

# Python [solutions]

Python and R for Data Science

Data Science and Management



# Exercise 1: find how many equal numbers

Define a function `count_equals` that:

- takes as arguments four numbers
- returns:
  - the maximum number of equal numbers between the four

Example:

- `count_equals(1,2,3,4)` should return `0`
- `count_equals(1,2,5,4)` should return `0`
- `count_equals(1,2,2,2)` should return `3` because there are three `2` in the sequence
- `count_equals(1,1,1,2)` should return `3` because there are three `1` in the sequence

In [6]: *# Solution goes here*

## Test your code

Run this code to test your solution:

```
In [7]: try: assert count_equals(1,2,3,4) == 0 and count_equals(1,5,3,4) == 0 a  
except: print('Test failed')
```

Test failed

## Solution

```
In [2]: def count_equals(a, b, c, d):  
        numbers = [a, b, c, d]  
        max_count = 0  
  
        for num in numbers:  
            count = 0  
            for other in numbers:  
                if num == other:  
                    count += 1  
            if count > max_count:  
                max_count = count  
  
        return max_count if max_count > 1 else 0  
  
# test  
assert count_equals(1,2,3,4) == 0 and count_equals(1,5,3,4) == 0 and cc
```

Test passed

## Exercise 2: Fibonacci's sequence

Define a function `fibonacci` that:

- takes as arguments:
  - an integer number `n`
- returns:
  - a list containing the first `n` numbers of the Fibonacci's sequence

Note: The Fibonacci sequence is a series of numbers where each number is the sum of the two previous ones, starting with 0 and 1. To calculate it, you begin with 0 and 1, then add these to get the next number. Continue this process to generate the sequence. It goes 0, 1, 1, 2, 3, 5, 8, and so on.

In [6]: *# Solution goes here*

## Test your code

Run this code to test your solution:

```
In [7]: try: assert fibonacci(1) == [0] and fibonacci(3) == [0,1,1] and fibonacci(5) == [0,1,1,2,3]
except: print('Test failed')
```

Test failed

## Solution

```
In [13]: def fibonacci(n):  
         fib_sequence = [0, 1]  
  
         if n == 1:  
             return [0]  
  
         for i in range(2, n):  
             next_fib = fib_sequence[-1] + fib_sequence[-2]  
             fib_sequence.append(next_fib)  
  
         return fib_sequence[:n]  
  
         # test  
         assert fibonacci(1) == [0] and fibonacci(3) == [0,1,1] and fibonacci(7)
```

Test passed

## Exercise 3: zero-sum triplets

Define a function `zero_sum_triplets` that:

- takes as arguments:
  - a list of integers `numbers`
- returns:
  - the number of triplets whose sum is zero

Example:

- `zero_sum_triplets([1, -1, 0, 7, 12])` should return `1` because the sum of 1, -1, 0 is `0`
- `zero_sum_triplets([1, 9, 0, 7, 12])` should return `0` because there are no triplets that sum up to zero
- `zero_sum_triplets([1, -9, 8, 6, -14])` should return `2` because the sum of 1, -1, 0 is `0` and the sum of 8, 6, -14 is `0`

In [8]: *# Solution goes here*



## Test your code

Run this code to test your solution:

```
In [9]: try: assert zero_sum_triplets([1,-1,0,7,12]) == 1 and zero_sum_triplets
except: print('Test failed')
```

Test passed

## Solution

```
In [10]: def zero_sum_triplets(numbers):  
    n = len(numbers)  
    count = 0  
  
    for i in range(n - 2):  
        for j in range(i + 1, n - 1):  
            for k in range(j + 1, n):  
                if numbers[i] + numbers[j] + numbers[k] == 0:  
                    count += 1  
  
    return count  
  
# test  
assert zero_sum_triplets([1, -1, 0, 7, 12]) == 1 and zero_sum_triplets([1, 9
```

Test passed

## Exercise 4: Collatz

Define a function `collatz` that:

- takes as argument an integer number `n`
- returns:
  - a list containing all the numbers generated by the Collatz conjecture (stopping when reaching `1`)

Note: The Collatz Conjecture is a mathematical problem that starts with any positive integer. The process involves two steps: if the number is even, divide it by 2; if it's odd, multiply it by 3 and add 1. Repeat this process with the resulting number. The conjecture suggests that, no matter what number you start with, you'll eventually reach the number 1.

In [12]: `# Solution goes here`

## Test your code

Run this code to test your solution:

```
In [13]: try: assert collatz(12) == [6,3,10,5,16,8,4,2,1] and collatz(1) == [] a  
except: print('Test failed')
```

Test failed

## Solution

```
In [12]: def collatz(n):  
        sequence = []  
  
        while n != 1:  
            if n % 2 == 0:  
                n = n // 2  
            else:  
                n = 3 * n + 1  
            sequence.append(n)  
  
        return sequence  
  
        # test  
        assert collatz(12) == [6,3,10,5,16,8,4,2,1] and collatz(1) == [] and co
```

Test passed

## Exercise 5: Greatest Common Divisor (GCD)

Define a function `gcd` that:

- takes as argument two integer numbers `a` and `b`
- returns:
  - the gcd between `a` and `b`

In [15]: *# Solution goes here*

## Test your code

Run this code to test your solution:

```
In [16]: try: assert gcd(1,2) == 1 and gcd(7,2) == 1 and gcd(4,2) == 2 and gcd(1,1) == 1  
except: print('Test failed')
```

Test failed

## Solution

```
In [18]: def gcd(x, y):  
          while y != 0:  
              (x, y) = (y, x % y)  
          return x  
  
          # test  
          assert gcd(1,2) == 1 and gcd(7,2) == 1 and gcd(4,2) == 2 and gcd(15,25)
```

Test passed



