Class 7 - Notes

Upcoming Schedule

Due now: **Project 1**
Before **Wednesday, 10 February**:
Complete Udacity cs101 Lesson 3: *How to Manage Data* (Notes) and Lesson 3: *Problem Set*
Due on **Monday, 15 February**:
Project 2 (will be posted in a few days)
Before **Friday, 19 February**:
Udacity cs101 Lesson 4: *Responding to Queries* (Notes) and Lesson 4: *Problem Set*

Python Tutor

One great resource for understanding python code is Philip Guo's Python Tutor. It lets you step through Python code forwards and backwards, and visualize what is going on.

Binary Numbers and Rules of Evaluation

Binary numbers are base 2, instead of the base 10 decimal numbers we commonly use. This means we can represent any number using only **0**s and **1**s, and the value of each *Bit* scales as a power of two (so instead of having a "ones" place, "tens" place, "hundreds" place, "thousands" place, we have a "ones" place","twos" place, "fours" place, "eights" place, etc.). In Python, a number literal that starts with `0b` is interpreted as a binary number:

What is the value of `0b11111111` (as a decimal number)?

IntegerLiteral ::= *BinLiteral*
BinLiteral ::= **0b***BinDigits*
BinDigits ::= *BinDigits BinDigit*
BinDigits ::= *BinDigit*
BinDigit ::= **0**
BinDigit ::= **1**

Show how to derive **0b101** with this grammar starting with *IntegerLiteral*:

Provide semantic rules for the grammar that give the value (as a decimal number) for every *BinaryLiteral*:

(3) *BinDigit* ::= **0**
Value(*BinDigit*) =

(4) *BinDigit* ::= **1**
Value(*BinDigit*) =

(5) *BinDigits* ::= *BinDigit*
Value(*BinDigits*) =

(6) *BinDigits* ::= *BinDigits BinDigit*
Value(*BinDigits*) =

Test Grammar

Here is a simplified excerpt of the *Test* grammar from *https://docs.python.org/3/reference/grammar.html*.

Test ::= *NotTest*
NotTest ::= **not** *NotTest*
NotTest ::= *Expression*
Expression ::= **True**
Expression ::= **False**

Develop rules of evaluation for the grammar above that matches how things are interpreted by the Python interpreter.

