

POLYTECHNIC COMPUTER SCIENCE - SPRING 2012
PROGRAMMING PROJECT #4
INDEPENDENT PROJECT #2

ASSIGNMENT OVERVIEW

In this assignment you'll be creating a program called *independentproject2.py*, the subject of which you will choose from the options listed below.

This assignment is worth 100 points and is due on the crashwhite.polytechnic.org server at midnight (00:00:00) on the date given in class.

BACKGROUND

Now that we've learned a little about how to write functions and plot graphics using Python, apply what you've learned to one of the following programs. The programs range in difficulty, and while only some of them require graphics, all of them should be written using functions to a) organize your work, and b) avoid repeating code.

PROGRAM SPECIFICATIONS

Create any one of the following programs:

a. **RandomWalker**

A blindfolded person walks in random directions, and eventually (perhaps?) returns to his starting position. Model this situation using a pixel initially placed in the middle of a graphic screen.

b. **Spirograph**

A hypocycloid is a certain type of periodic mathematical function. Write a program that accepts parameters from the user and uses them to create a hypocycloid on the screen using Pygame graphics.

c. **MontyHall**

The "Monty Hall" problem is this:

In the TV game show "Let's Make a Deal," master of ceremonies Monty Hall invites contestants to choose door #1, #2, or #3, behind one of which is a prize car. Behind the other two doors is typically a goat. After the contestant picks a door (say Door #1 for our example), Monty Hall will open one of the other doors (say Door #2) to reveal one of the goats. "Now that you've seen that, do you want to change your mind? You can still pick Door #3 if you choose!"

Write a text or graphics-based program to model this game.

d. **Mandlebrot**

A "Mandlebrot plot" is a type of fractal, a recursive function that exhibits a variety of features depending on the scale of view. Do a search for *mandlebrot* and *fractal*, and find a function that you can use to create and display a mandlebrot plot on the screen using Pygame graphics.

e. **Life**

John Conway's game of *Life* is a simple model that demonstrates how simple rules can lead to complex patterns or behaviors. Research the game of *Life*, identify the rules that govern the game, and write a text-based program that allows the user to "play" the game.

DELIVERABLES

independentproject2-lastnamefirstinitial.py

You should keep a copy of this file in your home folder on the server. To submit your assignment for grading, copy your file to the instructor's Public/Dropbox folder at crashwhite.polytechnic.org before the deadline.

1. Please be sure to use the specified file-naming convention.
2. Save a copy of your file on your hard drive, flash drive, etc..
3. Your grade will be based on the file you upload to the instructor's Public/Dropbox folder.

ASSIGNMENT NOTES

- For the RandomWalker, the simulation should allow the walker to step in any direction. You can generate a random direction as an angle off of the x-axis.

```
angle = random() * 2 * math.pi
# The new x and y positions are then given by these formulas:
x = x + cos(angle)
y = y + sin(angle)
```

Not just any x and y values can be plotted, however—you can only plot integer values.

- For the Spirograph program, you'll need to use the following parametric equations for a hypocycloid:

```
x = (a-b)*cos(theta) + b*cos(((a-b)*theta)/b)
y = (a-b)*sin(theta) - b*sin(((a-b)*theta)/b)
```

where

- a is the radius of the fixed circle,
- b is the radius of the rolling circle, and
- θ is the angle between the x-axis and the line connecting the two circles' centers, in degrees.

Begin at $\theta = 0$ and increase it by some value in a loop. Calculate successive x, y points and plot them on the screen to display the hypocycloid.

- The MontyHall program is actually at the heart of a basic statistics question: does switching doors give you a better chance of winning the car? The program you write isn't required to keep statistics on the probability of success for a user, although an advanced version of the program might include that capability.

It's possible to write this program so that it only displays text prompts. If you want some practice with graphics, write your program so that it visually displays three doors, and allows the user to click on the doors that they want to open.

- For the Mandelbrot program, you need to identify functions that you can plot pixel by pixel on the computer screen. You can start here—<http://mathworld.wolfram.com/MandelbrotSet.html>—but there are other references as well. If you find that you need additional, more concrete, hints on getting started, see <http://warp.povusers.org/Mandelbrot/>.
- For the Life program, find information at http://en.wikipedia.org/wiki/Conway's_Game_of_Life. Part of the challenge of this program is reasonable way to represent cells on a screen: single pixels are extremely small, and depending on the resolution of the screen, may be difficult to observe. By using an 5-by-5 collection of pixels to represent one “cell” in the game, you can improve visibility (at the

expense of making your program slightly more complicated), or you can write a text-based version of the game that is more easily viewed (see Sample Interactions).

GETTING STARTED

1. These programs are starting to become more complex, so it's more important than ever to try to organize your thoughts *before* you start coding. With paper and pencil, and perhaps in collaboration with a partner, identify what the main components are that you'll need to include in your program.
2. Sketch out the basic flow of your program using a flowchart, and write some pseudocode that you can use to begin implementing those main components.
3. You can use commented pseudocode to start coding, of course. You may find it convenient to include definitions for functions that you haven't yet written. If so, be sure to include a **pass** statement as the body for that function so that your program will compile without throwing up an error.

```
def process_data():  
    """This function will process data, once we've got it all entered.  
    This function hasn't yet been coded, though."""  
    pass
```

4. Write smaller functions to test your code as you go. The vast majority of programs don't work as anticipated the first time one writes them; programming a problem of any reasonable difficulty is simply too challenging for us to do in one try.
5. Don't panic if you have to throw out old versions of functions. That's part of the learning process.
6. Keep backup copies of your programs on the *crashwhite.polytechnic.org* server.
7. Fix old problems before adding new components. You'll repeatedly run through this edit-run, edit-run process to find bugs and fix them. Consider keeping several versions of your program as you work on editing it and improving it. Although there are sophisticated Version Control Systems that help keep track of multiple versions of programs in a large scale project, for our purposes you can get away with simply saving successive versions of your program with different names, perhaps with the date as part of the filename: *blackjack-whiter-2012-03-13.py*, and *blackjack-whiter-2012-03-14.py*, etc.

QUESTIONS FOR YOU TO CONSIDER (NOT HAND IN)

1. What is the difference between *local* and *global* scope for variables? What are the advantages of *not* using global variables in a function?
2. In Python, which data types are *mutable* and which are *immutable*?
3. What is the difference between "assigning by value" and "assigning by reference"?

