



# Interview Questions

## Whiteboarding Code

# During the interview

- Make sure you understand exactly what problem you're trying to solve. Ask the interviewer questions if anything is unclear or underspecified.
- Make sure you explain your plan to the interviewer before you start writing any code, so that they can help you avoid spending time going down less-than-ideal paths.
- If you can't think of a good way to do something, it often helps to start by talking through a dumb way to do it.
- Mention what invalid inputs you'd want to check for (e.g. input variable type check). Don't bother writing the code to do so unless the interviewer asks. In all my interviews, nobody has ever asked.
- Before declaring that your code is finished, think about variable initialization, end conditions, and boundary cases (e.g. empty inputs). If it seems helpful, run through an example. You'll score points by catching your bugs yourself, rather than having the interviewer point them out.

# During the interview

- Make sure you understand exactly what problem you're trying to solve. Ask the interviewer questions if anything is unclear or underspecified.
- Make sure you explain your plan to the interviewer before you start writing any code, so that they can help you avoid spending time going down less-than-ideal paths.
- If you can't think of a good way to do something, it often helps to start by talking through a dumb way to do it.
- Mention what invalid inputs you'd want to check for (e.g. input variable type check). Don't bother writing the code to do so unless the interviewer asks. In all my interviews, nobody has ever asked.
- Before declaring that your code is finished, think about variable initialization, end conditions, and boundary cases (e.g. empty inputs). If it seems helpful, run through an example. You'll score points by catching your bugs yourself, rather than having the interviewer point them out.

# During the interview

- Make sure you understand exactly what problem you're trying to solve. Ask the interviewer questions if anything is unclear or underspecified.
- Make sure you explain your plan to the interviewer before you start writing any code, so that they can help you avoid spending time going down less-than-ideal paths.
- If you can't think of a good way to do something, it often helps to start by talking through a dumb way to do it.
- Mention what invalid inputs you'd want to check for (e.g. input variable type check). Don't bother writing the code to do so unless the interviewer asks. In all my interviews, nobody has ever asked.
- Before declaring that your code is finished, think about variable initialization, end conditions, and boundary cases (e.g. empty inputs). If it seems helpful, run through an example. You'll score points by catching your bugs yourself, rather than having the interviewer point them out.

# During the interview

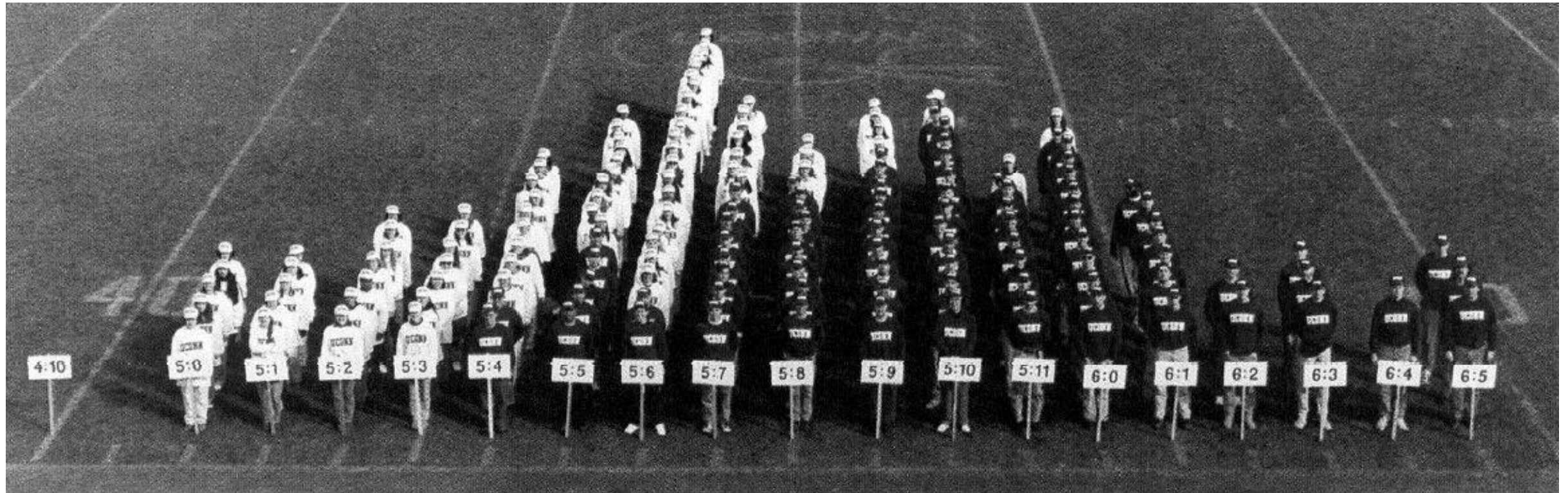
- Make sure you understand exactly what problem you're trying to solve. Ask the interviewer questions if anything is unclear or underspecified.
- Make sure you explain your plan to the interviewer before you start writing any code, so that they can help you avoid spending time going down less-than-ideal paths.
- If you can't think of a good way to do something, it often helps to start by talking through a dumb way to do it.
- **Mention what invalid inputs you'd want to check for (e.g. input variable type check). Don't bother writing the code to do so unless the interviewer asks. In all my interviews, nobody has ever asked.**
- Before declaring that your code is finished, think about variable initialization, end conditions, and boundary cases (e.g. empty inputs). If it seems helpful, run through an example. You'll score points by catching your bugs yourself, rather than having the interviewer point them out.

# During the interview

- Make sure you understand exactly what problem you're trying to solve. Ask the interviewer questions if anything is unclear or underspecified.
- Make sure you explain your plan to the interviewer before you start writing any code, so that they can help you avoid spending time going down less-than-ideal paths.
- If you can't think of a good way to do something, it often helps to start by talking through a dumb way to do it.
- Mention what invalid inputs you'd want to check for (e.g. input variable type check). Don't bother writing the code to do so unless the interviewer asks. In all my interviews, nobody has ever asked.
- Before declaring that your code is finished, think about variable initialization, end conditions, and boundary cases (e.g. empty inputs). If it seems helpful, run through an example. You'll score points by catching your bugs yourself, rather than having the interviewer point them out.

# Line up by alphabetically by first name

---



# FizzBuzz

In pseudo-code or whatever language you would like: write a program that prints the integers from 1 to 100. But for multiples of three print “Fizz” instead of the number and for the multiples of five print “Buzz”. For numbers which are multiples of both three and five print “FizzBuzz”.



# FizzBuzz Discussion

- This problem is an extremely famous interview question in software development, so it's common to have the exact question as part of a data science interview.
- A common mistake is for people to first if the number is divisible by 3 or 5 before checking if it's divisible by 15, but any number that is divisible by 15 is also divisible by 3 or 5. Thus, if 3 or 5 is checked first "Fizz" or "Buzz" might be printed in cases where "FizzBuzz" should.
- For fun, check out [FizzBuzz Enterprise Edition](#) or a [FizzBuzz TensorFlow machine learning model](#).

Tell if a number is prime

Write a function that, given a number, returns true if it's a prime number and false otherwise.

# Tell if a number is prime discussion

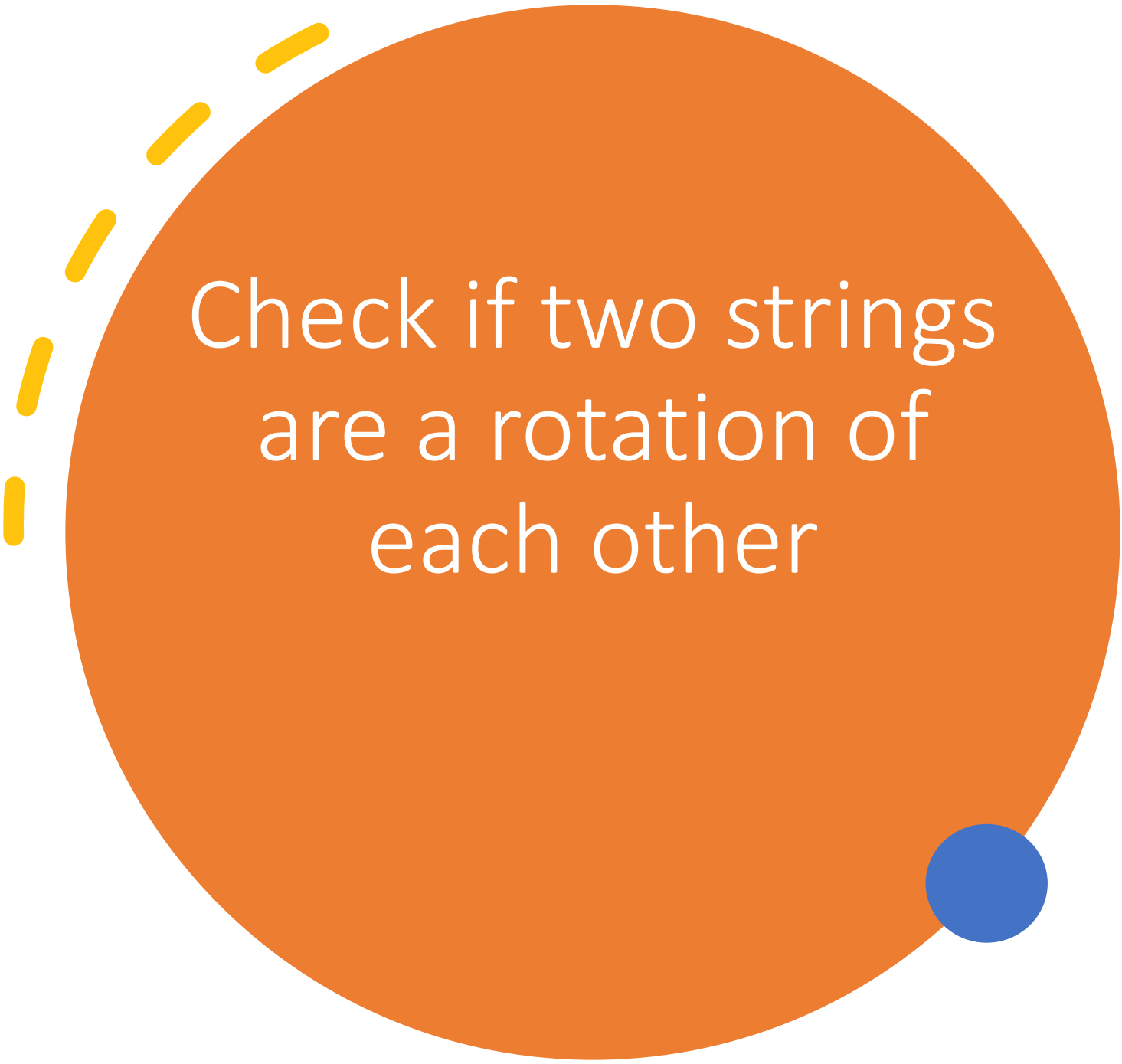
- Similar to fizzbuzz, this problem tests if you can write a for loop and also tests if you know how to right a function. You also need to know how to stop iterating when a condition is reached, so you can safely return true at the end if the for loop completes. There are small tricks you can add like realizing you don't have to check if the number is divisible by all numbers lower than it, just those lower than its square root, but the main point of the problem is simply to write a function that works.



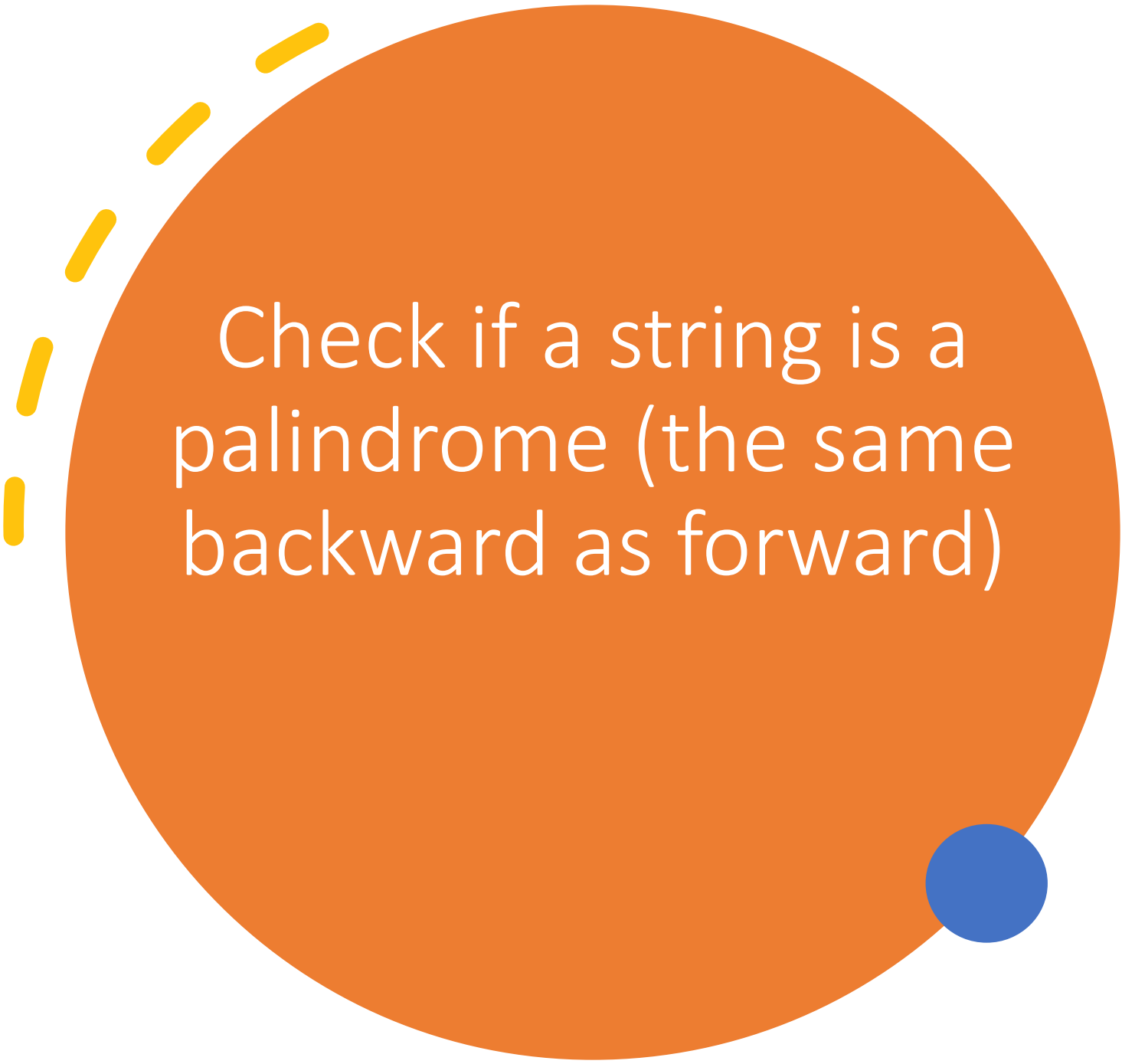
Write a function that  
adds all the numbers in  
a list



Find the missing number  
in a given integer array  
of 1 to 100.



Check if two strings  
are a rotation of  
each other



Check if a string is a  
palindrome (the same  
backward as forward)

John works at a clothing store. He has a large pile of socks that he must pair by color for sale. Given an array of integers representing the color of each sock, determine how many pairs of socks with matching colors there are.

For example, there are  $n=7$  socks with colors  $ar=[1,2,1,2,1,3,2]$ . There is one pair of color 1 and one of color 2. There are three odd socks left, one of each color. The number of pairs is 2.

**Input:** list of integers

**Output:** total number of matching pairs of socks John can sell.





Lilah has a string,  $s$ , of lowercase English letters that she repeated infinitely many times.

Given an integer,  $n$ , find and print the number of letter  $a$ 's in the first  $n$  letters of Lilah's infinite string.

Example 1:

$s = \text{"aba"} , n = 10$

Output: 7 (since there are 7  $a$ 's in abaabaabaa)

Example 2:

$s = \text{"a"} , n = 1000000000$

Output: 1000000000



# Links for more practice



<https://www.hackerrank.com>



<https://leetcode.com/>