1. Write a function that receives a number, $x$, and returns a list of all the divisors of $x$.

Before the loop, create an empty list to store the divisors e.g., `divisors = []`. Inside the loop, every time you find a divisor, use the method `append` to add the found divisor to the list e.g., `divisors.append(i)`, where $i$ is the divisor number being added to the list.

2. Write a function that takes a list of numbers and returns the cumulative sum; that is, a new list where the $i$th element is the sum of the first $i+1$ elements from the original list. For example, the cumulative sum of `[1, 2, 3]` is `[1, 3, 6]`.

3. Two words are anagrams if you can rearrange the letters from one to spell the other. Write a function called `is_anagram` that takes two strings and returns `True` if they are anagrams.

   **Hint:** Loop through the letters in the first of the words. See if you can find that letter in the second word. If you cannot find it, then they are not anagrams - you can return false. If you do find the letter in the second word, make sure to remove the letter from the second word, so that it doesn’t get used twice. Use the string library to see what methods are available for finding letters in a string.

   Strings are immutable, so you cannot simply delete one of the letters. But you can re-assign the string with just the letter that you want deleted missing by using slicing. For example, if you know you want the $i$th letter removed from the string $s$, you can say: $s = s[:i]+s[i+1:]$. Try this in the interpreter to get a better feel for it.

4. Write a function that receives a number and calculates its `fibonacci`.

   The Fibonacci number of $n$ is the Fibonacci number of $n-1$ added to the Fibonacci number of $n-2$. The first two numbers in the Fibonacci sequence are 0 and 1. So the function `fib(0)` should return 0; `fib(1)` should return 1; `fib(2)` is $0 + 1$, so returns 1; `fib(3)` is $1 + 1$, so returns 2. `fib(4)` is $2+1$, so returns 3; `fib(5)` is $3+2$, so returns 5; etc, etc.

   To calculate the fibonacci of a number, you will need to use a list. Before the loop, create an empty list to store the fibonacci sequence iteratively e.g., `fibseq = []`. Inside the loop, every time you calculate a new fibonacci number, add it to the list. To calculate the next Fibonacci number, you will need to look in the list and access the past two numbers e.g., `fibseq[n] = fibseq[n-1] + fibseq[n-2]`. (Note: We are not asking you to implement a recursive version of the algorithm - we’ll do that in a few weeks).