Lists

◎ Objectives

At the end of this lesson, you should be able to:

- **describe** the concept of lists
- **manipulate** lists to represent data
- **construct** lists from different data fields

1. Context

Programmers are used to manipulating data to simplify solving problems. Imagine the problem of having to grade students. You want to store the grade in a variable. If you have **one** student, it's easy.

</> Code

```
student_grade = 12
```

If we add another student, we start to see the problem:

</> Code

```
student1_grade = 12
student2_grade = 4
```

We have to create a variable **for each** student.

Idea

This is highly impractical. Imagine you have 12 students, this implies 12 variables. What happens if a new student enters the class?

2. Lists

The need to handle multiple elements such as these implied the creation of a more complicated variable type: **lists**.

A list is a set of variables grouped into one, making it ideal to handle larger quantities of data.

</> Code

```
l = ["string", "test", "moodle", "rubika"]
grades = [12, 8, 17, 4, 19, 20, 14]
```

As you can see in the example above, you can create a list just like any other variable. The notable difference is that the elements are enclosed by **square brackets** `[` and separated by **commas** `,`.

Indexation

Elements in a list are **indexed**, meaning they have an **index** that allows us to access it.

List indexes start at the value `0` and go up for each element.

Consider the following example:

</> Code

```
week = ["mon", "tue", "wed", "thu", "fri", "sat", "sun"]
```

The indexes are the following:

item	mon	tue	wed	thu	fri	sat	sun
index	0	1	2	3	4	5	6

Information

Accessing an element in a list is done by using it's index. To get the element corresponding to "thursday", you would do:
`day = week[3]`

Operations on list

A couple of the most common operations that can be done on lists are: **adding** an element, **adding** multiple elements or **removing** an element.

This can be done in the following ways:

</> Code

```
# Adding individual elements to a list
week = ["mon", "tue", "wed", "thu", "fri"]
week.append("sat")
week.append("dim")

# Adding two lists together
week_days = ["mon", "tue", "wed", "thu", "fri"]
week_end = ["sat", "sun"]

week = week_days + week_end
```

```
# Removing an element
del week[2]
```

3. Useful functions

Here are a couple of useful functions for dealing with lists:

</> Code

```
# Add an element
l.append(element)
# Reverse list
l.reverse()
# Get the size of the list
len(l)
# Find out if an element is in a list
element in l
# Return a sorted version of the list
s = sorted(l)
# Get the smallest / biggest element in the list
min(l) / max(l)
```

Exercise

Create a list that contains the four families in card games.

Exercise

Create a list that contains the thirteen types of cards available in a family.

Exercise

Try multiplying a list by a number. What do this give you? Can you relate this to what we have seen with strings?

Exercise

Sort the following list: `l = ["1", "5", "10", "21", "2", "0"]`
Do you find this normal?

Exercise

Use the **sum** function to find the mean of the students' grades from earlier.