1. Use the inverse transform method to generate random variates of X.

   Suppose the random variable X has the pdf

$$f_X(x) = \begin{cases} 2xe^{-x^2} & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

   Solution. We start by getting the cdf of X.

$$F_X(x) = P(X \leq x) = \int_0^x 2te^{-t^2}\, dt$$

   Solving the integral using u-substitution, we obtain the cdf.

$$F_X(x) = 1 - e^{-x^2}, x \geq 0$$

   The next step is to get the inverse of the cdf over (0,1].

$$F_X^{-1}(x) = \sqrt{-ln(1-x)}, 0 < x \leq 1.$$

   Since $x \in (0, 1]$, the value of $ln(1 - x)$ will be less than or equal to 0. Thus, the value of the inverse cdf will still be real-valued.

   Given that we have the inverse cdf of X, we can obtain random variates that have the same pdf as X using the inverse transform method. The first step is to obtain uniform random variables $u \sim U(0, 1)$. Then our algorithm should return $F_X^{-1}(u)$. Note that in our cdf, $1 - u$ can be simplified to u because both u and $1 - u$ are both uniform random variables from (0,1).

   The code for the algorithm is in the attached jupyter notebook.

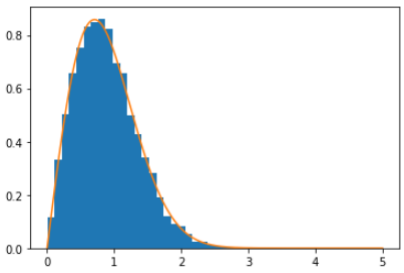

   Figure 1: Histogram and target pdf (Inverse Transform Method)

   We can see from figure 1 that the random variates generated by the inverse transform method closely match the pdf of X (the orange line).

2. Generate random variates of X via the generalized acceptance-rejection method with a Exp(1) proposal distribution. Find, with proof, the best value of C to use for the algorithm, and state the efficiency of the resulting algorithm.

   Solution.

Let $g(x) = e^{-x}$, $x \geq 0$ be the proposal distribution. We want to find the optimal (smallest) value of the constant C such that $f(x) \leq Cg(x)$ on $[0, \infty)$.

Rearranging, we have

$$\frac{f(x)}{g(x)} \leq C$$

for all $x \in [0, \infty)$.

Let $h(x) = \frac{f(x)}{g(x)}$. We want to find the maximum value of $h(x)$ over $[0, \infty)$.

First, we get the first and second derivatives of $h(x)$.

$$\begin{aligned} h(x) &= \frac{2xe^{-x^2}}{e^{-x}} \\ &= 2xe^{-x^2+x}. \end{aligned}$$

By Chain Rule, we find $h'(x)$ and $h''(x)$.

$$\begin{aligned} h'(x) &= 2xe^{-x^2+x}(-2x+1) + 2e^{-x^2+x} \\ &= -4x^2e^{-x^2+x} + 2xe^{-x^2+x} + 2e^{-x^2+x} \\ &= (-4x^2 + 2x + 2)e^{-x^2+x}. \end{aligned}$$

$$\begin{aligned} h''(x) &= (-4x^2 + 2x + 2)e^{-x^2+x}(-2x+1) + e^{-x^2+x}(-8x+2) \\ &= (-4x^2 + 2x + 2)(-2x+1)e^{-x^2+x} + (-8x+2)e^{-x^2+x} \\ &= (8x^3 - 8x^2 - 10x + 4)e^{-x^2+x}. \end{aligned}$$

First we do the first derivative test. We equate $h'(x) = 0$ and solve for the critical points. We have $(-4x^2 + 2x + 2)e^{-x^2+x} = 0$ and we obtain the zeros $x = 1, -\frac{1}{2}$. Since the pdf of X has support $[0, \infty)$, we only consider $x = 1$ as a critical point.

By the second derivative test, we get that $h''(1) < 0$ so $x = 1$ is an absolute maximum.

Since we have obtained the maximum value of $h(x)$ at $h(1) = 2$, we choose $C = 2$ as the optimal value. The efficiency of the accept-reject method is $\frac{1}{C} = \frac{1}{2}$.

The implementation of the accept-reject method is in the attached jupyter notebook.

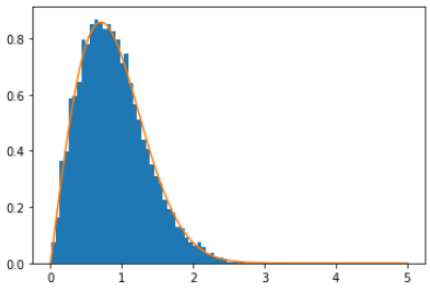

Figure 2: Histogram and target pdf (Accept-Reject Method)

Generalized Acceptance-rejection method.

The goal is to generate test points over a proposal density $g(x)$, where for some $C \in \mathbb{R}, Cg(x) \geq f(x)$.

Step 1: Generate x' from the proposal density $g(x)$.
Step 2: Generate y' uniformly from 0 to $Cg(x')$.
Step 3: If $y' \leq f(x')$, return x'.

3. We are given a snakes-and-ladders game.

   Let X be the number of tosses required to reach the finish using a fair six-sided die. (You can use **random.randint**). Use a simulation to generate a 95% confidence interval for $E[X]$. Choose any number of runs N large enough so that the absolute width of the confidence interval is at most 0.5.

   Solution.

   Code Explanation.

   - First, we set the initial state to be 1, corresponding to the starting point of the player at tile 1. The while loop for the simulation will stop once the player lands exactly at tile 20.
   - The function **random.randint(1,6)** is used to simulate the dice roll.
   - If the dice roll causes the player to go over 20, the player must go back the number of tiles the player is over 20.
   - There are 2 if statements each for the ladders and the snakes in the game.

   Estimating $E[X]$.

   Let X be the number of tosses required to reach the finish using a fair six-sided die.

   We set the number of simulation runs to be $N = 100,000$. Our simulation function saves the number of tosses required to reach tile 20 in random variables: $X_1, X_2, X_3, ..., X_N$ for each simulation run.

   First, we calculate the sample mean using the formula:

$$\bar{X} = \frac{1}{N}\sum_{i=1}^{N} X_i = 13.542$$

   Next, we calculate the sample variance:

$$S^2 = \frac{1}{N-1}\sum_{i=1}^{N}(X_i - \bar{X})^2 = 86.085$$

   The 95% confidence interval for $E[X]$ is

$$(\bar{X} - Z_{0.025}\frac{S}{\sqrt{N}}, \bar{X} + Z_{0.025}\frac{S}{\sqrt{N}}) = (13.485, 13.600)$$

   The absolute width of the confidence interval is

$$2Z_{0.025}\frac{S}{\sqrt{N}} = 0.115$$

4. A mouse is let loose in the following maze.

   From each compartment, the mouse chooses one of the adjacent compartments with equal probability, independent of the past. The mouse spends an exponentially distributed amount of time in each compartment. The mean time spent in each of the compartments

1, 3, and 4 is two seconds; the mean time spent in compartments 2, 5, and 6 is four seconds. Let $\{X_t, t \geq 0\}$ be the Markov jump process that describes the position of the mouse for times $t \geq 0$, where t is in seconds. Assume that the mouse starts in compartment 1 at time $t = 0$.

Solution.

(a) **Code Explanation.**

- The main simulation function has two parameters: initial (set at 1) for the initial room of the mouse and end_time (set at 10800 seconds) for the total duration of the simulation which is given to be 3 hours.
- The while loop for the function stops when the current time is equal to or greater than the end time.
- During each iteration of the while loop, there are six if statements for each of the possible current rooms of the mouse. Since the mouse chooses one of the adjacent compartments with equal probability, the **rand.choices** function is used to determine the next room.
- The holding time of the mouse in the room follows an exponential distribution according to the problem. The **rand.expovariate** function is used with parameter λ. The value of λ is equal to the reciprocal of the mean time in seconds that the mouse spends in the room.
- The total amount of time that the mouse spends in each of the six rooms is stored in the list T, where $T[i]$ is the total time spent by the mouse in room i.

(b) Let T_r be the total time spent by the mouse in room r.

(c) Choose $N = 1000$ as the number of simulation runs.

Estimating $E[T_r]$.

The formula for the 95% confidence interval for $E[T_r]$ is given by

$$(\bar{T}_r - Z_{0.025}\frac{S}{\sqrt{N}}, \bar{T}_r + Z_{0.025}\frac{S}{\sqrt{N}}),$$

where $\bar{T}_r$ is the sample mean of T_r and S is the sample standard deviation.

The estimated relative error RE is given by

$$RE = \frac{S}{\bar{T}_r\sqrt{N}}$$

Table 1 shows the estimation for $E[T_r]$ for the simulation runs. Note that the units of the values are in seconds.

Room	$\bar{T}_r$	S^2	95% C.I.	RE
1	1274.619	11715.583	(1267.910, 1281.328)	0.003
2	1278.020	19249.208	(1269.421, 1286.619)	0.003
3	1272.151	6595.518	(1267.118, 1277.185)	0.002
4	1902.900	6972.387	(1897.725, 1908.076)	0.001
5	2539.947	21621.098	(2530.833, 2549.061)	0.002
6	2535.521	21022.613	(2526.534, 2544.508)	0.002

Table 1: The Sample Mean, Sample Variance, 95% Confidence Interval, and Relative Error