

University of Pittsburgh

CS 0007 Fall 2009 Test 2

There are four questions for a total of 100 points.

Some Python function/method descriptions are given to you.

Otherwise, no other aides or notes are permitted.

We can't help you answer the questions. Otherwise, it gets too noisy, and some people would get more information than others.

Please just follow the instructions, and use your best judgment.

Good luck!

1. (25 points) Write a program that prompts the user for a list of strings and an integer, and then prints a list of all of the elements of the list whose length is greater than the integer. First, complete the following function according to the docstring description. Then, write a main program that calls your function.

```
def longer(lst,x):
    '''Given a list of strings lst, return a new list that contains each
    element of lst whose length is greater than int x. For example,
    longer(['abcd','abc','ab'],2) returns ['abcd','abc']'''

def longer(lst,x)
    new = []
    for s in lst:
        if len(s) > x:
            new.append(s)
    return new
if __name__ == '__main__':
    #Actually, you cannot enter a list like this
    #Nothing was taken off if you tried to fix that
    x = raw_input('Enter a list of strings ')
    y = int(raw_input('Enter an integer'))
    print longer(x,y)
```

2. (25 points) Complete the following function according to its docstring description.

```
def no_duplicates(l):
    '''Return True if list l does *not* contain adjacent elements with the same
    value, and False otherwise. For example, no_duplicates([0,1,2,3])
    returns True, and no_duplicates([0,1,2,3,3]) returns False.'''

    # warning! make sure your loop does not cause a "list index out of
    # range" error!

    i = 0
    while i+1 < len(l):
        if l[i] == l[i+1]:
            return False
        i += 1
    return True
```

3. (25 points) Here is a mystery program.

```
def mystery(students,numbers):
    y = []
    t = 0
    i = 0
    while i < len(students):
        y.append(students[i][1])
        print i,y
        i = i + 1
    m = -1
    for n1 in numbers:
        print "n1 is",n1
        for n2 in n1:
            if n2 > m:
                m = n2
                print "m is", m
            if n2 in [25,32,100]:
                print "Found one"
    print "Finally, m is",m
    return y

if __name__ == '__main__':
    people = [["John Smith","1123425",95],["Juan Martinez","1238765",94],["Sally Jones","1010101",90]]
    measurements = [[25,32,67],[33,21,67,76],[10,25]]
    x = mystery(people,measurements)
    print x
```

What is the output of the program? That is, exactly what lines are printed in what order?

```
%0 ['1123425']
%1 ['1123425', '1238765']
%2 ['1123425', '1238765', '3324567']
%n1 is [25, 32, 67]
%m is 25
%Found one
%m is 32
%Found one
%m is 67
%n1 is [33, 21, 67, 76]
%m is 76
%n1 is [10, 25]
%Found one
```

```
%Finally, m is 76  
%['1123425', '1238765', '3324567']
```

4. (25 points) Complete the following function according to its docstring description.

```
def ave_word_length(word_and_punctuation_instances, punctuation):
    '''word_and_punctuation_instances is a list of strings; punctuation is
    a list of punctuation characters. First, create a new list that is the
    vocabulary of word_list, that is, all the distinct elements of
    word_and_punctuation_instances that are not in the punctuation list.
    Then, calculate and return the average length of the words in that
    list.'''

    vocabulary = []
    for word in word_and_punctuation_instances:
        if not word in vocabulary and not word in punctuation:
            vocabulary.append(word)
    total = 0.0
    for w in vocabulary:
        total = total + len(w)
    return total/len(vocabulary)
```

Help on some functions and methods. You will not use all of these!

`raw_input([prompt]) -> string`

Read a string from standard input.

`len(...)`

`len(object) -> integer`

Return the number of items of a sequence or mapping.

***Some List Methods:**

`append(...)`

`L.append(object) -- append object to end`

`extend(...)`

`L.extend(iterable) -- extend list by appending elements from the iterable`

`remove(...)`

`L.remove(value) -- remove first occurrence of value`

`count(...)`

`L.count(value) -> integer -- return number of occurrences of value`

`insert(...)`

`L.insert(index, object) -- insert object before index`

Some String Methods

`split(...)`

`S.split([sep [,maxsplit]]) -> list of strings`

Return a list of the words in the string S, using sep as the delimiter string. If maxsplit is given, at most maxsplit splits are done. If sep is not specified or is None, any whitespace string is a separator.

`count(...)`

`S.count(sub[, start[, end]]) -> int`

Return the number of non-overlapping occurrences of substring sub in string S[start:end]. Optional arguments start and end are interpreted as in slice notation.

`find(...)`

`S.find(sub [,start [,end]]) -> int`

Return the lowest index in S where substring sub is found, such that sub is contained within s[start:end]. Optional arguments start and end are interpreted as in slice notation. Return -1 on failure.