SPIS Wednesday 10:15am Lecture

Computational complexity

- A way to describe an efficiency of an algorithm
- Turning on lights: O(1)
- Call out counting: O(n)
- Stand up counting: O(log2(n))

Python coding

- # comment
- Displaying output:
  - print ('Hello World') # can take multiple strings separated by commas
- Scalar Object Types:
  - int for whole numbers
  - float for real numbers
    - consider precision
  - bool for True or False
  - type (xyz) reports the type of xyz
- Non-Scalar Object Types:
  - class

- Operations:
  - + addition (overloaded for strings)
  - - subtraction
  - * multiplication (overloaded for strings)
  - // int division
  - / float division
  - % modulus
  - ** power
  - Can include addition, too:
    - abc = abc + 3 is the same as: abc += 3
- Comparison
  - == equality
  - != inequality
  - < less than
  - <= less than or equal to
  - > greater than
  - >= greater than or equal to

- Bool operators
  - and
  - or
  - not

- Variables
  - = assignment: associates variable names with objects (scalar: int, float, bool)
    - abc, def = 3, 4
  - Select names well
  - Case sensitive
  - Can contain letters, digits, _, (can't start with digit)
  - Can't be reserved words (keywords in language)
  - Typing by context
- Blocks of code are defined by indenting (not curly braces)
- If statement example: (elif means: else if)

```
if abc == 2:
    print ("abc is 2")
else:
    print ("abc is not 2")
```

- Loops:
  - # while loop example

```
abc = 0
while abc != 10:
    print abc
    abc = abc + 1
```

- # for loop example
  abc = 0
  for xyz in range (0, abc):
      print xyz
- break leaves loop early