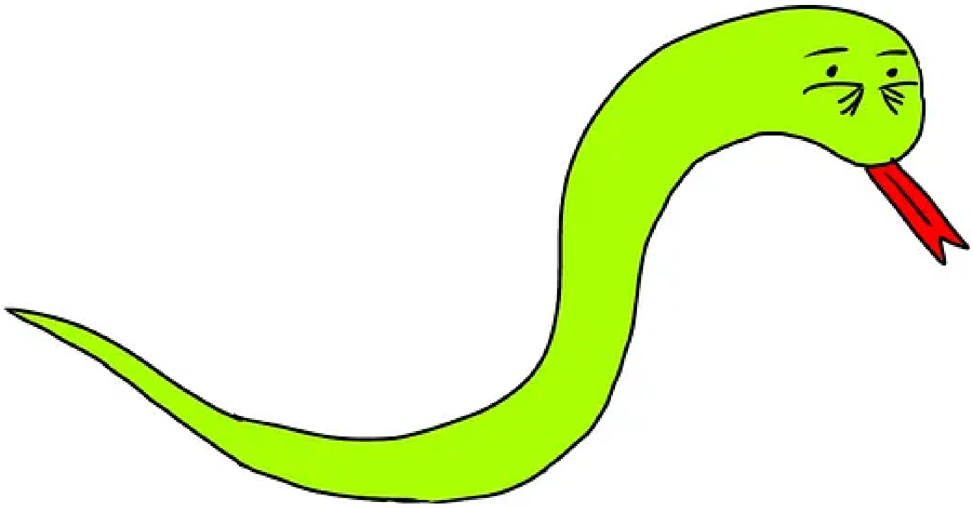
12 Difficult Python Questions You Might Take Days To Solve

Liu Zuo Lin · Follow

Published in Level Up Coding

6 min read · Apr 28

Listen Share More

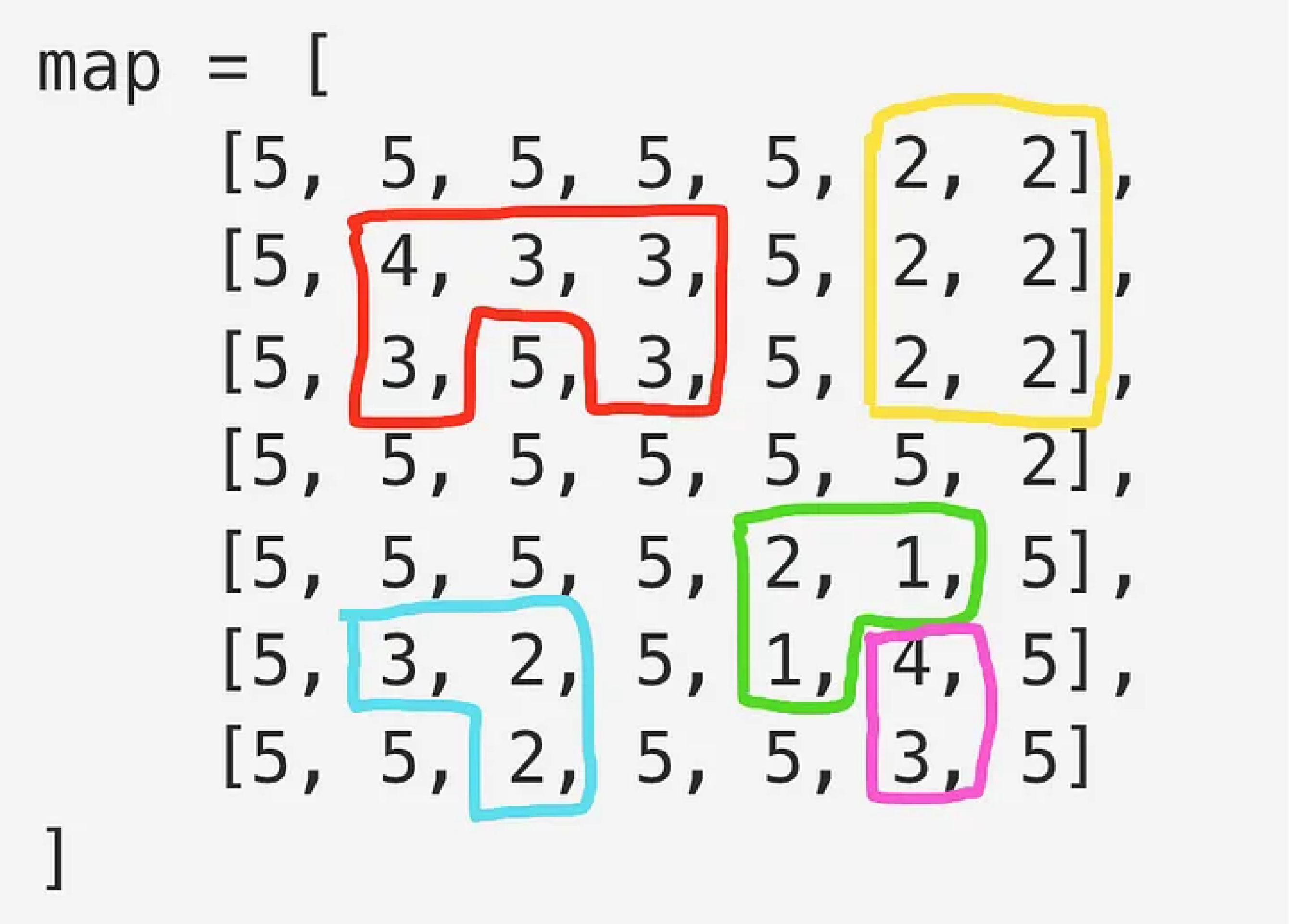


1) Largest Puddle

You are given a 2D list of integers.

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

Each number represents the height of the land. When it rains, water flows from larger numbers to smaller numbers (horizontally/vertically). Puddles collect when water cannot flow out of the map. Assume that water that flows outside of the map will *not* form a puddle.



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

Open in app ↗

- The blue area is NOT a puddle, as water can flow out
- The green area is a puddle as water cannot flow out
- The yellow area is NOT a puddle as water can flow out
- The pink area is also NOT a puddle as water can flow out

Our task here would be to write a function `largest_puddle(map)` that takes in a 2D list `map` , and returns the coordinates of the largest puddle on the map. For the above map:

```
{(1,1), (2,1), (1,2), (1,3), (2,3)}
```

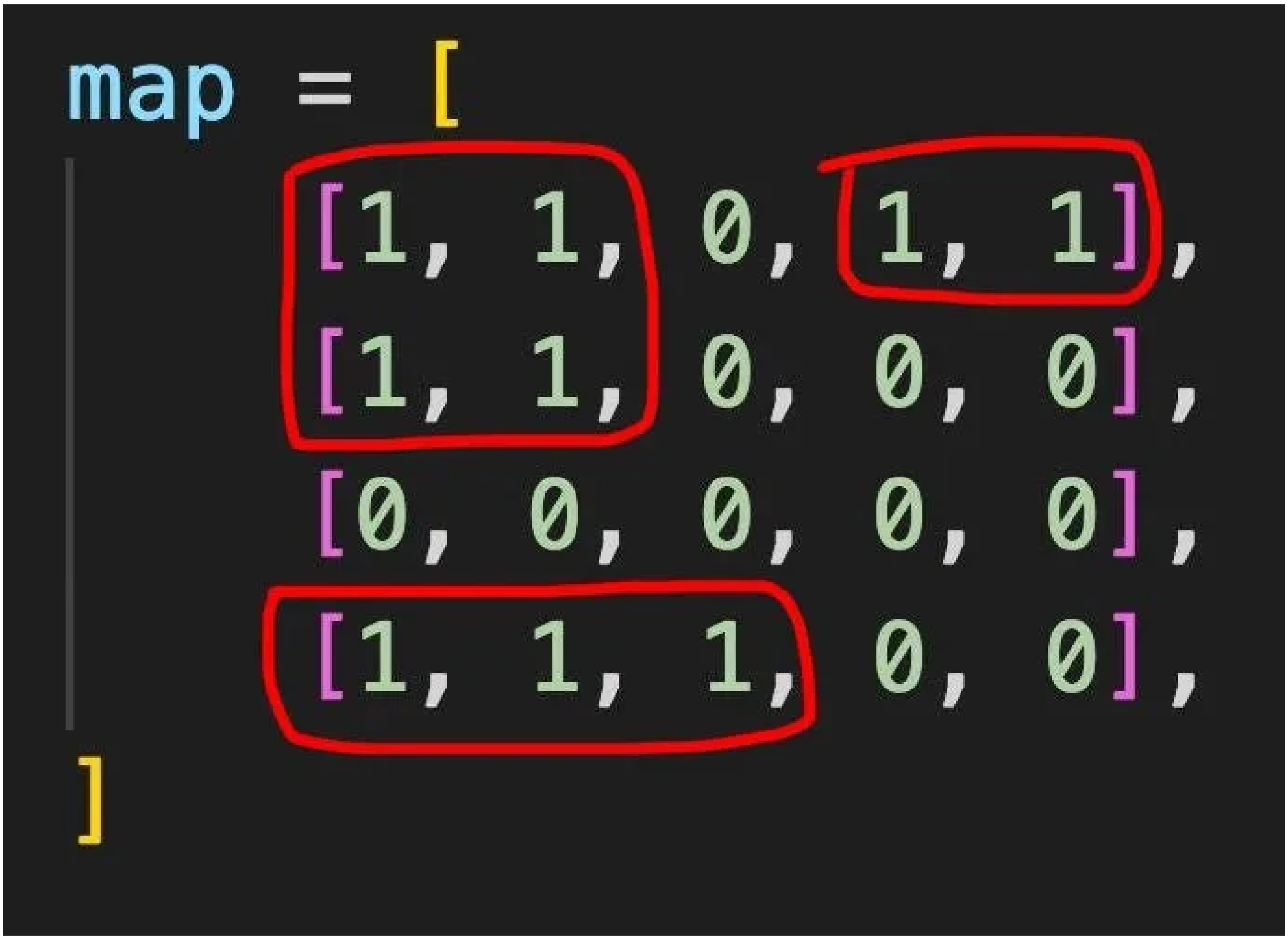
^ the coordinates of the red puddle

2) Finding Island Coordinates

You are given a 2D list of integers (either 0 or 1)

```
map = [
    [1,1,0,1,1],
    [1,1,0,0,0],
    [0,0,0,0,0],
    [1,1,1,0,0],
]
```

1’s are land, while 0’s are sea. 1’s that are next to each other (horizontal + vertical only) make up an island. Here, we have 3 islands:



```
map = [
    [1, 1, 0, 1, 1],
    [1, 1, 0, 0, 0],
    [0, 0, 0, 0, 0],
    [1, 1, 1, 0, 0],
]
```

Write a function `get_islands(map)` that takes in a 2D list `map`, and returns a list of sets, each set representing 1 island.

```
[
    {(0, 1), (1, 0), (0, 0), (1, 1)},  # the top-left island
    {(0, 3), (0, 4)},                  # the top-right island
    {(3, 0), (3, 2), (3, 1)}           # the bottom-right island
]
```

3) Longest Word Chain

You are given a list of English words:

```
words = ["apple", "orange", "tank", "elephant", "kitten"]
```

A word chain is a list of English words where the last letter of each word is equal to the first letter of the next word. For instance:

- `["apple", "elephant"]` is a valid word chain
- `["tank", "kitten"]` is a valid word chain
- `["elephant", "apple"]` is not a valid word chain as `"apple"` does not begin with `"t"`
- `["apple", "kitten"]` is not a valid word chain as `"kitten"` does not begin with `"e"`

Write a function `longest_word_chain(words)` that takes in a list of English words `words`, and returns the longest possible valid word chain. For the example above, the longest possible valid word chain is:

```
["apple", "elephant", "tank", "kitten"]
```

Or:

```
["orange", "elephant", "tank", "kitten"]
```

4) Hollow Diamond

Write a function `hollow_diamond(string)` that takes in a string `string`, and prints the following pattern:

```
hollow_diamond('abcdefgh')

  a
 b h
c   g
 d f
  e
```

If there are insufficient characters to form a perfect hollow diamond shape, append your string with `*` characters.

```
hollow_diamond('abcdefghij')

   a
  b *
 c   *
d     j
 e   i
  f h
   g
```

Another example:

```
hollow_diamond('abcdefghijklmnop')

    a
   b p
  c   o
 d     n
e       m
 f     l
  g   k
   h j
    i
```

Yet another example:

```
hollow_diamond('abcdefghijklmn')

    a
   b *
  c   *
 d     n
e       m
 f     l
  g   k
```

```
      h j
       i
```

5) Upslope Coordinates

```
lis = [1, 3, 4, 6, 2, 3, 5, 3, 3, 8, 9]
```

You are given a list of numbers representing land height. Write a function `upslope(lis)` that takes in this list of numbers, and returns a 2D list containing sections that are upslope (increasing).

```
x = upslope(lis)

# [[1, 3, 4, 6], [2, 3, 5], [3, 8, 9]]
```

6) Contains 1 (No Strings Allowed)

Write a function `contains1(n)` that takes in an integer `n`, and returns True if n contains the digit 1, and False otherwise. You cannot use any strings or string methods. (if you could, it's way too easy)

```
contains1(21)    # True
contains1(201)   # True
contains1(617)   # True

contains1(22)    # False
contains1(202)   # False
contains1(627)   # False
```

7) Minimum number of coins

Write a function `min_coins(coins, value)` that takes in a list `coins` and an integer `value`, and returns a dictionary representing the *minimum* total number of coins we need to make up `value`. For instance:

```
min_coins([1, 2, 5], 101)
```

- We have an infinite number of $1 coins, $2 coins and $5 coins
- We need to find the *minimum* number of coins to make $101
- In this case, the best possible answer is 20 $5 coins and 1 $1 coin
- The function hence returns `{1:1, 5:20}`

```
min_coins([2, 3], 20)
```

- We have an infinite number of $2 and $3 coins
- We need the minimum number of coins to make up $20
- The answer is 6 $3 coins and 1 $2 coin.
- The function hence returns `{2:1, 3:6}`

```
min_coins([2, 4, 6], 5)
```

- We have an infinite number of $2, $4 and $6 coins to make $5
- This is not possible, so we simply return `{}`

8) Dictionary Parsing

```
string = '{"apple":4, "orange":5, "pear":6}'
```

You are given a string representing a Python dictionary. Write a function `parse(string)` that takes in this string, parses it, and returns the actual dictionary.

Assume that keys and values will either be numbers or string, and that there are no nested lists/dicts etc.

Note — You cannot use libraries or the `eval` or `exec` function

```
parse(string)

# {"apple":4, "orange":5, "pear":6}
```

9) Magic Square

A magic square is a 3x3 grid containing numbers 1 to 9 (each number should appear ONCE). Every 3 consecutive numbers (row, column or diagonal) must add up to 15. An example:

```
[  [2, 7, 6],
   [9, 5, 1],
   [4, 3, 8]  ]
```

You are given an *incomplete* magic square.

```
magic_square = [
  [2, 0, 0],
  [0, 0, 0],
  [0, 3, 8]
]
```

Here, 0 means you need to fill it in. Write a function `solve(magic_square)` that takes in the incomplete magic square, fills it in with the correct numbers, and returns the complete magic square.

```
def solve(magic_square):
    # stuff

solve(magic_square)
```

```
# [  [2, 7, 6],
#    [9, 5, 1],
#    [4, 3, 8]  ]
```

10) Square root

Write a function `sqrt(n)` that takes in an integer `n`, and returns its square root. You are not allowed to use any built-in operations or functions to automatically find the square root.

Hint — build the answer using a string

```
def sqrt(n):
    # stuff

sqrt(0) # 0
sqrt(1) # 1.0
sqrt(2) # 1.414213562
sqrt(3) # 1.732050808
sqrt(4) # 2.0
sqrt(5) # 2.236067977
```

11) Letter pyramid

Write a function `pyramid(string)` that takes in a string `string`, and prints the following pattern.

```
pyramid('abcdef')

a
bc
def
```

If there are insufficient letters, add `*` characters to form a perfect triangle

```
pyramid('abcdefg')

a
bc
```

```
def
g***
```

```
pyramid('abcdefgh')

a
bc
def
gh**
```

```
pyramid('abcdefghijk')

a
bc
def
ghij
k****
```

12) Evaluating Math Expressions

You are given a string representing a math expression. For instance:

```
string = '1+2x3/4-5'
```

Without using built-in functions like `eval` or `exec`, write a function `evaluate(string)` that takes in a math expression `string`, solves it, and returns the result.

- Assume that only addition, subtraction, multiplication and division operators will exist in the string
- PEDMAS rule applies — multiplication/division before addition/subtraction

```
def evaluate(string):
    # stuff

evaluate('1+1')       # 2
evaluate('1+2x3')     # 7
evaluate('1-2x3+4x5') # 15
```

Conclusion

Let me know how long you took to solve them all!

Some Final words

If this story provided value and you wish to show a little support, you could:

1. *Clap 50 times for this story (this really, really helps me out)*
2. *Sign up for a Medium membership using my link ($5/month to read unlimited Medium stories)*

Get my free Ebooks: https://zlliu.co/books

Get an email whenever Liu Zuo Lin publishes.

Get an email whenever Liu Zuo Lin publishes. By signing up, you will create a Medium account if you don't already have...

zlliu.medium.com

Level Up Coding

Thanks for being a part of our community! Before you go:

- Clap for the story and follow the author 👉
- View more content in the Level Up Coding publication
- 💰 Free coding interview course ⇒ View Course
- Follow us: Twitter | LinkedIn | Newsletter

👉 **Join the Level Up talent collective and find an amazing job**

Python Python Programming

Follow

Written by Liu Zuo Lin

1.97K Followers · Writer for Level Up Coding

Software Engineer, Python Tutor, Tech Writer. My Free Ebooks — 1) Python Zero To One 2) 40 Python Practice Questions For Beginners — https://zlliu.co/books

More from Liu Zuo Lin and Level Up Coding

Liu Zuo Lin in Level Up Coding

20 Python Concepts I Wish I Knew Way Earlier

# Stuff I wish I learnt earlier as a beginner

✦ · 9 min read · Apr 16

3K 12

Sanjay Priyadarshi in Level Up Coding

I Spent 14 Days Studying A Programmer Who Built a $167 B Company — Here Are His Weird Rules To...

Steal This Programmer Blueprint

✦ · 11 min read · Apr 23

1.1K 16

Alexander Nguyen in Level Up Coding

Why I Keep Failing Candidates During Google Interviews…

They don’t meet the bar.

✦ · 4 min read · Apr 13

3.1K 98

Liu Zuo Lin

36 Things I Didn’t Know About Python Until Recently (Compilation)

# Despite learning Python since 2017

✦ · 9 min read · Apr 7

460 12

See all from Liu Zuo Lin

See all from Level Up Coding

Recommended from Medium

Patrick Kalkman in ITNEXT

Dependency Injection in Python

Building flexible and testable architectures in Python

✦ · 13 min read · Apr 14

542 5

Jan Kammerath

Boomer Developers: 10 Lessons I Learned From Them

Boomers taught me valuable lessons about software engineering which I want to share with you. Especially if you're a younger generation.

✦ · 17 min read · Apr 23

30

Kasper Müller in Cantor's Paradise

The 7 Most Important Mathematical Constants

The beauty of π, the magic of φ, the mystery of γ, and the powers of e

✦ · 13 min read · Apr 27

22

Patrick Kalkman in ITNEXT

Mastering Asyncio — Unleashing the Power of Async/Await in Python

Discover the secrets of asynchronous programming and boost your Python projects

✦ · 12 min read · Apr 29

2

Mike Huls in Towards Data Science

What is the difference between UNION and JOIN in SQL?

5 minute guide to UNION, EXCEPT and INTERSECT in SQL

✦ · 7 min read · 5 days ago

Alexander Nguyen in Level Up Coding

Why I Keep Failing Candidates During Google Interviews…

They don’t meet the bar.

✦ · 4 min read · Apr 13

3.1K 98

See more recommendations