Estocasticos_Tarea2_Variables_aleatorias_Ibarra_Sergio_v2

March 2, 2023

[ ]:

0.1 Maestria en Ing. de Sistemas UNAM- Procesos estocásticos . Dra: Patricia Aguilar

0.2 Alumno: Sergio Ibarra R (414025796)

0.3 Problemas sobre la aplicación de conceptos de variable aleatoria. Marzo 2023

0.3.1 Problema 1

En cada uno de los siguientes enunciados, indique con una D si la variable aleatoria es discreta o con una C si es continua.

a) El tiempo de espera para un corte de cabello. ……(Continuo)
b) El número de automóviles que rebasa un corredor cada mañana. ……(Discreto)
c) El número de hits de un equipo femenil de softbol de preparatoria. ……(Discreto)
d) El número de pacientes atendidos en el South Strand Medical Center entre las seis y diez de la noche, cada noche. ……(Discreto)
e) La distancia que recorrió en su automóvil con el último tanque de gasolina. ……(Continuo)
f) El número de clientes del Wendy's de Oak Street que utilizaron las instalaciones. (Discreto)
g) La distancia entre Gainesville, Florida, y todas las ciudades de Florida con una población de por lo menos 50 000 habitantes. ……(Continuo)

[ ]:

0.3.2 Problema 2

En un estudio realizado en Estados Unidos en 2001 (USA Today, 6 de septiembre, 2001), 38% de los alumnos de cuarto grado de primaria no podía leer un libro apropiado para su edad. Los datos siguientes muestran el número de sujetos, por edad, que se identificaron como niños con problemas de aprendizaje que requieren educación especial. La mayoría tiene problemas de lectura que debieron identificarse y corregirse antes del tercer grado. La ley federal estadounidense actual prohíbe que la mayoría de los niños reciba ayuda adicional de programas de educación especial hasta que el retraso sea de aproximadamente dos años de aprendizaje, y por lo general eso significa hasta tercer grado o grados superiores.

Edad	Número de niños
6	37,369

Edad	Número de niños
7	87,436
8	160,840
9	239,719
10	286,719
11	306,533
12	310,787
13	302,604
14	289,168

Suponga que se desea seleccionar una muestra de menores con problemas de aprendizaje y que deben tomar educación especial a efecto de incluirlos en un programa diseñado para mejorar su capacidad de lectura. Sea una variable aleatoria que indica la edad de un niño seleccionado al azar,

a) Use los datos para construir la función de probabilidad para y trace su gráfica.
b) Calcule la edad promedio de los niños con problemas de aprendizaje..
c) Genere una muestra de las edades de 30 niños con problemas de aprendizaje y calcule la edad promedio. Muestre paso a paso su procedimiento realizado ya sea manualmente o con algún software, en cuyo caso es necesario mostrar la codificación.

a) Use los datos para construir la función de probabilidad para y trace su gráfica. Vamos a importar el documento donde se guardó la tabla con la información sobre la cantidad de niños en problemas de aprovechamiento en USA

[129]:
```
import pandas as pd

# Read the Excel file into a dictionary of data frames
df_importado = pd.read_excel(R'C:
 ↪\Users\sergi\OneDrive\Documentos\MIS_UNAM\Segundo_semestre\Estocasticos_UNAM\UNAM_Estocasti
 ↪xlsx', sheet_name=['Problema2'])
print(df_importado)
type(df_importado)
```

```
{'Problema2':    Edad  Número de niños
0     6            37369
1     7            87436
2     8           160840
3     9           239719
4    10           286719
5    11           306533
6    12           310787
7    13           302604
8    14           289168}
```

[129]:
```
dict
```

[130]:
```
df_USA_niños_escolar = df_importado['Problema2']
print(df_USA_niños_escolar)
type(df_USA_niños_escolar)
```

```
   Edad  Número de niños
0     6            37369
1     7            87436
2     8           160840
3     9           239719
4    10           286719
5    11           306533
6    12           310787
7    13           302604
8    14           289168
```

[130]:
```
pandas.core.frame.DataFrame
```

Vamos a "separar" la información de la edad y guardarla en una variable llamada df_USA_niños_escolar_edad

[131]:
```
df_USA_niños_escolar_edad= df_USA_niños_escolar['Edad']
df_USA_niños_escolar_edad
```

[131]:
```
0     6
1     7
2     8
3     9
4    10
5    11
6    12
7    13
8    14
Name: Edad, dtype: int64
```

[132]:
```
type(df_USA_niños_escolar_edad)
```

[132]:
```
pandas.core.series.Series
```

[133]:
```
df_USA_niños_escolar_edad[0]
```

[133]:
```
6
```

Vamos a "separar" la información del número de nuños por edad y guardarla en una variable llamada df_USA_niños

[134]:
```
df_USA_niños= df_USA_niños_escolar['Número de niños']
df_USA_niños
```

[134]:
```
0     37369
1     87436
2    160840
3    239719
4    286719
5    306533
6    310787
7    302604
8    289168
Name: Número de niños, dtype: int64
```

Transformando la data de numero de niños en una lista para poder sumar sus elementos usarlo para calcular el promedio de cada edad

[135]:
```
# Transform the series to a list
df_USA_niños_list = df_USA_niños.tolist()

df_USA_niños_list
```

[135]:
```
[37369, 87436, 160840, 239719, 286719, 306533, 310787, 302604, 289168]
```

[136]:
```
# Get the sum of all elements in the list
total_niños_USA = sum(df_USA_niños_list)
total_niños_USA
```

[136]:
```
2021175
```

Calculando las probabilidades por edad como:

$$P(x = edad) = \frac{\text{Numero de ninios en edad X} = \text{x}}{\text{Numero total de ninios}}$$

y las añadimos a la tabla llamada df_USA_niños_escolar

[137]:
```
df_USA_niños_escolar['Probabilidad_por_edad'] = (df_USA_niños /␣
 ↪total_niños_USA)*100
print(df_USA_niños_escolar['Probabilidad_por_edad'])
```

```
0     1.848875
1     4.325998
2     7.957747
3    11.860378
4    14.185758
5    15.166079
6    15.376551
7    14.971687
8    14.306925
Name: Probabilidad_por_edad, dtype: float64
```

[138]:
```
print(df_USA_niños_escolar)
```

	Edad	Número de niños	Probabilidad_por_edad
0	6	37369	1.848875
1	7	87436	4.325998
2	8	160840	7.957747
3	9	239719	11.860378
4	10	286719	14.185758
5	11	306533	15.166079
6	12	310787	15.376551
7	13	302604	14.971687
8	14	289168	14.306925

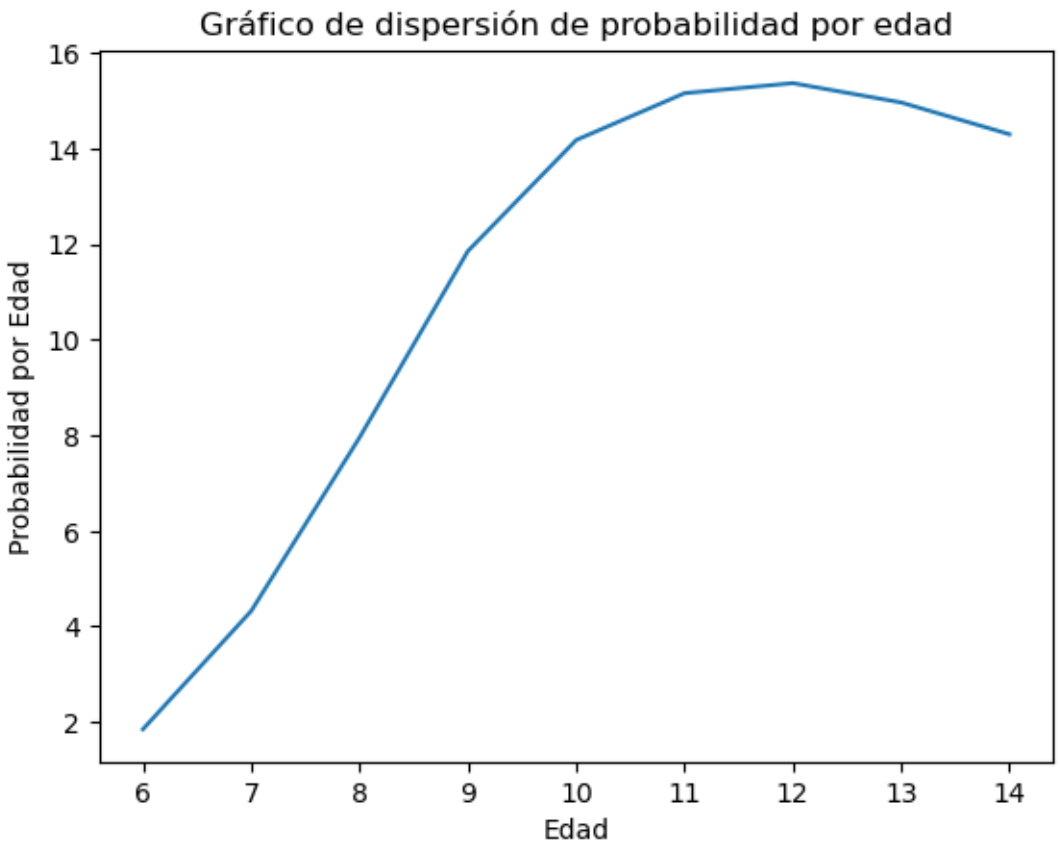
Por lo tanto la función de probabilidad de X sería

Edad	Valor de probabilidad
6	1.85
7	4.32
8	7.96
9	11.87
10	14.19
11	15.17
12	15.38
13	14.97
14	14.31

Y la gráfica de la función de probabilidad de X sería

[139]:
```
import matplotlib.pyplot as plt

# Assuming that df_USA_niños_escolar is a pandas dataframe
plt.plot(df_USA_niños_escolar['Edad'],␣
  ↪df_USA_niños_escolar['Probabilidad_por_edad'])
plt.xlabel('Edad')
plt.ylabel('Probabilidad por Edad')
plt.title('Gráfico de dispersión de probabilidad por edad')
plt.show()
```



b) Calcule la edad promedio de los niños con problemas de aprendizaje. La edad promedio se calculará como la esperanza de los valores de la función de probabilidad

[140]:
```
esperanza_edad =␣
↪round(sum(df_USA_niños_escolar['Edad']*(df_USA_niños_escolar['Probabilidad_por_edad']/
↪100)),2)
esperanza_edad
```

[140]:
```
11.0
```

[141]:
```
from IPython.core.display import HTML


texto = f'<span style="color:red">_La edad promedio de los niños con problemas␣
↪de aprendizaje__ es: {esperanza_edad} años</span>'

HTML(texto)
```

[141]:
```
<IPython.core.display.HTML object>
```

c) Genere una muestra de las edades de 30 niños con problemas de aprendizaje y calcule la edad promedio. Muestre paso a paso su procedimiento realizado ya sea manualmente o con algún software, en cuyo caso es necesario mostrar la codificación. Para poder generar números aleatorios en Python a partir de una x y una f(x) ambos deben estar en un formato de dato tipo 'lista'

[142]:
```
#Transformando los datos de la columna probabilidad en una tipo lista

df_USA_niños_probabilidad_list = (df_USA_niños_escolar['Probabilidad_por_edad']/
 ↪100).tolist()
df_USA_niños_probabilidad_list
```

[142]:
```
[0.01848875035560998,
 0.04325998490976783,
 0.07957747349932588,
 0.11860378245327594,
 0.1418575828416639,
 0.15166079137135577,
 0.15376550768736008,
 0.14971687261122862,
 0.143069254270412]
```

[143]:
```
#Transformando los datos de la columna edad en una tipo lista
df_USA_niños_escolar_edad_list = df_USA_niños_escolar_edad.tolist()

df_USA_niños_escolar_edad_list
```

[143]:
```
[6, 7, 8, 9, 10, 11, 12, 13, 14]
```

Ahora procederemos a generar los 30 numeros random, con la función de python que se llama random.choice y que toma 3 parametros: a)Los valores de la variable x en la función de distribución b)Los valores de f(x) en la función de distribución c) El número de valores random que se quiere generar

[144]:
```
import numpy as np

random_30_edades = np.random.choice(
df_USA_niños_escolar_edad_list, size=30, p=df_USA_niños_probabilidad_list )
random_30_edades
```

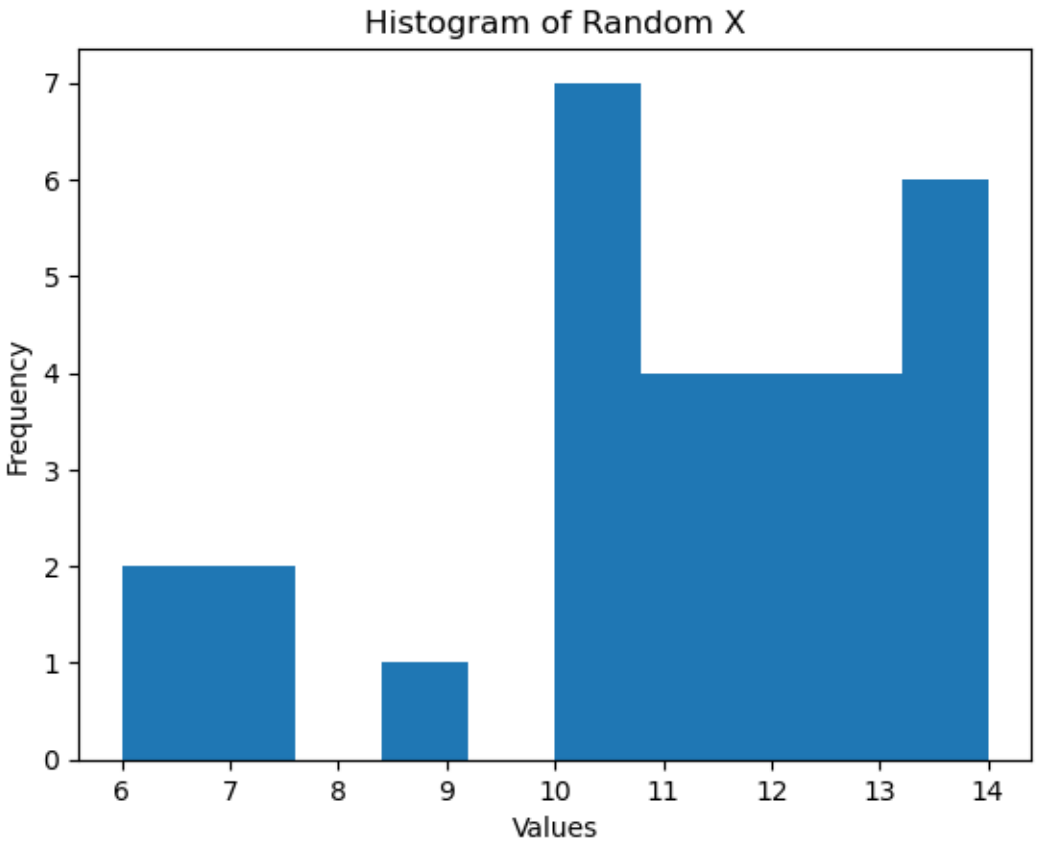
[144]:
```
array([ 7, 12,  6, 14, 11, 10, 12, 12, 10, 10, 11, 13, 10, 10,  9, 13, 13,
       14, 11, 14, 10, 12, 13, 14, 10,  6, 14,  7, 14, 11])
```

Vamos a graficar los valores generados

[145]:
```
import matplotlib.pyplot as plt

plt.hist(random_30_edades, bins=10)
plt.xlabel('Values')
```

```
plt.ylabel('Frequency')
plt.title('Histogram of Random X')
plt.show()
```



Ahora calculemos la edad promedio de los valores simulados

[146]:
```
import numpy as np

edad_promedio_simulada = round(np.mean(random_30_edades),2)
```

[147]:
```
from IPython.core.display import HTML


texto = f'<span style="color:red">_La edad promedio de los valores simulados␣
 ↪es__ es: {edad_promedio_simulada} años</span>'

HTML(texto)
```

[147]: <IPython.core.display.HTML object>

ES MUY CERCANA A LA ESPERANZA_EDAD QUE NO ES SIMULADA!

[ ]:

0.3.3 Problema 3

Considere la variable aleatoria cuya función de distribución (acumulativa) es la que se indica a continuación. a) Decida si la variable aleatoria es discreta o continua. Justifique su respuesta. b) Construya la función ( ). c) Calcule la media y la varianza de Y

[ ]:

$$F_Y(y) = \begin{cases} 0, & y <= 0 \\ y/8, & 0 < y < 2 \\ y^2/16, & 2 <= y < 4 \\ 1, & y > 4 \end{cases}$$

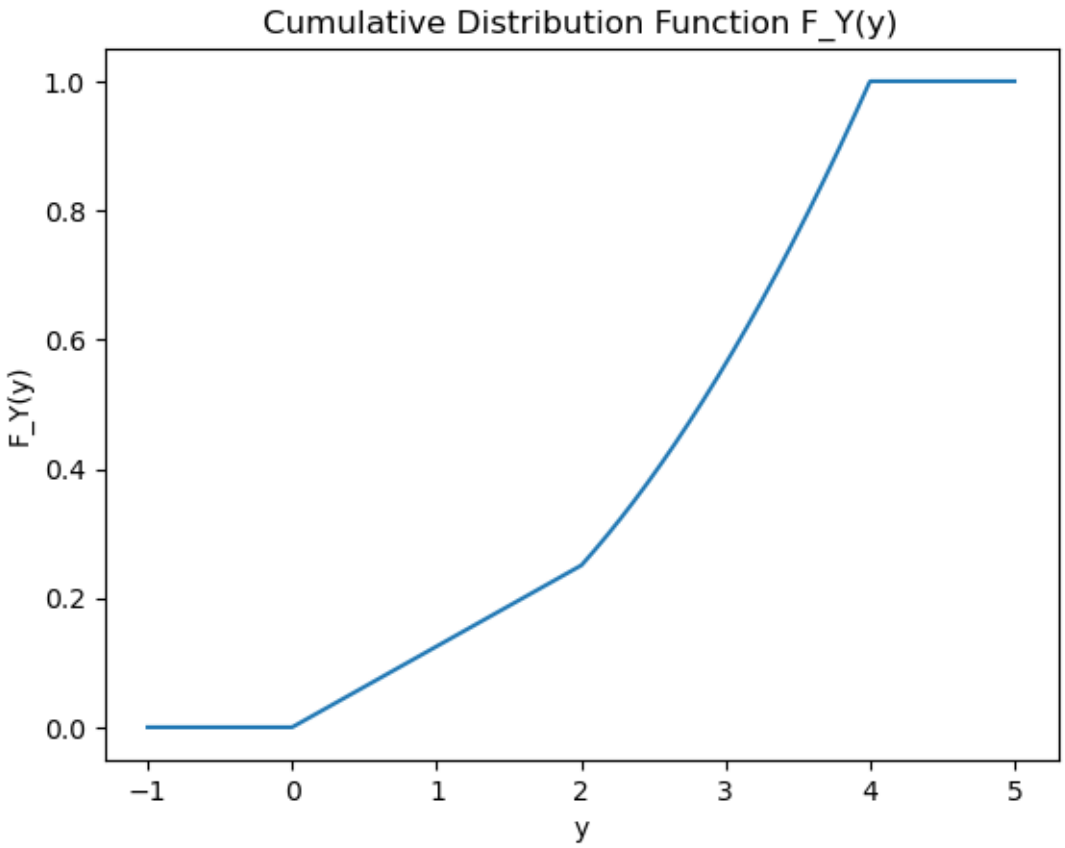
Primero que todo vamos a graficar la funcion de distribución acumulada para la variable Y y observar que forma tiene

[148]:
```
import numpy as np
import matplotlib.pyplot as plt

def F_Y(y):
    return np.piecewise(y, [y <= 0, (y > 0) & (y < 2), (y >= 2) & (y < 4), y >=␣
 ↪4],
                        [0, lambda y: y/8, lambda y: y**2/16, 1])

y = np.linspace(-1, 5, 1000)
F = F_Y(y)

plt.plot(y, F)
plt.title('Cumulative Distribution Function F_Y(y)')
plt.xlabel('y')
plt.ylabel('F_Y(y)')
plt.show()
```



Se observa claramente una función cotinua y creciente hasta 1, por lo que podemos estar seguros que F(y) es una función de distribución acumulativa

a) Decida si la variable aleatoria es discreta o continua. Justifique su respuesta. Debido a que la función de distribución $F_Y(y)$ SI ES CONTINUA. Se puede decir entonces que la Variable aleatoria X de la que proviene TAMBIÉN ES CONTINUA

b) Construya la función ( ). Nosotros sabemos que para cualquier variable aleatoria Y su función de distribución acumulativa se define como:

$$F_y(y) = P(Y <= y)$$

Y sabemos que la relación entre la función de distribución acumulada F(Y) y la función de distribcuión de densidad de probabilidad f(y) es:

$$F_y(Y) = \int_{-\infty}^{y} f_y(t)dt$$

Y por lo tanto se podría decir que es posible calcular la función de densidad de probabilidad f(y) a partir de su función de densidad acumuladad de probabilidad Fy(Y) de la siguiente manera:

$$f_y(y) = F'_y(t) = \frac{dFy}{dy}$$

Por lo tanto en nuestro caso habría que deribar cada una de las secciones de la función acumulativa F(y) con respecto a y para obtener f(y)

Por lo tanto la función de densidad probabilidad de Y sería - f(Y:)

$$f(y) = \begin{cases} 0, & y <= 0 \\ 1/8, & 0 < y < 2 \\ y/8, & 2 <= y < 4 \\ 0, & y >= 4 \end{cases}$$

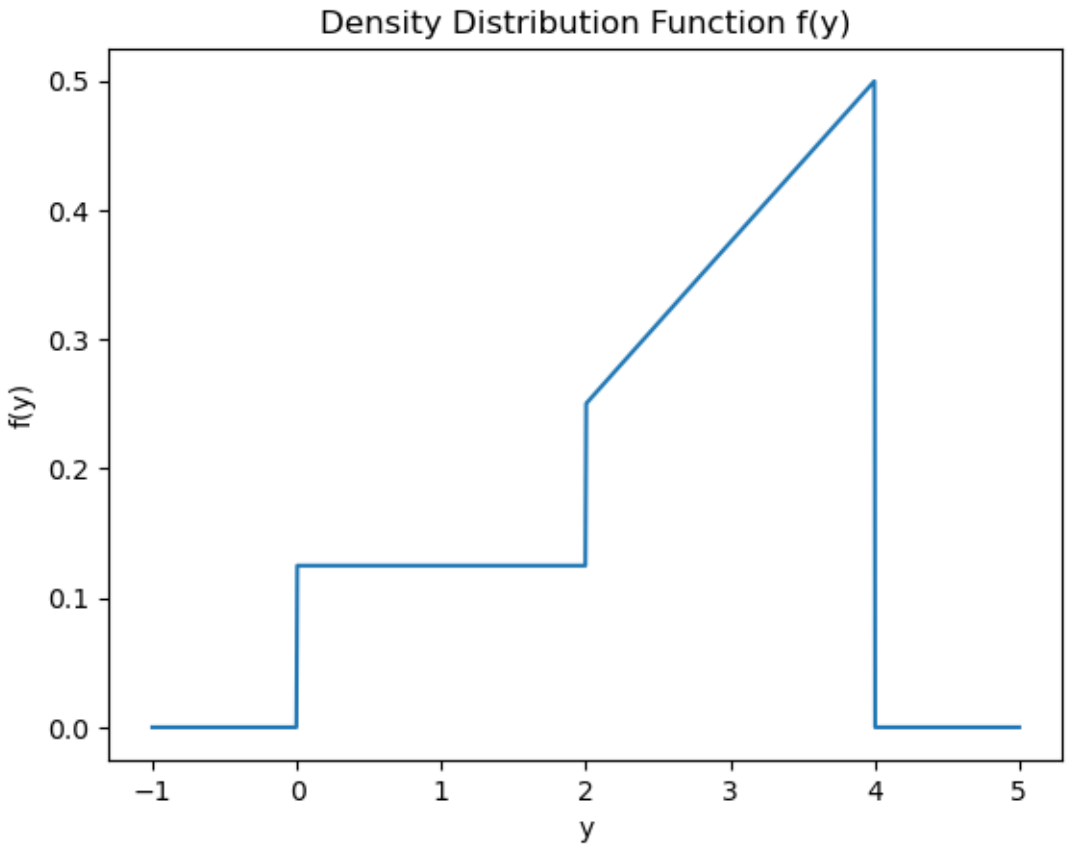
Vamos a graficar la función de densidad de probabilidad f(y) para ver como luce

[149]:
```
import numpy as np
import matplotlib.pyplot as plt

def f_y(y):
    return np.piecewise(y, [y<=0, (y>0)&(y<2), (y>=2)&(y<4), y>=4], [0, 1/8,␣
 ↪lambda y: y/8, 0])

y = np.linspace(-1, 5, 1000)
f = f_y(y)

plt.plot(y, f)
plt.title('Density Distribution Function f(y)')
plt.xlabel('y')
plt.ylabel('f(y)')
plt.show()
```



Se observa una función con valores mayores a cero y cuya área parece justamente tender al valor de la unidad

c) Calcule la media y la varianza de Y Recordando que la esperanza de una variable aleatoria Y, dada su función de distribución f(y) se definia para el caso de las variables discretas como:

$$E(Y) = \sum_{-\infty}^{\infty} y * p(y)$$

Para el caso de las variables continas dicha esperanza o media se define como:

$$E(Y) = \int_{-\infty}^{\infty} y * p(y)$$

Que para nuestro caso sería:

$$E(y) = \int_0^4 f(y) * y = \int_0^4 \begin{cases} 0, & y <= 0 \\ 1/8, & 0 < y < 2 \\ y/8, & 2 <= y < 4 \\ 0, & y >= 4 \end{cases} * y dy$$

Entonces re solviendo la integralen Python tenemos:

[150]:
```
import numpy as np
from scipy.integrate import quad

def f_y(y):
    return np.piecewise(y, [y<=0, (y>0)&(y<2), (y>=2)&(y<4), y>=4], [0, 1/8,␣
 ↪lambda y: y/8, 0])

##Se define la función que integrará y por lo tanto calculará la media
def mean_f(f, a, b):
    integrand = lambda y: y*f(y)
    ## FUNCIÓN QUE REALIZA LA INTEGRACIÓN!
    return quad(integrand, a, b)[0]

##Aqui se usa/llama la función que integra y calcula media
mean_Y = round(mean_f(f_y, 0, 4),2)
```

[151]:
```
from IPython.core.display import HTML


texto = f'<span style="color:red">_La media de la variable Y sería: {mean_Y} </
 ↪span>'

HTML(texto)
```

[151]:
```
<IPython.core.display.HTML object>
```

Para el caso de la varianza de la variable aleatoria Y, se define como, la esperanza del cuadrado de la desviación con respecto a la media, es decir:

$$Var(Y) = E[(Y - \mu)^2]$$

Ahora en el caso de conocer la funcion de probabilidad de densidad dicha varianza se calcula como:

$$Var(Y) = \sigma^2(y) = \int_{-\infty}^{\infty} (Y - \mu)^2 * f(y)$$

Y en nuestro caso:

$$Var(y) = \int_0^4 f(y) * (Y - \mu)^2 = \int_0^4 \begin{cases} 0, & y <= 0 \\ 1/8, & 0 < y < 2 \\ y/8, & 2 <= y < 4 \\ 0, & y >= 4 \end{cases} * (Y - \mu)^2 dy$$

Y en Python el código para integrar y genera la varianza es:

[152]:
```
import numpy as np
from scipy.integrate import quad

def f_y(y):
    return np.piecewise(y, [y<=0, (y>0)&(y<2), (y>=2)&(y<4), y>=4], [0, 1/8,␣
 ↪lambda y: y/8, 0])

##Se define la función que integrará y por lo tanto calculará la varianza
def variance_f(f, mean_Y, a, b):
    integrand = lambda y: (y-mean_Y)**2*f(y)
    return quad(integrand, a, b)[0]

##Aqui se usa/llama la función que integra y calcula varianza
variance_Y = round(variance_f(f_y, mean_Y, 0, 4),2)
```

[153]:
```
from IPython.core.display import HTML


texto = f'<span style="color:red">_La varianza de la variable Y sería:␣
 ↪{variance_Y} </span>'

HTML(texto)
```

[153]:
```
<IPython.core.display.HTML object>
```

0.3.4 Problema 4

[ ]:
```

```

La proporción de tiempo por día en la que todas las cajas de un supermercado están ocupadas, es una variable aleatoria con función de densidad

$$f_Y(y) = \begin{cases} c * y^2(1-y)^4, & 0 <= y <= 1 \\ 0, & enotrocaso \end{cases}$$

En donde es una constante.

a) Encuentre el valor de c que hace de ( ) una función de densidad
b) Obtenga la media y la varianza de .
c) Genere 30 valores aleatorios de y calcule el promedio de dichos valores. Muestre paso a paso su procedimiento realizado ya sea manualmente o con algún software, en cuyo caso es necesario mostrar la codificación

a) Encuentre el valor de c que hace de ( ) una función de densidad Entonces se debe encontrar los valores de c para los cuales la integral de f(x) desde 0 hasta 1 sea exactamente igual a 1. que es una de las propiedades de la función de distribución de una v.a. X

La propiedad de la función de distribución f(x) se expresa entonces como:

$$\int_0^1 f_x(x) = 1$$

Que en nuestro caso es:

$$\int_0^1 c * y^2 * (1-y)^4 = 1$$

Entonces resolvamos la integral usando Python

[154]:
```
import sympy as sym

# define the symbol y and the constant c
y = sym.symbols('y')
c = sym.symbols('c')

# define the integrand
f = c * y**2 * (1-y)**4

# integrate the function with respect to y
F = sym.integrate(f, y)

# print the result
print("The antiderivative of f(y) = c*y^2*(1-y)^4 with respect to y is:")
print(F)
```

```
The antiderivative of f(y) = c*y^2*(1-y)^4 with respect to y is:
c*y**7/7 - 2*c*y**6/3 + 6*c*y**5/5 - c*y**4 + c*y**3/3
```

Entonces el resultado de la integral es:

$$\int_0^1 c * y^2 * (1-y)^4 dy = \frac{c*y^7}{7} - \frac{2*c*y^6}{3} + \frac{6*c*y^5}{5} - c*y^4 + \frac{c*y^3}{3}$$

Y evaluando los limites de intetración tenemos:

$$\left[\frac{c*y^7}{7} - \frac{2*c*y^6}{3} + \frac{6*c*y^5}{5} - c*y^4 + \frac{c*y^3}{3}\right]_0^1 = [\frac{c}{7} - \frac{2*c}{3} + \frac{6*c}{5} - c* + \frac{c}{3}] - (0)$$

[155]:
```
Sumando los terminos, tenemos:
```

```
  File "C:\Users\sergi\AppData\Local\Temp\ipykernel_27736\218246960.py", line 1
    Sumando los terminos, tenemos:
            ^
SyntaxError: invalid syntax
```

$$1 = \frac{1c}{105}$$

Por lo tanto el valor de C que hace que le función f(y) sea una función de densidad de probabilidad es C=105-

b) Obtenga la media y la varianza de . La media se calcula como:

$$E(Y) = \int_{-\infty}^{\infty} y * p(y)$$

Que en nuestro caso es:

$$E(X) = \int_{-\infty}^{\infty} y * p(y) = \int_{-\infty}^{\infty} \begin{cases} c * y^2(1-y)^4, & 0 <= y <= 1 \\ 0, & enotrocaso \end{cases} * ydy$$

Entonces resolviendo la integral en Python tenemos:

[156]:
```
import numpy as np
from scipy.integrate import quad

def f_y(y):
    return np.piecewise(y, [y<=0,(y>0)&(y<1),  y>1], [0, lambda y:␣
 ↪105*(y**2)*(1-y)**4, 0])

##Se define la función que integrará y por lo tanto calculará la media
def mean_f(f, a, b):
    integrand = lambda y: y*f(y)
    ## FUNCIÓN QUE REALIZA LA INTEGRACIÓN!
    return quad(integrand, a, b)[0]

##Aqui se usa/llama la función que integra y calcula media
mean_Y2 = round(mean_f(f_y, 0, 4),2)
```

[157]:
```
from IPython.core.display import HTML


texto = f'<span style="color:red">_La MEDIA de la variable Y sería: {mean_Y2} </
 ↪span>'

HTML(texto)
```

[157]:
```
<IPython.core.display.HTML object>
```

Y para el cálculo de la varianza tenemos:

$$E(X) = \int_{-\infty}^{\infty} (Y-\mu)^2 * p(y) = \int_{-\infty}^{\infty} \begin{cases} c * y^2(1-y)^4, & 0 <= y <= 1 \\ 0, & enotrocaso \end{cases} * (Y-\mu)^2 dy$$

[158]:
```
import numpy as np
from scipy.integrate import quad

def f_y(y):
    return np.piecewise(y, [y<=0,(y>0)&(y<1),  y>1], [0, lambda y:␣
 ↪105*(y**2)*(1-y)**4, 0])

##Se define la función que integrará y por lo tanto calculará la varianza
def variance_f(f, mean, a, b):
    integrand = lambda y: (y-mean_Y2)**2*f(y)
    return quad(integrand, a, b)[0]

##Aqui se usa/llama la función que integra y calcula varianza
variance_Y2 = round(variance_f(f_y, mean_Y2, 0, 4),2)
```

[159]:
```
from IPython.core.display import HTML


texto = f'<span style="color:red">_La VARIANZA de la variable Y sería:␣
 ↪{variance_Y2} </span>'

HTML(texto)
```

[159]: <IPython.core.display.HTML object>

c) Genere 30 valores aleatorios de y calcule el promedio de dichos valores. Muestre paso a paso su procedimiento realizado ya sea manualmente o con algún software, en cuyo caso es necesario mostrar la codificación

[ ]:

Paso 1. Se generan los 30 numeros aleatorios ente [0,1] con una probabilida de distribución uniforme

[160]:
```
import random

# generate 30 random numbers between 0 and 1
random_numbers_30 = [random.random() for _ in range(30)]

print(random_numbers_30)
```

```
[0.9806787939195338, 0.6373228621636793, 0.8671642108005828, 0.731334915741731,
0.5814038843744755, 0.030530241894779153, 0.7683749526362508,
0.5585599805109415, 0.4820551570490622, 0.3215062668339316, 0.9289311548356329,
0.5135022761977845, 0.777444998907035, 0.5145538607604121, 0.7199896590117759,
0.49254363099716025, 0.40729045955237, 0.4758043809833101, 0.06367841777531169,
0.9665078775355336, 0.7292828155843758, 0.6980366737540419, 0.8881862766957436,
0.3605390291229362, 0.9420075543813922, 0.4389308897607932, 0.37468918073193413,
0.09878942535855406, 0.8382744389276778, 0.939035662646074]
```

Paso 2. Igualar el valor aleatorio con la funcion acumulada de probabilidad de la variable x

Se procede a igular el valor aleatorio generado ri con la funcion de distribucion acumulada:

$$r_i = F(y_i)$$

Que en nuestro caso recodar que

$$F(y_i) = \int_0^1 105 * y^2 * (1-y)^4 dy$$

Y resolviendo en Python tenemos:

[161]:
```
import sympy as sym

# define the symbol y and the constant c
y = sym.symbols('y')

# define the integrand
f_ = 105 * y**2 * (1-y)**4

# integrate the function with respect to y
F_ = sym.integrate(f_, y)

# print the result
print("The antiderivative of f(y) = 105*y^2*(1-y)^4 with respect to y is:")
print(F_)
```

```
The antiderivative of f(y) = 105*y^2*(1-y)^4 with respect to y is:
15*y**7 - 70*y**6 + 126*y**5 - 105*y**4 + 35*y**3
```

Es decir:

$$valor_random_i = 15y^7 - 70y^6 + 125y^5 - 105y^4 + 35y^3$$

Entonces para nuestro primer valor simulado que es 0.105, lo igualamos a nuestra F(yi):

Paso 3. Calcular el valor de la variable x para cada elemento previamente simulado

Te calcula las 7 soluciones por cada valor de ri

[162]:
```
import numpy as np

# Define the coefficients of the polynomial
coeffs = [15, -70, 125, -105, 35, 0, -0.53]

# Find the roots of the polynomial
roots = np.roots(coeffs)
```

```
# Print the roots
print(roots)
```

```
[ 1.79584535+0.j          0.96850278+0.63368849j  0.96850278-0.63368849j
  0.88154565+0.j          0.157833  +0.j         -0.1055629 +0.j        ]
```

[163]:
```
Te calcula solo las  soluciones reales por cada valor de ri
```

```
  File "C:\Users\sergi\AppData\Local\Temp\ipykernel_27736\3617862813.py", line
    Te calcula solo las  soluciones reales por cada valor de ri
       ^
SyntaxError: invalid syntax
```

[164]:
```
import numpy as np

# Define the coefficients of the polynomial
coeffs = [15, -70, 125, -105, 35, 0, -0.53]

# Find the roots of the polynomial
roots = np.roots(coeffs)

# Find the real roots
real_roots = [root for root in roots if np.isreal(root)]

# Print the real roots
print(real_roots)
```

```
[(1.795845347276372+0j), (0.8815456538932235+0j), (0.1578329991294134+0j),
(-0.10556289649685834+0j)]
```

[165]:
```
Te calcula solo soluciones reales para todos los valres en random_numbers_30
```

```
  File "C:\Users\sergi\AppData\Local\Temp\ipykernel_27736\704867932.py", line 1
    Te calcula solo soluciones reales para todos los valres en random_numbers_3
       ^
SyntaxError: invalid syntax
```

[166]:
```
from numpy import roots

random_numbers_30

for i in range(len(random_numbers_30)):
    roots_of_equation = roots([15, -70, 125, -105, 35, 0, 0,␣
 ↪-random_numbers_30[i]])
```

```
    real_roots = [root.real for root in roots_of_equation if root.imag == 0]
    print(f"For {random_numbers_30[i]}: {real_roots}")
```

```
For 0.9806787939195338: [1.7961893174657182]
For 0.6373228621636793: [1.7920325929651895, 0.8149267928566021,
0.4345264275183866]
For 0.8671642108005828: [1.794831476678376]
For 0.731334915741731: [1.793185605429109, 0.7629591941365178,
0.4920428520361721]
For 0.5814038843744755: [1.7913412712192647, 0.8394785898450381,
0.40567446992985634]
For 0.030530241894779153: [1.7842977510982982, 0.9938179881017878,
0.10669369601084153]
For 0.7683749526362508: [1.7936367470719945, 0.7353561933843674,
0.5212976203542592]
For 0.5585599805109415: [1.7910576558007452, 0.8486393197019009,
0.39452299930097395]
For 0.4820551570490622: [1.7901026921123728, 0.8765473055533888,
0.35885702446901757]
For 0.3215062668339316: [1.7880723966319252, 0.9253450042140581,
0.2869146195587798]
For 0.9289311548356329: [1.7955722817426085]
For 0.5135022761977845: [1.7904961891629476, 0.8655378621739112,
0.3732685837079013]
For 0.777444998907035: [1.7937469516521543, 0.7273871963744867,
0.5296273374843563]
For 0.5145538607604121: [1.7905093243358319, 0.8651593044402212,
0.3737555962760065]
For 0.7199896590117759: [1.7930470697719243, 0.7703066659807155,
0.48413089466630527]
For 0.49254363099716025: [1.7902340841599391, 0.8729401904217142,
0.36363359713893895]
For 0.40729045955237: [1.7891617038409773, 0.9006235163031343,
0.3253380483558949]
For 0.4758043809833101: [1.7900243152513575, 0.8786677503767851,
0.35602256296628576]
For 0.06367841777531169: [1.7847341738273435, 0.9869269446886817,
0.1415577132387234]
For 0.9665078775355336: [1.796020668268184]
For 0.7292828155843758: [1.7931605597418476, 0.7643189341815212,
0.4905830714675286]
For 0.6980366737540419: [1.7927785324968946, 0.7834948326977891,
0.46977048823562007]
For 0.8881862766957436: [1.795084135115167]
For 0.3605390291229362: [1.7885693327120307, 0.9144322211122903,
0.30449785767831333]
For 0.9420075543813922: [1.795728512184853]
For 0.4389308897607932: [1.789560874325653, 0.8907615607633855,
```

```
0.33944502036848057]
For 0.37468918073193413: [1.788748948751217, 0.9103422851834199,
0.31081877397910235]
For 0.09878942535855406: [1.7851946040787416, 0.9794116115933807,
0.16888939099778247]
For 0.8382744389276778: [1.794483361016012]
For 0.939035662646074: [1.7956930238382889]
```

[167]:
```
type(real_roots)
```

[167]:
```
list
```

[168]:
```
Te calcula solo soluciones reales para todos los valores en random_numbers_30 y␣
↪las guarda en all_real_roots
```

```
  File "C:\Users\sergi\AppData\Local\Temp\ipykernel_27736\2000383129.py", line
    Te calcula solo soluciones reales para todos los valores en␣
↪random_numbers_30 y las guarda en all_real_roots
       ^
SyntaxError: invalid syntax
```

[169]:
```
from numpy import roots

random_numbers_30  # your list of 30 random numbers

all_real_roots = []  # create an empty list to store real roots for each␣
↪iteration

for i in range(len(random_numbers_30)):
    roots_of_equation = roots([15, -70, 125, -105, 35, 0, 0,␣
↪-random_numbers_30[i]])
    real_roots = [root.real for root in roots_of_equation if root.imag == 0]
    all_real_roots.append(real_roots)  # append the real roots to the list of␣
↪all real roots

print(all_real_roots)
```

```
[[1.7961893174657182], [1.7920325929651895, 0.8149267928566021,
0.4345264275183866], [1.794831476678376], [1.793185605429109,
0.7629591941365178, 0.4920428520361721], [1.7913412712192647,
0.8394785898450381, 0.40567446992985634], [1.7842977510982982,
0.9938179881017878, 0.10669369601084153], [1.7936367470719945,
0.7353561933843674, 0.5212976203542592], [1.7910576558007452,
0.8486393197019009, 0.39452299930097395], [1.7901026921123728,
0.8765473055533888, 0.35885702446901757], [1.7880723966319252,
0.9253450042140581, 0.2869146195587798], [1.7955722817426085],
```

```
[1.7904961891629476, 0.8655378621739112, 0.3732685837079013],
[1.7937469516521543, 0.7273871963744867, 0.5296273374843563],
[1.7905093243358319, 0.8651593044402212, 0.3737555962760065],
[1.7930470697719243, 0.7703066659807155, 0.48413089466630527],
[1.7902340841599391, 0.8729401904217142, 0.36363359713893895],
[1.7891617038409773, 0.9006235163031343, 0.3253380483558949],
[1.7900243152513575, 0.8786677503767851, 0.35602256296628576],
[1.7847341738273435, 0.9869269446886817, 0.1415577132387234],
[1.796020668268184], [1.7931605597418476, 0.7643189341815212,
0.4905830714675286], [1.7927785324968946, 0.7834948326977891,
0.46977048823562007], [1.795084135115167], [1.7885693327120307,
0.9144322211122903, 0.30449785767831333], [1.795728512184853],
[1.789560874325653, 0.8907615607633855, 0.33944502036848057],
[1.788748948751217, 0.9103422851834199, 0.31081877397910235],
[1.7851946040787416, 0.9794116115933807, 0.16888939099778247],
[1.794483361016012], [1.7956930238382889]]
```

[170]:
```
all_real_roots[0]
```

[170]:
```
[1.7961893174657182]
```

Te Guarda el valor mínimo entre 0 y 1 como resultado de resolver la ecuación de F(xi)=ri para cada i en los 30 elementos simulados anteriormente

[171]:
```
final_real_roots = []

for roots_list in all_real_roots:
    smallest_root = min(roots_list)
    final_real_roots.append(smallest_root)
```

[172]:
```
from IPython.core.display import HTML


texto = f'<span style="color:red">La lista de los 30 números simulados finales␣
 ↪es:</span>' + f'<span style="color:orange"> {final_real_roots} </span>'

HTML(texto)
```

[172]:
```
<IPython.core.display.HTML object>
```

[173]:
```
El promedio de los valores simulados es:
```

```
  File "C:\Users\sergi\AppData\Local\Temp\ipykernel_27736\1822171954.py", line
    El promedio de los valores simulados es:
       ^
SyntaxError: invalid syntax
```

[174]:
```
import numpy as np

average_final_real_roots = round(np.mean(final_real_roots),2)
```

[175]:
```
from IPython.core.display import HTML


texto = f'<span style="color:red">El valor promedio de los valores simulados es␣
 ↪: {average_final_real_roots}:</span>'
HTML(texto)
```

[175]:
```
<IPython.core.display.HTML object>
```

[ ]:

0.3.5 Problema 5

[ ]:

Considere la v.a. X con función de probabilidad

x	-5	-1	1	1.5	3
fx(X)	0.2	0.01	0.3	0.29	0.2

[ ]:

Construya muestras de la v.a. X, de los tamaños indicados en clase, y en cada caso sobreponga la gráfica de la distribución de X (histograma de probabilidad) al histograma de frecuencias de la muestra correspondiente. Adicionalmente, calcule la media y la varianza de cada muestra y compárelas con ( ) y ( )

Creamos los objetos "tipo lista" de los valores de x y f(x)

[176]:
```
x1 = [-5,-1,1,1.5,3]
x1
fx1 =  [0.2,0.01,0.3,0.29,0.2]
fx1
```

[176]:
```
[0.2, 0.01, 0.3, 0.29, 0.2]
```

Importamos las librerias necesarias para realizar las muestras de tamaño n

[177]:
```
from random import random
import numpy as np
import matplotlib.pyplot as plt
```

Usamos la función np.random.choice (previamente usada y explicada en el problema 2 inciso c)

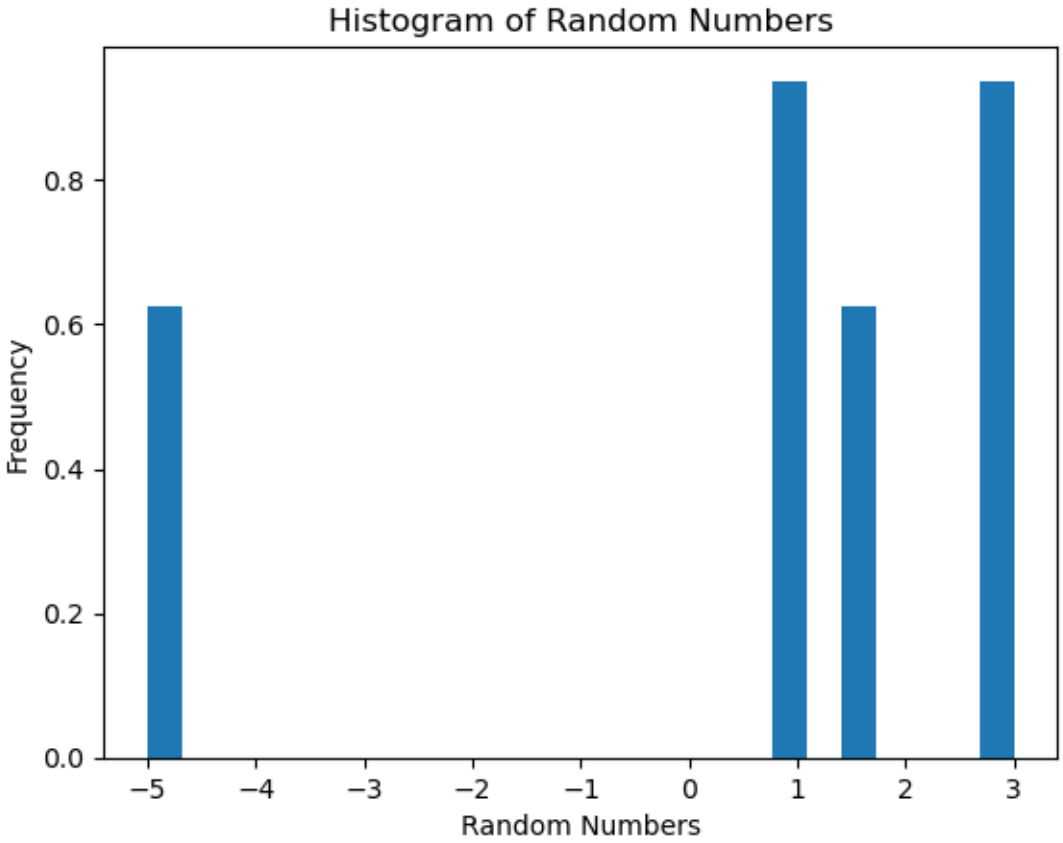
Se generarán entonces los números para tamaño de muestra n=10

[178]:
```
random_numbers_fx1_10 = np.random.choice(x1, size=10, p=fx1)
print(random_numbers_fx1_10)
```

```
[ 3.  -5.   1.   1.5  1.   3.   1.   3.   1.5 -5. ]
```

Graficamos el reusltado para tamaño de muestra n=10

[179]:
```
plt.hist(random_numbers_fx1_10, bins=25, density=True)
plt.xlabel('Random Numbers')
plt.ylabel('Frequency')
plt.title('Histogram of Random Numbers')
plt.show()
```



Calculamos el promedio para tamaño de muestra n=10

[180]:
```
import numpy as np

average_random_numbers_fx1_10= round(np.mean(random_numbers_fx1_10),2)
average_random_numbers_fx1_10
```

[180]: 0.5

[181]:
```
Calculamos la varianza para tamaño de muestra n=10
```

```
  File "C:\Users\sergi\AppData\Local\Temp\ipykernel_27736\388996276.py", line 1
    Calculamos la varianza para tamaño de muestra n=10
                ^
SyntaxError: invalid syntax
```

[182]:
```
import numpy as np

variance_random_numbers_fx1_10= round(np.var(random_numbers_fx1_10),2)
variance_random_numbers_fx1_10
```

[182]:
```
8.2
```

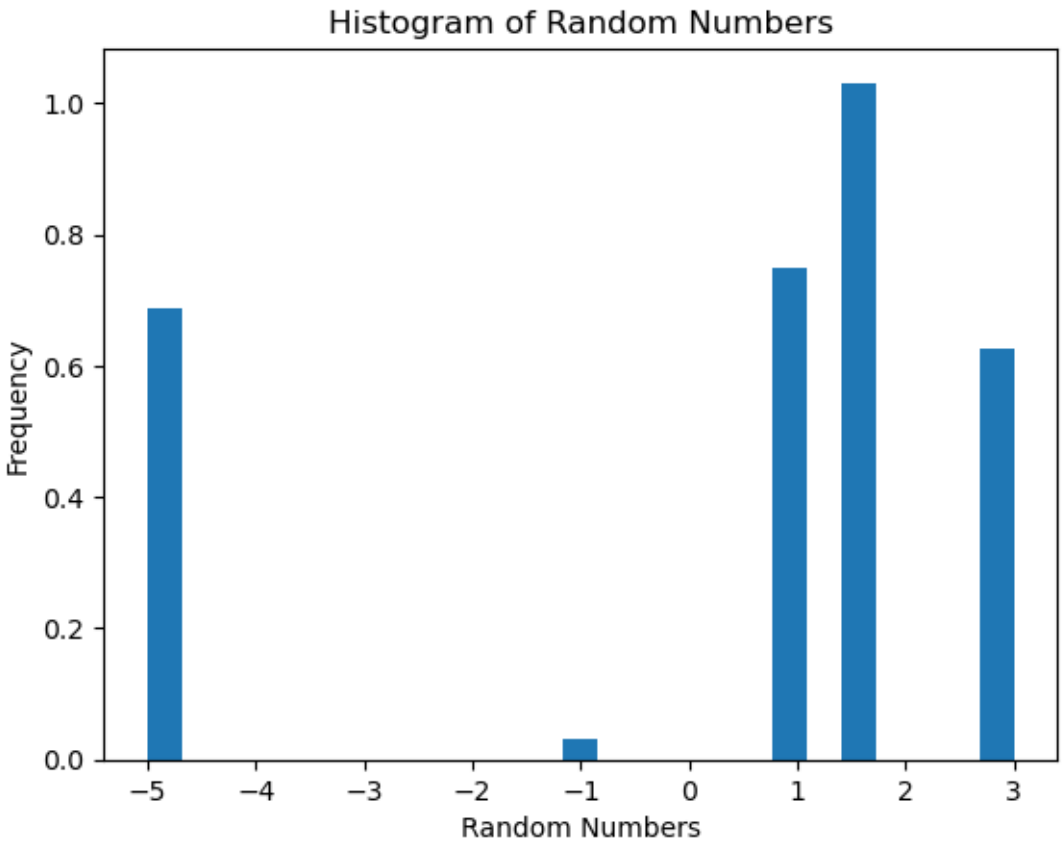
Se generarán entonces los números para tamaño de muestra n=100

[183]:
```
random_numbers_fx1_100 = np.random.choice(x1, size=100, p=fx1)
print(random_numbers_fx1_100)
```

```
[-5.   1.5  3.  -5.   1.5 -5.   3.   1.5 -5.   3.  -5.   1.5  1.5  1.
  1.5  1.5 -5.   1.  -5.   1.   3.   1.   1.   1.5  1.5 -1.   3.   1.
  1.5  3.   1.   3.  -5.   3.   1.5 -5.   1.   1.5 -5.   1.5  1.5  1.5
  1.   1.5  1.5  3.   1.5  1.  -5.   1.   1.5  1.  -5.  -5.   3.   1.
 -5.   1.5  1.5  1.  -5.   1.   1.5  1.  -5.   1.   1.   1.   1.5 -5.
  3.   1.5  3.   1.  -5.   1.5 -5.   1.   1.5  1.   3.   3.  -5.   1.5
  1.5 -5.   1.5  1.5  1.5  3.   3.   1.5  3.   3.  -5.   3.   3.   1.5
  1.   1. ]
```

Graficamos el reusltado para tamaño de muestra n=100

[184]:
```
plt.hist(random_numbers_fx1_100, bins=25, density=True)
plt.xlabel('Random Numbers')
plt.ylabel('Frequency')
plt.title('Histogram of Random Numbers')
plt.show()
```



Calculamos el promedio para tamaño de muestra n=100

[185]:
```
import numpy as np

average_random_numbers_fx1_100= round(np.mean(random_numbers_fx1_100),2)
average_random_numbers_fx1_100
```

[185]: 0.22

Calculamos la varianza para tamaño de muestra n=100

[186]:
```
import numpy as np

variance_random_numbers_fx1_100= round(np.var(random_numbers_fx1_100),2)
variance_random_numbers_fx1_100
```

[186]: 8.24

Se generarán entonces los números para tamaño de muestra n=1000

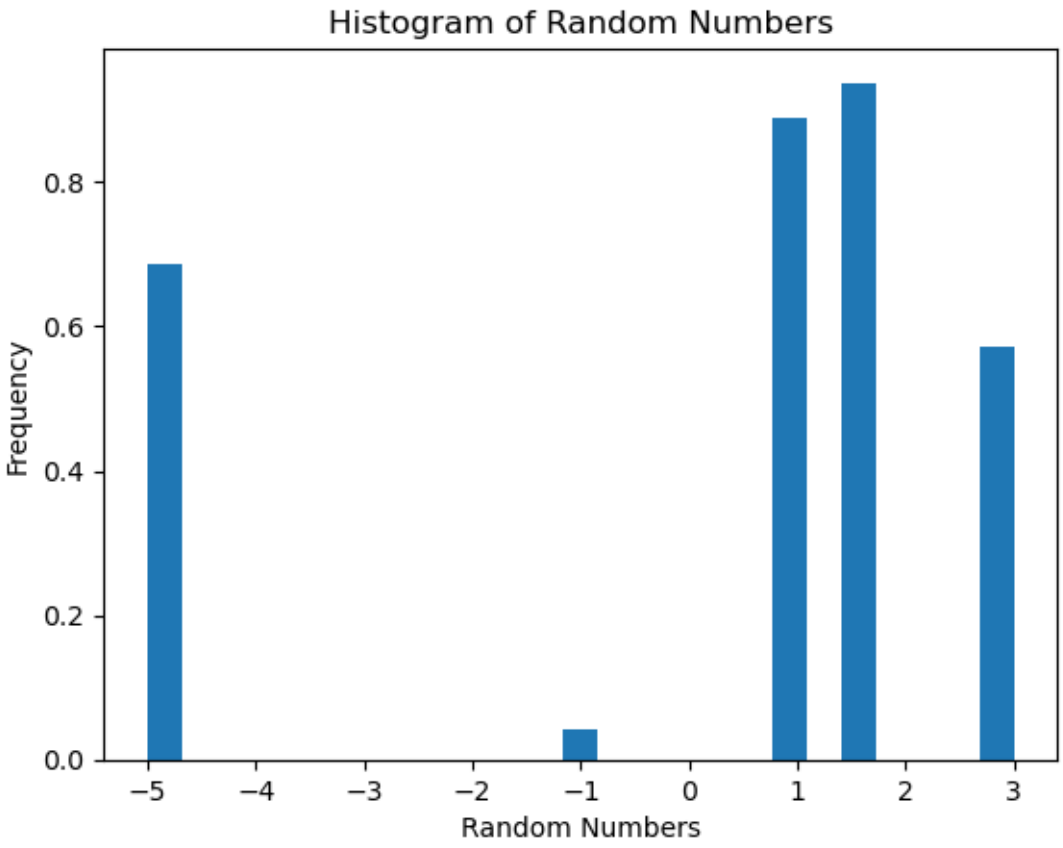
[187]:
```
random_numbers_fx1_1000 = np.random.choice(x1, size=1000, p=fx1)
print(random_numbers_fx1_1000)
```

```
[ 3.  -5.   1.  -5.   1.5  1.   1.5 -5.  -5.   1.   1.5 -5.   1.5  3.
 -5.   1.   1.5  3.  -5.  -5.   1.   1.   1.5  1.5  1.5  1.   1.5 -5.
  1.5  1.5 -5.  -5.   1.   1.5  1.5 -5.   1.5  1.   1.5  1.5  1.   1.
 -5.   1.5 -5.   1.   3.   1.  -1.   1.   1.5  3.   1.   1.   1.5  1.
  1.5 -5.   3.  -5.   1.5  3.  -5.   3.   3.  -5.   1.   1.   1.5  1.
  1.   1.  -5.  -1.  -5.  -5.   1.5 -5.  -5.  -1.   1.   3.   1.   1.
 -5.   1.5  1.   1.5  3.   3.   1.   1.  -5.   1.5 -1.   1.5  1.   1.5
  3.   1.5 -5.  -5.  -5.   1.   1.5  1.   1.   3.   1.   1.  -5.   3.
  1.5 -5.   1.  -5.   1.5  1.5 -5.   1.5  3.   1.  -5.   3.   1.5  1.5
  3.   1.5  3.   1.   3.   1.  -5.  -5.   3.   3.   1.5  3.   1.5 -5.
  3.  -5.   1.5 -1.   3.   3.  -5.   1.   1.5  1.5 -5.   1.5  3.   3.
  1.5 -5.  -5.  -5.  -5.  -5.   3.   1.   3.   1.  -5.   3.  -5.   1.
 -5.  -5.   1.   1.   1.   1.   1.  -1.  -5.   1.5 -5.   1.5  1.   1.5
  1.5  1.   1.   1.5 -5.   1.  -5.  -5.   1.5  1.5 -5.  -5.   3.   1.
  1.  -5.   3.   1.   1.  -5.   1.5  1.5  1.5  3.   1.   1.  -5.  -5.
 -5.  -5.   3.   1.5  3.   1.5 -5.  -5.   3.   3.   1.5  1.5  1.   1.
 -5.  -5.   1.   1.5  1.  -5.   1.5  3.   1.5  1.5  1.   1.   1.   1.5
  3.   3.   1.   1.   3.   3.   1.5  3.   3.   1.5  1.   1.   1.   1.5
  1.5  1.5  1.   1.   3.  -5.   1.5  3.   1.   1.5  1.   1.5  1.   1.
  1.5  1.   1.5  1.   1.5  1.5 -5.   1.5  1.   1.   1.5  3.  -5.   1.5
 -5.   1.  -5.   1.5  1.5  3.   1.   1.5  1.5  1.5  1.5  3.   1.5  3.
  3.  -5.   1.  -5.   1.   1.   1.5  3.   1.5  3.   1.5 -5.   3.   1.5
  3.   3.   1.5  1.5  1.5  1.   1.5 -5.   1.  -5.   3.  -5.   1.   1.5
 -5.   1.5  1.5  1.   1.5  3.   1.5 -5.   1.5  1.  -5.   1.5  1.5  1.5
  1.5  1.   1.  -5.   1.   3.   1.5  1.5  1.  -5.  -5.   1.5  1.   3.
  3.   1.   1.  -5.   1.5 -5.   1.   3.   1.   1.   1.5 -5.   3.   1.
  1.5  1.5  1.5 -5.   3.   1.  -5.  -5.  -5.   3.   1.   1.5  3.   1.5
  3.   1.   1.5  3.   1.5  3.   1.5  3.   3.   1.   1.5  1.   1.5  1.
  1.5  3.   3.   1.   1.   1.   3.   3.   1.5  3.   1.5  1.   1.5 -5.
 -5.   1.5  1.   3.  -5.  -5.   1.   1.   1.  -5.   3.   1.5  1.5  1.
  1.   1.5  3.   1.5 -5.  -5.   1.   3.   1.5  1.  -5.  -5.  -5.   1.
  3.  -5.  -5.  -5.   3.   1.   1.   1.  -5.   1.5 -5.   3.   1.5  1.
  1.5  1.5  1.5  3.  -5.   1.  -5.   3.   1.   1.5  3.   1.   1.   3.
 -5.   3.   1.5  1.5 -5.   3.   1.   1.5  1.   1.5  1.5  1.   1.   1.
 -5.   3.   1.   1.5  1.5  1.  -5.   1.5 -5.   3.   3.   1.5  1.   3.
  1.5  1.5 -1.   1.  -5.   1.5 -5.  -5.   3.   1.5  1.   1.5  1.   1.
  1.5  1.5  3.  -5.   1.   1.5 -5.   1.   1.5  1.   3.   1.  -5.   1.5
  3.   1.5  1.  -5.   1.   1.   3.   3.   1.   1.5  1.5 -5.   1.   3.
 -5.  -5.  -5.   1.   1.  -5.   1.5  1.   1.5  3.   1.  -5.   1.5  1.5
  1.5 -5.   3.  -5.   3.  -5.   3.   1.5  3.   1.5  1.5  1.5 -5.   1.5
 -5.   1.5 -5.   1.   1.5  1.5  1.   1.5  1.5  1.5  1.5  1.5  1.  -5.
  1.5  1.   1.  -5.   1.5  1.   3.   1.5  1.5  1.5 -5.   1.   1.5  3.
  1.5  1.   1.   1.  -5.   1.5  3.  -5.   1.   1.5 -5.   1.5 -5.   1.5
  1.   3.   1.5 -5.   1.  -5.  -5.   1.5  1.  -5.   1.5  3.   1.   3.
 -5.   1.   1.5  1.   1.  -5.  -5.   3.   1.   1.5 -5.   1.5  1.5  1.5
```

```
 1.5  1.   1.5  1.  -5.   1.   1.5  1.  -5.   1.5  1.   3.  -5.  -5.
 1.5  1.   1.5  1.   3.   1.   1.5  1.  -5.   1.   1.5 -1.   1.5  1.
 3.   1.5  3.   1.  -5.   1.   1.   3.   1.5  1.5  1.   1.  -5.  -5.
 1.5  3.   1.   3.  -5.   1.5  1.5  1.   3.  -5.   1.5 -5.   1.   1.
 1.5  1.   1.5  1.5  3.   1.5 -5.   3.   1.5 -5.   1.5  1.   1.5  1.
 3.   1.5 -5.   3.   1.5  1.5  1.5  1.   3.   1.   1.   3.   1.   1.5
-5.   1.   1.  -5.   1.   1.   1.5 -5.   3.   1.   1.   3.   1.5  1.
 3.   1.   3.   1.   1.  -5.   1.5  1.5  1.   1.5  1.5  1.5 -5.   3.
 1.5  3.   1.5 -5.   1.5  1.5  3.  -5.   1.5  1.5  1.  -5.   1.5  1.5
 1.   3.   1.   1.5  1.5  3.   1.   1.  -5.   3.   1.5  1.5  1.   3.
-1.   1.   1.   1.5 -5.  -5.   1.5  3.   3.  -5.   1.   3.   3.  -5.
 1.  -5.   1.   1.5 -5.   1.  -5.  -5.   3.   3.   3.   1.5  1.5  1.5
 1.   3.  -5.   1.5  3.   1.5  1.5  1.   1.5  1.5  1.   1.5  1.5  3.
-1.   3.   1.   1.   1.   3.   3.  -5.  -5.   1.   1.5 -5.   1.5  1.5
 1.   1.  -5.   1.5 -5.   1.5  1.   1.   3.  -5.   3.   1.5 -5.   1.
-1.   1.   1.5  3.  -5.   1.   1.  -5.   1.5  1.   1.5 -5.  -5.  -5.
-5.  -5.   1.5  1.5  1.   1.5  1.5  1.5  3.   3.  -5.   1.  -5.   3.
-5.   3.   1.  -5.  -5.   1.  -5.   1.5  3.   3.   1.5 -5.   1.   1.5
 3.   1.  -5.   1.5  1.5  1.   3.   3.  -1.   1.   1.5  1.5  3.   1.5
 1.5  1.   1.5  1.   1.   1.5  3.   1.5  1.  -5.   1.5  1.   1.   3.
 3.   1.5  1.   1.5  1.   1.   1.5  3.   1.5  1.5  1.5  1.   1.   3.
 1.  -5.   1.  -5.  -5.   1.   1.5  3.   1.5  1.   1.  -5.   1.5  3.
 1.5 -5.   1.   3.   1.  -1.   1.   3.   1.   1.5  1.   1.  -5.  -5.
 1.5  3.   1.5  1.   3.   1.5  1.   3.   1.  -5.   3.   1.5  1.   1.
-5.   1.   1.  -5.   3.   3.   1.5 -5.   1.   1.5  1.   1.   3.   1.
 1.   1.5 -5.  -5.  -5.   1.   1.   1.5 -5.   1.   3.   1.   1.5  3.
-5.   1.   3.  -5.  -5.  -5. ]
```

Graficamos el reusltado para tamaño de muestra n=1000

[188]:
```
plt.hist(random_numbers_fx1_1000, bins=25, density=True)
plt.xlabel('Random Numbers')
plt.ylabel('Frequency')
plt.title('Histogram of Random Numbers')
plt.show()
```



Calculamos el promedio para tamaño de muestra n=1000

[189]:
```
import numpy as np

average_random_numbers_fx1_1000= round(np.mean(random_numbers_fx1_1000),2)
average_random_numbers_fx1_1000
```

[189]: 0.17

Calculamos la varianza para tamaño de muestra n=1000

[190]:
```
import numpy as np

variance_random_numbers_fx1_1000= round(np.var(random_numbers_fx1_1000),2)
variance_random_numbers_fx1_1000
```

[190]: 8.09

[ ]: