

Marco Nicola

- making software for 20+ years
- ML & NLP, full-stack, SaaS, cloud
- working at **EXOP** www.exop-group.com



marco-nicola

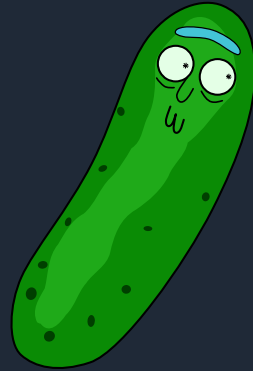


@marconicoladev



nicola-marco

Deserializing Python objects in Go with GoPickle

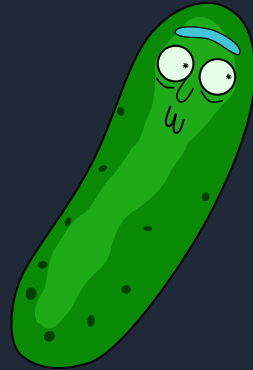


Outline

Python pickle
serialization



pickle
format



reading from Go
(without running Python)

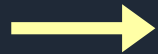


Python pickle module

**binary protocols for serializing
and de-serializing Python objects**



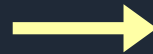
serialize
“pickling”



`pickle.dump()`

```
01001000  
01100101  
01101100  
01101100  
01101111  
00100001
```

deserialize
“unpickling”



`pickle.load()`



Who uses Python pickle?

...everybody!



```
numpy.save()  
numpy.load()
```



```
torch.save()  
torch.load()
```



```
DataFrame.to_pickle()  
Series.to_pickle()  
pandas.read_pickle()
```

Why pickle?



```
obj = {  
    'foo': 'hi',  
    'bar': 42,  
    'baz': {  
        'qux': [1, 'x']  
    }  
}
```

```
json.dumps(obj)
```



JSON

```
{  
    "foo": "hi",  
    "bar": 42,  
    "baz": {  
        "qux": [1, "x"]  
    }  
}
```

Why pickle?



```
class Greeter:  
    def __init__(self, name):  
        self._name = name  
    def greet(self):  
        print(f'Hi, {self._name}!')
```

```
obj = Greeter('Gopher')  
obj.greet()
```

Hi, Gopher!

Why pickle?



```
class Greeter:  
    def __init__(self, name):  
        self._name = name  
    def greet(self):  
        print(f'Hi, {self._name}!')
```

```
obj = Greeter('Gopher')
```

```
json.dumps(obj)
```



JSON

```
TypeError: Object of type  
Greeter is not JSON  
serializable
```

(... mumble mumble ...)

Why pickle?

custom objects / external libraries

object identity / shared objects
[a, a]

recursive objects
L = []; L.append(L)

pickle

~~data format~~

~~parsing~~

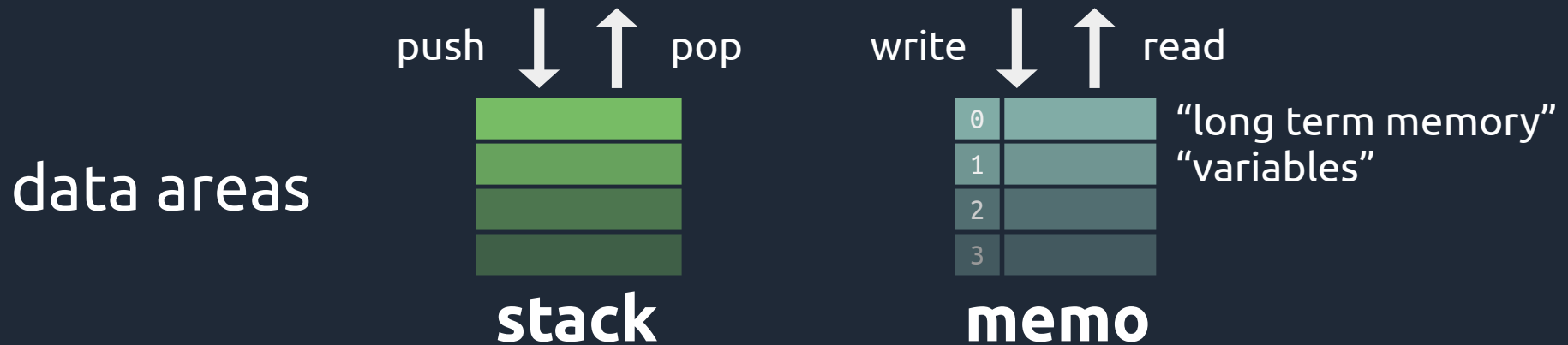
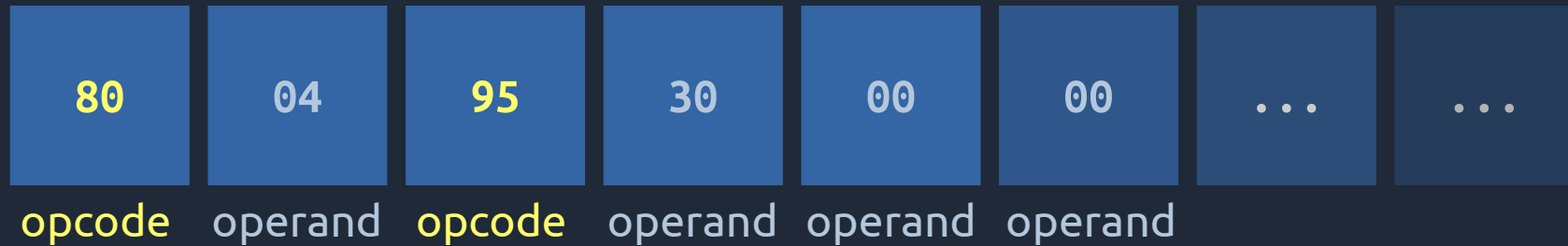
Virtual Machine

pickle program

unpickling machine

Stack-Based Virtual Machine

pickle program: sequence of instructions



Stack-Based Virtual Machine

NO

looping

testing

conditional

arithmetic

function calls

Serialization



```
class Greeter:
    def __init__(self, name):
        self._name = name
    def greet(self):
        print(f'Hi, {self._name}!')

obj = Greeter('Gopher')

import pickle
with open('./obj.pickle', 'wb') as file:
    pickle.dump(obj, file)
```

pickle program

obj.pickle

```
000  80 04 95 30  00 00 00 00  00 00 00 8C  08 5F 5F 6D  ...0.....__m
010  61 69 6E 5F  5F 94 8C 07  47 72 65 65  74 65 72 94  ain___...Greeter.
020  93 94 29 81  94 7D 94 8C  05 5F 6E 61  6D 65 94 8C  ..)..}..._name..
030  06 47 6F 70  68 65 72 94  73 62 2E      .Gopher.sb.
```

pickle program

```
$ python -m pickletools --annotate ./obj.pickle
```

```
0: \x80 PROTO      4          Protocol version indicator.
2: \x95 FRAME      48          Indicate the beginning of a new frame.
11: \x8c SHORT_BINUNICODE '.__main__' Push a Python Unicode string object.
21: \x94 MEMOIZE    (as 0)      Store the stack top into the memo. The stack is not popped.
22: \x8c SHORT_BINUNICODE 'Greeter' Push a Python Unicode string object.
31: \x94 MEMOIZE    (as 1)      Store the stack top into the memo. The stack is not popped.
32: \x93 STACK_GLOBAL Push a global object (module.attr) on the stack.
33: \x94 MEMOIZE    (as 2)      Store the stack top into the memo. The stack is not popped.
34: ) EMPTY_TUPLE Push an empty tuple.
35: \x81 NEWOBJ Build an object instance.
36: \x94 MEMOIZE    (as 3)      Store the stack top into the memo. The stack is not popped.
37: } EMPTY_DICT Push an empty dict.
38: \x94 MEMOIZE    (as 4)      Store the stack top into the memo. The stack is not popped.
39: \x8c SHORT_BINUNICODE '_name' Push a Python Unicode string object.
46: \x94 MEMOIZE    (as 5)      Store the stack top into the memo. The stack is not popped.
47: \x8c SHORT_BINUNICODE 'Gopher' Push a Python Unicode string object.
55: \x94 MEMOIZE    (as 6)      Store the stack top into the memo. The stack is not popped.
56: s SETITEM Add a key+value pair to an existing dict.
57: b BUILD Finish building an object, via __setstate__ or dict update.
58: . STOP Stop the unpickling machine.
```

```
highest protocol among opcodes = 4
```

Deserialization



```
class Greeter:
    def __init__(self, name):
        self._name = name
    def greet(self):
        print(f'Hi, {self._name}!')
```

custom classes / functions
must still be defined

```
import pickle
with open('./obj.pickle', 'rb') as file:
    obj = pickle.load(file)
obj.greet()
```

Hi, Gopher!

pickle protocol versions

0 - 5

- Better efficiency
- New instructions for specific types
- Back compatibility

`pickle` and `pickletools`

`pickle` module documentation

<https://docs.python.org/3/library/pickle.html>

`pickletools` tools for `pickle` developers

<https://docs.python.org/3/library/pickletools.html>

(docs, details, analysis tools)

OK... but what about Go?

Wish List

- unpickling data in Go
- support for all pickle protocols (0-5)
- basic types working out-of-the box
- easy to expand with custom types (ext. libs)
- without running Python (at any step)
- minimal/no dependencies, no unsafe/cgo

introducing

GoPickle

<https://github.com/nlpodyssey/gopickle>

Go library
for loading Python data
serialized with pickle

GoPickle

- focus on deserialization only
- porting pure-Python `Unpickler` class
- basic types are mapped easily
- “emulate” the rest with structs and interfaces
- reassurance: CPython 3.9 `Lib/pickle.py` < 2KLOC

GoPickle – basic unpickling



```
obj = {  
    'foo': 'hi',  
    'bar': 42,  
    'baz': {  
        'qux': [1, 'x']  
    }  
}
```

```
import pickle  
with open('./obj.pickle', 'wb') as file:  
    pickle.dump(obj, file)
```


GoPickle – basic unpickling



```
import pickle
with open('./obj.pickle', 'rb') as file:
    obj = pickle.load(file)
```


GoPickle – basic unpickling

```
$ go get -u github.com/nlpodyssey/gopickle
```

```
 import "github.com/nlpodyssey/gopickle/pickle"  
  
// ...  
  
obj, err := pickle.Load("./obj.pickle")
```

GoPickle – basic unpickling



```
{  
  'foo': 'hi',  
  'bar': 42,  
  'baz': {  
    'qux': [1, 'x']  
  }  
}
```



```
&Dict{  
  &DictEntry{Key: "foo", Value: "hi"},  
  &DictEntry{Key: "bar", Value: 42},  
  &DictEntry{Key: "baz", Value: &Dict{  
    &DictEntry{  
      Key: "qux",  
      Value: &List{1, "x"},  
    },  
  }},  
}
```

(types from: github.com/nlpodyssey/gopickle/types)

GoPickle – built-in types

Python	GoPickle (gopickle/types)
list	List (ListAppender)
dict	Dict (DictSetter)
tuple	Tuple
set	Set (SetAdder)
frozenset	FrozenSet
bytearray	ByteArray
collections.OrderedDict	OrderedDictClass, OrderedDict

GoPickle – custom objects




```
class Greeter:
    def __init__(self, name):
        self._name = name
    def greet(self):
        print(f'Hi, {self._name}!')

obj = Greeter('Gopher')

import pickle
with open('./obj.pickle', 'wb') as file:
    pickle.dump(obj, file)
```

GoPickle – custom objects

```
 obj, err := pickle.Load("./obj.pickle")  
if err != nil {  
    panic(err)  
}
```

```
panic: BUILD requires a PyDictSettable instance:  
&types.GenericObject{  
    ConstructorArgs: []interface {}{},  
    Class: (*types.GenericClass)(0xc0000c0020),  
}
```

↓

```
&GenericClass{Module: "__main__", Name: "Greeter"}
```

GoPickle – custom objects



```
type Greeter struct {  
    name string  
}
```

```
func (g *Greeter) PyDictSet(key, value interface{}) error {  
    if key == "_name" {  
        g.name = value.(string)  
        return nil  
    }  
    return fmt.Errorf("unexpected key: %v", key)  
}
```

satisfies interface `types.PyDictSettable`
emulates Python `object.__dict__[key] = value`

GoPickle – custom objects



```
type GreeterClass struct{}

func (c *GreeterClass) PyNew(args ...interface{}) (
    interface{}, error,
) {
    return &Greeter{}, nil
}
```

satisfies interface `types.PyNewable`
emulates Python `Class.__new__(...)`

GoPickle – custom objects



```
import "github.com/nlpodyssey/gopickle/pickle"
```

```
// ...
```

```
f, err := os.Open("obj.pickle")
```

```
if err != nil {
```

```
    panic(err)
```

```
}
```

```
defer f.Close()
```

```
unpickler := pickle.NewUnpickler(f)
```

...customize unpickler, then call unpickler.Load()

GoPickle – custom objects

```
⇒ GO unpickler.FindClass =  
    func(module, name string) (interface{}, error) {  
        if module == "__main__" && name == "Greeter" {  
            return &GreeterClass{}, nil  
        }  
  
        return nil, fmt.Errorf("class not found")  
    }
```

GoPickle – custom objects


```
⇒ GO obj, err := unpickler.Load()
    if err != nil {
        panic(err)
    }
    fmt.Printf("#v\n", obj)
```

```
&Greeter{
    name: "Gopher",
}
```

obj type is actually `interface{}`

GoPickle – custom objects

(the extra mile)

```
 func (g *Greeter) Greet() {  
    fmt.Printf("Hi, %s!\n", g.name)  
}
```

```
greeter := obj.(*Greeter)  
greeter.Greet()
```

Hi, Gopher!

GoPickle – advanced stuff

interfaces

Python	GoPickle (gopickle/types)
<i>ClassOrFunction(*args)</i>	Callable
<i>object.__new__(cls[, ...])</i>	PyNewable
<i>object.__setstate__(state)</i>	PyStateSettable
<i>object.__dict__[key] = value</i>	PyDictSettable
<i>setattr(object, name, value)</i>	PyAttrSettable

GoPickle – advanced stuff

Unpickler callbacks

Resolve custom classes and functions

```
u.FindClass = func(module, name string) (interface{}, error) { ... }
```

Resolve objects by persistent ID

```
u.PersistentLoad = func(persID interface{}) (interface{}, error) { ... }
```

Handle custom pickle extensions

```
u.GetExtension = func(code int) (interface{}, error) { ... }
```

GoPickle – advanced stuff

Unpickler callbacks

Handle out-of-band Buffers

```
u.NextBuffer = func() (interface{}, error) { ... }
```

Low-level function to handle pickle protocol 5 READONLY_BUFFER instruction

```
u.MakeReadOnly = func(obj interface{}) (interface{}, error) { ... }
```

BONUS: PyTorch models & spaGO



```
import torch  
obj = torch.load('./module.pt')
```



```
import "github.com/nlpodyssey/gopickle/pytorch"  
  
// ...  
  
model, err := pytorch.Load("./module.pt")
```

BONUS: PyTorch models & spaGO



PyTorch

```
import torch  
obj = torch.load('./module.pt')
```



```
import "github.com/nlpodyssey/gopickle/pytorch"  
  
// ...  
  
model, err := pytorch.Load("./module.pt")
```


BONUS: PyTorch models & spaGO

<https://github.com/nlpodyssey/spago>

```
$ ./do-cool-stuff --model=F00
```

 **Go & spaGO**

download model



Hugging Face models hub

models/F00/pytorch_model.bin
(+ other meta files)

load (w/ GoPickle)
& convert

do cool stuff

text classification, question answering,
machine translation, named entities recognition...

To Do

- More tests
- Documentation
- Better errors / inspection
- Support for more Python standard classes
- Support for more PyTorch classes
- Performance

Call To Action!

<https://github.com/nlpodyssey/gopickle>

- Share and ★
- Try it!
- Suggest or make fixes and improvements!
- Get in touch!
- Support us: <https://opencollective.com/nlpodyssey>