

# Hexaware Coding Challenge Plan Day - 9

## Problem - 1 Functions

**HackerRank**

Prepare > Python > Python Functionals > Map and Lambda Function

Exit Full Screen View

Problem

Submissions

Leaderboard

Discussions

Let's learn some new Python concepts! You have to generate a list of the first  $N$  fibonacci numbers, 0 being the first number. Then, apply the map function and a lambda expression to cube each fibonacci number and print the list.

**Concept**

The `map()` function applies a function to every member of an iterable and returns the result. It takes two parameters: first, the function that is to be applied and secondly, the iterables.

Let's say you are given a list of names, and you have to print a list that contains the length of each name.

```
>> print(list(map(len, ['Tina', 'Raj', 'Tom'])))
[4, 3, 3]
```

Lambda is a single expression anonymous function often used as an inline function. In simple words, it is a function that has only one line in its body. It proves very handy in functional and GUI programming.

```
>> sum = lambda a, b, c: a + b + c
>> sum(1, 2, 3)
6
```

**Note:**

Lambda functions cannot use the return statement and can only have a single

You have earned 20.00 points!  
42/115 challenges solved. 37%

**Congratulations**  
You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#) [Next Challenge](#)

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Compiler Message

Success

Input (stdin)  
1 5  
[Download](#)

Expected Output  
1 [0, 1, 1, 8, 27]  
[Download](#)

## Problem - 2 Functions

**HackerRank**

Prepare > Python > Python Functionals > Validating Email Addresses With a Filter

Exit Full Screen View

Submissions

Leaderboard

Discussions

Editorial

Complete the function `filter_mail` in the editor below.

`filter_mail` has the following parameters:

- string `s`: the string to test

**Returns**

- boolean: whether the string is a valid email or not

**Input Format**

The first line of input is the integer  $N$ , the number of email addresses.  $N$  lines follow, each containing a string.

**Constraints**

Each line is a non-empty string.

**Sample Input**

```
3
lara@hackerrank.com
brian-23@hackerrank.com
britts_54@hackerrank.com
```

**Sample Output**

```
['brian-23@hackerrank.com', 'britts_54@hackerrank.com', 'lara@hackerrank.com']
```

You have earned 20.00 points!  
43/115 challenges solved. 37%

**Congratulations**  
You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#) [Next Challenge](#)

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message

Success

Input (stdin)  
1 3  
2 lara@hackerrank.com  
3 brian-23@hackerrank.com  
4 britts\_54@hackerrank.com  
[Download](#)

Expected Output  
1 ['brian-23@hackerrank.com', 'britts\_54@hackerrank.com', 'lara@hackerrank.com']  
[Download](#)

## Problem - 3 Functions

**HackerRank** Prepare > Python > Python Functionals > Reduce Function

Exit Full Screen View

Problem

Given a list of rational numbers, find their product.

**Concept**  
The `reduce()` function applies a function of two arguments cumulatively on a list of objects in succession from left to right to reduce it to one value. Say you have a list, say `[1,2,3]` and you have to find its sum.

```
>>> reduce(lambda x, y : x + y, [1,2,3])
6
```

You can also define an initial value. If it is specified, the function will assume initial value as the value given, and then reduce. It is equivalent to adding the initial value at the beginning of the list. For example:

```
>>> reduce(lambda x, y : x + y, [1,2,3], -3)
3
>>> from fractions import gcd
>>> reduce(gcd, [2,4,8], 3)
1
```

**Input Format**  
First line contains  $n$ , the number of rational numbers.  
The  $i^{\text{th}}$  of next  $n$  lines contain two integers each, the numerator ( $N_i$ ) and denominator ( $D_i$ ) of the  $i^{\text{th}}$  rational number in the list.

Submissions

Leaderboard

Discussions

**Congratulations**  
You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#) [Next Challenge](#)

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Test case 6

Compiler Message  
Success

**Hidden Test Case**  
Unlock this test case for 5 hacks.  
**Unlock**

## Problem - 4 Collections

**HackerRank** Prepare > Python > Collections > Collections.deque()

Exit Full Screen View

Submissions

Leaderboard

Discussions

Editorial

You have earned 20.00 points!  
45/115 challenges solved. 39%

**Congratulations**  
You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#) [Next Challenge](#)

Test case 0

Test case 1

Test case 2

Test case 3

Test case 4

Test case 5

Compiler Message  
Success

Input (stdin) [Download](#)  

```
1 6
2 append 1
3 append 2
4 append 3
5 appendLeft 4
6 pop
7 popLeft
```


## Problem - 5 Collections

Submissions

Leaderboard

Discussions

Editorial

**HackerRank** 

Prepare > Python > Collections > Piling Up!

`1 >= sideLength < 4`

**Output Format**  
For each test case, output a single line containing either Yes or No.


**Sample Input**

STDIN	Function
2	T = 2
6	blocks[] size n = 6
4 3 2 1 3 4	blocks = [4, 3, 2, 1, 3, 4]
3	blocks[] size n = 3
1 3 2	blocks = [1, 3, 2]

**Sample Output**




Yes
No

**Explanation**  
In the first test case, pick in this order: **left** - 4, **right** - 4, **left** - 3, **right** - 3, **left** - 2, **right** - 1.  
In the second test case, no order gives an appropriate arrangement of vertical cubes.  
3 will always come after either 1 or 2.

 You have earned 50.00 points!  
46/115 challenges solved.

40%

**Congratulations**

You solved this challenge. Would you like to challenge your friends?   

**Next Challenge**

Test case 0

Success

Test case 1

🔒

Test case 2

🔒

Test case 3

🔒

Test case 4

🔒

Input (stdin)

Download

1	2
2	6
3	4 3 2 1 3 4
4	3
5	1 3 2

Expected Output

Download

1	Yes
2	No