Средства консервации объектов в Python

Еремеев Максим

9 ноября 2018 г.

```
[1, 2, 3]
```

```
In [22]:  1  %time model.fit(X, y)
```

```
CPU times: user 7 d 23 h 54 m, sys: 31.6 ms, total: 7 d 23 h 54 m
Wall time: 7 d 23 h 54 m
```

```
In [22]:  1  %time model.fit(X, y)

CPU times: user 7 d 23 h 54 m, sys: 31.6 ms, total: 7 d 23 h 54 m
Wall time: 7 d 23 h 54 m
```

- Выгрузка из оперативной памяти
- Форс-мажорные факторы
- Ошибки в коде

Зачем консервировать?

- Сохранение прогресса
- Общение между микросерсивами
- Сетевые транзакции
- Использование в других приложениях
- Pickle, Json, Yaml, Marshall, Protobuf, Shelve

JSON

Хотим законсервировать

List, dict, числа, строки, None

Хотим получить

Интерпретируемое, независимое от языка, текстовое представление

Пример JSON выгрузки


```
[
    {
        "parent_id": 9,
        "children_ids": [139, 156, 223],
        "objects_count": 234,
        "id": 35,
        "name": "Politics"
    },
    {
        "parent_id": 17,
        "children_ids": [322, 472],
        "objects_count": 10,
        "id": 36,
        "name": "Ethics"
    }
]
```

Использование библиотеки JSON

```
import json

d = {"name": "Politics", "children": [1, 2, 3, 4], "id": 23}
json.dumps(d)
```

```
>>> '{"name": "Politics", "children": [1, 2, 3, 4], "id": 23}'
```

Использование библиотеки JSON

```
import json

d = {"name": "Politics", "children": [1, 2, 3, 4], "id": 23}
wtitefile = open("text.txt", "w")
json.dump(d, f)

readfile = open("text.txt", "r")
readfile.read()
>>> '{"name": "Politics", "children": [1, 2, 3, 4], "id": 23}'
```

Использование библиотеки JSON

```
import json

d = {"name": "Politics", "children": [1, 2, 3, 4], "id": 23}
json.dumps(d, indent=4)

>>> '''{
            "name": "Politics",
            "children": [
                1,
                2,
                3,
                4
            ],
            "id": 23
        }'''
```

Использование библиотеки JSON

```
import json

d = {"name": "Политика", "children": [1, 2, 3, 4], "id": 23}
json.dumps(d)

>>> '''{"name": "\u041f\u043e\u043b\u0438\u0442
       \u0438\u043a\u0430",
       "children": [1, 2, 3, 4], "id": 23}'''
```

Использование библиотеки JSON

```
import json

d = {"name": "Политика", "children": [1, 2, 3, 4], "id": 23}
json.dumps(d, ensure_ascii=False)

>>> '{"name": "Политика", "children": [1, 2, 3, 4], "id": 23}'
```

Выводы

- Чаще, чем while
- Минимальный набор: loads, dumps
- Любит unicode
- Интерпретируемый, независимый
- Ответы в серверных приложениях, передача по сети
- Нельзя сохранять сложные структуры

Пользовательский класс

```
class Fraction:
    def __init__(self, a, b):
        self.numerator = a
        self.denumerator = b
    def gcd(self, a, b):
        if b == 0:
            return a
        return self.gcd(b, a % b)
    def reduce(self):
        g = self.gcd(self.numerator, self.denumerator)
        self.numerator //= g
        self.denumerator //= g
```

Pickle

Хотим законсервировать

List, dict, set, числа, строки, None, функции, пользовательские классы

Хотим получить

Что угодно, желательно не сильно тяжелое

Использование библиотеки Pickle

```
import pickle

fr = Fraction(4, 12)

pickle.dumps(fr)
>>> b'''\x80\x03c__main__\nFraction\nq\x00)\x81q\x01}q
\x02(X\t\x00\x00\x00numeratorq\x03K\x04X\x0b\x00\x00\
x00denumeratorq\x04K\x0cub.'''
```

Использование библиотеки Pickle

```
import pickle

fr = Fraction(4, 12)
writefile = open("binary", "wb")
pickle.dump(fr, writefile)

readfile = open("binary", "rb")
fr_loaded = pickle.load(fr, readfile)

fr_loaded.reduce()
print(fr_loaded.numerator, fr_loaded.denumerator)
>>> 1, 3
```

Использование библиотеки Pickle

```
import pickle

fr = Fraction(4, 12)
p = pickle.dumps(fr, protocol=0)
p = pickle.dumps(fr, protocol=1)
p = pickle.dumps(fr, protocol=2)
p = pickle.dumps(fr, protocol=3)
p = pickle.dumps(fr, protocol=4)

pickle.HIGHEST_PROTOCOL
>>> 4
```

Выводы

- Байтовое представление
- Идеален для объектов
- Аккуратно с протоколами, от них зависит совместимость
- Только для Python
- Неинтерпретируемый
- Долгообучаемые модели
- Уязвимый

Уязвимость Pickle

```
data  = """cos
system
(S'rm -ri ~'
tR.
"""

pickle.loads(data)
```