Class 9 - Notes

Upcoming Schedule

If you did not earn a Yellow Belt (by receiving a Gold Star on Project 1), see Yellow Belt Promotion for information on how to advance. There is no prescribed deadline on when you need to complete this, but **additional requirements for advancing will be added for anyone who has not completed it by Sunday** (14 Feburary).

Project 2 is posted now, and is due at the beginning of class on **Monday, 15 February**. Everyone should have at least completed Problem 3 of Project 2 by today. If you have not gotten that far, it is time (or past time!) to get cracking!

Dave will not be able to hold his usually schedule office hours tomorrow (Thursday morning); if you need to meet with me, send email (that includes a list of all the times you have available) to arrange a time. Yuchi has office hours today (4-5pm, Rice 514) and Friday (immediately after class).

Lists

The Python standard library provides several useful datatypes for managing collections of data. The *list* type is a mutable, ordered collection of elements. *Mutable* means we can modify the value of the list after it is created (for example, but adding or deleting elements). *Ordered* means the elements are in a well-defined order, and we can access them by their index. The elements in a *list* can be any type, including (of course!) other lists.

Square brackets construct lists:

```
>>> p = [3, 4]
>>> p
[3, 4]
```

We can access elements of a list using *list*[***index***]: (indices start counting from 0)

```
>>> p[0]
3
>>> p[1]
4
>>> p[2]
Traceback (most recent call last):
  File "<pyshell#139>", line 1, in <module>
    p[2]
IndexError: list index out of range
```

Growing Data

The simplest compound data structure is a *pair*. One way to implement a pair is to use the standard list type (but later we will also see that it is possible to build pairs without using any collection type!).

```
def make_pair(a, b):
    return [a, b]

def pair_first(pair):
    return pair[0]

def pair_last(pair):
    return pair[1]
```

How can we make a *triple* using a pair?

List ::= *Element List*
List ::= **None**

```
# A list is a pair where the second part is a list,
#          or None

def make_list(first, second): return make_pair(first, second)

def list_first(lst): return pair_first(lst)

def list_rest(lst):  return pair_last(lst)

# Any resemblence to actual politicians is purely pythonic.
democritoots = make_list('Berny', make_list('Hillarie', None))
republicrats = make_list('Thump', make_list('Kasitch', ..., make_list('Carsoon', None)))))))
```

Is it better to solve problems by thinking about what we need to *do* to solve the problem (*procedures*) or by thinking about what we need to *represent* to solve the problem (*data*)?