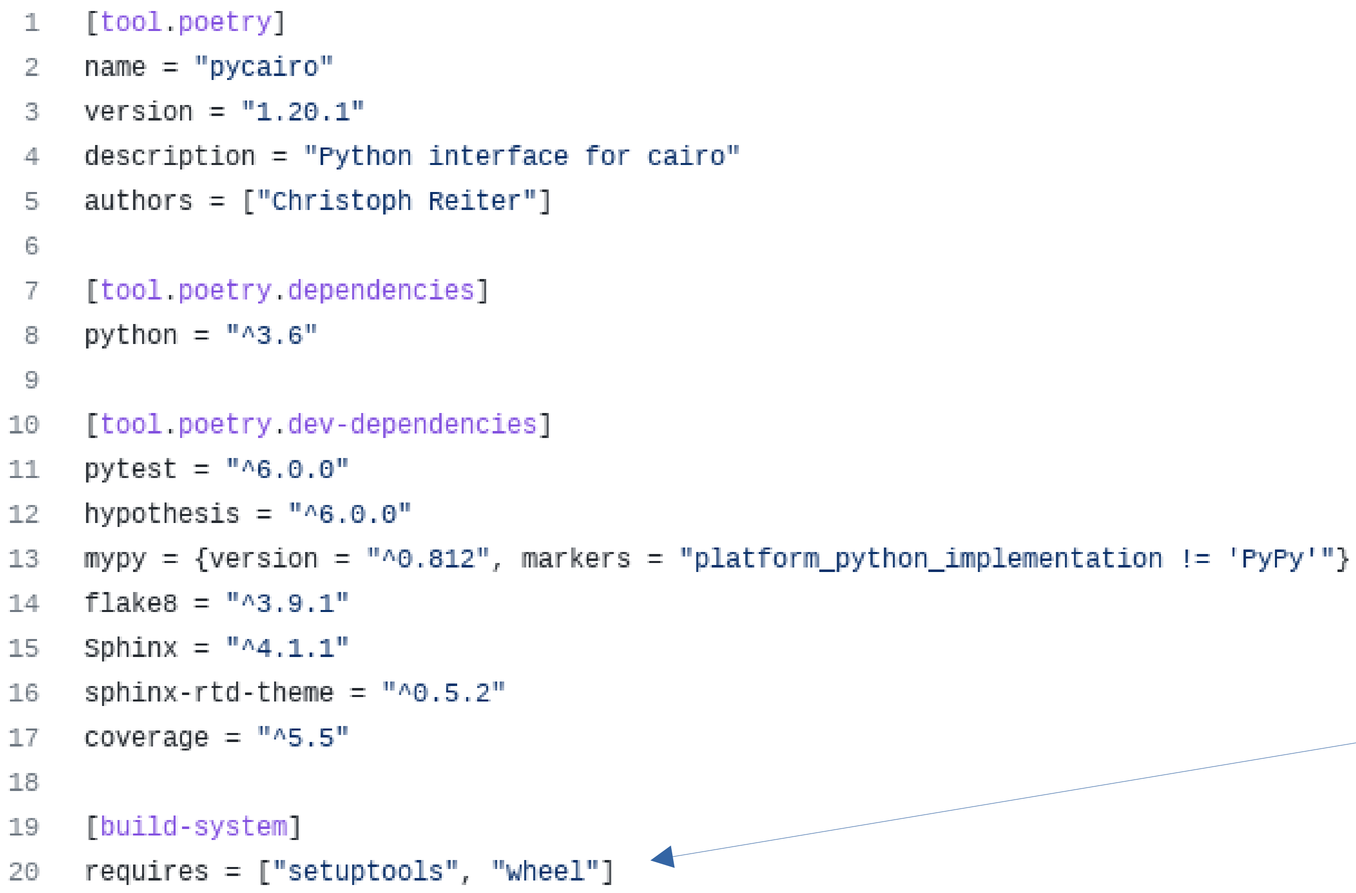
TOML

https://toml.io

Why?

- pyproject.toml → PEP 517 A build-system independent format for source trees
- Poetry etc. → https://python-poetry.org/docs/pyproject/
- PEP-665 A file format to list Python dependencies for reproducibility of an application

Example pyproject.toml

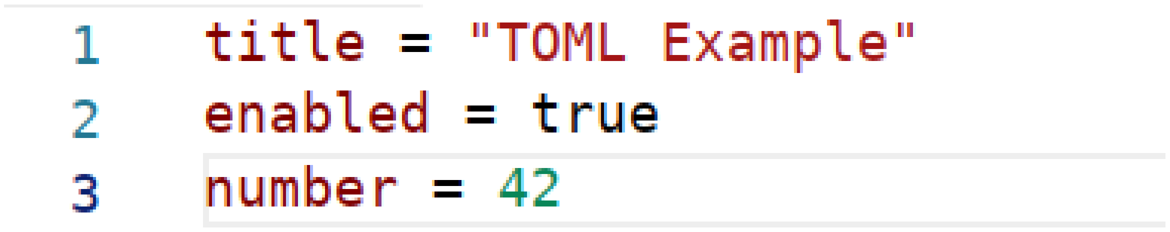


```toml
[tool.poetry]
name = "pycairo"
version = "1.20.1"
description = "Python interface for cairo"
authors = ["Christoph Reiter"]

[tool.poetry.dependencies]
python = "^3.6"

[tool.poetry.dev-dependencies]
pytest = "^6.0.0"
hypothesis = "^6.0.0"
mypy = {version = "^0.812", markers = "platform_python_implementation != 'PyPy'"}
flake8 = "^3.9.1"
Sphinx = "^4.1.1"
sphinx-rtd-theme = "^0.5.2"
coverage = "^5.5"

[build-system]
requires = ["setuptools", "wheel"]
```

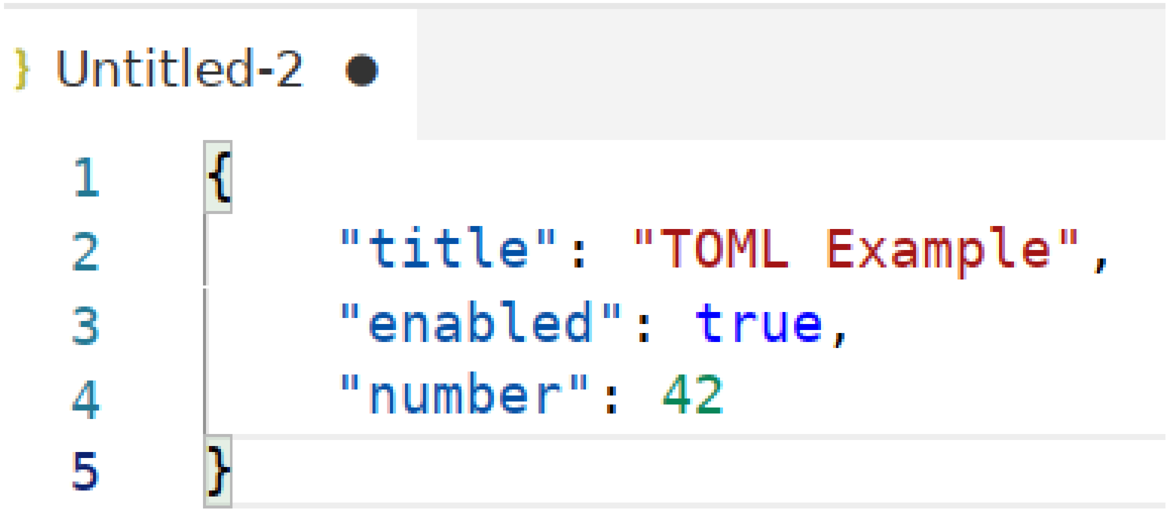
Similar structure to .ini, but more features and has a spec

Only this is PEP 517

Basics

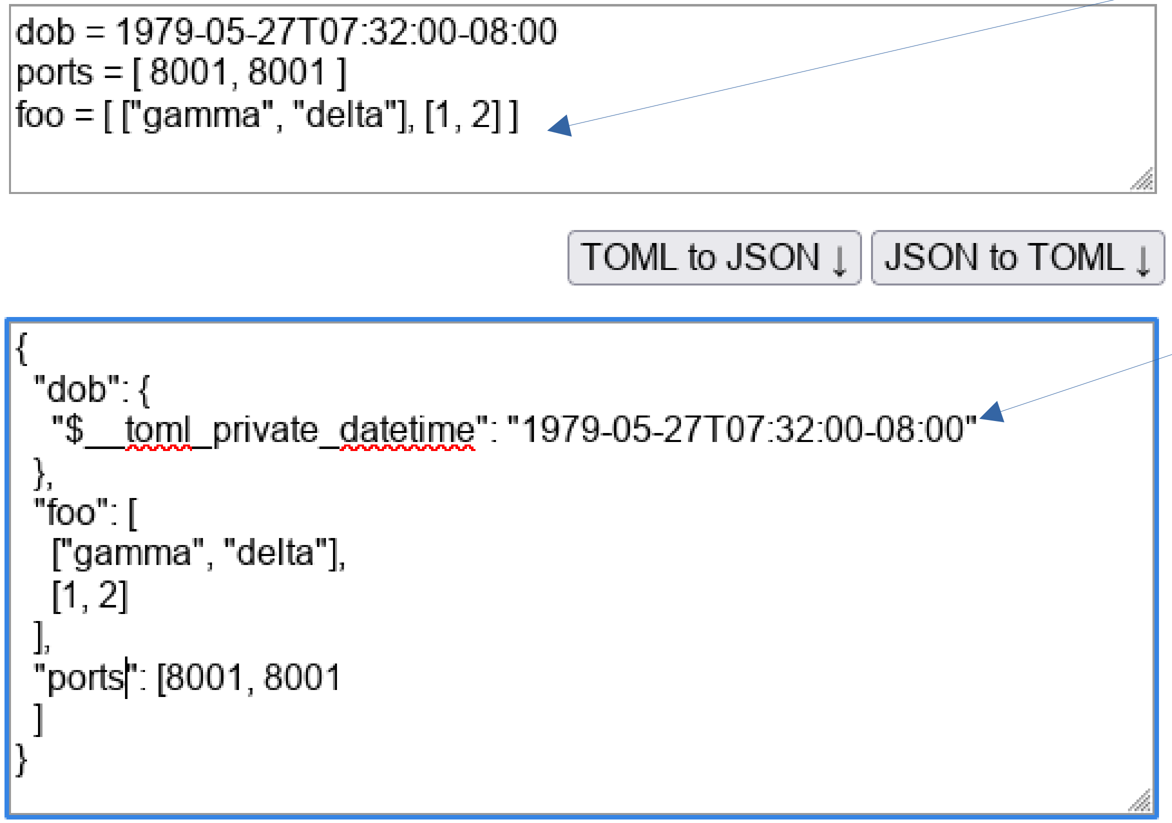


```toml
title = "TOML Example"
enabled = true
number = 42
```



```json
{
    "title": "TOML Example",
    "enabled": true,
    "number": 42
}
```

More Stuff



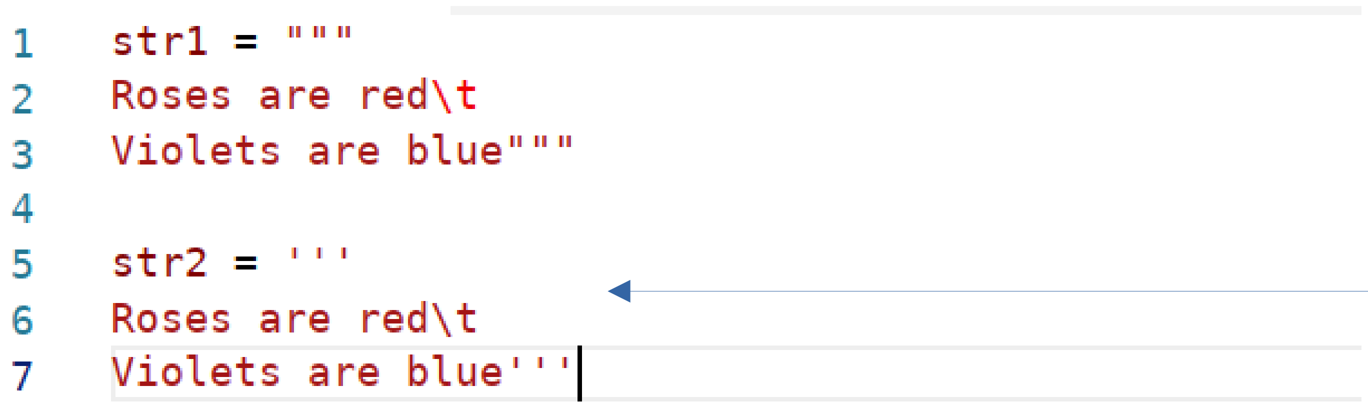
```
dob = 1979-05-27T07:32:00-08:00
ports = [ 8001, 8001 ]
foo = [ ["gamma", "delta"], [1, 2] ]
```

```
{
  "dob": {
    "$__toml_private_datetime": "1979-05-27T07:32:00-08:00"
  },
  "foo": [
    ["gamma", "delta"],
    [1, 2]
  ],
  "ports": [8001, 8001
  ]
}
```

List items need to be the same type

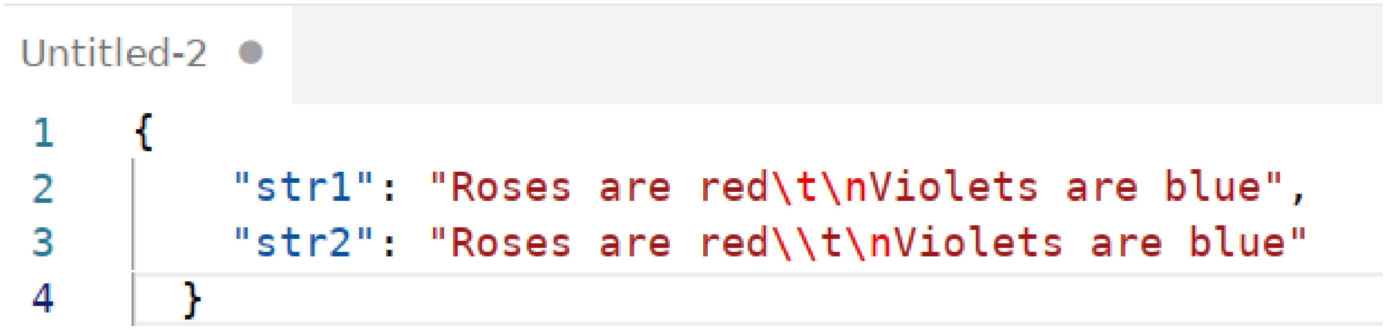
In Python this is a datetime object

Multi Line Strings



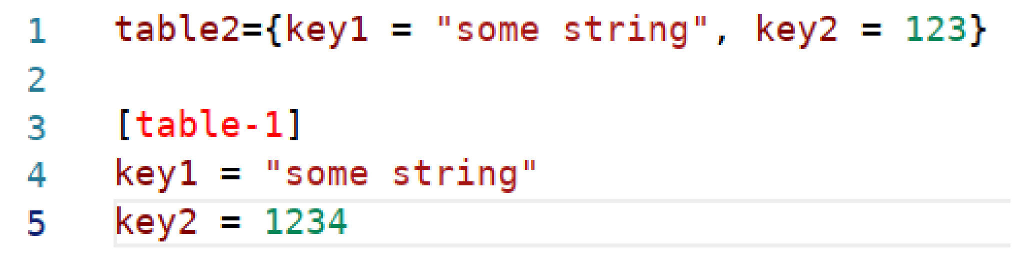
```
str1 = """
Roses are red\t
Violets are blue"""

str2 = '''
Roses are red\t
Violets are blue'''
```

Literal string

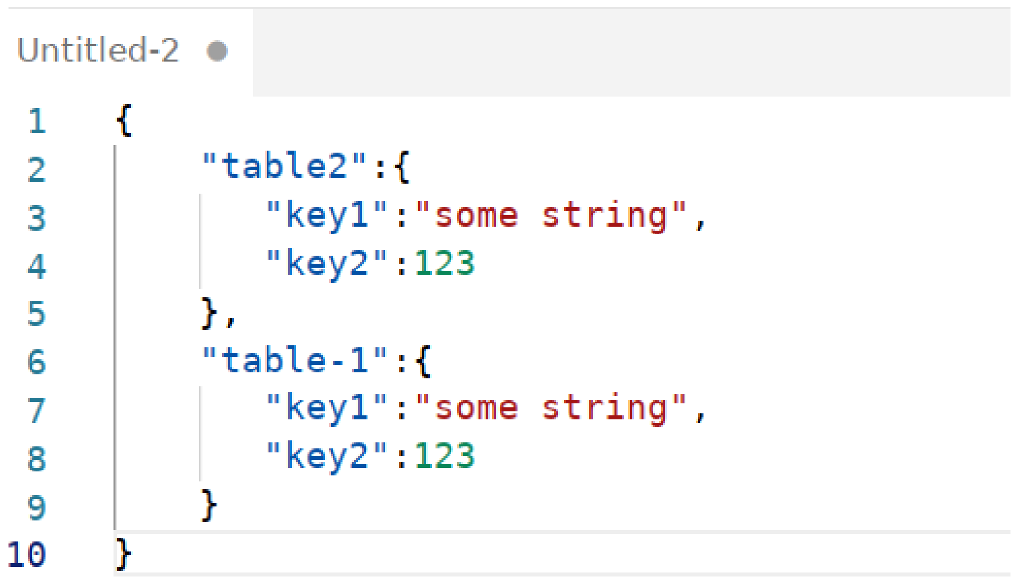


```
{
    "str1": "Roses are red\t\nViolets are blue",
    "str2": "Roses are red\\t\nViolets are blue"
 }
```

Tables or Dicts

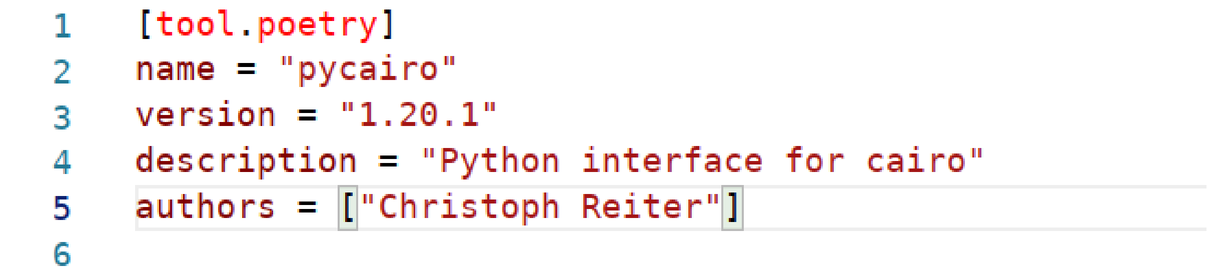


```
table2={key1 = "some string", key2 = 123}

[table-1]
key1 = "some string"
key2 = 1234
```

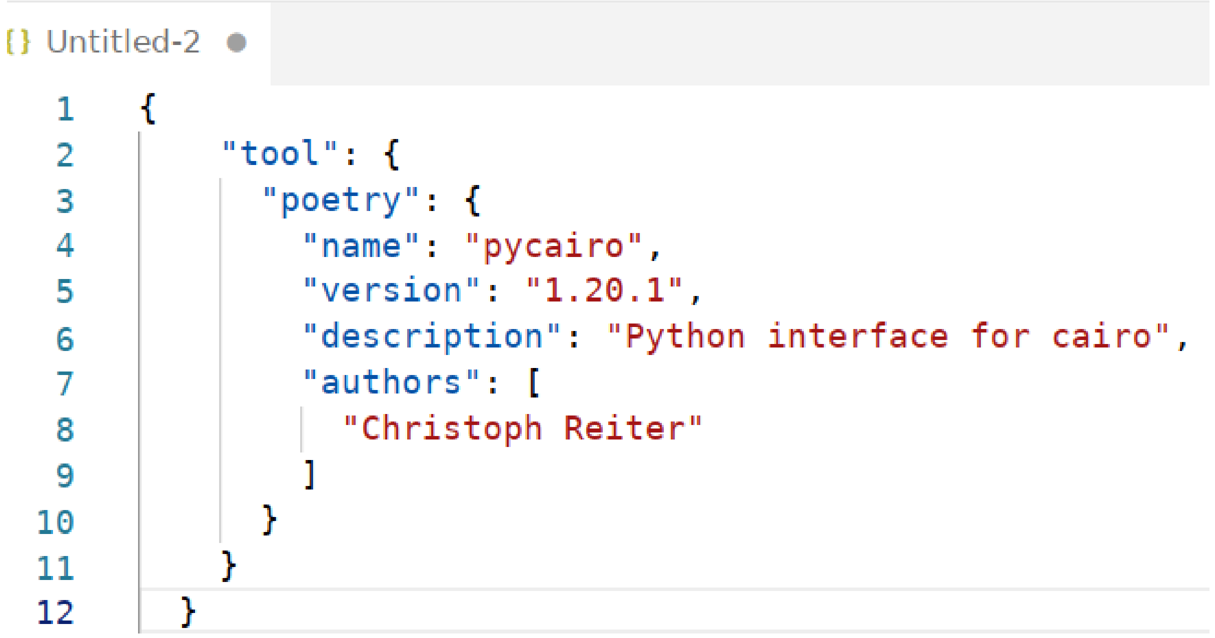


```
{
    "table2":{
        "key1":"some string",
        "key2":123
    },
    "table-1":{
        "key1":"some string",
        "key2":123
    }
}
```

Tables or Dicts

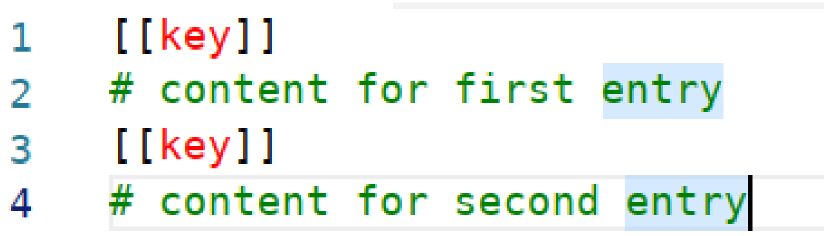


```toml
[tool.poetry]
name = "pycairo"
version = "1.20.1"
description = "Python interface for cairo"
authors = ["Christoph Reiter"]

```

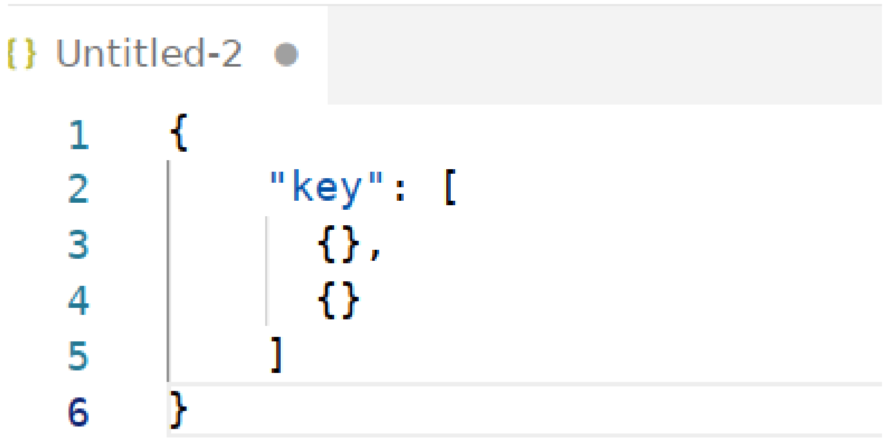


```json
{
    "tool": {
      "poetry": {
        "name": "pycairo",
        "version": "1.20.1",
        "description": "Python interface for cairo",
        "authors": [
          "Christoph Reiter"
        ]
      }
    }
  }
```

Array of Tables – 1/3



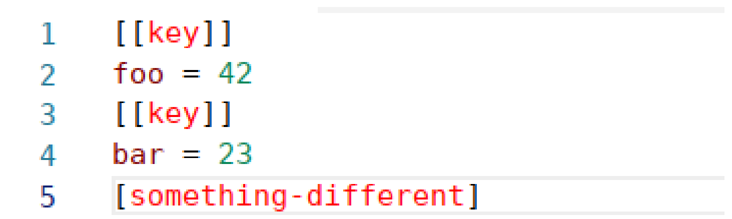
```toml
[[key]]
# content for first entry
[[key]]
# content for second entry
```



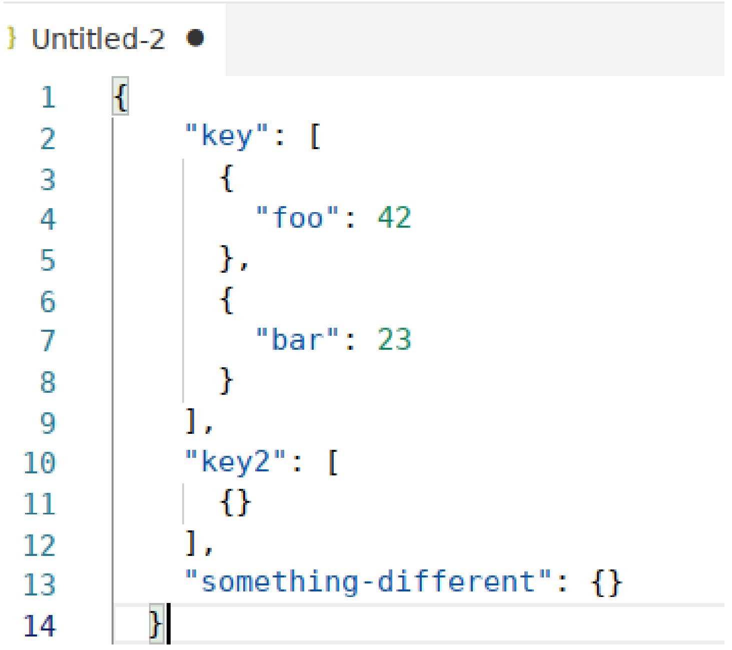
Untitled-2

```json
{
    "key": [
      {},
      {}
    ]
}
```

Array of Tables – 2/3

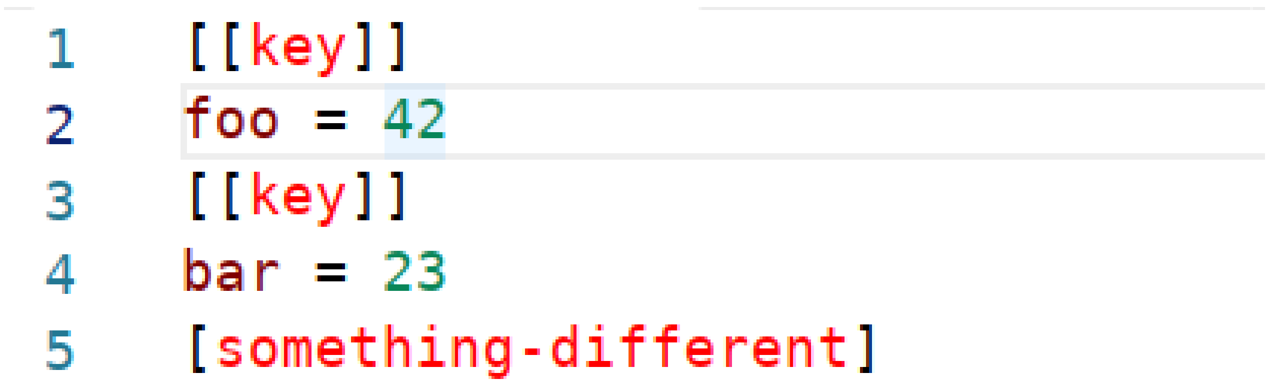


```
[[key]]
foo = 42
[[key]]
bar = 23
[something-different]
```

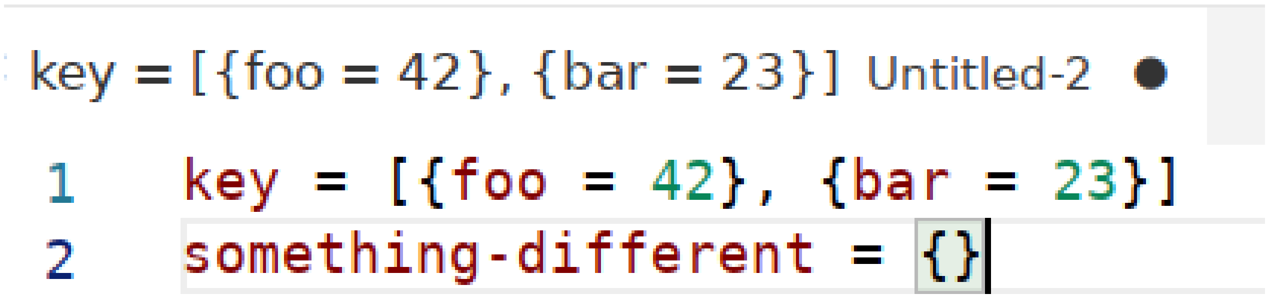


```
{
    "key": [
      {
        "foo": 42
      },
      {
        "bar": 23
      }
    ],
    "key2": [
      {}
    ],
    "something-different": {}
}
```

Alternative Syntax 3/3



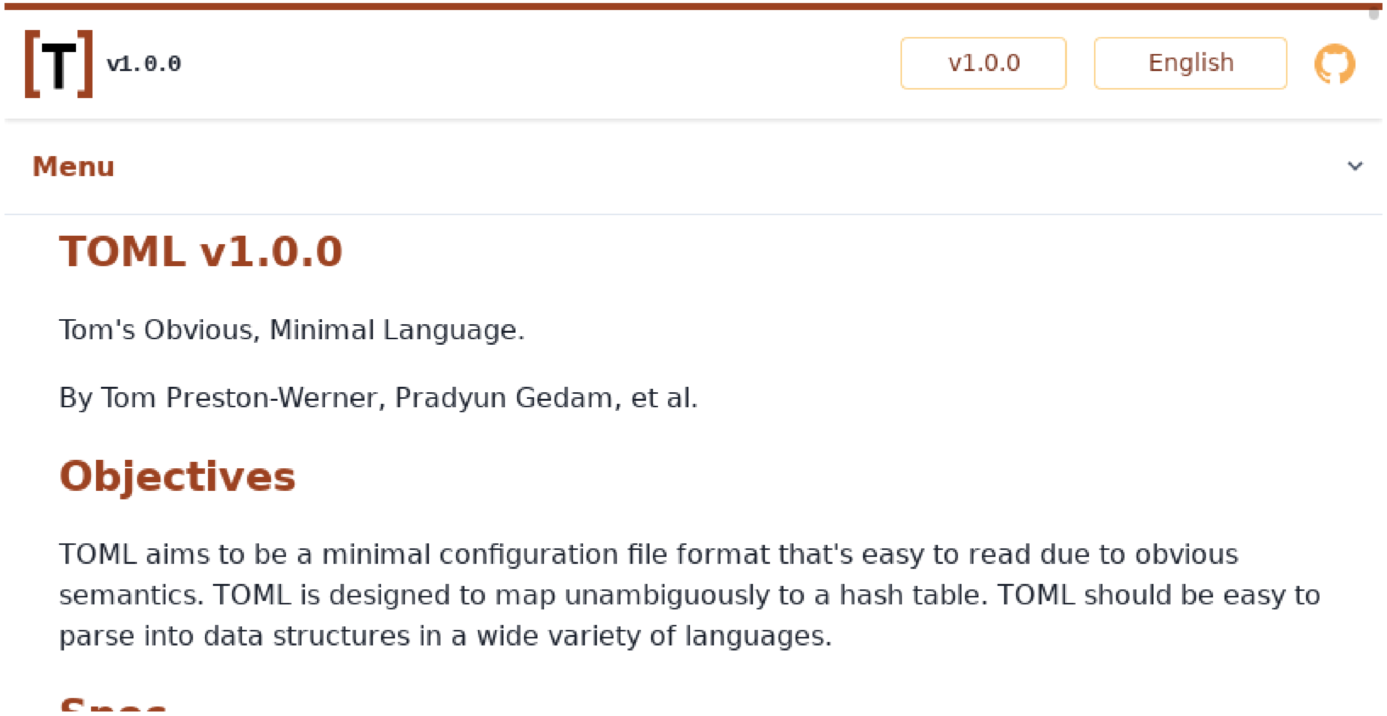
```
[[key]]
foo = 42
[[key]]
bar = 23
[something-different]
```



key = [{foo = 42}, {bar = 23}] Untitled-2 ●

```
key = [{foo = 42}, {bar = 23}]
something-different = {}
```

Spec for Details



TOML v1.0.0

Tom's Obvious, Minimal Language.

By Tom Preston-Werner, Pradyun Gedam, et al.

Objectives

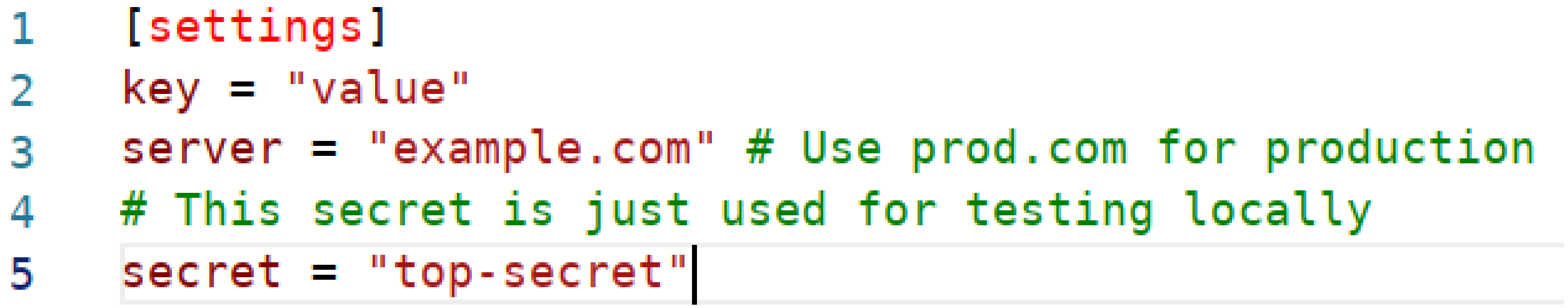
TOML aims to be a minimal configuration file format that's easy to read due to obvious semantics. TOML is designed to map unambiguously to a hash table. TOML should be easy to parse into data structures in a wide variety of languages.

https://toml.io/en/v1.0.0

poetry uses tomlkit

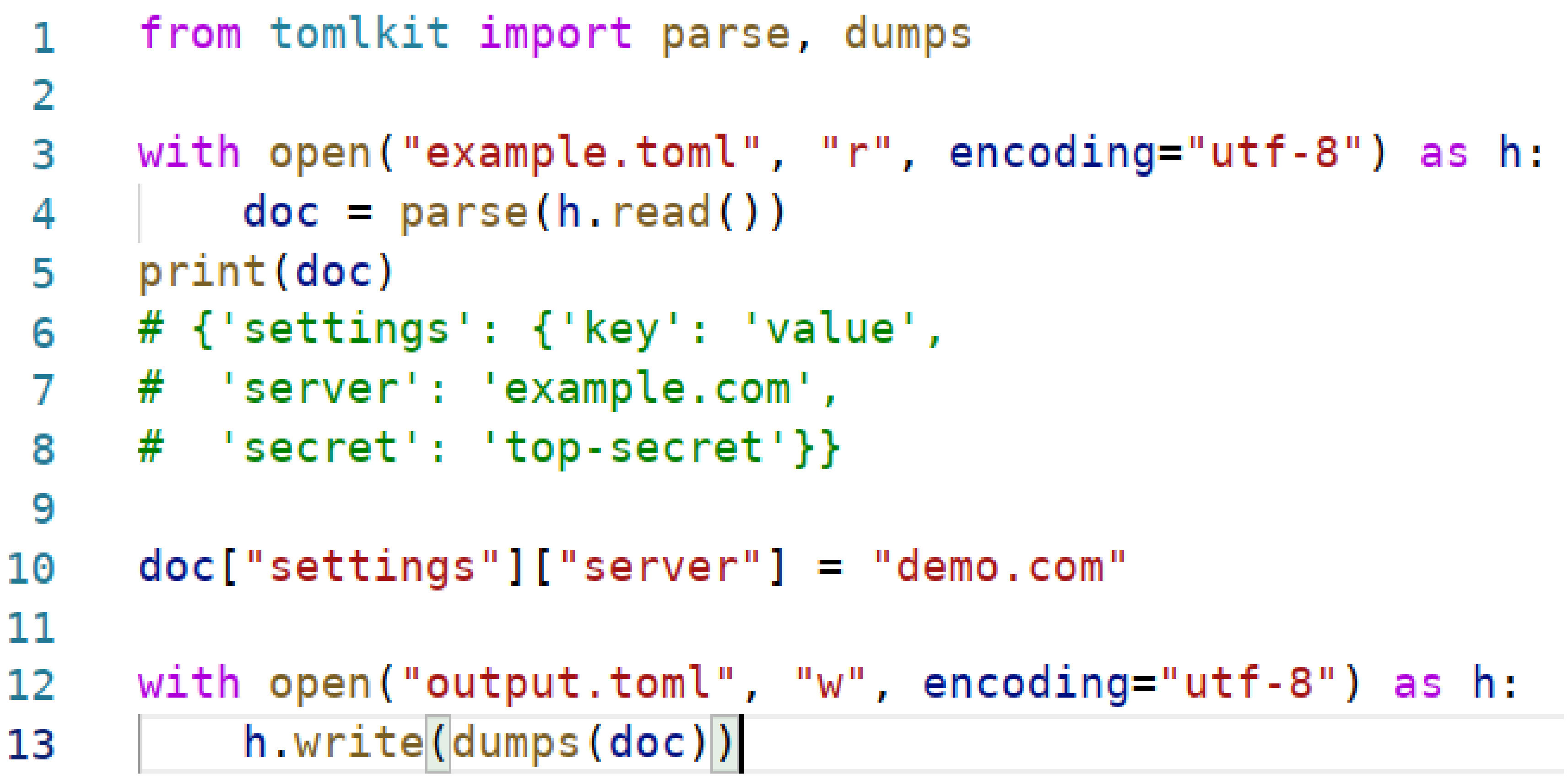
- “It includes a parser that preserves all comments, indentations, whitespace and internal element ordering, and makes them accessible and editable via an intuitive API.”
- https://github.com/sdispater/tomlkit

Input TOML file



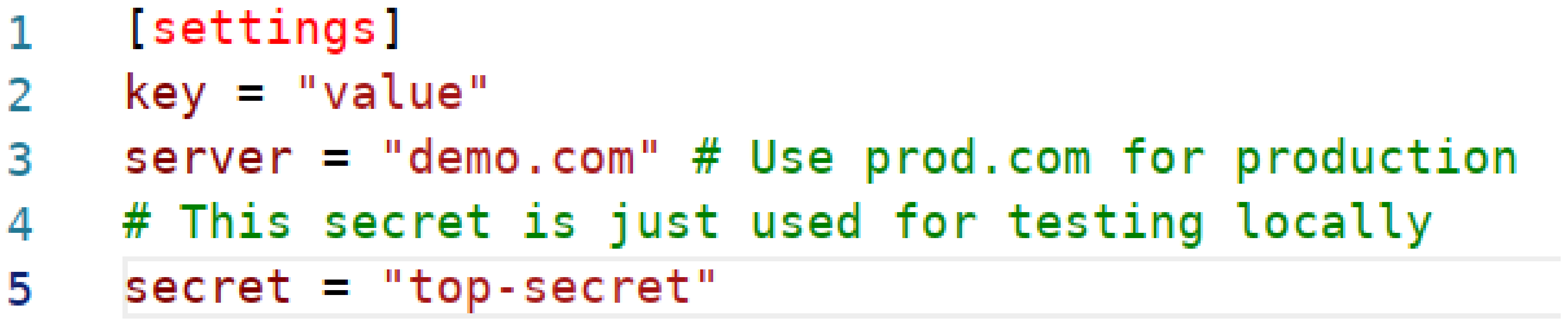
```toml
[settings]
key = "value"
server = "example.com" # Use prod.com for production
# This secret is just used for testing locally
secret = "top-secret"
```

Now we change a value



```python
from tomlkit import parse, dumps

with open("example.toml", "r", encoding="utf-8") as h:
    doc = parse(h.read())
print(doc)
# {'settings': {'key': 'value',
#  'server': 'example.com',
#  'secret': 'top-secret'}}

doc["settings"]["server"] = "demo.com"

with open("output.toml", "w", encoding="utf-8") as h:
    h.write(dumps(doc))
```

Result



```
[settings]
key = "value"
server = "demo.com" # Use prod.com for production
# This secret is just used for testing locally
secret = "top-secret"
```

Use cases?

- Have a config file that mixes state and config?
- Migrate user config on version upgrades?
- Have commands that extend the user config?

END