# Take Home Final Exam

CSC 401 **Name**: **Jasmine Dumas**

## Preliminaries

- The exam is open book and open notes.
- Do not use the internet, beyond the resources mentioned in class (e.g. d2l, Python reference pages at python.org).
- This exam requires individual work, do not talk to anybody about the exam solutions (except the instructor).
- Read the questions **carefully** and **completely** before answering.
- Include screenshots of both your program and of test-runs.
- Good luck.

## Submission

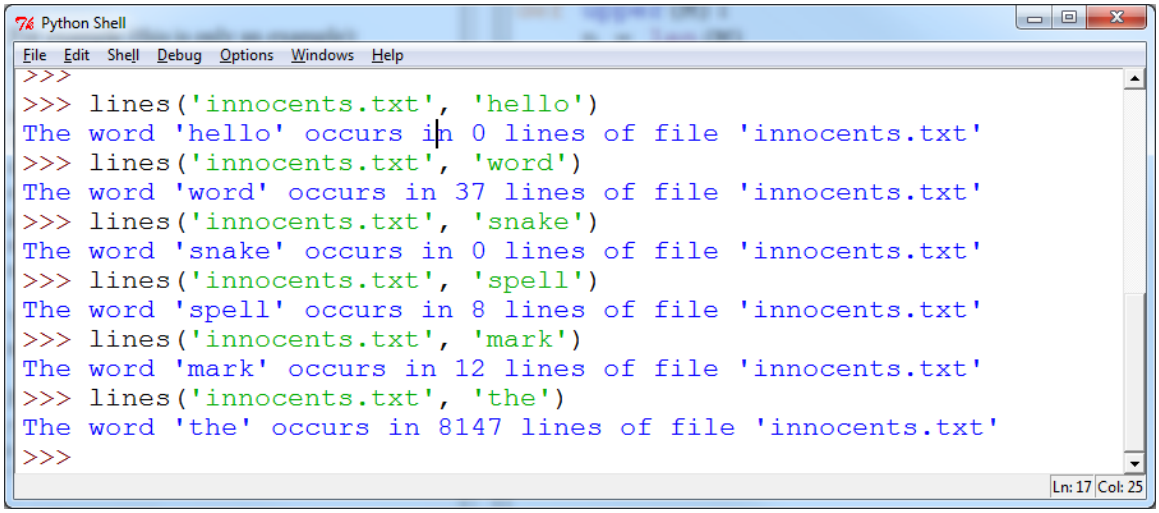
- **Copy all answers into this file**, including copies of the code and screenshots of test-runs (it's ok if doing so changes the page breaks).
- Do not submit Python files.
- There is a drop-box for the final on https://d2l.depaul.edu.

1. [Counting the lines, 25pt]

You want to see how frequent certain words are in a given text. Instead of counting their total number of occurrences, you want to count the number of lines they occur in (so if a word occurs multiple times in the same line, the line still only counts once). Write a program **line(filename, word)** that writes a message of the form

The word … occurs in … lines of file … .

See below for some test-runs with the file 'innocents.txt' included with the exam.




*Hint*: to transform a line into a list of words, simply use **.split()**, don't worry about cleaning the text of punctuation, etc. (so the count won't be entirely accurate).

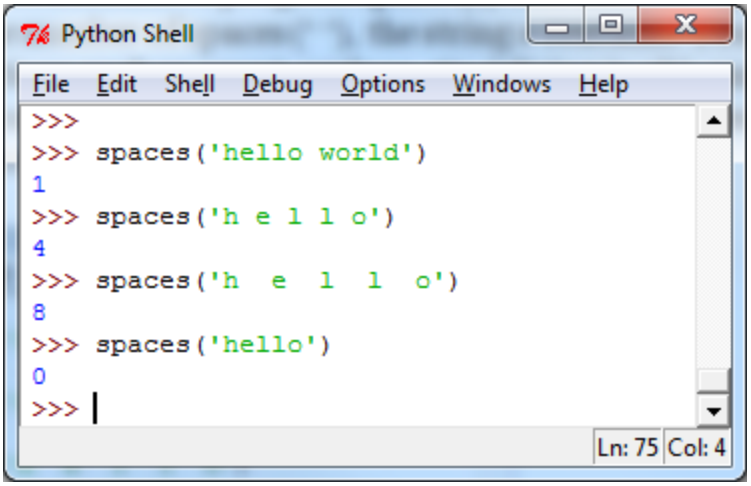
## Program:

```
def line(filename, word):
    infile = open(filename, 'r')  # open file
    content = infile.read()  # reads in as a string
    infile.close()   # close file
    lst = []   # empty list
    new = content.split() # splits the string into lists of lines
    lst2 = []

    for lin in new:
        if word in lin:
            lst.append(lin.index(word))
            if lin == word:  # retrieves the exact match of the word
                x = "got it"
                lst2.append(x)



    print("The word '{}' occurs in {} lines of file 'innocents.txt'".format(word,len(lst2)))
                # len() will count all of the empty list contents



print(line('innocents.txt', 'spell'))
```

## Test runs:

```
Python 3.4.3 Shell
>>> ================================ RESTART ==============
==================
>>>
The word 'spell' occurs in 8 lines of file 'innocents.txt'
None
>>> line('innocents.txt', 'hello')
The word 'hello' occurs in 0 lines of file 'innocents.txt'
>>> line('innocents.txt', 'word')
The word 'word' occurs in 37 lines of file 'innocents.txt'
>>> line('innocents.txt', 'snake')
The word 'snake' occurs in 0 lines of file 'innocents.txt'
>>> line('innocents.txt', 'spell')
The word 'spell' occurs in 8 lines of file 'innocents.txt'
>>> line('innocents.txt', 'mark')
The word 'mark' occurs in 12 lines of file 'innocents.txt'
>>> line('innocents.txt', 'the')
The word 'the' occurs in 11292 lines of file 'innocents.txt
'
                                                Ln: 1225 Col: 0
```

2. [Recursion, 25pt] Write a recursive program **spaces(s)** that takes as input a string s and returns the number of blank spaces (that is, ' ') the string s contains. You may not use string functions such as count, replace, etc. (slicing is ok), and loops and global variables are not allowed. Your solution has to calculate the number of spaces using recursion. Here are some test-runs:

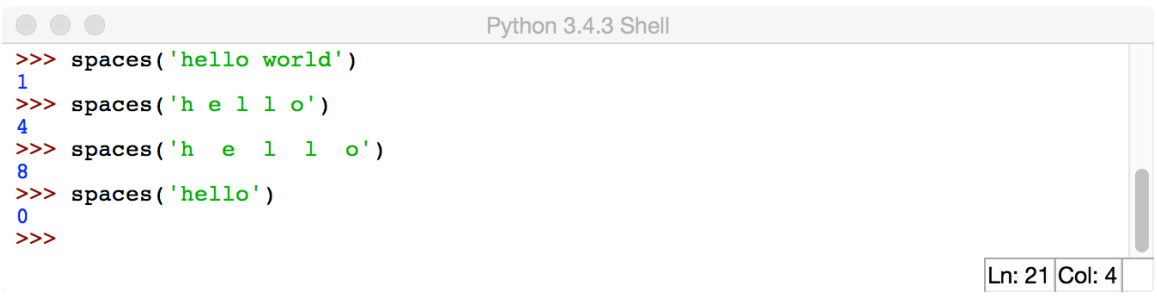



*Hint*: how do you do it by hand if you do it one letter at a time? And what's the base case?

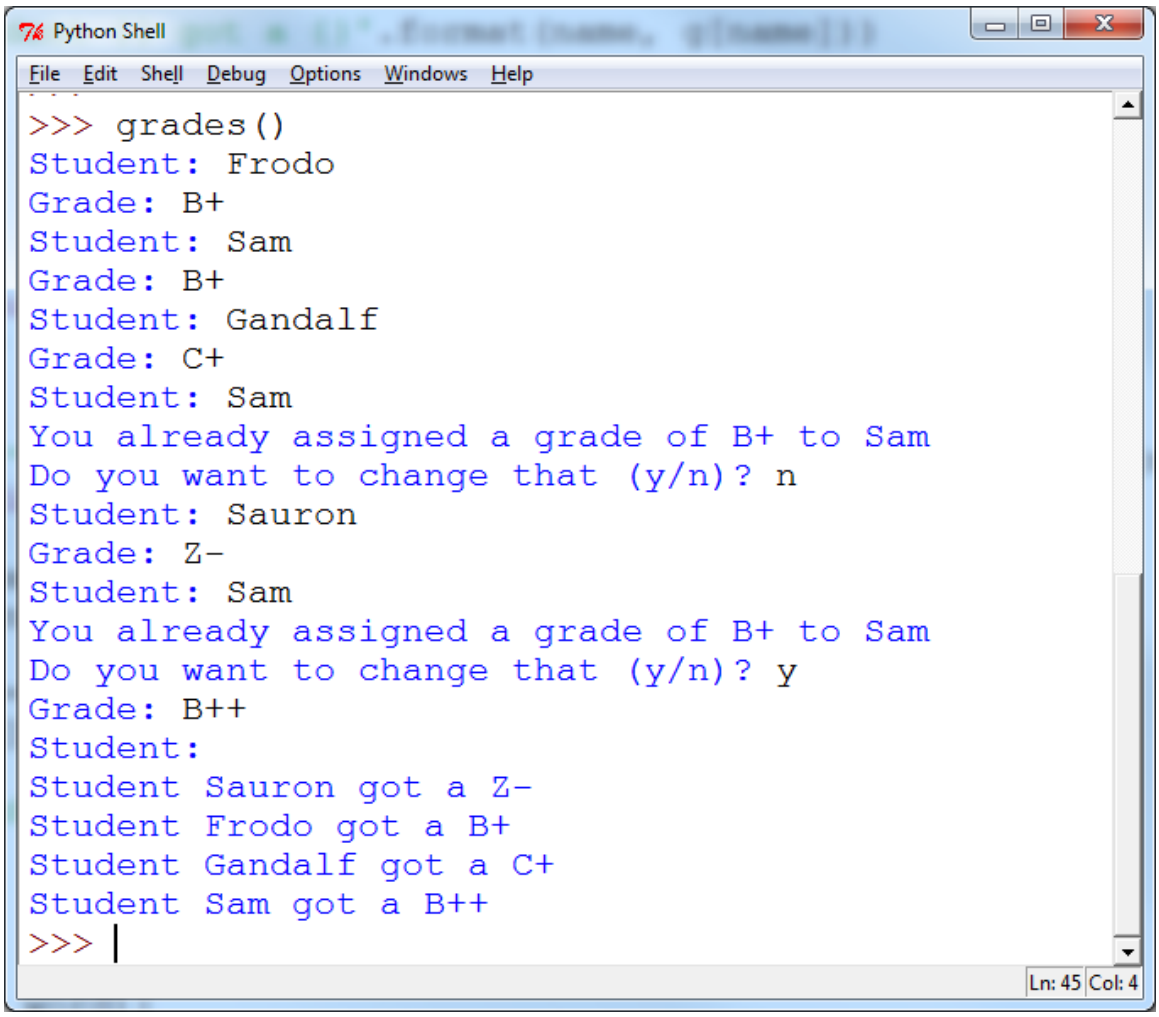
## Program:

```
def spaces(s):
    length = len(s)

    if length == 1: # base case
        return 0
    else:
        return (1 if s[0] == " " else 0) + spaces(s[1:])

s = "h e l l o"
print("recursive “, spaces(s))
```

## Test runs:



3. [Grading, 25pt] While you are grading your class you realize your life would be simpler with a program that allows you to enter students' grades as you grade their homework. Write a program **grades()** that keeps prompting the user for a student name, and the grade of that student. If a user does not enter a name (just hitting return), the program displays all grades that have been entered. If you enter a student's name for the second time, the program should tell you that the student has already been graded (and what their grade was), and ask you whether you want to enter a new grade. If so, the user is allowed to enter the new grade, otherwise, the program prompts for the next student without changing the student's grade. See sample run below.



## Program

```
def grades():
    for student, grade in classlst.items():
        print("Student {} got a {}".format(student,classlst[student]))

student = input("Student: ")
grade = input("Grade: ")
classlst = {}

while student != '':
    classlst[student] = grade
    student = input("Student: ")

    if student not in classlst:
        grade = input("Grade: ")

    if student in classlst:
        print("You already assigned a grade of {} to {}".format(classlst[student], student))
        ask = input("Do you want to change that (y/n)? ")
        if ask == "y":
            grade = input("Grade: ")
            classlst[student] = grade
        else:
            student = input("Student: ")
            if student == '':
                print(grades())
            else:
                grade = input("Grade: ")

    if student == '' or grade == '':
        print(grades())
```

## Test runs:

```
Python 3.4.3 Shell
ESTART ================================
=
>>>
Student: Frodo
Grade: B+
Student: Sam
Grade: B+
Student: Gandalf
Grade: C+
Student: Sam
You already assigned a grade of B+ to
Sam
Do you want to change that (y/n)? n
Student: Sauron
Grade: Z-
Student: Sam
You already assigned a grade of B+ to
Sam
Do you want to change that (y/n)? y
Grade: B++
Student:
Grade:
Student Sauron got a Z-
Student Gandalf got a C+
Student Sam got a B++
Student Frodo got a B+
None
```

Ln: 618 Col: 16

4. [Classes, 25pt] Develop a class BankAccount that supports these methods:

- __init__() : Initializes the bank account balance to the value of the input argument or to 0 if no input argument is given
- withdraw() : Takes an amount as given in the parameter and withdraws it from the balance
- deposit() : Takes an amount as given in the parameter and adds it to the balance
- printBalance() : Prints the balance on the account

Try out the various methods as in the example below:

```
*Python 3.4.2 Shell*
File  Edit  Shell  Debug  Options  Windows  Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22
:15:05) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for mor
e information.
>>> ================================ RESTART =====
==========================
>>>
>>> savings = BankAccount()
>>> savings.printBalance()
The balance is $0.00.
>>> savings.deposit(75)
>>> savings.printBalance()
The balance is $75.00.
>>> savings.withdraw(30)
>>> savings.printBalance()
The balance is $45.00.
>>> checking = BankAccount(100)
>>> checking.printBalance()
The balance is $100.00.
>>> checking.withdraw(64)
>>> checking.printBalance()
The balance is $36.00.
>>> checking.deposit(212)
>>> checking.printBalance()
The balance is $248.00.
>>>
Ln: 21 Col: 27
```

## Program:

```
class BankAccount:

    def __init__(self,initial = 0.00):
        self.s = float(initial)

    def withdraw(self, moneyOut):
        self.s = float(self.s - moneyOut)

    def deposit(self, moneyIn):
        self.s = float(self.s + moneyIn)

    def printBalance(self):
        print("The balance is $", self.s)
```

## Test runs:

Python 3.4.3 Shell

```
>>> savings = BankAccount()
>>> savings.printBalance()
The balance is $ 0.0
>>> savings.deposit(75)
>>> savings.printBalance()
The balance is $ 75.0
>>> savings.withdraw(30)
>>> savings.printBalance()
The balance is $ 45.0
>>> checking = BankAccount(100)
>>> checking.printBalance()
The balance is $ 100.0
>>> checking.withdraw(64)
>>> checking.printBalance()
The balance is $ 36.0
>>> checking.deposit(212)
>>> checking.printBalance()
The balance is $ 248.0
>>>
```

Ln: 99 Col: 0