

# Generador de números aleatorios

Decidimos trabajar con el generador de números aleatorios ya que es un tema muy interesante que es muy usado en la programación.

Nuestro equipo está conformado por: **Jorge Reyes, Jorge Vázquez y Augusto Reyes.**

## Descripción

En la computación es básicamente imposible obtener un número 100% aleatorio. Debido a esto los programadores que requieren usar números “aleatorios” se las ingenian para poder generarlos de alguna forma.

En este caso, utilizamos el **método congruencial lineal**. El cual toma la fórmula:

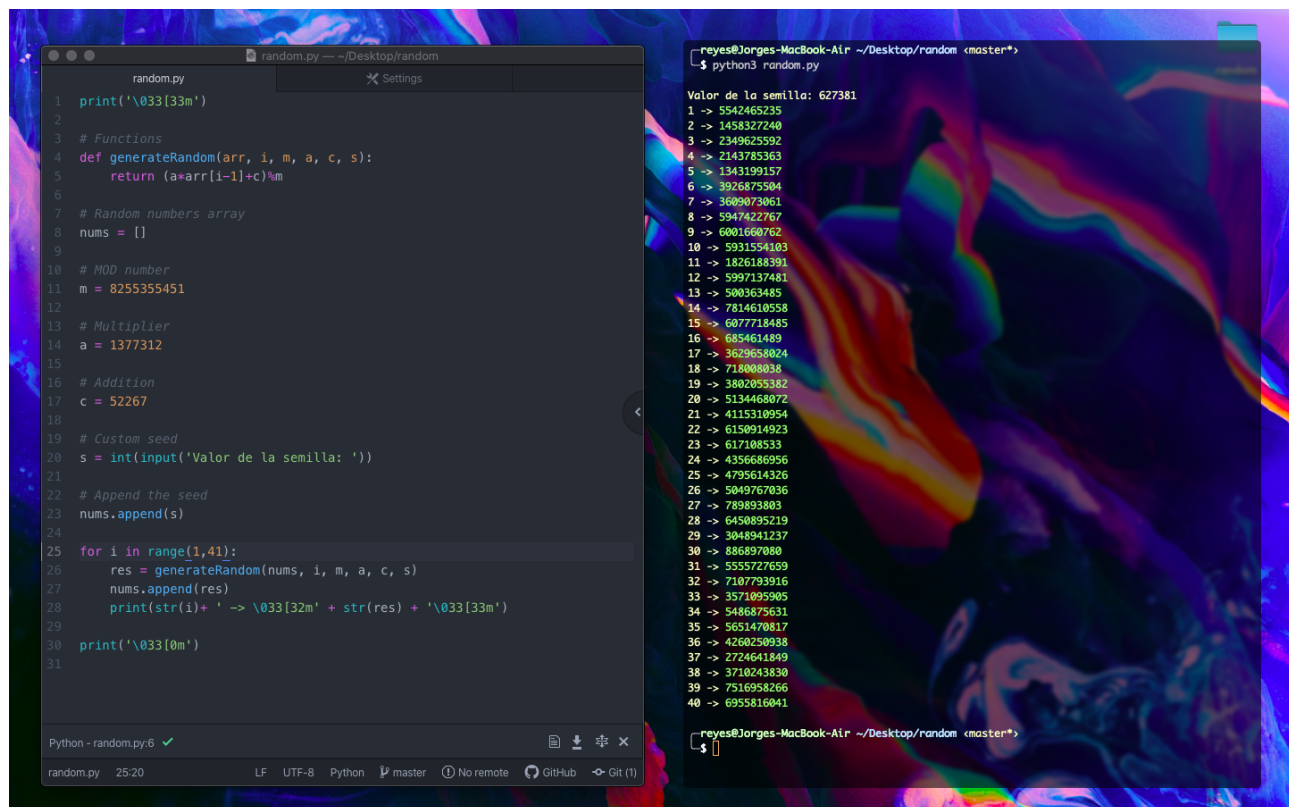
$$x_n = (ax_{n-1} + c) \bmod m$$

Dónde ***m*** es el módulo, ***a*** es el multiplicador, ***c*** es el incremento y ***s*** es la semilla.

Los métodos para generar números aleatorios utilizan el concepto de semilla, el cual es un número *arbitrario*, y con sólo ese número se pueden generar números *pseudo-aleatorios*. Mientras más grandes sean los números utilizados para ***m*** y ***a*** mejores resultados obtenemos.

Se les llama números *pseudo-aleatorios* porque en realidad no son aleatorios. Si conocemos la fórmula usada para generarlos, junto con la semilla, podríamos predecir que números se van a generar. Es meramente una función matemática.

## Capturas de pantalla del programa



The image shows a code editor with two panes. The left pane displays the source code for a Python script named `random.py`. The script defines a `generateRandom` function and a loop that generates 41 random numbers based on a custom seed. The right pane shows the terminal output of the script, displaying the seed value and the sequence of 41 generated random numbers.

```
random.py
1 print('\033[33m')
2
3 # Functions
4 def generateRandom(arr, i, m, a, c, s):
5     return (a*arr[i-1]+c)%m
6
7 # Random numbers array
8 nums = []
9
10 # MOD number
11 m = 8255355451
12
13 # Multiplier
14 a = 1377312
15
16 # Addition
17 c = 52267
18
19 # Custom seed
20 s = int(input('Valor de la semilla: '))
21
22 # Append the seed
23 nums.append(s)
24
25 for i in range(1,41):
26     res = generateRandom(nums, i, m, a, c, s)
27     nums.append(res)
28     print(str(i)+ ' -> \033[32m' + str(res) + '\033[33m')
29
30 print('\033[0m')
```

```
reyes@Jorges-MacBook-Air ~/Desktop/random (master*)
$ python3 random.py
Valor de la semilla: 627381
1 -> 5542465235
2 -> 1458327240
3 -> 2349625592
4 -> 2143785363
5 -> 1343199157
6 -> 3926875504
7 -> 3609073061
8 -> 5947422767
9 -> 6001600762
10 -> 5931554103
11 -> 1826188391
12 -> 5997137481
13 -> 500363485
14 -> 7814610558
15 -> 6077718485
16 -> 685461489
17 -> 3629658024
18 -> 718080036
19 -> 3801055382
20 -> 5134468072
21 -> 4115310954
22 -> 6150914923
23 -> 617108533
24 -> 4356686956
25 -> 4795614326
26 -> 5049767036
27 -> 789893803
28 -> 6450895219
29 -> 3048941237
30 -> 886897080
31 -> 5555727659
32 -> 7107793916
33 -> 3571095905
34 -> 5486875631
35 -> 5651470817
36 -> 4268250938
37 -> 2724641049
38 -> 3710243830
39 -> 7516958266
40 -> 6955816041
```



```
reyes@Jorges-MacBook-Air ~/Desktop/random <master*>
$ python3 random.py

Valor de la semilla: 9263481
1 -> 4179423544
2 -> 1653213656
3 -> 7121885570
4 -> 3834584652
5 -> 6274357735
6 -> 837876532
7 -> 1263598961
8 -> 742111632
9 -> 7187046239
10 -> 6436818559
11 -> 6926157716
12 -> 4144788609
13 -> 6240770265
14 -> 3424343296
15 -> 5480330907
16 -> 6372721421
17 -> 7940017654
18 -> 3973823164
19 -> 985315298
20 -> 5211892255
21 -> 6945286483
22 -> 7585878772
23 -> 3663444864
24 -> 299504831
25 -> 7996670971
26 -> 3647444216
27 -> 4214060825
28 -> 296830999
29 -> 7184302533
30 -> 195061394
31 -> 6366303302
32 -> 1418034096
33 -> 6873573737
34 -> 3832875784
35 -> 5406258454
36 -> 5176998543
37 -> 1841104610
38 -> 2704843270
39 -> 2328858835
40 -> 4646805894

reyes@Jorges-MacBook-Air ~/Desktop/random <master*>
$
```

## Conclusión

En este trabajo aprendimos uno de los métodos para generar números *pseudo-aleatorios* como también construir el programa que efectúa las operaciones para generar los números.

Investigando y leyendo sobre el tema logramos entender la importancia y utilidad de los números *pseudo-aleatorios*. Así como también, logramos entender que ya teniendo nuestro programa que genera números *pseudo-aleatorios* podemos “limitarlos” para conseguir números en el rango que nosotros deseemos. Por ejemplo, si necesitamos números aleatorios entre el **0** y el **100** podemos aplicar al resultado final de nuestro generador de números una operación de *mod* **100** para que nos limite los resultados al rango deseado.