8/30 CSP(L) notes:

Overview

Notes:

- ACM meeting, thursday sept 5 in W181 PBB at 6-7
- No classes monday
- First lab will released today to look at
- OH and SI also begins sept 3
- Lab0 is not like other labs, not handing anything in. It's about installing anaconda.If u get stuck maybe post on piazza.
- **Make sure to download and read lab0 carefully before discussion**
- **.**
- **.**
- All programs manipulate *data,* that is, abstract representations of real world concepts.(y)
- Ultimately, all data is represented inside the computer as a collection of *binary digits*, or *bits*
- If u randomly reorder bits(0 and 1s), it will mess up interpretation
- Binary (“bit”) representations are convenient for the kind of electronic manipulation that takes place inside the machine, where memory is basically a collection of switches, and all calculations are simple operations are hits
- Origianlly u could decide….
- Boolean dara(true/false), they may elect to have 1 denote true and 0 denote false, or vise versa
- All high level PL, adopt conventions that attribute meanings to bits, whether jpg images, floating point numbers, integers, stringers, mpeg video, arrays.
- Today, today cross machines have all been standardized, consistent rules, cons input output format
- Every programming language provides some built in *primitive* data types, which hides ugliness ofhow something is represented internally
- Each *primitive* data types comes complete with predefined set of operations to manipulate them
- In general, more complex, custom, types can be constructed by layering and combing *primitive types* together
- Python’s built tin data provides somewhat richer and more sophisticated than those previous generation languages
- Python is also dynamically typed(type flexible), something that is explored later
- Standard data type:
- Numeric types: **integer, floating points,** complex

- Logic types: **Boolean.**
- Sequence types**: string,** byte, bytearray, **list, tuple,** and **range**
- Mapping types**: dict.**
- Set types**: set,** frozenset
- Each data type comes “bundled” with a set of operations on those types
- Also later, objects, functions, methods, modules, classes and others
- Interanlly, all data is represented as collections of bits.
- Integer:
- Used to represent whole numbers: -6,-5 … 0,1,2
- Legal integers are input as[+,-] (optional) followed by a series of 1 or more digits[0-9]
- Assumes a base 10 interpretation, with exception of the inter 0, leading 0 digits are not allowed.
- Other bases(octal, hexadecimal) are also possible but not relevant rn
- Integers are not limited in size, so they can require an arbitrary number of digits(no max_int or min_int)
- Properties of integers using Python read, eval, print loop, or REPL
- REPL= interactive calculator
- R= attempts to parse expression, errors in syntax are reported instantly
- E= intercepts the expression, caching the result in memory, errors in semantics are reported here
- P= fetches the result from memory and outputs a printed representation of result
- L=?
- Python builds internal representation
- -0, +0, are same thing, same number, same effect basically
- 1,024, will give syntax error, 020, syntax error, no leading zeros,
- Base conversion 0o20(octal base 8?) + 0x10(hexadecimal base 16) = 32
- 3 + +4, 7, 3=3, +4=4
- 3 * 4, 12
- -3 -3, -6, first minus is part of integer representation,
- -3 // 3, -1,
- -3 % 3 #modulo, 0(remainder) in integer division.
- -3 / 3, -1.0 (also division?)
- 4 / 3 =1.3333333333, “regular”, 4 // 3, “integer division”, 1
- 34566**45, exponentation, u get whole answer, no limit to size to integer u can represent
- Python does not care about spaces, like4/3 same as 4 / 3
- WE have 1 data type now: built in functions, modules and methods
- max(3, 4), 4, min(3, 4), 3, min (-1,1), -1, min(-1,1) + max(-1,1),0
-
- all type of parentheses or brackets have different meanings
-