**Mapping**

**Introduction:**

We have just created a contract that stores multiple `Person`’s names and favorite numbers in a list. In this session, you will learn about mappings, their functionality, and when it is more advantageous to use them.

**Avoiding Costly Iterations:**

If we want to know just one person’s favorite number (e.g. Chelsea’s) but our contract holds a (long) array of `Person`, we would need to iterate through the whole list to find the desired value:

List\_of\_people.add(Person(“Pat”, 7));

List\_of\_people.add(Person(“John”, 8));

List\_of\_people.add(Person(“Mariah”, 10));

List\_of\_people.add(Person(“Chelsea”, 232));

// Go through all the people to check their favorite number.

// If name is “Chelsea” -> return 232

Iterating through a long list of data is usually expensive and time-consuming, especially when we do not need to access elements by their index.

**Mapping:**

To directly access the desired value without the need to iterate through the whole array, we can use mappings. They are sets of (unique) “keys” linked to a “value” and they are similar to hash tables or dictionaries in other programming languages. In our case, looking up a name (key) will return its correspondent favorite number (value).

A mapping Is defined using the mapping keyword, followed by the key type, the value type, the visibility, and the mapping name. In our example, we can construct an object that maps every name to its favorite number.

mapping (string => uint256) public nameToFavoriteNumber;

Previously, we created an `addPerson` function that was adding a struct `Person` to an array `list\_of\_people`. Let’s modify this function and add the struct `Person` to a mapping instead of an array:

nameToFavoriteNumber[\_name] = \_favoriteNumber;

**important:** Mappings have a constant time complexity for lookups, meaning that retrieving a value by its key is done in constant time.

**note:** The default value for all key types is zero. In our case, `nameToFavoriteNumber[“ET”]` equals 0.

**Conclusion:**

Mapping can be a versatile tool to increase efficiency when attempting to find elements within a larger set of data.