1 Slicing Lists

Remember that we are able to slice strings? We can do the same with lists.

For example,

```python
subjects = ["Computer Science", "Biology", "French", "History"]
print(subjects[0:2])
print(subjects[:1])
print(subjects[:2])
print(subjects[2:])
print(subjects[:])
```

This prints:

2 Aliasing

Sometimes we have two variables that are names of the same object, they refer to the same object. This is called aliasing.

Q. If the object is mutable and we change what one variable refers to, what happens to the other variable?

A.

Note: Slicing creates a new object.

Consider the following code snippet:

```python
subjects = ["Computer Science", "Biology", "French", "History"]
other = subjects # We have 2 names for the same list object.
subjects[0] = "Commerce"
```

Q. What is subjects now?

A.

Q. What is other now?

A.
Another code snippet:

```python
subjects = ['Computer Science', 'Biology', 'French', 'History']
subjects_clone = subjects[:]
subjects[0] = 'Philosophy'
```

Q. What is `subjects` now?

A.

Q. What is `subjects_clone` now?

A.

3 Nested Lists

We can make lists that contain anything: `ints`, `bools`, `strings`, even other `lists`!

Example.

Let’s create a list of student and grade info:

```python
grades = [['999888777', 78], ['111222333', 90], ['444555666', 83]]
```

Q. How do we refer to elements of the lists within the list?

A.

Design and implement each of the following functions:

- `average_grade(grade_list)` return the average grade of all of the students in the list of lists `grade_list`, where the inner lists contain a student ID and a grade.

- `get_student_IDS(grade_list)` return the list of student IDs. The input list `grade_list` is as above.
4 Nested Loops

How can we iterate through *everything* in a nested list? For example consider the list:

```python
measurements = [[33, 34, 30], [29, 31, 34], [28, 27, 30]]
```

Use `for` loops to print the average of each inner list:

Q. How can we do the same thing using a built-in function called `sum`?

A.

Modify this code to create a list of the *averages*:
5  Aliasing again

Consider very carefully what happens in the following:

```python
>>> lst = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> lst_clone = lst[:]
>>> lst
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> lst_clone
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> lst[0][2] = 'surprise!
```

>>> lst
[[1, 2, 'surprise!'], [4, 5, 6], [7, 8, 9]]
>>> lst_clone
[[[1, 2, 'surprise!'], [4, 5, 6], [7, 8, 9]]

Draw the memory after each assignment statement to make sure you understand what’s going on!

6  Practice with lists

Design and implement each of the following functions:

- **count(lst, s)**: given a list `lst` and a string `s`, return the number of occurrences of `s` in `lst`.
- **round(num_lst)**: given a list `num_lst` of numbers, return a list of corresponding integers, in order.
- **upper_all(str_lst)**: given a list of strings, convert each string to upper-case. Notice that here you need to modify the input list! Think carefully how to test this function!
- **get_grade(grades, grade)** given a list (of lists) of all student grades, return a list of student grades for a particular work. Here is an example of an input list:

  ```
  [['0123456789', "John Smith", 89, 67.5, 80, 55.8],
  ['9876543210', "Teddy Bear", 100, 35.0, 77, 64],
  ['7894561230', "Bill Gates", 34, 78, 100, 100]]
  ```

That is, the input list contains a list per student. This list contains student number, name, and then a list of zero or more grades. You may assume that the number of grades for each student is the same. The argument `grade` specifies the grade we wish to retrieve (a number from 0). For example, if `grades` is the list above, then the call `get_grade(grades, 1)` should return the list:

```
[['0123456789', "John Smith", 67.5],
 ['9876543210', "Teddy Bear", 35.0],
 ['7894561230', "Bill Gates", 78]]
```