Functional Python

Unlock the power of functional python

This Talk

What we will be covering.

My motivations for this topic.

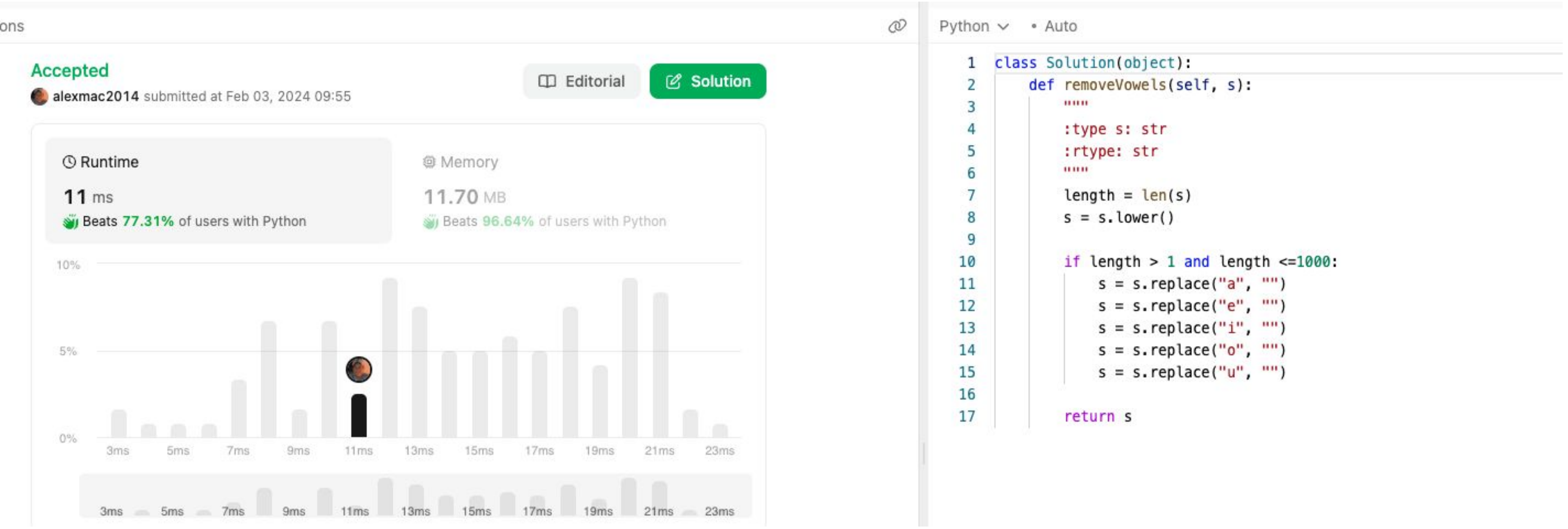
What is Functional Programming exactly?

What is it good for?

WHAT I WILL TALK ABOUT

- How I got interested in Functional Programming
- What is Functional Programming
- What use cases are good fits for Functional Programming

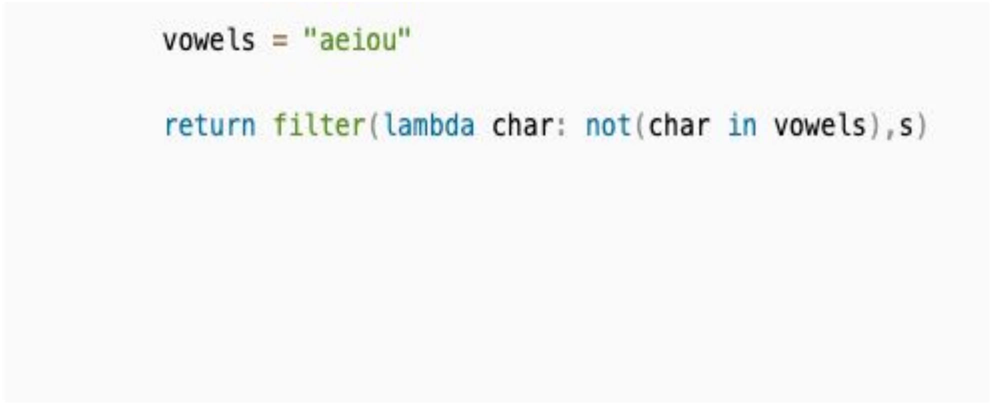
Motivation



```python
class Solution(object):
    def removeVowels(self, s):
        """
        :type s: str
        :rtype: str
        """
        length = len(s)
        s = s.lower()

        if length > 1 and length <=1000:
            s = s.replace("a", "")
            s = s.replace("e", "")
            s = s.replace("i", "")
            s = s.replace("o", "")
            s = s.replace("u", "")

        return s
```

Motivation

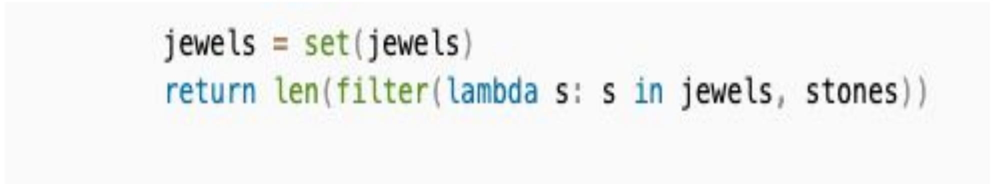
Notice how it is on one line, all packed in there.



```
vowels = "aeiou"

return filter(lambda char: not(char in vowels),s)
```

Motivation

Lots going on there. Functions calling functions.



```
jewels = set(jewels)
return len(filter(lambda s: s in jewels, stones))
```

My Way of Thinking about Functional Coding

What it is not:

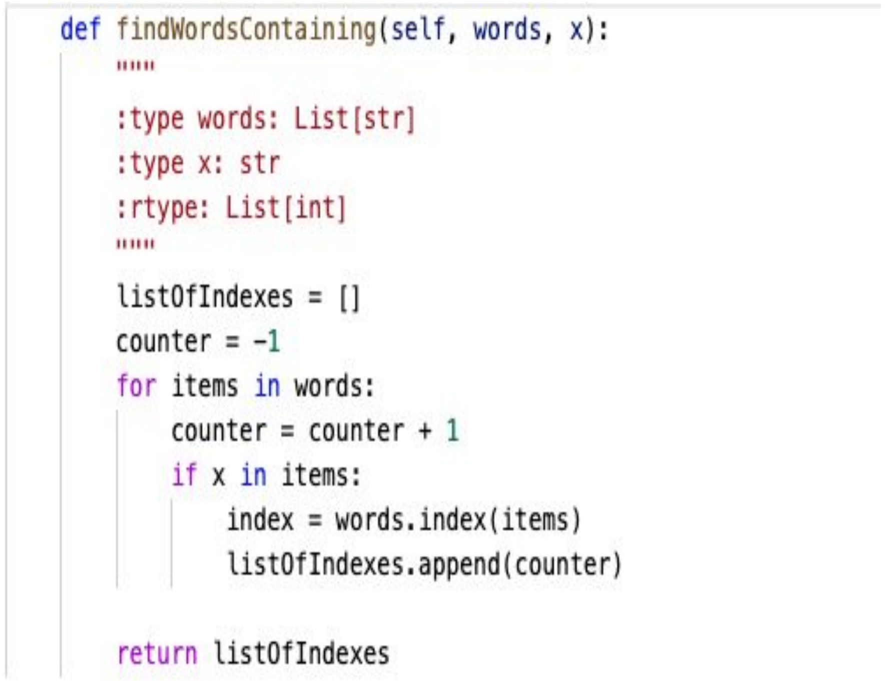
If statements

Lots of for loops

Mutable data structures

Step by step instructions

Counters



```
def findWordsContaining(self, words, x):
    """
    :type words: List[str]
    :type x: str
    :rtype: List[int]
    """
    listOfIndexes = []
    counter = -1
    for items in words:
        counter = counter + 1
        if x in items:
            index = words.index(items)
            listOfIndexes.append(counter)

    return listOfIndexes
```

My Way of Thinking about Functional Coding

<u>What it is:</u>

Functions that operate on data structures

Functions calling functions and returning functions

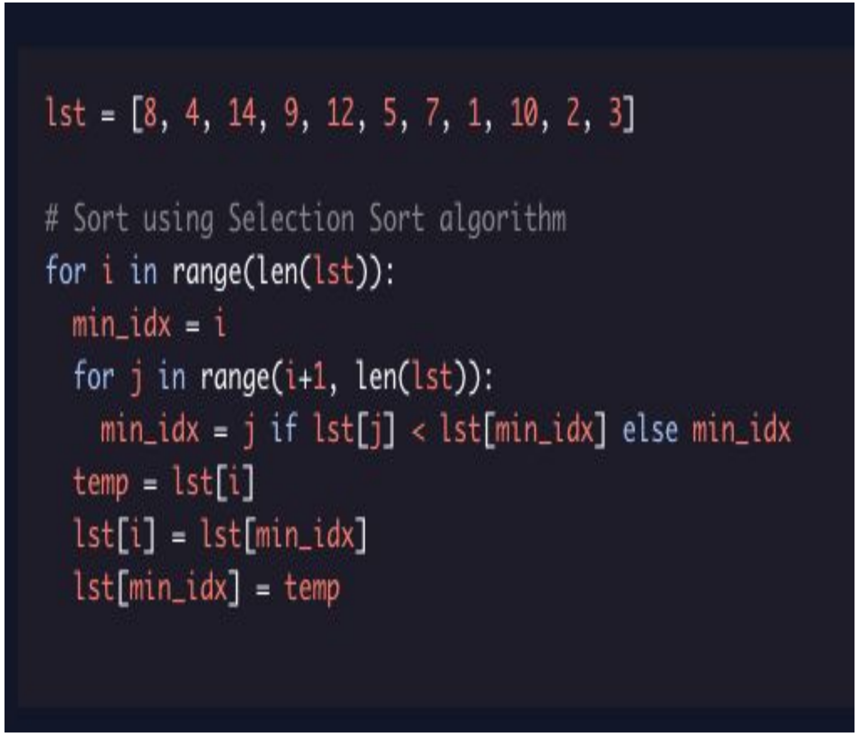
Immutable data structures

All the code is concise and packed into a line or a few succinct lines

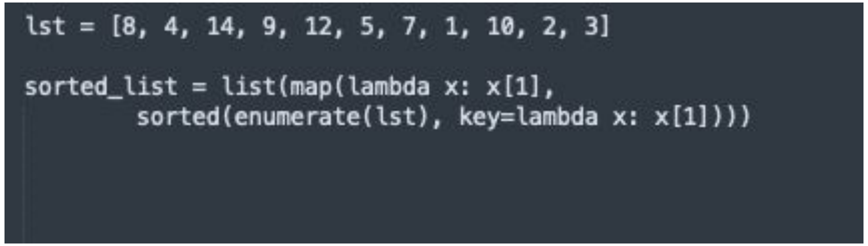
Iterable objects that return elements one at a time when it is time

```
nums =
(1,2,3,4,5,6,7,8)


s = reduce(lambda x,
y: x+y, map(lambda h:
h**2, nums))
```

Compare the styles - Side by Side



```python
lst = [8, 4, 14, 9, 12, 5, 7, 1, 10, 2, 3]

# Sort using Selection Sort algorithm
for i in range(len(lst)):
  min_idx = i
  for j in range(i+1, len(lst)):
    min_idx = j if lst[j] < lst[min_idx] else min_idx
  temp = lst[i]
  lst[i] = lst[min_idx]
  lst[min_idx] = temp
```



```python
lst = [8, 4, 14, 9, 12, 5, 7, 1, 10, 2, 3]

sorted_list = list(map(lambda x: x[1],
        sorted(enumerate(lst), key=lambda x: x[1])))
```

Each code snippet sorts a list.

Back to our Motivation

- Why is this functional style code beating my more imperative style of coding in runtime speed?

- It is not universally true that it is faster, but why was it faster on my leetcode problems?

Under the hood: Optimizations by compilers and runtimes

So What is Functional Coding?

Let's Define Functional Programming

A programming paradigm where:

The primary method of computation is function evaluation

Also, there are a lot of characteristics that we strive for in this paradigm

Pure Functions

First Order Functions

First Class Functions

Avoid Side Effects + State

Immutability

- Lazy Eval
- map()

Map()
filter()
Reduce()

```
numbers = [1,2,3,4,5]

squared_numbers = map(lambda x:x**2,
numbers)


results = filter(lambda x: x.nobel is
True, scientists_tuple)

add_lambda = lambda x, y: x + y
```

Comprehensions
Generators
Lambda Functions

Recursion vs. Iteration

Declarative
What vs. How

So What is Functional coding Good at?

What is Functional Programming Good for in real life?

- Caching - memoization
  - Stores the results of calls
  - Pure functions + immutable data
  - Immutable data - is easy to test equality
- Laziness (Lazy Evaluation)
  - Filter() returns an iterator object that processes at the time of use, not before
  - Processing streams of data
- Concurrency
  - Immutable data is good for concurrency because no thread can change it

- Data Transformation
  - Mapping, filtering, reducing
- Pattern Matching
- Domain-specific Language creation
  - SaSS for stylesheets
- Decorators
  - Flask uses them to define routes
  - Decorators pass functions as arguments and return functions and are pure functions

What is Functional Programming Good for in real life?

- Caching - memoization
- Laziness (Lazy Evaluation)
- Concurrency
- Data Transformation
- Pattern Matching
- Domain-specific Language creation
- Decorators

Caching and Memoization - faster

Lazy Eval - processing large data sets and streams of data

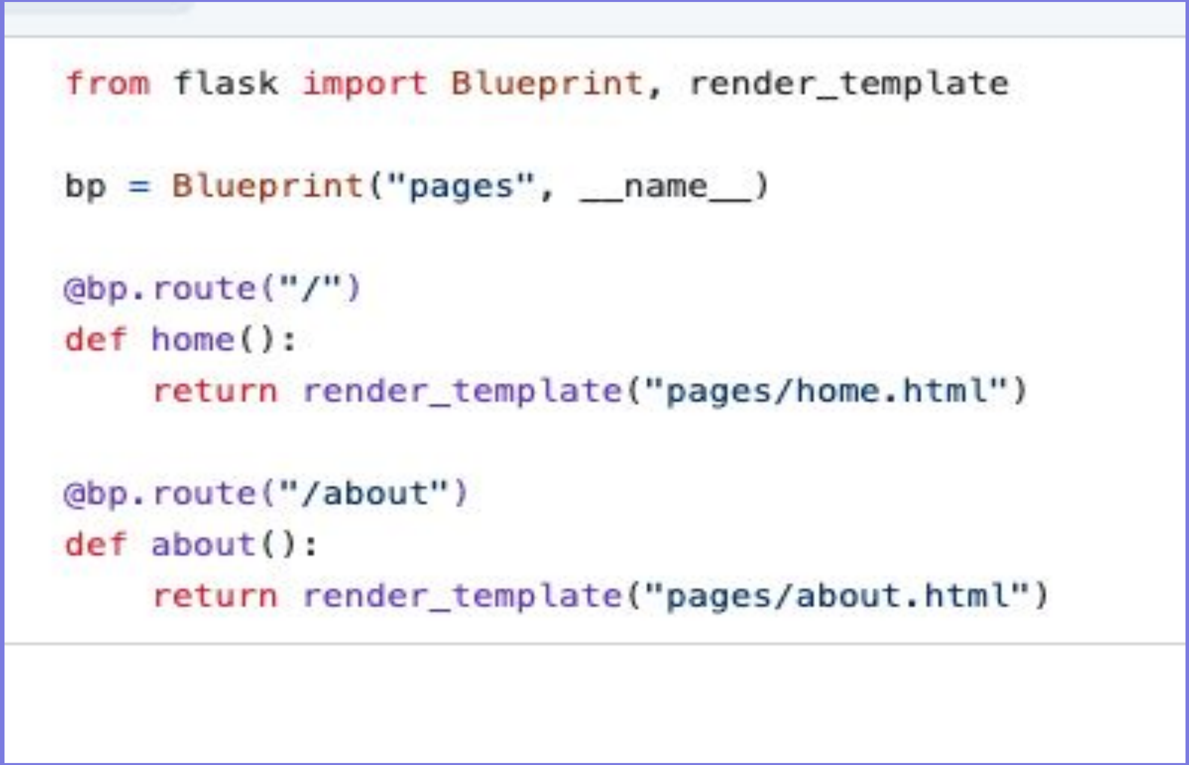
Concurrency and Bug reduction in general

Data Transformation

Pattern Matching

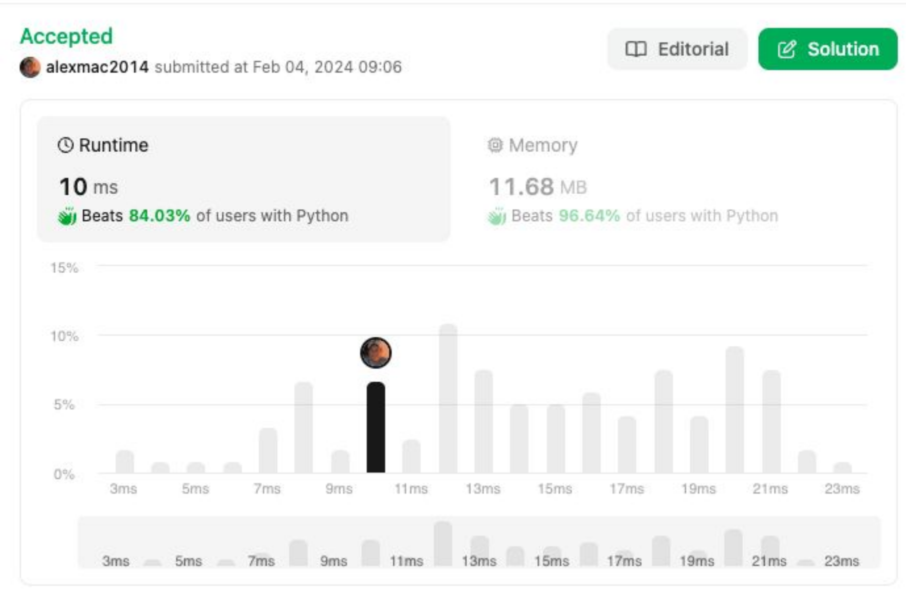
Domain-specific languages

DECORATORS



```python
from flask import Blueprint, render_template

bp = Blueprint("pages", __name__)

@bp.route("/")
def home():
    return render_template("pages/home.html")

@bp.route("/about")
def about():
    return render_template("pages/about.html")
```

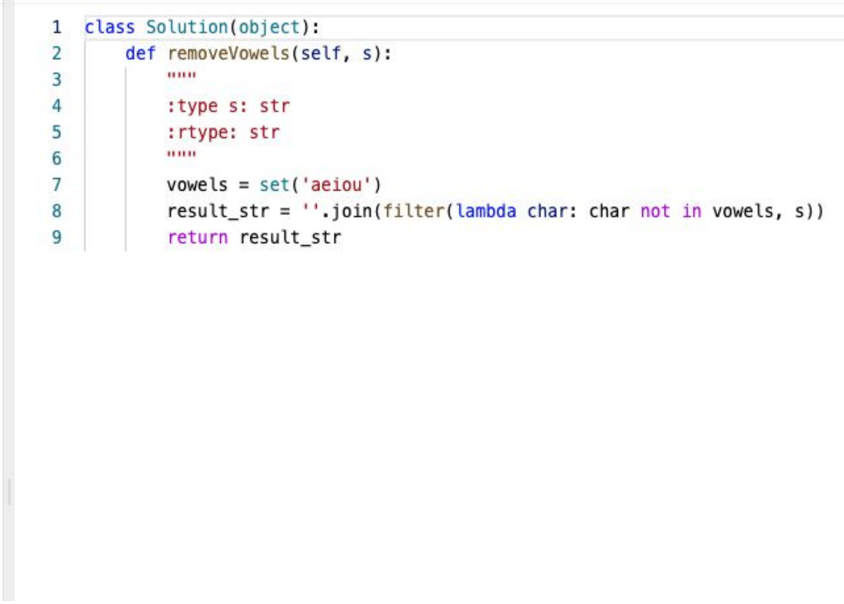
Orig problem with a functional solution



Accepted
alexmac2014 submitted at Feb 04, 2024 09:06

Runtime
10 ms
Beats 84.03% of users with Python

Memory
11.68 MB
Beats 96.64% of users with Python



```python
class Solution(object):
    def removeVowels(self, s):
        """
        :type s: str
        :rtype: str
        """
        vowels = set('aeiou')
        result_str = ''.join(filter(lambda char: char not in vowels, s))
        return result_str
```

Conclusions

What to remember.

Faster runtime optimizations in some cases

Functions and Immutable Data Structures

Bug reduction, Processing data streams, Parallel programming

What we talked about

- How I got interested in Functional Programming
- What is Functional Programming
- What use cases are good fits for Functional Programming

Where to learn More?

Alex McFerron

https://www.linkedin.com/in/alexmcferron/
Book a meeting: https://calendly.com/alexmac2010/

https://github.com/alexmac05/learnFunctionalPython/blob/main/README.md

This lists all of the resources I used for this Talk

Special thanks to my new coworker ChatGPT 3.5.