Name_____

CSCI 150
Final Exam
December 16, 2015

There are 10 questions here, each worth 10 points. When you are done with the exam sign the Honor Pledge on the last page. If you are not taking the exam with the rest of the class please write your starting and stopping times on this page as well.

1. I am trying to write a program that will repeatedly ask for a name and then say hi to that person. The program should halt after it does this 100 times, or if it ever gets a blank name. Here is what I have:

```
def main():
        done = False
        for x in range(0, 100):
                name = input( "Who? " )
                if name == "":
                                done = True
                else:
                                print( "Howdy %s!" % name )

main()
```

   This program doesn't halt when I give it a blank name**. Explain why.** You don't need to fix the program; just say why it doesn't stop when I give it a blank name.

   Setting variable *done* to True doesn't get you out of a *for*-loop,

2. **What will the following program print?**

```
def G(n):
        while n > 0:
                print(n, end = "" )
                if n%2 ==  0 and n%3 == 0:
                        print()
                        return
                 n = n-1
        print()

def F( L ):
        s = 0
        for x in L:
                s = s + x
        G(s)

def main():
        L = [3, 1, 5]
        for i in range(0, 3):
                F( L[i:] )

main()
```

9876
6
54321

3. What will the following program print?

```
def main():
        string = ""
        s = "abcd"
        for letter in s:
                string = letter + string + letter
        print(string)

main()
```

dcbaabcd

4. Consider the following program. Function F returns a number that divides into x, main has a loop that prints values and the result of calling F on each value.

```
def F(x):
        # This finds a number that divides into x
        for d in range(2, 20):
                if x%d == 0:
                        return d

def main():
        x = 5
        while x >= 0:
                print( "%d is divisible by %d."%(x, F(x)))
                x = x-1
main()
```

When I run this it prints

5 is divisible by 5.
4 is divisible by 2.
3 is divisible by 3.
2 is divisible by 2.

and then it crashes. The error message refers to the print statement and says:

TypeError: %d format: a number is required, not NoneType

a) **Explain in one English sentence** what this error message means. What is "NoneType"?

The *print* statement in *main*() tried to print the value None as an integer.

b) **Explain what in the program causes this error**. You don't need to fix the program, just say why it crashes.

F(1) doesn't explicitly return anything, so it returns None.

5. Here is a program using classes and subclasses.

```
class Person:
        def __init__(self, name):
                self.name = name
                self.age = 0
        def __str__(self):
                return self.name
class Student( Person ):
        def __init__(self):
                self.year = "Freshman"
        def changeYear(self, year):
                self.year = year
        def __str__(self):
                return "%s is a %s."%(self.name, self.year)
def main():
        p = Person( "Dumbledore" )
        print( p )
        h = Student( "Harry" )
        print( h )
main()
```

The first few lines of main( ) run correctly, but it crashes on the line

h = Student("Harry")

with the error message

TypeError: __init__() takes 1 positional argument but 2 were given

a) **Explain in one sentence why the program crashes**.

The __init__( ) method for class Student takes no argument but *self*.
It is called with a name argument: "Harry".

b) Give a fix for the program so that it runs correctly and prints "Harry is a Freshman." You don't need to copy the program; just write onto the code above.

```
def __init__(self, name):
        Person.__init__(self, name)
        self.year = "Freshman"
```

6. In the following program class Person has an instance variable self.friends that is a list of the Person objects who are friends with self.

```
class Person:
    def __init__(self, name):
        self.name = name
        self.friends = []
    def __str__(self):
        return self.name
    def makeFriend(self, x):
        # x should be a Person
        self.friends.append(x)
        x.friends.append(self)
    def printFriends(self):
        print( "Here are the friends of %s:"%(self.name))
        for x in self.friends:
            print("   ", x)
def main():
    f = Person( "Fred Flinstone" )
    f.makeFriend( "Barney Rubble" )
    f.makeFriend( "Mr. Slate" )
    f.printFriends( )
main()
```

This program crashes on the line `x.friends.append(self)`
called from the line `f.makeFriend( "Barney Rubble" )`.
The error message says

```
AttributeError: 'str' object has no attribute 'friends'
```

a) Explain in one sentence what is wrong.

The makeFriend( ) method needs to be called with an object of class Person rather than a string.

b) **Fix the program**. You can write directly over my code.

In main() the two calls to makeFriend should be

f.makeFriend(Person("Barney Rubble"))

f.makeFriend(Person("Mr. Slate"))

7. You have probably used software systems that require you to create a password that includes certain kinds of characters. Let's simplify this down to requiring passwords to include at least one digit. Write a program that repeatedly asks the user to enter a new password. If the string the user supplies contains a digit, your program should say "Your new passworod is " and then print the string and exit the loop. If the string doesn't contain a digit, your program should say "Bad password" and go around the loop again. If the user gives the empty string your program should say "You have no password." and exit the loop. Here is a typical run of the program, with what the computer prints in bold:

Enter new password: bob

Bad password.

Enter new password: frodoLives

Bad password.

Enter new password: python3

Your new password is 'python3'.

```
def isOk(s):
    for d in "0123456789":
        if d in s:
            return True
    return False

def main():
    done = False
    while not done:
        pword = input( "Enter new password: " )
        if isOk(pword):
            done = True
            print( "Your new password is '%s'."%pword)
        elif pword == "":
            done = True
            print( "You have no password." )
        else:
            print( "Bad password." )

main()
```

8. File “F.txt” is a dictionary file consisting of lines of text with one word per line, such as

```
apple
bob
cat
d
exams
```

Write a program that prints the words in this file, first printing all of the 1-letter words, then all of the 2-letter words, then the 3-letter words, and so forth. For the data above it would print

```
1 letter words:
        a
3 letter words:
        bob
        cat
5 letter words
        apple
        exams
```

```
def main():
    D = {}
    F = open( "F.txt", "r")
    for line in F:
        word = line.strip()
        n = len(word)
        if n in D.keys():
            D[n].append(word)
        else:
            D[n] = [word]
    lengths = list(D.keys())
    lengths.sort()
    for n in lengths:
        print( "%d letter words:"%n )
        for word in D[n]:
            print( "    ", word )
main()
```

9. Write a **recursive** function HasDuplicates(L) that takes a list L and returns True if at least one of the entries in L appears more than once, and False if the elements are all unique. For example, HasDuplicates([1, 3, 2, 1,4]) returns True and HasDuplicates([1, 3, 2, 7, 4]) returns False. If you don't see how to do this recursively you will get some credit for doing it iteratively (with a loop).

```
def HasDuplicates( L ):
    if len(L) <= 1:
        return False
    elif L[0] in L[1:]:
        return True
    else:
        return HasDuplicates( L[1:] )
```

10. **Write a class Dice** that represents a pair of dice, each of which randomly chooses a value between 1 and 6. The constructor for Dice needs no arguments. Your class should have the following methods. These methods need no arguments (except self)

    roll( ) gives random values between 1 and 6 to both dice
    values( ) returns a tuple with the two dice values
    total( ) returns the sum of the values of the two dice
    doubles( ) returns True if both dice have the same value

For example, we might have an application program with the code

```
d = Dice( )
d.roll( )
if d.total( ) == 7:
        print( "Yippee, I rolled a 7!" )
```

You only need to write class Dice.

```
class Dice:
    def __init__(self):
        self.a = 0
        self.b = 0

    def roll(self):
        self.a = random.randint(1, 6)
        self.b = random.randint(1, 6)

    def values(self):
        return (self.a, self.b)

    def total(self):
        return self.a+self.b

    def doubles(self):
        if self.a == self.b:
            return True
        else:
            return False
```