Common problems that we haven't looked at so far:

- How to represent real life data
  - Something that has more than one or two variables
- How to organize such real life data
- How to extract information from large amount of data efficiently
- What is a collection
  - What are some properties of collection we care about

Solutions:

What is Data?
Data are individual facts, statistics, or items of information, often numeric. In a more technical sense, data are a set of values of qualitative or quantitative variables about one or more persons or objects.
(Source: Wikipedia)

What is Information?
Information is processed, organized and structured data. It provides context for data and enables decision making process.

For example, a single customer's sale at a restaurant is data – this becomes information when the business is able to identify the most popular or least popular dish
(Source: Wikipedia)

How to extract Information from that Data???

Well! That's what we are going to learn in this Unit..
Data structures, ADT, difference between them..

But before that let's take few examples of real life problems:

Let's consider some real life problems:

- Arranging books in a library:
  - Book1: name, author, genre, date_published, ISBN
  - Book2: ~~name~~, ~~author~~, ~~genre~~, ~~date_published~~, ~~ISBN~~

```
char name[1024] = "Book1 * Book2 * Book3....."
char author[ ] = "a1 * a2 * a3 ....."
char genre[ ]
int date_published[ ]
int ISBN[ ] =
```

- Item records in supermarket
- A catalog of art pieces in a museum
- Apartment hunting: Record of apartments

Let's consider apartment hunting:
You are looking for an apartment
You consider a bunch of apartments listed from different websites.

You note down apartments that you liked before your friend arrives.

Properties of an Apartment:

address
rent
walk_score 10 ← is the best

Let's turn into data that we can store in a C program:

object ≠ CDT ⇐ only a collection of data

object ↓ data + functions

CDT ⇑ x no functions

```
typedef struct ~~ap~~ apartment... missing tag
{
    int rent;
    char address[100];
    float score;
} ~~B~~ apt ;
```

One Compound Data Type (CDT)

Why didn't we use arrays instead of CDT?

- Arrays are fixed size, we have to 'forecast' how many we will need. Once we defined them they're fixed in size and we can't add more
- If we make the arrays big enough we'll likely be wasting space
- If we make it too small - toast
- And it's very annoying to maintain and keep things in sync since code has to handle multiple arrays all at once, every function we add to our program need to ensure they're updating all the arrays correctly and nothing gets forgotten
- Easy to end up with a mess with our data
- Conceptually it's not nice, we have data about individual items of some type spread into multiple arrays

So, we need a better way

- first, we will find a way to bundle up (bento box) all the information pertaining a single item in our collection in a single variable (it's NOT an object!).
- Generally speaking, objects bring the full object oriented functionality (methods, data, virtual functions, inheritance, etc, etc) whereas structs are just organized memory. Structs may or may not have support for methods / functions, but they generally won't support inheritance and other full OOP features.
- We will create a special type of variable that contains multiple chunks of information

- a compound variable called a compound data type or CDT

Let's create a CDT based on our apartment hunting example.