1 Python Style Requirements

One of our goals is to write programs that are easily understood by the reader. This becomes particularly important when programs get large. For example, suppose you find a reference to an unfamiliar identifier in a program. You may have to look through many screens of code to find the declaration of that identifier to find out that it is the name of a class. A much better alternative is to use identifier names that tell you something about the identifier. Ideally, you should be able to look at an identifier and determine whether it is the name of a function, variable or class, and the meaning of the identifier. This document gives you style guidelines that we would like you to use in this course. These are typical guidelines used in industry. You will notice a *big* difference in the readability of your program.

The TAs will deduct points off homework assignments if you do not follow the style requirements.

2 Docstrings and defs

- Every program *must* have a docstring at top of file as follows:

```
''' Name: Happy Student (and Partner's name if applicable)
    Date: mm/dd/yy
    Course: CSCI 203; Prof. LectureProf
    Lab Section: Tuesday 8-10 am
    Assignment: Homework number and part
    Integrity statement: I have read and adhered to the Academic
                         Honesty policy as outlined in the CSCI 203
                         "Course Description" document. '''
```

- Every program *must* have at least one function (`def`).

```
''' Name: Happy Student (and Partner's name if applicable)
    Date: mm/dd/yy
    Course: CSCI 203; Prof. LectureProf
    Lab Section: Tuesday 8-10 am
    Assignment: Homework number and part
    Integrity statement: I have read and adhered to the Academic
                         Honesty policy as outlined in the CSCI 203
                         "Course Description" document. '''

def happy():
    ''' The happy function prints a happy message.
        No inputs. '''

    print('This is a happy function!')

# The following line is the first line to be executed,
# i.e., a call of the happy function
happy()
```

- Every function or method *must* have a docstring right after the `def` line that explains what the functions does and describes each of its input arguments.

```
def findDistance(rate, hours):
    ''' findDistance calculates the distance by multiplying
        the rate and the time in hours.
        Inputs:
           rate - a real number in mph
           hours - a real number in hours '''

    distance = rate * hours
    return distance

myRate = float(input('Enter rate in mph: '))
myTime = float(input('Enter the number of hours: '))
print('Distance is', findDistance(myRate, myTime), 'miles')
```

3 Spacing and Comments

- Use **blank lines** to separate logical sections of your program. Leave a blank line after the docstring of a function.

- Use **spaces** *around* = and around operators and *after* commas. For example:

```
def computeBMI(weight, height):
    ''' computeBMI computes the Body Mass Index given
        an individual's weight and height.
        Inputs:
           weight - a number in pounds
           height - a number in inches '''

    bmi = (weight * 4.88) / (height / 12.0) ** 2
    return bmi

myWeight = float(input('Enter your weight in pounds: '))

print('Enter your height in feet and inches')
myFeet = float(input('How many feet? '))
myInches = float(input('How many inches? '))
myHeight = myFeet * 12 + myInches

myBMI = computeBMI(myWeight, myHeight)
print('BMI:', myBMI)

if myBMI <= 18.5:
    print('Underweight')
elif myBMI < 25:
    print('Normal weight')
elif myBMI < 30:
    print('Overweight')
else:
    print('Obseity')
```

- Use **comments** to describe major sections of programming or where something needs to be clarified.

  Use the automatic indenting feature of idle3.2. Indent 4 spaces for each structure.

4 Variable names

A variable name should start with a lower case letter and should describe its purpose.

5 Class Names

Any classes that you define must begin with an uppercase letter. This helps you to distinguish them from variable and function (method) names. For example, a class defining rational numbers might be called `Rational`.

6 Instance Variable Names

Instance variable names begin with a lowercase letter and use embedded uppercase letters for word breaks. Do *not* use underscores in object names. The following are examples of appropriate variable names.

```
thisCopy
nameOfBook
myGrade
numOfStudents
```

7 Constants

Constants are all uppercase letters. For example:

```
DEFAULT = -1
UNDEFINED = -2
```

8 Function and Method Names

Function and method names begin with an lowercase letter and use embedded uppercase letters for word breaks. A function or method name typically starts with a verb since it generally involves some actions. Here are some examples:

```
readCharacter()
refreshScreen()
```

If a method answers a yes/no question and returns a `boolean`, begin its name with `is` or some other form of the verb "to be" so that it reads well. For example, you might define a function called `isEven()` to test if a number is a even number.

```
def isEven(n):
    ''' isEven checks to see if n is even.  It returns True
        if even and False otherwise.
        Input:
            n - an integer number '''

    return n % 2 == 0
```

9 Official Python Style Guide

If you have questions about other programming style issues, please consult the official Python Style Guide by Guido van Rossum and Barry Warsaw at <http://www.python.org/dev/peps/pep-0008/>. This document is well written and worth browsing.