Write Clean Python Code Using Pipes

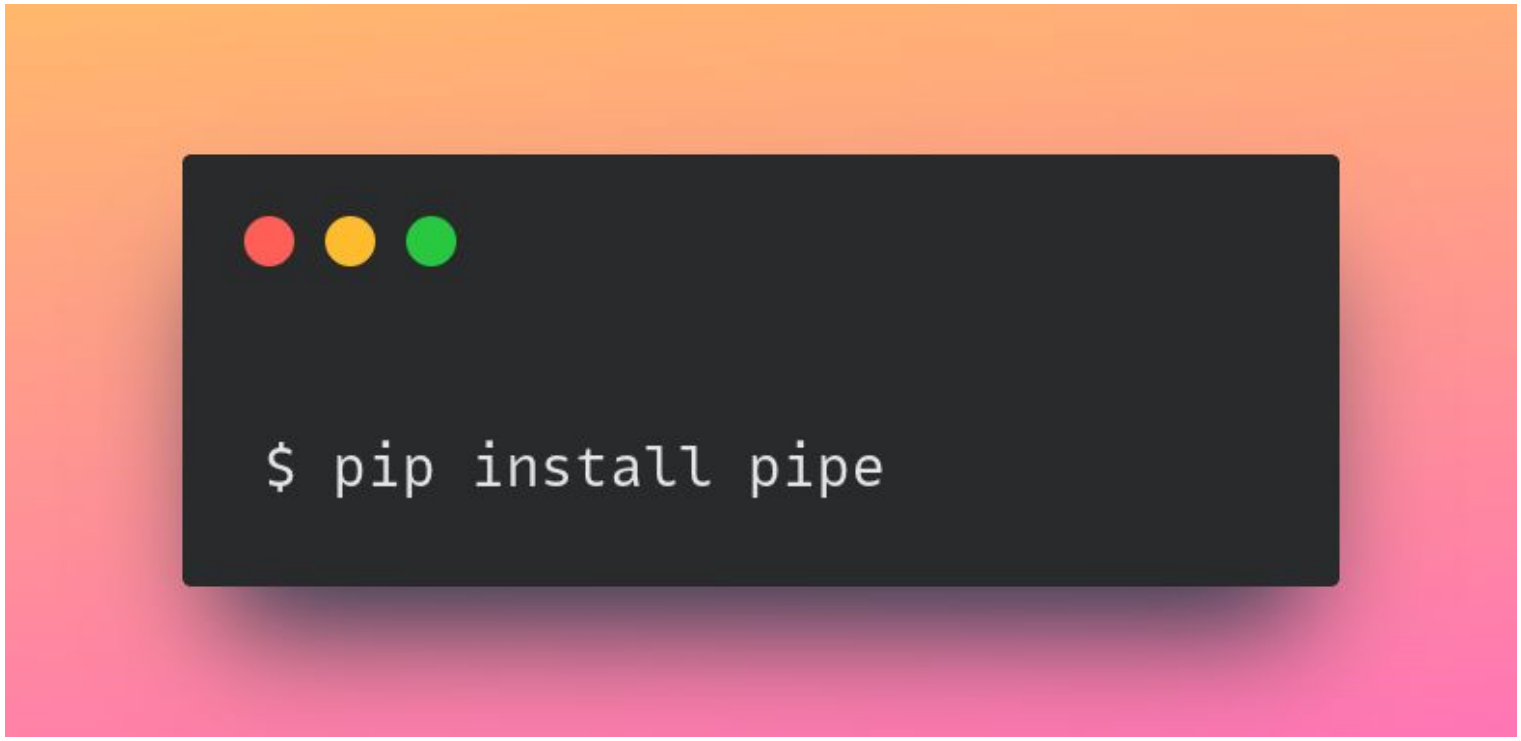
Khuyen Tran

@khuyentran1401

What is Pipe?

Pipe is a Python library that enables you to use pipes in Python. A pipe passes the results of one method to another method.

To install Pipe, type:



Khuyen Tran

@khuyentran1401

Where — Filter Elements in an Iterable

Similar to SQL, Pipe's where method can also be used to filter elements in an iterable.

```python
from pipe import where

arr = [1, 2, 3, 4, 5]
list(arr | where(lambda x: x % 2 == 0))
"""
[2, 4]
"""
```

Khuyen Tran

@khuyentran1401

Select — Apply a Function to an Iterable

The select method is similar to the map method. select applies a method to each element of an iterable.

```
from pipe import select

arr = [1, 2, 3, 4, 5]

list(arr | select(lambda x: x * 2)
"""
[2, 4, 6, 8, 10]
"""
```

Khuyen Tran

@khuyentran1401

But Why Not Map and Filter?

Now, you might wonder: Why do we need the methods where and select if they have the same functionalities as map and filter ?

It is because you can insert one method after another method using pipes. As a result, using pipes removes nested parentheses and makes the code more readable.

Khuyen Tran

@khuyentran1401

```python
from pipe import select, where

arr = [1, 2, 3, 4, 5]

# Instead of this
list(map(lambda x: x * 2,
     filter(lambda x: x % 2 == 0, arr)))

# Use pipe
list(arr
     | where(lambda x: x % 2 == 0)
     | select(lambda x: x * 2))
"""
[4, 8]
"""
```

Khuyen Tran

@khuyentran1401

chain — Chain a Sequence of Iterables

It can be a pain to work with a nested iterable. Luckily, you can use chain to chain a sequence of iterables.

```
from pipe import chain

nested = [[1, 2, [3]], [4, 5]]
list(nested | chain)

"""
[1, 2, [3], 4, 5]
"""
```

Khuyen Tran

@khuyentran1401

traverse — Recursively Unfold Iterables

The traverse method can be used to recursively unfold iterables. Thus, you can use this method to turn a deeply nested list into a flat list.

```
from pipe import traverse

list(nested | traverse)
"""
[1, 2, 3, 4, 5]
"""
```

Khuyen Tran

@khuyentran1401

Group Elements in a List

Sometimes, it might be useful to group elements in a list using a certain function. That could be easily done with the groupby method.

```
from pipe import groupby

list(
    (1, 2, 3, 4, 5, 6, 7, 8, 9)
    | groupby(lambda x: "Even" if x % 2==0 else "Odd")
    | select(lambda x: {x[0]: list(x[1])})
)
"""
[{'Even': [2, 4, 6, 8]}, {'Odd': [1, 3, 5, 7, 9]}]
"""
```

Khuyen Tran

@khuyentran1401

Combine groupby and where

To get only the values that are greater than 2, we can add the where method inside the select method:

```
list(
(1, 2, 3, 4, 5, 6, 7, 8, 9)
| groupby(lambda x: "Even" if x % 2==0 else "Odd")
| select(lambda x: {x[0]: list(x[1] | where(lambda x: x > 2))})
)
```

Khuyen Tran

@khuyentran1401