Using Immutable Data with Python

Josh Reed – Release Engineering – Aiven

Language

Expressing things in Language

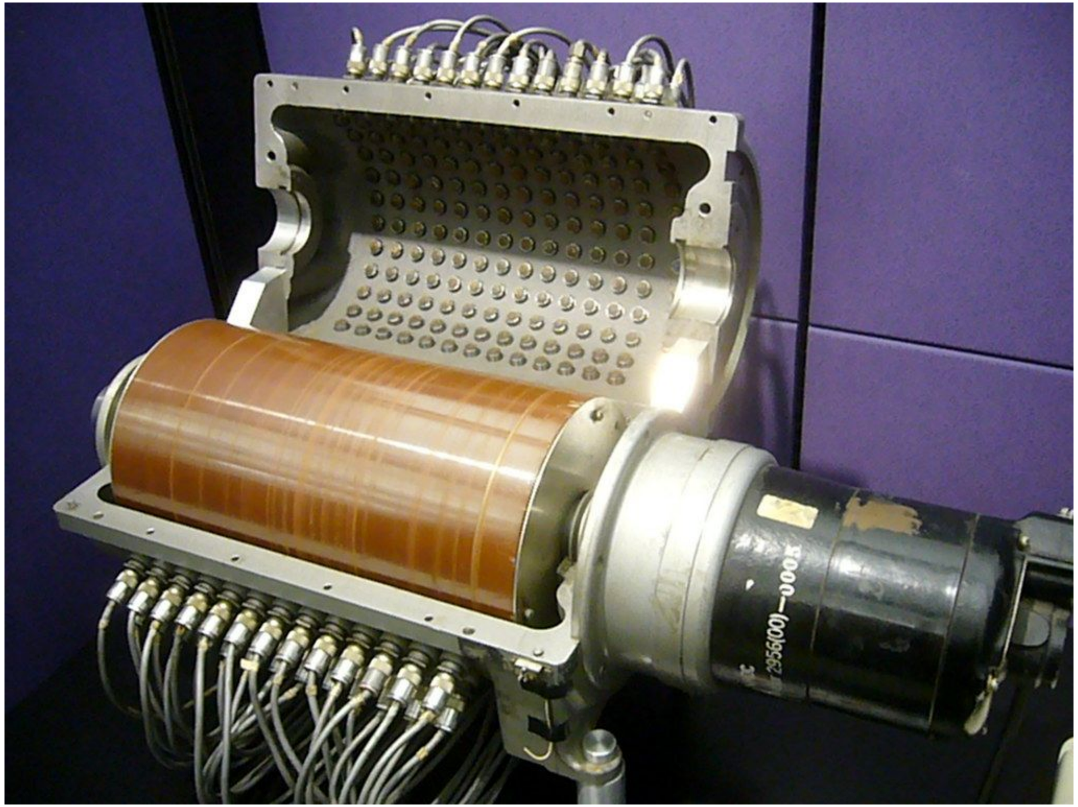
- Meaning
- Idiom
- Metaphor

Metaphor is the Model

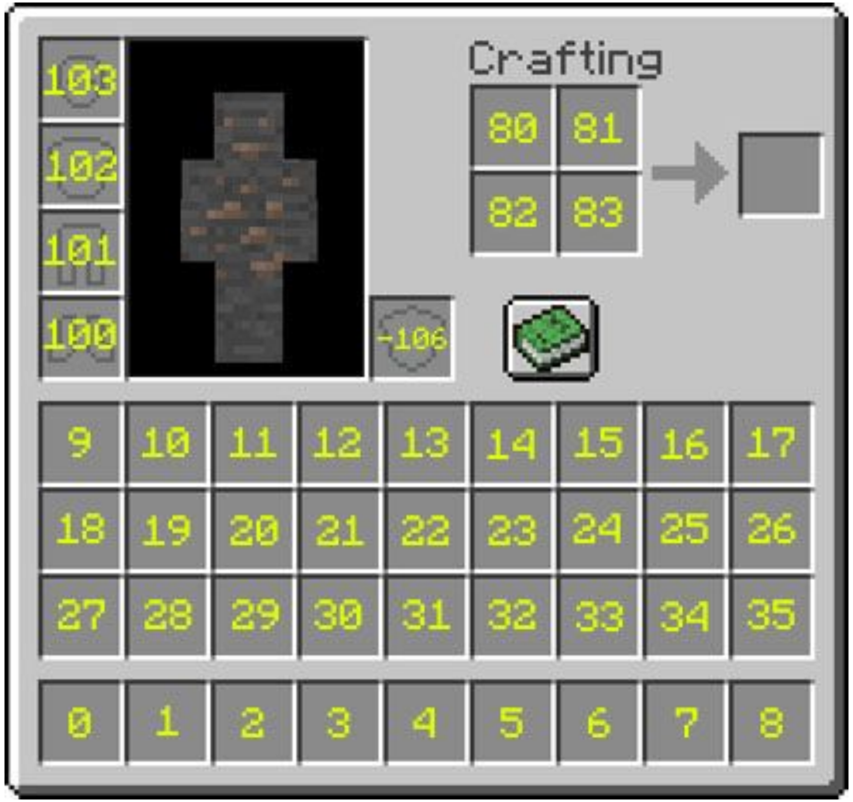
Metaphors of Programming

A small history

Early Days



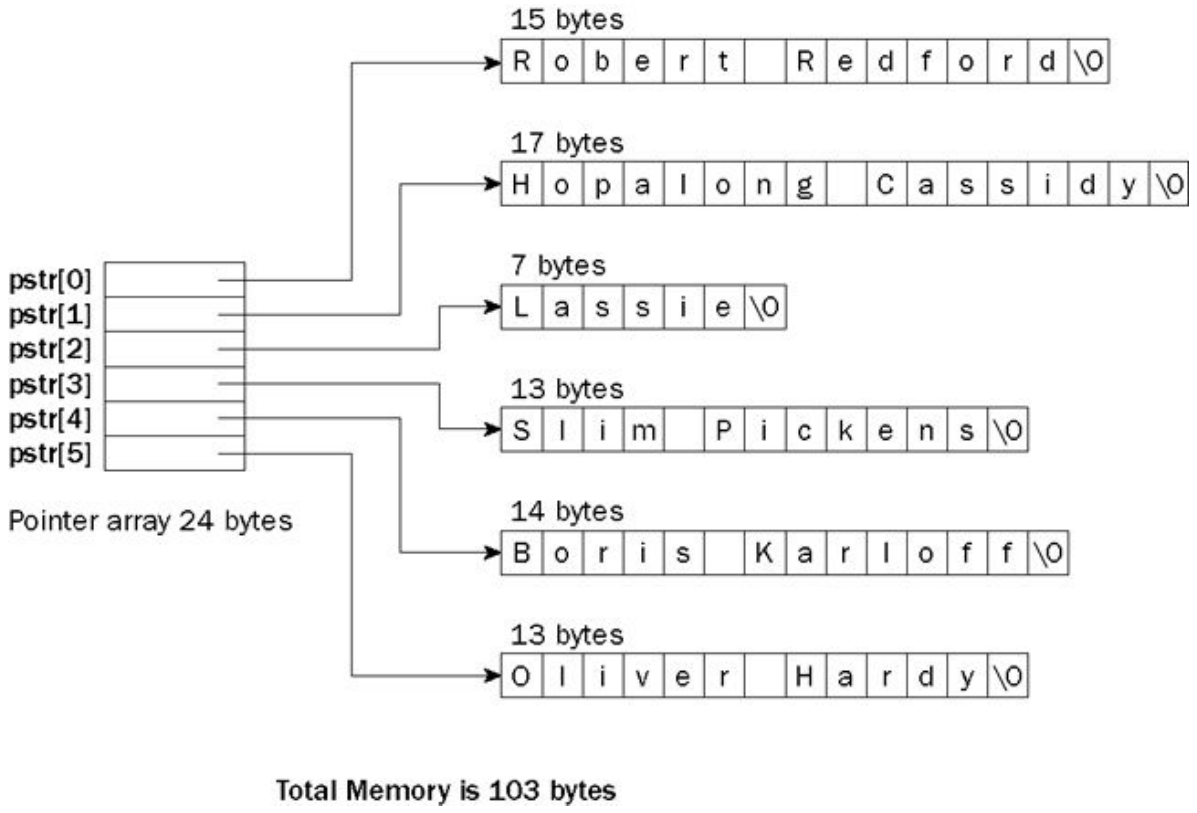
Addressable Buckets of Data



How do we deal with scale?

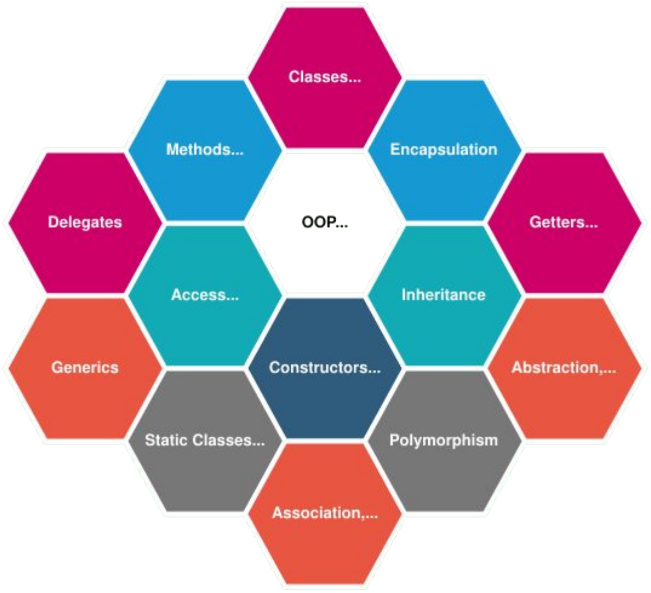
- More people?
- More memory?
- More programs?

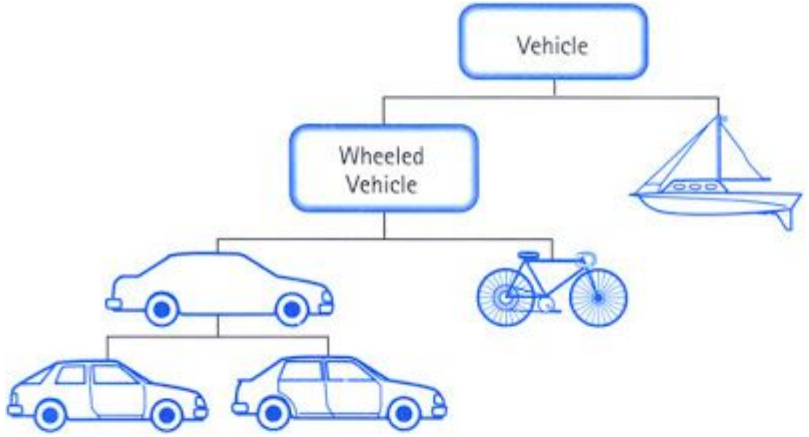
Names for addresses


15 bytes
R o b e r t   R e d f o r d \0
17 bytes
H o p a l o n g   C a s s i d y \0
7 bytes
L a s s i e \0
13 bytes
S l i m   P i c k e n s \0
14 bytes
B o r i s   K a r l o f f \0
13 bytes
O l i v e r   H a r d y \0
pstr[0]
pstr[1]
pstr[2]
pstr[3]
pstr[4]
pstr[5]
Pointer array 24 bytes
Total Memory is 103 bytes

Managed Memory

Object-Orientation





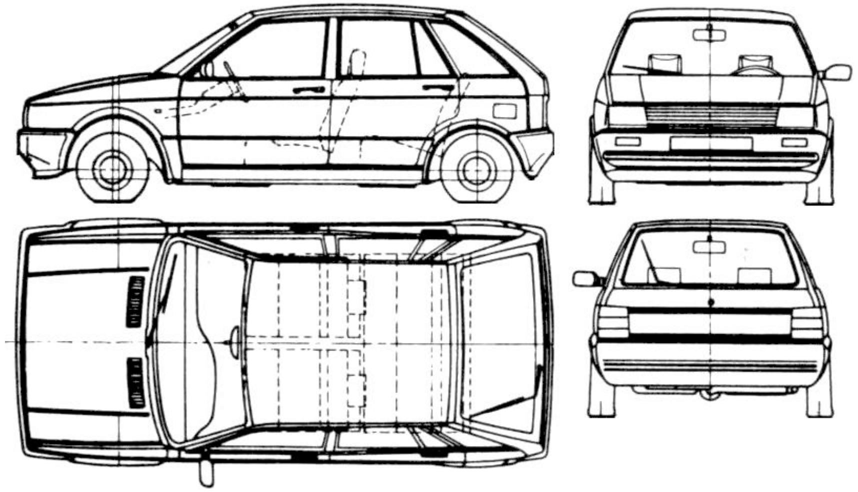
Where are we at?

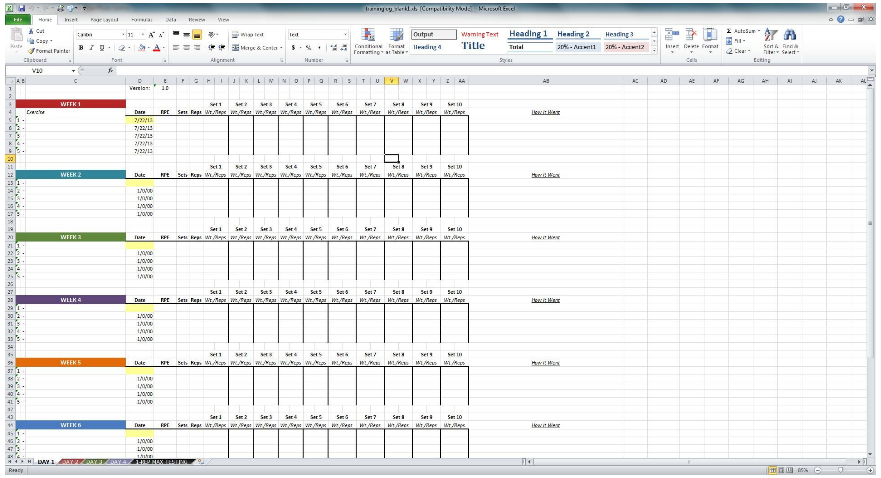
We have an **objects** metaphor

on top of a **reference** metaphor

on top of a **buckets** metaphor

How does this model data?





traininglog_blank1.xls [Compatibility Mode] - Microsoft Excel

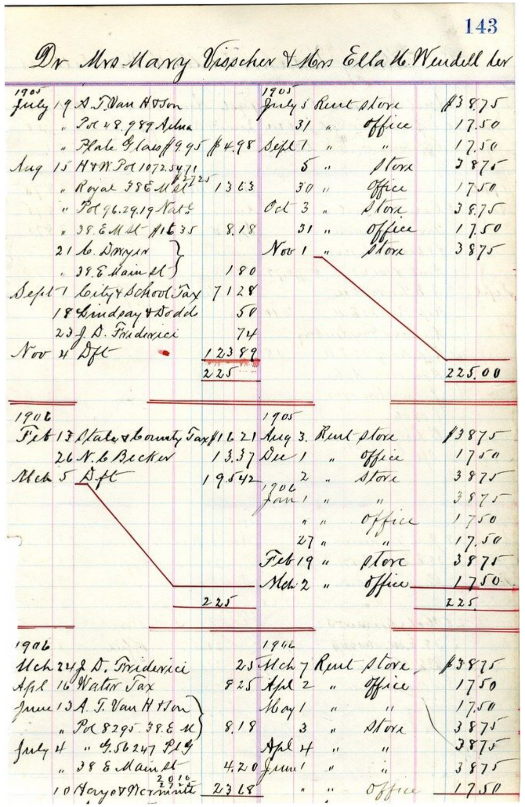
V10

	A	B	C	D	E	F	G	H–I	J–K	L–M	N–O	P–Q	R–S	T–U	V–W	X–Y	Z–AA	AB
1				Version:	1.0													
2																		
3	WEEK 1							Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8	Set 9	Set 10	
4			*Exercise*	Date	RPE	Sets	Reps	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*How It Went*
5	1	-		7/22/13														
6	2	-		7/22/13														
7	3	-		7/22/13														
8	4	-		7/22/13														
9	5	-		7/22/13														
10																		
11								Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8	Set 9	Set 10	
12	WEEK 2			Date	RPE	Sets	Reps	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*How It Went*
13	1	-																
14	2	-		1/0/00														
15	3	-		1/0/00														
16	4	-		1/0/00														
17	5	-		1/0/00														
18																		
19								Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8	Set 9	Set 10	
20	WEEK 3			Date	RPE	Sets	Reps	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*How It Went*
21	1	-																
22	2	-		1/0/00														
23	3	-		1/0/00														
24	4	-		1/0/00														
25	5	-		1/0/00														
26																		
27								Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8	Set 9	Set 10	
28	WEEK 4			Date	RPE	Sets	Reps	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*How It Went*
29	1	-																
30	2	-		1/0/00														
31	3	-		1/0/00														
32	4	-		1/0/00														
33	5	-		1/0/00														
34																		
35								Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8	Set 9	Set 10	
36	WEEK 5			Date	RPE	Sets	Reps	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*How It Went*
37	1	-																
38	2	-		1/0/00														
39	3	-		1/0/00														
40	4	-		1/0/00														
41	5	-		1/0/00														
42																		
43								Set 1	Set 2	Set 3	Set 4	Set 5	Set 6	Set 7	Set 8	Set 9	Set 10	
44	WEEK 6			Date	RPE	Sets	Reps	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*Wt./Reps*	*How It Went*
45	1	-																
46	2	-		1/0/00														
47	3	-		1/0/00														

DAY 1 | DAY 2 | DAY 3 | DAY 4 | 1-REP MAX TESTING

Ready

Data before computers



143

Dr Mrs Mary Visscher & Mrs Ella M. Wendell Cr

1905				1905			
July	19	A. T. Van Hoson		July	5	Rent Store	$38.75
	"	Pol 48.989 Aetna			31	" Office	17.50
	"	Plate Glass $9.95	$4.98	Sept	1	" "	17.50
Aug	15	H & W Pol 1072.5471			5	" Store	38.75
	"	Royal 38 E M St $27.25	13.63		30	" Office	17.50
	"	Pol 96.29.19 Natl		Oct	3	" Store	38.75
	"	38 E M St $16.35	8.18		31	" Office	17.50
	21	C. Dwyer		Nov	1	" Store	38.75
	"	38 E Main St	1.80				
Sept	1	City & School Tax	71.28				
	18	Lindsay & Dodd	.50				
	23	J. D. Friderici	.74				
Nov	4	Dft	123.89				
			225				225.00
1906				1905			
Feb	13	State & County Tax	$16.21	Aug	3.	Rent Store	$38.75
	26	N. C. Becker	13.37	Dec	1	" Office	17.50
Mch	5	Dft	195.42		2	" Store	38.75
				1906 Jan	1	" "	38.75
					"	" Office	17.50
					27	" "	17.50
				Feb	19	" Store	38.75
				Mch	2	" Office	17.50
			225				225
1906				1906			
Mch	24	J. D. Friderici	.25	Mch	7	Rent Store	$38.75
Apl	16	Water Tax	8.25	Apl	2	" Office	17.50
June	13	A. T. Van Hoson		May	1	" "	17.50
	"	Pol 8295. 38 E M	8.19		3	" Store	38.75
July	4	" G.56247 PLG		Apl	4	" "	38.75
	"	38 E Main St	4.20	June	1	" "	38.75
	10	Hayo & Mcrminte 20.10 27.22	23.68		"	" Office	17.50

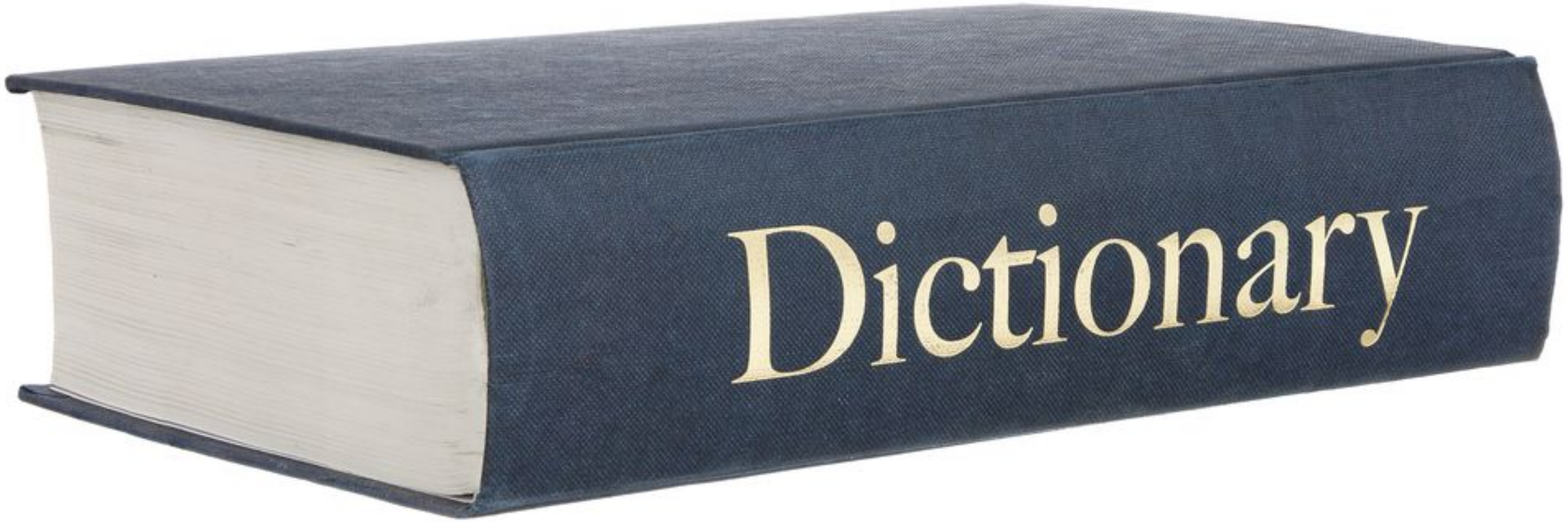
Values Metaphor for Data

- Don’t need to hide it
- Can use freely at boundaries
- Easy to reason about

Functions are value-based

Message Passing

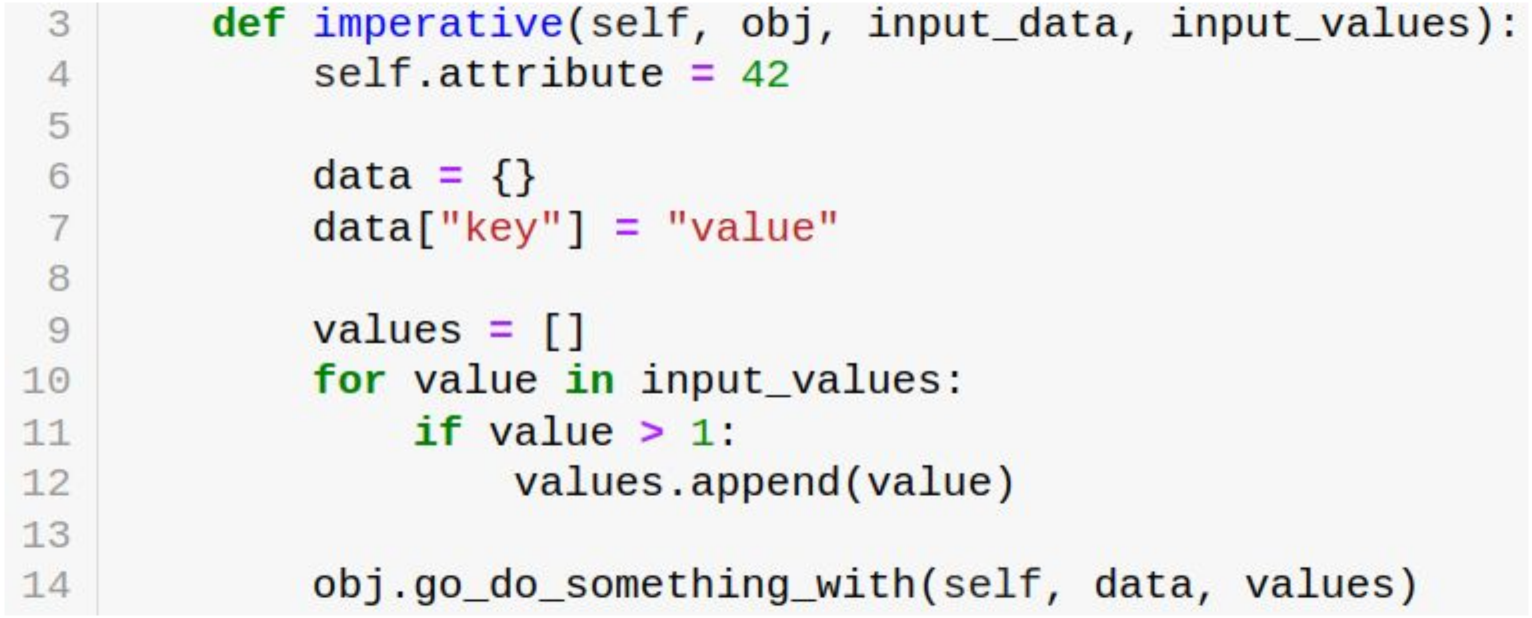
Python is a Bucket Language



Value Metaphors in Idiomatic Python

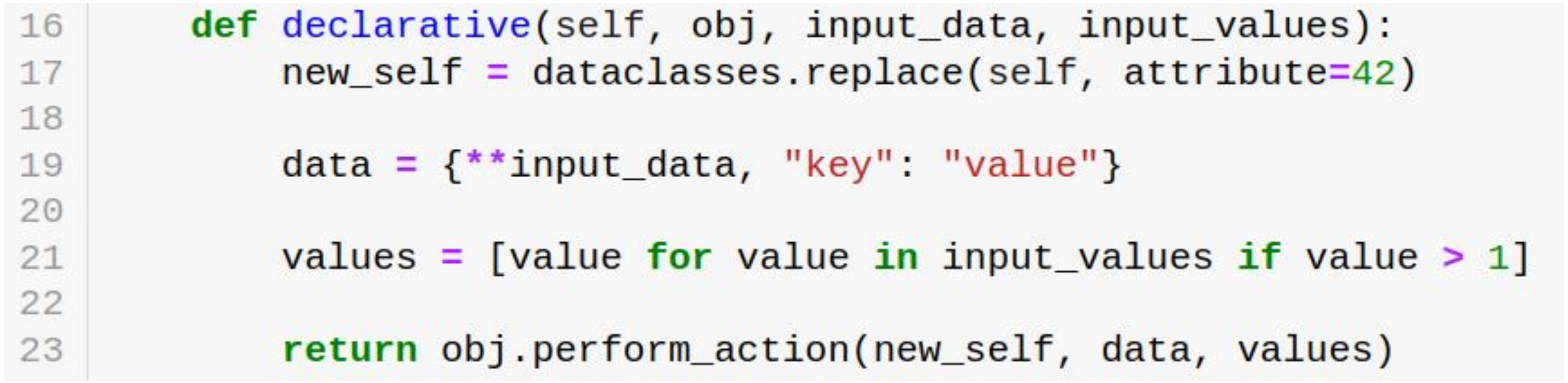
How can we adapt?

Do our idioms support this?



```python
    def imperative(self, obj, input_data, input_values):
        self.attribute = 42

        data = {}
        data["key"] = "value"

        values = []
        for value in input_values:
            if value > 1:
                values.append(value)

        obj.go_do_something_with(self, data, values)
```

Do our idioms support this?



```python
    def declarative(self, obj, input_data, input_values):
        new_self = dataclasses.replace(self, attribute=42)

        data = {**input_data, "key": "value"}

        values = [value for value in input_values if value > 1]

        return obj.perform_action(new_self, data, values)
```

Immutable things in Python

- Numbers
- Strings
- Tuples
- Frozenset … are we counting that?
- But no frozen dictionary

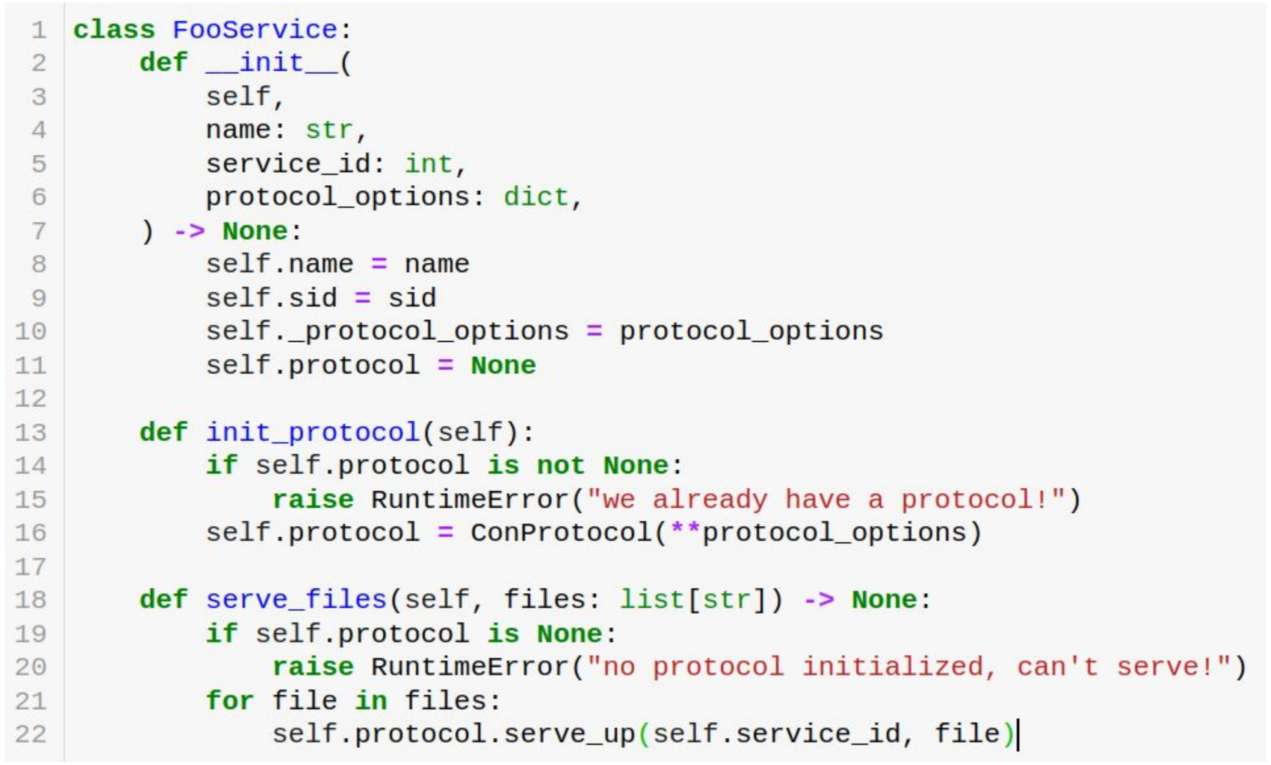
Could we use libraries?

- Not first-class citizens
- Often get poor efficiency
- Fights against common idioms

Just Don’t Mutate!

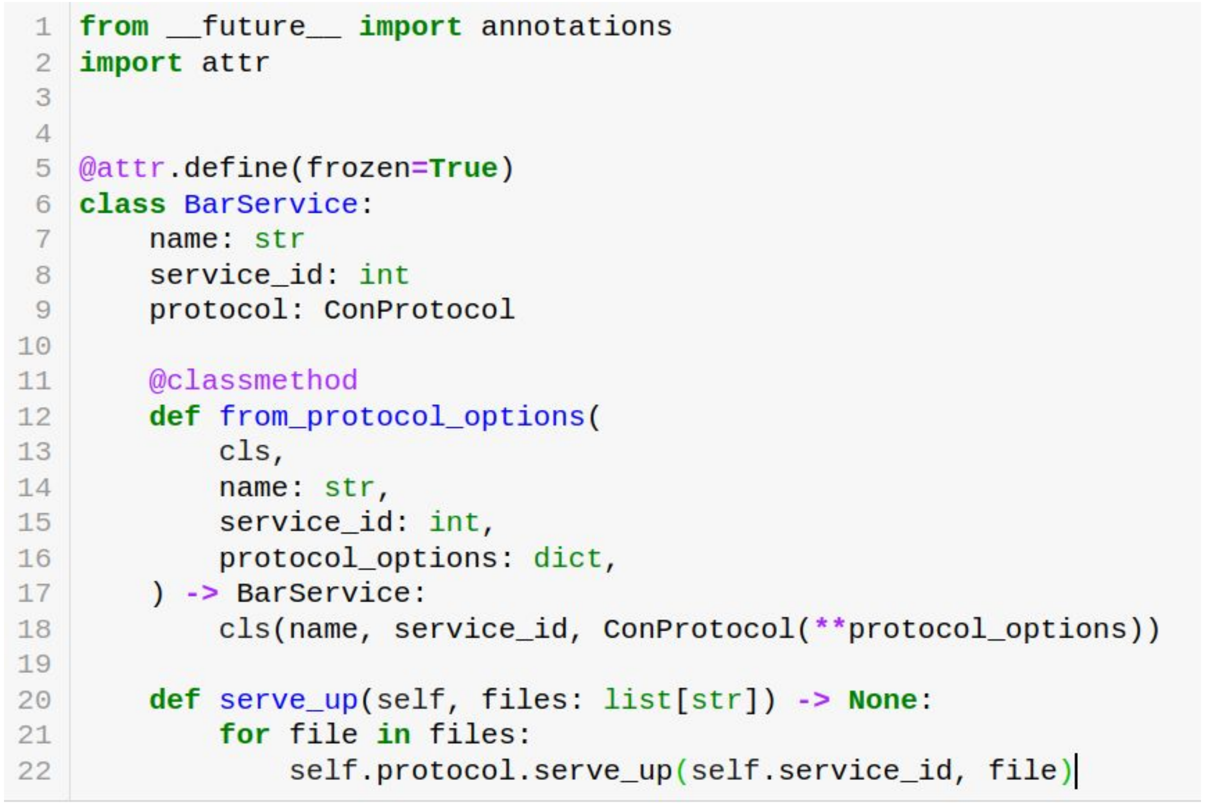
Avoid it as best you can

Avoid Partial Initialization



```python
class FooService:
    def __init__(
        self,
        name: str,
        service_id: int,
        protocol_options: dict,
    ) -> None:
        self.name = name
        self.sid = sid
        self._protocol_options = protocol_options
        self.protocol = None

    def init_protocol(self):
        if self.protocol is not None:
            raise RuntimeError("we already have a protocol!")
        self.protocol = ConProtocol(**protocol_options)

    def serve_files(self, files: list[str]) -> None:
        if self.protocol is None:
            raise RuntimeError("no protocol initialized, can't serve!")
        for file in files:
            self.protocol.serve_up(self.service_id, file)
```

Avoid Partial Initialization

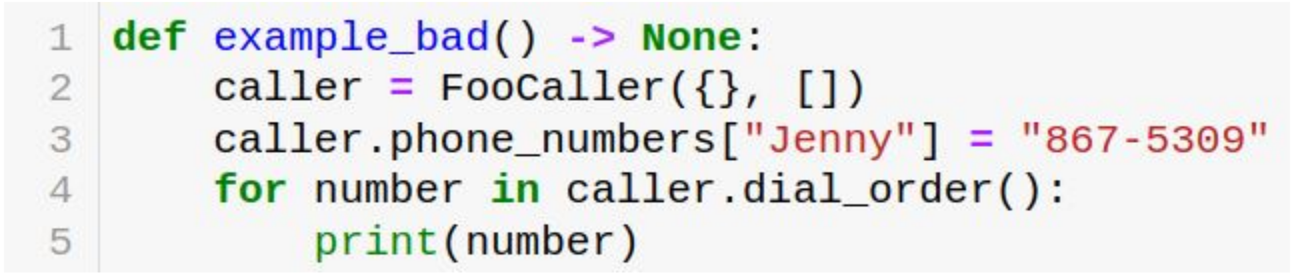


```python
from __future__ import annotations
import attr


@attr.define(frozen=True)
class BarService:
    name: str
    service_id: int
    protocol: ConProtocol

    @classmethod
    def from_protocol_options(
        cls,
        name: str,
        service_id: int,
        protocol_options: dict,
    ) -> BarService:
        cls(name, service_id, ConProtocol(**protocol_options))

    def serve_up(self, files: list[str]) -> None:
        for file in files:
            self.protocol.serve_up(self.service_id, file)
```

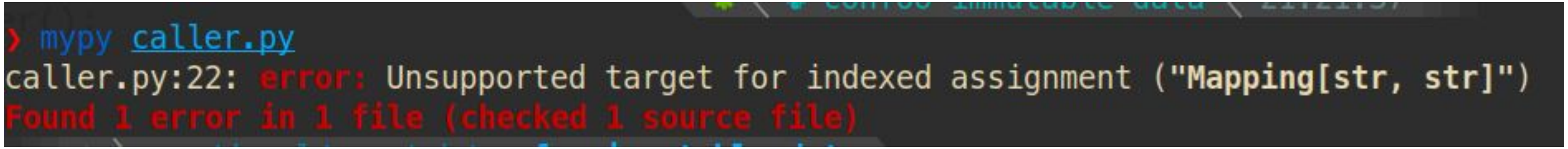
Use Typing to Your Advantage

```python
from collections.abc import Iterable, Collection, Mapping, Sequence
from dataclasses import dataclass


@dataclass(frozen=True)
class FooCaller:
    phone_numbers: Mapping[str, str]
    call_order: Sequence[str]
    excludes: Collection[str] = frozenset()

    def dial_order(self) -> Iterable[str]:
        return (
            self.phone_numbers[name]
            for name in self.call_order
            if name not in self.excludes
        )
```

Use Typing to Your Advantage



```python
def example_bad() -> None:
    caller = FooCaller({}, [])
    caller.phone_numbers["Jenny"] = "867-5309"
    for number in caller.dial_order():
        print(number)
```

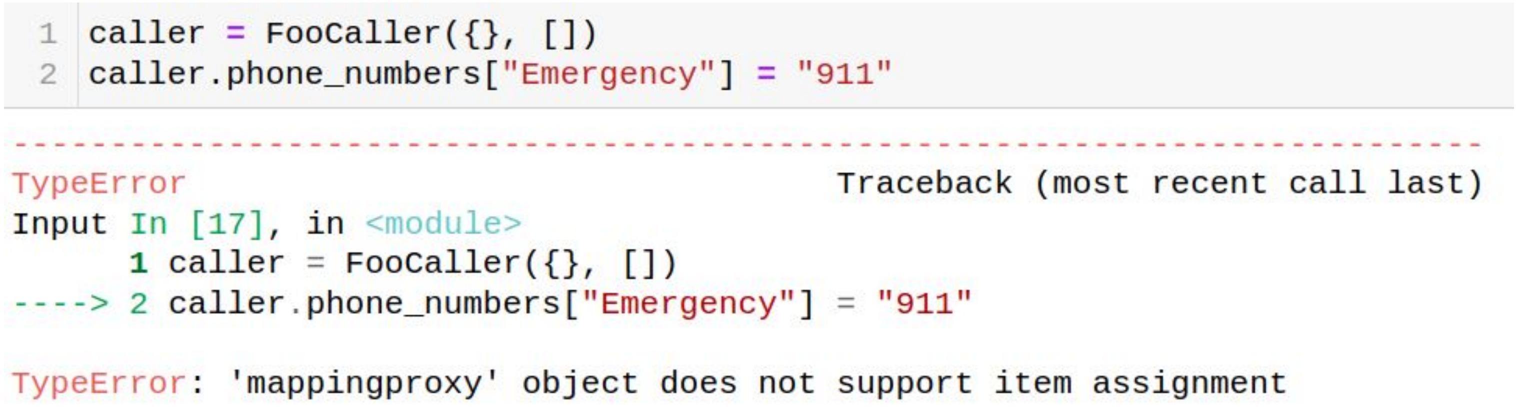


```
❯ mypy caller.py
caller.py:22: error: Unsupported target for indexed assignment ("Mapping[str, str]")
Found 1 error in 1 file (checked 1 source file)
```

Return Read-Only Objects

```python
import attr
from collections.abc import Iterable, Collection, Mapping, Sequence
from types import MappingProxyType


@attr.define(frozen=True)
class FooCaller:
    _phone_numbers: Mapping[str, str]
    _call_order: Sequence[str] = attr.field(converter=tuple)
    _excludes: Collection[str] = attr.field(default=frozenset(), converter=frozenset)

    @property
    def phone_numbers(self) -> Mapping[str, str]:
        return MappingProxyType(self._phone_numbers)

    @property
    def call_order(self) -> Sequence[str]:
        return self._call_order

    @property
    def excludes(self) -> frozenset[str]:
        return self._exludes

    def dial_order(self) -> Iterable[str]:
        return (
            self.phone_numbers[name]
            for name in self.call_order
            if name not in self.excludes
        )
```

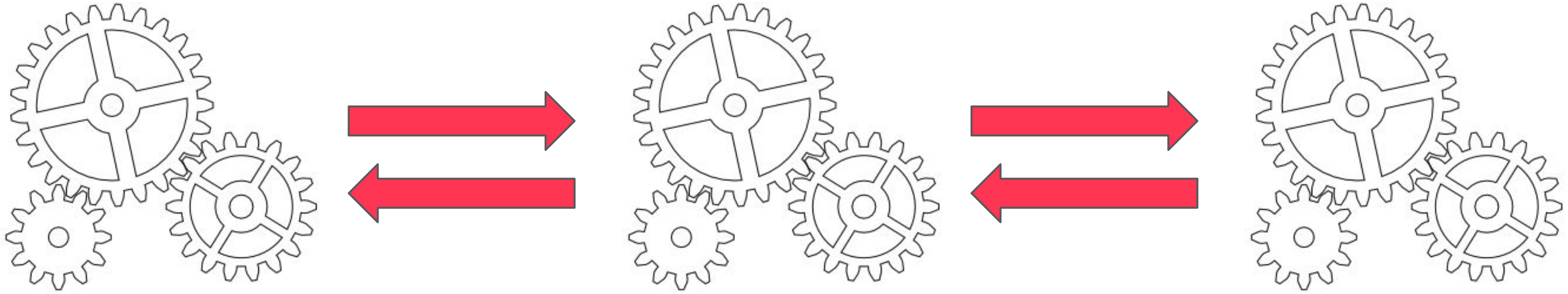
Return Read-Only Objects



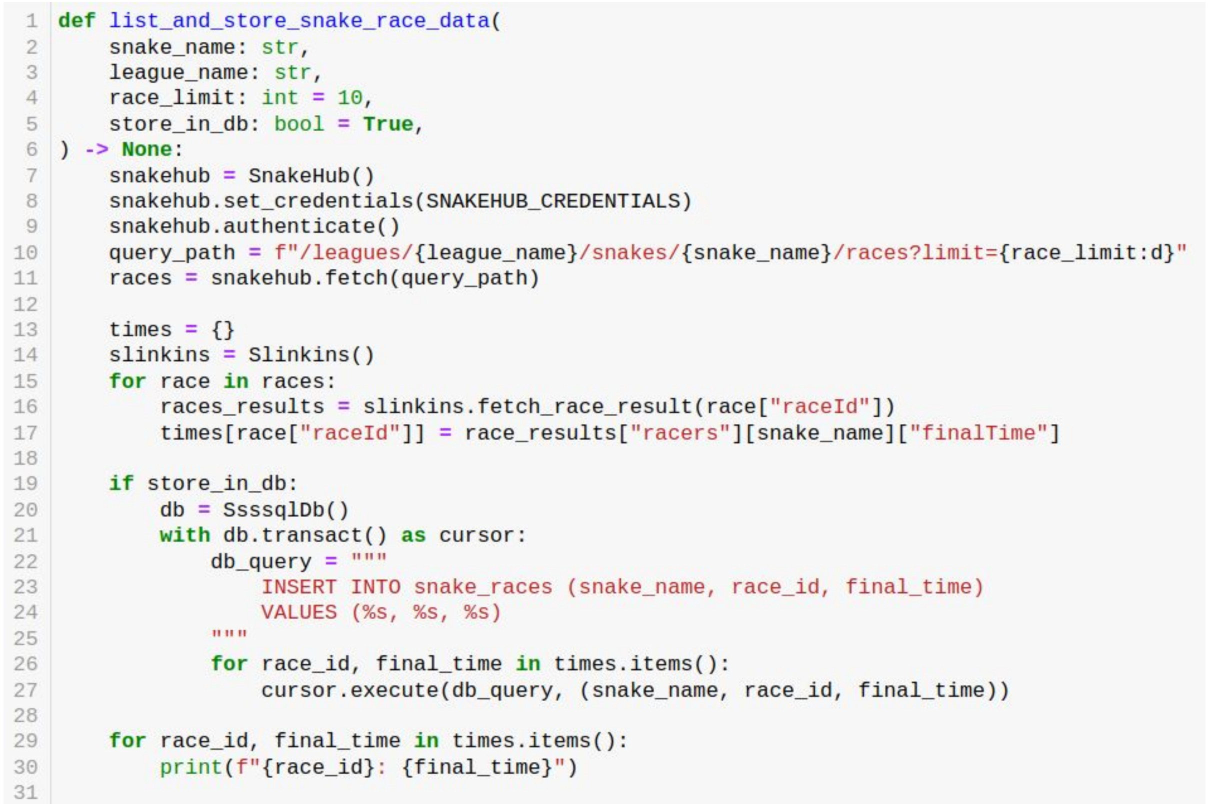
```python
caller = FooCaller({}, [])
caller.phone_numbers["Emergency"] = "911"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [17], in <module>
      1 caller = FooCaller({}, [])
----> 2 caller.phone_numbers["Emergency"] = "911"

TypeError: 'mappingproxy' object does not support item assignment
```

Let's put this all together



Snake Races: Existing Code



```python
def list_and_store_snake_race_data(
    snake_name: str,
    league_name: str,
    race_limit: int = 10,
    store_in_db: bool = True,
) -> None:
    snakehub = SnakeHub()
    snakehub.set_credentials(SNAKEHUB_CREDENTIALS)
    snakehub.authenticate()
    query_path = f"/leagues/{league_name}/snakes/{snake_name}/races?limit={race_limit:d}"
    races = snakehub.fetch(query_path)

    times = {}
    slinkins = Slinkins()
    for race in races:
        races_results = slinkins.fetch_race_result(race["raceId"])
        times[race["raceId"]] = race_results["racers"][snake_name]["finalTime"]

    if store_in_db:
        db = SsssqlDb()
        with db.transact() as cursor:
            db_query = """
                INSERT INTO snake_races (snake_name, race_id, final_time)
                VALUES (%s, %s, %s)
            """
            for race_id, final_time in times.items():
                cursor.execute(db_query, (snake_name, race_id, final_time))

    for race_id, final_time in times.items():
        print(f"{race_id}: {final_time}")

```

SnakeHub: Slots

```python
class SnakeHub:
    BASE_URL = "https://snakehub.io/api"

    def __init__(self):
        self._creds = None
        self._auth = None

    def set_credentials(self, creds):
        self._creds = creds
        self._auth = None

    def authenticate(self):
        r = requests.post(f"{self.BASE_URL}/authenticate", data={"creds": self.creds})
        self._auth = r.json()["token"]

    def fetch(self, path):
        headers = {}
        if self._auth:
            headers["token"] = self._auth
        return requests.get(f"{self.BASE_URL}/{path}")

```

SnakeHub: Data

```python
import dataclasses
from typing import Mapping

BASE_URL = "https://snakehub.io/api"

@dataclasses.dataclass(frozen=True)
class SnakeHub:
    base_url: str = BASE_URL
    headers: Mapping[str, str] = dataclasses.field(default_factory=dict)

    def authenticate(self, creds) -> "SnakeHub":
        r = requests.post(f"{self.base_url}/authenticate", data={"creds": creds})
        headers = self.headers | {"Token": r.json()["token"]}
        return dataclasses.replace(self, headers=headers)

    def fetch(self, path):
        return requests.get(f"{self.base_url}/{path}")

```

```python
def get_snakehub_query_path(snake_name: str, race_limit: int) -> str:
    return f"/leagues/{league_name}/snakes/{snake_name}/races?limit={race_limit:d}"
```

Slinkins

```python
import os

class Slinkins:
    def __init__(self):
        # Must contain username and password in URL to work
        self._url = os.getenv("SLINKINS_URL")

    def fetch_race_result(self, race_id):
        url = f"{self._url}/races/{race_id}/dataapi"
        return requests.get(url).json()

```

```python
import dataclasses
import os

@dataclasses.dataclass(frozen=True)
class Slinkins:
    # Must contain username and password in URL to work
    url: str = dataclasses.field(default_factory=lambda: os.getenv("SLINKINS_URL"))

    def fetch_race_result(self, race_id):
        url = f"{self._url}/races/{race_id}/dataapi"
        return requests.get(url).json()

    def fetch_race_results(self, race_ids):
        fetch1 = self.fetch_race_result
        return {race_id: fetch1(race_id) for race_id in race_ids}

```

Snake Race Database

```python
def db_from_conn_str(conn_str):
    driver = get_db_driver(conn_str)
    return driver.connect(conn_str)

def insert_race_times_into_db(db, times):
    with db.transact() as cursor:
        db_query = """
            INSERT INTO snake_races (snake_name, race_id, final_time)
            VALUES (%s, %s, %s)
        """
        for race_id, final_time in times.items():
            cursor.execute(db_query, (snake_name, race_id, final_time))
```

Snake Races: Final Glue Code

```python
from typing import Optional, Mapping

def get_snake_race_data(
    snake_name: str,
    league_name: str,
    race_limit: int = 10,
) -> None:
    snakehub = Snakehub().authenticate(SNAKEHUB_CREDENTIALS)
    query_path = get_snakehub_query_path(snake_name, race_limit)
    races = snakehub.fetch(query_path)

    slinkins = Slinkins()
    race_ids = [race["raceId"] for race in races]
    return {
        race_id: race_result["racers"][snake_name]["finalTime"]
        for race_id, race_result
        in slinkins.fetch_race_results(race_ids)
    }

def store_snake_data(db_conn_str: str, times: Mapping[str, float]) -> None:
    db = db_from_conn_str(db_conn_str)
    insert_race_times_into_db(db, times)

def print_snake_data(times: Mapping[str, float]) -> None:
    for race_id, final_time in times.items():
        print(f"{race_id}: {final_time}")
```

The point of immutability

- Sleep easy at night
- Test quickly and effectively
- Always ready for concurrency
- Design with confidence

You’ve been doing it the hard way

Values make it easier

Libraries

- *Attrs*
  Dataclasses on steroids
  https://www.attrs.org/en/stable/
- *Immutables*
  For when you actually need an efficient immutable mapping
  https://pypi.org/project/immutables/

Further study

- *Boundaries*
  Destroy All Software
  https://www.destroyallsoftware.com/talks/boundaries
- *The Value of Values*
  Rich Hickey
  https://youtube.com/watch?v=-6BsiVyC1kM
- The One Python Library Everyone Needs
  Glyph
  https://glyph.twistedmatrix.com/2016/08/attrs.html

Contact

Josh Reed

@jriddycuz

https://www.linkedin.com/in/josh-reed-3469383a/

Release Engineering @ aiven.io