Type *Markdown* and LaTeX: α^2

Type *Markdown* and LaTeX: α^2

a) Genere una tabla con el histograma de los datos con el intervalo, frecuencia observada, Frecuencia relativa, frecuencia acumulada.
b) Grafique la función fdp((frecuencia relativa) y FDA (frecuencia acumulada)

#	Intervalo-Clase	Frecuencia Observada	Frecuencia relativa(fdp)	Frecuencia acumulada (FDA)
1	(min, max)			
2				
..				
...				
...				

c) Estimar la media y la varianza .. etc.
d) Genere 10 números aleatorios con los siguientes parámetros Xo=349, a= 347,c=47,M=1000
e) Encuentre los valores de los tiempos de servicio para los 10 números aleatorios generados y grafique los resultados de los 10 datos.

#	# ALEATORIO	FDA	TIEMPO DE SERVICIO SIMULADO
1			
2			
3			
8			
9			
10			

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import scipy.stats as stats
from scipy.stats import poisson

datos = pd.read_excel(r"",)
datos
```

Out[1]:

	tiempo_servicio
0	0.1
1	4.1
2	4.9
3	4.8
4	15.9
...	...
67	6.9
68	3.1
69	1.6
70	2.1
71	1.9

72 rows × 1 columns

a) Genere una tabla con el histograma de los datos con el intervalo, frecuencia observada, Frecuencia relativa, frecuencia acumulada.

Frecuencia relativa

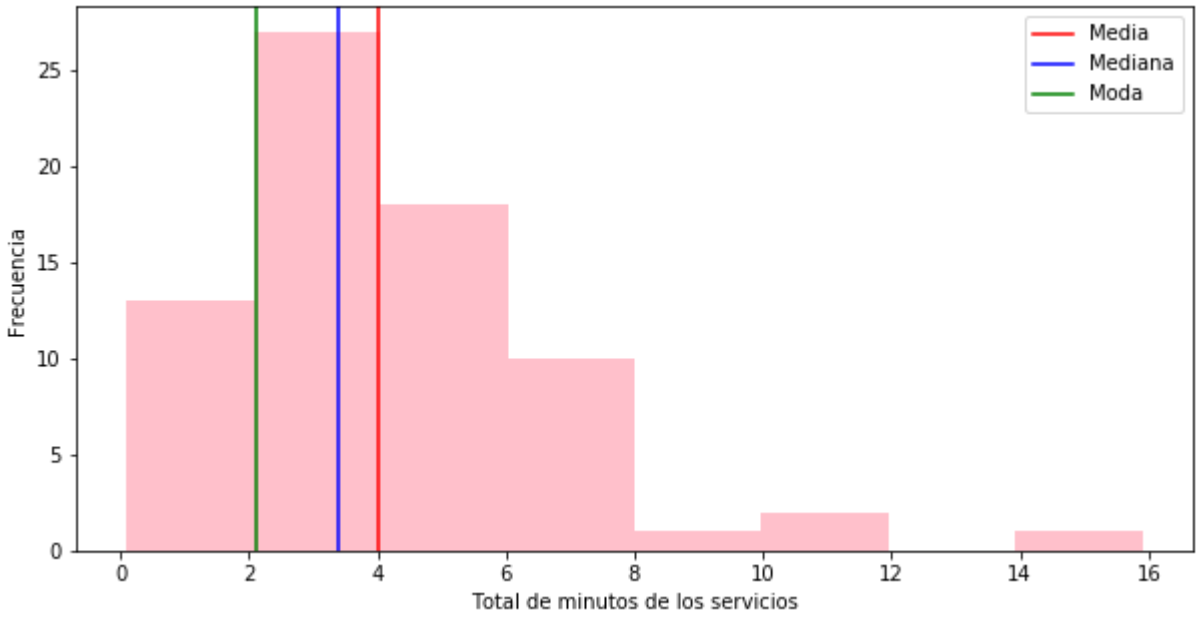
In [5]:
```
# OBTENER LOS DATOS UNICOS DE LA TABLA
lis = datos["tiempo_servicio"].unique()
lis
dfclases=pd.DataFrame(lis,columns=["tiempo_servicio"])
datafi=pd.crosstab(index=datos["tiempo_servicio"], columns = "Fi")
li = datafi.values
dfclases["Fi"] = li
total = dfclases.sum(axis=0)
datahi = dfclases["Fi"]/total["Fi"] # aqui calculamos la frecuencia
datahi.values
# agregamos nueva columna de frecuencia relativa
dfclases["hi"] = datahi
dfclases
```

Out[5]:

	tiempo_servicio	Fi	hi
0	0.1	2	0.027778
1	4.1	1	0.013889
2	4.9	1	0.013889
3	4.8	3	0.041667
4	15.9	3	0.041667
5	6.7	1	0.013889
6	2.1	1	0.013889
7	2.3	1	0.013889
8	2.5	4	0.055556
9	3.3	1	0.013889
10	3.8	1	0.013889
11	6.1	1	0.013889
12	2.8	1	0.013889
13	5.9	2	0.027778
14	3.4	1	0.013889
15	3.1	2	0.027778
16	0.4	3	0.041667
17	2.7	3	0.041667
18	0.9	3	0.041667
19	2.9	2	0.027778
20	4.5	1	0.013889
21	1.1	2	0.027778
22	4.2	3	0.041667
23	4.6	1	0.013889
24	7.2	2	0.027778
25	5.1	1	0.013889
26	2.6	1	0.013889

	tiempo_servicio	Fi	hi
27	2.4	1	0.013889
28	11.5	2	0.027778
29	10.3	4	0.055556
30	4.3	1	0.013889
31	2.2	1	0.013889
32	5.2	1	0.013889
33	8.2	2	0.027778
34	7.3	1	0.013889
35	3.5	2	0.027778
36	7.9	1	0.013889
37	6.2	1	0.013889
38	5.8	1	0.013889
39	1.4	1	0.013889
40	0.5	1	0.013889
41	7.1	1	0.013889
42	6.9	1	0.013889
43	1.6	1	0.013889
44	1.9	1	0.013889

In [2]:
```
x=datos["tiempo_servicio"]
plt.figure(figsize=(10,5))
plt.hist(x,bins=8,color='pink')
plt. axvline(x. mean(),color='red',label='Media')
plt. axvline(x. median(),color='blue',label='Mediana')
plt. axvline(x. mode()[0],color='green',label='Moda')
plt.xlabel('Total de minutos de los servicios')
plt.ylabel('Frecuencia')
plt.legend()
plt.show()
```



Frecuencia absoluta

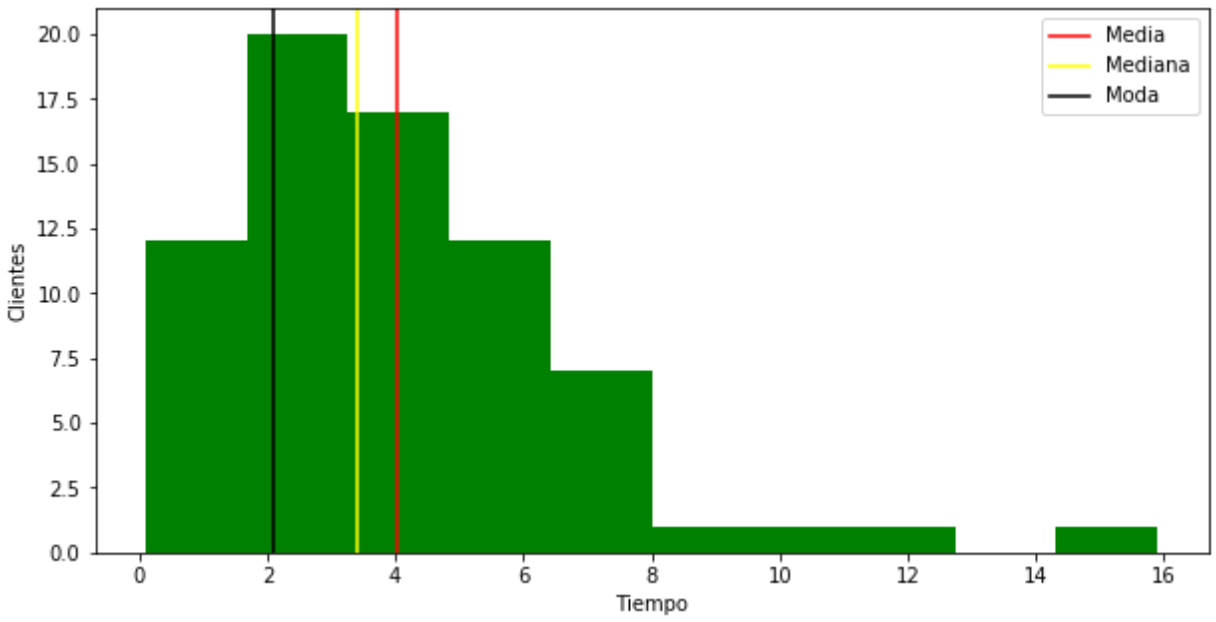
In [6]:
```
lis = datos["tiempo_servicio"].unique()
dfclases=pd.DataFrame(lis,columns=["tiempo_servicio"])
datafi=pd.crosstab(index=datos["tiempo_servicio"], columns = "Fi")
li = datafi.values
dfclases["Fi"] = li
dfclases
```

Out[6]:

	tiempo_servicio	Fi
0	0.1	2
1	4.1	1
2	4.9	1
3	4.8	3
4	15.9	3
5	6.7	1
6	2.1	1
7	2.3	1
8	2.5	4
9	3.3	1
10	3.8	1
11	6.1	1
12	2.8	1
13	5.9	2
14	3.4	1
15	3.1	2
16	0.4	3
17	2.7	3
18	0.9	3
19	2.9	2
20	4.5	1
21	1.1	2
22	4.2	3
23	4.6	1
24	7.2	2
25	5.1	1
26	2.6	1
27	2.4	1
28	11.5	2
29	10.3	4
30	4.3	1

	tiempo_servicio	Fi
31	2.2	1
32	5.2	1
33	8.2	2
34	7.3	1
35	3.5	2
36	7.9	1
37	6.2	1
38	5.8	1
39	1.4	1
40	0.5	1
41	7.1	1
42	6.9	1
43	1.6	1
44	1.9	1

In [3]:
```
x=datos["tiempo_servicio"]
plt.figure(figsize=(10,5))
plt.hist(x,bins=None,color='green')
plt. axvline(x. mean(),color='red',label='Media')
plt. axvline(x. median(),color='yellow',label='Mediana')
plt. axvline(x. mode()[0],color='black',label='Moda')
plt.xlabel('Tiempo')
plt.ylabel('Clientes')
plt.legend()
plt.show()
```



b) Grafique la función fdp((frecuencia relativa) y FDA

(frecuencia acumulada)

In [34]:
```
lis = datos["tiempo_servicio"].sort_values(ascending=True).unique()
dat = pd.DataFrame(lis, columns=["intervalo-clases"])
datafi = pd.crosstab(index=datos["tiempo_servicio"], columns = "fi")
li = datafi.values
dat["fi"] = li
datahi = 100 * datafi["fi"] / 50
datahi = datahi.values
dat["hi"] = datahi
Fi = dat["fi"].values
a = []
b = 0
for c in Fi:
    b = c + b
    a.append(b)
dat["Fi"] = a
dat["Hi"] = a
dat
```

Out[34]:

	intervalo-clases	fi	hi	Fi	Hi
0	0.1	2	4.0	2	2
1	0.4	1	2.0	3	3
2	0.5	1	2.0	4	4
3	0.9	3	6.0	7	7
4	1.1	3	6.0	10	10
5	1.4	1	2.0	11	11
6	1.6	1	2.0	12	12
7	1.9	1	2.0	13	13
8	2.1	4	8.0	17	17
9	2.2	1	2.0	18	18
10	2.3	1	2.0	19	19
11	2.4	1	2.0	20	20
12	2.5	1	2.0	21	21
13	2.6	2	4.0	23	23
14	2.7	1	2.0	24	24
15	2.8	2	4.0	26	26
16	2.9	3	6.0	29	29
17	3.1	3	6.0	32	32
18	3.3	3	6.0	35	35
19	3.4	2	4.0	37	37
20	3.5	1	2.0	38	38
21	3.8	2	4.0	40	40
22	4.1	3	6.0	43	43
23	4.2	1	2.0	44	44

	intervalo-clases	fi	hi	Fi	Hi
24	4.3	2	4.0	46	46
25	4.5	1	2.0	47	47
26	4.6	1	2.0	48	48
27	4.8	1	2.0	49	49
28	4.9	2	4.0	51	51
29	5.1	4	8.0	55	55
30	5.2	1	2.0	56	56
31	5.8	1	2.0	57	57
32	5.9	1	2.0	58	58
33	6.1	2	4.0	60	60
34	6.2	1	2.0	61	61
35	6.7	2	4.0	63	63
36	6.9	1	2.0	64	64
37	7.1	1	2.0	65	65
38	7.2	1	2.0	66	66
39	7.3	1	2.0	67	67
40	7.9	1	2.0	68	68
41	8.2	1	2.0	69	69
42	10.3	1	2.0	70	70
43	11.5	1	2.0	71	71
44	15.9	1	2.0	72	72

c) Estimar la media y la varianza .. etc

In [19]:
```
datos.describe()
```

Out[19]:

	tiempo_servicio
count	72.000000
mean	4.018056
std	2.714298
min	0.100000
25%	2.275000
50%	3.400000
75%	5.100000
max	15.900000

In [23]:
```
print('La varianza es: ', datos.var())
```

```
La varianza es:  tiempo_servicio    7.367416
dtype: float64
```

In [24]:
```
print('La media de los tiempos es: ', datos.median())
```

```
La media de los tiempos es:  tiempo_servicio    3.4
dtype: float64
```

In [25]:
```
print('La desviacion estandar del tiempo es: ', datos.std())
```

```
La desviacion estandar del tiempo es:  tiempo_servicio    2.714298
dtype: float64
```

In [26]:
```
print('La moda de los datos es ', datos.mode())
```

```
La moda de los datos es     tiempo_servicio
0                2.1
1                5.1
```

d) Genere 10 números aleatorios con los siguientes parámetros Xo=349, a=347,c=47,M=1000

In [27]:
```
n, m, a, x0, c = 10, 1000, 347, 349, 47
x = [1] * n
r = [0.1] * n
print (" Método de Congruencia Lineal ")
print("------------------------------")
print ("n=cantidad de números generados : ", n)
print()
print ("m : ", m)
print ("a : ", a)
print ("c : ", c)
print ("Xo : ", x0 )
for i in range(0, n):
    x[i] = ((a*x0)+c) % m
    x0 = x[i]
    r[i] = x0 / m
# Guardamos los datos en un dataframe
d = {'Xn': x, 'ri': r }
dfMCL = pd.DataFrame(data=d)
dfMCL
```

```
 Método de Congruencia Lineal
------------------------------
n=cantidad de números generados :  10

m :  1000
a :  347
c :  47
Xo :  349
```

Out[27]:

	Xn	ri
0	150	0.150
1	97	0.097
2	706	0.706
3	29	0.029
4	110	0.110
5	217	0.217
6	346	0.346
7	109	0.109
8	870	0.870
9	937	0.937

e) Encuentre los valores de los tiempos de servicio para los 10 números aleatorios generados y grafique los resultados de los 10 datos.

In [47]:
```
#Longitud de la clase
r=max(datos['tiempo_servicio'])-min(datos['tiempo_servicio'])
long=len(datos)/r
print('La longitud es ->',long)
```

```
La longitud es -> 4.556962025316455
```

In [45]:
```
landa=15.9-0.1
dfexp = dfMCL['ri']
exp_x = dfexp.values*(-1/landa)*np.log(dfexp)
dfMCL["FDA"] = exp_x
dfMCL[:10]
```

Out[45]:

	Xn	ri	exp_x	FDA
0	150	0.150	0.018011	0.018011
1	97	0.097	0.014323	0.014323
2	706	0.706	0.015556	0.015556
3	29	0.029	0.006498	0.006498
4	110	0.110	0.015367	0.015367
5	217	0.217	0.020984	0.020984
6	346	0.346	0.023241	0.023241
7	109	0.109	0.015290	0.015290
8	870	0.870	0.007668	0.007668
9	937	0.937	0.003859	0.003859

In [ ]:
```

```