Consider a G(N,p) network with N = 3000 nodes, connected to each other with probability p = 1/1000

(a) What is the expected number of links, ⟨L⟩?

In [2]:

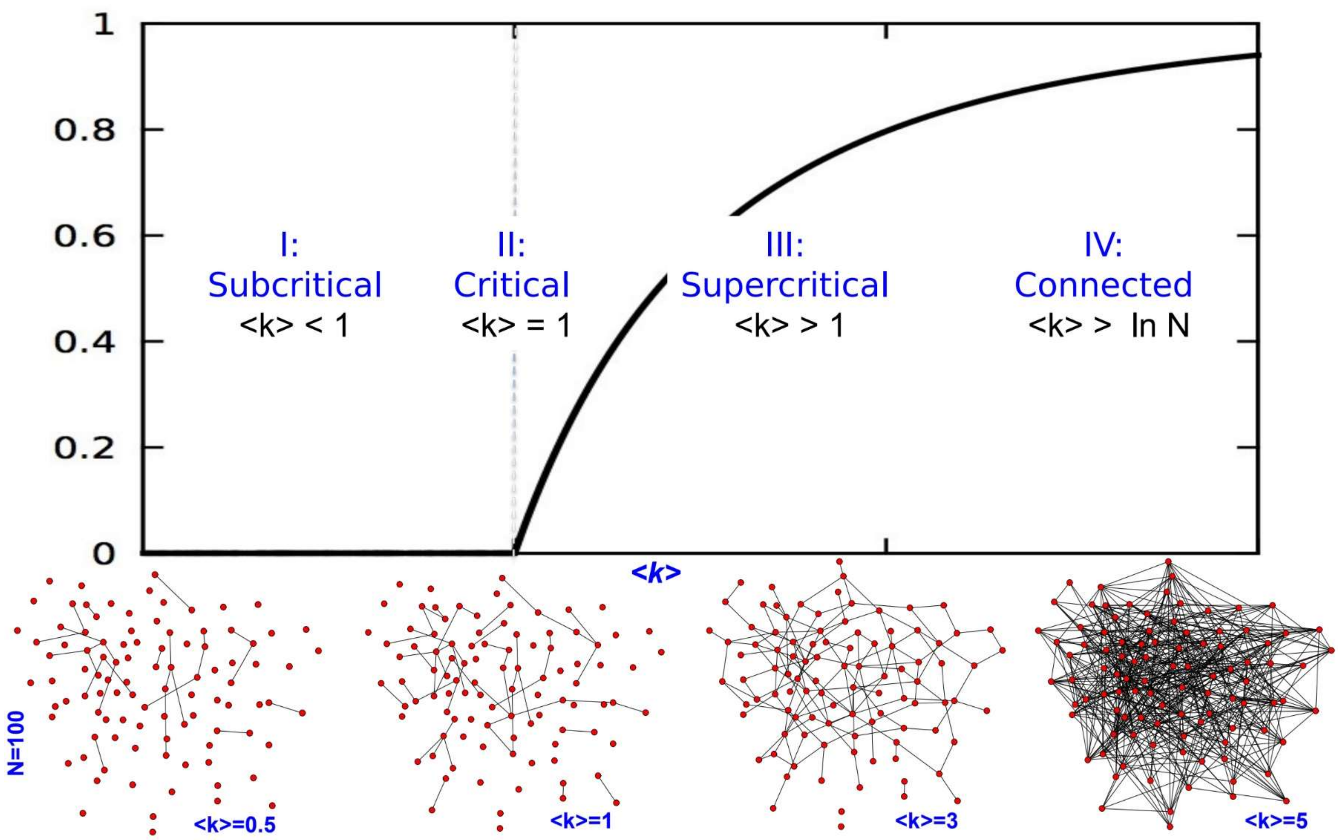
```python
N = 3000
p = 1/1000
# According to the formulas <k> = 2L/N and <k> = p * (N - 1) we have:
# p*(N-1) = 2*L/N
# So on average :
L = (p * N * (N - 1))/2
L
```

Out[2]:

```
4498.5
```

(b) In which regime is this network?

There are four regimes:
Subcritical, Critical, Supercritical, and, Connected



In [3]:

```
# So, calculating <k> we can determine it's regime
avg_k = p * (N - 1)
avg_k # This is a supercritical network
```

Out[3]:

```
2.999
```

(c) Plot the degree distribution p_k of this network (approximate with a Poisson degree distribution).

The poisson distribution uses the formula :
pk = (e^-k) * (<k>^k)/k!
So, we have :

In [4]:

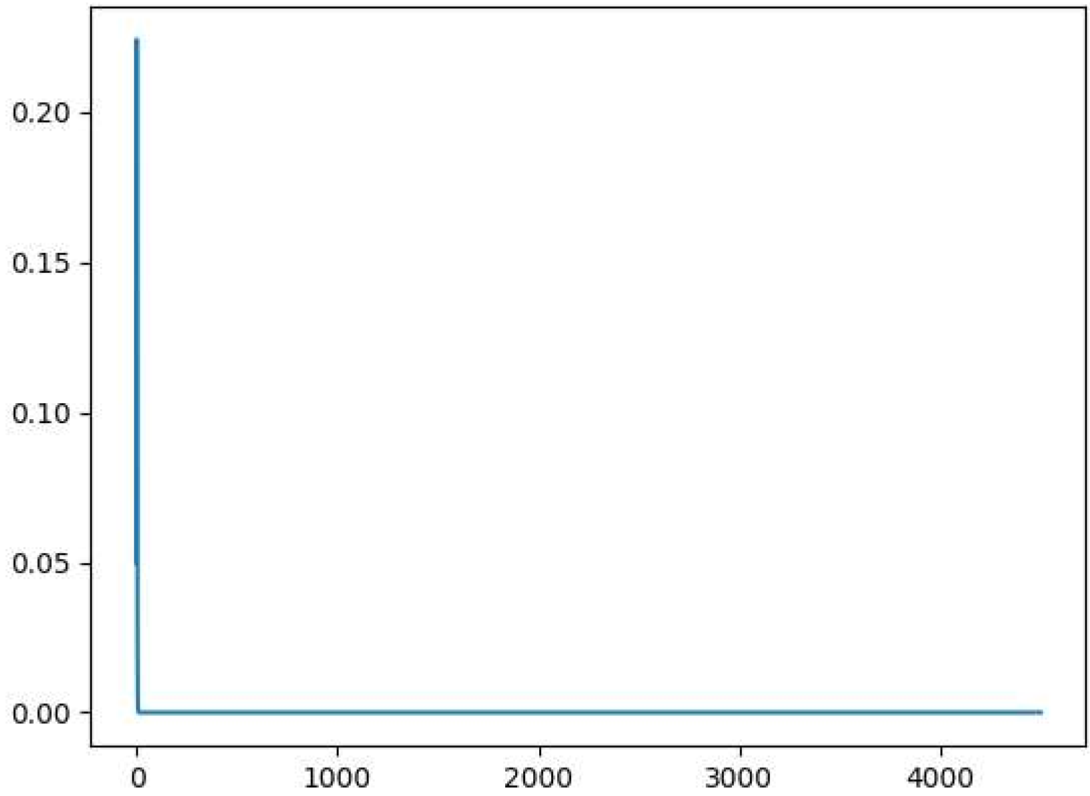
```
from scipy.stats import poisson
import numpy as np
import matplotlib.pyplot as plt

# Building an array to represent the x_axis, in this case, k
# Since L is the average number of Links, it does not make sence to have a k bigger
x_axis = np.arange(0, L, 1)

# mu is <k>
# loc is where the sequence starts, moves the graph to tthe right
y_axis = poisson.pmf(x_axis, mu=avg_k, loc=0)
```

In [5]:

```
# Showing the distribution
plt.plot(x_axis, y_axis)
plt.show()
```



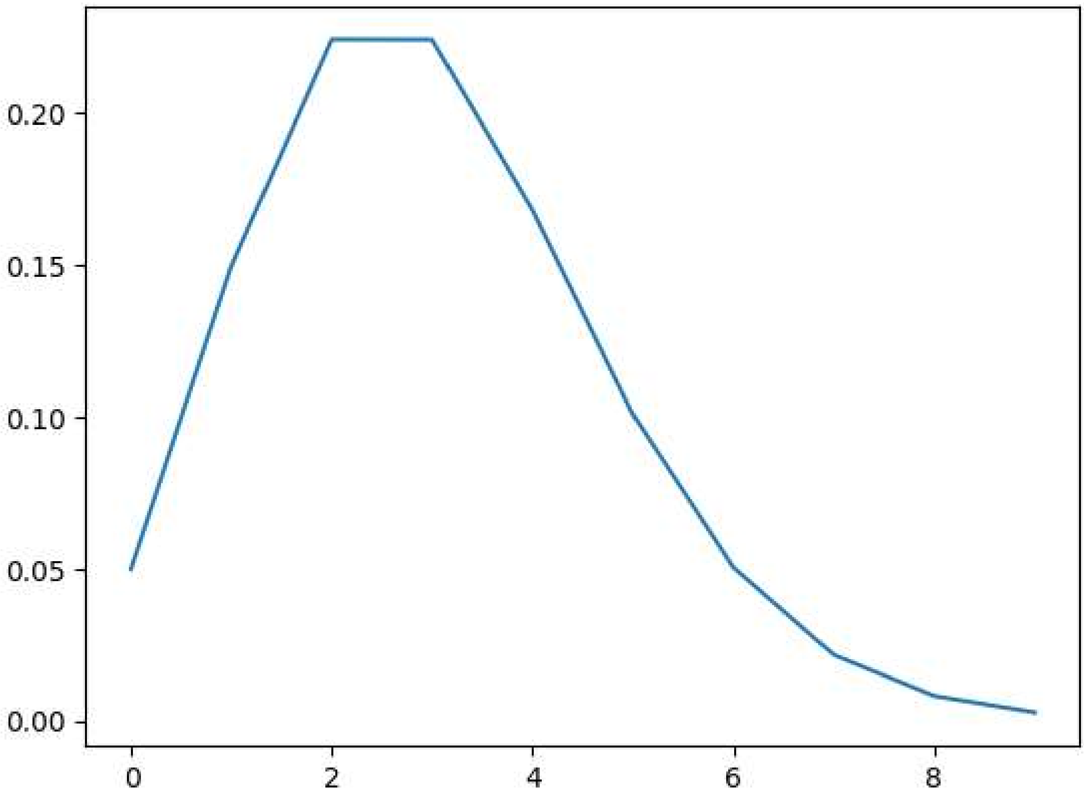
The graph does not look good, because, this shows the distribution of the probability (y axys) of a random node having a degree k (x axys).
Since the average degree is 2.999, a radom node has a higher probability of having a degree 2.999
This probability is diferent than the one used before

In [6]:

```
# Building a better view
x_axis = np.arange(0, 10, 1)
y_axis = poisson.pmf(x_axis, mu=avg_k, loc=0)
```

In [7]:

```
# Showing the distribution
plt.plot(x_axis, y_axis)
plt.show()
```



(d) Given the number of nodes N=3000, calculate the probability p_c so that the network is at the critical point.

In [8]:

```
# <k> must be 1, so:
avg_k = 1
p = avg_k / (N - 1)
p
```

Out[8]:

```
0.00033344448149383126
```

(e) Given the linking probability p = 1/1000, calculate the number of nodes N^{cr} so that the network has only one component.

<k> must be bigger than ln(n), so:
1/1000 = ln(N) / (N - 1)
(N - 1)/1000 = ln(N)
e^(N - 1) = N^1000
N = 1

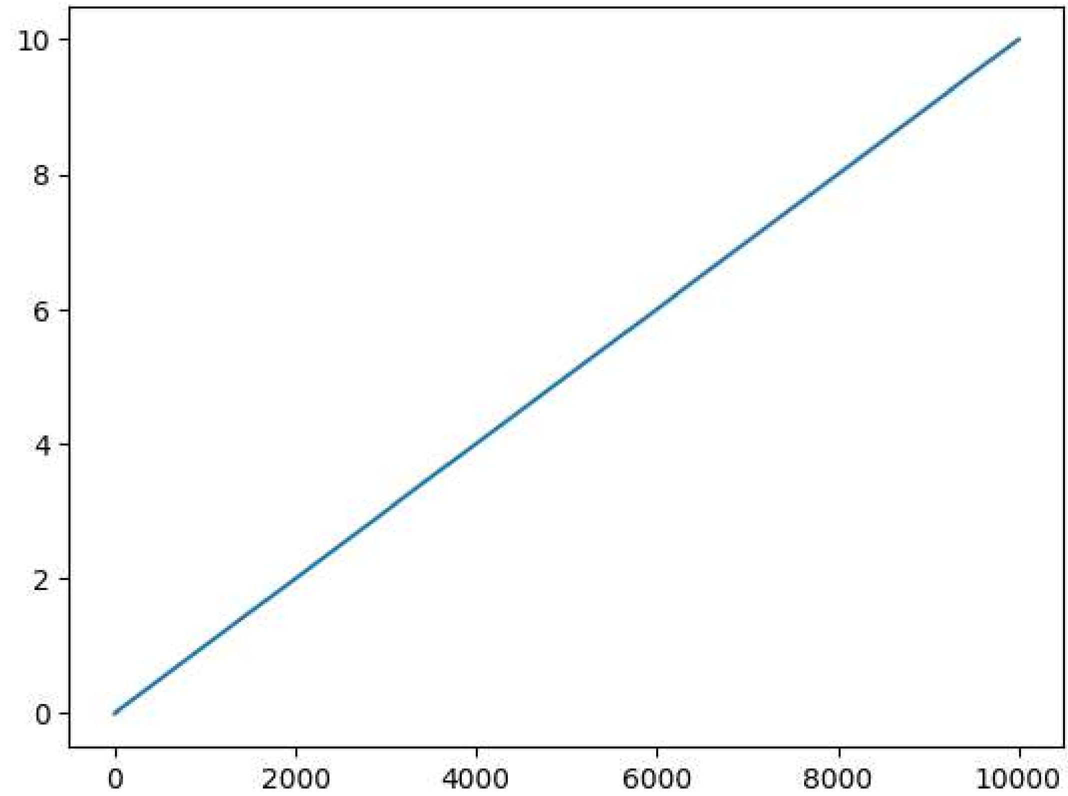
This is more of a phylosofical question than a mathematical one, p is the probability of any give pair of nodes
having a link between them. So, a graph with only one node, is connected.

In [9]:

```
p = 1/1000
# This is the number of nodes
x_axis = np.arange(0, 10000, 1)
# This is the average degrees
y_axis = np.arange(0, p*(10000), p)
```

In [10]:

```
# Showing the distribution to analyse each component
plt.plot(x_axis, y_axis)
plt.show()
```



And, also, if one in a one thousand nodes has a connection to other node, that also implies that if N = 1000, there is a network that is connected.
Since we are talking about a random network, it can also be connected.

So :
N = 1
or
N = 1000