CIS 210 Review

What have we learned, anyway?
(And will it be on the exam?*)

*The absolute least favorite question of any instructor anywhere.
Schedule

Today, Friday: Review
Contest entry due tonight; Optional assignment (symcalc) due Friday

Wednesday: Idutorob & review

Friday 2013.3.22 10:15-12:15 Final exam
(alternative time Wednesday is full)

Also: Remember to do course evaluations
(I do read them, and they do matter)
Idutorob: The Incredible Duck Tournament of Robots

(Actually Sudoku, but Sudoku is kind of like duck-sled racing.)

Coming Wednesday to a classroom near you

Photo credits: http://www.city-data.com/forum/alaska/1522185-robot-ken-robodog-go-dog-sledding.html,
http://steamcommunity.com/groups/rdzgroup
Additional contests

Juilfs programming contest: late spring term
(specifically for 1xx/2xx students)

Luks programming contest: Saturday April 13
(all students, even grads; hard but fun!)

ACM regional contest (November)
(team practices as class in Fall term)
Review for Final

Sample questions from prior exams, and a few extras
Goals for CIS 210

Learn computer science concepts
Problem solving with computation
General programming skills
  • includes designing programs to be understood and modified by humans
  • includes testing, debugging
Expressing programs in Python
  • but the programming concepts apply to other languages
Things to expect

Functions and methods:
  Scope, parameter passing, returning results

Boolean values

Recursion

Loop design

Classes/objects/modules

In the “what does this print” part
and in the “write a function/method to do x” part
Fall 2012 is a good model ...

Also Python, with introduction to classes and nearly the same projects; expect final exam to be similar but not identical

Also Winter 2012, with some differences:
We haven’t studied subclasses and inheritance
We have studied functions as objects
(and we’re using Python)
F12, Q2: What does q2() print?

def rescale(x, low, high):
    if x < low:
        return 0
    if x > high:
        return 100
    x = (100 * (x - low)) // (high - low)
    return x

def q2():
    minscore = 10
    maxscore = 30
    x = rescale(35, minscore, maxscore)
    y = rescale(20, minscore, maxscore)
    print(x, y)
def swap(ar, i, j):
    t = ar[i]
    ar[i] = ar[j]
    ar[j] = t

def q4():
    a_lis = [ 10, 20, 30 ]
    b_lis = a_lis
    c_lis = [ 1, 2, 3 ]
    swap(a_lis, 0, 2)
    swap(b_lis, 1, 2)
    swap(c_lis, 0, 1)
    print(a_lis[0] + c_lis[0])
    print(a_lis[1] + c_lis[1])
    print(a_lis[2] + c_lis[2])
def present(dict, word):
    """Determine whether string word is in dict.
    Arguments:
    dict: a list of strings, in sorted order
        (dict[i] > dict[i-1] for all
         i in range 1..len(dict)-1 )
    word: a string to search for
    Returns:
    True if word is in dict (else False)
    """

(up to 15 points for binary search,
10 points for linear search)
10F, q2: loop & conditional count

sum = 0
for k in range(10):
    if k % 3 == 0:
        sum = sum + 1
print(sum)
def abdiff(x, y):
    if x > y:
        x = x - y
    else:
        x = y - x
    return x

def q3():
    x = 7
    y = 4
    x = abdiff(x, 5)
    y = abdiff(y, 14)
    y = y + x
    print(y)
def avg_pos(ar, k):
    """Find the average (arithmetic mean) of the positive entries in the first k elements of an array.
Arguments:
    ar: array (list) of numbers, with at least k elements.
    k: an integer: how many elements in ar to consider.
Returns:
    Average of the positive numbers in the range ar[0]..ar[k-1], or -1.0 if there are no positive numbers in that range.
Example: avg_pos([2.0, -2.0, 3.0, 0.0, 5.0], 4)
    returns 2.5.  (2.0 + 3.0)/2; 0.0 and -2.0 are omitted as negative, and 5.0 is beyond k position k.
    """
In the imaginary word game *Words with Pets*, a vowel is worth 2 points, and a consonant (any other letter) is worth 3 points. Write a function to determine the value of a word, using the is_vowel function to distinguish vowels from consonants.

```python
def is_vowel(letter):
    return letter in ['a', 'e', 'i', 'o', 'u']

def score(word):
    """Calculate value of word.
    Arguments: word is a string containing only lower case letters ('a'-'z').
    Returns: score for word, 2 pts for each vowel and 3 for each other letter."""
```

(11F final was a bit too easy)
longest run

A “run” is a sequence of consecutive, equal elements. For example, “abcccdedfgchci” has a run of 3 c’s and 2 f’s (and runs of 1 ‘a’, and 0 ‘z’).

def longest_run_length( str ):
    """Return the length of the longest run in string str""

def longest_run_letter( str ):
    """What character appears in the longest run. str has at least one character. In case of a tie, return the letter that appears in the first of the longest runs."""
def flatten(tree):

given a nested array like [[a, b], c, [d, e, [f], [ ]]]

(which is really a tree)

return a flattened version like [a, b, c, d, e, f]

(all the nutritious leaves, none of the stringy bark)
filter (function variables)

Write a function

filter(ar, f)
which returns a list of elements in ar for which f (el) is True

def pos(x):
    return x > 0

filter([0, -1, 1, 2, 3, -5], pos) returns [1, 2, 3]
def q1():
    x = 3
    y = 5
    if x > y:
        x = x // 2
    else:
        y = y // 2
    if x > y:
        x = x // 2
    else:
        y = y // 2
    print(x, y)
def q1():
    x = 3
    y = 5
    if x > y:
        x = x // 2
    else:
        y = y // 2
    if x > y:
        x = x // 2
    else:
        y = y // 2
    print(x, y)
W12 Q2:

def win(ar, low, high):
    count = 0
    for el in ar:
        if el > low and el < high:
            count += 1
    return count

def q2():
    vals = [ 2, 3, 4, 5, 6, 7, 8, 9, 10 ]
    winners = win(vals, 4, 7)
    print(winners)
def present(ar, v):
    """Determine whether v is in array.
    Arguments:
    ar: a list
    v: an integer
    Returns:
    True if v is an element of ar.
    """
    ## Solve without Python library functions
    ## for searching lists
def count_leaves( ar ):
    """Count all the non-list objects in ar.
    Arguments:
        ar: a (nested) list structure
    Returns:
        The number of leaf (non-list) elements in ar
    Ex: count_leaves( [a, [ b, c, [ d, e ], f ] ] = 6
    """