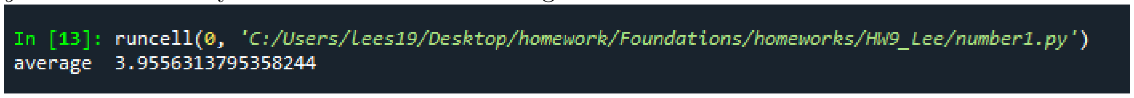
Foundations of Applied Math
HW #9
Due Friday, Nov. 13 by 7 am

**Reminder** You need to turn in a .zipped folder that contains your .tex file, your image files, your python files and the tex file must compile. Rename the .tex file: HW9_YourLastName.tex and call the folder which you will compress: HW9_YourLastName

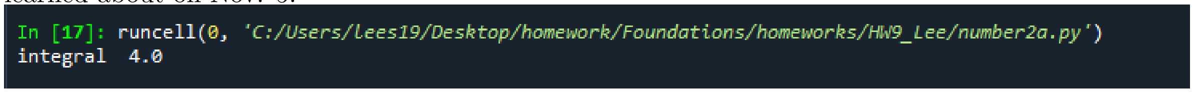
1. Given $y = 4x$, find an approximation to the average value of $y$ for $x \in [0, 2]$. Please do this by evaluating $y$ at 1000 randomly chosen $x$ values and average the results.



```
In [13]: runcell(0, 'C:/Users/lees19/Desktop/homework/Foundations/homeworks/HW9_Lee/number1.py')
average  3.9556313795358244
```

number1.py is included.

2. a) Compute $\frac{1}{2-0}\int_0^2 4x dx$- recall we have a python command to compute a definite integral that we learned about on Nov. 5.



```
In [17]: runcell(0, 'C:/Users/lees19/Desktop/homework/Foundations/homeworks/HW9_Lee/number2a.py')
integral  4.0
```

number2a.py is included.

b) How does this number relate to your answer that you found in the previous problem? Why would this be the case?
The numbers are very similar. This is because we are choosing random points on the line and averaging them so we expect the average of our random points to be around 4.

3. Explain every line of this program at the link below is doing and what this program is doing. Feel free to change items to smaller parameters to help you understand what each line is doing:

https://scipy-lectures.org/intro/numpy/auto_examples/plot_randomwalk.html#sphx-glr-intro-numpy-auto-examples-plot-randomwalk-py
First, we import numpy and matplotlib.pyplot. Then we make variables $n_stories$, $t_max$, t and set them as 1000, 200, and an array from 0 to 199. Then we will create steps which is going to be a matrix which is $(1000, 200)$ and filled with steps of size 1 or $-1$. Then we will use that matrix to create positions, which will count up each step for each of our realizations. $Sq_distance$ will square everything in positions and $mean_sq_distance$ will average all of the square values by column so we get a $(200, )$ matrix which has the average square distance for amount of steps taken. plt.figure(figsize = (4, 3)) will set the size of the plot that shows up, plt.plot will plot the distance from the origin as a function of time and compare it with the theoretical result. plt.xlabel and plt.ylabel will create $x$ and $y$ labels respectively. $plt.tight_layout()$ seems to fit the correct axes together on the plot. plot.show() shows the plot.

4. Suppose we wish to write a Monte Carlo program that returns an estimate of the volume under $z = 25 - x^2 - y^2$ and above the $xy$-plane.

a) Find the equation of the curve of intersection between $z = 25 - x^2 - y^2$ and the $xy$-plane. (Note the $xy-$plane is where $z = 0$.)
Taking $z = 0$, we find that

$$0 = 25 - x^2 - y^2$$
$$x^2 + y^2 = 25$$

So the intersection between $z = 25 - x^2 - y^2$ and the xy-plane is is the circle $x^2 + y^2 = 25$.

b) Calculate the volume of the smallest rectangular prism that contains this solid.
Since the intersection is a circle of radius five and the maximum height above the positive xy-plane the smallest box will be 10 by 10 by 25 which has a volume of 2500.

c) Write a python program that outputs an estimate of the volume of this solid. Do this in two ways:

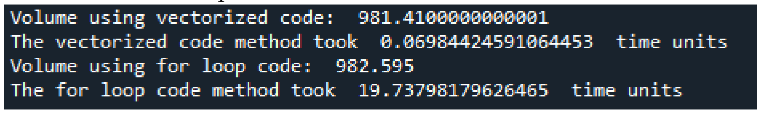
i. Using vectorized code like we did Week 9- please time the code.

ii. Using a loop $N$ times as we spoke about in class and as the book in 5.1 uses- please time the code.

Note. To time the code the following are the commands you will need.

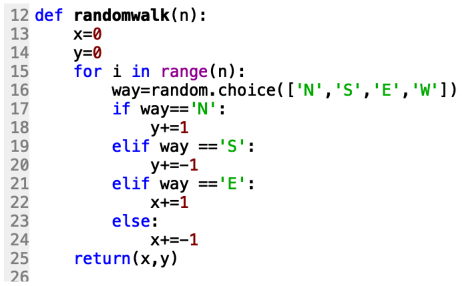
```
import time

t0 = time.time()
#For i. have this right before the np.random.uniform(a,b,N) assignment statement
#For ii have this right before the loop

t1=time.time() #have this at the end after the computation.
print("The total time using vectorized code this took was: ",t1-t0)
#and write "using a loop" instead if it's for ii.
```

number4.py is included.
Using vectorized vs for loop code shows quite a difference in the amount of time it takes to estimate the volume of the paraboloid.

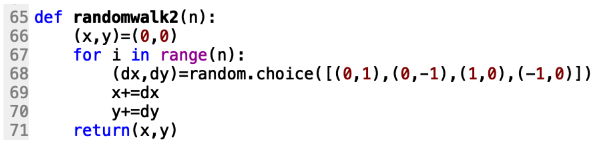


```
Volume using vectorized code:  981.4100000000001
The vectorized code method took  0.06984424591064453  time units
Volume using for loop code:  982.595
The for loop code method took  19.73798179626465  time units
```

5. Please state whether or not the two functions `randomwalk` and `randomwalk2` are equivalent below:



```python
def randomwalk(n):
    x=0
    y=0
    for i in range(n):
        way=random.choice(['N','S','E','W'])
        if way=='N':
            y+=1
        elif way =='S':
            y+=-1
        elif way =='E':
            x+=1
        else:
            x+=-1
    return(x,y)
```



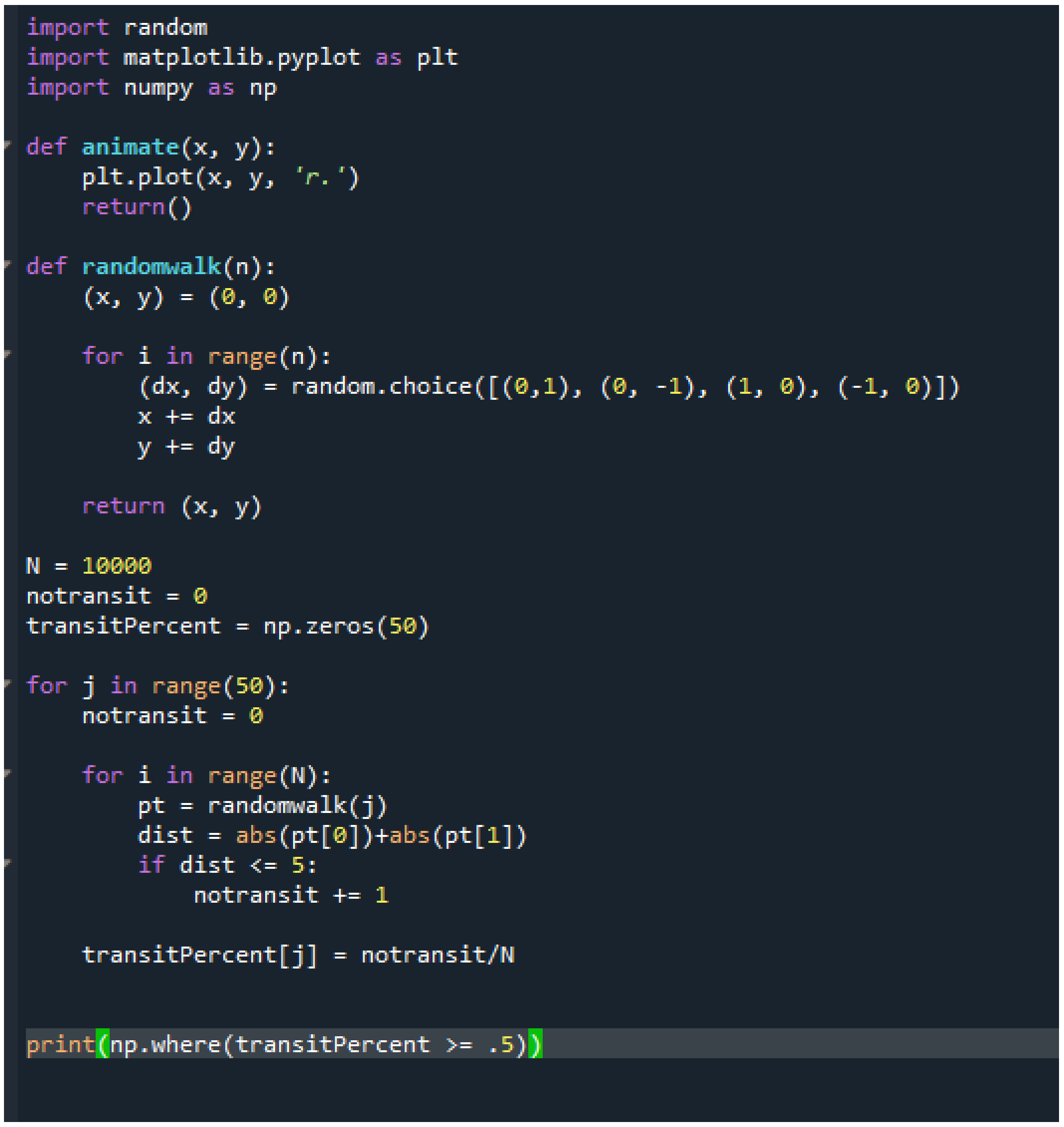
```python
def randomwalk2(n):
    (x,y)=(0,0)
    for i in range(n):
        (dx,dy)=random.choice([(0,1),(0,-1),(1,0),(-1,0)])
        x+=dx
        y+=dy
    return(x,y)
```

   Your answer must include a detailed explanation.
   Both methods are the same, and we can break it down into four cases. When randomwalk chooses N, we are adding 1 to our y value, which is the same when randomwalk2 chooses $(0, 1)$. When randomwalk chooses S, it is the same as randomwalk2 choosing $(0, -1)$. When randomwalk chooses E, the equivalent in randomwalk2 is $(1, 0)$ and when randomwalk chooses W, the equivalent is $(-1, 0)$.

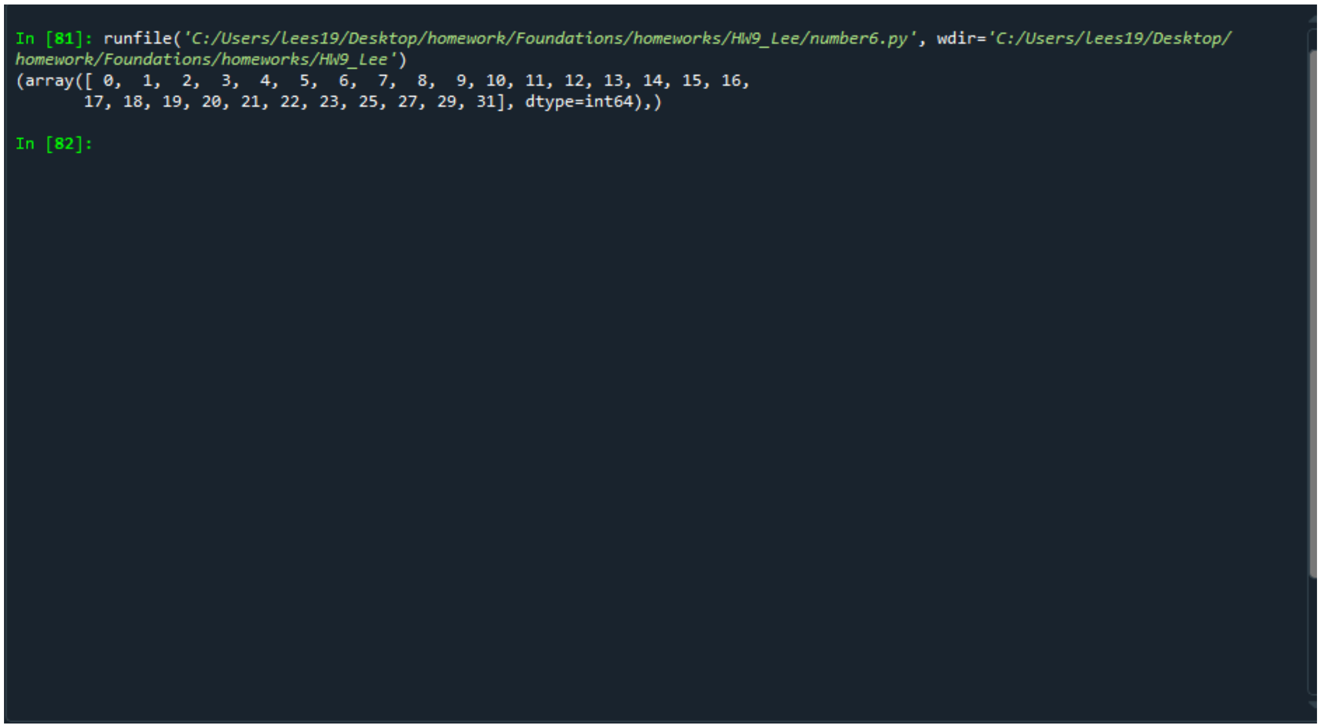
   Since all of the cases in randomwalk are covered in randomwalk2, they produce the same values thus they are equivalent.

6. Find the longest random walk which will, with probability greater than or equal to 50%, leave you less than or equal to 5 blocks from the your starting point. Use a Monte Carlo simulation to solve this and include your python code pasted in here and attached as a file in your zipped folder.

   number6.py is included.
   Using the Monte Carlo method, we find that the longest random walk which will leave you less than or equal to 5 blocks away from the start with probability greater than or equal to 50% is 31 steps.



```
import random
import matplotlib.pyplot as plt
import numpy as np

def animate(x, y):
    plt.plot(x, y, 'r.')
    return()

def randomwalk(n):
    (x, y) = (0, 0)

    for i in range(n):
        (dx, dy) = random.choice([(0,1), (0, -1), (1, 0), (-1, 0)])
        x += dx
        y += dy

    return (x, y)

N = 10000
notransit = 0
transitPercent = np.zeros(50)

for j in range(50):
    notransit = 0

    for i in range(N):
        pt = randomwalk(j)
        dist = abs(pt[0])+abs(pt[1])
        if dist <= 5:
            notransit += 1

    transitPercent[j] = notransit/N


print(np.where(transitPercent >= .5))
```



```
In [81]: runfile('C:/Users/lees19/Desktop/homework/Foundations/homeworks/HW9_Lee/number6.py', wdir='C:/Users/lees19/Desktop/
homework/Foundations/homeworks/HW9_Lee')
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 25, 27, 29, 31], dtype=int64),)

In [82]:
```