This week, we will be practicing lists and loops. We will be reimplementing some of Python’s functionality in order to understand how it might work under the hood. We will also implement some custom functions that Python doesn’t supply.

**Question 1.**

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

```python
>>> str(42)
42
```

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

```python
>>> 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 if 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:

```python
>>> hello + python
helloython
```

**Question 2.**

You can compare lists using the `==` operator:

```python
>>> 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)

**Question 3.**

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]`), and using a loop.
Question 4.

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 <= i <= 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()` returns True since the pattern `[2, 3, 50]` appears in `list1`.

Question 5.

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