Nombre: ***Macias Pico Josselyn Stefany***

Curso: ***Sexto "B"***

Materia: ***Modelamiento y simulacion***

Docente: ***Ing. Jorge Anibal Moya Delgado***

1. Los datos siguientes representan el periodo en segundos, necesarios para trasmitir un mensaje:

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

datos = pd.read_excel('C:/Users/Villamar/Desktop/JOSS/SEXTOSEMESTRE/MODELAMIENTOS
datos
```

Out[1]:

	segundos
0	5.60
1	5.80
2	12.80
3	17.30
4	21.10
5	22.10
6	23.80
7	23.90
8	25.80
9	28.60
10	29.20
11	30.01
12	35.20
13	35.90
14	36.40
15	36.40
16	36.40
17	36.80
18	38.70
19	38.70
20	39.20
21	47.90
22	48.10
23	51.90
24	56.40
25	58.60
26	58.60
27	58.70
28	59.30
29	61.30

	segundos
30	63.60
31	65.30
32	66.10
33	67.30
34	70.90
35	71.30
36	71.30
37	72.50
38	76.40
39	76.70
40	77.40
41	78.70
42	82.70
43	83.90
44	88.90
45	89.50
46	89.50
47	93.40
48	93.50
49	94.80

58.6	89.5	35.2	36.4	58.7
71.3	38.7	61.3	59.3	93.4
94.8	47.9	56.4	22.1	48.1
67.3	25.8	65.3	30.01	72.5
70.9	5.8	88.9	76.4	17.3
66.1	77.4	23.9	23.8	36.8
36.4	5.6	93.5	36.4	76.7
39.2	83.9	78.7	51.9	63.6
58.6	89.5	12.8	28.6	82.7
71.3	38.7	21.1	35.9	29.2

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

a. Frecuencia Observada (Absoluta)

In [44]:
```
lis = datos["segundos"].unique()
dfclases=pd.DataFrame(lis,columns=["segundos"])
datafi=pd.crosstab(index=datos["segundos"], columns = "Fi")
li = datafi.values
dfclases["Fi"] = li
dfclases
```

Out[44]:

	segundos	Fi
0	5.60	1
1	5.80	1
2	12.80	1
3	17.30	1
4	17.80	1
5	21.10	1
6	22.10	1
7	23.80	1
8	23.90	1
9	25.80	1
10	28.60	1
11	29.20	1
12	30.01	1
13	34.70	1
14	35.20	1
15	35.80	1
16	35.90	1
17	36.40	3
18	36.80	1
19	38.70	1
20	39.20	1
21	47.90	1
22	48.10	1
23	48.20	1
24	51.90	1
25	56.40	1
26	58.60	1
27	58.70	1
28	59.30	1
29	61.30	1
30	63.60	1

	segundos	Fi
31	65.30	1
32	66.10	1
33	67.30	1
34	70.90	1
35	71.30	1
36	72.50	1
37	76.40	1
38	76.70	1
39	77.40	1
40	78.70	1
41	82.70	1
42	83.90	1
43	88.90	1
44	89.50	1
45	93.40	1
46	93.50	1
47	94.80	1

b. Frecuencia Relativa

In [6]:
```
# OBTENER LOS DATOS UNICOS DE LA TABLA
lis = datos["segundos"].unique()
lis
dfclases=pd.DataFrame(lis,columns=["segundos"])
datafi=pd.crosstab(index=datos["segundos"], 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[6]:

	segundos	Fi	hi
0	5.60	1	0.02
1	5.80	1	0.02
2	12.80	1	0.02
3	17.30	1	0.02
4	21.10	1	0.02
5	22.10	1	0.02
6	23.80	1	0.02
7	23.90	1	0.02
8	25.80	1	0.02
9	28.60	1	0.02
10	29.20	1	0.02
11	30.01	1	0.02
12	35.20	1	0.02
13	35.90	1	0.02
14	36.40	3	0.06
15	36.80	1	0.02
16	38.70	2	0.04
17	39.20	1	0.02
18	47.90	1	0.02
19	48.10	1	0.02
20	51.90	1	0.02
21	56.40	1	0.02
22	58.60	2	0.04
23	58.70	1	0.02
24	59.30	1	0.02
25	61.30	1	0.02
26	63.60	1	0.02

	segundos	Fi	hi
27	65.30	1	0.02
28	66.10	1	0.02
29	67.30	1	0.02
30	70.90	1	0.02
31	71.30	2	0.04
32	72.50	1	0.02
33	76.40	1	0.02
34	76.70	1	0.02
35	77.40	1	0.02
36	78.70	1	0.02
37	82.70	1	0.02
38	83.90	1	0.02
39	88.90	1	0.02
40	89.50	2	0.04
41	93.40	1	0.02
42	93.50	1	0.02
43	94.80	1	0.02

c. Frecuencia absoluta acumalada

In [21]:
```
lis = datos["segundos"].sort_values(ascending=True).unique()
dat = pd.DataFrame(lis, columns=["intervalo-clases"])
datafi = pd.crosstab(index=datos["segundos"], columns = "frecuencia")
li = datafi.values
dat["frecuencia"] = li
datahi = 100 * datafi["frecuencia"] / 50
datahi = datahi.values
Fi = dat["frecuencia"].values
a = []
b = 0
for c in Fi:
    b = c + b
    a.append(b)
dat["Frecuencia_absoluta_acomulada"] = a
dat
```

Out[21]:

	intervalo-clases	frecuencia	Frecuencia_absoluta_acomulada
0	5.60	1	1
1	5.80	1	2
2	12.80	1	3
3	17.30	1	4
4	21.10	1	5
5	22.10	1	6
6	23.80	1	7
7	23.90	1	8
8	25.80	1	9
9	28.60	1	10
10	29.20	1	11
11	30.01	1	12
12	35.20	1	13
13	35.90	1	14
14	36.40	3	17
15	36.80	1	18
16	38.70	2	20
17	39.20	1	21
18	47.90	1	22
19	48.10	1	23
20	51.90	1	24
21	56.40	1	25
22	58.60	2	27
23	58.70	1	28
24	59.30	1	29
25	61.30	1	30

	intervalo-clases	frecuencia	Frecuencia_absoluta_acomulada
26	63.60	1	31
27	65.30	1	32
28	66.10	1	33
29	67.30	1	34
30	70.90	1	35
31	71.30	2	37
32	72.50	1	38
33	76.40	1	39
34	76.70	1	40
35	77.40	1	41
36	78.70	1	42
37	82.70	1	43
38	83.90	1	44
39	88.90	1	45
40	89.50	2	47
41	93.40	1	48
42	93.50	1	49
43	94.80	1	50

Tabla

In [22]:
```
lis = datos["segundos"].sort_values(ascending=True).unique()
dat = pd.DataFrame(lis, columns=["intervalo-clases"])
datafi = pd.crosstab(index=datos["segundos"], columns = "frecuencia")
li = datafi.values
dat["frecuencia"] = li
datahi = 100 * datafi["frecuencia"] / 50
datahi = datahi.values
dat["frecuencia_relativa"] = datahi
Fi = dat["frecuencia"].values
a = []
b = 0
for c in Fi:
    b = c + b
    a.append(b)
dat["Frecuencia_absoluta_acomulada"] = a
dat
```

Out[22]:

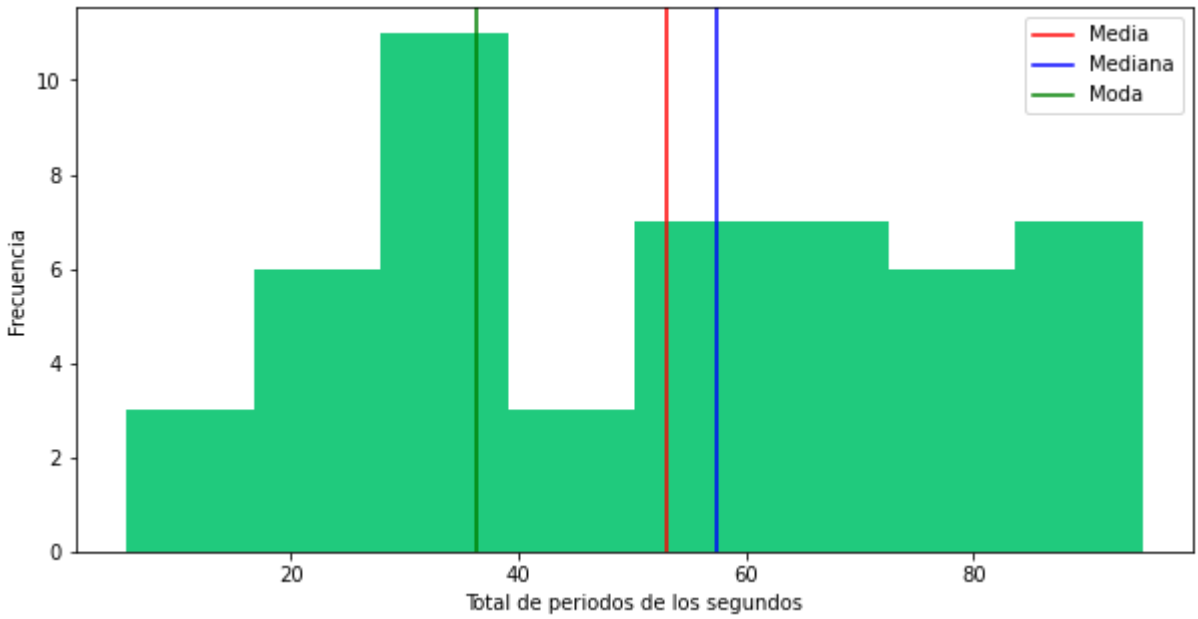
	intervalo-clases	frecuencia	frecuencia_relativa	Frecuencia_absoluta_acomulada
0	5.60	1	2.0	1
1	5.80	1	2.0	2
2	12.80	1	2.0	3
3	17.30	1	2.0	4
4	21.10	1	2.0	5
5	22.10	1	2.0	6
6	23.80	1	2.0	7
7	23.90	1	2.0	8
8	25.80	1	2.0	9
9	28.60	1	2.0	10
10	29.20	1	2.0	11
11	30.01	1	2.0	12
12	35.20	1	2.0	13
13	35.90	1	2.0	14
14	36.40	3	6.0	17
15	36.80	1	2.0	18
16	38.70	2	4.0	20
17	39.20	1	2.0	21
18	47.90	1	2.0	22
19	48.10	1	2.0	23
20	51.90	1	2.0	24
21	56.40	1	2.0	25
22	58.60	2	4.0	27
23	58.70	1	2.0	28
24	59.30	1	2.0	29

	intervalo-clases	frecuencia	frecuencia_relativa	Frecuencia_absoluta_acomulada
25	61.30	1	2.0	30
26	63.60	1	2.0	31
27	65.30	1	2.0	32
28	66.10	1	2.0	33
29	67.30	1	2.0	34
30	70.90	1	2.0	35
31	71.30	2	4.0	37
32	72.50	1	2.0	38
33	76.40	1	2.0	39
34	76.70	1	2.0	40
35	77.40	1	2.0	41
36	78.70	1	2.0	42
37	82.70	1	2.0	43
38	83.90	1	2.0	44
39	88.90	1	2.0	45
40	89.50	2	4.0	47
41	93.40	1	2.0	48
42	93.50	1	2.0	49
43	94.80	1	2.0	50

3. Grafique la función fdp((frecuencia relativa) y FDA (frecuencia acumulada)

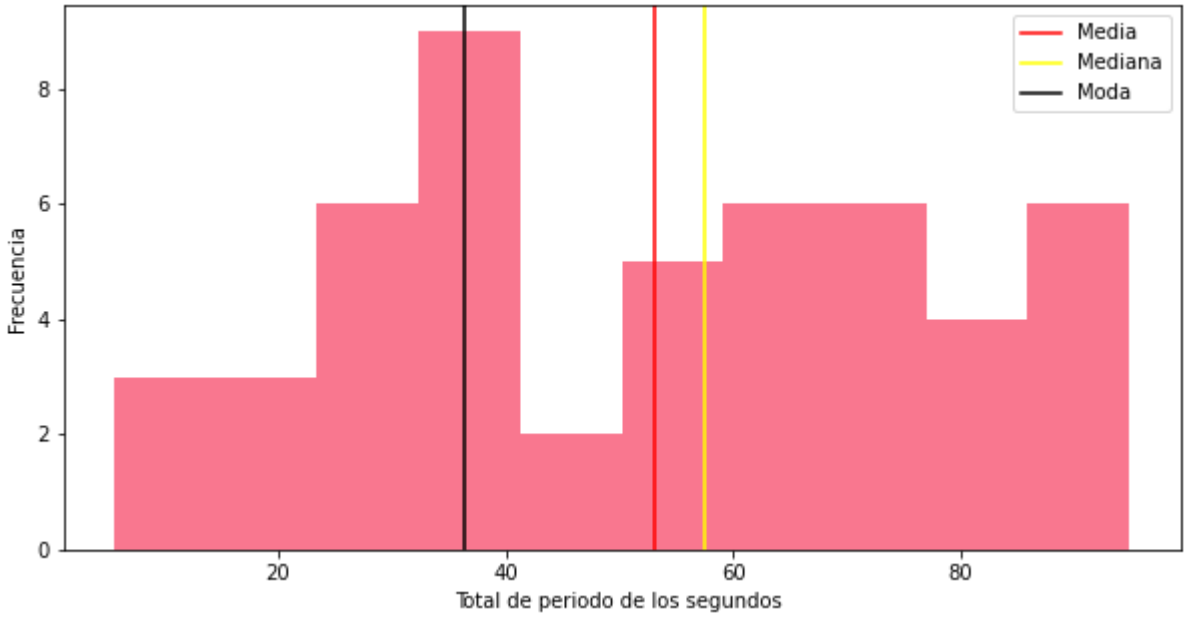
a. Histograma de la frecuencia relativa

In [23]:
```
x=datos["segundos"]
plt.figure(figsize=(10,5))
plt.hist(x,bins=8,color='#20CA7D')
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 periodos de los segundos')
plt.ylabel('Frecuencia')
plt.legend()
plt.show()
```



b. Histograma de la frecuencia absoluta acomulada

In [24]:
```
x=datos["segundos"]
plt.figure(figsize=(10,5))
plt.hist(x,bins=None,color='#F9778F')
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('Total de periodo de los segundos ')
plt.ylabel('Frecuencia')
plt.legend()
plt.show()
```



4. Estimar la media y la varianza .. etc.

a. Varianza

In [25]:
```
print('La varianza de los segundos es: \n\n', datos. var())
```

```
La varianza de los segundos es:

 segundos    637.068278
dtype: float64
```

b. Mediana

In [26]:
```
print('La mediana de los segundos es: \n\n', datos. median())
```

```
La mediana de los segundos es:

 segundos    57.5
dtype: float64
```

c. Moda

In [27]:
```
print('La moda de los datos de los segundos es: \n\n ', datos. mode())
```

```
La moda de los datos de los segundos es:

     segundos
0      36.4
```

d. Media Aritmetica

In [28]:
```
print('La media aritmetica de los datos de los segundos es: \n\n ', datos. mean()
```

```
La media aritmetica de los datos de los segundos es:

  segundos    53.0842
dtype: float64
```

e. Cuartiles

In [29]:
```
print('-----------------------------------------------------')
print('Los cuartiles de los datos de los segundos es: \n\n')
datos.describe()
```

```
-----------------------------------------------------
Los cuartiles de los datos de los segundos es:
```

Out[29]:

	segundos
count	50.000000
mean	53.084200
std	25.240212
min	5.600000
25%	35.375000
50%	57.500000
75%	72.200000
max	94.800000

f. Desviacion estandar

In [30]:
```
print('La desviacion estandar de los datos de los segundos es: \n\n', datos. std(
```

```
La desviacion estandar de los datos de los segundos es:

 segundos    25.240212
dtype: float64
```

5. Genere 10 números aleatorios con los siguientes parámetros Xo=349, a= 347,c=47,M=120

In [51]:
```
n, m, a, x0, c = 10, 120, 347, 349, 47
x = [1] * n
r = [0.1] * n
print (" Método de Congruencia Lineal ")
print("------------------------------")
print ("n  : ", n)
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  :  10
m  :  120
a  :  347
c  :  47
Xo :  349
```

Out[51]:

	Xn	ri
0	70	0.583333
1	97	0.808333
2	106	0.883333
3	109	0.908333
4	70	0.583333
5	97	0.808333
6	106	0.883333
7	109	0.908333
8	70	0.583333
9	97	0.808333

6. Encuentre los valores de los segundos para los 10 números aleatorios generados y grafique los resultados de los 10 datos

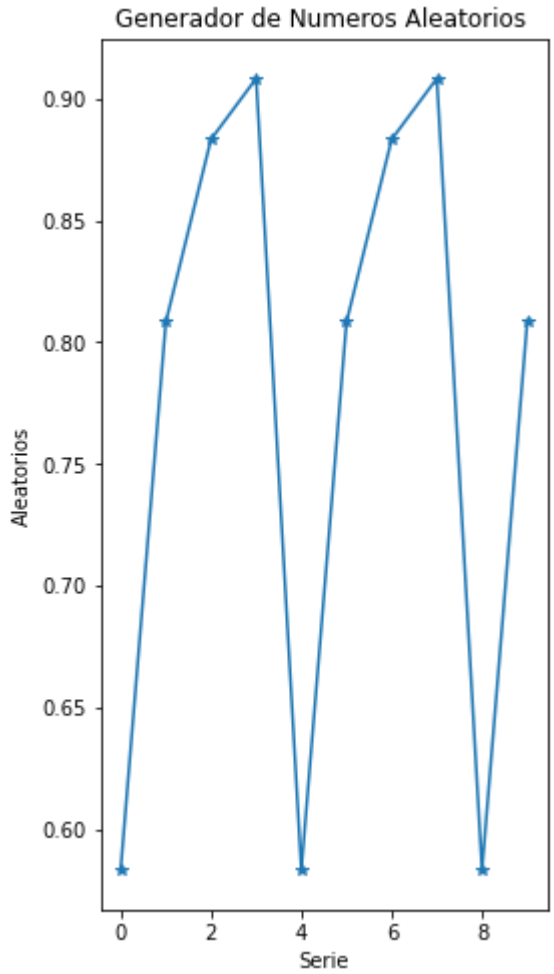
a. Valores de lo segundos para los numeros generados

#	# ALEATORIO	FDA	TIEMPO DE SERVICIO SIMULADO
1	70	1	0.583333
2	97	0	0.808333
3	106	0	0.883333
4	109	0	0.908333
5	70	1	0.583333
6	97	0	0.808333
7	106	0	0.883333
8	109	0	0.908333
9	70	1	0.583333
10	97	0	0.808333

b.Grafica de los numeros aleatorios generados

In [39]:
```
# Graficamos los numeros generados
plt.plot(r,marker='*')
plt.title('Generador de Numeros Aleatorios ')
plt.xlabel('Serie')
plt.ylabel('Aleatorios')
```

Out[39]:
```
Text(0, 0.5, 'Aleatorios')
```



7. Numero de binds

a. Valor mayor

In [41]:
```
x = datos["segundos"]
max_value = max(x)
print('Valor mayor es --> ', max_value)
```

```
Valor mayor es -->  94.8
```

b. Valor menor

In [43]:
```
x = datos["segundos"]
min_value = min(x)
print('Valor minimo es --> ', min_value)
```

```
Valor minimo es -->  5.6
```

b. Valor de R --> Valor de binds

In [65]:
```
R = max_value - min_value
print(R)
```

```
89.2
```

c. Valores de K

- Menos de 50

In [45]:
```
contador=0
for i in range(datos. count(). values[0]):
    valor = datos.values[i][0]
    if(valor<=50):
        contador = contador +1
proba_50 = contador +1
proba_50 = contador/len(datos)
print("Probabilidad de menos de 50" , proba_50, '%')
```

```
Probabilidad de menos de 50 0.46 %
```

- Entre 50 y 100

In [46]:
```
contador=0
for i in range(datos. count(). values[0]):
    valor = datos.values[i][0]
    if(valor>=50):
        if(valor<=100):
            contador = contador +1
proba_50_100 = contador +1
proba_50_100 = contador/len(datos)
print("Probabilidad entre 50 y 100" , proba_50_100, '%')
```

```
Probabilidad entre 50 y 100 0.54 %
```

- Entre 100 y 250

In [47]:
```
contador=0
for i in range(datos. count(). values[0]):
    valor = datos.values[i][0]
    if(valor>=100):
        if(valor<=250):
            contador = contador +1
proba_100_250 = contador +1
proba_100_250 = contador/len(datos)
print("Probabilidad entre 100 y 250" , proba_100_250, '%')
```

```
Probabilidad entre 100 y 250 0.0 %
```

- Mas de 250

In [48]:
```
contador=0
for i in range(datos. count(). values[0]):
    valor = datos.values[i][0]
    if(valor>250):
        contador = contador +1
proba_250 = contador +1
proba_250 = contador/len(datos)
print("Probabilidad mayores de 250" , proba_250, '%')
```

```
Probabilidad mayores de 250 0.0 %
```

- Longitud

In [49]:
```
#Longitud de la clase
r=max(datos['segundos'])-min(datos['segundos'])
long=len(datos)/r
print('La longitud es ->',long)
```

```
La longitud es -> 0.5605381165919282
```

- Valores de los tiempos de servicio

In [50]:
```
landa=15.9-0.1
dfexp = dfMCL['ri']
exp_x = dfexp.values*(-1/landa)*np.log(dfexp)
dfMCL["FDA"] = exp_x
dfMCL[:10]
```

Out[50]:

	Xn	ri	FDA
0	70	0.583333	0.019900
1	97	0.808333	0.010886
2	106	0.883333	0.006935
3	109	0.908333	0.005527
4	70	0.583333	0.019900
5	97	0.808333	0.010886
6	106	0.883333	0.006935
7	109	0.908333	0.005527
8	70	0.583333	0.019900
9	97	0.808333	0.010886

In [ ]: