1. **Drunken Python**

language\_fundamentalsnumbersstrings

Python got *drunk* and the built-in functions str() and int() are acting odd:

str(4) ➞ 4

str("4") ➞ 4

int("4") ➞ "4"

int(4) ➞ "4"

You need to create two functions to substitute str() and int(). A function called int\_to\_str() that converts **integers into strings** and a function called str\_to\_int() that converts **strings into integers**.

Solutions:

**str, int = int, str**

**def int\_to\_str(n):**

**return str(n)**

**def str\_to\_int(s):**

**return int(s)**

**Examples:**

int\_to\_str(4) ➞ "4"

str\_to\_int("4") ➞ 4

int\_to\_str(29348) ➞ "29348"

1. **Length of Number**

]numbers

Create a function that takes a number num and returns its length.

**Examples**

number\_length(10) ➞ 2

number\_length(5000) ➞ 4

Solutions :

**def number\_length(num):**

**count = 0**

**if num == 0:**

**return 1**

**else:**

**while num != 0:**

**num//=10**

**count += 1**

**return count**

## FizzBuzz Interview Question

interviewlanguage\_fundamentalslogicmathsorting

Create a function that takes a number as an argument and returns "Fizz", "Buzz" or "FizzBuzz".

* If the number is a multiple of 3 the output should be "Fizz".
* If the number given is a multiple of 5, the output should be "Buzz".
* If the number given is a multiple of both 3 and 5, the output should be "FizzBuzz".
* If the number is not a multiple of either 3 or 5, the number should be output on its own as shown in the examples below.
* The output should always be a string even if it is not a multiple of 3 or 5.

### Examples

fizz\_buzz(3) ➞ "Fizz"

fizz\_buzz(5) ➞ "Buzz"

fizz\_buzz(15) ➞ "FizzBuzz"

fizz\_buzz(4) ➞ "4"

Solutions :

**def fizz\_buzz(num):**

**return "Fizz"\*(num%3==0) + "Buzz"\*(num%5==0) or str(num)**

1. **Designing Rugs**

arrayslanguage\_fundamentalsloops

Write a function that accepts the width and height (m, n) and an optional **proc** s and generates a list with m elements. Each element is a string consisting of either:

* The default character (hash #) repeating n times (if no **proc** is given).
* The character passed in through the proc repeating n times.

**Examples**

make\_rug(3, 5) ➞ [

"#####",

"#####",

"#####"

]

make\_rug(3, 5, '$') ➞ [

"$$$$$",

"$$$$$",

"$$$$$"

]

make\_rug(2, 2, 'A') ➞ [

"AA"

"AA"

]

Solutions :

***def make\_rug(m, n, s='#'):***

***return [s\*n]\*m***

## Sum of Missing Numbers

arraysmathnumbers

Create a function that returns the sum of missing numbers from the given list.

### Examples

sum\_missing\_numbers([4, 3, 8, 1, 2]) ➞ 18

# 5 + 6 + 7 = 18

sum\_missing\_numbers([17, 16, 15, 10, 11, 12]) ➞ 27

# 13 + 14 = 27

sum\_missing\_numbers([1, 2, 3, 4, 5]) ➞ 0

# No Missing Numbers (i.e. all numbers in [1, 5] are present in the list)

Answer :

**def sum\_missing\_numbers(lst):**

**return sum(range(min(lst), max(lst) + 1)) - sum(lst)**