Lab Four A202/A598 Spring 2007

Text: How to think like a computer scientist (learning with Python[1]).

Note: as discussed, asking these questions in class (lab or lecture) or working on them together is perfectly fine (as long as when you write up the answer you do it by yourself, in your own words).

Reading Assignment 12: Chapter 12 (Classes and objects).

Questions: Explain what happens in the code below:

```
class Potato:
    pass

a = Potato()
a.x = 2

b = Potato()
b.y = 3

print a.x

print b.y
```

Are objects like dictionaries?

Are the two instances in the code above created equal?

Are they still equal by the time the code is finished? Is this good or bad (or irrelevant?). Consider the following code:

```
class Potato:
    pass

a = Potato()
a.x = 2

def fun(n):
    if n == 0:
        return n
    else:
        return n + fun(n-1)

a.fun = fun

print a.fun(10)
```

[1] http://www.cs.indiana.edu/classes/a202-dger/fall2006/a201-text/thinkCSpy.pdf

Explain what happens in it. Can you achieve the same using a dictionary?

```
a = {}
a['x'] = 2

def fun(n):
    if n == 0:
        return n
    else:
        return n + fun(n-1)

a['fun'] = fun

print a['fun'](10)
```

Isn't that ugly[2]? Please explain. What is the solution?

Consider the following code and explain how it relates to the code above:

```
class Potato:
    def __init__(self):
        self.x = 0
    def fun(self, n):
        if n == 0:
            return 0
        else:
            return n + self.fun(n-1)

a = Potato()
b = Potato()
c = Potato()

print a.fun(10), b.fun(10), c.fun(10)
```

Define: instances, objects, instantiation. Can you implement the functionality of class Point using a dictionary instead? Is it better or worse?

```
>>> a = {}
>>> id(a)
13590576
>>> a["one"] = "sdhf"
>>> a["two"] = "d9f6"
>>> id(a)
13590576
>>> a["three"] = "sdvc"
>>> a
{'three': 'sdvc', 'two': 'd9f6', 'one': 'sdhf'}
>>>
```

Can a dictionary be taught to report its `id`?

---

[2] It is in fact a combination of *powerful* and *ugly*. *Powerful* sometimes means *beautiful*. So there.

In other words can you implement this behavior with dictionaries?

```
class One:
    def whoami(self):
        return id(self)

a = One()
b = One()
print id(a) == a.whoami()
print id(b) == b.whoami()
```

Is this close in spirit to your solution:

```
>>> a = {}
>>> a["self"] = id(a)
>>> a
{'self': 13590576}
>>> id(a)
13590576
>>>
```

Define a function that takes two points as arguments and reports the distance between them. Points are created and initialized outside, they each contain an `x` and a `y` coordinate.

Does shallow equality imply deep equality? Is the reverse true?

Define a function that takes two rectangle objects (whatever your design choice is for rectangles) and decides if they overlap or not. Same problem for circles.

Define a class Circle as a Point (for the center) and a float (the radius).

Define a class Line as two Points.

Define a class Triangle as being three Lines coming out of three Points.
Define a function that finds the midpoint[3] of a segment[4].

Define a Fraction[5] as a pair of numbers.

Define a class Robot that has a direction and can move forward. Then create such an object and simulate a random walk for it[6]. When the random walk is over report the distance from the robot's location and the location where the random walk started.

Write a function that can translate a circle, rectangle, triangle object.

---

[3] That is, it returns a Point.
[4] A segment is the same as a Line in this context.
[5] Define a gcd function that finds the greatest common divisor of two numbers.
[6] There's more than one way to do it, so implement the most appealing of these.

Describe the difference between shallow and deep copying.

Reading Assignment 13: Chapter 13 (Classes and functions).

Questions: Solve the exercises on page 137 bottom, 138 top. Give an example[7] of a pure function that adds two Fractions, producing a new one. Make sure that your function produces results in their lowest terms. Write a modifier function that changes a Fraction object by adding a second Fraction to it. Make sure the result is in its lowest terms.

Define functional programming style. What does the book have to say about it?

Discuss prototype development vs. planned development.

Explain why multiplying a digit n by 9 gives a two-digit number where the first digit is `n-1` and the second one is `10-n` (see comment on page 142, the section on algorithms).

Design an algorithm to calculate the square root of a positive number.

Reading Assignment 14: Chapter 14 (Classes and methods).

Questions: Explain what the following code does:

```
class One:
    def fun(x):
        return x

a = One()
print a.fun()
```

What is (by convention) the first argument to an instance method?

Define a class of objects that implement the game of Nim.

Here's how my game runs:

```
>>> game = Nim(50)
>>> game.move()
How many? 24
The height is now: 26
The computer moves.
>>> game.move()
The computer chooses 7
The height is now: 19
```

[7] That is, define.

```
The user moves.
>>> game.move()
How many? 9
The height is now: 10
The computer moves.
>>> game.move()
The computer chooses 3
The height is now: 7
The user moves.
>>> game.move()
How many? 1
The height is now: 6
The computer moves.
>>> game.move()
The computer chooses 2
The height is now: 4
The user moves.
>>> game.move()
How many? 2
The height is now: 2
The computer moves.
>>> game.move()
The computer chooses 1
The height is now: 1
The user moves.
>>> game.move()
user has lost the game.
New height for new game:24
>>> game.report()
The height is now: 24
The user moves.
>>>
```

Define a class of objects that implements a Matrix type:

```
>>> m = Matrix(3, 5)
>>> m.show()
7 6 6 9 5
7 1 1 8 2
2 7 9 2 6
>>> m = Matrix(4, 2)
>>> m.show()
7 2
9 8
4 1
2 9
>>> m = Matrix(3, 3)
>>> m.show()
2 3 4
6 2 0
2 3 7
>>>
```

5. Design and implement a class called **Coin** that represents a coin that can be flipped, showing either heads or tails. Create a driver class, called **CoinFlip**, whose main method flips a coin 100 times to determine how many times each side comes up, and test your class:

```
Coin coin = new Coin();
for (int i = 0; i < 100; i++)
  coin.flip();
coin.report();
```

Use object-oriented programming to solve the following problems:

- Write a program to solve the `Point` problem. Define a class of objects called `Point` (in the two-dimensional plane). A `Point` is thus a pair of two coordinates (`x` and `y`). Every `Point` should be able to calculate its distance to any other `Point` once the second point is specified.

- Define a class of objects called `Line`. Every `Line` is a pair of two `Point`s. A `Point` is a pair of two numbers (the `Line`s are also in the plane in these exercises). `Point`s should be able to determine their distance to other `Point`s (see above). Lines are created by passing two `Point`s to the `Line` constructor. A `Line` object must be able to report its `length`, which is the distance between its two end points. Make `length` a method and write a test program in which you create a few `Line`s and ask them to report their lengths.

- Define a class of objects called `Triangle`. A `Triangle` should be a set of three `Line`s (which for the purpose of this problem should be a very adequate representation). However, a `Triangle` is created by specifying three `Points` (which are located in the plane as discussed above). Using Heron's formula every `Triangle` should be able to calculate and report its area. (If the three `Point`s are collinear the `Triangle` is extremely flat, its area is 0 (zero), and that should be acceptable.)

- Define a class of objects called `Clock`. An object of type `Clock` stores time (hours and minutes) in military time format, in two instance variables of type `int`. Objects of type `Clock` have two instance methods: `report` which is simply returning the time, and `tick` which advances the clock by one minute. The constructor for class `Clock` takes one argument, a `String` that represents the time the clock is originally set to. Write a test program too, that illustrates how your objects are working (tick the clock 10-20 minutes and show it).

- Define a class of objects called `Player` that could be used in a Paper Scissors Rock game. Such a `Player` object should have a method, called `makeGuess` that could be used to generate (randomly) one of the following guesses: `"paper"`, `"rock"`, or `"scissors"`. The guess made by this method should be stored in an

instance variable as well (a `String`, called `guess`). Another method of class `Player` should be able to compare the choice of the player it belongs to with the choice of any other `Player` and determine if the first player's guess is stronger than the second player's guess. Call this method `strongerThan` and make it return `true` or `false`. A `report` method should return the `guess` instance variable for printing purposes.

- Design an `Elevator` class for objects of this type that go up and down in a building with 100 floors.
- Nobody bounces like Tigger! Years of research have finally revealed the special mechanism of Tigger's bouncy step. You are to design a `Tigger` class that implements this unique movement, which I will describe below. A `Tigger` always starts in a random point (with coordinates `x` and `y`). When it decides to `bounce` a `Tigger` changes its `x` and `y` by the following rule(s): `x` becomes the sum of the squares of its digits and `y` becomes the sum of the squares of its digits. Example: if `x` is `37`, then one `bounce` turns `x` into `32 + 72 (= 58)`. Both `x` and `y` change by this rule. And the `bounce` goes on (as expected).

Describe optional arguments. Solve the first exercise on page 150.

Clearly identify the initialization methods in the list of problems provided above.

Define a class Fraction whose initialization method ensures that the fraction created is already in its lowest terms. Teach these fractions to add, subtract. multiply and divide. Add a `__str__()` function that prints Fractions nicely (`-2 / 3` or some such thing).

Overload `+`, `-`, `*` and `/` for Fractions.

Define polymorphism. Give an example.