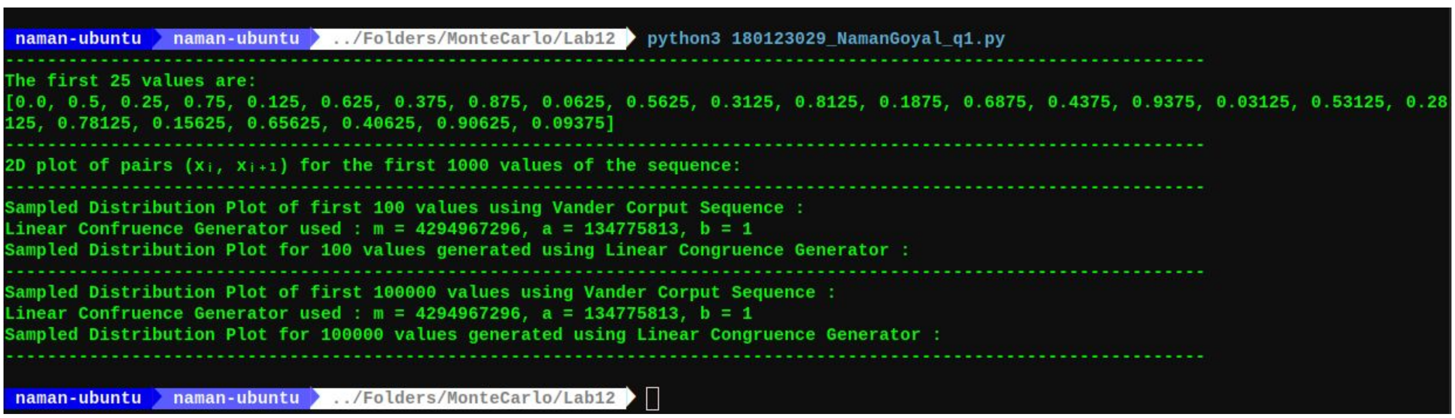
Monte Carlo Simulation Assignment 12

Name: **Naman Goyal**
Roll No: **180123029**

Problem.

- To execute the .py file, run the following command:
  $ python3 180123029_NamanGoyal_q1.py (for Ques 1)
  $ python3 180123029_NamanGoyal_q2.py (for Ques 2)

Output for Ques.1

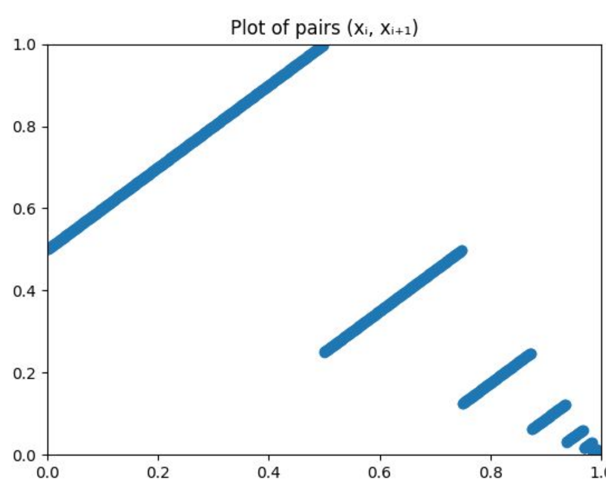


```
naman-ubuntu  naman-ubuntu  ../Folders/MonteCarlo/Lab12  python3 180123029_NamanGoyal_q1.py
--------------------------------------------------------------------------------------------------------------
The first 25 values are:
[0.0, 0.5, 0.25, 0.75, 0.125, 0.625, 0.375, 0.875, 0.0625, 0.5625, 0.3125, 0.8125, 0.1875, 0.6875, 0.4375, 0.9375, 0.03125, 0.53125, 0.28
125, 0.78125, 0.15625, 0.65625, 0.40625, 0.90625, 0.09375]
--------------------------------------------------------------------------------------------------------------
2D plot of pairs (xᵢ, xᵢ₊₁) for the first 1000 values of the sequence:
--------------------------------------------------------------------------------------------------------------
Sampled Distribution Plot of first 100 values using Vander Corput Sequence :
Linear Confruence Generator used : m = 4294967296, a = 134775813, b = 1
Sampled Distribution Plot for 100 values generated using Linear Congruence Generator :
--------------------------------------------------------------------------------------------------------------
Sampled Distribution Plot of first 100000 values using Vander Corput Sequence :
Linear Confruence Generator used : m = 4294967296, a = 134775813, b = 1
Sampled Distribution Plot for 100000 values generated using Linear Congruence Generator :
--------------------------------------------------------------------------------------------------------------

naman-ubuntu  naman-ubuntu  ../Folders/MonteCarlo/Lab12
```

- Firstly plot the first 25 values of the **Van der Corput Sequence** using the radical function $x_i := \phi_2(i)$. The **Radical Inverse** function is implemented.

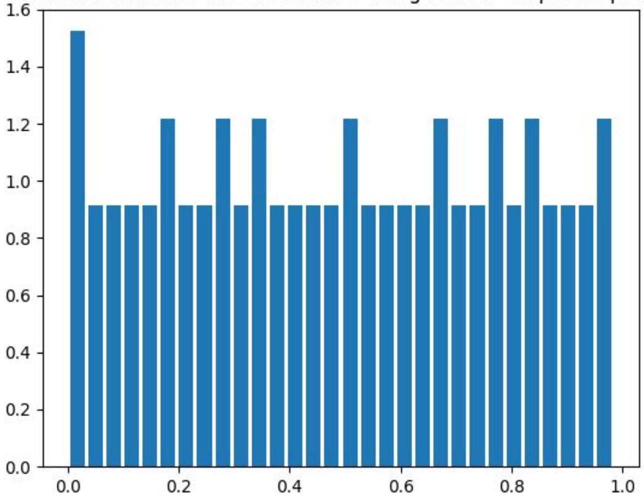
FIRST 25 VALUES:
[0.0, 0.5, 0.25, 0.75, 0.125, 0.625, 0.375, 0.875, 0.0625, 0.5625, 0.3125, 0.8125, 0.1875, 0.6875, 0.4375, 0.9375, 0.03125, 0.53125, 0.28125, 0.78125, 0.15625, 0.65625, 0.40625, 0.90625, 0.09375].

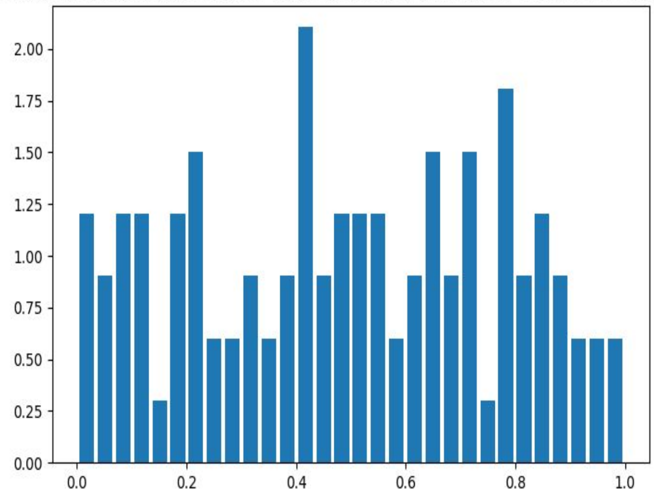
- **2D Plot** for (x_i, x_{i+1}) for first 1000 values.
- We can observe that the length of the segments is decreasing exponentially.



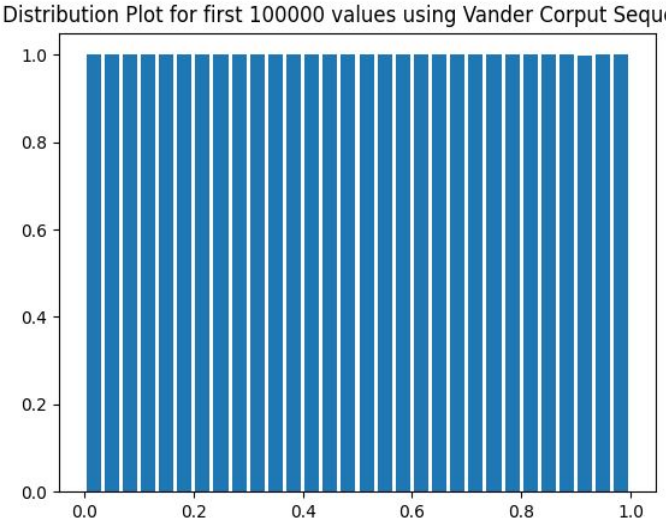
- We now compare the plots of the **Van der Corput Sequence** and values generated by the **Linear Congruence Generator**.

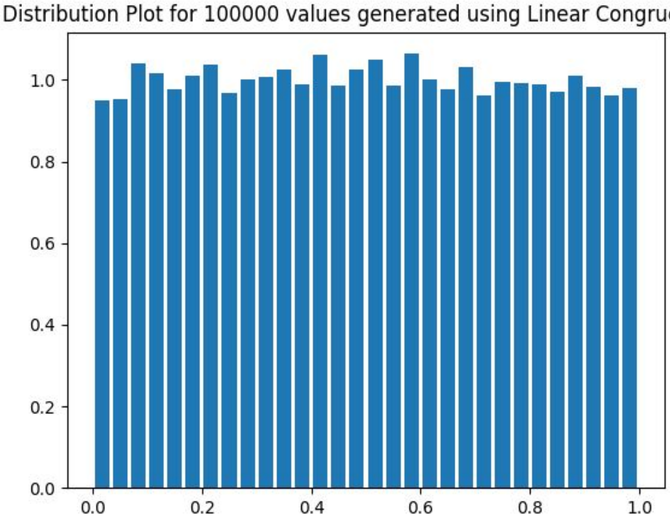
For N = 100





For N = 100000





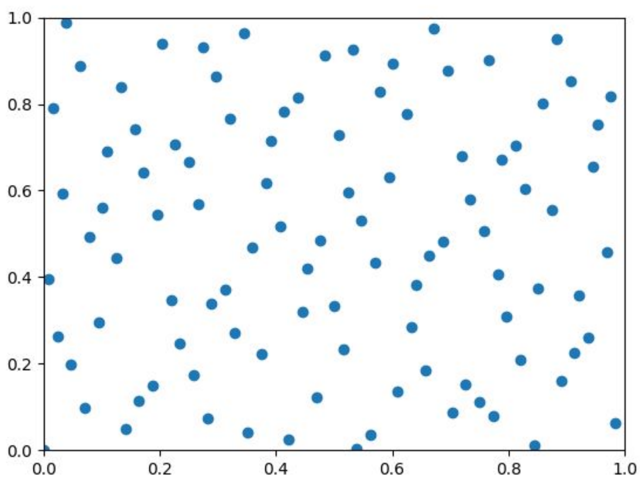
- Van der Corput Sequence is **more uniform** as compared to LCG. The distribution plot approaches the same value as the value of N increases. More uniformity is attained in the Van der Corput Sequence compared to LCG.

Output for Ques.2

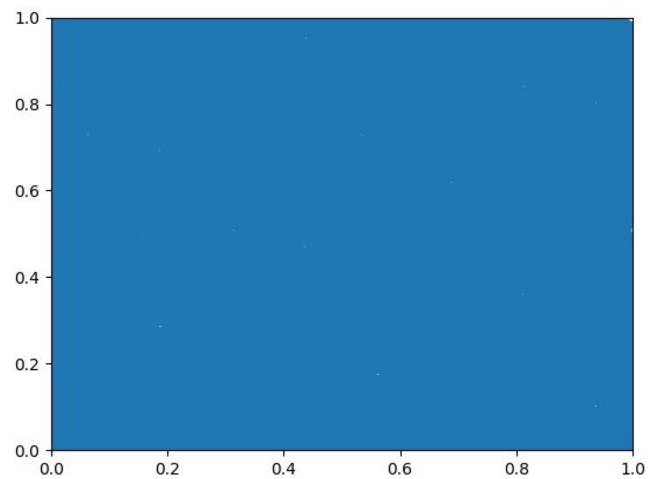
```
naman-ubuntu  naman-ubuntu  ../Folders/MonteCarlo/Lab12  python3 180123029_NamanGoyal_q2.py
------------------------------------------------------------------------------------------------
Halton Sequence with base 2 and base 3
------------------------------------------------------------------------------------------------
Calculating for 100 values
------------------------------------------------------------------------------------------------
Calculating for 100000 values
------------------------------------------------------------------------------------------------

naman-ubuntu  naman-ubuntu  ../Folders/MonteCarlo/Lab12
```

- **Hamilton Sequence** is generated using $x_i := (\phi_2(i), \phi_3(i))$.

For N =100



For N = 100000



- Since the base values are taken to be 2, 3 which are coprime. So many distributed values are being observed. So as **N increases** we can see that plot becomes **more uniform**.
