|  |  |
| --- | --- |
| Lecture 5 Example 1 | Introduction to Python - Lecture 5 |
| Diffusion of a particle | In this example we will be building on what was discussed in the lecture notes to “model” the diffusion of a particle in a 2D space. There won’t be much new programming introduced here, it will be using what we have done so far and applying it to the problem at hand. Movement of a particle In the first part of this example, we are going to be imagining a particle as existing in some 2-dimensional space. In reality, a particle can move anywhere in this space, but we are going to assume for simplicity that our particle is fixed to existing on a grid as depicted below.  We can furthermore assume that this particle can only move up, down, left and right.  **Task – start with a particle at some point (say at the origin (0,0)) and generate random numbers (say 25 of them) between 0-3, as we did previously, and move the particle according to those random numbers. Output the x/y position to the shell as the particle moves.** |

**Some points to consider:**

* Can you write a function to update the particle’s positions given a random number as an argument?
* Can you use an array anywhere?
* There are several ways to achieve this. Think about your algorithm before you start coding.

**Functions that return more than one argument**

Recall, that when we did curve fitting we saw slightly strange syntax whereby two arguments (popt and pcov). Something along the lines of:

popt, pcov = curve\_fit(func, xdata, ydata)

This simply means that the function curve\_fit returns two things (popt and pcov). This is fairly straightforward to do with functions. Consider the function below:

def func\_multi\_ret(a,b):

return a+b,a\*b,a+b<a\*b

To assign variables to the 3 returned values you can do (e.g. for 1.1 and 1.9):

sumnum,prodnum,compbool=func\_multi\_ret(1.1,1.9)

This assigns the first results (a+b) to *sumnum*, the second (a\*b) to *prodnum* and the Boolean (a+b<a\*b) to *compbool*.

# Periodic Boundary Conditions

Have you ever played asteroids on the Atari? <https://www.youtube.com/watch?v=WYSupJ5r2zo>



No, I guess not, I’m old.

In that game, if you went off the left hand side of the screen you reappeared on the right. Or, if you went off the bottom, you reappeared at the top. This is an example of what is known as boundary conditions. An alternative would be that things bounced off the sides or disappeared. Reappearing at the other side is what is often called *periodic boundary conditions*.

**Task: modify your code from above to apply periodic boundary conditions. To do this, you will need to determine the maximum/minimum grid points the particle can access.**

See my solution on Blackboard *Lecture5Example1\_Periodic\_Visualise.py.*

# Simple Visualisation

As a final task, you can combine what you have done above with matplotlib to visually see where the particle is in a graph. One (very crude) way to do this is to create a plot that gets updated within a loop. Each time the loop executes we have to clear the current plot, otherwise the previous points will still be visible. The main thing we have to do is to allow the program to continue after the *plt.show()* command, which we can do by calling *plt.show(block=False)*, which will achieve this.

plt.axis([-5,5,-5,5])

plt.show(block=False)

plt.pause(0.1)

plt.clf()

The final two commands here will pause the graph for some short period of time so that the graph can be viewed and then clear (*plt.pause(0.1)* and *plt.clf()*, respectively). Note, that the 0.1 is 0.1 seconds. This may need to be adjusted.

Take a look at my solution on Blackboard *Lecture5Example1\_Periodic\_Visualise.py* if you get stuck.