QUIZ 1 REVIEW, PART 1
WHAT IS COMPUTATION?

- Declarative knowledge
  - statement of fact
- Imperative/Procedural knowledge
  - “how-to” knowledge
  - Frame your problem so a computer can solve it
EXAMPLE – IMPERATIVE KNOWLEDGE:

- Compute the sum of the odd numbers between 1 and N

- Initialize a counter variable to zero. For each number from 1 to N, if the number is not divisible by two, add the number to the counter.
RULES OF THE LANGUAGE

- Syntax – which statements are well-formed
- Static Semantics – which statements have meaning
  - Static semantic errors happen when you put the right types of pieces in the right order, but the result has no meaning
- Semantics – association of each syntactically correct statement that has no semantic errors with some meaning
**Variables**

**Math**
- Alphabetic character
  - x, y, z
- Represents arbitrary or unknown number

**Computer Science**
- Alphanumeric, underscores
  - x, letters_dict
- Represents a storage location containing some value
**Types**

- **Booleans**: True, False
- **Strings**: ‘abc’, ‘123’, ‘!@#$%’
- **Numbers**:  
  - int: -1, 0, 5, 27  
  - float: -5.4, 0.0, 17.9
TYPE ISSUES

- $1 / 2 = 0$ (integer division)
- $1 / 2.0 = 0.5$ (float division)
- float(1) / 2 = 0.5 (casting)

NOTE: integer division truncates the answer – it does NOT round

7.0 / 3 = 2.33333333  
7 / 3 = 2
7.0 / 4 = 1.75  
7 / 4 = 1
**Operations**

- **Arithmetic operations (follow PEMDAS rules)**
  - +, -, *, /
  - ** for exponents
  - % modulo to get remainder

- **String operations**
  - + for concatenation
  - * to repeat

- **Boolean comparators**
  - >, >=, <, <=, ==, !=

- **Logical operators**
  - and, or, not
CONTROL: IF

Only run a block of code if a certain condition is True

if condition:
    #some code to run
elif other_condition:
    #some other code to run instead
else:
    #some more code to run if the other conditions weren’t met
**CONTROL: LOOPS**

- **for**:
  - Repeat this block of code once per element in the given iterable
  
  - **for var in iterable:**
    - #code

- **while**:
  - Repeat this block of code until a given condition is False
  
  - **while condition:**
    - #code
CONTROL: FOR LOOPS

```python
>>> word = 'hello'
>>> for letter in word:
    print letter

h
e
l
l
o

>>> char_list = ['a', 'b', 'c']
>>> for char in char_list:
    print char

a
b
c

>>> word = 'hello'
>>> for i in range(len(word)):
    print word[i]

h
e
l
l
o

>>> char_list = ['a', 'b', 'c']
>>> for i in range(len(char_list)):
    print char_list[i]

a
b
c
```
I’m thinking of a number between 1 and 100.

```python
secret_number = 33
successfully_guessed = False

while not successfully_guessed:
    guess = int(raw_input("guess a number from 1 to 100: "))
    if guess == secret_number:
        successfully_guessed = True
        print "you win!"
    else:
        print "wrong number :(")
```
Bisection
FUNCTIONS: WHY ARE THEY USEFUL?

- Decomposition
  - hangman
    - word_guessed()
    - print_guessed()
    - play_hangman()

- Abstraction
  - get_frequency_dict(sequence)
    - input sequence
    - output freq_dict
**FUNCTIONS: FORMAT**

Keyword | Name | Parameters/Arguments
--- | --- | ---

```python
def add_as(word):
    output = ''
    for i in range(len(word)):
        output = output + word[i] + 'a'
    return output
```

Return statement (if omitted, function will return None by default)

```python
word_with_as = add_as(word)
```