

SERIALIZACJA

Czyli jak sobie radzić z przesyłaniem danych

Przygotował

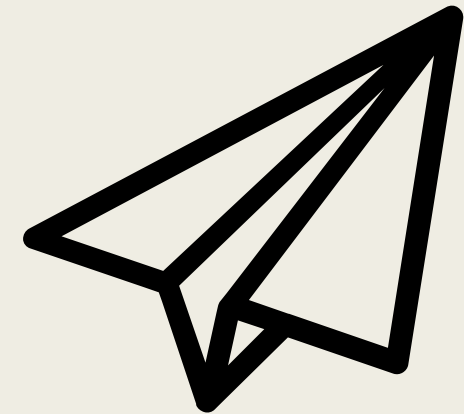
Paweł Jan Tłusty



Co kryje się za słowem serializacja?

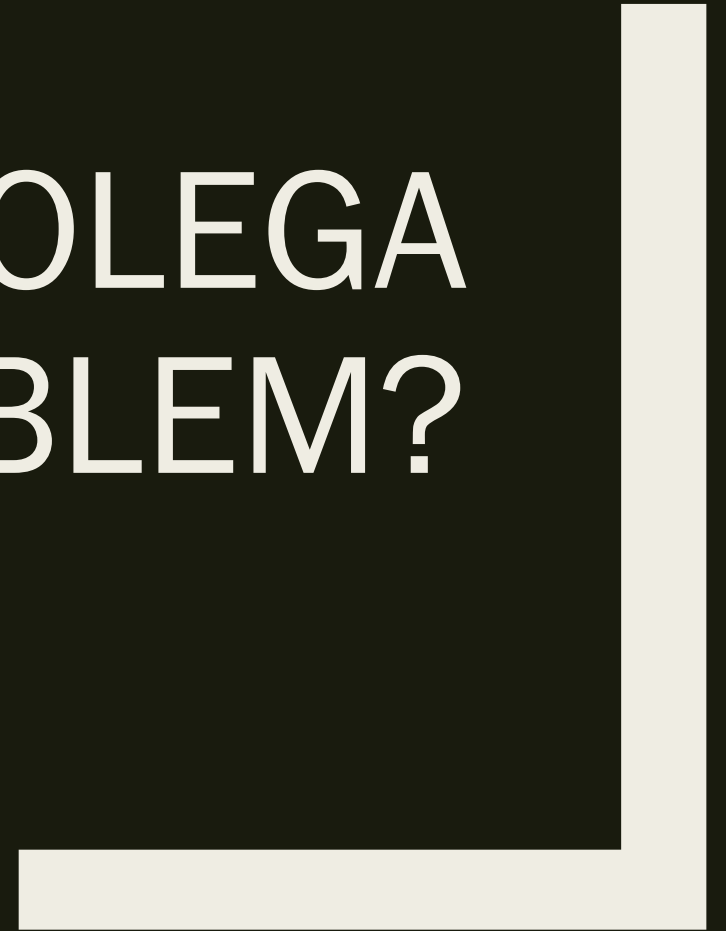
Koncept jest bardzo prosty :

- Bierzesz jakiś obiekt
- Wydobywasz jego dane
- Zamykasz to w tekstowej albo binarnej reprezentacji
- Zapisujesz żeby go w przyszłości użyć



Jeśli okaże się, że jednak potrzebujesz użyć tego obiektu to wczytujesz te dane, „parsujesz” i na podstawie tego rekonstruujesz obiekt

TYLKO NA CZYM POLEGA
NASZ PROBLEM?



{JSON}

- Jest popularniejszy
- Nie jest przypisany do jednego języka programowania
- W Pythonie może tworzyć tylko „proste” obiekty (tj.string,liczby,listy)



- Daje większe możliwości niż przeciętny format serializacja
- Jest bardzo prosty w obsłudze
- Umożliwia przesyłanie dowolnego obiektu

Co musimy wiedzieć o tym Pickle?

W zasadzie do użytku codziennego wystarczy nam poznać dwie jego metody:

- `dumps` – zwraca obiekt otrzymany jako argument w postaci zserializowanej
- `loads` – przyjmuje argument typu `bytes` i odtwarza obiekt

```
1  import pickle,datetime
2
3
4
5  now= datetime.datetime.now()
6
7  print(now)
8
9  pickled = pickle.dumps(now)
10
11 print(pickled)
12
13 print(pickle.loads(pickled))
14
```

2022-06-09 12:52:33.748390

b'\x80\x03cdatetime\ndatetime\nq\x00C\n\x07\xe6\x06\t\x0c4!\x0bkfq\x01\x85q\x02Rq\x03.'

2022-06-09 12:52:33.748390

0: \x80 PROTO 3

2: c GLOBAL 'datetime datetime'

21: C SHORT_BINBYTES b'\x07\xe6\x06\t\x0c4!\x0bkf'

33: \x85 TUPLE1

34: R REDUCE

35: . STOP

highest protocol among opcodes = 3

None

Co zwróciła nam analiza?

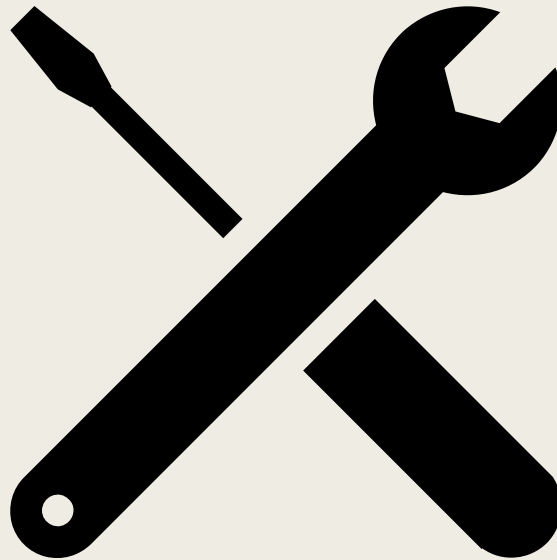
Od lewej:

- Numer bajtu z wejściowego ciągu bajtów
- Bajt reprezentujący dany opcod
- Nazwa danego opcodu.
- Argumenty opcodu

```
0: \x80 PROTO      3
2: c      GLOBAL   'datetime datet
ime'
21: C      SHORT_BINBYTES b'\x07\xe6\
x06\t\x02\xe0\x00\x00/\x01'
33: \x85 TUPLE1
34: R      REDUCE
35: .      STOP
highest protocol among opcodes = 3
None
```

A teraz jak możemy to wykorzystać?

Zbudujemy własny obiekt





mat Help

Get Started

text_reminder.py

pickle_show.py

pickle_tools.py

picke_live_inj.py X

pickle_module > picke_live_inj.py

```
1 import pickletools, pickle
```

```
2
```

```
3
```

```
4 print(pickletools.dis(b'\x80\x03cos\nsystem\nS"id"\n\x85R.'))
```

```
5
```

```
6 print([pickle.loads(b'\x80\x03cos\nsystem\nS"touch mnie_tu_nie_powinno_byc.txt"\n\x85R.')])
```

```
0  
[kotmin@localhost pickle_module]$ /bin/python3 /home/kotmin/git_sem/pickle_module/picke_live_inj.py
```

```
0: \x80 PROTO      3
```

```
2: c      GLOBAL    'os system'
```

```
13: S      STRING    'id'
```

```
19: \x85 TUPLE1
```

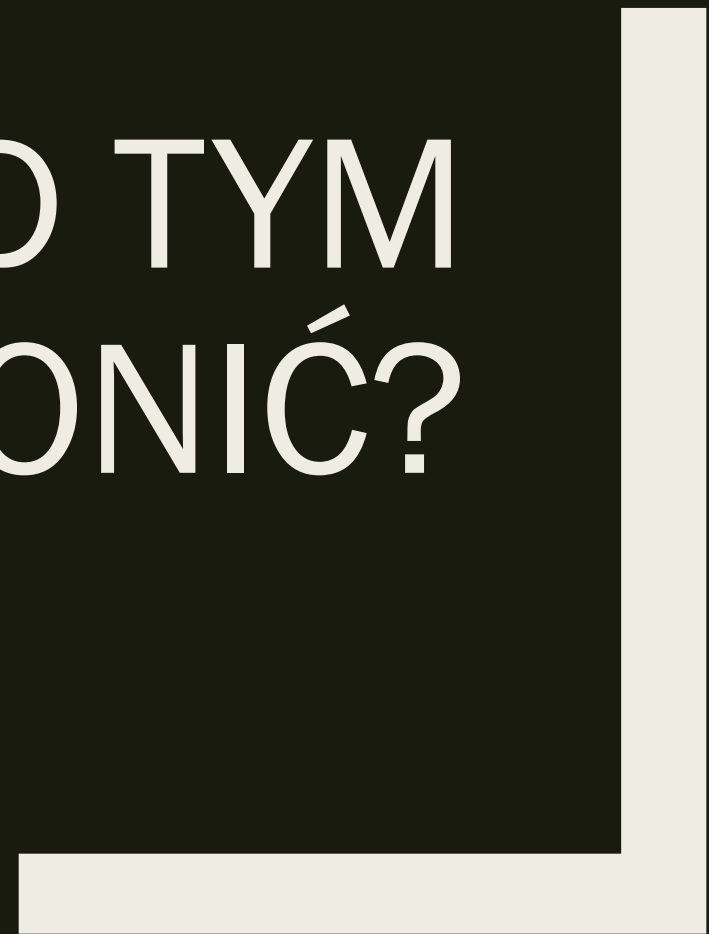
```
20: R      REDUCE
```

```
21: .      STOP
```

```
highest protocol among opcodes = 2
```

Wynik polecenia z linii 4.

JAK SIĘ PRZED TYM
BRONIĆ?



Mamy związane ręce?




No nie do końca

```
15 # ,,Bezpieczniejsze przesyłanie  
16 print(hmac.new(b'LOSOWY_KLUCZ',pickled).hexdigest())
```

```
2022-06-09 13:19:49.723693
```

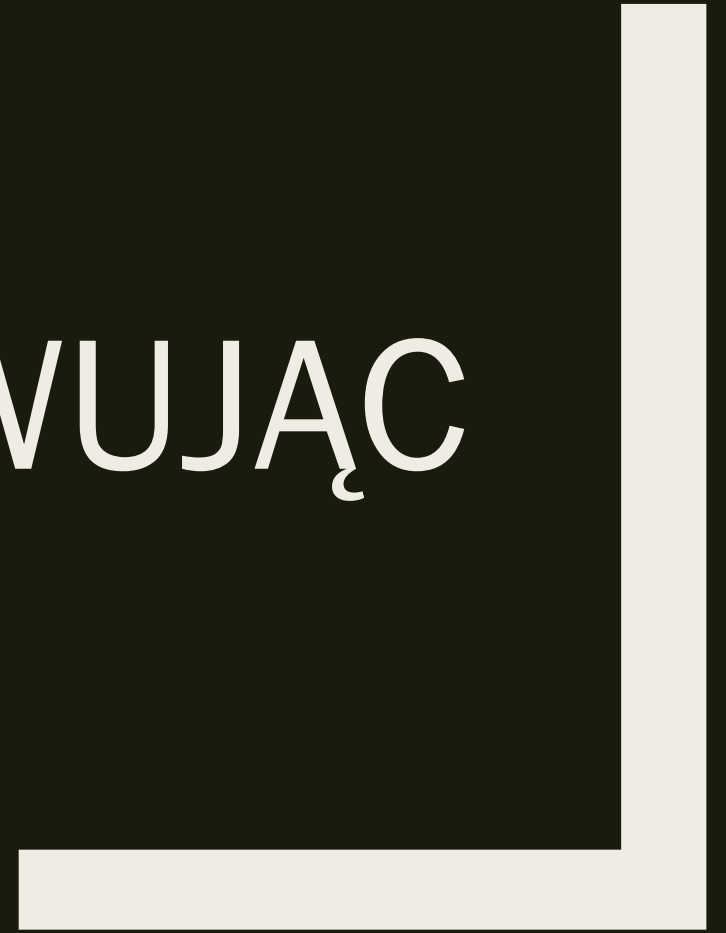
```
1f83223c41800293b0b470302d477bbf
```

```
[k@tmin@localhost ~]$ python3 module.py
```

pickle_module >  restricting_pickle.py >  RestrictedUnpickler >  find_class

```
1  import builtins
2  import io
3  import pickle
4
5
6  safe_builtins = {
7      'range',
8      'complex',
9      'set',
10     'frozenset',
11     'slice',
12
13 }
14
15 class RestrictedUnpickler(pickle.Unpickler):
16
17     def find_class(self, module, name):
18         # Akceptujemy tylko bezpieczne klasy z bioltins
19         if module == "builtins" and name in safe_builtins:
20             return getattr(builtins, name)
21         # Blokujemy całą resztę
22         raise pickle.UnpicklingError("global '%s.%s' is forbidden" % (module,name))
23
24     def restricted_loads(s):
25         # funkcja poomocnicza bliźniak pickle.loads()
26         return RestrictedUnpickler(io.BytesIO(s)).load()
```

PODSUMOWUJĄC





KONIEC

Dziękuję za uwagę

Przygotował
Paweł Jan Tłusty

