In [39]:

```
#Import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
```

In [48]:

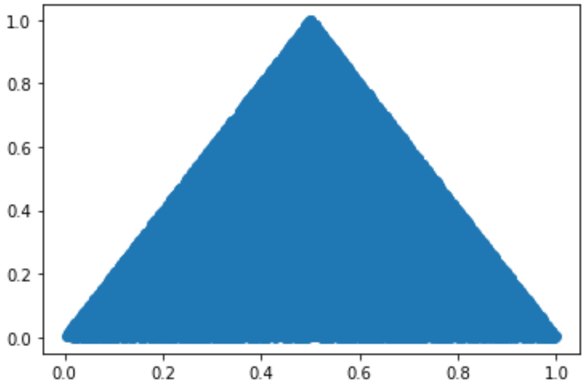
```
#Define functions as needed
def triangle_random_sampler(random_coords):

    x_put = []
    y_put = []

    #Loop through random coordinates
    for u in random_coords:

        #Understand which cdf we need to perform inverse sampling on
        if u[0]<=0.5:

            inv_tranf = (u[0]/2)**.5


        else:

            mirror = u[0]-.5
            inv_tranf = 1- (mirror/2)**.5 #takes advantage of symmetry


        x_put.append(inv_tranf)

        if inv_tranf <=0.5:

            y_max = inv_tranf*2

        else:
            y_max = 2 - (2*inv_tranf)

        y=y_max*u[1]
        y_put.append(y)

    return(x_put, y_put)

def coord_maker(uniform_sample):

    x=0
    y=1
    coordinates = []

    while y<len(uniform_sample):

        sample = [uniform_sample[x],uniform_sample[y]]
        coordinates.append(sample)
        y+=2
        x+=2

    return(coordinates)


testy = np.random.uniform(0.0,1.0,100000)
sample_output =coord_maker(testy)
```

In [49]:

```
first, second = triangle_random_sampler(sample_output)
```

```
In [50]: plt.scatter(first,second)
```

```
Out[50]: <matplotlib.collections.PathCollection at 0x28c5c6b7588>
```



```
In [ ]:
```

```
In [ ]:
```