

# Let's implement useless Python objects

役に立たない Python オブジェクトを作ろう

Hayao Suzuki

PyCon APAC 2023

October 28, 2023



# Share it

## GitHub

- <https://github.com/HayaoSuzuki/pyconapac2023>

## Hashtag on ~~Twitter~~ X

- #pyconapac #pyconapac2023 #pyconapac\_2 #pyconjp

# Who am I ?




## Who am I ? (お前誰よ)

Name Hayao Suzuki (鈴木 駿)

~~Twitter~~ X @CardinalXaro

Work Software Developer @ BeProud Inc.



- We are hiring <https://www.beproud.jp/careers/en/>
-  The event site for building connections
-  The best way to learn Python online
-  TRACERY A documentation service for system development

# Who am I ?

## Translated Books

- Python Distilled(O'Reilly Japan) New!

## Supervised Translated Books

- Introducing Python 2nd ed.(O'Reilly Japan)
- Robust Python(O'Reilly Japan)

# Who am I ?

## Selected My Talks

- Symbolic Mathematics using SymPy(PyCon JP 2018)
- How to Use In-Memory Streams(PyCon JP 2020)
- Unknown Evolution of the Built-in Function pow(PyCon JP 2021)

Listed at <https://xaro.hatenablog.jp/> .

Let's implement `useless` Python  
objects

# What is it mean useless?

## From LDOCE

- ① not useful or effective in any way
- ② (informal) unable or unwilling to do anything properly

# Is the useless object really useless?

From Zhuangzi Ren-jian shi (莊子 人間世篇)

人皆知有用之用 而莫知無用之用也

Everyone knows the usefulness of the useful, but no one knows the usefulness of the useless.



# Today's Theme

## Let's implement useless Python objects

The useless objects are useless, but how to make a useless object is very useful.

# What is a useless Python object?

## Example: LiarContainer

```
>>> c = LiarContainer(["spam", "egg", "bacon"])
>>> "spam" in c
False
>>> "tomato" in c
True
```

# What is a useless Python object?

## Example: FibonacciSized

```
>>> s = FibonacciSized(range(50))  
>>> len(s)  
12586269025
```

# What is a useless Python object?

## Example: ShuffledIterable

```
>>> it = ShuffledIterable([1, 2, 3, 4, 5])
>>> for _ in range(3):
...     for v in it:
...         print(v, end=" ")
...     print()
...
5 3 4 2 1
4 1 2 3 5
2 5 3 1 4
```

# What is a useless Python object?

## Definition of a useless Python object in this talk

A useless Python object behave *Pythonic*, but does not work as expected.

# Data Structures and Operations

## Basic Data Structures of Python

List [1, 2, 3, 4, 5]

Tuple ("pen", "pineapple", "apple", "pen")

Dictionary {"Answer": 42}

Set {41, 43, 47, 53, 57, 59}

## Common Operations of Data Structure

len() Length of object

in Membership test

for Iteration

# in and Container

`object.__contains__()`

Called to implement membership test operators.

## Example: LiarContainer

```
class LiarContainer(Container):  
    def __contains__(self, item) -> bool:  
        return item not in self._data
```

# len() and Sized

`object.__len__()`

Called to implement the built-in function `len()`.

## Example: FibonacciSized

```
class FibonacciSized(Sized):  
    PHI: Final[float] = (1 + math.sqrt(5)) / 2  
    def __len__(self) -> int:  
        return math.floor(  
            (1 / math.sqrt(5)) * pow(self.PHI, len(self._data))  
            + (1 / 2)  
        )
```



# for and Iterable

```
object.__iter__()
```

Called when an iterator is required for a container.

## Example: ShuffledIterable

```
class ShuffledIterable(Iterable):  
    def __iter__(self) -> Iterator:  
        return iter(random.sample(self._data, k=len(self._data)))
```

# Object Protocols

## How to implement Pythonic Python objects

We need to understand [object protocols](#).

Ref: <https://docs.python.org/3/reference/datamodel.html>

## collections.abc

This module provides **abstract base classes** that can be used to test whether a class provides **a particular interface**.

From <https://docs.python.org/3/library/collections.abc.html>

## collections.abc and Interface

ABC	Interface
Sized	<code>__len__()</code>
Container	<code>__contains__()</code>
Iterable	<code>__iter__()</code>
Collection	Sized, Container, Iterable

# Collection

`collections.abc.Collection`

Sized and Container and Iterable

## Example: Collection

```
class UselessCollection(  
    FibonacciSized, LiarContainer, ShuffledIterable  
):  
    pass
```

## collections.abc and Built-in Objects

ABC	built-in objects
Sequence	tuple
MutableSequence	list
MutableSet	set
MutableMapping	dict

# Sequence

## Example: ModularSequence

```
class ModularSequence(Sequence):
    def __getitem__(self, key):
        if isinstance(key, int):
            return self._data[key % len(self._data)]
        elif isinstance(key, slice):
            s = slice(
                key.start % len(self._data),
                key.stop % len(self._data),
                key.step,
            )
            return self._data[s]
        else:
            raise TypeError
```

# Sequence

## Example: ModularSequence

```
>>> seq = ModularSequence(range(20))
>>> print(seq[21:44])
[1, 2, 3]
>>> print(seq[65543])
3
>>> seq.count(13) # It does not stop.
```



## Example: MisprintedDictionary

```
class MisprintedDictionary(Mapping):
    def __init__(self, _dict: dict):
        shuffled_keys = random.sample(
            list(_dict.keys()), k=len(_dict.keys())
        )
        shuffled_values = random.sample(
            list(_dict.values()), k=len(_dict.keys())
        )
        self._data = dict(
            zip(shuffled_keys, shuffled_values)
        )
```

## Example: MisprintedDictionary

```
>>> d = MisprintedDictionary({"a": 1, "b": 2, "c": 3})
>>> for key in d:
...     print(f"d[{key}]={d[key]}", end=" ")
...
d[c]=3 d[b]=2 d[a]=1
```

## Example: CrowdSet

```
@functools.total_ordering
class CrowdSet(Set):
    def __init__(self, data=None):
        if data is not None:
            self._data = set(v for v in data)
        else:
            self._data = set()

    def __lt__(self, other):
        return self._data >= other
```

## Example: CrowdSet

```
>>> s = CrowdSet(("egg", "bacon", "spam"))  
>>> t = CrowdSet(("egg", "egg", "spam", "spam"))  
>>> s > t  
True
```

# Conclusion

## Let's implement useless Python objects

- Useless Python objects are useful
- `collections.abc` module is very useful
- Once you understand object protocol, you can do anything