Problem 1.

Here is a function that computes the sum of a list of numbers.

```
def sum_nums(L):
  s = 0
  for num in L:
    s += num

  return s
```

Write a function with the signature `def count_evens(L)` that returns the number of even integers in the list `L`. Assume `L` only contains integers.

Problem 2.

You can use str() to convert objects to strings:

```
>> str(42)
'42'
```

In particular, you can obtain the string representation of a list list0 by using str()

```
>> list0 = [1, 2, 3]
>> str(list0)
'[1, 2, 3]'
```

Without using `str()` with arguments that are lists (using it with arguments that are not lists is fine), write a function `list_to_str(lis)` which returns the string representation of the list `lis`. You may assume `lis` only contains integers.
Reminder:

```
>> "hello" + "python"
"hellopython"
```

Problem 3.

You can compare lists using the == operator:

```
>> l1 = [1, 2, 3]
>> l2 = [4, 5, 6]
>> l3 = [1, 2, 3]
>> l1 == l2
False
>> l1 == l3
True
```

Without using the `==` operator to compare lists (you can still compare individual elements of the lists), write a function `lists_are_the_same(list1, list2)` which returns `True` iff `list1` and `list2` contain the same elements in the same order. You'll need to use a loop (either while or for)

Problem 4.

Write a function with the signature `list1_start_with_list2(list1, list2)`, which returns `True` iff `list1` is at least as long as `list2`, and the first `len(list2)` elements of `list1` are the same as `list2`. Note: `len(lis)` is the length of the list `lis`, i.e., the number of elements in `lis`.

First write the function without using slicing ("slicing" means saying things like `list1[2:5]` we haven't covered that), and using a loop.

Problem 5.

Write a function with the signature `match_pattern(list1, list2)` which returns True iff the pattern `list2` appears in `list1`. In other words, we return `True` iff there is an `i` such that $0 \leq$ i $\leq$ len(list1)-len(list2) and

```
list1[i] = list2[0]
list1[i + 1] =list2[1]
.
.
.
list1[i + len(list2) - 1] = list2[-1]
```

For example, if `list1` is `[4, 10, 2, 3, 50, 100]` and `list2` is `[2, 3, 50]`, `match_pattern(list1, list2)` returns `True` since the pattern `[2, 3, 50]` appears in `list1`

Problem 6.

Write a function with the signature `duplicates(list0)`, which returns `True` iff `list0` contains at least two adjacent elements with the same value.

Hint: you need to compare `list[i]` and `list[i+1]` for all i.

Problem 7.

Assume that you have a list of coordinates at times $t = 0.1, 0.2, 0.3, ...$ The list might look like `x = [0.5, 0.6, 0.89,` In lecture, we computed the instantaneous velocity at time t_i as $v_i = \frac{x_{i+1}-x_i}{0.1}$. But if the position measurements are inaccurate, it would be better to estimate the velocity using the average over multiple measurements.

Part (a)

Write a function that would estimate the instantaneous velocity at time t_i by computing a weighted average of the estimate using coordinates $i-1$ and $i+1$ and the estimate using coordinates $i-2$ and $i+2$.

The input would be a list of coordinates `x` and an index `i`. The output would be the estimated velocity at time t_i.

You have flexibility here about choosing how to compute the weighted sum and how to deal with coordinates near the beginning and the end of the list.

Part (b)

Generate data with random noise added to it: add something like `0.1*random.random()` to each coordinate.

Estimate the instantaneous velocity using both the functions you wrote, and show that the weighted average is more accurate than the simple difference if the measurement is noisy enough.