# Probability distributions with Python

Dhafer Malouche

## Contents

## 1 Introduction

For any random experience, we try to define a numerical application that represents the outcomes of the experience. For example, when we flip a coin. The outcomes are either *head* or *tail*, and we will then define an application $X$ with values $\{0,1\}$, where $X=0$ if we obtain *tail* and $X=1$ if we obtain *head*.

Such an application $X$ will be then called a numerical random variable.

**Definition** A random variable $X$ is an application from a set $\Omega$, the outcomes of the random experience and a subset of real numbers:

$$X : \Omega \longrightarrow \mathbb{R} \quad \omega \longmapsto X(\omega) \in \mathbb{R}$$

There are two types of random variables:

- *Discrete* random variables where the possible values of $X$ is a discrete subset of $\mathbb{R}$
- *Continuous* random variables where the possible values of $X$ falls in a subinterval of $\mathbb{R}$

## 2 Random variables

**Bernoulli random variables** We flip a coin and the outcomes are either head or tail, The values of $X$ are either 0 or 1. $X$ will be called a Bernoulli random variable. We will be interested to the probability to observe *head*, $\mathbb{P}(X=1)$, and the probability to observe *tail*, $\mathbb{P}(X=0)$.

- $\mathbb{P}(X=0)$ and $\mathbb{P}(X=1)$ belong to $[0,1]$
- $\mathbb{P}(X=0)+\mathbb{P}(X=1)=1$
- Since the values of $X$ are in the set $\{0,1\}$, then $X$ is a discrete variable

If we assume $p=\mathbb{P}(X=1)$, we will say that $X$ is a Bernoulli random variable with parameter $p$

**Tossing a dice** When we toss a dice the outcomes are the numbers : 1,2,3,4,5,6. We can then define a random variable $X$ with values in $\{1,2,\ldots,6\}$ where we will be interested to the probabilities:

$$\mathbb{P}(X=i)=p_i,\ \ \forall\, i=1,\ldots,6$$

- $\forall\, i=1,\ldots,6$, $p_i\in[0,1]$
- $\sum_{i=1}^{6} p_i=1$
- $X$ is a discrete random variable
- When $\forall\ i=1,\ldots,6$, $p_i=1/6$, $X$ will be called a uniform discrete random variable in $\{1,\ldots,6\}$

**Poisson random variables** Assume that we are interested in the number of customers arriving in a store during one hour. Let's denote by $X$ this number. It's a random number, and it can be any number belonging to the set of integers $\mathbb{N}$.

$X$ is then a discrete random variables with values in $\mathbb{N}$. Some scientists have proved that we can find a positive constant $\lambda>0$ such that for all $n\in\mathbb{N}$, we have

$$\mathbb{P}(X=n)=e^{-\lambda}\frac{\lambda^n}{n!}$$

**Life time random variables** In this example, we're interested in the lifetime of electronic components. A positive random variable \$ X\$ can represent this random experience, where the event $\{X>t\}$, for any positive $t$, means that the electronic component is working after the time $t$. Event $\{X<t\}$ means that the electronic component stopped working before the instant $t$.

The random variable $X$ is then a continuous random variable and we will be interested to compute for any positive numbers $a<b$, the following probability:

$$\mathbb{P}(a<X<b)$$

Some scientist proved that we can find a positive constant $\lambda>0$ such that for all $a<b$:

$$\mathbb{P}(a<X<b)=\int_a^b \lambda e^{-\lambda x}dx$$

## 3 What should we know about a random variable?

Any random variable is characterized by its **probability distribution** that can be described by the **cumulative distribution function** (CDF) $F_X$:

$$\begin{aligned} F_X\ :\ & \mathbb{R}\longrightarrow[0,1] \\ & t\longmapsto F_X(t)=\mathbb{P}(X\leq t) \end{aligned}$$

- $F_X$ is an increasing function

- $\lim_{t \to -\infty} F_X(t) = 0$
- $\lim_{t \to +\infty} F_X(t) = 1$
- $F_X$ is a scale function when $X$ is a discrete random variable.
- $F_X$ is of $C^1-$class (continuous and admit a first order derivative) when $X$ is a continuous random variable.

If $X$ is a discrete we define from $F_X$ the probability function:

$$\forall x \;\; f_X(t) = F_X(t) - \lim_{x \to t, x < t} F_X(x) = \mathbb{P}(X \leq t) - \mathbb{P}(X < t)$$

If $X$ is a continuous random variable we can define from $F_X$ the density probability function (pdf) $f_X$. It's the derivative of $F_X$: $f_X = F_X'$.

The density probability function can be used to derive the probability that $X$ falls in an iterval $(a, b)$:

$$\mathbb{P}(X \in (a, b)) = \mathbb{P}(a < X < b) = \int_a^b f_X(t)dt$$

A random variable is fully determined if we know either the CDF or PDF. We will also to write codes that imitate the random experience that provides us the outcomes of $X$. We call this process simulating the random variable.

We will see now with the library *scipy* Python functions that will help us to compute the CDF, the PDF, and to simulate numbers from the probabilty distribution of the random variable.

## 4 Probability distributions with Python

We should first installing SciPy

[1]:
```
!pip install scipy
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: scipy in
c:\users\dhafe\appdata\roaming\python\python310\site-packages (1.7.3)
Requirement already satisfied: numpy<1.23.0,>=1.16.5 in
c:\users\dhafe\appdata\roaming\python\python310\site-packages (from scipy)
(1.22.1)
```

SciPy is a free and open source library for scientific computing that is built on top of NumPy. We have already seen that NumPy has the capability of drawing samples from many common distributions (type `help(np.random)` in the python interpreter), but SciPy has the added capability of computing the probability of observing events, and it can perform computations directly on the probability mass/density functions.

[2]:
```
import numpy as np
```

### 4.1 Bernoulli distribution

[3]:
```
from scipy import stats
```

We define the Bernoulli random variable

```
[4]: X= stats.bernoulli(.3)
```

The Probability function

$\mathbb{P}(X=1)$

```
[5]: X.pmf(1)
```

```
[5]: 0.3
```

$\mathbb{P}(X=0)$

```
[6]: X.pmf(0)
```

```
[6]: 0.7
```

For any $x \notin \{0,1\}$, $\mathbb{P}(X=x)=0$

```
[7]: X.pmf(4)
```

```
[7]: 0.0
```

```
[8]: X.pmf(-2)
```

```
[8]: 0.0
```

The cumulative distribution function $F_X(x)=\mathbb{P}(X\leq x)$

```
[9]: X.cdf(-1)
```

```
[9]: 0.0
```

```
[10]: X.cdf(0)
```

```
[10]: 0.7
```

```
[11]: X.cdf(0.2)
```

```
[11]: 0.7
```

```
[12]: X.cdf(1)
```

```
[12]: 1.0
```

```
[13]: X.cdf(10)
```

```
[13]: 1.0
```

Let's draw the CDF of a Bernoulli random variable. We will first generate random number in $\mathbb{R}$.

```
[14]: import numpy as np
```

```
[15]: data = np.random.random(1000)
```

```
[16]: data[1:10]
```

```
[16]: array([0.62485778, 0.12170412, 0.25287054, 0.78623599, 0.34879936,
             0.74056271, 0.42610447, 0.2546339 , 0.63981476])
```

```
[17]: data.min()
```

```
[17]: 0.0005284937096231568
```

```
[18]: data.max()
```

```
[18]: 0.9998493971559735
```

```
[19]: data=-3*(2*data-1)
```

```
[20]: data.min()
```

```
[20]: -2.999096382935841
```

```
[21]: data.max()
```

```
[21]: 2.996829037742261
```

```
[22]: data[:8]
```

```
[22]: array([-2.05663816, -0.74914665,  2.26977529,  1.48277674, -1.71741591,
              0.90720382, -1.44337625,  0.4433732 ])
```

We sort the data

```
[23]: x = np.sort(data)
```

```
[24]: x[:8]
```

```
[24]: array([-2.99909638, -2.99394417, -2.97917872, -2.97467941, -2.97431385,
             -2.97152315, -2.96997004, -2.95919471])
```

We calculate then the CDF values. We calcule here $F_X(x)$ when $X$ follows a Bernoulli with parameter $p = .4$

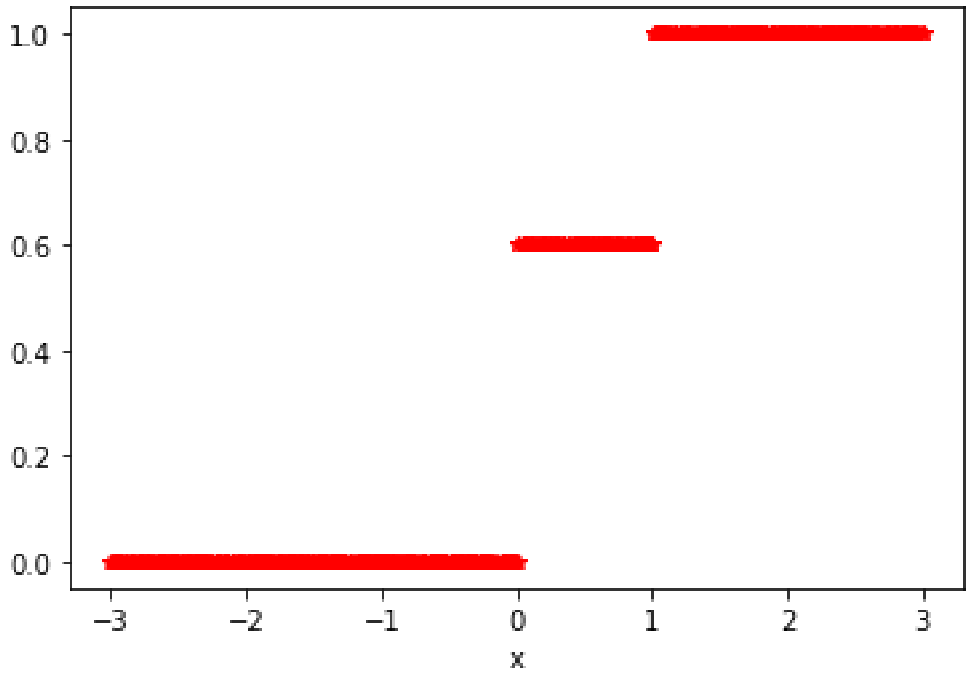
```
[25]: import scipy
```

```
[26]: y = scipy.stats.bernoulli.cdf(x,.4)
```

```
[27]: import matplotlib.pyplot as plt
```

[28]:
```
plt.step(x, y,'r*', where='post')
plt.xlabel('x')
```

[28]:
```
Text(0.5, 0, 'x')
```



The mean of $X$, $\mathbb{E}(X) = p$

[29]:
```
X.mean()
```

[29]:
```
0.3
```

The variance of $X$, $\mathrm{Var}(X) = p(1 - p)$

[30]:
```
X.var()
```

[30]:
```
0.21
```

We also can generate random numbers according to the Bernoulli distribution

[31]:
```
X.rvs()
```

[31]:
```
1
```

[32]:
```
X.rvs(20)
```

[32]:
```
array([1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

```
[33]: z=X.rvs(100)
```

```
[34]: z.mean()
```

```
[34]: 0.37
```

```
[35]: z.var()
```

```
[35]: 0.23309999999999995
```

### 4.2 Poisson distribution

A random variable $X$ with Poisson distribution, with parameter $\lambda > 0$, is a discrete random variable with values in $\mathbb{N}$, with probability function: for all $k \in \mathbb{N}$,

$$p(k) = \mathbb{P}(X = k) = e^{-\lambda} \frac{x^k}{k!}$$

$\mathbb{E}(X) = \lambda$ and $\mathrm{Var}(X) = \lambda$

```
[36]: from scipy import stats
```

We consider $X$ a random variable with Poisson distribution $\lambda = 1.4$

```
[37]: X= stats.poisson(1.4)
```

```
[38]: X.pmf(3)
```

```
[38]: 0.11277701150929471
```

```
[39]: X.pmf(4)
```

```
[39]: 0.039471954028253146
```

Let's draw the probability function of a Poisson random variable

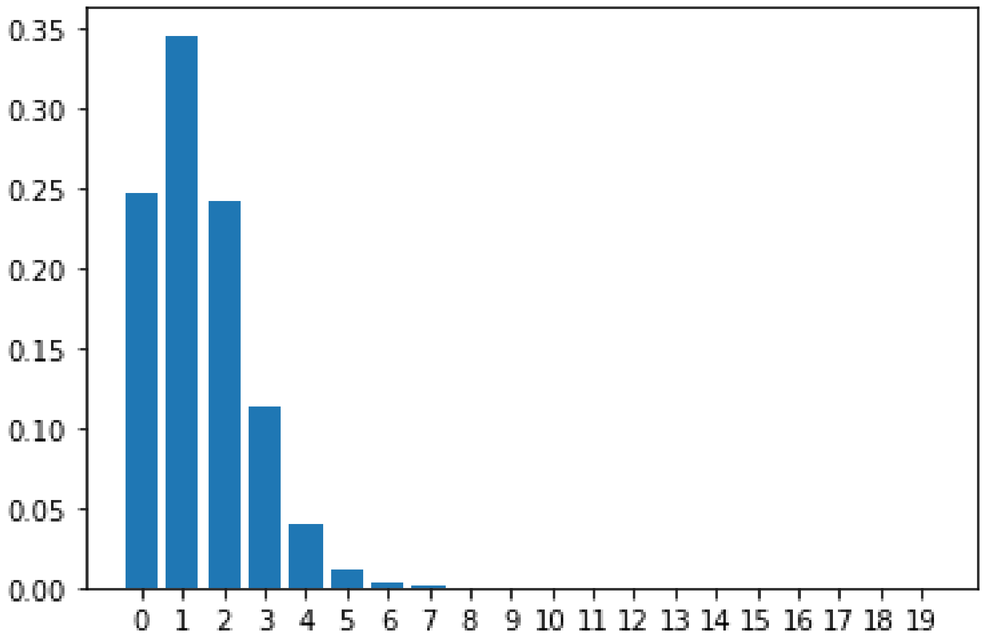
```
[40]: x=np.arange(0,20)
```

```
[41]: y = scipy.stats.poisson.pmf(x,1.4)
```

```
[42]: y
```

```
[42]: array([2.46596964e-01, 3.45235750e-01, 2.41665025e-01, 1.12777012e-01,
              3.94719540e-02, 1.10521471e-02, 2.57883433e-03, 5.15766866e-04,
              9.02592015e-05, 1.40403202e-05, 1.96564483e-06, 2.50172979e-07,
              2.91868475e-08, 3.14319896e-09, 3.14319896e-10, 2.93365237e-11,
              2.56694582e-12, 2.11395538e-13, 1.64418752e-14, 1.21150659e-15])
```

```
[43]: plt.bar(x, y)
      plt.xticks(x)
plt.show()
```



[44]:
```
X.mean()
```

[44]:
```
1.4
```

[45]:
```
X.var()
```

[45]:
```
1.4
```

Cumulative distribution function

[46]:
```
X.cdf(2)
```

[46]:
```
0.8334977381226298
```

[47]:
```
X.cdf(5)
```

[47]:
```
0.9967988507880886
```

Let's draw the CDF then

[48]:
```
data = np.random.random(1000)
```

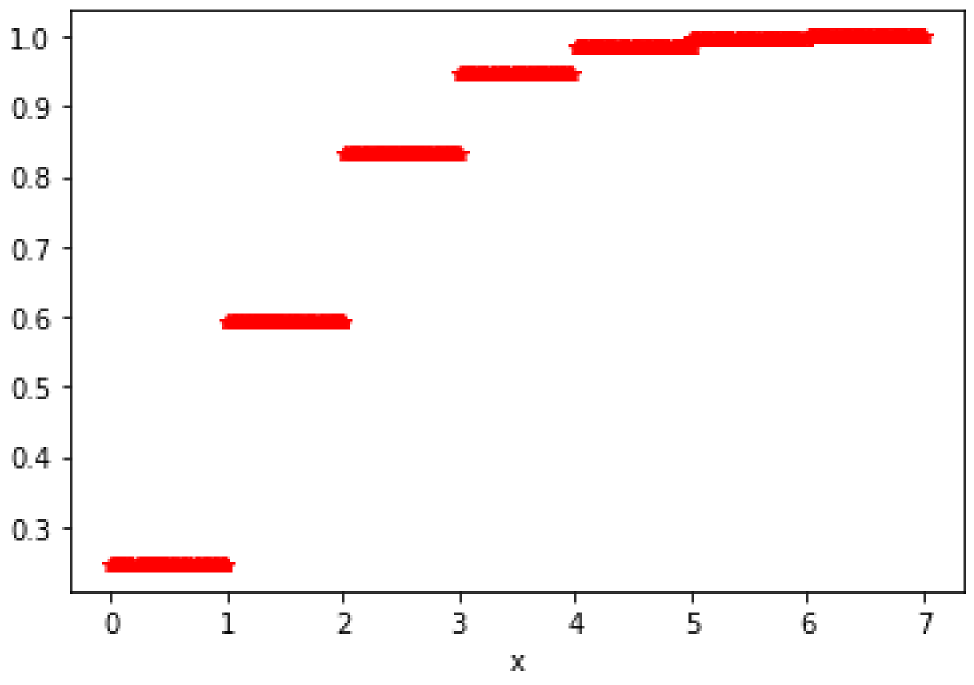
[49]:
```
data=7*data
```

[50]:
```
x=np.sort(data)
```

```
[51]: y = scipy.stats.poisson.cdf(x,1.4)
```

```
[52]: plt.step(x, y,'r*', where='post')
      plt.xlabel('x')
```

```
[52]: Text(0.5, 0, 'x')
```



Generate Poisson random numbers

```
[53]: X.rvs()
```

```
[53]: 5
```

```
[54]: X.rvs(20)
```

```
[54]: array([0, 0, 2, 1, 2, 2, 3, 0, 0, 2, 4, 2, 1, 1, 1, 3, 0, 1, 2, 2])
```

```
[55]: z=X.rvs(100)
```

```
[56]: z.mean()
```

```
[56]: 1.47
```

```
[57]: z.var()
```

```
[57]: 1.5090999999999999
```

Let's estimate the probability of $\mathbb{P}(X < 3)$ and compare it with the theorical value. We will call an **empirical estimation** of $\mathbb{P}(X < 3)$

```
[58]: z<3
```

```
[58]: array([False,  True,  True,  True,  True,  True,  True,  True,  True,
        True, False,  True,  True,  True,  True,  True,  True,  True,
        True,  True, False,  True,  True,  True,  True, False, False,
        True,  True,  True,  True,  True,  True,  True,  True,  True,
       False,  True,  True, False,  True, False,  True, False,  True,
        True,  True,  True, False,  True,  True,  True,  True,  True,
        True,  True,  True, False,  True, False,  True,  True,  True,
       False,  True,  True,  True,  True,  True,  True,  True,  True,
        True,  True,  True, False,  True,  True,  True,  True,  True,
        True,  True,  True,  True,  True,  True,  True, False,  True,
        True, False,  True,  True, False,  True,  True,  True,  True,
        True])
```

```
[59]: zz=z<3
```

```
[60]: zz.sum()
```

```
[60]: 83
```

```
[61]: phat=zz.sum()/len(zz)
```

```
[62]: phat
```

```
[62]: 0.83
```

The theorical value is

```
[63]: X.cdf(2.999)
```

```
[63]: 0.8334977381226298
```

We will show now that we can estimate with a good accuracy the theoritical probability with the empirical estimation by increasing the number of generations

```
[64]: x=np.arange(1,1001)
```

```
[65]: z=X.rvs(1000)
```

```
[66]: zz=z<3
```

```
[67]: phat=zz.cumsum()/x
```

plt.plot(x, phat) plt.axhline(y=X.cdf(2.999),color='red', linestyle='–') plt.xlabel('Number of generations')

**Exercise**

The following table provides the Python commands corresponding to each discrete probability distributions

| Name | Python command | parameters |
|---|---|---|
| Geometric $p$ | `stats.geom(p)` | $p$ is the probability |
| Binomial $(n, p)$ | `stats.binom(n, p)` | $n$ is the size and $p$ is the probability |
| Poisson $(\lambda)$ | `stats.binom(n, λ)` | $\lambda$ is the mean |

## 4.3 Normal distribution

A random variable variable $X$ with Normal distribution if its probability density function is defined as follows:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\left(\frac{x-\mu}{\sigma^2}\right)^2\right]$$

where $\mu \in \mathbb{R}$ and $\sigma > 0$.

$X$ is a continuous random variable with values in $\mathbb{R}$. We write $X \sim \mathcal{N}(\mu, \sigma^2)$.

$\mathbb{E}(X) = \mu$ and $\mathrm{Var}(X) = \sigma^2$

$\mathbb{P}\left(\mu - 1.96\sigma \leq X \leq \mu + 1.96\sigma\right) = .95$

$Z = \dfrac{X-\mu}{\sigma}$ follows the Normal distribution $\mathcal{N}(0, 1)$.

We consider a random variable with mean $\mu = 1.4$ and variance $\sigma^2 = 2^2$

[68]:
```
import math
from scipy import stats
```

[69]:
```
X= stats.norm(loc=1.4,scale=2)
```

[70]:
```
X.cdf(3)
```

[70]:
```
0.7881446014166034
```

[71]:
```
X.mean()
```

[71]:
```
1.4
```

[72]:
```
X.var()
```

[72]:
```
4.0
```

[73]:
```
X.pdf(3)
```

[73]:
```
0.14484577638074136
```

[74]:
```
X.pdf(1.4)
```

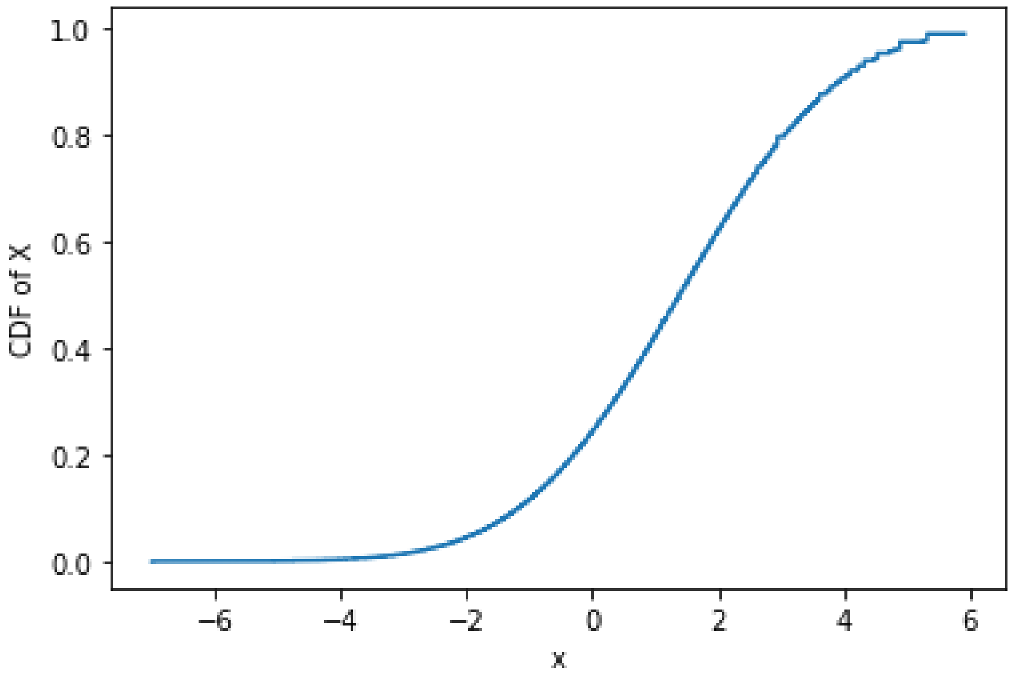
[74]:
```
0.19947114020071635
```

```
[75]: 1/math.sqrt(2*math.pi*4)

[75]: 0.19947114020071635

[76]: X.cdf(-10)

[76]: 5.99037140106353e-09

[77]: X.cdf(0)

[77]: 0.24196365222307303

[78]: X.cdf(1.4)

[78]: 0.5

[79]: X.cdf(1.4-1.96*2).round(3)

[79]: 0.025

[80]: X.cdf(1.4+1.96*2).round(3)

[80]: 0.975

[81]: X.rvs()

[81]: -1.4972065404834658

[82]: data = X.rvs(1000)

[83]: import numpy as np

[84]: data = np.random.normal(size=1000,loc=0,scale=2)

[85]: x = np.sort(data)

[86]: x[0:8]

[86]: array([-7.00527363, -6.40776959, -5.53323088, -5.22003149, -5.07977012,
              -4.95514965, -4.87125955, -4.83750203])

[87]: import scipy

[88]: y = scipy.stats.norm.cdf(x,1.4,2)

[89]: y[0:8]

[89]: array([1.31911847e-05, 4.73304266e-05, 2.63527680e-04, 4.66453618e-04,
              5.97889450e-04, 7.42561820e-04, 8.57427691e-04, 9.08096829e-04])
```

[90]:
```
import matplotlib.pyplot as plt
```

[91]:
```
plt.step(x, y)
plt.xlabel('x')
plt.ylabel('CDF of X')
```

[91]:
```
Text(0, 0.5, 'CDF of X')
```



### 4.4 Exponential distribution

A random variable variable $X$ with Exponential distribution if its probability density function is defined as follows:

$$f(x) = (1/\lambda)\ \exp\left(-x/\lambda\right),$$

if $x \geq 0$, and $f(x) = 0$, if $x < 0$, where $\lambda > 0$ and called the scale parameter.

$X$ is a continuous random variable with values in $\mathbb{R}_+^*$. We write $X \sim \mathcal{E}(\lambda)$.

$\mathbb{E}(X) = \lambda$ and $\text{Var}(X) = \lambda^2$.

[92]:
```
import math
from scipy import stats
```

[93]:
```
X= stats.expon(scale=2.2)
```

[94]:
```
X.mean()
```

```
[94]: 2.2
```

```
[95]: X.var()
```

```
[95]: 4.840000000000001
```

```
[96]: 2.2**2
```

```
[96]: 4.840000000000001
```

```
[97]: X.pdf(2)
```

```
[97]: 0.1831319643314241
```

```
[98]: X.pdf(-1)
```

```
[98]: 0.0
```

The cumulative distribution function of an exponential random variable is expressed as following:

$$F_X(x) = 1 - e^{-x/\lambda}$$

when $x > 0$ and $F_X(x) = 0$, when $x \leq 0$.

```
[99]: X.cdf(3)
```

```
[99]: 0.7442708400868994
```

```
[100]: 1-math.exp(-3/2.2)
```

```
[100]: 0.7442708400868994
```

```
[101]: math.exp(0)
```

```
[101]: 1.0
```

```
[102]: X.pdf(0)
```

```
[102]: 0.45454545454545453
```

```
[103]: 1/2.2
```

```
[103]: 0.45454545454545453
```

```
[104]: X.pdf(3)
```

```
[104]: 0.11624052723322756
```

```
[105]: (1/2.2)*math.exp(-3/2.2)
```

```
[105]: 0.11624052723322756
```

```
[106]: X.rvs()
```

```
[106]: 2.2496880342664096
```

Let's the curve of the CDF of $X$

```
[107]: import numpy as np
```

We simulate 1000 random numbers from a Normal distribution with mean $\mu = 0$ and $\sigma = 20$.

```
[108]: data  = np.random.normal(size=1000,loc=0,scale=20)
```

```
[109]: x=np.sort(data)
```

```
[110]: x.min()
```

```
[110]: -56.402034243025184
```

```
[111]: x.max()
```

```
[111]: 58.70478274607458
```

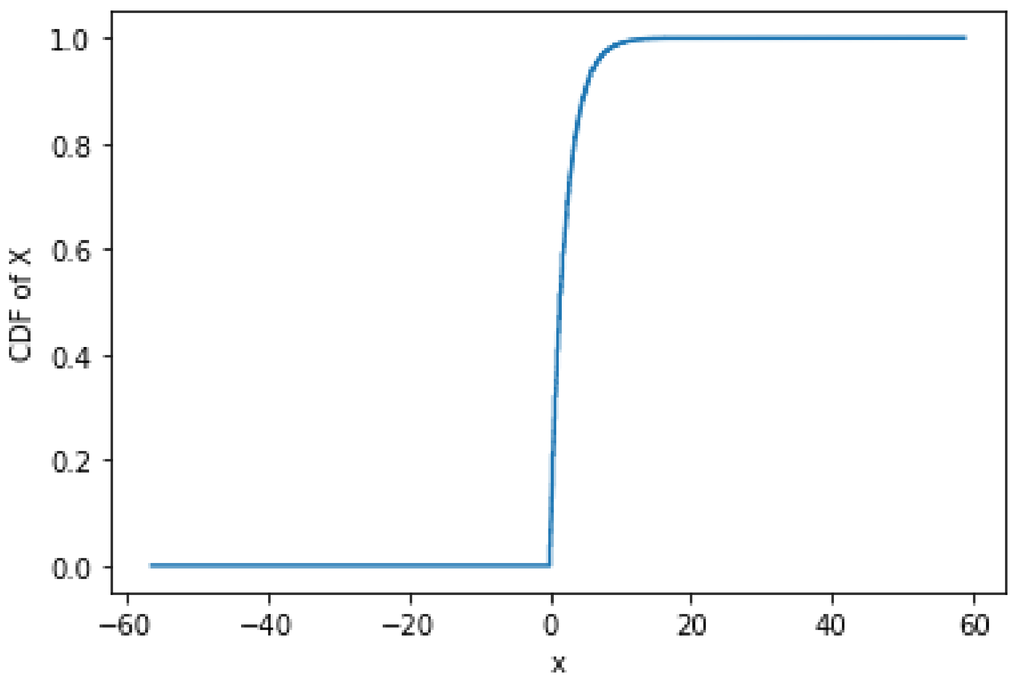
We compute the CDF of each element in x

```
[112]: y = scipy.stats.expon.cdf(x,scale=2.2)
```

```
[113]: import matplotlib.pyplot as plt
```

```
[114]: plt.step(x, y)
       plt.xlabel('x')
       plt.ylabel('CDF of X')
```

```
[114]: Text(0, 0.5, 'CDF of X')
```


CDF of X
x
1.0
0.8
0.6
0.4
0.2
0.0
−60
−40
−20
0
20
40
60