LIST BASICS

COMP130 – INTRODUCTION TO COMPUTING

DICKINSON COLLEGE

LISTS

- A ***`List`*** is a Python *object* that that holds a ***sequence*** of ***values*** called ***elements*** (or ***items***).
  - A Python `List` is represented as comma delimited values between `[ ]`.
    - `cs_profs = ['Braught','Fruchter','MacCormick','Siddiqui','Skalak']`
    - `cs_classes = [130, 132, 190, 232, 256, 290, 314, 332, 491, 492]`
  - When a `List` is an element in another list, it is a ***nested `List`***:
    - `sections=[[130, 1, 'Fruchter'], [130, 2, 'Braught']]`

STRINGS ARE SEQUENCES – LISTS ARE SEQUENCES

- Many of the things we could do with a String (a sequence of characters) we can do with a List (a sequence of elements).
  - `List` elements and slices are accessed using `[ ]` in the same way characters are in a `String`.
    - `cs_profs = ['Braught','Fruchter','MacCormick','Siddiqui','Skalak']`
    - `cs_classes = [130, 132, 190, 232, 256, 290, 314, 332, 491, 492]`
    - `next_class = cs_classes[1]` → `132`
    - `soph_classes = cs_classes[2:6]` → `[190, 232, 256, 290]`
    - `sr_seminar = cs_classes[8:]` → `[491, 492]`

ACCESSING NESTED LISTS

- Elements of a nested list are accessed using the index of the desired element(s) from the outside in:
  - `sections=[[130, 1, 'Fruchter'], [130, 2, 'Braught']]`
  - `sect2 = sections[1]` → `[130, 2, 'Braught']`
    `prof2 = sect2[2]` → `'Braught'`

Or

- `prof2 = sections[1][2]` → `'Braught'`

LISTS ARE MUTABLE

- A `List` is ***mutable***, meaning that the elements of the sequence can be changed.
  - `schedule = ['COMP132', 'MATH170', 'HIST203', 'SPAN116']`
  - Elements can be changed (replaced)
    - `schedule[2] = 'MUSC204'`
  - Slices can be changed (replaced)
    - `schedule[1:3] = ['PHIL101', 'ECON116']`
  - Elements can be added (inserted, appended)
    - `schedule.append('ARTH378')`
    - `schedule.extend(['ARTH378', 'INBM207'])`
  - Elements can be deleted (removed)
    - `drop = schedule.pop(3)`
    - `drop = schedule.remove('MATH170')`

TRAVERSING LISTS

- A `List` can be ***traversed*** in a number of equivalent ways:

```
table=['Liverpool', 'Leicester City', 'Chelsea', 'Manchester City', 'Sheffield United']
```

- a `while` loop

```
def print_table(table):
    index = 0
    while index < len(table):
        print(str(index + 1) + '. ' + table[index])
        index = index + 1
```

- a `for in` loop

```
def print_table(table):
    index = 0
    for team in table:
        print(str(index + 1) + '. ' + table[index])
        index = index + 1
```

- a `for in range` loop

```
def print_table(table):
    for index in range(len(table)):
        print(str(index + 1) + '. ' + table[index])
```

Each time through the loop `index` takes on the next value in the `range`.

The `range` is: 0, 1, 2, 3, 4