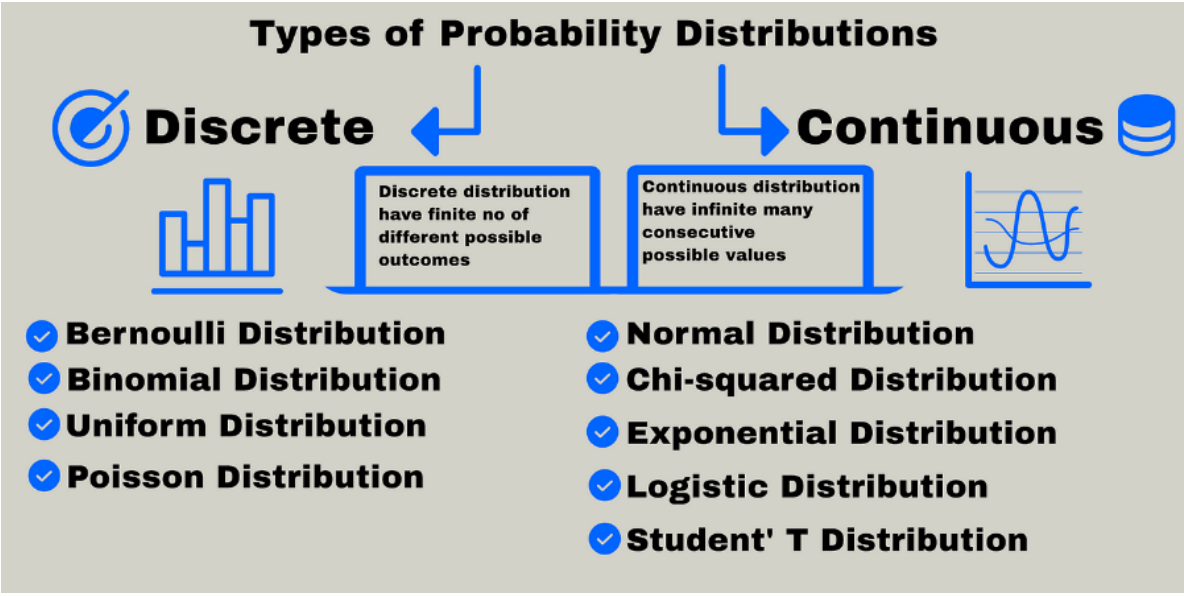
Why should we care about the Distributions ?

- Distributions of data can determine what analytics you perform and what model you utilize.
- During Model Selection Distributions aid in selecting appropriate models for data analysis. Different distributions are suitable for different types of data. For instance, if the data follows a normal distribution, we may choose linear regression models, while count data following a Poisson distribution may lead us to consider generalized linear models. Understanding the distribution of the data guides us in selecting the most appropriate models for accurate analysis and predictions.
- Predicting number of car crashes during a specific time period we use the Poisson distribution.
- Predicting the birth weight of a baby we consider as Normal distribution.

Probability Distributions

A probability distribution is a statistical function that describes all the possible values and probabilities for random variable within a given range. This range will be bunded by minimum and maximum possible values but where the possible value would be plotted on the probability distribution will be determined by a number of factors.like **Mean,standard deviation and skewness.**

Types of Probability distribution



Discrete Probability Distributions

These distributions model the probabilities of random variables that can have **discrete values as outcomes** we called as Discrete Probability Distribution.

Discrete probability distributions are graphs of the outcomes of test results, such as a value of 1, 2, 3, true, false, success, or failure

Continuous Probability Distributions

A continuous distribution describes the probabilities of a continuous random variable's possible values.For example, the possible values for the random variable X that represents weights of citizens in a town which can have any value like 34.5, 47.7, etc.,

Density Functions

One of the critical things in any study related to distributions is Density functions. So, what are density functions? The density functions are mathematical functions that describe the probability distribution of a random variable X.

Probability Mass Function (PMF):

Probability Mass Functions describe the probability of a random variable X taking on a particular value x, and It is only **applicable for discrete distributions**. Mathematically PMF is given as

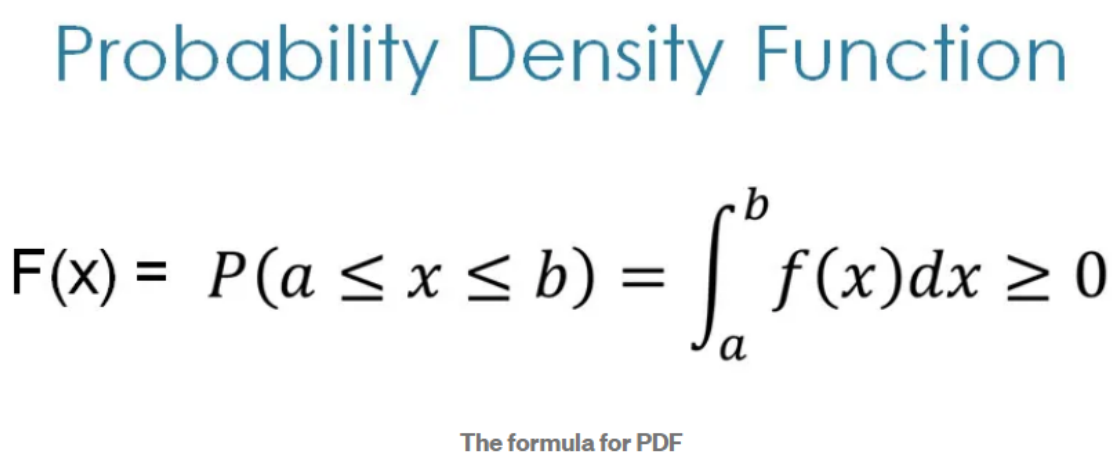
$$p(x) = P(X = x)$$

The probability of x = the probability(X = one specific x)

Fig:- Formula for PMF

Probability Density Function (PDF):

The Probability Density Function is PMF equivalent but for continuous random variables. A continuous distribution is characterised by infinite numbers of the random variables, which means the probability of any random sample at a given point is infinitesimally low. So, a range of values is used to infer the likelihood of a random sample. And doing an integration over the range will fetch our likelihood for the same sample. Mathematically,



Cumulative Distribution Function**

The cumulative distribution function is applicable for describing the distribution of random variables either it is continuous or discrete

Discrete Probability Distribution:

1. Bernoulli Distribution:

Bernoulli Distribution is a discrete Probability distribution representing the discrete probability of a random variable which can take only one of the **two possible values** such as 1 or 0 yes or no, true or false the probability of random variable taking value as 1 is p and value as 0 is (1-p).

The PMF is Given as

$$P(n) = \begin{cases} 1-p & \text{for } n = 0 \\ p & \text{for } n = 1, \end{cases}$$

Where,

***p* = probability of success**

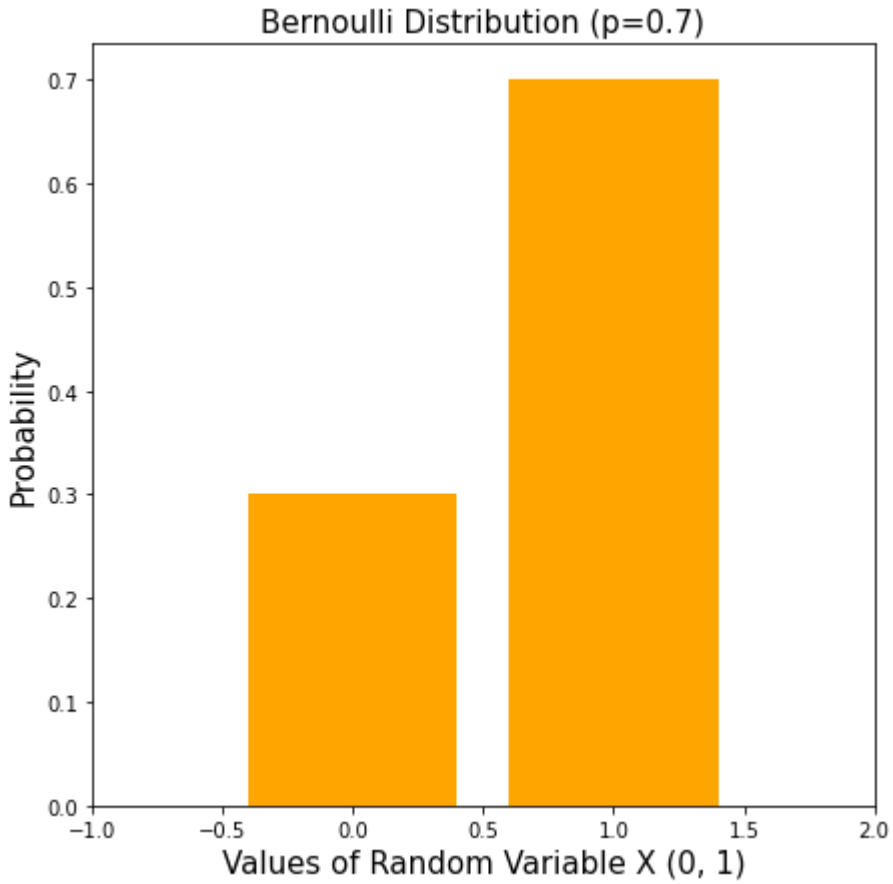
(1 - *p*) = q = probability of failure

In [1]:
```
import matplotlib.pyplot as plt
from scipy.stats import bernoulli

bd = bernoulli(0.7)

X =[0,1]

plt.figure(figsize=(7,7))
plt.xlim(-1, 2)
plt.bar(X, bd.pmf(X), color='orange')
plt.title('Bernoulli Distribution (p=0.7)', fontsize='15')
plt.xlabel('Values of Random Variable X (0, 1)', fontsize='15')
plt.ylabel('Probability', fontsize='15')
plt.show()
```



2. Binomial Distribution :

It is the collection of Bernoulli trials for the same event,i.e it contains more than 1 Bernoulli event for the same scenario for which the Bernoulli trial is calculated.

To Understand the What is the Binomial Distribution we Solve the one questions.

There are 21 unbiased coins. each of them flipped what is the probability of getting even number of heads?

An Unbiased coin has a 50/50 chance of landing on either head or tails. The binomial distribution is a probability distribution that can be used to describe the number of successful or unsuccessful outcomes in a series of events, which must be independent of each other.it is when there are only two possible outcomes like heads or tails and the probability of success is the same for each trial.

To find the probability of getting an even number of heads when flipping 21 coins, we need to calculate the probability of getting 0, 2, 4, 6, 8, 10, 12, 14, 16, 18, or 20 heads.

We will do this by using the binomial distribution:

$$P(X = k) = \binom{n}{k} * p^k * (1-p)^{n-k}$$

It means the following:

P(X = k) — The probability of obtaining k successful outcomes in a total of n independent trials. n — The number of trials. (In this case, 21.) k — The number of successes. (In this case, heads.) p — The chance that a trial is successful. (In this case, 0.5) $\binom{n}{k}$ — The number of ways k successes can be chosen from a total of n independent trials.

Calculation:

$$P(0) = \binom{21}{0} * 0.5^{0} * (1-0.5)^{21-0} = 4,76837e-07$$

$$P(2) = \binom{21}{2} * 0.5^{2} * (1-0.5)^{21-2} = 0.00010013580322265625$$

$$P(4) = \binom{21}{4} * 0.5^{4} * (1-0.5)^{21-4} = 0.002853870391845703$$

$$P(6) = \binom{21}{66} * 0.5^{6} * (1-0.5)^{21-6} = 0.025875091552734375$$

$$P(8) = \binom{21}{8} * 0.5^{8} * (1-0.5)^{21-8} = 0.0970315933227539$$

$$P(10) = \binom{21}{10} * 0.5^{10} * (1-0.5)^{21-10} = 0.16818809509277344$$

$$P(12) = \binom{21}{12} * 0.5^{12} * (1-0.5)^{21-12} = 0.14015674591064453$$

$$P(14) = \binom{21}{14} * 0.5^{14} * (1-0.5)^{21-14} = 0.055446624755859375$$

$$P(16) = \binom{21}{16} * 0.5^{16} * (1-0.5)^{21-16} = 0.00970315933227539$$

$$P(18) = \binom{21}{18} * 0.5^{18} * (1-0.5)^{21-18} = 0.0006341934204101562$$

$$P(20) = \binom{21}{20} * 0.5^{20} * (1-0.5)^{21-20} = 1.0013580322265625e-05$$

In [2]:
```
from scipy.stats import binom
n=21
p=0.5

binom_dist = binom(n,p)

even_probs = binom_dist.pmf(range(0,22,2))

for i, prob in enumerate(even_probs):
    print("The probability of getting {} heads in 21 flips is {}".format(2*i,p
```

```
The probability of getting 0 heads in 21 flips is 4.7683715820312495e-07
The probability of getting 2 heads in 21 flips is 0.00010013580322265595
The probability of getting 4 heads in 21 flips is 0.002853870391845687
The probability of getting 6 heads in 21 flips is 0.025875091552734274
The probability of getting 8 heads in 21 flips is 0.09703159332275389
The probability of getting 10 heads in 21 flips is 0.16818809509277344
The probability of getting 12 heads in 21 flips is 0.14015674591064511
The probability of getting 14 heads in 21 flips is 0.05544662475585908
The probability of getting 16 heads in 21 flips is 0.009703159332275337
The probability of getting 18 heads in 21 flips is 0.0006341934204101569
The probability of getting 20 heads in 21 flips is 1.001358032226562e-05
```

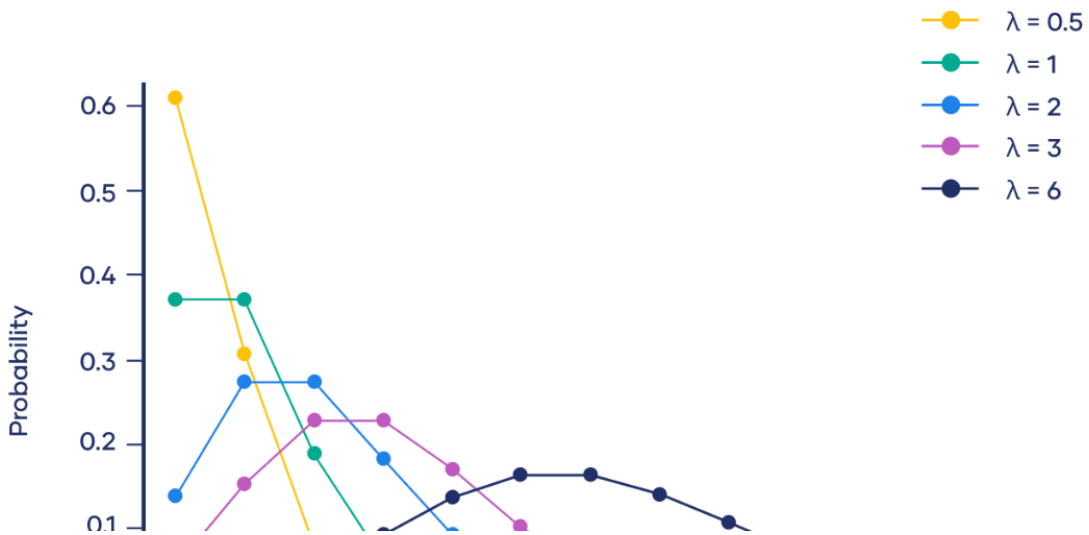
3. Poisson Distribution :

A Possion Distribution is a discrete probability distribuion. it gives the probability of an event happening a certain number of times (k) within a given interval of time or space.

You can use a poission distribution to predict or explain the number of events occuring within a given interval of time or space.For Examples :

- **Phone calls at call center:** The number of incoming phone calls at a call center during a specific time period can often be modeled using a poission distribution the assumption is that the arrival of calls follows a random process with a constant average rate of call arrivals.
- **Website Visits:** The number of visits to a website within a fixed time interval can be modeled using a Poisson distribution. The assumption is that website visits occur randomly and independently, with a constant average rate of visits.
- **Accidents at an Intersection:** The number of accidents that occur at a particular intersection in a given time period can be modeled using a Poisson distribution. The assumption is that accidents happen randomly and independently of each other, with a constant average rate of accidents.

The Poisson distribution has only one parameter, λ (lambda), which is the mean number of events. The graph below shows examples of Poisson distributions with different values of λ.



Possion distribution takes parameter λ, which is the number of expected events happening in a specific time interval and gives us the probability of k events happing in the time fram. It is defined Formula:

$$Poisson(k;\lambda) = \frac{e^{-\lambda} * \lambda^{k}}{k!}$$

Poisson Distribution

Consider one example, the average number of customers arriving at your shop per day is 5. Now you want to know what is the probability of arriving of 8 customers in a single day. You can substitute λ = 5 and k = 8 in the above equation and get the probability 0.06 which is quite low.

$$Poisson(k = 8; \lambda = 5) = \frac{e^{-5} * 5^{8}}{8!} = 0.06$$

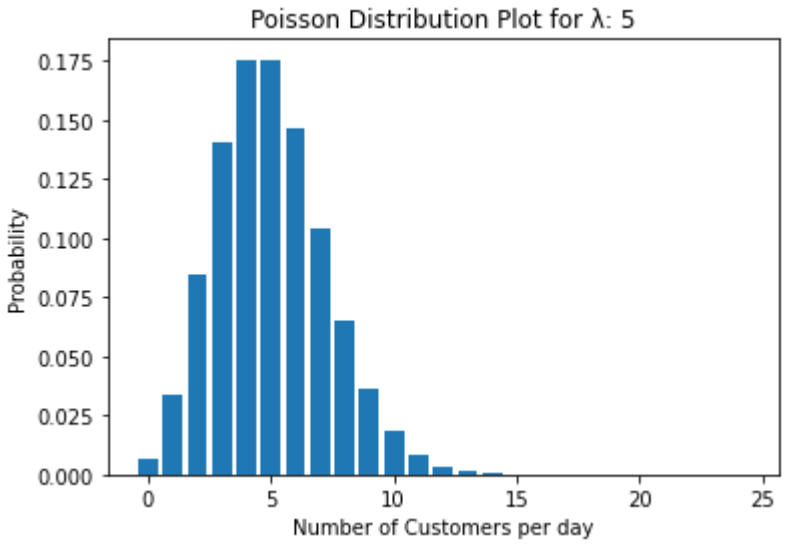
In [5]:
```
from scipy.stats import poisson
import matplotlib.pyplot as plt
poisson_p= [0]*25

k_Val = [ i for i in range(0,25)]
lambda_avg = 5

for i in k_Val:
    poisson_p[i] = poisson.pmf(k=i,mu=lambda_avg)

plt.bar(k_Val, poisson_p)
plt.title(f"Poisson Distribution Plot for λ: {lambda_avg}" )
plt.xlabel("Number of Customers per day")
plt.ylabel("Probability")
```

Out[5]:
```
Text(0, 0.5, 'Probability')
```



Continous Distribution:

1. Normal Distribution

The Normal distribution also known as the Gaussian distribution is a continuous probability distribution that is symmetric and bell- shaped it is characterized by it's mean (μ) and standared deviation (σ).The shape of the normal distribution is determined by parameters.

- The mean (μ) represents the center of the distribution and determines the location of the peak.
- The standard deviation (σ) controls the spread or dispersion of the data points around the mean. A larger standard deviation results in a wider distribution, while a smaller standard deviation makes the distribution narrower.

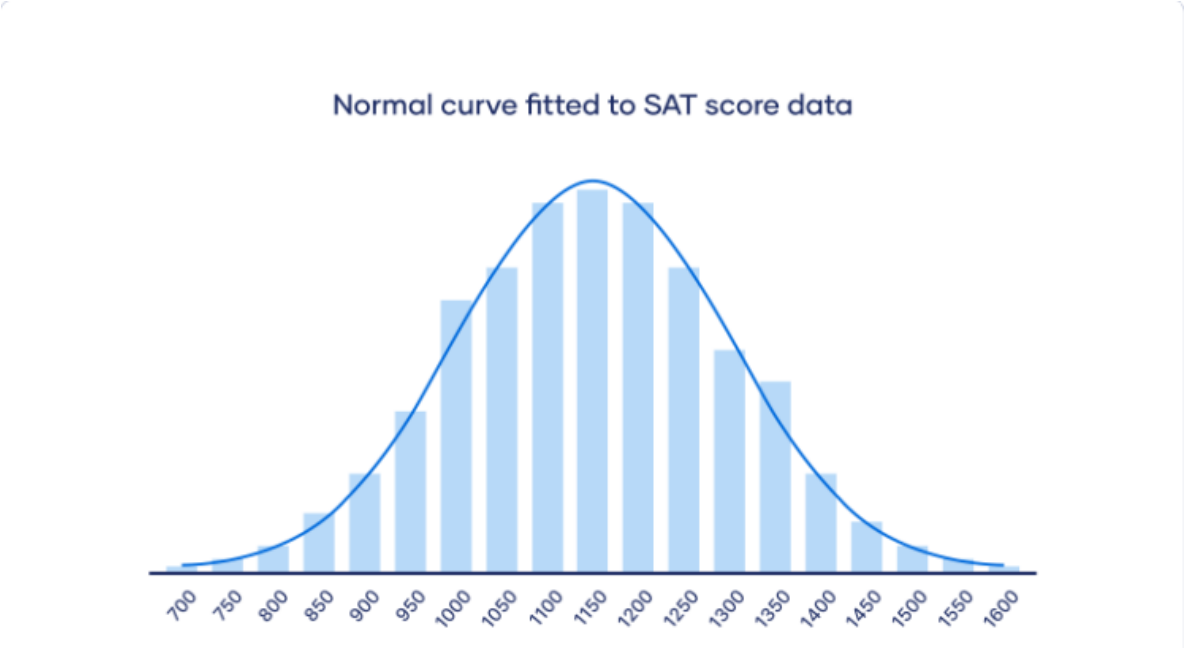
Normal Distribution Formula: The Probability density fuction of normal or gaussian distribution is given by :

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

Where,

- X is the variable
- μ is the mean
- σ is the standard deviation

Once you have the mean and standared deviation of a normal distribution you can fit a normal curve to your data using a probability dencity fucation.



Properties of Normal Distribution:

1. It is symmetrical in nature, that is if you cut the distribution from its center point the two half would be the same.
2. The mean, median,and mode all are equal in the normal distribution.
3. The normal distribution is drawn for continuous random variables.
4. The area under the normal curve is 1 as it is a probability distribution.
5. The empirical rule or the 68-95-99.7 rule tells you where most of your values lies in a normal distribution:

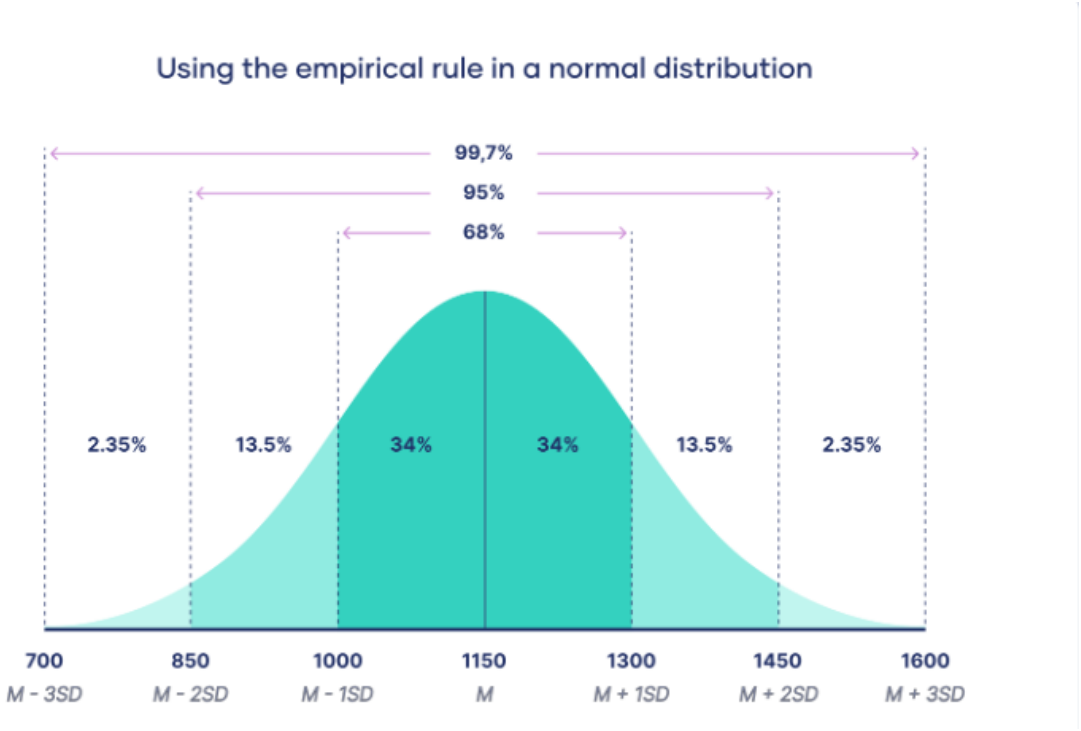
    - Around 68% of values are within 1 standard deviation from the mean.
    - Around 95% of values are within 2 standard deviations from the mean.
    - Around 99.7% of values are within 3 standard deviations from the mean.

You collect SAT scores from students in a new test preparation course. The data follows a normal distribution with a mean score (M) of 1150 and a standard deviation (SD) of 150. Following the empirical rule:

Around 68% of scores are between 1,000 and 1,300, 1 standard deviation above and below the mean.

Around 95% of scores are between 850 and 1,450, 2 standard deviations above and below the mean.

Around 99.7% of scores are between 700 and 1,600, 3 standard deviations above and below the mean.

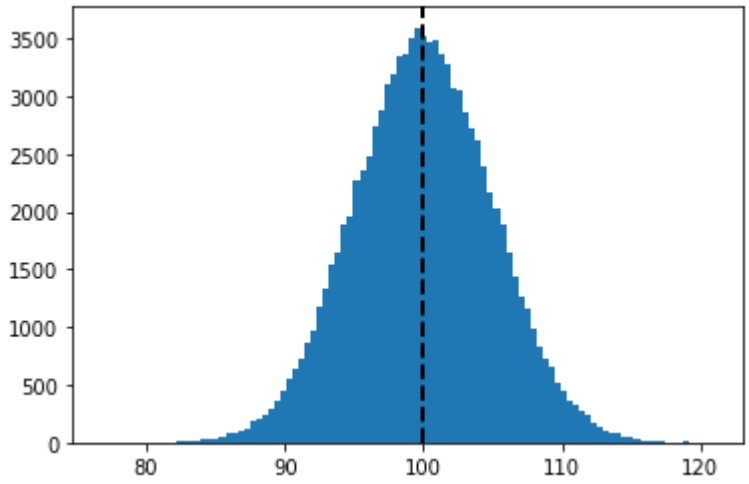


In [2]:
```
import numpy as np
import matplotlib.pyplot as plt

mean = 100
stand_dev= 5
size = 100000

value = np.random.normal(mean,stand_dev,size)

plt.hist(value,100)
plt.axvline(value.mean(), color='k', linestyle='dashed', linewidth=2)
plt.show()
```



2. Standard Normal Distribution

The standard normal distribution, also called the z-distribution, is a special normal distribution where the mean is 0 and the standard deviation is 1.We convert normal distributions into the standard normal distribution by using the Z score formula

Z-score Formula	Explanation
$z = \frac{x - \mu}{\sigma}$	• x = individual value • μ = mean • σ = standard deviation

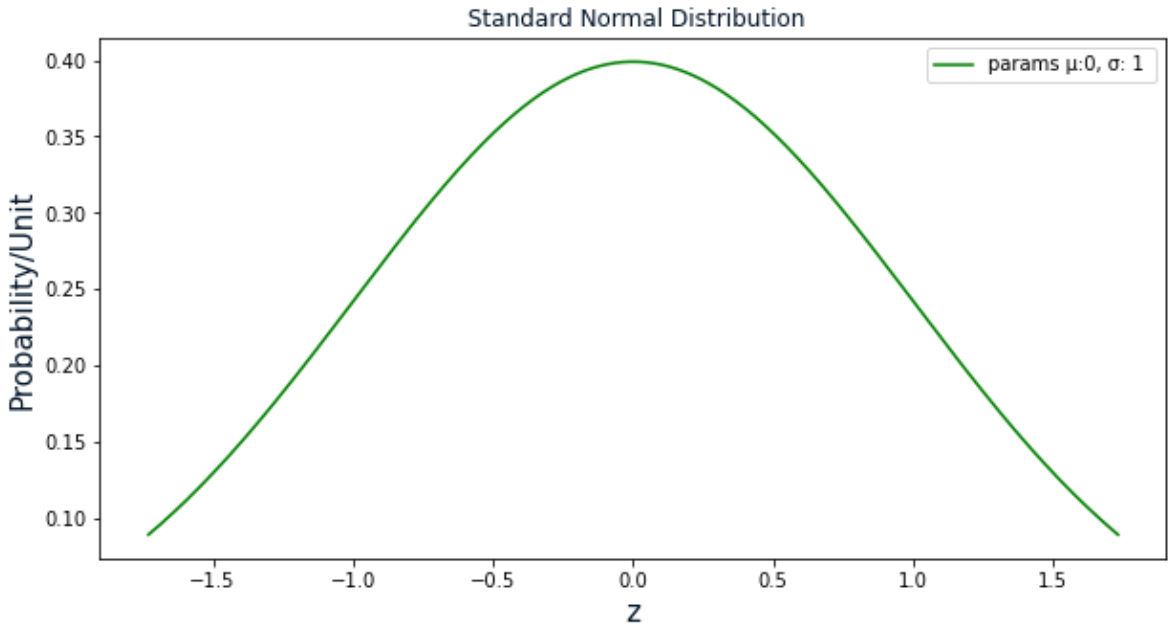
Why we convert the normal distribution into standard normal distribution?

- To find the probability of observations in a distribution falling above or below a given value.
- To find the probability that a sample mean significantly differs from a known population mean.

- To compare scores on different distributions with different means and standard deviations.

In [3]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stat
x = np.linspace(1, 100, 1000)
mean = np.mean(x)
std = np.std(x)
z = (x - mean)/std
# standard normal function has mean i.e loc = 0 and std i.e scale = 1
# z.mean() = 0, z.std = 1
std_normalDist = stat.norm.pdf(z, loc = 0, scale = 1)
plt.figure(figsize = (10, 5))
# now plotting will be done with respect to z on x-axis
plt.plot( z, std_normalDist, color="g", label = f'params μ:{0}, σ: {1} ')
plt.xlabel('z', fontsize=15 ,color="#001427")
plt.ylabel('Probability/Unit', fontsize=15, color="#001427")
plt.title(" Standard Normal Distribution", color="#001427")
plt.legend()
plt.show()
```

```
<frozen importlib._bootstrap>:219: RuntimeWarning: numpy.ufunc size changed,
may indicate binary incompatibility. Expected 216 from C header, got 232 from
PyObject
<frozen importlib._bootstrap>:219: RuntimeWarning: numpy.ufunc size changed,
may indicate binary incompatibility. Expected 216 from C header, got 232 from
PyObject
<frozen importlib._bootstrap>:219: RuntimeWarning: numpy.ufunc size changed,
may indicate binary incompatibility. Expected 216 from C header, got 232 from
PyObject
```



How will you Calculate the Probability of Specific Data Occurrence?

In [4]:
```
# Calculating Probability of Specific Data Occurance,
# the probabaility of randomly picked value will be less than or ewual to 20,
import numpy as np
import matplotlib.pyplot as plt
import scipy.stats as stat
x = np.linspace(1, 100, 1000)
mean = np.mean(x)
std = np.std(x)
normal_distn = stat.norm(loc =mean, scale = std)
normal_pdf = normal_distn.pdf(x)
print("the probability of picked value being less than or equal to 20 is: {}%"
```

```
the probability of picked value being less than or equal to 20 is: 14.32%
```

3. Chi- Square Distribution

What is a chi-square distribution?

Chi-square (X2) distributions are a family of continuous probability distributions. They're widely used in hypothesis tests, including the chi-square goodness of fit test and the chi-square test of independence The shape of a chi-square distribution is determined by the parameter k, which represents the degrees of freedom.

Relationship to the standard normal distribution

Chi-square distributions are useful for hypothesis testing because of their close relationship to the standard normal distribution. The standard normal distribution, which is a normal distribution with a mean of zero and a variance of one, is central to many important statistical tests and theories.

Imagine taking a random sample of a standard normal distribution (Z). If you squared all the values in the sample, you would have the chi-square distribution with k = 1.

X21 = (Z)2

Now imagine taking samples from two standard normal distributions (Z1 and Z2). If each time you sampled a pair of values, you squared them and added them together, you would have the chi-square distribution with k = 2.

X22 = (Z1)2 + (Z2)2

More generally, if you sample from k independent standard normal distributions and then square and sum the values, you'll produce a chi-square distribution with k degrees of freedom.

X2k = (Z1)2 + (Z2)2 + … + (Zk)2

Chi-square test statis

Formula	Explanation
$X^2 = \sum \frac{(O-E)^2}{E}$	Where • X^2 is the chi-square test statistic • $\sum$ is the summation operator (it means "take the sum of") • O is the observed frequency • E is the expected frequency

The shape of chi-square distributions

We can see how the shape of a chi-square distribution changes as the degrees of freedom (k) increase by looking at graphs of the chi-square probability density function.

When k is greater than two

When k is greater than two, the chi-square distribution is hump-shaped. The curve starts out low, increases, and then decreases again. There is low probability that X^2 is very close to or very far from zero. The most probable value of X^2 is $X^2 - 2$.

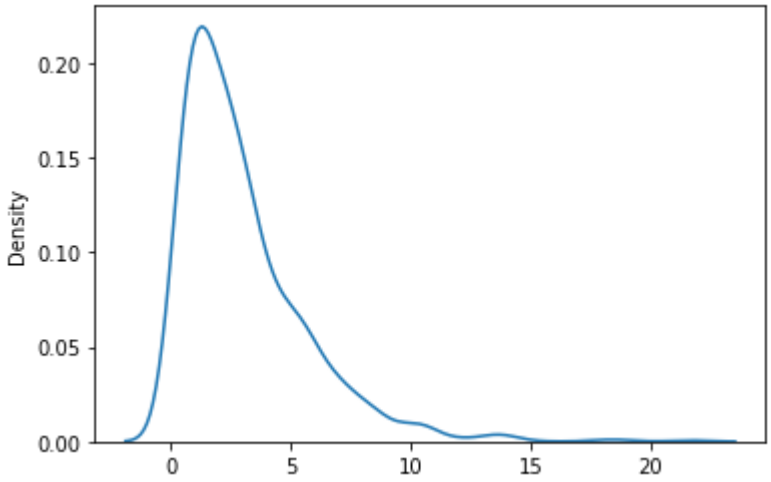
When k is only a bit greater than two, the distribution is much longer on the right side of its peak than its left (i.e., it is strongly right-skewed).

In [7]:
```
from numpy import random
import matplotlib.pyplot as plt
import seaborn as sns

sns.distplot(random.chisquare(df=3, size=1000), hist=False)

plt.show()
```

```
c:\python38\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. P
lease adapt your code to use either `displot` (a figure-level function with s
imilar flexibility) or `kdeplot` (an axes-level function for kernel density p
lots).
  warnings.warn(msg, FutureWarning)
```



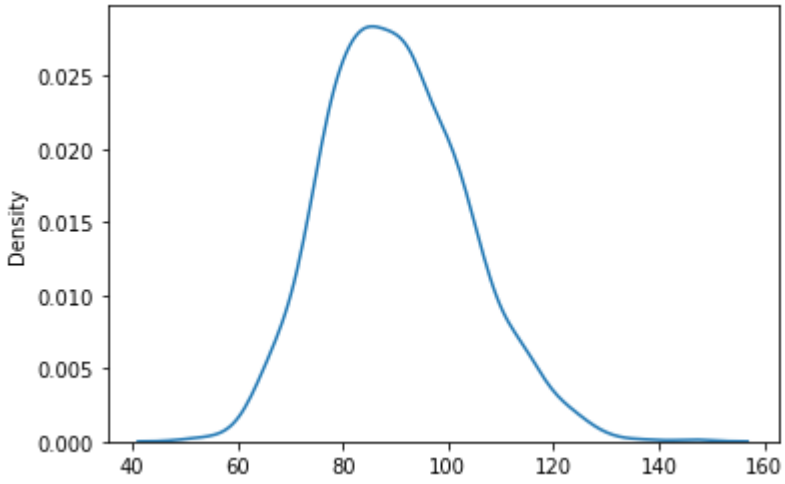
As k increases, the distribution looks more and more similar to a normal distribution. In fact, when k is 90 or greater, a normal distribution is a good approximation of the chi-square distribution.

In [6]:
```
from numpy import random
import matplotlib.pyplot as plt
import seaborn as sns

sns.distplot(random.chisquare(df=90, size=1000), hist=False)

plt.show()
```

```
c:\python38\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. P
lease adapt your code to use either `displot` (a figure-level function with s
imilar flexibility) or `kdeplot` (an axes-level function for kernel density p
lots).
  warnings.warn(msg, FutureWarning)
```



4. LogNormal Distribution (Only Right skewed):

The lognormal distribution is a **continuous probability distribution** of a randome variable whose logarithm follows a normal distribution.In other words if X is a random variable with a lognormal distribution then Y=ln(X) follows a normal distribution.where 'ln' is natural Log.

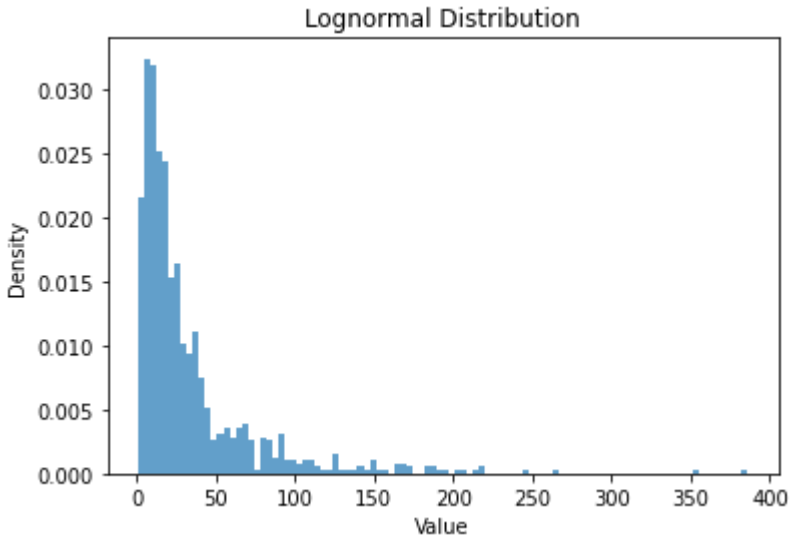
The lognormal distribution is typically used to model variables that are naturally postive and skewed such as **Finance, Economics, Geology, Earth Sciences, Biology,Ecology, Insurence and Risk Analysis.** These are just a few example. where lognormal distribution can be applied.

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

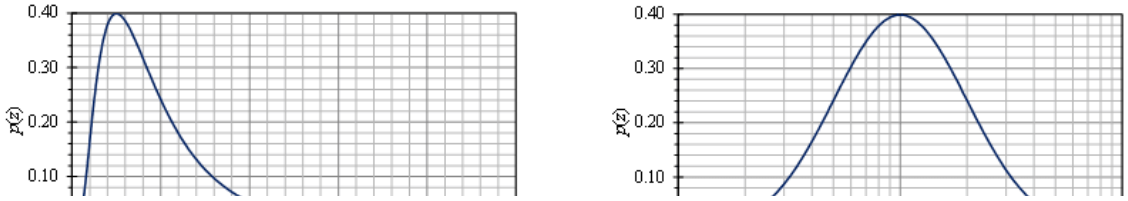
In [4]:
```
# Creat Feature X where X is Lognormal Distribution
Mean,Sigma = 3.,1.
X = np.random.lognormal(Mean,Sigma,1000)
X
```

Out[4]:
```
array([ 63.97225557,  56.46859346,  11.20921107,   1.7875839 ,
        20.71407854,  14.45320252,  36.94097399,  63.24727895,
         2.36455759,  80.63203261,  10.24979847,   8.03139808,
         8.67016619,  17.23982418,  22.71519675,  11.0120035 ,
        18.6324688 ,  16.11044357,  40.72436299,   7.98758264,
        15.08632399,  57.79026054,  34.08199839,  19.61770268,
       188.93638821,  16.12701007,  62.5906344 ,  16.64142602,
        23.37651372,  21.10222368,   8.67864168,  24.50927991,
        12.79396912,   9.29070199,  28.05032313,  21.42657572,
         9.39093588, 218.86335931,  28.76607052,  40.88578198,
        41.58646653,  16.46887396,  28.04078509,  16.30642388,
         3.10610832,   6.77589893,  12.95107095, 149.23735315,
        16.56124511,   9.78949321,  18.68442678,   2.31600809,
        18.84132239,   4.3210536 ,   7.81285955, 202.6488069 ,
        31.94122306,  11.33109028,   4.0433585 ,  18.0522605 ,
        26.22619879, 102.6120237 ,  36.19942782,  37.18590104,
        70.26199713,  46.49882272,  70.57482058,  12.47779633,
        10.54711074, 114.6362038 ,  43.37873159,  23.43278684,
         3.29782659,  54.14096861,   4.39211476,  54.19659223,
```

In [7]:
```
plt.hist(X, bins=100, density=True, alpha=0.7)
plt.xlabel('Value')
plt.ylabel('Density')
plt.title('Lognormal Distribution')
plt.show()
```



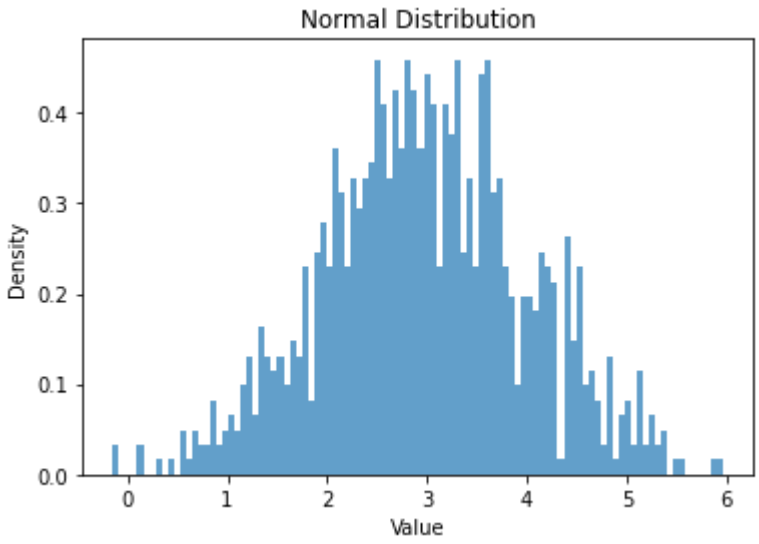
In above figure you can see that Varible X is Lognormal distribtion mean if i am applying the log on variable 'X' Then Data will transform into normal distribution.



Log Transformation

This transformation involves taking the logarithm of the data values. It is useful when dealing with data that has a skewed distribution or when the data spans a wide range of magnitudes. Logarithmic transformation can help normalize the data and make it more symmetric.

In [6]:
```
Y = np.log(X)
plt.hist(Y, bins=100, density=True, alpha=0.7)
plt.xlabel('Value')
plt.ylabel('Density')
plt.title('Normal Distribution')
plt.show()
```



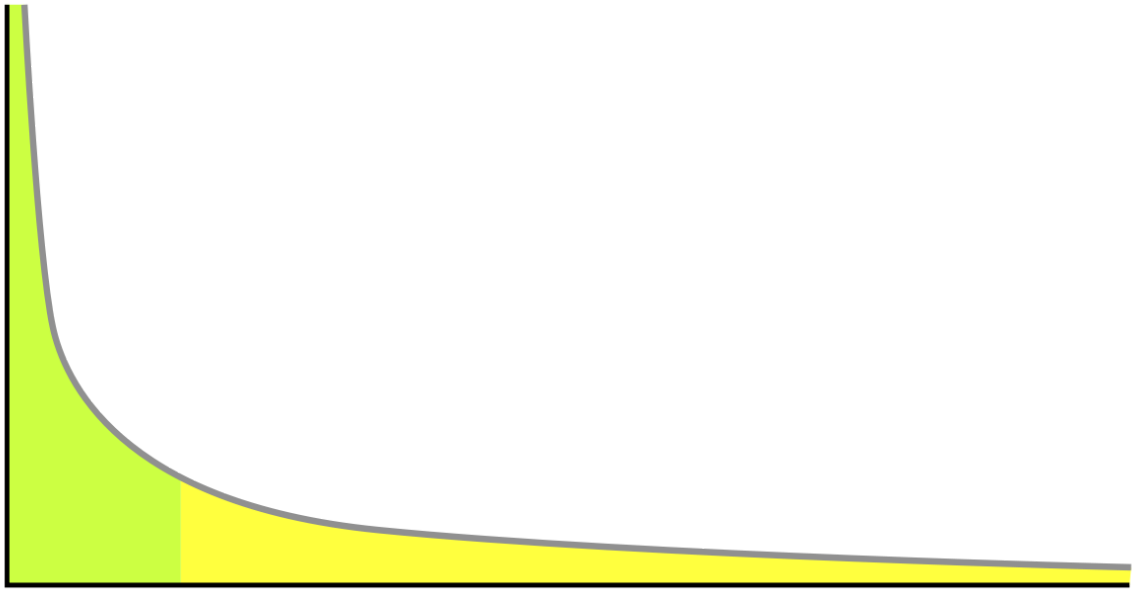
May be you Have question why we transform our data into normal distribution?

Becuse of May statistical models and machine learning algorithem assume that the data follows a normal distribution. In some cases transforming the data into normal distribution can imporove the performance of the model **Example** certain machine learning algorithms, like logistic regression or neural networks, can benefit from normally distributed inputs, as it facilitates the convergence of optimization algorithms and improves the overall predictive accuracy.

Note: However, it's important to note that not all models require data transformation into a normal distribution. Some models, such as decision trees or random forests, are not sensitive to the distributional assumptions of the input data.

5. Power Law Distribution (Continous Randome Variable)

Power Law distribution is a relationship between two quantites where in a relative change in one quantity results in a proportional change in another quantity



It basically Reflects the 80-20 rule:

- 80 per cent of sales are coming from 20 per cent of overall products.
- 80 per cent of window crashes are only due to 20 per cent of overall bugs.

One distribution is an example of Power-law distribution is Pareto distribution.

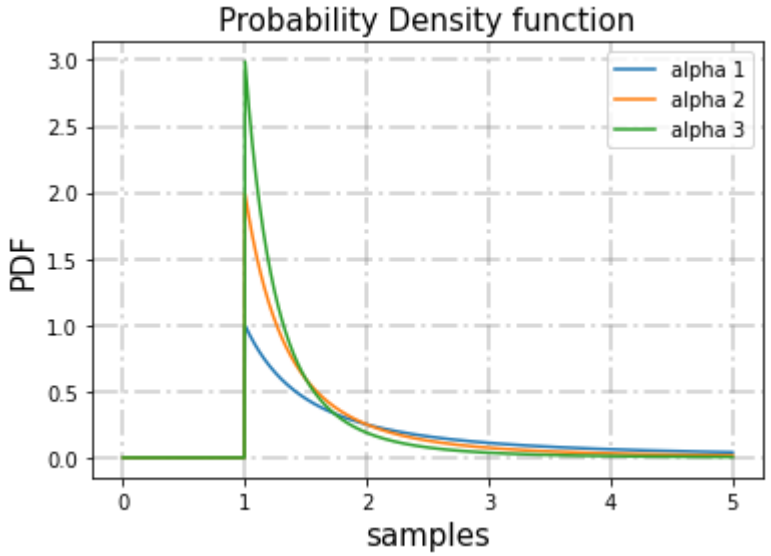
Pareto Distribution

In [2]:
```
import numpy as np
from scipy.stats import pareto

X_m =1 #scale
alpha = [1,2,3]
samples = np.linspace(start=0,stop=5,num=1000)

for a in alpha:
    output = np.array([pareto.pdf(x=samples,b=a,loc=0,scale=X_m)])
    plt.plot(samples,output.T,label ='alpha {0}'.format(a))

plt.xlabel('samples', fontsize=15)
plt.ylabel('PDF', fontsize=15)
plt.title('Probability Density function', fontsize=15)
plt.grid(b=True, color='grey', alpha=0.3, linestyle='-.', linewidth=2)
plt.rcParams["figure.figsize"] = [5, 5]
plt.legend(loc='best')
plt.show()
```



we see What is the pareto distribution but now the question is how i transforma my data into Gaussian distribution.

Box-Cox Transform :

Box-cox Transform is basically a technique to transform any random variable from Pareto distribution (Power-law) to Gaussian Distribution.When our feature is right-skewed then most practitioners use this particular technique to bring variables to normal distribution for easy analysis. Hence it is a feature transformation technique.

In [25]:
```
import numpy as np
from scipy import stats
import seaborn as sns
import matplotlib.pyplot as plt

original_data = np.random.exponential(size=1000)
fitted_data,fitted_lambda = stats.boxcox(original_data)

fig, ax = plt.subplots(1, 2)

sns.distplot(original_data, hist = False, kde = True,
             kde_kws = {'shade': True, 'linewidth': 2},
             label = "Non-Normal", color ="green", ax = ax[0])

sns.distplot(fitted_data, hist = False, kde = True,
             kde_kws = {'shade': True, 'linewidth': 2},
             label = "Normal", color ="green", ax = ax[1])

plt.legend(loc = "upper right")

fig.set_figheight(5)
fig.set_figwidth(10)
```

```
c:\python38\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. P
lease adapt your code to use either `displot` (a figure-level function with s
imilar flexibility) or `kdeplot` (an axes-level function for kernel density p
lots).
  warnings.warn(msg, FutureWarning)
c:\python38\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `
distplot` is a deprecated function and will be removed in a future version. P
lease adapt your code to use either `displot` (a figure-level function with s
imilar flexibility) or `kdeplot` (an axes-level function for kernel density p
lots).
  warnings.warn(msg, FutureWarning)
```

```
Lambda value used for Transformation: 0.2503164084227288
```