The Uniform Distribution

Contents


```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set(rc={"figure.dpi":100, "savefig.dpi":300})
sns.set_context("notebook")
sns.set_style("ticks")
```

Objectives

- To practice with the uniform distribution.

The uniform distribution

The uniform distribution is the most common continuous distribution. It corresponds to a random variable that is equally likely to take a value within a given interval. We write:

$$X \sim U([0,1]),$$

and we read X follows a uniform distribution taking values in $[0,1]$.

The probability density of the uniform is constant in $[0,1]$ and zero outside it. We have:

$$p(x) := U(x|[0,1]) := \begin{cases} 1, & 0 \le x \le 1, \\ 0, & \text{otherwise.} \end{cases}$$

The cumulative distribution funciton of the uniform is for $xin[0,1]$:

$$F(x) = p(X \le x) = \int_0^x p(u)du = \int_0^x du = x.$$

Obviously, we have $F(x) = 0$ for $x < 0$ and $F(x) = 1$ for $x > 1$.

The probability that X takes values in $[a, b]$ for $a < b$ in $[0,1]$ is:

$$p(a \le X \le b) = F(b) - F(a) = b - a.$$

The expectation of the uniform is:

$$\mathbb{E}[X] = \int_0^1 x dx = \frac{1}{2}.$$

The variance of the uniform is:

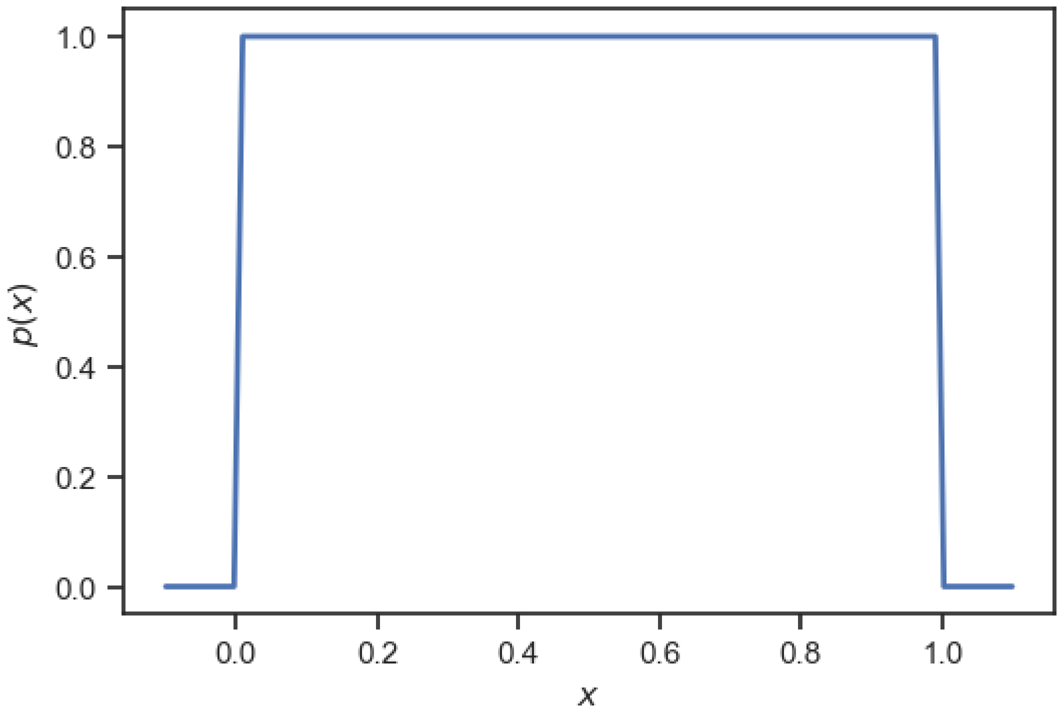
$$\mathbb{V}[X] = \mathbb{E}[X^2] - (\mathbb{E}[X])^2 = \frac{1}{3} - \frac{1}{4} = \frac{1}{12}.$$

Let's create a uniform random variable using scipy:

```
import scipy.stats as st
X = st.uniform()
```

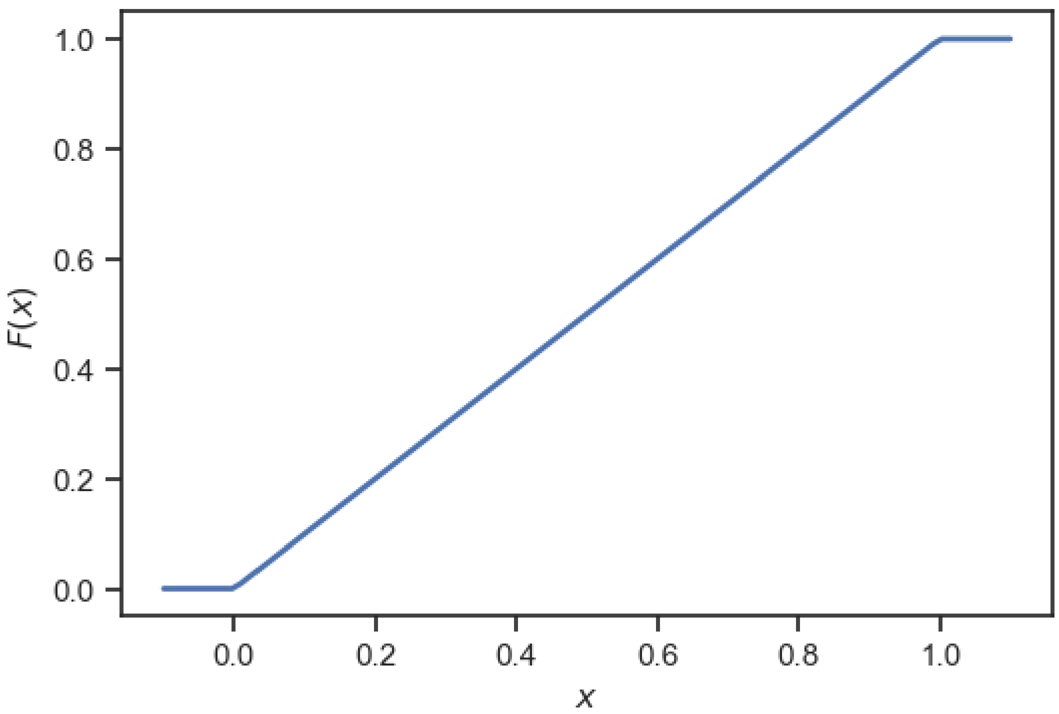
Let's plot the PDF:

```
fig, ax = plt.subplots()
xs = np.linspace(-0.1, 1.1, 100)
ax.plot(xs, X.pdf(xs), lw=2)
ax.set_xlabel("$x$")
ax.set_ylabel("$p(x)$");
```



Now the CDF:

```
fig, ax = plt.subplots()
ax.plot(xs, X.cdf(xs), lw=2)
ax.set_xlabel("$x$")
ax.set_ylabel("$F(x)$");
```



The expectation is:

```
print(f"E[X] = {X.expect():.2f}")
```

```
E[X] = 0.50
```

The variance:

```
# The variance is:
print(f"V[X] = {X.var():.2f}")
```

```
V[X] = 0.08
```

Here is how you can sample from the uniform one hundred times:

```
X.rvs(size=100)
```

```
array([0.66879238, 0.75736348, 0.7812501 , 0.27933322, 0.56074956,
       0.22117569, 0.13367789, 0.08781332, 0.92979958, 0.23459637,
       0.92809888, 0.5814472 , 0.26280797, 0.9108148 , 0.50478432,
       0.6190595 , 0.40672951, 0.17830193, 0.82812523, 0.08030685,
       0.73094922, 0.57757433, 0.30082494, 0.34891424, 0.42321286,
       0.73301765, 0.97325906, 0.85121544, 0.79344264, 0.43710437,
       0.47301335, 0.32680782, 0.32762497, 0.79189128, 0.4932426 ,
       0.39353935, 0.73214114, 0.51196885, 0.82466441, 0.96251777,
       0.57712978, 0.16213913, 0.96023262, 0.8256694 , 0.60125674,
       0.95639564, 0.1813989 , 0.1370642 , 0.72564754, 0.85959444,
       0.74086014, 0.72892095, 0.10661595, 0.48914761, 0.64530463,
       0.78829441, 0.67443356, 0.61846556, 0.87410386, 0.13069682,
       0.73299368, 0.72562731, 0.95585603, 0.58951981, 0.41871065,
       0.25931519, 0.38000015, 0.5878755 , 0.6890931 , 0.14515412,
       0.13497739, 0.30664024, 0.7905363 , 0.48192938, 0.34733383,
       0.21752499, 0.93600126, 0.60442971, 0.20210764, 0.97991932,
       0.13104272, 0.64441741, 0.46702161, 0.30004285, 0.78337234,
       0.19566825, 0.93463603, 0.18010226, 0.54768973, 0.70985843,
       0.33181683, 0.56047789, 0.66093359, 0.13612503, 0.00240562,
       0.406723  , 0.94597236, 0.33125943, 0.79064208, 0.16893134])
```

An alternative way is to use the functionality of numpy:

```
np.random.rand(100)
```

```
array([0.29279788, 0.4337704 , 0.93829631, 0.70369052, 0.8588247 ,
       0.08551964, 0.13681101, 0.53497048, 0.73944118, 0.76798519,
       0.14274447, 0.58691282, 0.27223556, 0.38984886, 0.86537729,
       0.00271175, 0.97873385, 0.37150024, 0.03565874, 0.51056284,
       0.41651455, 0.13996376, 0.71905168, 0.55034888, 0.04843597,
       0.73228376, 0.63664816, 0.13718881, 0.61372497, 0.28604202,
       0.07053062, 0.56984752, 0.9935996 , 0.94679784, 0.41333832,
       0.21864241, 0.89530563, 0.26852291, 0.33993037, 0.99379449,
       0.52043625, 0.69054632, 0.94599312, 0.39209509, 0.15515099,
       0.11913377, 0.36727144, 0.16305775, 0.32788912, 0.93186119,
       0.94213692, 0.38479948, 0.79121509, 0.28130922, 0.61572918,
       0.19639303, 0.23152655, 0.46311709, 0.25954506, 0.37761004,
       0.2387859 , 0.58126836, 0.39016289, 0.87688937, 0.28628025,
       0.09591198, 0.92969956, 0.93099493, 0.15935987, 0.59326533,
       0.28115257, 0.59453267, 0.7654102 , 0.78502993, 0.13140112,
       0.52031933, 0.48488842, 0.21941024, 0.39177227, 0.17355688,
       0.29763789, 0.69964469, 0.37279314, 0.06386166, 0.72562404,
       0.32715819, 0.67465422, 0.13327048, 0.53307795, 0.9865744 ,
       0.99863065, 0.58321671, 0.30553495, 0.10131245, 0.36573083,
       0.86776942, 0.49106171, 0.41292578, 0.11182442, 0.81445401])
```

Finally, let's find the probability that X is between two numbers. In particular, we will find $p(-1 \le X \le 0.3)$:

```
a = -1.0
b = 0.3
prob_X_is_in_ab = X.cdf(b) - X.cdf(a)
print(f"p({a:.2f} <= X <= {b:.2f}) = {prob_X_is_in_ab:.2f}")
```

```
p(-1.00 <= X <= 0.30) = 0.30
```

The uniform distribution over an arbitrary interval $[a,b]$

The uniform distribution can also be defined over an arbitrary interval $[a,b]$. We write:

$$X \sim U([a,b]).$$

The PDF of this random variable is:

$$p(x) = \begin{cases} c, & x \in [a,b], \\ 0, & \text{otherwise}, \end{cases}$$

where c is a positive constant. This simply tells us that the probability density of finding X in $[a,b]$ is something positive and that the probability density of findinig outside is exactly zero. The positive constant c is determined by imposing the normalization condition:

$$\int_{-\infty}^{+\infty} p(x)dx = 1.$$

This gives:

$$1 = \int_{-\infty}^{+\infty} p(x)dx = \int_a^b cdx = c\int_a^b dx = c(b-a).$$

From this we get:

$$c = \frac{1}{b-a},$$

and we can now write:

$$p(x) = \begin{cases} \frac{1}{b-a}, & x \in [a,b], \\ 0, & \text{otherwise}, \end{cases}$$

From the PDF, we can now find the CDF for $x \in [a,b]$: Use these bounds of integration when calculating the cdf from the pdf

$$F(x) = p(X \le x) = \int_{-\infty}^{x} p(u)du = \int_a^x \frac{1}{b-a}du = \frac{1}{b-a}\int_a^x du = \frac{x-a}{b-a}.$$

The expectation is:

$$\mathbb{E}[X] = \frac{1}{2}(a+b),$$

and the variance is:

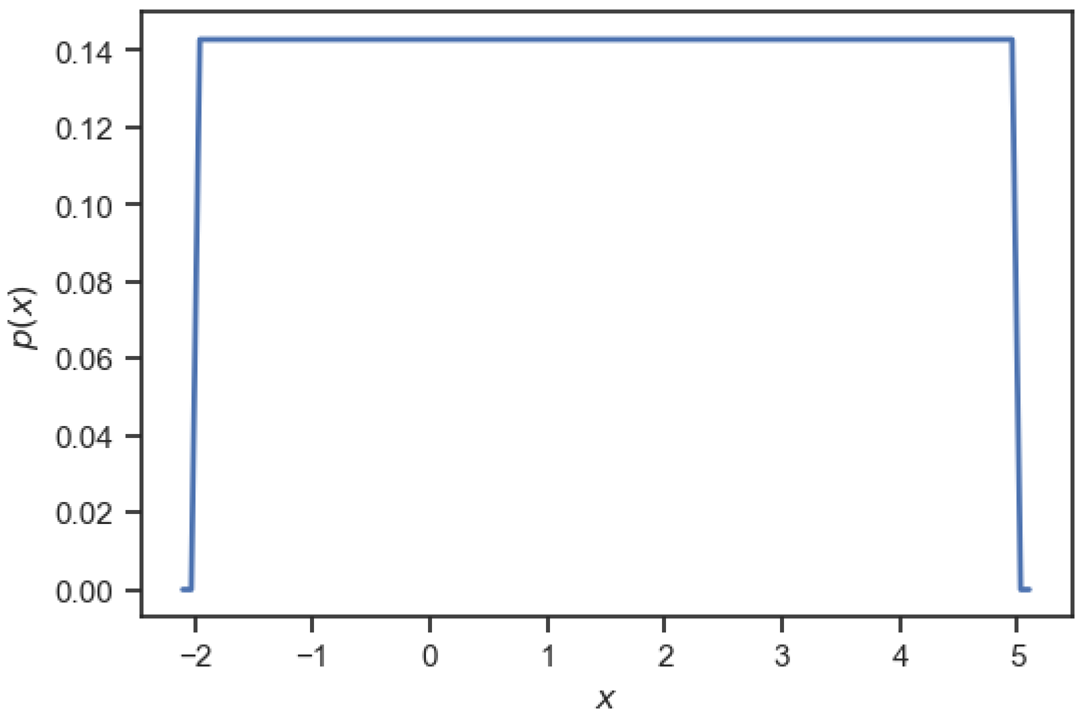
$$\mathbb{V}[X] = \frac{1}{12}(b-a)^2.$$

This is how you can do this using `scipy.stats` for $a=-2$ and $b=5$:

```
a = -2.0
b = 5.0
X = st.uniform(loc=a, scale=(b-a))
```

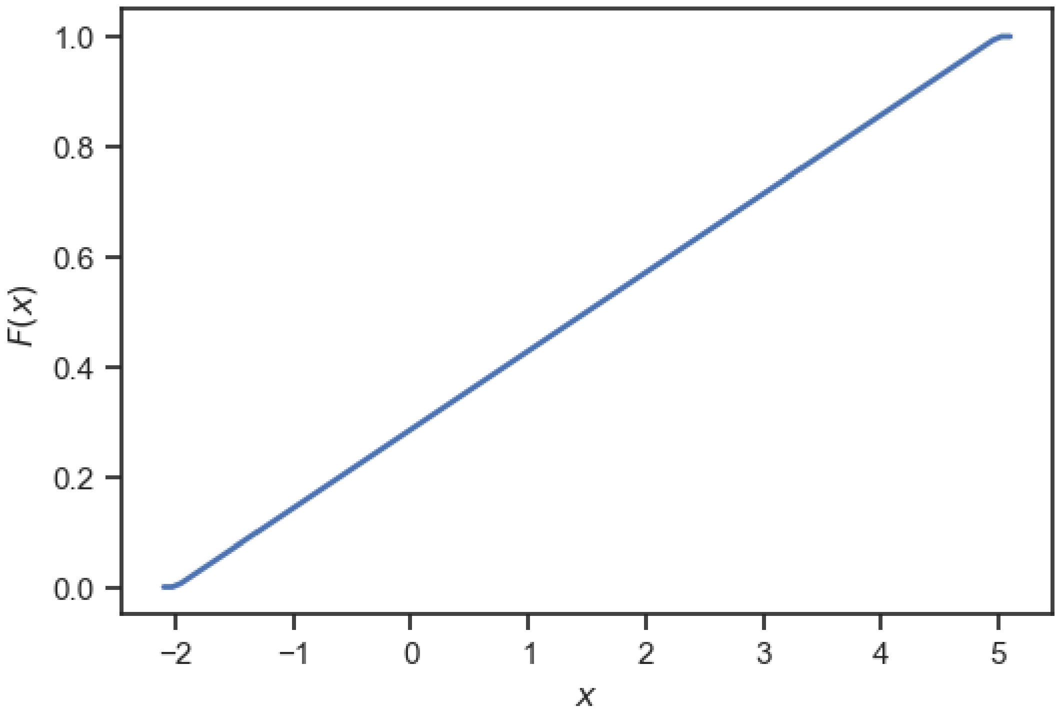
The PDF is:

```
fig, ax = plt.subplots()
xs = np.linspace(a - 0.1, b + 0.1, 100)
ax.plot(xs, X.pdf(xs), lw=2)
ax.set_xlabel("$x$")
ax.set_ylabel("$p(x)$");
```



The CDF is:

```
fig, ax = plt.subplots()
xs = np.linspace(a - 0.1, b + 0.1, 100)
ax.plot(xs, X.cdf(xs), lw=2)
ax.set_xlabel("$x$")
ax.set_ylabel("$F(x)$");
```



The expectation is:

```
print(f"E[X] = {X.expect():.2f}")
```

```
E[X] = 1.50
```

And the variance:

```
# The variance is:
print(f"V[X] = {X.var():.2f}")
```

```
V[X] = 4.08
```

And here are a few random samples:

```
X.rvs(size=100)
```

```
array([-1.45270321,  3.75427557,  3.40194509,  3.82358559,  0.03669189,
        1.43871971,  4.88467637,  3.44841689,  0.42726523,  2.89913466,
        0.85346694,  2.92994211,  3.14694184,  2.01940974, -1.72196249,
        3.82110494,  2.39971611,  1.42718044, -0.90717363,  2.96824853,
        3.70726804, -0.01298326, -0.75734356,  2.54421778,  1.91741186,
       -1.16488972,  4.94482528, -0.17088186, -0.50243453,  1.40639053,
       -1.70192443,  0.42499389,  0.3012409 ,  3.76880934,  2.76757534,
        4.24870146,  3.73095241,  0.17390123,  4.53686561,  4.73901669,
        0.74036288,  0.61196484,  0.612601  , -1.63111776,  0.78341418,
        4.9549641 ,  1.80276436,  2.01231155, -0.20889983, -1.021387  ,
        0.47748887,  4.16589221,  0.69374002,  3.90882291, -1.9144582 ,
        2.14138314, -0.28114618,  4.92789125,  4.91344386, -0.23949328,
        3.41058834,  0.54531666,  3.66371658,  4.82879482,  4.95741444,
       -0.23216379,  3.76404766,  2.53737447,  2.39744778,  4.39600112,
       -0.56450068,  1.88831075,  3.47208535, -1.64247583, -0.2076848 ,
       -0.80705925, -1.91120655,  4.42397359,  2.81177404,  1.25991348,
       -0.36517137, -1.71818306,  2.20497205,  2.36817653, -0.76984531,
        1.24821457,  4.8328534 ,  1.25374121,  2.69404508,  3.42178582,
       -0.63912198,  0.77610503,  0.63436757,  3.99859416,  3.60810124,
        2.53581295, -0.88861836,  0.37966487,  4.47828904, -1.67851729])
```

Alternative way to get $U([a,b])$

There is another way to obtain samples from $U([a,b])$ that uses only samples from $U([0,1])$. Here is how. Let Z be a standard uniform random variable:

$$Z \sim U([0,1]).$$

Then define the random variable:

$$X = a + (b-a)Z.$$

Then, $X \sim U([a,b])$. Why? Well, let's just show that the CDF of X has the right form:

$$p(X \le x) = p(a + (b-a)Z \le x) = p((b-a)Z \le x - a) = p\left(Z \le \frac{x-a}{b-a}\right) = \frac{x-a}{b-a},$$

where the last step follows from the fact that the CDF of Z is simply: $p(Z \le z) = z$. Equipped with this result, we see that we can sample X by sampling Z and then scaling it appropriately (by the way this is what `scipy.stats` is doing internally). Here it is using `numpy.random.rand` to sample in $[0,1]$:

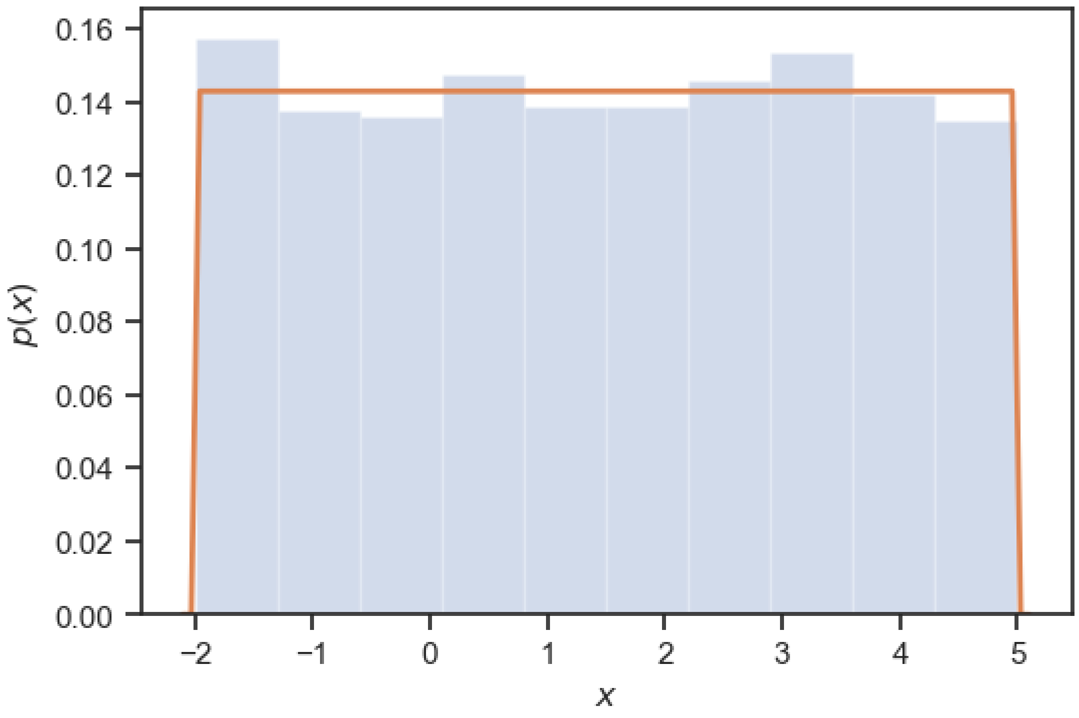
```
x_samples = a + (b - a) * np.random.rand(1000)
print(x_samples)
```

```
[ 4.54012805  0.63093735  0.90144289  0.05800096  1.62693978  3.00228583
  2.49185347  3.46022424  3.17169053  2.81803692  1.41188949  3.9113902
  2.45299521  2.70474767  2.72780705  4.17581858  1.79614172  3.77161182
  0.98587008  1.02207083  2.50454716  2.30094021  0.35108196  4.87606038
 -0.39959862 -0.03394866  3.41444689  2.13816798  1.70894267  3.73298258
  3.55803823  0.74474686  0.0094886   1.73812834  3.32123959  1.07621825
  1.42369461 -1.08534156 -1.01895819  0.88108717  2.90410362 -1.15288838
  3.7235189   2.95665859  1.43415829  4.67755695  2.47122731  4.27130164
  1.90232171  0.97640793  2.9878023   0.38856739  2.42497378  4.76409813
  0.37391265 -0.75895861  1.5330697   0.95776467 -1.82795386  2.27768227
  3.60435931 -1.29115986  0.16143348 -1.60982776 -1.89311832  1.72326649
 -1.89257755 -0.62824376  0.97827286  1.09663399  1.67463107  1.14842597
 -0.59675701 -0.69488226  2.08472798 -1.37617122  0.97509488 -0.78447101
 -1.66418163 -1.30692186  3.52315738  2.13932916  1.02575226  3.21822066
  3.11800606  3.14967611  3.00308075 -1.2496673   1.00405374 -1.84286926
 -0.04042662 -0.89313242  4.05163474  3.5342759  -1.77901089  0.50378369
  1.64527785  4.89515105  0.97769774  3.46789566  1.58052834 -1.66601862
  3.79944451  1.46785539  2.43442109  0.70817196  4.73522284  3.3869134
  1.07514272 -1.73645332 -1.44739472  2.323479    3.45366968 -0.55184505
 -0.9140461   0.1966022  -1.29914729  3.35319498  3.79536254  3.33165376
  2.26143055 -1.32684752 -1.22530666 -1.39626063 -0.92651549  3.35671237
  4.81462062  2.46288284 -1.46607685 -1.24090241 -1.23294568 -1.56912698
  4.66831624  0.686041    3.98398852  4.86627798  1.96640792  0.66791183
  0.68441271 -1.05892421  1.73838562  2.73217607  1.2651706   1.76595343
  1.60877727  0.82040813 -0.35654084  2.52336686  0.43519655  1.8715501
  2.12820366  0.36791691  2.42290218 -0.52588172  1.44906508 -1.7500195
 -1.41940761  3.96755332 -0.5949305   4.09569082  4.52380941  0.87819477
 -0.67588255  0.41031717  3.21683854  1.79369673  1.76690939  0.60897279
  1.10421319 -1.35793146  2.04972414  3.67168613  4.46844628  3.72651001
  4.29441764  0.18773032 -1.32775352 -1.21504872  0.9350142   3.70619763
 -1.82898508  1.25402563  2.74155677  3.12378277  0.9264476   1.94123802
  1.43588226  0.53346701 -0.27034021 -1.54310025  0.72451672  0.39283582
  2.53052477  4.78050882  0.5693789  -1.62031418  1.66100059  3.68058857
 -1.25957808 -1.31662463  1.4699022   4.67639196  1.84246256  2.64608802
  2.9200593  -1.0184883   0.49723891  4.63917008 -0.22944297  2.6075482
  2.5109707  -1.23834958  4.54371278 -1.01375977 -1.56252096  1.24928323
  3.45186481 -0.30351365 -0.46686639 -1.09470559  0.83014114 -0.144043
 -1.19644077 -1.47464543  2.62050936  2.08602961  1.23067488 -1.48132675
  4.62337835 -1.37620384  1.72331331  4.33164601  0.55959691  0.46032144
 -0.52775897  1.19959107 -1.02117185  2.40922543 -0.44483116  0.70498949
  1.96514103  4.4495707   2.95040408  1.57733798 -1.57470159 -1.69733233
  4.01710438  2.60413658 -0.22550645  0.67941362 -1.35444006  4.952672
  2.86155979  3.27553638 -0.45252359 -1.46637225 -1.88148097  4.22372506
  2.23996045  2.75570493 -1.51673753  3.4003881   1.45530423  4.67629357
 -1.4267469  -0.15661376  4.26480937  3.40549474  2.64100016  1.44626742
 -0.43850585  0.54809466  1.89710111 -0.4707166   1.72020357 -1.1635945
 -0.31850855  3.73205578 -1.92175145  2.98156359 -1.47998029  3.75055241
  1.50700641  0.37077133  3.41436594  1.18373487  2.72717927 -1.06580755
  4.68317493  4.76346616 -1.67187272  2.85749249  0.27442237  0.34543127
  4.18961343 -0.43303749  1.37323187  1.65771358 -0.23281606  3.33010106
  1.1680405   4.56784211  4.65576685  3.08562824  1.64690956 -1.64832303
  3.18986631  4.47901372 -1.881509    4.90836645  3.57186874  4.24964561
  2.63920783  0.80456612  1.35485896  0.39726432  4.19705782  3.78980207
  2.75431755  0.32534877  0.92899073  1.0545013   2.8405971  -0.54969927
  2.59748661 -1.82639374  4.31531677  4.10915768  0.60597888  4.720314
  3.9032133  -1.1161665   0.30358842 -1.68297135  2.93634656  3.39253599
  3.4118846  -1.00894065  4.48253461  0.00607812 -1.06529139 -1.13834022
 -0.69043599  2.09556928  1.07403303  0.08656396 -0.44438455  4.1159228
 -0.26274868  0.726663    0.89516659  3.02940077 -0.67746423  2.81075863
  4.51493841  4.32368249  4.05475258 -0.6625652   3.02213619  2.88882912
  2.82893553 -1.47980677 -0.65064965  2.01309892  2.51680344  4.07505282
  4.39429796  2.28799247 -0.37793228  0.04428675  0.42826964 -0.01812578
  0.52851423  0.61199718 -0.38510949  3.85387998  1.38077614 -0.59861592
  4.68491359  1.44018285  4.67289519  2.19207037  2.58376511  1.84765476
  4.90181609  4.77290842  4.27167717  0.52644165 -0.54771385  3.34858939
  3.54218152 -0.93237623  1.52716301 -1.14290007  1.15658494 -1.77502861
  1.22382535  3.46178443  0.78937546  0.79004838  0.6243222   2.61050885
  3.51820221 -1.21950125 -0.83312778  0.20735181  1.56611332 -0.85667995
  1.69705403  1.79250436  1.96903859  1.27643386 -0.23445677 -0.97903593
  4.12677911  3.96315627 -1.86950642 -1.29623442  3.31047806 -0.08092786
  3.9887627   3.85207617  3.5673641   3.38817879 -1.1652378   2.27026367
  4.2271716   4.48743313 -1.70614933  1.81685179  1.79079263  2.42254017
  2.53963764  3.71728123 -1.65627035  2.24365225  0.29033687  0.20769354
  0.63696029  1.43902549  0.67961565  3.32610748  1.4121958   1.67378538
  2.83470997  2.85903009  0.49535456  2.94691564  4.38024872  3.25363266
  2.41308879  1.58569795  2.62048974  3.4257214   3.50321313 -0.22228553
  1.76233003  3.30861372 -0.57262454  2.22256903  0.87292245  4.34319237
  3.01712994 -1.14630122 -1.65121707  1.95020608 -0.28311276 -0.22659966
  0.72512378  3.66262655  2.82435783  1.9872588   0.41334608 -1.06861851
  2.73949083  2.96750474 -0.37920308 -0.36533118 -0.04380134 -1.47110536
 -1.89098766  2.66551809  0.0453812   2.75416909  2.20421314  0.95610241
  1.05898135 -0.30602204  3.42416265  4.48111459 -1.21250398  4.19118372
  3.53293866  4.48819639 -0.54357528 -1.97449018  3.72346197  0.66831669
  3.31145767 -1.6381453   0.94103169  1.94447431 -1.78491314  0.69813676
```

```
 4.78478176  2.64643045 -0.81119251 -0.69293542  4.11997823  1.18103262
 3.60781221  1.32409046 -0.73398348 -0.28780341  3.34243873  1.91667172
 1.18159692 -0.10324218 -1.58727569 -1.51853486  1.38837932  2.08971366
 0.19410193  2.13653567  0.27117566  3.33242973 -1.91318847  4.45370608
 0.19128331 -0.73742552 -1.85219159  0.50341595 -1.89536732  2.63075009
 1.33640537  3.24288374  0.29255559  3.66135661  3.09128713  0.80289172
 4.29966913  0.13753378 -1.98405366  4.76222518  4.31529536 -0.71046947
 1.54023685  1.20271489  2.56691526  1.47288419 -0.96344105  4.8206923
 2.00021148  4.58948415 -0.42989112  0.26457618  4.67277717  1.63976296
-0.73634217 -1.80523353  2.85745361  3.34884201  4.33147178 -1.98779781
 4.42273904  3.6589149  -1.62476272  2.89434321  3.50206263  2.37257118
 4.40475819  4.47570612  4.76776332  3.80694529 -1.02800892  2.76054992
-0.37985165  2.89294883  1.26511612  0.06700568  2.57730344 -1.19440362
 4.33841815  1.14025378  0.33800418 -1.35338156  1.87385139  3.58085187
-1.45304157  0.2268498  -1.72259329  3.56795141  2.92250277  1.9741994
 4.205105    1.86106301  4.23755341 -1.51954827 -1.32650749  3.03649191
 0.12852233  3.61764443  1.26119924 -0.47150364  2.53233688  0.23403892
 4.18887712  3.08938892  0.19239832 -0.99606447  2.9936364   3.95932732
 4.00621613  2.43423948  1.18039283 -1.69635378 -1.68457643  0.1641055
 1.18456628  2.63474804  3.79141126  3.50198092  2.80082815  0.45121429
 3.5838873   2.54221706  1.30021153  2.38601202 -0.51326484 -1.07345381
-1.16876389 -1.27908151 -1.0321661   1.12701823 -0.37811187  3.56995119
 0.67802182  4.10597241  1.19377643  4.95997678  4.38273428  3.24979379
-1.51467871 -1.33273707  2.76801893  4.34743609  0.14779383  4.64903001
 1.16616605  1.39241847  4.08914484  2.71059564 -1.36105619  4.44687036
-1.16429557  2.88595479  0.29159503 -0.41289603  1.40612853 -0.5719092
 1.71251086  1.79191826  0.45218248 -1.88454234 -0.51748556 -0.10919733
 2.99042064  3.87960484 -0.4763365   0.79291494  3.61881606 -1.01408551
 1.47244241  0.0698617  -0.31624673  0.30364898  1.52259784  2.77437235
 0.44045863  3.17476655  2.74183387  3.27991401  4.39700922 -1.66620999
 0.96644951  0.74321612  3.50363106  4.36263835 -1.75321254 -0.88795085
-0.73447439  2.08939235  1.54950954 -1.13177195  3.08109897  2.39744788
-0.89210106  1.80968281 -0.24316483  4.10665417 -0.14779883 -0.45280557
 2.69851306  2.37168568  1.79090034  1.22269458 -0.55673914  4.09712149
 3.38574135 -1.44541829 -0.42595069  0.3825223   0.20169564 -1.55870649
 0.16090142  0.02821181  3.03090947  1.51013045  2.32551085  2.87476932
 1.53447861  0.33002836  3.9029573   3.54862835  4.12775308  2.51344996
-1.63381719 -1.87900063 -1.65446396  3.37810297  4.28590799  4.25187015
 1.62966314  1.74502132  2.9824214   3.87375703 -1.46175142  2.30684504
 2.48860937  3.95445413  4.16088179 -0.36435402 -0.95641676 -1.89549093
 1.69352169 -1.42216203  3.28638533  3.66926438 -1.57134367  4.86541703
 1.14350454 -0.30472861  3.81539189  2.64103118  1.13974523  4.29775577
-1.86922322  2.24410509  3.67589511 -0.3748163   4.98633032  2.39374999
 1.97759204  1.04366332  0.59314015 -1.42252969  1.9494341   4.26624462
 4.17761072 -0.09193143  3.97546085  2.5280174   2.01885744 -1.20881458
 2.50179585  4.81909362  0.52226199  1.50551614  3.929883    3.89660395
 1.31320423  1.06959916  1.54743617  0.79858527  1.56610113  3.24737506
-0.48389865 -0.5243353  -0.66077586  2.95420581  3.97244036  0.04683744
 0.92217994  4.93813189 -1.09525964 -0.18959074 -0.02329482 -1.84295624
 4.902746    0.45348295  3.29411244  4.51282152  0.74628793  0.48868756
 2.70607886  4.86490983  4.58961334  4.16119433  4.20736673  3.89586137
-1.49794489 -0.90634006  4.68354475 -1.8963266   2.74695488  3.70661243
 4.34904419  3.60891343 -0.71115864  2.00281345  3.91017687  4.79083606
 1.75504109 -0.72953271 -1.94470255  3.57170258  2.31770855  0.20023821
-0.24737482  2.28141688 -1.81029781  0.02602673 -1.93483034  1.54567757
-0.07015399  0.60517121 -1.28677477  1.6225301   2.97518223 -1.56072386
 2.66600344  1.48017039 -1.73236081 -0.27171219  2.21606164  4.22982408
 0.56996751  4.94351908  3.03754516 -1.8775004   4.0273634   3.62691754
-1.10210215  4.24244654  4.18967262 -0.17979236  2.28863322 -1.11109766
 2.87052604  4.93202755  4.54376969  4.56525777  3.619656    2.47008608
 1.15169484 -1.95308058 -0.16604931 -1.60266972  2.86819036  0.64780625
 4.04264609  2.6785706   4.37726905  4.04084507  4.87496574  0.59084926
-0.65192388 -1.25740823  0.69850374  1.63639584  1.64426889 -0.8563514
 2.18492788  0.17364558 -0.43042076  3.31138711  4.32455736 -1.99943118
 0.93987087  0.25032178  3.64033676  1.59012683 -1.11978535  1.43621071
 4.56275407  1.73722476  4.39210897 -1.04208998 -0.7398546  -1.25113285
 4.89951936  3.35056526  1.21705843 -1.52230738 -0.85185633  1.83174161
-0.39783145  1.54030416  0.04972766  2.72866791  3.06013525  1.85890148
 0.99750378 -0.63990134  1.25299557 -0.87468718  2.97627543  0.55936602
 0.18146095 -0.63041467  0.28161553  2.9467452   0.4026108  -1.09346561
 3.61295281 -1.55075658  3.94143327  1.78761491 -1.98609687  3.24706696
 0.54537893  3.95794652  1.25120958 -1.02163952  0.90201323  3.33362568
-0.4474226  -0.2796221  -1.80091384  1.61193076  1.7719493   0.85317588
 4.16878488  0.70720761  3.30205068  2.9761882  -1.12166115  2.54078908
 3.09024385  4.67407535  4.95003734  3.31418504  1.73129679  0.91349273
-1.44971612  4.01323435  0.04367766  3.28996789  1.4408434  -1.39248903
 0.20327581  3.57025052  2.08885856  3.49015216  3.99172474  1.95454886
 4.78957703 -0.42405268  3.71866501 -0.61264867  4.93537979  4.30774832
-0.05588545  2.59142424  0.92696566 -0.85246866 -1.00636259 -1.00416361
 0.76692622 -1.73116616  0.5615049   1.92037596 -1.86241323 -0.51326778
-1.1734484  -1.99768343  4.41380572 -0.25117396  0.66616402 -0.14273677
 0.12383402  1.49413883  4.46813919 -1.44834426 -1.32406592  4.20462047
 4.80146952  4.56390693  0.76845199 -1.65596206]
```

Let's also do the histogram of x_samples to make sure they are distributed the right way:

```
fig, ax = plt.subplots()
ax.hist(x_samples, density=True, alpha=0.25, label="Histogram")
ax.plot(xs, X.pdf(xs), lw=2, label="True PDF")
ax.set_xlabel("$x$")
ax.set_ylabel("$p(x)$");
```



Questions

- Repeat the code above so that the random variable is $U([1,10])$.

By Ilias Bilionis (ibilion[at]purdue.edu)