

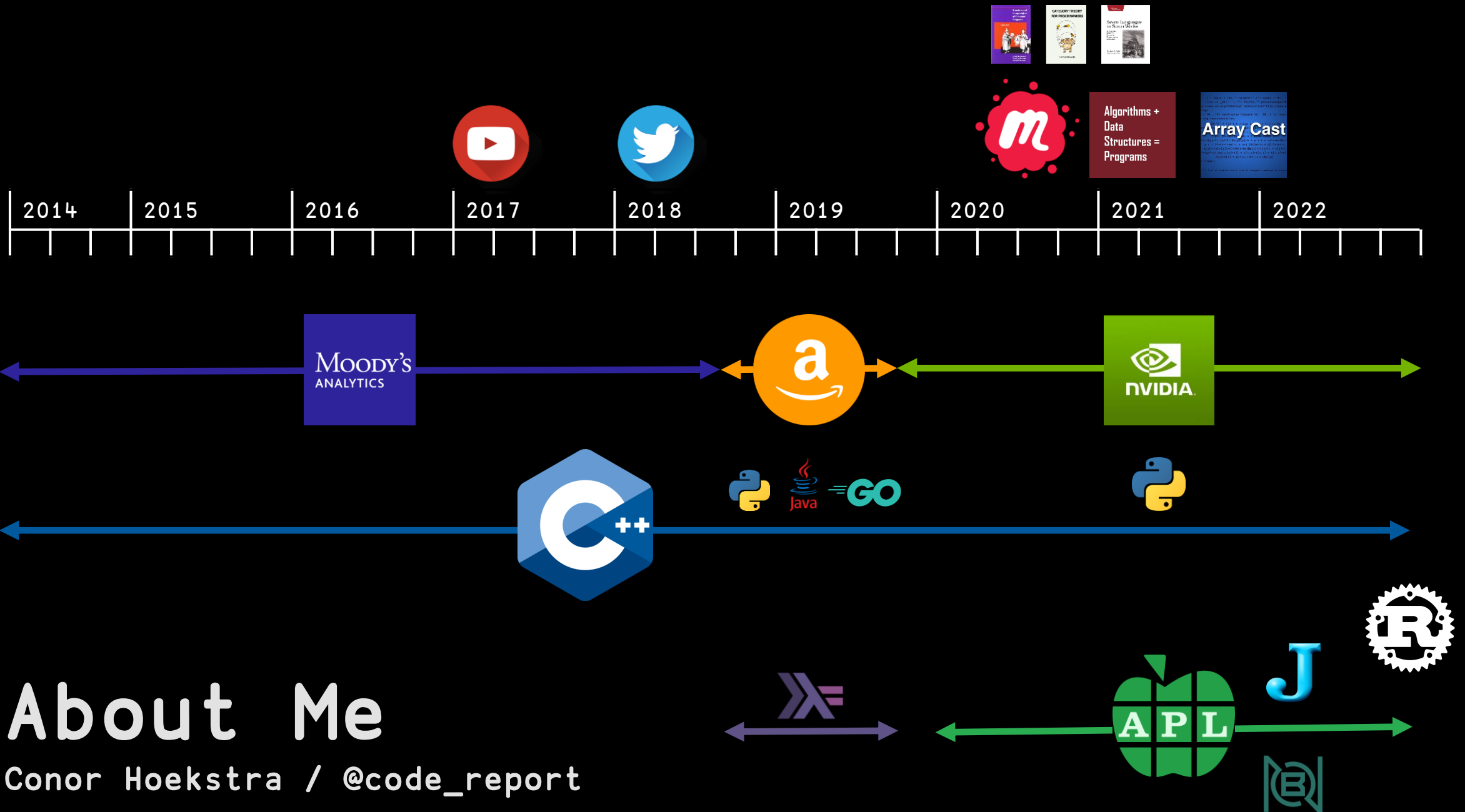
Beautiful **II** Python Refactoring

Conor Hoekstra



code_report







PyCon US

@PyConUS

22K subscribers

HOME

VIDEOS

PLAYLISTS

COMMUNITY

CHANNELS

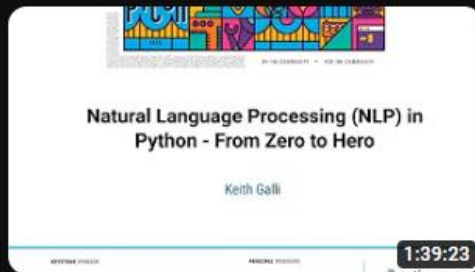
ABOUT



Subscribe

Recently uploaded

Popular



Tutorial: Keith Galli - Natural Language Processing (NLP) in Python - From Zero to...

41K views • 2 years ago

1:39:23



Tutorial: Sebastian Witowski - Modern Python Developer's Toolkit

35K views • 2 years ago

2:24:39



Tutorial: Santiago Basulto - Python Concurrency: from beginner to pro

25K views • 2 years ago

2:57:13



Talk: Conor Hoekstra - Beautiful Python Refactoring

21K views • 2 years ago

30:05

<https://youtu.be/W-lZttZhsUY>

Talk Python



To Me

Beautiful Pythonic Refactorings

Episode #275, published Sat, Aug 1, 2020, recorded Thu, Jul 9, 2020

▶ 0:00 / 0:00



Download

Transcript

Embed

CO₂ Neutral



Overcast



Apple



Google



Castbox



PocketCasts



RadioPublic



Spotify



Pro Edition



YouTube

Every episode in your player of choice

<https://talkpython.fm/episodes/show/275/beautiful-pythonic-refactorings>

**Algorithms +
Data
Structures =
Programs**

Array Cast



I refactored code from a

Blog Post



```
url = 'http://pokemondb.net/pokedex/all'

#Create a handle, page, to handle the contents of the website
page = requests.get(url)

#Store the contents of the website under doc
doc = lh.fromstring(page.content)

#Parse data that are stored between <tr>..</tr> of HTML
tr_elements = doc.xpath('//tr')

#Check the length of the first 12 rows
# [len(T) for T in tr_elements[:12]]

tr_elements = doc.xpath('//tr')

#Create empty list
col = []
i = 0

#For each row, store each first element (header) and an empty list
for t in tr_elements[0]:
    i += 1
    name = t.text_content()
    print('%d: "%s"'%(i, name))
    col.append((name, []))

#Since out first row is the header, data is stored on the second row onwards
for j in range(1, len(tr_elements)):
    #T is our j'th row
    T = tr_elements[j]

    #If row is not of size 10, the //tr data is not from our table
    if len(T) != 10:
        break

    #i is the index of our column
    i = 0

    #Iterate through each element of the row
    for t in T.iterchildren():
        data = t.text_content()
        #Check if row is empty
        if i > 0:
            #Convert any numerical value to integers
            try:
                data = int(data)
            except:
                pass
        #Append the data to the empty list of the i'th column
        col[i][1].append(data)
        #Increment i for the next column
        i += 1

# [len(C) for (title,C) in col]

Dict = {title:column for (title,column) in col}
df = pd.DataFrame(Dict)

print(df.head())
```





```
url = 'http://pokemondb.net/pokedex/all'

#Create a handle, page, to handle the contents of the website
page = requests.get(url)

#Store the contents of the website under doc
doc = lh.fromstring(page.content)

#Parse data that are stored between <tr>..</tr> of HTML
tr_elements = doc.xpath('//tr')

col = [(t.text_content(), []) for t in tr_elements[0]]

for T in tr_elements[1:]:
    for i, t in enumerate(T.iterchildren()):
        data = t.text_content()
        col[i][1].append(int(data) if data.isnumeric() else data)

Dict = {title:column for (title,column) in col}
df    = pd.DataFrame(Dict)

print(df.head())
```




```
url = 'http://pokemondb.net/pokedex/all'

page = requests.get(url) # page handle
doc = lh.fromstring(page.content) # website contents
tr = doc.xpath('//tr') # html <tr> data
titles = [t.text_content() for t in tr[0]] # column titles

fmt = lambda data : int(data) if data.isnumeric() else data
cols = zip(*[fmt(t.text_content()) for t in T.iterchildren()]
           for T in tr[1:])

Dict = {title: column for title, column in zip(titles, cols)}
df = pd.DataFrame(Dict)

print(df.head())
```



```
url = 'http://pokemondb.net/pokedex/all'

header = { "User-Agent": # ...
r       = requests.get(url, headers=header)
df      = pd.read_html(r.text)[0]

print(df.head())
```



`enumerate()`
`list comprehensions`
`conditional expressions`
`slicing`



I refactored code from a

Blog Post



I refactored code from ...

HookStar 🏴‍☠️ ⭐

License MIT Python 3 Follow 1.3k Star 4 @code_report 5.6k

HookStar ⚡ ⭐

						A	G	I O							382 (14)	305 (29)		
						NMX	U	T M N P H S	H				W	0	1: HM (15)			
						F T L D Z M N P A H K Y B	A	R	M	Y		J	E	E				
						0	N	O E A U		O	I		L		3: ETH (10)			
							W O R K S		A	R		O E A	D	O E A				
						0	I			B	E	A	R	S				
							C		T	E	P	A	S	I E L T X D G M N A H R Y S W		6: HAY (9)		
							D	E	N	O	T	E	S	E A L T B X G M P H A K O S Q		7: HA (9)		
							R	A	M		T	H	O	L	I	8: HAND (8)		
							Q		Z	E	E			B	I A O			
							U		X	I		T	I	P	I			
							V	E	G	E	S		A H F T D N S	I E A O F T D N S	P M K			
							A	Y		D			F T L D C G N K S J W	O A 0				
							I						N M X		N T M S			
															I A O			
							F	R	I	E	R		B	E	A	N	O	S

H L N T

{hmm=interj} [interj] || used to express thoughtful consideration [interj]
0 T:1 U:1

HookStar

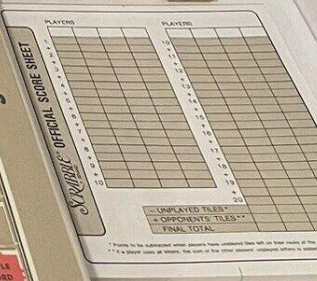
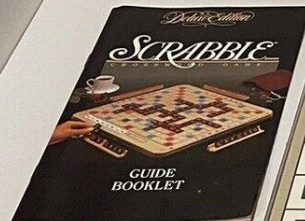
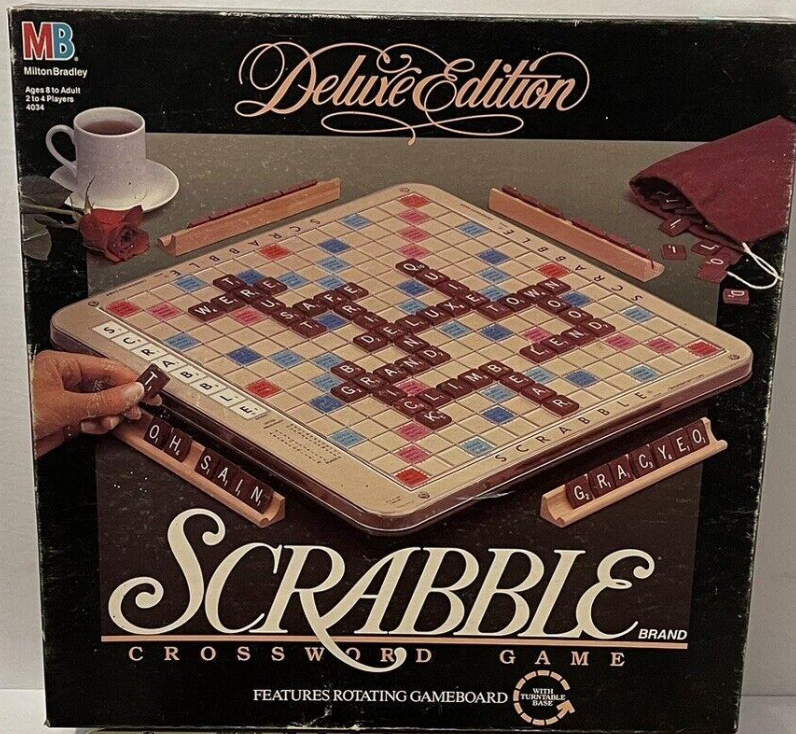
License [MIT](#) Python [3](#) [Follow](#) 1.3k [Star](#) 4 [@code_report](#) 5.6k

HookStar 

						A	G	I	O							382 (14)	305 (29)													
						NMX	U	T	N	P	H	S	H		W	0	1: HM (15)													
						F	I	L	D	Z	H	K	Y	A	R	M	Y	J	E	E										
						O								N	O	I	L			3: ETH (10)										
						W	O	R	K	S				A	R		O	E	A	D										
						I								B	E	A	R	S												
						C		T	E	P	A	S		I	E	L	T	B	X	G	M	P	H	A	K	O	S		C	6: HAY (9)
						D	E	N	O	T	E	S																	O	7: HA (9)
						R	A	M						T	H	O	L	I	N	G										8: HAND (8)
						Q								Z	E	E													O	
						U								X	I															
						V	E	G	E	S				A	H															
						A	Y							F	T	D	N	I	E	A	O									
						I								L	B	R	R													
						F	R	I	E	R				B	E	A	N	O	S											

H L N T

{hmm=interj} [interj] || used to express thoughtful consideration [interj]
0 T:1 U:1




Live Demo

Time to Refactor!

Commits on Oct 7, 2022

Add tiles remaining to UI


 codereport committed on Oct 7



5884e67



REFACTOR: remove duplication


 codereport committed on Oct 7



5b75f41



REFACTOR: name the algorithm - [::-1].index()


 codereport committed on Oct 7



b7eb49b



REFACTOR: Replace word_info with Play @DataClass


 codereport committed on Oct 7



44f91a4



REFACTOR: replace two booleans with Phase enum


 codereport committed on Oct 7



67d6faa



Move classes/enums around


 codereport committed on Oct 7



5f68abb



REFACTOR: Position made @DataClass


 codereport committed on Oct 7



84e868c



Bug fix: showing bingo outline


 codereport committed on Oct 7



95ebc61



REFACTOR: combine (prefix|suffix)_tiles


 codereport committed on Oct 7



171db09



REFACTOR: deltas function


 codereport committed on Oct 7



c8c4aa9



REFACTOR: Make Cursor class

 codereport committed on Oct 7



37fb0da



Refactor #1

@dataclass

84e868c



```
class Position():  
    def __init__(self, dir, row, col):  
        self.row = row  
        self.col = col  
        self.dir = dir
```




```
class Position():  
    def __init__(self, dir, row, col):  
        self.row = row  
        self.col = col  
        self.dir = dir
```



```
class Position():
    def __init__(self, dir, row, col):
        self.row = row
        self.col = col
        self.dir = dir

    def __lt__(self, other):
        return self.row < other.row

    def __eq__(self, other):
        return self.row == other.row and \
               self.col == other.col and \
               self.dir == other.dir

    def __repr__(self):
        return str(self.tuple())


    def tuple(self):
        return (self.row, self.col, self.dir)
```



```
@dataclass(frozen=True, order=True)  
class Position():  
    dir: Direction  
    row: int  
    col: int
```



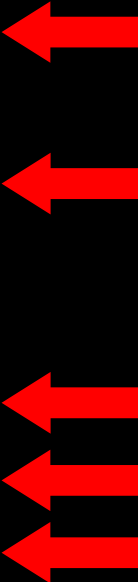
91		92	
92	- # TODO make immutable	93	+ @dataclass(frozen=True, order=True)
93	class Position():	94	class Position():
94	- def __init__(self, dir, row, col):	95	+ dir: Direction
95	- self.row = row	96	+ row: int
96	- self.col = col	97	+ col: int
97	- self.dir = dir		
98	-		
99	- def __lt__(self, other):		
100	- return self.row < other.row		
101	-		
102	- def __eq__(self, other):		
103	- return self.row == other.row and \		
104	- self.col == other.col and \		
105	- self.dir == other.dir		
106	-		
107	- def __repr__(self):		
108	- return str(self.tuple())		
109	-		
110	- def tuple(self):		
111	- return (self.row, self.col, self.dir)		
112		98	

@dataclass in  3.7

Refactor #2

f-strings

d8b1203



```
{hmm=interj} [interj] || used to express thoughtful consideration [interj]
0 T:1 U:1
```



```
str(render_row) + ": " + play.word + " (" + str(play.score) + ")"
```




```
f"{render_row}: {play.word} ({play.score})"
```



```
f"{render_row}: {play.word} ({play.score})"
```



```
str(render_row) + ": " + play.word + " (" + str(play.score) + ")"  
f"{render_row}: {play.word} ({play.score})"
```



```
tiles = ' '.join(a + ':' + str(b) for a, b in ...)
```



```
tiles = ' '.join(f"{a}:{b}" for a, b in ...)
```

[[digression]]



```
tiles = ' '.join(f"{a}:{b}" for a, b in ...)
```



```
tiles = ' '.join(f"{a}:{b}" for a, b in sorted(Counter(tile_bag[self.tile_bag_index:]).items()))
```




```
tiles = ' '.join(f"{a}:{b}"  
    for a, b in  
    sorted(  
        Counter(tile_bag[self.tile_bag_index:]).items()))
```

HookStar

						A	G	IO							382 (14)	305 (29)		
						NMX	U	TMNP HS	H				W		1: HM (15)			
						FTLD ZMNP AHKY B	A	R	M	Y		J	E	E				
				V			N	OE AU		O	I		L		3: ETH (10)			
				W	O	R	K	S		A	R		D					
				I					B	E	A	R	S					
				C		T	E	P	A	S	IELT BXDG MNAH RYSW	E		C	6: HAY (9)			
				D	E	N	O	T	E	S	EA	LTBX GMPS AKOS Q		O	7: HA (9)			
				R	A	M		T	H	O	L	I	n	G	8: HAND (8)			
				Q		Z	E	E			B	IAO		O				
				U		X	I		T	I	P	I		F	A	N		
				V	E	G	E	S		AH	FTDN S	IEAO	FTDN S	PMK	O	I		
FLBD CGMN PHRK YSJW				A	Y		D			FTLD CGNK SJSW	OA			NMX	U	NTMS		
				I		FTDM NPAH RYOB W				LBHR SW			A	N	D	L		
				F	R	I	E	R					B	E	A	N	O	S

H L N T

{hmm=interj} [interj] || used to express thoughtful consideration [interj]
0 T:1 U:1

			U			
D			L	I A O		
A	N	O	S			

{hmm=interj} [interj] || used to express thoughtful consideration [interj]

0 T:1 U:1





```
tiles = ' '.join(f"{a}:{b}"  
    for a, b in  
    sorted(  
        Counter(tile_bag[self.tile_bag_index:]).items()))
```



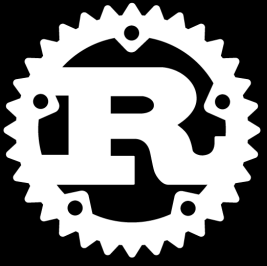
```
' '.join  
f"{a}:{b}"  
for a, b in  
sorted()  
Counter()  
tile_bag[self.tile_bag_index:]
```

```
method on string  
f-string formatting  
list comprehension  
built-in function  
collection  
slicing
```



```
tile_bag[self.tile_bag_index:]  
Counter()  
sorted()  
for a, b in  
f"{a}:{b}"  
' '.join
```

```
slicing  
collection  
built-in function  
list comprehension  
f-string formatting  
method on string
```



```
let tiles = tile_bag
```

```
.iter()
```



slicing

```
.skip(tile_bag_index)
```



Counter

```
.counts()
```

```
.iter()
```



sorted()

```
.sorted()
```



list

comprehension

```
.map(|(a, b)| format!("{a}:{b}"))
```

```
.collect::<Vec<_>>()
```



f-strings

```
.join(" ");
```



join

Hoogle Translate

counts



Clojure

frequencies

core

[Doc](#)



Racket

frequencies

list-utils

[Doc](#)



pandas

value_counts

Series

[Doc](#)



Python

Counter*

collections

[Doc](#)



Haskell

count

Data.List.Unique

[Doc](#)



Rust

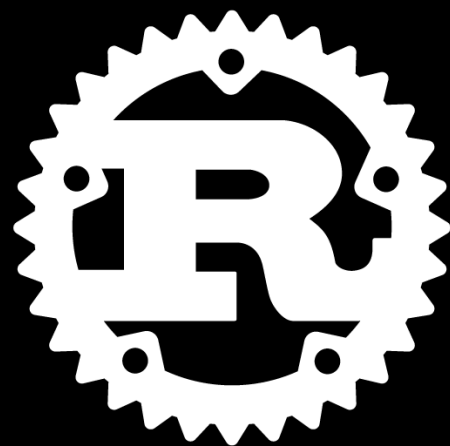
counts

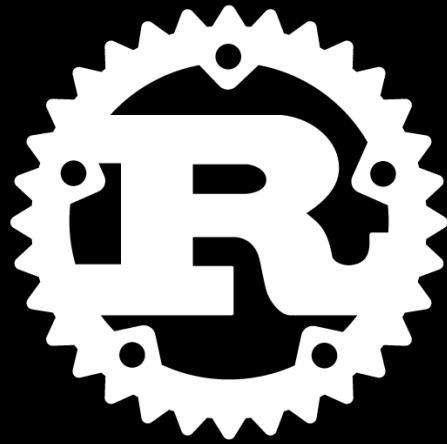
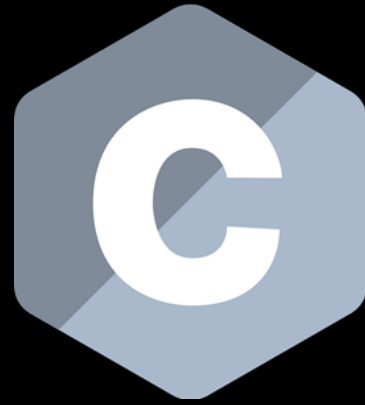
Itertools

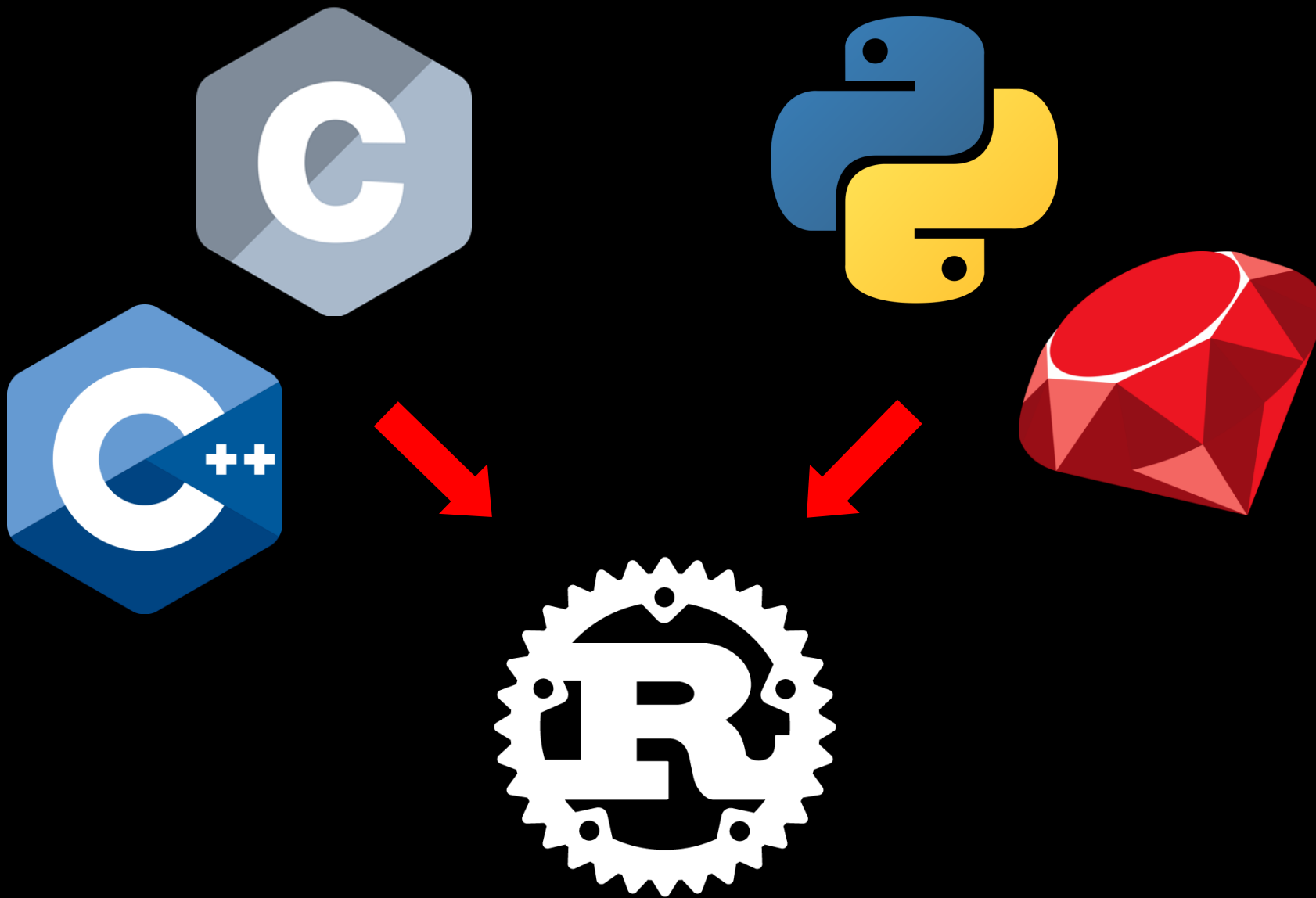
[Doc](#)



```
let tiles = tile_bag
    .iter()
    .skip(tile_bag_index)
    .counts()
    .iter()
    .sorted()
    .map(|(a, b)| format!("{a}:{b}"))
    .collect::<Vec<_>>()
    .join(" ");
```







[[end of digression]]

f-strings in  3.6

Refactor #3

Enum

Refactor #3a

Enum vs 2 Booleans

67d6faa



```
self.pause_for_analysis = False  
self.players_turn       = True
```



	<code>self.players_turn</code>	<code>self.pause_for_analysis</code>
PLAYERS_TURN	True	
COMPUTERS_TURN	False	



	<code>self.players_turn</code>	<code>self.pause_for_analysis</code>
PLAYERS_TURN	True	False
PAUSE_FOR_ANALYSIS	False	True
COMPUTERS_TURN	False	False



PLAYERS_TURN

PAUSE_FOR_ANALYSIS

COMPUTERS_TURN



```
class Phase(Enum):  
    PLAYERS_TURN          = 1  
    PAUSE_FOR_ANALYSIS   = 2  
    COMPUTERS_TURN        = 3
```



328	-	self.pause_for_analysis	= False	333	+	self.phase	= Phase.PLAYERS_TURN
329		self.pause_for_analysis_rank	= None	334		self.pause_for_analysis_rank	= None
330	-	self.players_turn	= True				



```
if (not self.players_turn and not self.pause_for_analysis):
```



```
if self.phase == Phase.COMPUTERS_TURN:
```




```
self.players_turn      = False  
self.pause_for_analysis = True
```



```
self.phase = Phase.PAUSE_FOR_ANALYSIS
```

Refactor #3b

Use Enums

acd6d66



```
# TODO convert to Enum
```

```
NO = 1
```

```
DL = 2
```

```
DW = 3
```

```
TL = 4
```

```
TW = 5
```



```
class Tile(Enum):  
    NO = 1  
    DL = 2  
    DW = 3  
    TL = 4  
    TW = 5
```



```
class Tile(Enum):
```

```
    NO = 1
```

```
    DL = 2
```

```
    DW = 3
```

```
    TL = 4
```

```
    TW = 5
```



```
class Tile(Enum):
```

```
    NO = 1
```

```
    DL = 2
```

```
    DW = 3
```

```
    TL = 4
```

```
    TW = 5
```

```
class Hooks(Enum):
```

```
    OFF = 0
```

```
    ALL = 1
```

```
    ON_RACK = 2
```

```
class Extension(Enum):
```

```
    PREFIX = 1
```

```
    SUFFIX = 2
```

```
class Phase(Enum):
```

```
    PLAYERS_TURN = 1
```

```
    PAUSE_FOR_ANALYSIS = 2
```

```
    COMPUTERS_TURN = 3
```

```
class Direction(IntEnum):
```

```
    ACROSS = 1
```

```
    DOWN = 2
```

Enum in 3.4

Refactor #1: @dataclass (3.7)

Refactor #2: f-strings (3.6)

Refactor #3: Enum (3.4)

Refactor #4

?



```
self.cursor = 0 # 0 = off, 1 = across, 2 = down
```



```
self.cursor = 0 # 0 = off, 1 = across, 2 = down
```



```
self.cursor = Optional.empty()
```



```
self.cursor = Optional.empty()
```

```
class Direction(IntEnum):
```

```
    ACROSS = 1
```

```
    DOWN   = 2
```

Refactor #4

Optional

9a378bf

[[digression]]



Option	Some	None
Maybe	Just	Nothing
optional	.value()	nullopt
Optional	some	none

12:15

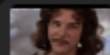
Category Theory for Programmers: Chapter 6 - Simple Algebraic Data Types

3.3K views • 1 year ago



code_report

PL Virtual Meetup: <https://www.meetup.com/Programming-Languages-Toronto-Meetup/> CTFP Textbook: ...



Introduction | Table of Contents | Product Types | Records | Duality | Algebraic Types | Exercise 1...

8 chapters ▾

CATEGORY THEORY FOR PROGRAMMERS



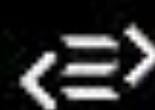
Bartosz Milewski



`Option`

`Some`

`None`



`Maybe`

`Just`

`Nothing`



`optional`

`.value()`

`nullopt`



`Optional`

`some`

`none`

12:15



Option

Some

None



Maybe

Just

Nothing



optional

.value()

nullopt



Optional

some

none



Option

Some

None



Maybe

Just

Nothing



optional

.value()

nullopt



Optional

some

none



Optional

.of()

.empty()

[[end of digression]]



Before

```
self.cursor
```

```
self.cursor == 0
```

```
self.cursor == 1
```

```
self.cursor == 2
```

```
self.cursor = 0
```

```
self.cursor = 1
```

```
self.cursor = 2
```

After

```
self.cursor.is_present()
```

```
self.cursor.is_empty()
```


```
self.cursor.get() == Direction.ACROSS
```

```
self.cursor.get() == Direction.DOWN
```

```
self.cursor = Optional.empty()
```

```
self.cursor = Optional.of(Direction.ACROSS)
```

```
self.cursor = Optional.of(Direction.DOWN)
```

Optional >=  2.7

<https://pypi.org/project/optional.py>

Refactor #5

Name the Algorithm

b7eb49b



```
rank = 1  
while play != self.player_plays[-rank]:  
    rank += 1
```

ITM

Initalize

Then

Modify



```
rank = 1  
while play != self.player_plays[-rank]:  
    rank += 1
```



```
int rank = 1;  
while (play != player_plays[player_plays.size() - rank]) {  
    rank += 1;  
}
```



```
auto const rank = std::distance(
    std::find(
        player_plays.rbegin(),
        player_plays.rend(),
        play),
    player_plays.rend());
```

<https://godbolt.org/z/4forbdGvM>



```
rank = 1  
while play != self.player_plays[-rank]:  
    rank += 1
```

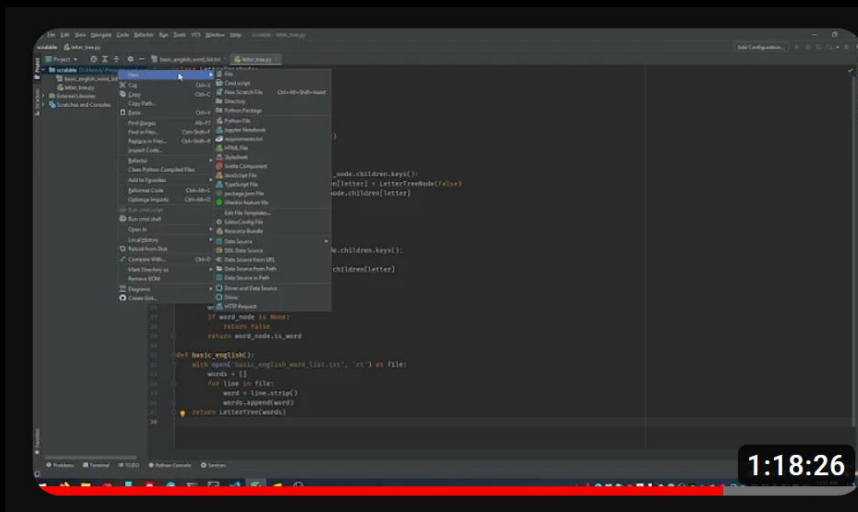


```
rank = self.player_plays[::-1].index(play) + 1
```


Refactor #6

Avoid ITM

17c2657



"The World's Fastest Scrabble Program" (1988) from the Ground Up

1.3K views • 1 year ago



boringcactus

original paper: <https://www.cs.cmu.edu/afs/cs/academic/class/15451-s06/www/lectures/scrabble.pdf> code: ...

Programming
Techniques and
Data Structures

Daniel Sleator
Editor

The World's Fastest Scrabble Program

ANDREW W. APPEL AND GUY J. JACOBSON

ABSTRACT: *An efficient backtracking algorithm makes possible a very fast program to play the SCRABBLE® Brand Crossword Game. The efficiency is achieved by creating data structures before the backtracking search begins that serve both to focus the search and to make each step of the search fast.*

1. INTRODUCTION

The SCRABBLE® Brand Crossword Game¹ (hereafter referred to as "Scrabble") is ill-suited to the adversary search techniques typically used by computer game-players. The elements of chance and limited information play a major role. This, together with the large number of moves available at each turn, makes sub-junctive reasoning of little value. In fact, an efficient generator of legal moves is in itself non-trivial to program, and would be useful as the tactical backbone of a computer crossword-game player.

The algorithm described here is merely that: a fast move generator. In practice, combining this algorithm

¹ As used in this paper, the mark SCRABBLE refers to one of the crossword-game products of Selchow and Richter Company.

This research was sponsored in part by a grant from the Amoco Foundation, in part by an NSF Graduate Student Fellowship, in part by NSF grant MCS-830805, in part by Presidential Young Investigator grant DCR-8352081, and in part by the Defense Advanced Research Projects Agency (DOD), ARPA Order No. 3597, monitored by the Air Force Avionics Laboratory Under Contract F33615-81-K-1539.

The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the US Government.

SCRABBLE is a registered trademark of Selchow and Richter Company for its line of wordgames and entertainment services.

© 1988 ACM 0001-0782/88/0500-0572 \$1.50

with a large dictionary and the heuristic of selecting the move with the highest score at each turn makes a very fast program that is rarely beaten by humans. The program makes no use of any strategic concepts, but its brute-force one-ply search is usually sufficient to overwhelm its opponent.

2. COMPUTER SCRABBLE-PLAYERS IN THE LITERATURE

A number of computer programs have been written to play SCRABBLE®. The best publicized is a commercial product named MONTY®², which is available for various microcomputers and as a hand-held device the size of a giant calculator. According to *Scrabble Players News*, it uses both strategic and tactical concepts [2]. The human SCRABBLE experts who reviewed MONTY beat it consistently, but said that it was a fairly challenging opponent.

Peter Turcan of the University of Reading in England has written a SCRABBLE-player [8, 9] for some unspecified micro-computer. It appears that he generates moves by iterating over the words in his lexicon in reverse order of length. He somehow decides for each word whether and where it can be played on the current board with the current rack. His program doesn't attempt any adversary search, but it does use an evaluation function more sophisticated than the score of the prospective move. It takes the score and conditionally adds terms depending on simple strategic features of the new position and tiles left in the rack.

² Registered trademark of Rital Corporation.



```
class Board:
    def __init__(self, size):
        self.size = size
        self._tiles = []
        for _ in range(size):
            row = []
            for _ in range(size):
                row.append('.')
            self._tiles.append(row)

    def all_positions(self):
        result = []
        for row in range(self.size):
            for col in range(self.size):
                result.append((row, col))
        return result

    def copy(self):
        result = Board(self.size)
        for pos in self.all_positions():
            result.set_tile(pos, self.get_tile(pos))
        return result

    # ...
```



```
class Board:
    def __init__(self, size):
        self.size = size
        self._tiles = []
        for _ in range(size):
            row = []
            for _ in range(size):
                row.append('.')
            self._tiles.append(row)
```



```
class Board:
    def __init__(self):
        self.size = 15
        self._tiles = []
        for _ in range(size):
            row = []
            for _ in range(size):
                row.append('.')
            self._tiles.append(row)
```



```
class Board:
    def __init__(self):
        self.size = 15
        self._tiles = []
        for _ in range(size):
            row = ['.'] * self.size
            self._tiles.append(row)
```



```
class Board:
    def __init__(self):
        self.size = 15
        self._tiles =
            [['.'] * self.size for _ in range(self.size)]
```




```
class Board:
    def __init__(self):
        self.size = 15
        self._tiles =
            [['.'] * self.size for _ in range(self.size)]

    def all_positions(self):
        result = []
        for row in range(self.size):
            for col in range(self.size):
                result.append((row, col))
        return result
```



```
import itertools as it

class Board:
    def __init__(self):
        self.size = 15
        self._tiles =
            [['.'] * self.size for _ in range(self.size)]

    def all_positions(self):
        return it.product(range(0, 15), range(0,15))
```

Hoogle Translate

product



Rust

`cartesian_product`

`trait.itertools`

[Doc](#)



D

`cartesianProduct`

`std.algorithm.setops`

[Doc](#)



Racket

`cartesian-product`

`base`

[Doc](#)



C++

`cartesian_product`

`range-v3`

[Doc](#)



Julia

`product`

`IterTools`

[Doc](#)



Python

`product`

`itertools`

[Doc](#)



Ruby

`product`

`Array`

[Doc](#)



APL

`,°. ,`

[Doc](#)



q

`cross`

`-`

[Doc](#)



F#

`allPairs`

`List`

[Doc](#)



```
import itertools as it

class Board:
    def __init__(self):
        self.size = 15
        self._tiles =
            [['.'] * self.size for _ in range(self.size)]

    def all_positions(self):
        return it.product(range(0, 15), range(0,15))
```



```
import itertools as it

class Board:
    def __init__(self):
        self.size = 15
        self._tiles =
            [['.'] * self.size for _ in range(self.size)]

    def all_positions(self):
        return it.product(range(0, 15), range(0,15))

    def copy(self):
        result = Board(self.size)
        for pos in self.all_positions():
            result.set_tile(pos, self.get_tile(pos))
        return result
```



```
import copy
import itertools as it

class Board:
    def __init__(self):
        self.size = 15
        self._tiles =
            [['.'] * self.size for _ in range(self.size)]

    def all_positions(self):
        return it.product(range(0, 15), range(0,15))

    def copy(self):
        return copy.deepcopy(self)
```



```
class Board:
    def __init__(self, size):
        self.size = size
        self._tiles = []
        for _ in range(size):
            row = []
            for _ in range(size):
                row.append('.')
            self._tiles.append(row)

    def all_positions(self):
        result = []
        for row in range(self.size):
            for col in range(self.size):
                result.append((row, col))
        return result

    def copy(self):
        result = Board(self.size)
        for pos in self.all_positions():
            result.set_tile(pos, self.get_tile(pos))
        return result

    # ...
```



```
import copy
import itertools as it

class Board:
    def __init__(self):
        self.size = 15
        self._tiles =
            [['.'] * self.size for _ in range(self.size)]

    def all_positions(self):
        return it.product(range(0, 15), range(0,15))

    def copy(self):
        return copy.deepcopy(self)
```


Refactor #1: @dataclass (3.7)

Refactor #2: f-strings (3.6)

Refactor #3: Enum (3.4)

Refactor #4: Optional

Refactor #5: Use Algorithms

Refactor #6: Avoid ITM


<https://github.com/codereport/scrabble/commits>




Thank You

<https://github.com/codereport/Content/Talks>

Conor Hoekstra

 code_report


 codereport




Questions?

<https://github.com/codereport/Content/Talks>

Conor Hoekstra

 code_report

 codereport