Gennaio 2016

In [14]:

```
import pandas as pd
import numpy as np
import scipy.stats as st
import matplotlib.pyplot as plt
```

Esercizio 0 ¶

0.1,0.2 Grafici

$D_X = 2, \dots, 14, X \sim Unif(D_X)$

$X \sim Geom(p), n \in \{0, 1, 2, \dots\}$

0.3

$$E(X) = \frac{1-p}{p}$$

0.4

$$p = \frac{1}{E(X)+1}$$

0.5

$$Var(X) = \frac{1-p}{p^2} = E(X)(E(X)+1)$$

Esercizio 1

In [10]:

```
df = pd.read_csv('Comune_Bergamo_-_Incidenti_stradali.csv')
df.columns
```

Out[10]:

```
Index(['Protocollo', 'Anno', 'Data', 'Ora', 'Localita', 'NaturaIncidente',
       'N_Illesi', 'N_Feriti', 'N_Riservata', 'N_Morti', 'Pedoni',
       'Velocipedi', 'Ciclomotori_Motocicli', 'Mezzi_Pesanti',
       'Localizzazione'],
      dtype='object')
```

1.1

In [11]:

```
len(df)
```

Out[11]:

```
28040
```

1.2

- Protocollo categorici
- Anno ordinali
- Data ordinali
- Ora ordinali
- Localita categorici
- NaturaIncidente categorici
- N_Illesi scalari
- N_Feriti scalari
- N_Riservata 28040 non-null int64
- N_Morti scalari
- Pedoni categorici
- Velocipedi categorici
- Ciclomotori_Motocicli categorici
- Mezzi_Pesanti categorici
- Localizzazione categorici

In [12]:

```
df.info()
```

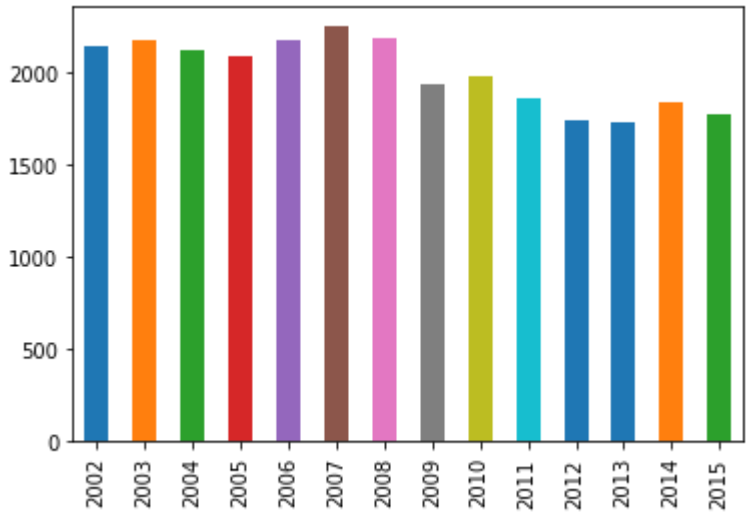
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28040 entries, 0 to 28039
Data columns (total 15 columns):
Protocollo               28040 non-null object
Anno                     28040 non-null int64
Data                     28040 non-null object
Ora                      28040 non-null object
Localita                 28040 non-null object
NaturaIncidente          28040 non-null object
N_Illesi                 28040 non-null int64
N_Feriti                 28040 non-null int64
N_Riservata              28040 non-null int64
N_Morti                  28040 non-null int64
Pedoni                   28040 non-null bool
Velocipedi               28040 non-null bool
Ciclomotori_Motocicli    28040 non-null bool
Mezzi_Pesanti            28040 non-null bool
Localizzazione           27954 non-null object
dtypes: bool(4), int64(5), object(6)
memory usage: 2.5+ MB
```

1.3

Modello Uniforme Anno intorno ai 2000

In [21]:

```
df['Anno'].value_counts(sort=False).plot.bar()
plt.show()
```



1.4

In [33]:

```
n_feriti = pd.crosstab(index=df['N_Feriti'],columns="Frequenza Relativa",colnames=[''],
normalize=True)
n_feriti
```

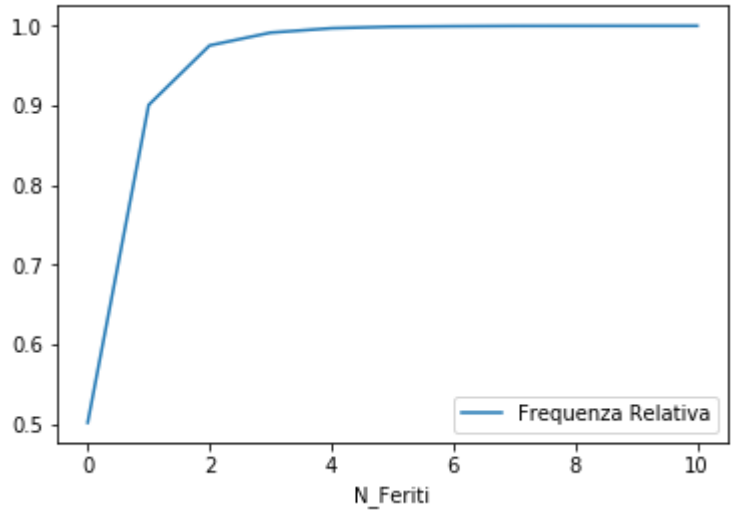
Out[33]:

	Frequenza Relativa
N_Feriti	
0	0.501676
1	0.398930
2	0.074750
3	0.016084
4	0.005528
5	0.001961
6	0.000571
7	0.000357
8	0.000071
9	0.000036
10	0.000036

1.5

In [31]:

```
### incidenti, feriti ???
n_feriti.cumsum().plot()
plt.show()
```



1.6

!A = "nessuno ferito"

A = "almeno un ferito" = 1-P(!A)

In [43]:

```
1-df['N_Feriti'].value_counts(normalize=True).sort_index()[0]
#1-n_feriti[:1]
```

Out[43]:

```
0.49832382310984313
```

1.7

valore atteso -> stimo con media campionaria

In [44]:

```
df['N_Feriti'].mean()
```

Out[44]:

```
0.6357703281027104
```

1.8

Osservando il grafico e notando dai dati quì sotto che media e deviazione standard sono vicini possiamo assumere che il modello probabilistico è geometrico

In [45]:

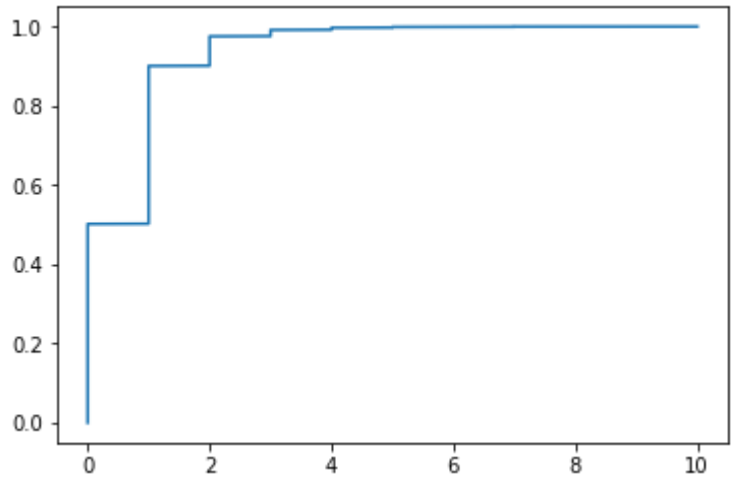
```
df['N_Feriti'].describe()
```

Out[45]:

```
count    28040.000000
mean         0.635770
std          0.790586
min          0.000000
25%          0.000000
50%          0.000000
75%          1.000000
max         10.000000
Name: N_Feriti, dtype: float64
```

In [48]:

```
from statsmodels.distributions.empirical_distribution import ECDF
dist = ECDF(df['N_Feriti'].dropna())
plt.plot(dist.x, dist.y)
plt.show()
```



1.9

stima parametro della distribuzione

In [53]:

```
p = (1/(1+df['N_Feriti'].mean()))#vedi esercizio 0
p
```

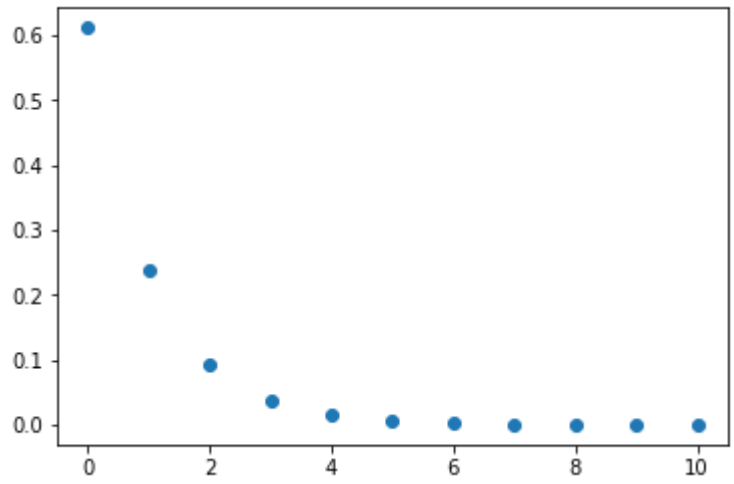
Out[53]:

```
0.6113327664769879
```

1.10

In [56]:

```
X = st.geom(p,loc=-1)
x = np.arange(11) #numero max di incidenti
plt.plot(x,X.pmf(x),'o')
plt.show()
```

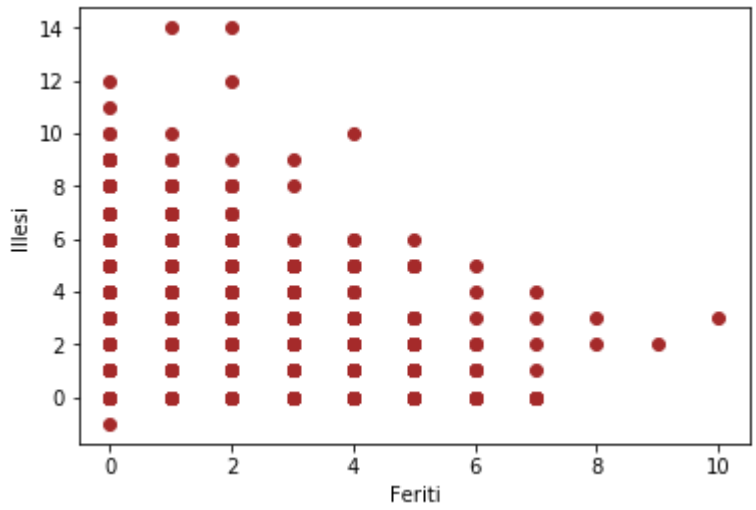


1.11

diagramma dispesione N_Feriti e N_Illesi

In [64]:

```
plt.scatter(df['N_Feriti'],df['N_Illesi'],color="brown")
plt.xlabel('Feriti')
plt.ylabel('Illesi')
plt.show()
```



1.12

Correlazione tra feriti e illesi

In [65]:

```
df['N_Feriti'].corr(df['N_Illesi'])
```

Out[65]:

```
-0.3070519976103214
```

Esercizio 2

Dati $X_1, \ldots, X_n$ campione casuale, con X= numero illesi incidente

2.1

Stimatore del valore atteso

$$T_n = \overline{X}$$

In [58]:

```
df['N_Illesi'].mean()
```

Out[58]:

```
1.7973965763195434
```

2.2

$Var(\overline{X}) = \frac{1}{n} Var(X)$

$$\sigma(\overline{X}) = \sqrt{\frac{1}{n} Var(X)}$$

In [59]:

```
std = (df['N_Illesi'].var()/df['N_Illesi'].dropna().count())**0.5
std
```

Out[59]:

```
0.007751809852772057
```

2.3

Dimostrare che $P(|Z| < k) = 2\Phi(k) - 1$

$$P(|Z| < k) = P(-k < Z < k) = \Phi(k) - \Phi(-k) = \Phi(k) - (1 - \Phi(k)) = 2\Phi(k) - 1$$

2.4

Determinare k tale che $P(|Z| < k) = 0.99$. Riprendo l'esercizio precedente

$$2\Phi(k) - 1 = 0.99$$

$$\Phi(k) = \frac{1.99}{2} = 0.995$$

$$k = \Phi^{-1}(0,955)$$

In [6]:

```
Z = st.norm()
Z.ppf(0.995)
```

Out[6]:

```
2.5758293035489004
```

2.5

$$P(|T_n - \mu| < \epsilon) = 0.99$$

$$2\Phi(\frac{\epsilon\sqrt{n}}{\sigma}) - 1 = 0.99$$

$$P(|\overline{X} - E(\bar{(X)})| \leq 0.99) > \frac{Var(X)}{n * 0.99}$$

$$\epsilon = \frac{\phi^{-1}(0.995)\sigma}{\sqrt{n}}$$

In [60]:

```
(Z.ppf(0.995) * df['N_Illesi'].std())/df['N_Illesi'].dropna().count()**0.5
```

Out[60]:

```
0.019967338974309353
```

Febbraio 2016

In [40]:

```
import pandas as pd
import numpy as np
import math
import scipy.stats as st
import matplotlib.pyplot as plt
```

Esercizio 0

0.1

$i = 1, \dots, 7$

$$X_i = \begin{cases} 0 & giorno\ i\ non\ piove \\ 1 & altrimenti \end{cases}$$

0.2

E_1 = "piove in almeno un giorno infrasettimanale"

$$P(E_1) = 1 - \sum_{i=1}^{5} f_X(i) = 1 - (1-p)^5$$

E_2 = "non piove nel fine settimana"

$$P(E_2) = \sum_{i=6}^{7} f_X(i) = (1-p)^2$$

0.3

Gli eventi considerati sono indipendenti, ma non mutuamente esclusivi. Questo perchè la pioggia in un giorno non influenza gli altri.

0.4

Calcolo la probabilità di E_1, E_2, E_3 = "piove in almeno un giorno infrasettimanale, ma non durante il fine settimana",E_4 = "piove in almeno un giorno infrasettimanale oppure non piove nel fine settimana"

$$P(E_3) = P(E_1 \cap E_2) = P(E_1) * P(E_2)$$

$E_4 = E_1 \cup E_2$

$$P(E_1 \cup E_2) = P(E_1) + P(E_2) - P(E_1 \cap E_2)$$

In [11]:

```
p = 0.4
e1 = 1 - (1-p)**5 ### E_1
e2 = (1-p)**2 ### E_2
e3 = e1*e2
e4 = e1+e2-e3
print(e1,e2,e3,e4)
```

```
0.9222400000000001 0.36 0.33200640000000003 0.9502336
```

Esercizio 1

1.1

$$D_X = \{1, \dots, 7\}$$

1.2

$$X \sim Binom(p, 7)$$

$$f_X(x) = \binom{7}{x} p^x (1-p)^{7-x} I_{0,\dots,7}(x)$$

1.3

$E(X) = p, Var(X) = p(1-p)$

$$P(|T_n - p| \leq 0.25) \geq 0.4$$

$$2\Phi(\frac{0.25\sqrt{n}}{\sqrt{p(1-p)}}) - 1 \geq 0.4$$

$$2\Phi(\frac{0.25\sqrt{n}}{\sqrt{p(1-p)}}) \geq 1.4$$

$$\frac{0.25\sqrt{n}}{\sqrt{p(1-p)}} \geq \Phi^{-1}(0.7)$$

Sappiamo che p = 0.4

$$\frac{0.25\sqrt{n}}{\sqrt{0.4(1-0.4)}} \geq \Phi^{-1}(0.7)$$

$$\sqrt{n} \geq \frac{\Phi^{-1}(0.7)\sqrt{0.24}}{0.25}$$

In [25]:

```
Z = st.norm()
p = .4
n = ((Z.ppf(0.7)*(math.sqrt(0.24)))/(0.25))**2
n
```

Out[25]:

```
1.0559842472772705
```

Esercizio 2

In [36]:

```
df = pd.read_csv('DATI-AMBIENTE.txt',sep=";",decimal=",",na_values=' ')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 15 columns):
PROVINCIA                           150 non-null object
C6H6                                27 non-null object
SO2                                 38 non-null object
CO                                  57 non-null object
NO2                                 138 non-null float64
O3                                  74 non-null float64
O3_GIORNI_SUPERAMENTO_TOLLERANZA    77 non-null float64
O3_GIORNI_SUPERAMENTO_ALLARME       77 non-null float64
PM10                                71 non-null float64
PM2_5                               26 non-null float64
Pb                                  14 non-null object
As                                  14 non-null object
Ni                                  14 non-null object
Cd                                  14 non-null object
BaP                                 13 non-null object
dtypes: float64(6), object(9)
memory usage: 17.7+ KB
```

2.1

In [37]:

```
len(df)
```

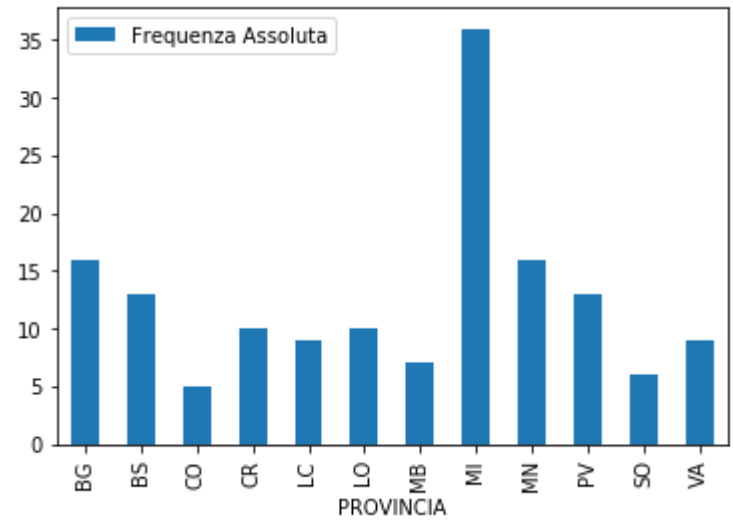
Out[37]:

150

2.2

In [42]:

```
fprov = pd.crosstab(index=df['PROVINCIA'],colnames=[''],columns="Frequenza Assoluta")
fprov.plot.bar()
plt.show()
```



2.3

La provincia meno rappresentata è Como, mentre quella più rappresentata è Milano

In [51]:

```
print(df['PROVINCIA'].value_counts().sort_values().head(1))
print(df['PROVINCIA'].value_counts().sort_values().tail(1))
```

```
CO    5
Name: PROVINCIA, dtype: int64
MI    36
Name: PROVINCIA, dtype: int64
```

2.4

L'eterogeneità è molto vicino a 1 quindi c'è una buona distribuzione.

In [52]:

```
def gini(series):
    return 1 - sum(series.value_counts(normalize=True)
                         .map(lambda f: f**2))

def normalized_gini(series):
    s = len(series.unique())
    return s * gini(series)/(s-1)

normalized_gini(df['PROVINCIA'].dropna())
```

Out[52]:

```
0.9639757575757577
```

2.5

In [54]:

```
df['C6H6'].value_counts().describe()
```

Out[54]:

```
count    18.000000
mean      1.500000
std       0.857493
min       1.000000
25%       1.000000
50%       1.000000
75%       2.000000
max       4.000000
Name: C6H6, dtype: float64
```

In [55]:

```
df['SO2'].value_counts().describe()
```

Out[55]:

```
count    13.000000
mean      2.923077
std       2.498718
min       1.000000
25%       1.000000
50%       1.000000
75%       4.000000
max       8.000000
Name: SO2, dtype: float64
```

In [56]:

```
df['CO'].value_counts().describe()
```

Out[56]:

```
count    11.000000
mean      5.181818
std       3.919647
min       1.000000
25%       2.000000
50%       4.000000
75%       7.000000
max      13.000000
Name: CO, dtype: float64
```

2.6

2.7

Concentrazione a: cadmio

Concentrazione b: benzoapirene

2.8

In [71]:

```
df['O3'].value_counts(normalize=True)
```

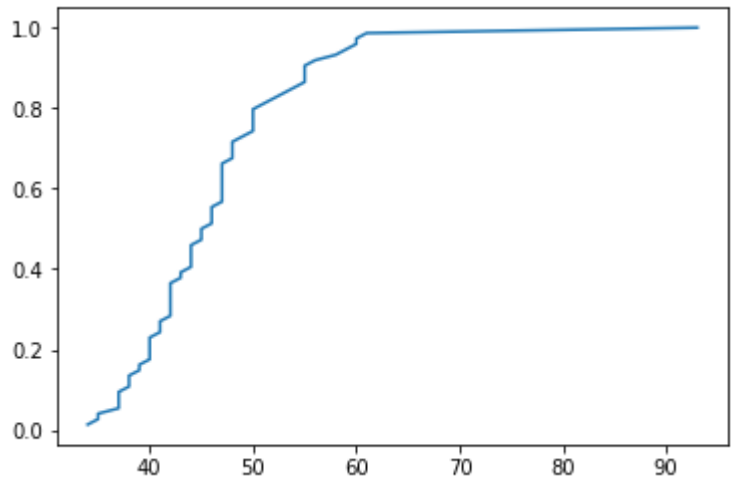
Out[71]:

```
47.0    0.108108
42.0    0.094595
50.0    0.067568
44.0    0.067568
40.0    0.067568
55.0    0.054054
48.0    0.054054
37.0    0.054054
46.0    0.054054
45.0    0.040541
41.0    0.040541
38.0    0.040541
60.0    0.027027
43.0    0.027027
35.0    0.027027
39.0    0.027027
61.0    0.013514
56.0    0.013514
34.0    0.013514
93.0    0.013514
53.0    0.013514
51.0    0.013514
52.0    0.013514
59.0    0.013514
49.0    0.013514
54.0    0.013514
58.0    0.013514
Name: O3, dtype: float64
```

2.9

In [72]:

```
import statsmodels.distributions as dm
dist = dm.ECDF(df.O3.dropna())
plt.plot(dist.x, dist.y)
plt.show()
```



2.10

In [75]:

```
len(df[df['O3'] == 46])
```

Out[75]:

4

2.11

In [81]:

```
mask1 = df['O3'] > 45
mask2 = df['O3'] < 50
len(df[mask1 & mask2]) + len(df[df['O3'] == 45]) + len(df[df['O3'] == 50])
```

Out[81]:

25

2.12

2.13

X = numero dei giorni inun anno nei quali viene superata la soglia per l'ozono

n = 150, $X_1, \ldots, X_n$ campione

In [93]:

```
tn = df['O3_GIORNI_SUPERAMENTO_TOLLERANZA'].mean() #valore atteso
dvn = df['O3_GIORNI_SUPERAMENTO_TOLLERANZA'].std()
print(tn,dvn)
```

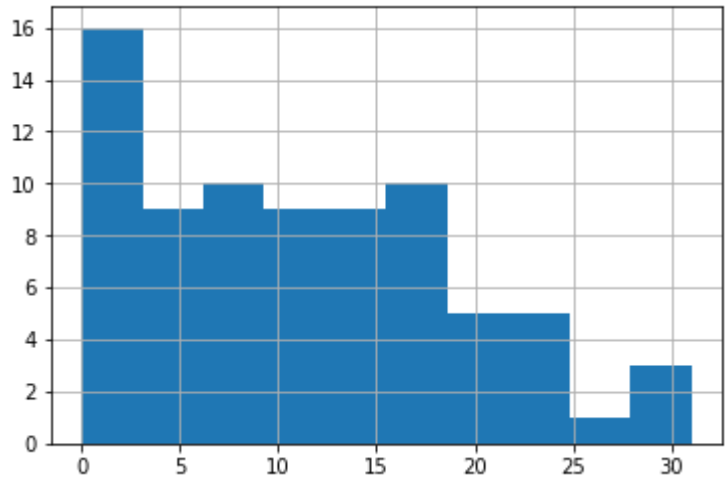
```
11.207792207792208 8.025186769982142
```

2.14

Si potremmo dedurre che segua una legge binomiale per via della definizione: (0 non superamento, 1 superamento). Anche il grafico sembra rispettare.

In [94]:

```
df['O3_GIORNI_SUPERAMENTO_TOLLERANZA'].hist()
plt.show()
```



25 gennaio 2017

Esercizio 0

0.1

- a => Funz cum emp: h, Istogramma: f
- b => Funz cum emp: g, Isogramma: d
- c => Funz cum emp: i, Istogramma e

0.2

$$P(B|A) = \frac{P(A,B)}{P(A)}$$

la probabilità condizionata di un evento A rispetto a un evento B è la probabilità che si verifichi A, sapendo che B è verificato.

0.3

$$T_n = \frac{\sum_{i=1}^{n} X_i}{n}$$

0.4

$$P(|T_n - \mu| < 0.5)$$

$$P(-0.5 < T_n - \mu < 0.5)$$

$$P(-\frac{0.5}{\frac{\sigma}{\sqrt{n}}} < \frac{T_n - \mu}{\frac{\sigma}{\sqrt{n}}} < \frac{0.5}{\frac{\sigma}{\sqrt{n}}})$$

$$P(-\frac{0.5}{\frac{\sigma}{\sqrt{n}}} < Z < \frac{0.5}{\frac{\sigma}{\sqrt{n}}})$$

$$\phi(\frac{0.5}{\frac{\sigma}{\sqrt{n}}}) - \phi(-\frac{0.5}{\frac{\sigma}{\sqrt{n}}})$$

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as st
import math
```

In [2]:

```
Z = st.norm()
n = 47
sigma = 2.5
z1 = 0.5 * math.sqrt(n)/ sigma
z2 = - (0.5 * math.sqrt(n)/ sigma)
Z.cdf(z1) -Z.cdf(z2)
```

Out[2]:

0.8296658522624347

Esercizio 1

In [15]:

```
df = pd.read_csv('dati-ospedali_new.csv', sep=';')
df[:10]
```

Out[15]:

	Odontoiatri_Universitari	Geologi_SSN	Collaboratori_tecnico-profes_Univ	Tecnico-sanitario_Univ	Inge
0	NaN	NaN	0.0	0.0	0.0
1	0.0	NaN	0.0	0.0	0.0
2	NaN	NaN	0.0	0.0	0.0
3	NaN	NaN	0.0	0.0	0.0
4	NaN	NaN	0.0	0.0	0.0
5	NaN	NaN	0.0	0.0	0.0
6	NaN	NaN	0.0	0.0	0.0
7	NaN	NaN	0.0	0.0	0.0
8	NaN	NaN	0.0	0.0	0.0
9	NaN	NaN	0.0	0.0	0.0

10 rows × 83 columns

In [4]:

```
# 1.1
len(df)
```

Out[4]:

146

In [5]:

```
# 1.2
len(df.columns)
```

Out[5]:

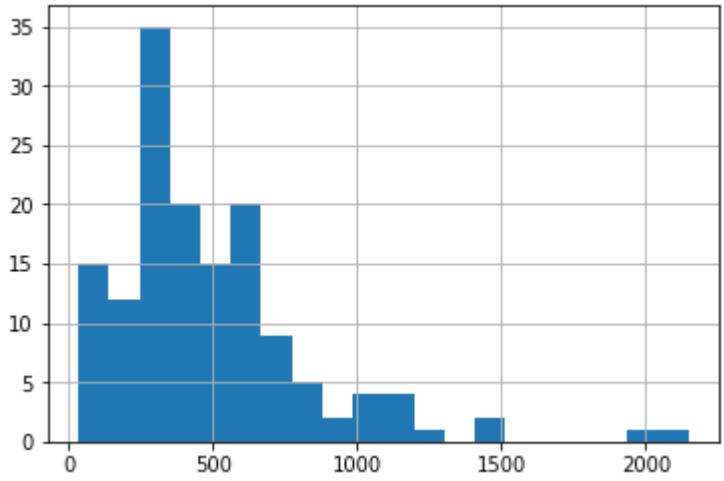
83

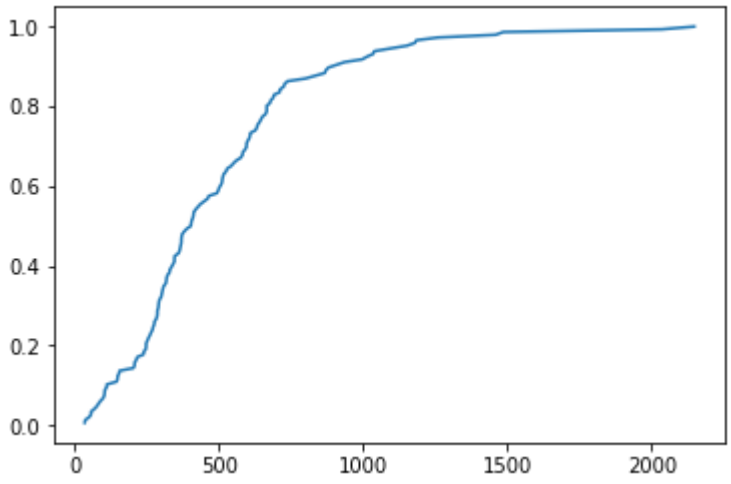
In [6]:

```
# 1.3
medici = df['MediciSSN'].dropna()
# 1.3.1
medici.hist(bins=20)
plt.show()

# medici.plot.box(vert=False)
# plt.show()

from statsmodels.distributions.empirical_distribution import ECDF
dist = ECDF(medici)
plt.plot(dist.x, dist.y)
plt.show()
```





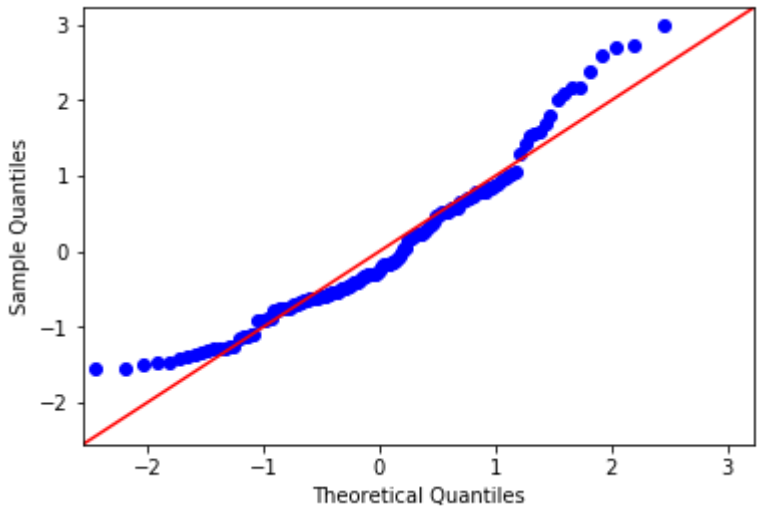
In [7]:

```
# 1.3.2
import statsmodels.api as sm
medici_no_outliers = df.loc[df['MediciSSN'] < 1300,'MediciSSN'].dropna()
sm.qqplot(medici_no_outliers,fit=True, line='45')
(medici_no_outliers.mean(), medici_no_outliers.median())

# dal grafico si può vedere come il parametro segua una legge normale essendo quasi sov
rapposto alla bisettrice del quadrante
# (media e mediana non mi convincono troppo, ma vabbé)
```

Out[7]:

```
(451.40140845070425, 383.5)
```



In [8]:

```
# 1.3.3
(medici.median(), medici.quantile(0.75) - medici.quantile(0.25))

# utilizzo la mediana come indice di centralità perché più robusto rispetto agli outlie
rs presenti nel carattere.
# di conseguenza utilizzo il range interquartile per lo stesso motivo e ben si sposa co
n la mediana

# (non uso la varianza perché deriva dalla media che soffre gli outliers)
```

Out[8]:

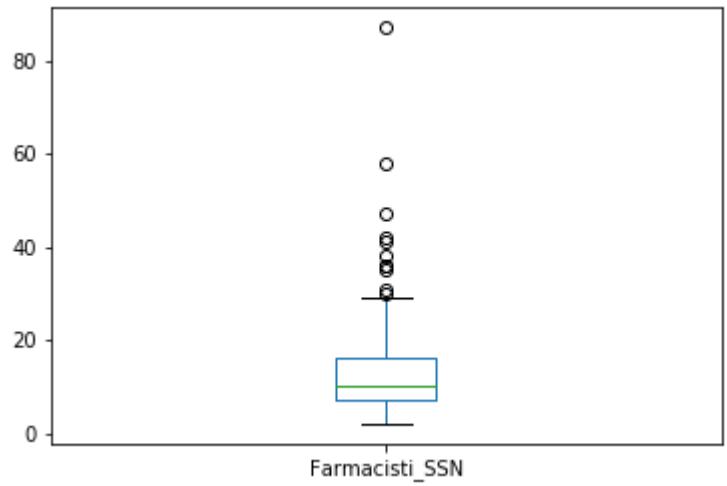
```
(402.5, 357.25)
```

In [9]:

```
# 1.4
farmacisti = df['Farmacisti_SSN']
```

In [10]:

```
# 1.4.1
farmacisti.plot.box()
plt.show()
```



In [11]:

```
# 1.4.2 / 1.4.3
# Coefficiente di variazione (o deviazione standard relativa)
medici_mean = 406.6
medici_dev = 160.7
medici_coeff_var = medici_dev / medici_mean

farmacisti_mean = 6.4
farmacisti_dev = 2.9
farmacisti_coeff_var = farmacisti_dev / farmacisti_mean

(medici_coeff_var, farmacisti_coeff_var)
```

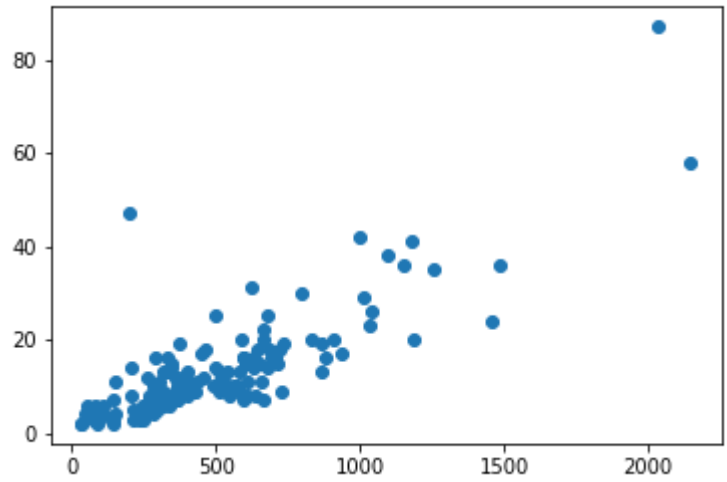
Out[11]:

```
(0.39522872602065906, 0.45312499999999994)
```

In [12]:

```
# 1.4.4
plt.scatter(medici, farmacisti)
plt.show()

medici.corr(farmacisti)
# il grafico mostra una dipendenza lineare tra i due caratteri
# il valore di correlazione molto vicino a 1 conferma la relazione
```



Out[12]:

0.8191729725239217

1.5

0 farmacisti => 2 \ 1 farmacisti => 22 \ 2 farmacisti => 4 \ 3 farmacisti => 1

0 grande struttura => 24 \ 1 grande struttra => 5

(L'esercizio richiedeva di basarsi sulla tabella sulla fotocopia. Nel caso richiedesse di farlo tramite codice bisogna fare così) pd.crosstab(index=df.Avvocati_SSN, columns= df.grandestruttura, margins=True)

Esercizio 2

veri positivi : 15 \ veri negativi : 12 \ falsi positivi : 35 \ falsi negativi : 2

Esercizio 3

In [13]:

```
# 3.1
ingegneri = df['Ingegneri_SSN']
(ingegneri.mean(), ingegneri.std())
```

Out[13]:

```
(4.708333333333333, 4.306240389679412)
```

3.2

$$P(|\bar{X}_n - u| < 0.5) >= 0.85$$

$$P(-0.5 < \bar{X}_n - u < 0.5) >= 0.85$$

$$P(-\frac{0.5}{\frac{\sigma}{\sqrt{n}}} < \frac{\bar{X}_n - u}{\frac{\sigma}{\sqrt{n}}} < \frac{0.5}{\frac{\sigma}{\sqrt{n}}}) >= 0.85$$

$$P(-\frac{0.5}{\frac{\sigma}{\sqrt{n}}} < Z < \frac{0.5}{\frac{\sigma}{\sqrt{n}}}) >= 0.85$$

$$P(Z < \frac{0.5}{\frac{\sigma}{\sqrt{n}}}) - P(Z < -\frac{0.5}{\frac{\sigma}{\sqrt{n}}}) >= 0.85$$

$$\phi(\frac{0.5}{\frac{\sigma}{\sqrt{n}}}) - \phi(-\frac{0.5}{\frac{\sigma}{\sqrt{n}}}) >= 0.85$$

$$\phi(\frac{0.5}{\frac{\sigma}{\sqrt{n}}}) - (1 - \phi(\frac{0.5}{\frac{\sigma}{\sqrt{n}}})) >= 0.85$$

$$2\phi(\frac{0.5}{\frac{\sigma}{\sqrt{n}}}) >= 1.85$$

$$\phi(\frac{0.5}{\frac{\sigma}{\sqrt{n}}}) >= 0.925$$

$$\frac{0.5}{\frac{\sigma}{\sqrt{n}}} >= \phi^{-1}(0.925)$$

$$n >= (\frac{\sigma}{0.5}\phi^{-1}(0.925))^2$$

In [14]:

```
sigma = ingegneri.std()
X = st.norm()
((sigma/0.5)*X.ppf(0.925))**2
```

Out[14]:

```
153.70884494899622
```

Per avere una stima del valore atteso con un errore inferiore a 1 è necessario un campione di almeno 154 elementi. Il nostro dataset ne contiene solo 146

Luglio 2017

Esercizio 0

1

Completare la tabella dei percentili:

- 15 -> decimo percentile
- 55 -> 30esimo percentile
- 200 -> 50esimo percentile
- x -> 70esimo percentile
- y -> 90esimo percentile

x = 200 + (200-55) = 345 y = 200 + (200-15) = 385

2

Grafico

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
```

Esercizio 1

In [2]:

```
#ESERCIZIO 1.1: Importare i dati (tenendo presente che il separatore di decimali per i
 numeri è la virgola
#e i valori sono separati dal carattere ";") e dire quanti casi sono presenti nel datas
et.
dati = pd.read_csv("BibliotecheQuartiere.csv",sep=";",decimal=",")
dati.columns
dati.count()
len(dati)
```

Out[2]:

```
402
```

In [3]:

```
#ESERCIZIO 1.2: Da quale anno a quale anno sono stati raccolti i dati?
dati['Anno'].unique()
print("dal 1996 al 2010")
```

```
dal 1996 al 2010
```

In [4]:

```
#ESERCIZIO 1.3: Quante sono le biblioteche rionali presenti nel dataset?
#Elencarne i nomi.
print(len(dati['Biblioteca'].unique()))
dt = dati['Biblioteca'].drop_duplicates().sort_values()
dt[:10]
```

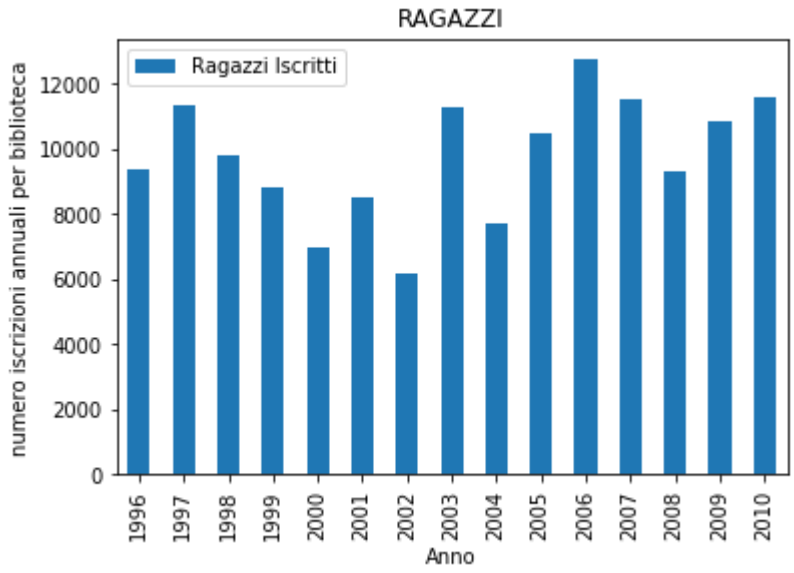
```
73
```

Out[4]:

```
0         Accursio
110     Accursio *
163      Accursio*
1           Affori
244       Affori *
2           Baggio
218       Baggio *
191        Baggio*
3        Bergamini
246    Bergamini**
Name: Biblioteca, dtype: object
```

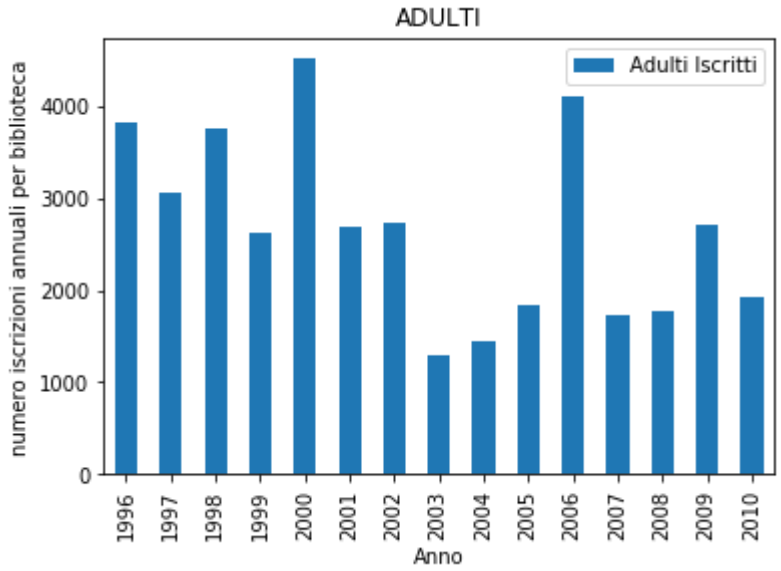
In [5]:

```
#ESERCIZIO 1.4.1: Tracciare il grafico che si ritiene più opportuno per descrivere il n
umero di ragazzi che
#si iscrivono in biblioteca all'anno. Il grafico deve avere il titolo "RAGAZZI" e sul
l'asse opportuno
#(a seconda del grafico che scegliete) deve apparire l'etichetta "numero iscrizioni ann
uali per biblioteca".
ria = dati[['Anno','Ragazzi Iscritti']].dropna().groupby("Anno").sum()
ria.plot.bar()
plt.title("RAGAZZI")
plt.ylabel("numero iscrizioni annuali per biblioteca")
plt.show()
```



In [6]:

```
#ESERCIZIO 1.4.2. Tracciare un grafico analogo che descriva il numero di adulti che si
 iscrivono in biblioteca all'anno.
aia = dati[['Anno','Adulti Iscritti']].dropna().groupby("Anno").sum()
aia.plot.bar()
plt.title("ADULTI")
plt.ylabel("numero iscrizioni annuali per biblioteca")
plt.show()
```



In [7]:

```
#ESERCIZIO 1.5.1. Calcolare la media, la deviazione standard e il coefficiente di varia
zione del numero di ragazzi
#che si iscrivono in biblioteca all'anno.
print(ria.mean())
print(ria.std())
print(ria.std()/ria.mean())
```

```
Ragazzi Iscritti    9759.9224
dtype: float64
Ragazzi Iscritti    1885.754181
dtype: float64
Ragazzi Iscritti    0.193214
dtype: float64
```

In [8]:

```
#ESERCIZIO 1.5.2. Fare lo stesso per il numero di adulti.
print(aia.mean())
print(aia.std())
print(aia.std()/aia.mean())
```

```
Adulti Iscritti    2670.385333
dtype: float64
Adulti Iscritti    1017.060859
dtype: float64
Adulti Iscritti    0.380867
dtype: float64
```

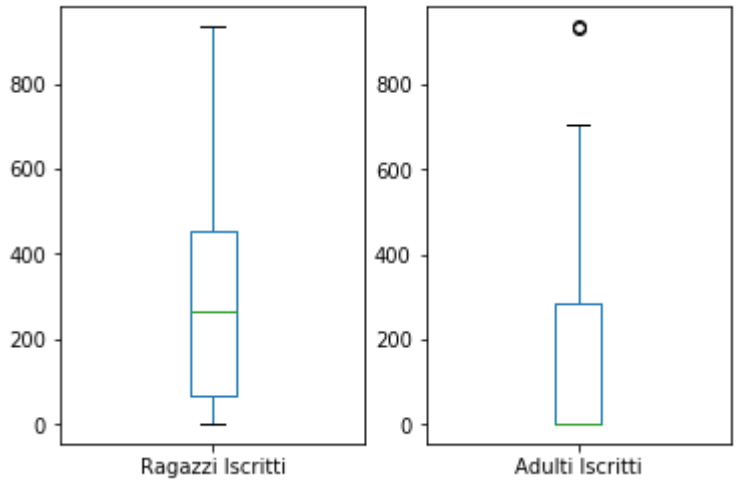
In [9]:

```
#ESERCIZIO 1.5.3 Confrontare la variabilità del numero di iscrizioni di ragazzi rispett
o a quella di adulti
print("Gli adulti iscritti all'anno sono più dispersi rispetto ai ragazzi iscritti")
```

```
Gli adulti iscritti all'anno sono più dispersi rispetto ai ragazzi iscritt
i
```

In [10]:

```
#ESERCIZIO 2:Concentriamoci sull'anno 2000.
#2.1. Tracciare, possibilmente nella stessa figura, il boxplot del numero di ragazzi ch
e si sono iscritti e del numero di
#adulti che si sono iscritti a una biblioteca rionale di Milano (nell'anno 2000).
plt.subplot(1,2,1)
dati[dati['Anno']==2000]['Ragazzi Iscritti'].plot.box()
plt.subplot(1,2,2)
dati[dati['Anno']==2000]['Adulti Iscritti'].plot.box()
plt.show()
```



Esercizio 2

In [11]:

```
#Esercizio 2.2. Utilizzare il risultato del comando summary per rispondere alle seguent
i domande:
#2.1. nell'anno 2000 quale percentuale (circa) di biblioteche ha avuto più di 300 nuovi
ragazzi iscritti?
mask1 = dati["Anno"]==2000
mask2 = dati["Ragazzi Iscritti"]<300
len(dati[mask1 & mask2]["Biblioteca"].get_values())/len(dati["Biblioteca"].unique())
```

Out[11]:

```
0.1917808219178082
```

In [12]:

```
#2.2. nell'anno 2000 quale percentuale (circa) di biblioteche ha avuto più di 950 nuovi iscritti?
mask1 = dati["Anno"]==2000
mask2 = dati["Ragazzi Iscritti"]>300
len(dati[mask1 & mask2]["Biblioteca"].get_values())/len(dati["Biblioteca"].unique())
```

Out[12]:

```
0.1232876712328767
```

Esercizio 3

In [13]:

```
#Esercizio 3.1 Concentriamoci ora sul servizio Bibliobus.
#3.1. Creare una variabile che contiene i soli casi del dataset che si riferiscono alla biblioteca Bibliobus
bb = dati[dati['Biblioteca'] == "Bibliobus"]
```

In [14]:

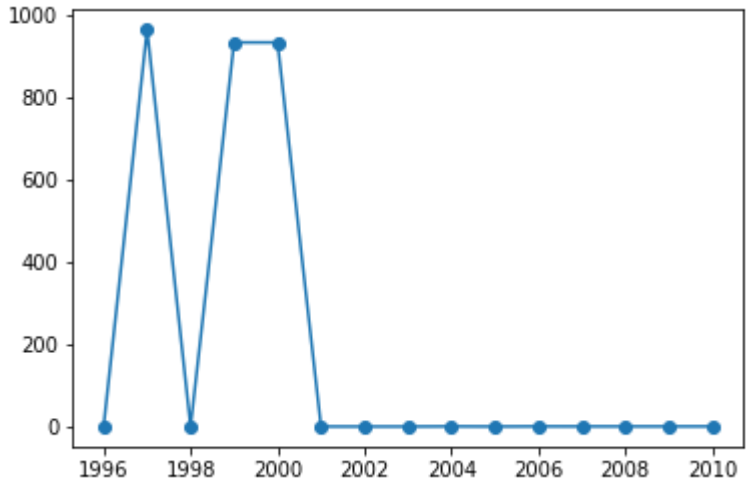
```
#3.2 Calcolare la tabella delle frequenze congiunte tra l'anno e il totale di nuovi iscritti al Bibliobus.
ann_iscritt = pd.crosstab(index=bb["Anno"],columns=bb["Totale Iscritti"])
ann_iscritt.head()
```

Out[14]:

Totale Iscritti	1.03	1.094	1.1	1.117	1.13	1.314	1.34	1.345	1.3780000000000001	1.434000
Anno										
1996	0	0	0	1	0	0	0	0	0	0
1997	0	0	0	0	0	0	0	0	0	0
1998	0	0	0	0	1	0	0	0	0	0
1999	0	0	0	0	0	0	0	0	0	0
2000	0	0	0	0	0	0	0	0	0	0

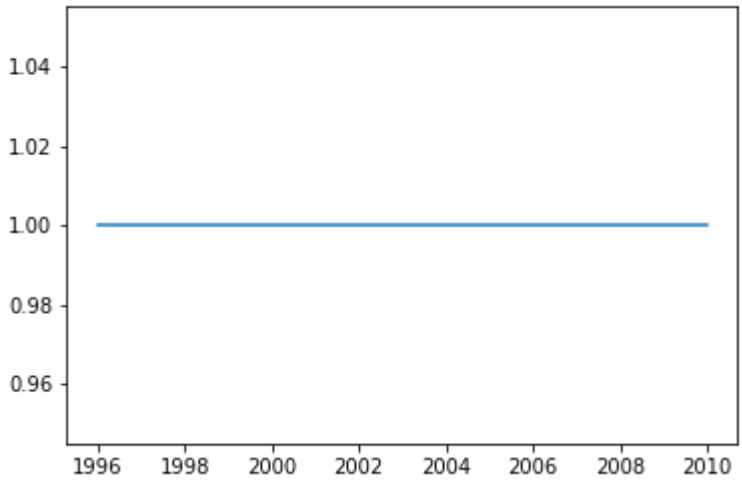
In [15]:

```
#3.3. Tracciare il grafico di dispersione dei caratteri Anno e Totale.Iscritti.
#Siccome in questo caso i dati sono ordinati per anno crescente, rigenerare il grafico
 di dispersione collegando
#ciascun punto al successivo tramite una linea spezzata, al fine di evidenziare una ten
denza.
x = bb["Anno"]
y = bb["Totale Iscritti"]
colors = np.random.rand(100)
plt.scatter(x,y)
plt.plot(x,y) #collegare i puntini!!!!
plt.show()
```



In [16]:

```
#3.4. Commentare, anche avvalendosi di strumenti formali, la seguente affermazione:
#"si può notare che nel corso degli anni c'è stato un incremento, seppur modesto, del n
umero di iscrizioni al
#servizio Bibliobus".
ecdf = sm.distributions.ECDF(bb["Totale Iscritti"])
x = bb["Anno"]
y = ecdf(x)
plt.step(x,y)
plt.show()
print("non cresce negli anni")
```



non cresce negli anni

In [17]:

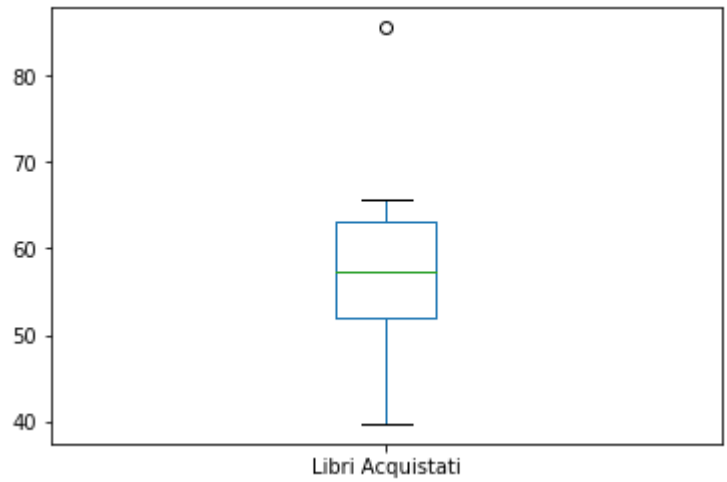
```
#3.5 Prendiamo ora in considerazione le biblioteche Affori e Quarto Oggiaro.
#In Figura 1 sono mostrati nella parte alta i grafici del numero totale di nuovi utenti in ciascun anno e nella parte
#bassa la funzione cumulativa, che indica quindi il totale degli utenti della bibliotec a in ciascun anno.
#I grafici della parte bassa della figura sono in ordine giusto? Cioè ciascuno corrispo nde al grafico delle frequenze
#soprastante? Giustificate la risposta.
print("Si sono nella giusta corrispondenza. Nel grafico di quarto oggiaro possiamo nota re una forte aumento delle frequenze nell'anno 2002 che si riflette nelle ecdf con un a umento dell'altezza dei gradini")
```

```
Si sono nella giusta corrispondenza. Nel grafico di quarto oggiaro possiam
o notare una forte aumento delle frequenze nell'anno 2002 che si riflette
nelle ecdf con un aumento dell'altezza dei gradini
```

Esercizio 4

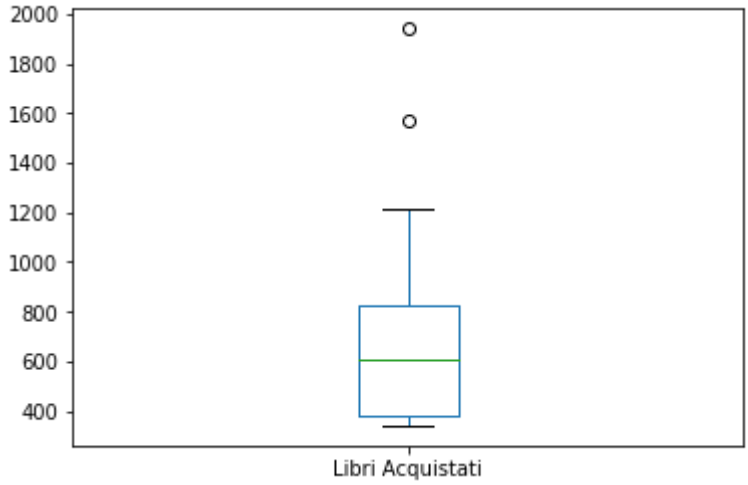
In [18]:

```
#Esercizio 4.1. Tracciare il boxplot oppure l’istogramma (se uno dei due grafici vi sem bra più rappresentativo) del numero di libri acquisiti in un anno da una biblioteca.
sor = dati[dati['Biblioteca'] == 'Biblioteca Centrale Sormani']
sory = sor[['Anno','Libri Acquistati']].groupby('Anno').sum()
sory.plot.box()
plt.show()
```



In [19]:

```
#Esercizio 4.2. I grafici del punto precedente rivelano la presenza di alcuni outlier.
#Questi valori sono tutti relativi alla Biblioteca Centrale Sormani.
#Tracciare il boxplot oppure l'istogramma (se uno dei due grafici vi sembra più rappres
entativo) del numero di libri acquisiti in un anno da una biblioteca, escludendo però l
a Biblioteca Centrale Sormani.
no_sor = dati[dati['Biblioteca'] != 'Biblioteca Centrale Sormani']
no_sory = no_sor[['Anno','Libri Acquistati']].groupby('Anno').sum()
no_sory.plot.box()
plt.show()
```



In [20]:

```
#Esericizio 4.3.
```

In [21]:

```
#Esercizio 4.3.1. Se si esclude la Biblioteca Centrale Sormani si vede che il numero di
libri acquisiti annualmente da una biblioteca ha un andamento "a campana":
# determinare i parametri di tale distribuzione;
```

In [22]:

```
#Esercizio 4.3.2. utilizzare la tecnica del qqplot per controllare se anche il numero d
i libri acquisiti annualmente dalla sola Biblioteca Centrale Sormani segue una legge no
rmale.
```

Esercizio 5

5.1.1

$$Var(\overline{X}) = \frac{1}{n} Var(X)$$
$$\sqrt{(Var(\overline{X}))} = \sqrt{\frac{Var(X)}{n}}$$

5.1.2

$$-\frac{100}{\frac{1}{\sqrt{n}\sigma}} < \frac{\overline{X} - E(X)}{\frac{1}{\sqrt{n}}\sigma} < \frac{100}{\frac{1}{\sqrt{n}\sigma}}$$

$$P(|Z| < \frac{100}{\frac{1}{\sqrt{n}\sigma}}) = 0.99$$

Poichè so che σ=90 ottengo:

$$P(|Z| < \frac{10\sqrt{n}}{9}) = 0.99$$

5.1.3 Per il teorema del limite centrale

$$P(Z < \frac{10\sqrt{n}}{9}) \approx \Phi(\frac{10}{9}\sqrt{n}) - \Phi(-\frac{10}{9}\sqrt{n}) = \Phi(\frac{10}{9}\sqrt{n}) - (1 - \Phi(\frac{10}{9}\sqrt{n})) = 2\Phi(\frac{10}{9}\sqrt{n})) -$$

5.2.1

5.2.2

$$n > \frac{Var(X)}{\sigma\epsilon^2} = \frac{90^2}{0.9 * 100^2} = ?$$

Settembre 2017

Esercizio 0

0.1 Il valore atteso di A - B è uguale a u - u = 0

0.2

$$Var(A-B) = E[(A-B)^2] - E[A-B]^2$$

$$Var(A-B) = E[A^2 - 2AB + B^2]$$

$$Var(A-B) = E[A^2] - 2E[AB] + E[B^2]$$

Perché A e B variabili indipendenti

$$Var(A-B) = E[A^2] - 2E[A]E[B] + E[B^2]$$

$$Var(A-B) = (E[A^2] - u^2) + (E[B^2] - u^2)$$

$$Var(A-B) = Var(A) + Var(B)$$

$$Var(A-B) = 2\sigma^2$$

0.3 La distribuzione di A - B sarebbe a sua volta una variabile aleatoria normale perché somma di variabili aleatorie normali. I parametri sarebbero: valore atteso 0 e varianza 2o^2

0.4

$$T_x = \frac{\sum_{i=1}^{n} (X_i - u)^2}{n}$$

0.5 Lo stimatore è non distorto perché il valore atteso di ogni (Xi -u) è uguale alla varianza di X per definizione:

$$E[(X_i - u)^2] = Var(X)$$

$$E[T_x] = \frac{n * Var(X)}{n} = Var(X)$$

0.6

$$Y = A - B$$

$$T_y = 2T_x$$

0.7

$$E[T_y] = E[2T_x]$$

$$E[T_y] = 2E[T_x]$$

$$E[T_y] = 2\sigma^2$$

Esercizio 1

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as st

df = pd.read_csv('SF_Park_Scores.csv', sep=',', decimal='.')
df[:5]
```

Out[1]:

	ParkID	PSA	Park	FQ	Score	Facility Type	Facility Name	Address	State	Z
0	86	PSA4	Carl Larsen Park	FY05Q3	0.795	Basketball Court	Ocean View Basketball Courts	Capitol & Montana St	CA	9
1	13	PSA4	Junipero Serra Playground	FY05Q3	0.957	Ball Field	Glen ball fields	Diamond & Farnum Street	CA	9
2	9	PSA4	Rolph Nicol Playground	FY05Q3	0.864	Dog Play Area	Douglass dog play area	26th & Douglass Street	CA	9
3	117	PSA2	Alamo Square	FY05Q4	0.857	Restroom	Gilman Bathrooms	Gilman Ave & Griffith	CA	9
4	60	PSA6	Jose Coronado Playground	FY05Q4	0.859	Basketball Court	GGP1 Panhandle Basketball Courts	Stanyan & Great Hwy	CA	9

In [7]:

```
# 1.2
df.describe()
```

Out[7]:

	ParkID	Score	Zipcode	Floor Count	Square Feet	Per
count	5494.000000	5494.000000	4719.000000	1324.000000	4719.000000	4719.(
mean	32991.238260	0.897962	94117.015469	1.205438	26631.239099	548.7
std	150843.356703	0.117428	7.789351	0.555411	63124.930195	793.5
min	1.000000	0.000000	94102.000000	1.000000	213.120658	60.72
25%	55.000000	0.859000	94112.000000	1.000000	1379.550956	169.1
50%	106.000000	0.931000	94116.000000	1.000000	4241.343735	285.1
75%	154.000000	0.976000	94122.000000	1.000000	10192.913376	513.5
max	957226.000000	1.000000	94134.000000	4.000000	515443.479217	5506.

In [8]:

```
# 1.3
len(df)
```

Out[8]:

```
5494
```

In [9]:

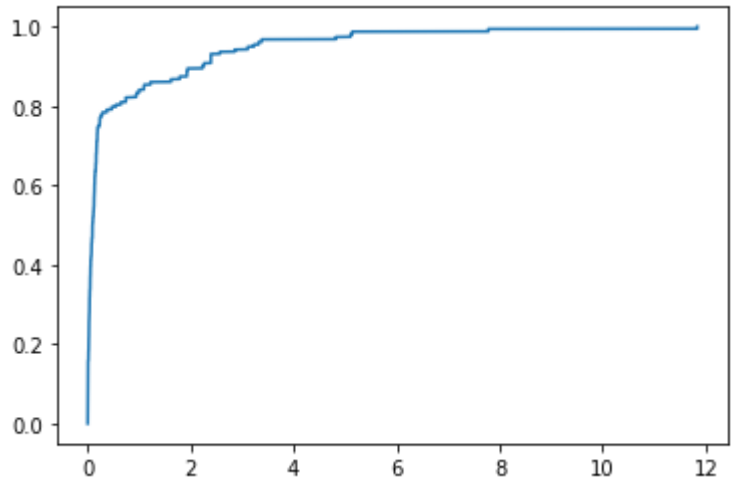
```
# 1.4
(df['PSA'].unique(), len(df['PSA'].unique()))
```

Out[9]:

```
(array(['PSA4', 'PSA2', 'PSA6', 'PSA3', 'GGP', 'PSA1', 'PSA5'],
       dtype=object), 7)
```

In [10]:

```
# 1.5
from statsmodels.distributions.empirical_distribution import ECDF
dist = ECDF(df.Acres.dropna())
plt.plot(dist.x, dist.y)
plt.show()
```



In [11]:

```
# 1.6
    # La metà dei parchi di San Frascisco ha una estensione maggiore di 0.097368 acri
```

In [12]:

```
# 1.7
    # estensione media: 0.611372 acri
```

In [13]:

```
# 1.8
    # media e mediana non sono simili, quindi probabilmente la distribuzione non seguir
à quindi una legge normale.
    # il grafico sarà quindi asimmetrico con una coda a destra
```

In [14]:

```
# 1.9
len(df[df.Acres < 50].dropna())
```

Out[14]:

1324

Esercizio 2

In [15]:

```
# 2.1
# -122.442014
# 37.755449
# 0.032165^2
# 0.025242^2
```

In [16]:

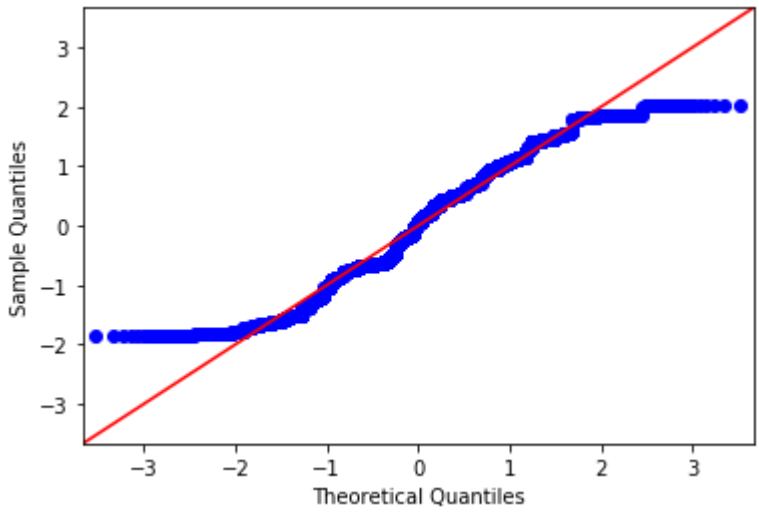
```
# 2.2
#!?
```

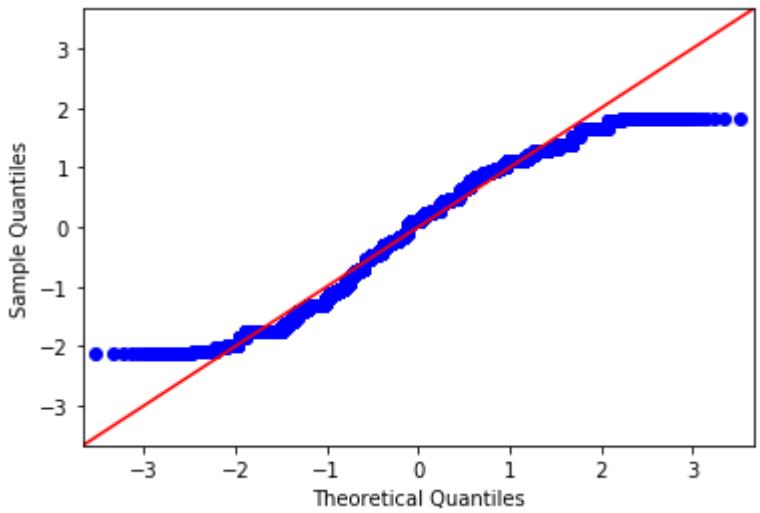
In [17]:

```
# 2.3
import statsmodels.api as sm
sm.qqplot(df.Latitude.dropna(), fit=True, line='45')
plt.show()

sm.qqplot(df.Longitude.dropna(), fit=True, line='45')
plt.show()

# I grafici confermano una distribuzione normale dei due caratteri.
# L'ipotesi è oltretutto confermata dai valori di media e mediana
```



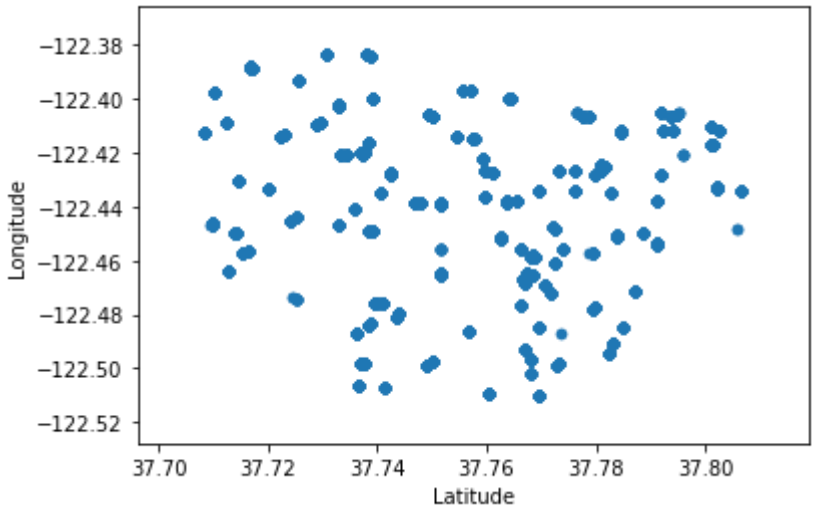


In [18]:

```
# 2.4
# Segue una legge normale perché summa di due variabili aleatorie normali.
# Con valore atteso 0 e varianza 2o^2 (non convintissimo)
```

In [19]:

```
# 2.5
df.plot.scatter('Latitude', 'Longitude')
plt.show()
print(df.Latitude.corr(df.Longitude))
```



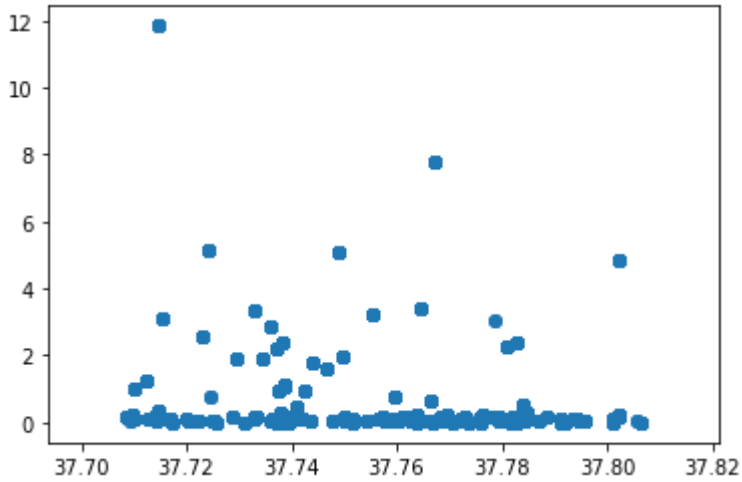

```
-0.07568466249043722
```

In [20]:

```
# 2.6
    # Sia dal grafico molto sparso che dall'indice di correlazione molto vicino a 0 è c
hiaro vedere come non ci sia alcuna
    # dipendenza tra i due caratteri.
```

In [21]:

```
# 2.7
plt.scatter(df.Latitude, df.Acres)
plt.show()
```



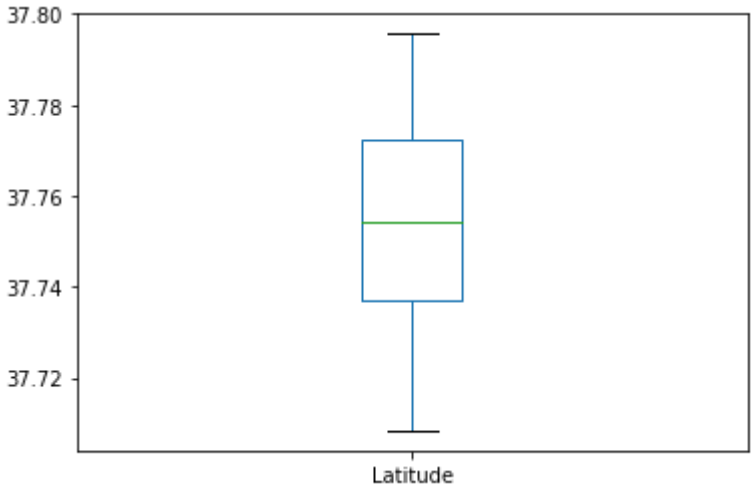
In [22]:

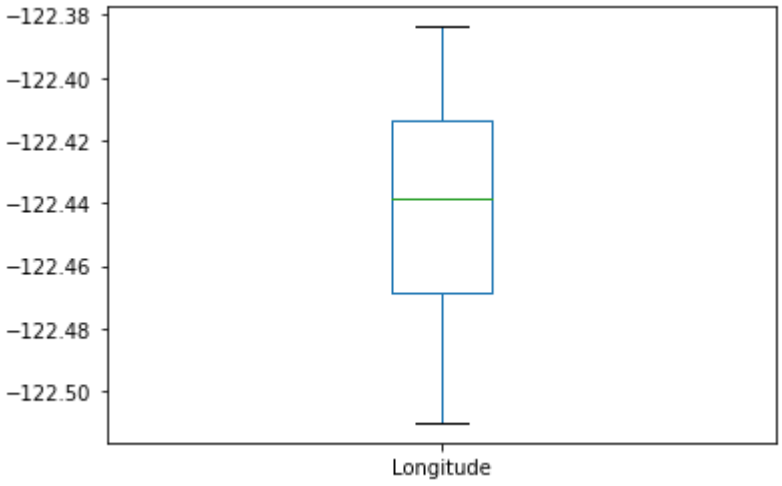
```
# 2.8
parchi = df[(df.Acres < 5) & (df.Latitude < 37.8)]
```

In [23]:

```
# 2.9
parchi['Latitude'].plot.box()
plt.show()

parchi['Longitude'].plot.box()
plt.show()
```





Esercizio 3

3.1

$$P(A - B < 0.1)$$

Visto che la Longitudine ha una distribuzione normale, standardizzo

$$P(\frac{A - B - u}{std(A - B)} < \frac{0.1 - u}{std(A - B)})$$

$$P(Z < \frac{0.1 - u}{std(A - B)})$$

In [27]:

```
import math
## per esercizio 0 la var(A - B) = 2var(X)
devstd = math.sqrt(df.Longitude.std()**2 * 2)
x = (0.1)/devstd
norm = st.norm()
norm.cdf(x)
# Potrebbe essere sbagliato
```

Out[27]:

```
0.9860392741005208
```

3.2

$$P(|A - B| < 0.1)$$

$$P(-0.1 < A - B < 0.1)$$

Visto che la Longitudine ha una distribuzione normale, standardizzo

$$P(-\frac{0.1 - u}{std(A - B)} < \frac{A - B - u}{std(A - B)} < \frac{0.1 - u}{std(A - B)})$$

$$P(Z < \frac{0.1 - u}{std(A - B)}) - P(Z < -\frac{0.1 - u}{std(A - B)})$$

In [28]:

```
devstd = math.sqrt(df.Longitude.std()**2 * 2)
x1 = (0.1)/devstd
x2 = - (0.1)/devstd
norm = st.norm()
norm.cdf(x1) - norm.cdf(x2)
```

Out[28]:

```
0.9720785482010417
```

Gennaio 2018

Esercizio 0

0.1

La media campionaria è uno stimatore non distorto per μ? $E(X) = \mu$

$$E(\overline{X}) = \frac{1}{n}\sum_{i=1}^{n} E(X_i) = \frac{n}{n}E(X) = E(X)$$

0.2

Grafico delle funzioni di ripartizione?

0.3

$X \sim Expon(\lambda)$

$E(X) = \frac{1}{\lambda}$ quindi

$$\lambda = \frac{1}{E(X)}$$

0.4

$Var(X) = \frac{1}{\lambda^2} = E(X)^2$

Deviazione standard = $\sqrt{Var(X)} = E(X)$

0.5

L'esponenziale B ha il parametro con valore più alto perchè per p più alto F_X è maggiore (vedi punto 2)

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import scipy.stats as st
import numpy as np
import statsmodels.api as sm
from scipy.stats import expon
from scipy.stats import geom
```

Esercizio 1

In [2]:

```
dati = pd.read_csv("astici.csv",delimiter=",",decimal=".")
dati.columns
```

Out[2]:

```
Index(['kg.di.pesce', 'settore.di.pesca', 'forza.del.mare', 'peso.astic
e'], dtype='object')
```

1.1

In [3]:

```
len(dati)
```

Out[3]:

```
281
```

1.2

In [4]:

```
len(dati['settore.di.pesca'].unique())
```

Out[4]:

```
9
```

1.3

In [5]:

```
len(dati[dati['settore.di.pesca'] == 'A'])
```

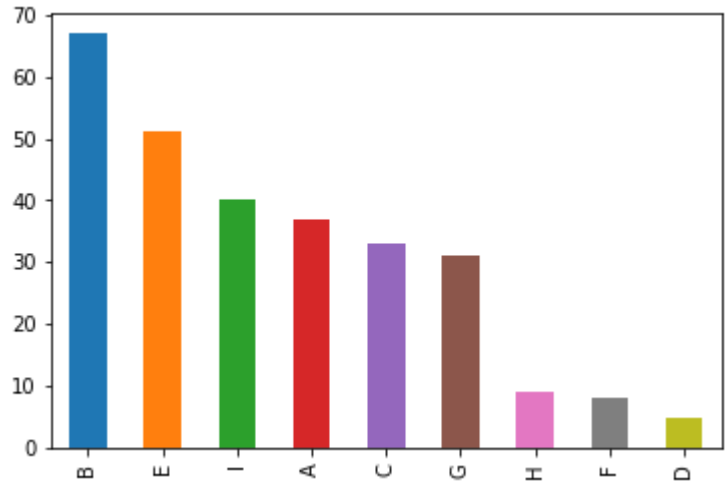
Out[5]:

```
37
```

1.4

In [6]:

```
dati['settore.di.pesca'].value_counts().plot.bar()
plt.show()
```



1.5

Settore B

1.6

In [7]:

```
p = len(dati[dati['settore.di.pesca'] == "B"])/len(dati)
p * 100
```

Out[7]:

```
23.843416370106763
```

1.7

In [8]:

```
dati['forza.del.mare'].head()
#CATEGORICO
```

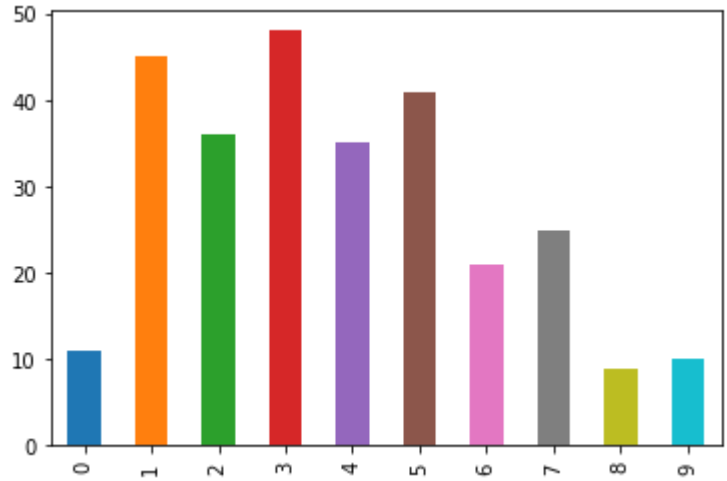
Out[8]:

```
0    9
1    9
2    9
3    7
4    8
Name: forza.del.mare, dtype: int64
```

1.8

In [9]:

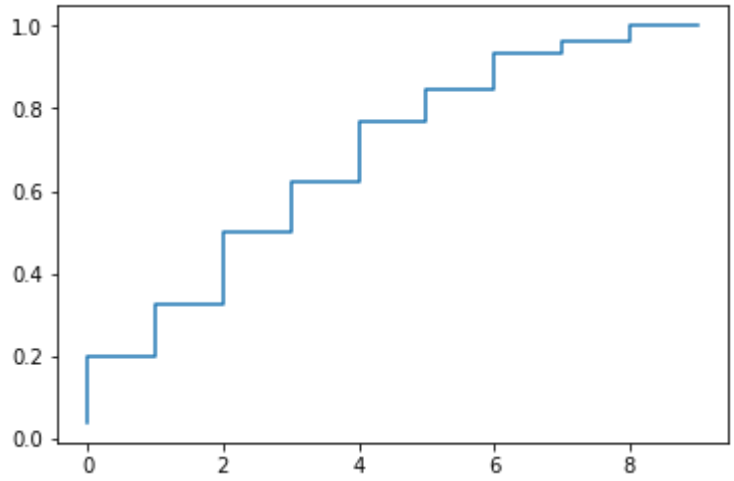
```
dati['forza.del.mare'].value_counts().sort_index().plot.bar()
plt.show()
```



1.9

In [10]:

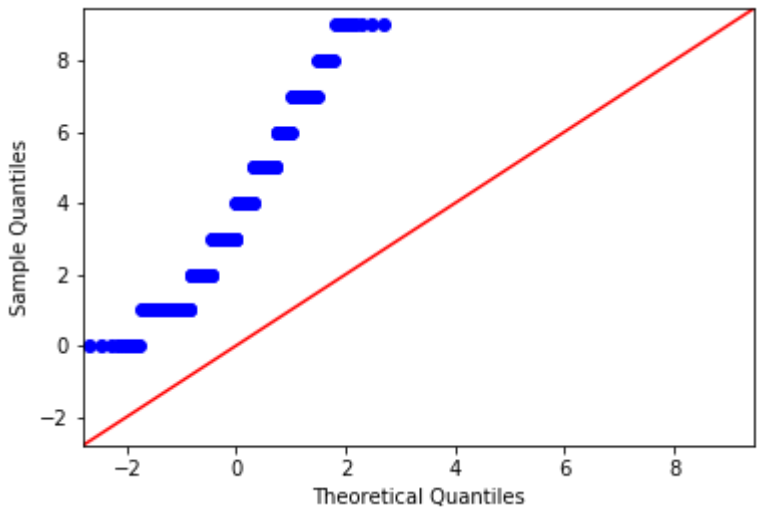
```
#1.9
ecdf = sm.distributions.ECDF(dati['forza.del.mare'])
x = np.arange(dati['forza.del.mare'].min(), dati['forza.del.mare'].max()+1)
y = ecdf(x)
plt.step(x,y)
plt.show()
```



1.10

In [11]:

```
sm.qqplot(dati['forza.del.mare'],line="45")
plt.show()
```



Il Q-Q Plot è la rappresentazione grafica dei quantili di una distribuzione. Confronta la distribuzione cumulata della variabile osservata con la distribuzione cumulata della normale. Se la variabile osservata presenta una distribuzione normale, i punti di questa distribuzione congiunta si addensano sulla diagonale che va dal basso verso l'alto e da sinistra verso destra. In questo caso i punti non si distribuiscono su questa diagonale e quindi possiamo affermare che non è approssimativamente normale.

1.11

In [12]:

```
dati['forza.del.mare'].mean()
```

Out[12]:

```
3.804270462633452
```

1.12

In [13]:

```
dati['forza.del.mare'].mode()
```

Out[13]:

```
0    3
dtype: int64
```

Il valore della forza del mare riscontrato più spesso è stato il 3.

1.13

In [14]:

```
dati['peso.astice'].head()
#QUANTITATIVO CONTINUO
```

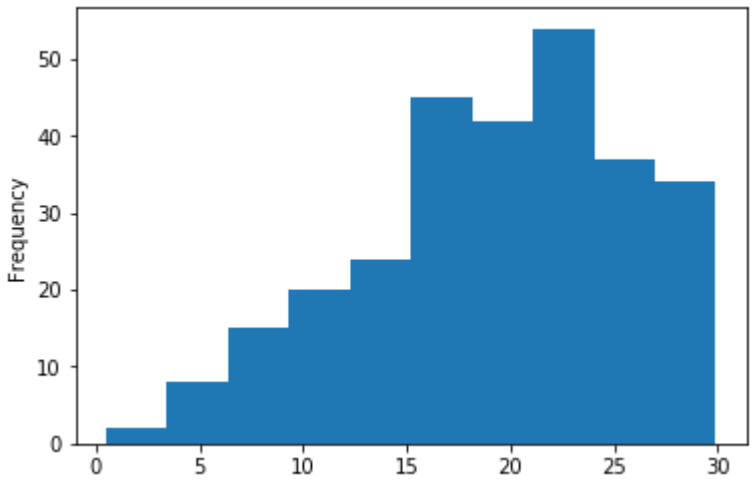
Out[14]:

```
0    29.9
1    29.3
2    29.9
3    29.9
4    28.4
Name: peso.astice, dtype: float64
```

1.14

In [15]:

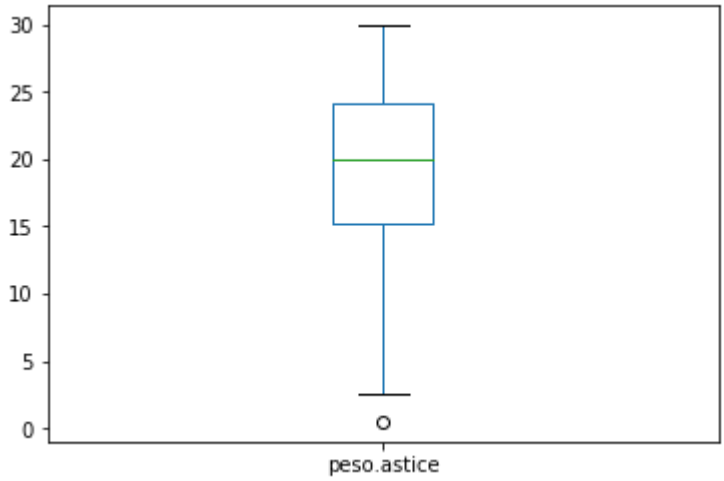
```
#1.14 QUANTITATIVO CONTINUO -> HIST?
dati['peso.astice'].plot.hist()
plt.show()
```



1.15

In [16]:

```
#EVIDENZIARE GLI OUTLIER
dati['peso.astice'].plot.box()
plt.show()
```



1.16

In [17]:

```
astici_filtrato = dati[dati['peso.astice'] != dati['peso.astice'].min()]
```

1.17

In [18]:

```
print(astici_filtrato['peso.astice'].var(),astici_filtrato['peso.astice'].mean())
```

```
40.49094930875574 19.354285714285737
```

1.18

$$P(|X - E(X)| < 10)$$
$$2\Phi(\frac{10\sqrt{n}}{\sigma}) - 1$$

In [19]:

```
import math
n = len(astici_filtrato['peso.astice'])
X = st.norm()
sigma = astici_filtrato['peso.astice'].std()
pi = (10*math.sqrt(n))/sigma
2*X.cdf(pi)-1
```

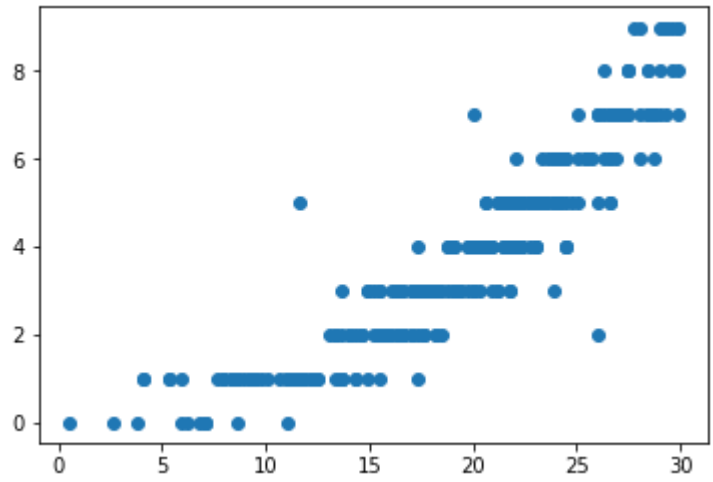
Out[19]:

```
1.0
```

1.19

In [20]:

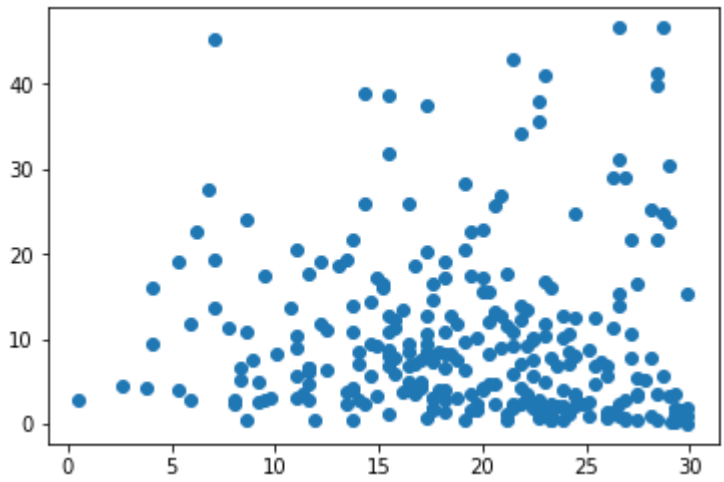
```
#RELAZIONE TRA peso.astice e forza.del.mare
plt.scatter(dati['peso.astice'],dati['forza.del.mare'])
plt.show()
```



In questo grafico è evidente una relazione tra i due caratteri. La relazione evidenziata è lineare positiva.

In [21]:

```
#RELAZIONE TRA peso.astice e kg.di.pesce.
plt.scatter(dati['peso.astice'],dati['kg.di.pesce'])
plt.show()
```



In questo grafico invece non si evidenzia nessun tipo di relazione tra i duie caratteri in quanto i puntini sono tutti sparsi.

1.20

In [22]:

```
#INDICE CHE INDICA LA RELAZIONE: INDICE DI CORRELAZIONE
dati["peso.astice"].corr(dati["forza.del.mare"])
```

Out[22]:

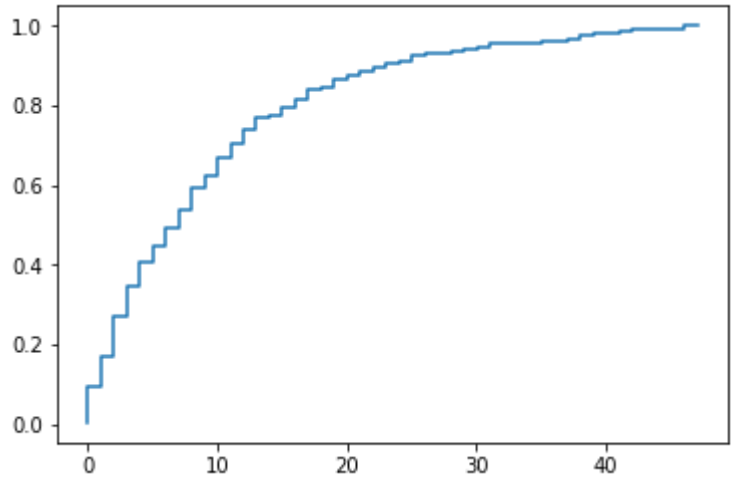
```
0.9156381240896676
```

Esercizio 2

2.1

In [23]:

```
#FUNZIONE CUMULATIVA EMPIRICA
ecdf = sm.distributions.ECDF(dati['kg.di.pesce'])
x = np.arange(dati['kg.di.pesce'].min(), dati['kg.di.pesce'].max()+1)
y = ecdf(x)
plt.step(x,y)
plt.show()
```



2.2

In [24]:

```
dati['kg.di.pesce'].var()/len(dati)
```

Out[24]:

0.346324073964187

In [25]:

```
dati['kg.di.pesce'].mean()
```

Out[25]:

10.078838086708181

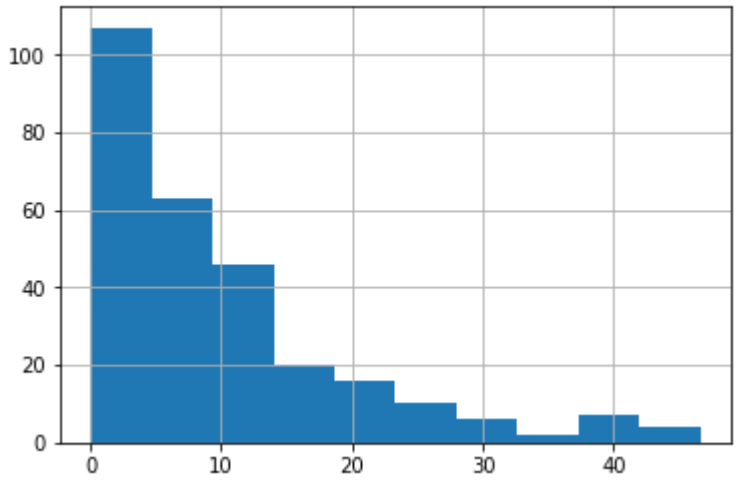
2.3

La media campionaria è sempre uno stimatore non deviato, mentre la varianza bisogna verificare

2.4

In [26]:

```
dati['kg.di.pesce'].hist()
plt.show()
```



Il grafico appena proposto suggerisce che kg.di.pesce potrebbe distribuirsi come un esponenziale in quanto il grafico approssima molto bene quello della distribuzione esponenziale. Poi sappiamo che nell'aponenziale il valore atteso a le deviazione standard si equivalgono e anche in questo caso se stimo i due valori sono molto vicini tra di loro.

In [27]:

```
dati["kg.di.pesce"].describe()
```

Out[27]:

```
count    281.000000
mean      10.078838
std        9.864941
min        0.020205
25%        2.886564
50%        7.182464
75%       13.533704
max       46.633104
Name: kg.di.pesce, dtype: float64
```

2.5

$\lambda = \frac{1}{E(X)}$

In [28]:

```
1/(dati["kg.di.pesce"].mean())
```

Out[28]:

```
0.09921778595875896
```

Febbraio 2018

Esercizio 0

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)}$$

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import statsmodels.api as sm
import scipy.stats as st
```

Esercizio 1

1.1

$$D_X = (0, 1)$$

1.2

$$P(X \leq 0.5) = (1 - p)$$

1.3

$$Var(X) \leq \frac{1}{4}$$

$Var(X) = p - p^2$

Facendo lo studio di funzione notiamo che nel max ho $p = \frac{1}{2}$ quindi nel punto massimo
$Var(X) = \frac{1}{2} - \frac{1}{4} = \frac{1}{4}$

1.4

$\overline{X} = \frac{1}{n} \sum X_i$ non distorto per p poichè la media campionaria è sempre uno stimatore non distorto.

$E[\frac{1}{n} \sum X_i] = \frac{1}{n} \sum E[X_i] = \frac{n}{n} E[X] = E(X) = p$

1.5

$$P(|T_n - p| \leq \epsilon) \geq 1 - \delta$$
$$\delta \geq \frac{1}{4n\epsilon^2}$$

Per Chebyshev

$P(|T_n - p| > \epsilon) \geq \frac{Var(T_n)}{\epsilon^2}$

$P(|T_n - p| < \epsilon) \leq 1 - \frac{Var(T_n)}{\epsilon^2}$

$1 - \frac{Var(X)}{\epsilon^2} = 1 - \frac{(1-p)p}{n\epsilon^2}$ poichè $Var(T_n) = \frac{Var(X)}{n}$

Nel valore max abbiamo dimostrato che $Var(X) = \frac{1}{4}$ quindi $= 1 - \frac{1}{4n\epsilon^2}$

Tornando all'espressione iniziale

$P(|T_n - p| < \epsilon) \leq 1 - \frac{1}{4n\epsilon^2} \leq 1 - \delta$ quindi l'espressione è verificata

1.6

Grafico

1.7

$$P(|Y - np| \leq 1.5)$$

Standardizzo: $Y^* = \frac{|Y-np|}{np(1-p)}$

$$\Phi(\frac{1.5}{np(1-p)}) - \Phi(-\frac{1.5}{np(1-p)})$$

$$\Phi(\frac{1.5}{37 * 0.35(0.65)})) = \Phi(8.4175)$$

$Z(np, np)$

$$P(|Z - np| \leq 1.5)$$

$P(-1.5 + np < Z < 1.5 + np)$

$$\Phi(\frac{1.5 + np}{np(1-p)}) - \Phi(\frac{-1.5 + np}{np(1-p)})$$

$$\Phi(\frac{1.5}{37 * 0.35(0.65)})) = \Phi(8.4175)$$

In [2]:

```
Y = st.norm()
n = 47
sigma = 37*0.35*0.65
y1 = 1.5/sigma
Y.cdf(y1)-Y.cdf(-y1)
```

Out[2]:

```
0.1414342302202134
```

In [3]:

```
p=0.35
Z =st.norm()
z1 = (1.5 + (27*0.35))/sigma
Z.cdf(z1)-Z.cdf(-z1)
```

Out[3]:

```
0.8066940654195229
```

Esercizio 2

In [4]:

```
pesca = pd.read_csv('pesca.csv', sep=',', decimal='.')
pesca.columns
```

Out[4]:

```
Index(['giorno.settimana', 'peso.pescato', 'settore.di.pesca', 'settore.nu
m',
       'forza.del.mare', 'tempesta'],
      dtype='object')
```

2.1

In [5]:

```
len(pesca['giorno.settimana'].unique())
#print("le giornate lavorative sono 5")
```

Out[5]:

5

2.2

In [6]:

```
len(pesca['giorno.settimana'])
```

Out[6]:

255

2.3

In [7]:

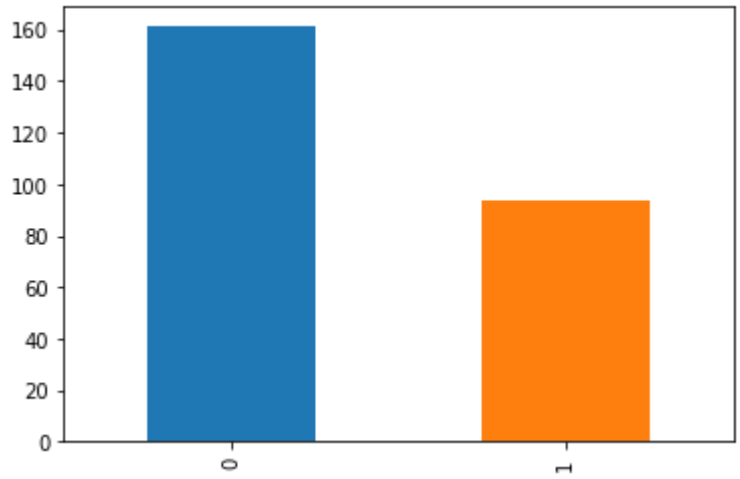
```
print("frequenza assoluta:")
fr = len(pesca[pesca['tempesta']==1])
print(fr)
print("freq relativa")
print(fr/len(pesca))
```

```
frequenza assoluta:
94
freq relativa
0.3686274509803922
```

2.4

In [8]:

```
pesca['tempesta'].value_counts().plot.bar()
plt.show()
```



2.5

In [9]:

```
len(pesca['settore.di.pesca'].unique())
```

Out[9]:

```
9
```

2.6

In [10]:

```
fres = pesca["settore.di.pesca"].value_counts(normalize = True)
mask = fres.index == "A"
fres[mask]
```

Out[10]:

```
A    0.145098
Name: settore.di.pesca, dtype: float64
```

2.7

In [11]:

```
pd.crosstab(pesca["settore.di.pesca"], pesca["tempesta"])
```

Out[11]:

tempesta	0	1
settore.di.pesca		
A	24	13
B	28	24
C	19	14
D	3	2
E	24	16
F	5	3
G	20	11
H	8	1
I	30	10

2.8

In [12]:

```
frea = pd.crosstab(pesca["settore.di.pesca"], pesca["tempesta"])
mask = frea.index == "A"
frea[mask]
```

Out[12]:

tempesta	0	1
settore.di.pesca		
A	24	13

2.9

In [13]:

```
(pesca["tempesta"].mean())
```

Out[13]:

```
0.3686274509803922
```

2.10

In [14]:

```
len(pesca["tempesta"])
```

Out[14]:

```
255
```

2.11

In [15]:

```
setta = pesca[pesca["settore.di.pesca"] == "A"]
setta["tempesta"].mean()
```

Out[15]:

```
0.35135135135135137
```

2.12

$$P(A|T=1) = \frac{P(A \cap T=1)}{P(T=1)}$$

In [16]:

```
setta["tempesta"].mean()/pesca['tempesta'].mean()
```

Out[16]:

```
0.9531339850488787
```

2.13

In [17]:

```
len(setta.dropna())
```

Out[17]:

```
37
```

2.14

n = 37

$$P(|p_{TA} - p| \leq 0.1) > 1 - \frac{Var(X)}{n * (0.1)^2}$$

Per il punto precedente so che n = 37

In [18]:

```
pta = setta["tempesta"].mean()
var = setta["tempesta"].var()
1-(var/(37*(0.1)**2))
```

Out[18]:

```
0.36693450206963685
```

Esercizio 3

3.1

In [19]:

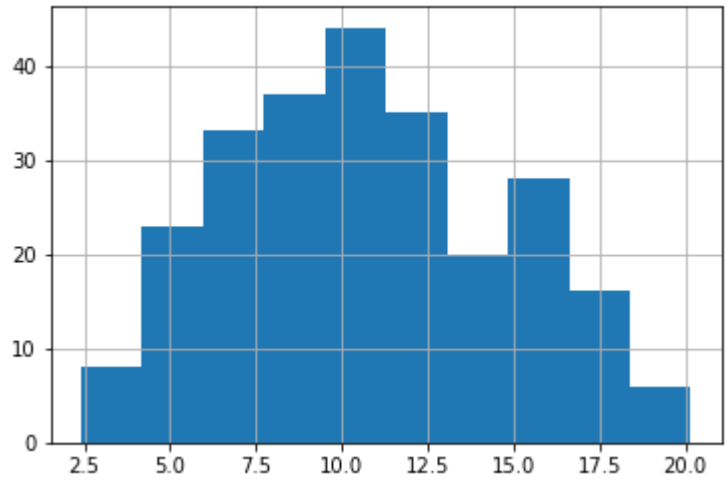
```
pesca['peso.pescato']
print("continuo")
```

```
continuo
```

3.2

In [20]:

```
pesca['peso.pescato'].hist()
plt.show()
```



3.3

In [21]:

```
mask1 = pesca['peso.pescato'] > 10
mask2 = pesca['peso.pescato'] < 15
pp = pesca[mask1 & mask2]
len(pp['giorno.settimana'])/len(pesca['giorno.settimana']) * 100
```

Out[21]:

```
37.64705882352941
```

3.4

In [22]:

```
print(pesca["peso.pescato"].var())
print(pesca["peso.pescato"].mean())
```

```
16.09024449815005
10.788632531936003
```

3.5

In [23]:

```
from scipy.stats import norm
X = norm(loc = 10.78, scale = 4)
z = X.cdf(10)
y = X.cdf(15)
y-z
```

Out[23]:

```
0.431590863816581
```

3.6

In [24]:

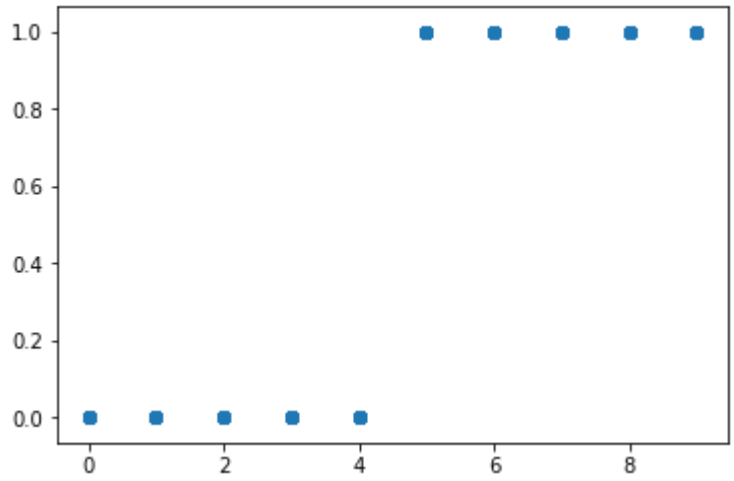
```
print("Si possono confrontare perchè 3.5 è un approssimazione")
```

```
Si possono confrontare perchè 3.5 è un approssimazione
```

3.8

In [25]:

```
plt.scatter(pesca['forza.del.mare'],pesca['tempesta'])
plt.show()
```



3.9

In [26]:

```
print("tra [0,4] non ho tempesta, poi si")
```

```
tra [0,4] non ho tempesta, poi si
```

3.10

No

In [27]:

```
pesca['forza.del.mare'].corr(pesca['tempesta'])
```

Out[27]:

```
0.8412359096077446
```

3.11

In [28]:

```
dati_senza_NA = pesca.dropna()
dati_senza_NA["peso.pescato"].std()/dati_senza_NA["peso.pescato"].mean()
```

Out[28]:

```
0.37180473880354103
```

Giugno 2018

In [1]:

```
import pandas as pd
import numpy as np
import math
import scipy.stats as st
import matplotlib.pyplot as plt
```

Esercizio 0

q_1, q_2, q_3 primo, secondo e terzo quartile

0.1 ¶

Quanto vale $P(X \geq q_2)$? 50\% per definizione di quartile quindi la probabilità è 0,5.

0.2

Quanto vale $P(q_1 \leq X \leq q_3)$? primo e terzo sono 25\% e 75\% quindi 0.5

$X \sim Z(\mu, \sigma^2)$

0.3.1 $P(|X - \mu| \leq \alpha \cdot \sigma) = 0.5$ determinare α

$P(|Z| \leq \alpha) = 0.5$

$2\Phi(\alpha) - 1 = 0.5$

$\Phi(\alpha) = 0.75$

$$\alpha = \Phi^-1(0.75)$$

In [2]:

```
X = st.norm()
# X.cdf(a) = 0.75
X.ppf(0.75)
```

Out[2]:

```
0.6744897501960817
```

0.3.2

q_1, q_2 in funzione dei parametri di X

$P(X < x) = 0.25$ $P(Z < \frac{x-\mu}{\sigma})$

0.3.3

In [3]:

```
mu =1
sigma = 1
Z = st.norm(mu,sigma)
```

0.3.4

$P(|X - \mu| < 2\sigma) = 0.95$

In [4]:

```
X = st.norm()
X.cdf(2) - X.cdf(-2)
```

Out[4]:

```
0.9544997361036416
```

Esercizio 1

In [5]:

```
cani = pd.read_csv('cani.csv', delimiter=";",decimal=",")
cani.columns
```

Out[5]:

```
Index(['Cartella', 'IP', 'GravitaIP', 'EtaAnni', 'MORTE', 'MC', 'SURVIVALT
IME',
       'Terapia', 'Antiaritmico', 'PesoKg', 'VTricuspide', 'AsxAo', 'Onda
E',
       'OndaEA', 'FrazEspuls', 'FrazAccorc', 'EDVI', 'ESVI', 'Allodiast',
       'Allosist'],
      dtype='object')
```

1.1

In [6]:

```
len(cani)
```

Out[6]:

```
161
```

1.2

In [7]:

```
caniIP = cani[cani['IP'] == "SI"]
len(caniIP)
```

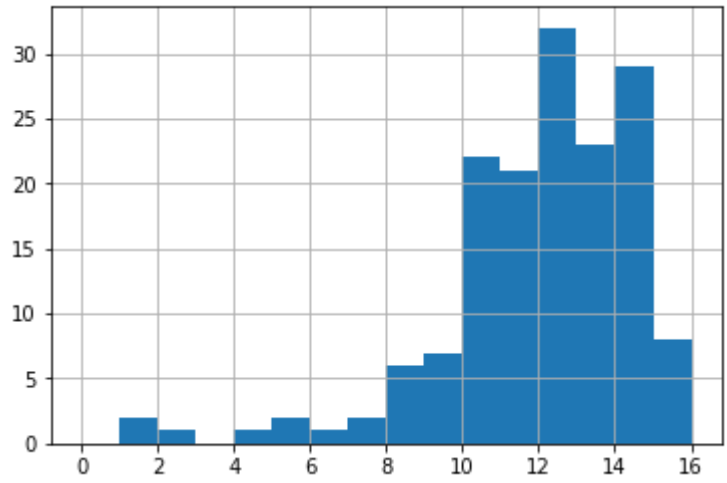
Out[7]:

58

1.3.1

In [8]:

```
età = cani['EtaAnni']
bins = np.arange(0,età.max(),1)
età.hist(bins = np.hstack(bins))
plt.show()
```



1.3.2

In [9]:

```
print(età.min(),età.mean(),età.var(),età.max())
```

1.22 12.124658385093174 6.905200038819876 16.84

1.3.3

In [10]:

```
mask1 = età < 13
mask2 = età >= 12
len(età[ mask1 & mask2])
```

Out[10]:

32

1.3.4

In [11]:

```
età.max()
```

Out[11]:

```
16.84
```

1.3.5

In [12]:

```
età.mode()
```

Out[12]:

```
0    14.25
1    14.73
dtype: float64
```

1.4.1

In [13]:

```
len(cani[cani['MORTE'] == 1])
```

Out[13]:

```
118
```

1.4.2

In [14]:

```
cani[cani.MORTE == 1].MC.isna().value_counts()
```

Out[14]:

```
False    115
True       3
Name: MC, dtype: int64
```

1.4.3

In [15]:

```
mask1 = cani.MORTE == 0
mask2 = cani.MC == 1
len(cani[mask1 & mask2])
```

Out[15]:

```
0
```

1.4.4

In [16]:

```
len(cani[cani.MC == 1])
```

Out[16]:

87

1.4.4

In [17]:

```
morti = cani[cani.MORTE == 1]
len(morti[morti.MC == 1])/len(morti)
```

Out[17]:

0.7372881355932204

1.5.1

In [18]:

```
gip = cani.GravitaIP
print('ordinale')
```

ordinale

1.5.2

In [19]:

```
gip.unique()
```

Out[19]:

array([0, 1, 2, 3], dtype=int64)

1.5.4

In [20]:

```
fgip = pd.crosstab(index=gip,colnames=[''],columns=['Frequenza Relativa'],normalize=True)
fgip
```

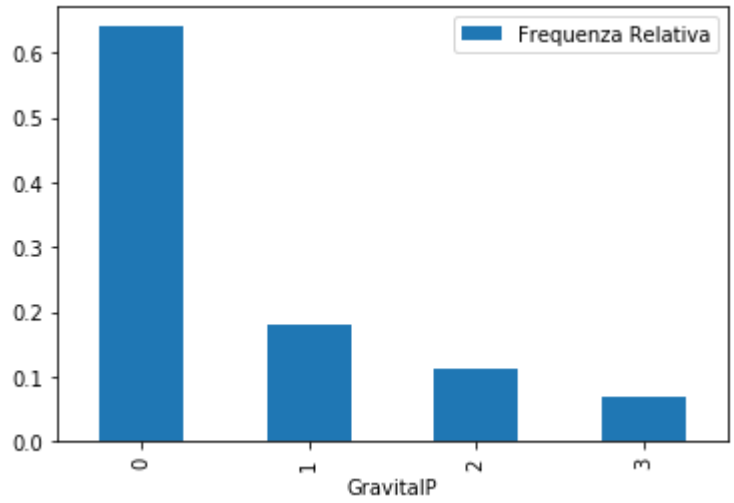
Out[20]:

	Frequenza Relativa
GravitaIP	
0	0.639752
1	0.180124
2	0.111801
3	0.068323

1.5.4

In [21]:

```
fgip.plot.bar()
plt.show()
```



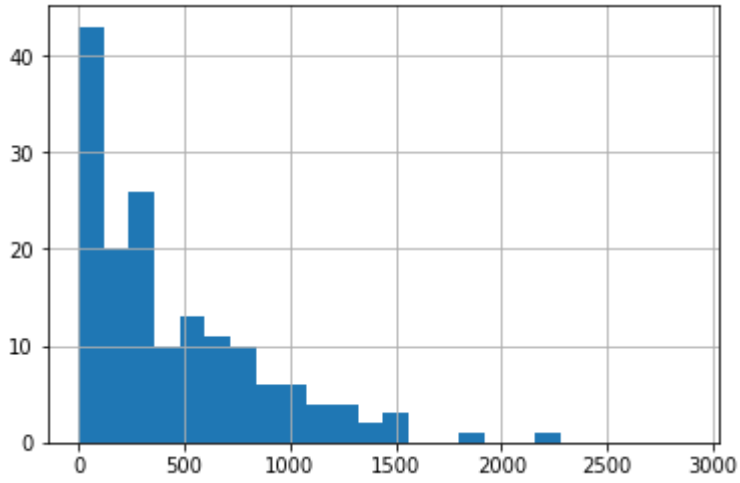
Esercizio 2

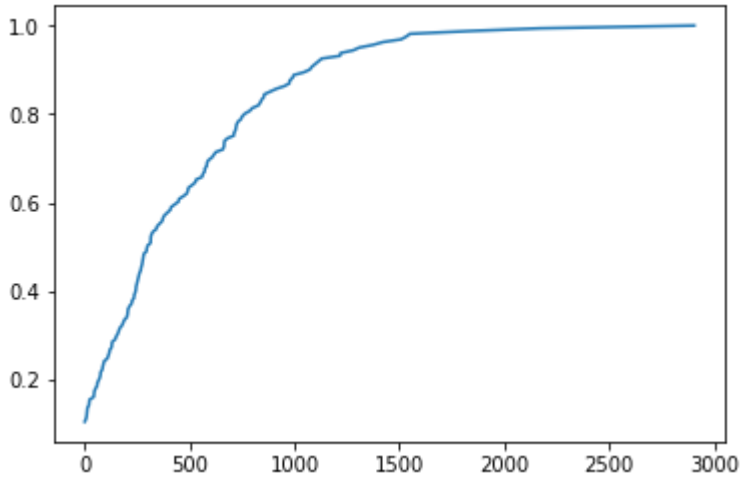
2.1

In [22]:

```
surv = cani.SURVIVALTIME
bins = np.arange(0,surv.max(),120)
surv.hist(bins = np.hstack(bins))
plt.show()

surv.value_counts(normalize=True).sort_index().cumsum().plot()
plt.show()
```





2.2

In [23]:

```
surv.mean()
```

Out[23]:

459.888198757764

2.3

$$T_n = \sum_{i=1}^{n} \frac{X_i}{n}$$

2.4

Non è distorto in quanto la media campionaria, che è lo stimatore utilizzato non è mai distorto rispetto al valore atteso.

$$E(T_n) = E(\frac{1}{n}\sum X_i) = \frac{1}{n}\sum E(X_i) = \frac{1}{n}nE(X) = E(X)$$

2.5

$$Var(T_n) = \frac{Var(X)}{n}$$
$$\sigma(T_n) = \sqrt{\frac{Var(X)}{n}} = \frac{\sigma}{\sqrt{n}}$$

2.6

In [24]:

```
surv.std()
```

Out[24]:

467.1967063479367

2.7

$$P(|T_n - E(X)| < 60) = 0.9$$
$$2\Phi(\frac{60\sqrt{n}}{\sigma}) - 1 = 0.9$$
$$\Phi(\dots) = 0.95$$
$$\sqrt{n} = \frac{\Phi^{-1}(0.95)\sigma}{60}$$

In [25]:

```
X = st.norm()
n = ((X.ppf(0.95)* surv.std())/60)**2
n
```

Out[25]:

164.04067877198327

2.8

Dall'esercizio precedente possiamo affermare che la nostra taglia non è sufficiente.

In [26]:

```
len(surv)
```

Out[26]:

```
161
```

2.9

In [27]:

```
(surv/365).mean()
```

Out[27]:

```
1.2599676678294895
```

2.10

E' non distorto perchè la media campionaria è sempre uno stimatore non distorto

2.11

In [28]:

```
surv.mean()/365
```

Out[28]:

```
1.2599676678294904
```

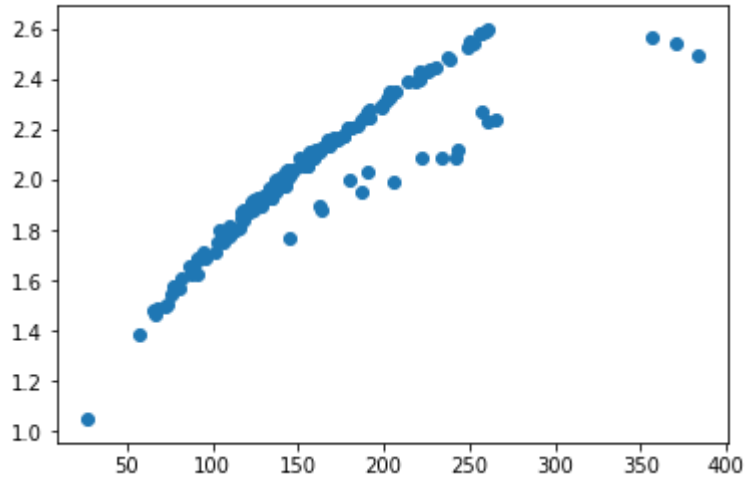
Esercizio 3

3.1

Sono strettamente dipendenti. Vedi il grafico e il valore tendente a 1 dell'indice di correlazione

In [29]:

```
EDVI  = cani.EDVI
Allodiast = cani.Allodiast
plt.scatter(EDVI,Allodiast)
plt.show()
EDVI.corr(Allodiast)
```



Out[29]:

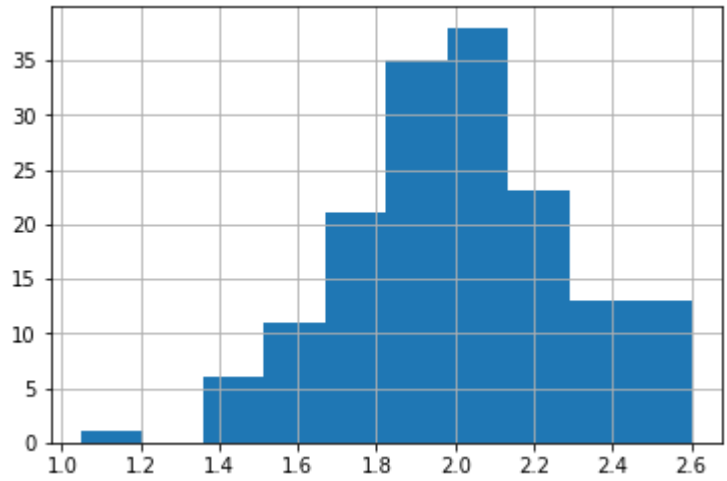
```
0.9073039817753574
```

3.2

Guardando l'istogramma e i valori di media e mediana simili posso affermare che segue una legge normale

In [30]:

```
Allodiast.hist()
plt.show()
```



In [31]:

```
Allodiast.describe()
```

Out[31]:

```
count    161.000000
mean       2.013354
std        0.279596
min        1.050000
25%        1.850000
50%        2.000000
75%        2.180000
max        2.600000
Name: Allodiast, dtype: float64
```

In [32]:

```
### 3.3
```

In [33]:

```
print(Allodiast.mean(),Allodiast.median())
```

```
2.0133540372670815 2.0
```

Luglio 2018

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import statsmodels.api as sm
import scipy.stats as st
```

Esercizio 0

0.1

1B -> 2A normale, 1A -> 2B esponenziale

0.2

Il valore atteso minore è b, poichè è l'area sopra la curva.

0.3

Il valore 2:

- (X) 70esimo p
- (Y) 20p

0.4

cinquantesimo percentile:

- (X,a) 1
- (Y,b) 3

0.5

$P(2 \leq X \leq 5) = F_X(5) - F_X(2) = 1 - 0.8 = 0.2$

$P(2 \leq Y \leq 5) = F_Y(5) - F_Y(2) = 0.8 - 0.2$

0.6

La mediana è minore della media.

Esercizio 1

$X \sim Exp(\nu)$

1) $f_X(x) = \nu e^{-\nu x}$

2) $E(X) = \frac{1}{\nu}$ = deviazione standard

3) $T_n = \overline{X}$, $E(T_n) = \frac{1}{n} E(\sum X_i) = \frac{1}{n} \sum E(X_i) = \frac{1}{n} n \frac{1}{\nu} = \frac{1}{\nu}$

4) Poichè $\frac{1}{\nu}$ è il valore atteso di x allora avremo $\nu = \frac{1}{E(X)}$ quindi

$$\frac{1}{T_n} = R_n$$

Esercizio 2

In [2]:

```
cani = pd.read_csv("cani.csv",delimiter=";",decimal=",")
cani.columns
```

Out[2]:

```
Index(['Cartella', 'IP', 'GravitaIP', 'EtaAnni', 'MORTE', 'MC', 'SURVIVALT
IME',
       'Terapia', 'Antiaritmico', 'PesoKg', 'VTricuspide', 'AsxAo', 'Onda
E',
       'OndaEA', 'FrazEspuls', 'FrazAccorc', 'EDVI', 'ESVI', 'Allodiast',
       'Allosist'],
      dtype='object')
```

2.1.1

In [3]:

```
ar = pd.crosstab(index=cani['Antiaritmico'],columns=["Abs. Freq."],colnames=[''])
ar
```

Out[3]:

	Abs. Freq.
Antiaritmico	
NO	150
SI	11

2.1.2

In [4]:

```
len(cani[cani['Antiaritmico'] == 'SI'])
```

Out[4]:

11

2.1.3

In [5]:

```
print("Si == 1, NO == 0")
```

Si == 1, NO == 0

2.1.4

In [6]:

```
pd.crosstab(index=cani['Antiaritmico'],columns=cani['MC'],colnames=['Cani Morti Cardiac
i'])
```

Out[6]:

Cani Morti Cardiaci	0.0	1.0
Antiaritmico		
NO	28	78
SI	0	9

2.1.5

In [7]:

```
len(cani.MC[cani.Antiaritmico == 'SI'])*100/len(cani.MC)
```

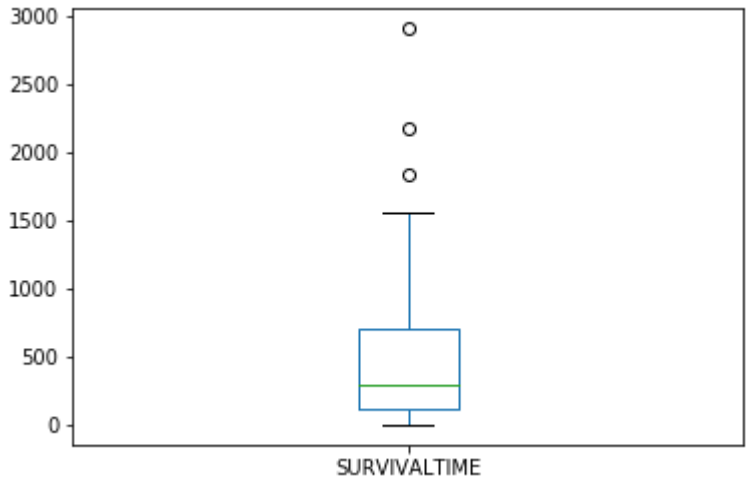
Out[7]:

6.832298136645963

2.2.1

In [8]:

```
cani['SURVIVALTIME'].plot.box()
plt.show()
```



2.2.2

In [9]:

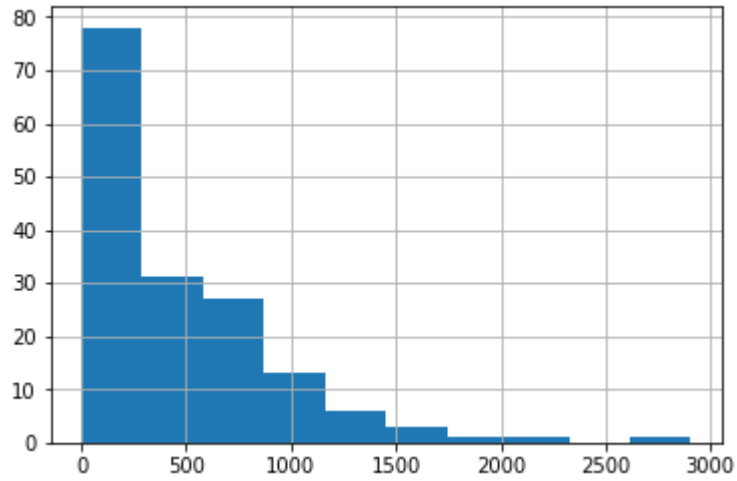
```
cani['SURVIVALTIME'].quantile(0.25),cani['SURVIVALTIME'].quantile(0.75)
canibox= cani[(cani['SURVIVALTIME']>=113)&(cani['SURVIVALTIME']<=711)]
print("I cani rappresentati dal quadrato all'interno del box plot sono : {}".format(len
(canibox)))
```

```
I cani rappresentati dal quadrato all'interno del box plot sono : 81
```

2.2.3

In [10]:

```
cani['SURVIVALTIME'].hist()
plt.show()
```



2.2.4

In [11]:

```
print("Come si può vedere dal grafico si può supporre un modello esponenziale. Inoltre
 solitamente il modello esponenziale viene usato per descrivere il tempo di vita di un
 fenomeno")
```

```
Come si può vedere dal grafico si può supporre un modello esponenziale. In
oltre solitamente il modello esponenziale viene usato per descrivere il te
mpo di vita di un fenomeno
```

2.2.5

In [12]:

```
cani['SURVIVALTIME'].mean()
```

Out[12]:

```
459.888198757764
```

2.2.6

In [13]:

```
cani['SURVIVALTIME'].std()
```

Out[13]:

467.1967063479367

2.2.7

In [14]:

```
print("Esponenziale: 1/valore atteso -> 1/(cani['SURVIVALTIME'].mean() cioè {}".format(
1/(cani['SURVIVALTIME'].mean())))
```

```
Esponenziale: 1/valore atteso -> 1/(cani['SURVIVALTIME'].mean() cioè 0.002
1744415331838686
```

Esercizio 3

In [15]:

```
canimorti = cani[cani['MORTE'] == 1]
canimortinna = canimorti.dropna(axis=0,subset=['MC'])
canimortinna = canimorti.dropna(axis=0,subset=['OndaEA'])
```

3.1

In [16]:

```
#1
canimortinna['OndaEA'].head()
print("Scalare")
```

```
Scalare
```

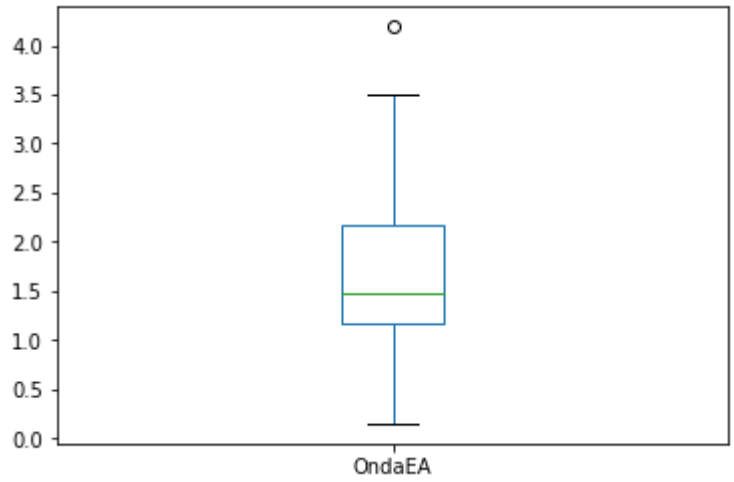
3.2

In [17]:

```
canimortinna['OndaEA'].plot.box()
```

Out[17]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fb747c0828>
```



3.3

In [18]:

```
max(canimortinna['OndaEA'])
```

Out[18]:

```
4.19
```

3.4

In [19]:

```
#4
canimortinna[canimortinna['OndaEA'] == 4.19]['MC']
print("Si è morto per cause cardiache")
```

```
Si è morto per cause cardiache
```

3.5

In [20]:

```
#5
s = canimortinna[canimortinna['MC'] == 0]['OndaEA'].quantile(0.75)
s
```

Out[20]:

1.41

3.6

In [21]:

```
#6
mask1 = canimortinna['MORTE'] == 1
mask2 = canimortinna['MC'] == 0
mask3 = canimortinna['MC'] == 1
print("I cani morti per cause cardiache sono: {}".format(len(canimortinna[mask1 & mask3
])))
print("I cani morti per cause non cardiache sono: {}".format(len(canimortinna[mask1 & m
ask2])))
```

```
I cani morti per cause cardiache sono: 66
I cani morti per cause non cardiache sono: 17
```

3.7

In [22]:

```
#7
mask4 = canimortinna['OndaEA'] >= s
mask5 = canimortinna['OndaEA'] < s
print(">= : {}".format(len(canimortinna[mask1 & mask3 & mask4])))
print("< :{}".format(len(canimortinna[mask1 & mask2 & mask5])))
```

```
>= : 41
< :12
```

In [23]:

```
print('Cani morti per altre cause con valore di OndaEA >=s\nFalso Positivo : {}\nCani m
orti per cause cardiache con valore di OndaEA < s\nFalso Negativo : {}'.format(len(cani
mortinna[mask1 & mask2 & mask4]),len(canimortinna[mask1 & mask3 & mask5])))
```

```
Cani morti per altre cause con valore di OndaEA >=s
Falso Positivo : 5
Cani morti per cause cardiache con valore di OndaEA < s
Falso Negativo : 25
```

3.8

In [24]:

```
print("Sensibilità : {}\nSpecificità : {}".format((41/66),(12/17)))
```

```
Sensibilità : 0.6212121212121212
Specificità : 0.7058823529411765
```

In [2]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import statsmodels.api as sm
import scipy.stats as st
```

Settembre 2018

Esercizio 0

1) $X \sim Bern(p)$ con $p \in (0,1)$

2) Var(X) = p(1-p) quindi $dvstd = \sqrt{p(1-p)}

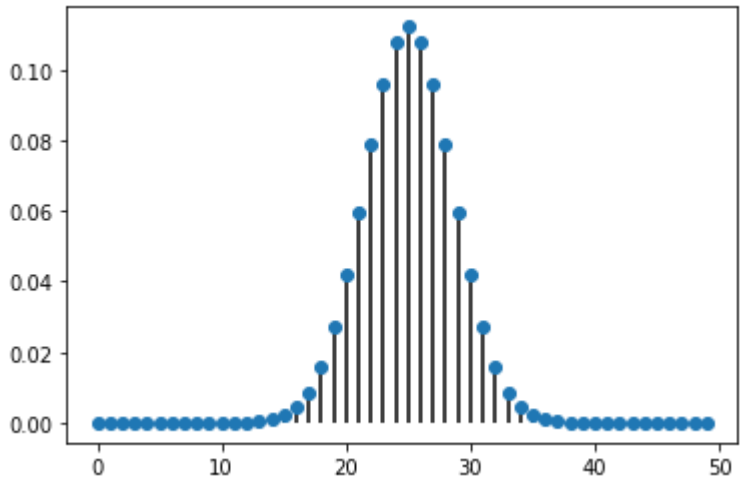
3) $k \in N, Y \sim Bin(k,p)$

$P(X=n) = \binom{k}{n} p^n (1-p)^{k-n}$

4) k = 50, p = 0.5

In [3]:

```
Y=st.binom(50,0.5)
x=np.arange(0,50)
plt.vlines(x,0,Y.pmf(x))
plt.plot(x,Y.pmf(x),'o')
plt.show()
```



5) Definizione Stimatore

6) $E(X) = p, T_n = \overline{X}$

$E(T_n) = E(\sum \frac{X_i}{n}) = \frac{1}{n} E(\sum X_i) = \frac{1}{n} \sum E(X_i) = \frac{n}{n} E(X) = p$

7) $n \geq 1$

Dimostrare che $P(-\epsilon < T_n - p < \epsilon) \approx 2\Phi(\epsilon \frac{\sqrt{n}}{\sigma}) - 1$

Standardizzo: $P(|T_n - p| < \epsilon)$ lo divido per la deviazione standard di T_n cioè $\sqrt{\frac{1}{n}\sigma^2}$ e ottengo

$$P(|Z| < \frac{\epsilon\sqrt{n}}{\sigma})$$

Applico il teorema del limite centrale:

$$P(|Z| < \frac{\epsilon\sqrt{n}}{\sigma}) \approx \Phi(\frac{\epsilon\sqrt{n}}{\sigma}) - \Phi(-\frac{\epsilon\sqrt{n}}{\sigma}) = \Phi(\frac{\epsilon\sqrt{n}}{\sigma}) - (1 - \Phi(\frac{\epsilon\sqrt{n}}{\sigma})) = 2\Phi(\frac{\epsilon\sqrt{n}}{\sigma}) - 1$$

Esercizio 1

In [4]:

```
fin = pd.read_csv("finanziamenti.csv",delimiter=";",decimal=",")
fin.columns
```

Out[4]:

```
Index(['id', 'TemaPrioritario', 'FONTE', 'CodiceCategoria', 'CATEGORIA',
       'UNITA', 'FinProvincia', 'FinRegione', 'TotSpese'],
      dtype='object')
```

In [5]:

```
#1
fin['CodiceCategoria']
print("Valore  nominale in quanto indica un codice")
```

```
Valore  nominale in quanto indica un codice
```

In [6]:

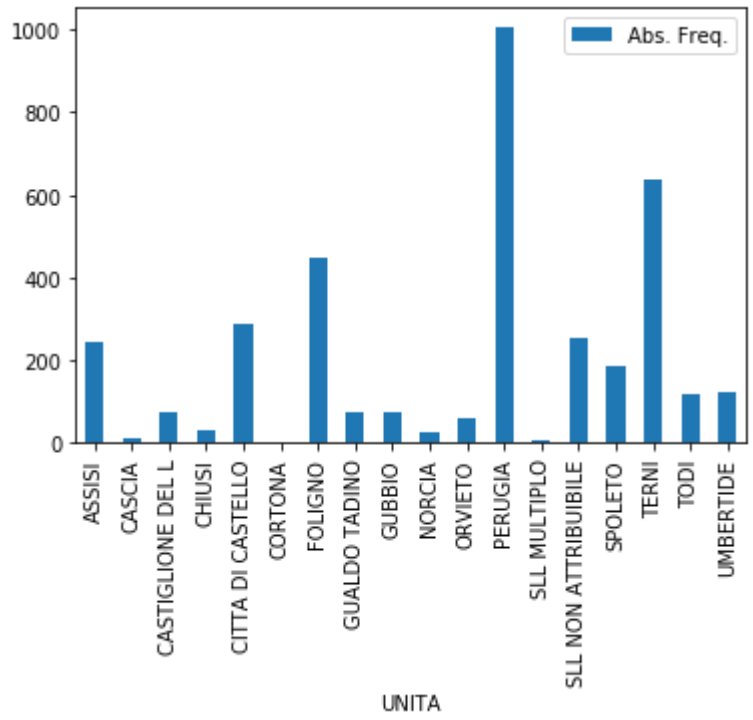
```
#2
finu = pd.crosstab(index=fin['UNITA'],columns=['Abs. Freq.'],colnames=[''])
finu
```

Out[6]:

	Abs. Freq.
UNITA	
ASSISI	243
CASCIA	13
CASTIGLIONE DEL L	75
CHIUSI	30
CITTA DI CASTELLO	288
CORTONA	1
FOLIGNO	449
GUALDO TADINO	75
GUBBIO	76
NORCIA	28
ORVIETO	60
PERUGIA	1005
SLL MULTIPLO	5
SLL NON ATTRIBUIBILE	255
SPOLETO	186
TERNI	638
TODI	117
UMBERTIDE	124

In [10]:

```
#3
finu.plot.bar()
plt.show()
```



In [25]:

```
#4  vedi foglio
```

In [27]:

```
#5
progetti_a=fin[fin['FinProvincia']<(fin['FinRegione'])]
progetti_b=fin[fin['FinProvincia']>=(fin['FinRegione'])]
```

In [34]:

```
print("A: {}".format(len(progetti_a)))
print("B: {}".format(len(progetti_b)))
```

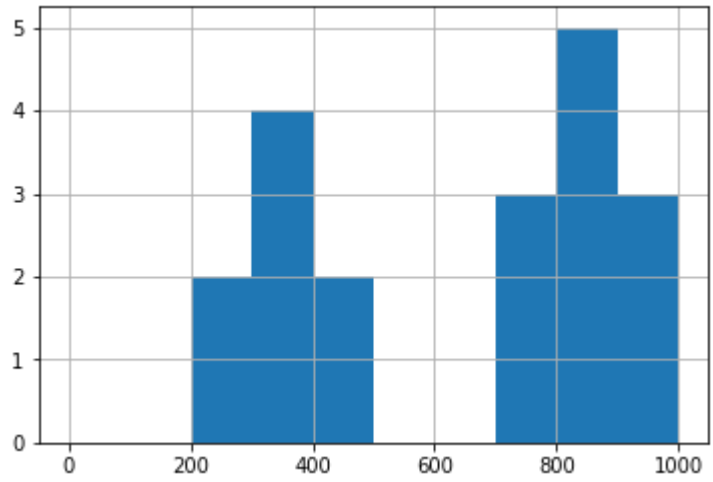
```
A: 368
B: 3284
```

In [36]:

```
#6.1
mask = progetti_a['FinProvincia'] >= 200
mask1 = progetti_a['FinProvincia'] < 1000
selezione_progetti_a = progetti_a[mask & mask1]
```

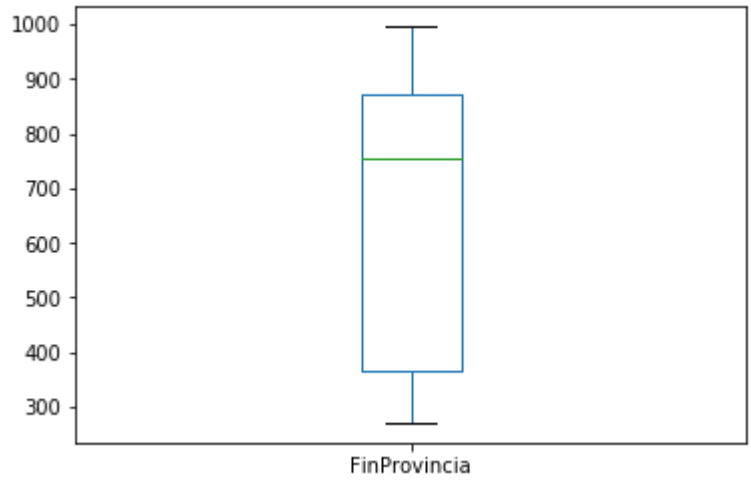
In [39]:

```
#6.2
bins=np.arange(0,1001,100)
selezione_progetti_a['FinProvincia'].hist(bins = bins)
plt.show()
```



In [40]:

```
#6.3
selezione_progetti_a['FinProvincia'].plot.box()
plt.show()
```



In [41]:

```
#6.4
print("Tra i due grafici ritengo che l'istogramma sia più informativo in quanto, nonost
ante nel box plot possiamo trarre un sacco di informazioni non rileviamo la più importa
nte : il carattere è suddiviso in due gruppi che sembrerebbero seguire una distribuzion
e normale 'BIMODALE'")
```

```
Tra i due grafici ritengo che l'istogramma sia più informativo in quanto,
nonostante nel box plot possiamo trarre un sacco di informazioni non rilev
iamo la più importante : il carattere è suddiviso in due gruppi che sembre
rebbero seguire una distribuzione normale 'BIMODALE'
```

In [42]:

```
#6.5
selezione_progetti_a['FinProvincia'].mean()
```

Out[42]:

```
636.9052631578948
```

In [43]:

```
selezione_progetti_a['FinProvincia'].std()
```

Out[43]:

```
264.80233322588253
```

In [47]:

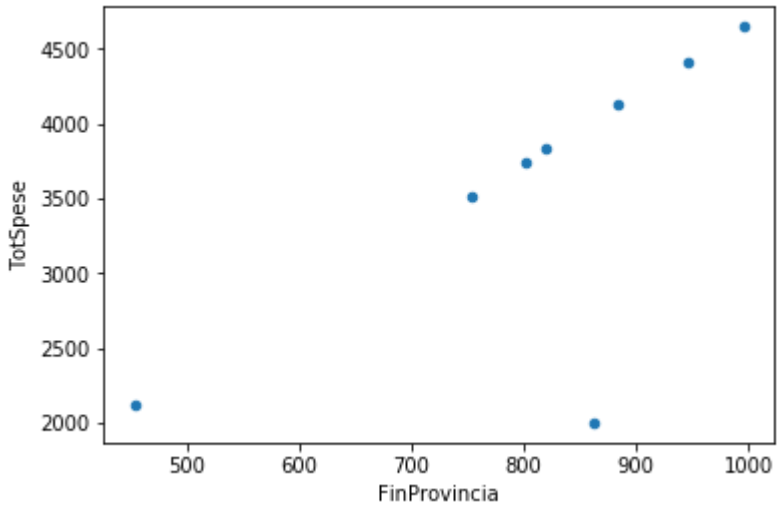
```
#6.6
print("Nessuno vedi grafico")
```

```
Nessuno vedi grafico
```

In [52]:

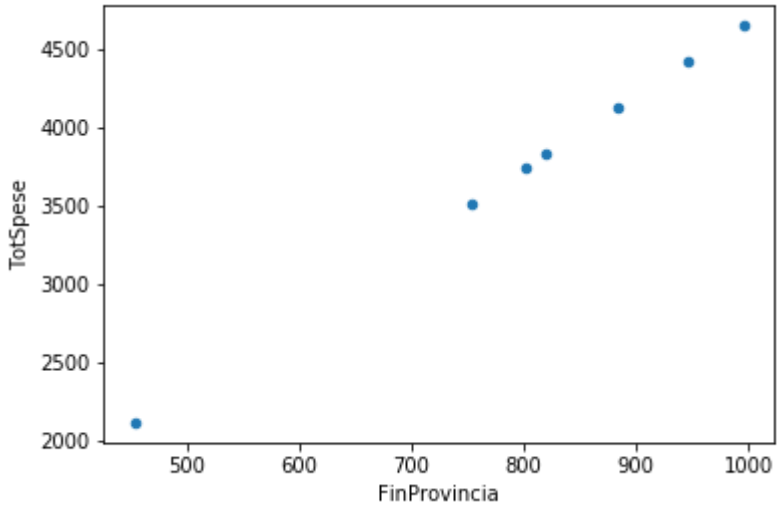
```
#6.7
print(selezione_progetti_a['FinProvincia'].corr(selezione_progetti_a['TotSpese']))
selezione_progetti_a.plot.scatter('FinProvincia','TotSpese')
plt.show()
```

```
0.6964011723762348
```



In [53]:

```
#6.8
selezione_progetti_a_noutliers=selezione_progetti_a[selezione_progetti_a['TotSpese']>20
00]
selezione_progetti_a_noutliers.plot.scatter('FinProvincia','TotSpese')
plt.show()
```



Esercizio 2

In [54]:

```
#1
len(fin)-len(fin.dropna(axis=0,subset=['TotSpese']))
```

Out[54]:

1134

In [56]:

```
#2
Z=st.norm()
dev=fin['TotSpese'].std()
(Z.ppf(1.95/2)*(len(fin))**0.5)/dev,(Z.ppf(1.95/2)*dev)/len(fin)**0.5
```

Out[56]:

(0.0005678277621445123, 6765.183171365392)

In [57]:

```
#3
```

In [58]:

```
#4
print("Normale")
```

```
Normale
```

In [59]:

```
#5
```

In [60]:

```
#6
```

In [61]:

```
#7
```

In [62]:

```
#8
```

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import statsmodels.api as sm
import numpy as np
import scipy.stats as st
```

Gennaio 2019

Esercizio 0

$X \in (-1, 1)$ con P(X = 1) = p

0.1

$$1 - P(X = 1) = P(X = -1) = 1 - p$$

0.2

$$E(X) = \sum x_i P(X = x_i) = p + (-1)(1 - p) = 2p - 1$$

0.3

$$p = \frac{E(X)}{2} + 1$$

0.4

$$Y = g(x) = X^2 = 1$$

sia per X=1 che per X = -1

0.5

$$E(Y) = E(g(X)) = E(X^2) = \sum g(X_i) P(X = X_i) = p + 1 - p = 1$$

0.6

$Var(X) = E(X^2) - E(X)^2 = E(Y) - E(X)^2 = 1 - (2p - 1)^2 = 1 - (4p^2 + 1 - 4p)$
$= 4p - 4p^2 =$

$$4p(1 - p)$$

0.7

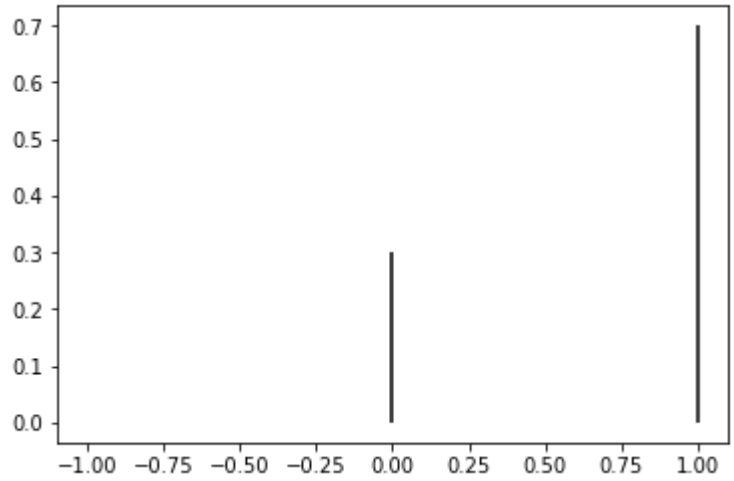
$h : R \to R$ tale che sia una bernoulliana.

$$Z = h(X) = \frac{X + 1}{2}$$

0.8

In [2]:

```
# grafico f_x,F_x,f_z,F_z
y = st.bernoulli(0.7)
x = np.arange(-1,2)
plt.vlines(x,0,y.pmf(x))
plt.show()
```



Esercizio 1

1.1

$$E(\overline{X}) = E(X) = 2p - 1$$

1.2-1.3

$$Var(\overline{X}) = \frac{1}{n}Var(X) = \frac{4p(1-p)}{n}$$

2

$T_n = \frac{1+\overline{X}}{2}$ dimostrare che non è distorto per p

$$E(\frac{1+\overline{X}}{2}) = E(\frac{1}{2} + \frac{1}{2}\sum\frac{X_i}{n}) = \frac{1}{2} + \frac{1}{2n}\sum E(X_i) = \frac{1}{2} + \frac{n}{2n}E(X) = \frac{1}{2} + \frac{2p-1}{2} = \frac{2p-1+1}{2} = p$$

3

$P(|T_n - p| \leq 0.05)$

Standardizzo

$$P(\frac{|T_n - 2p\sqrt{n}|}{\sigma} \leq \frac{2 * 0.05\sqrt{n}}{\sigma}) = P(|Z| \leq \frac{0.1\sqrt{n}}{\sigma}) \approx \Phi(\frac{0.1\sqrt{n}}{\sigma}) - \Phi(-\frac{0.1\sqrt{n}}{\sigma}) = 2\Phi(\frac{0.1\sqrt{n}}{\sigma}) - 1$$

Esercizio 2

$G \sim Exp(\nu)$

2.1

$$D_X = [0, \infty)$$

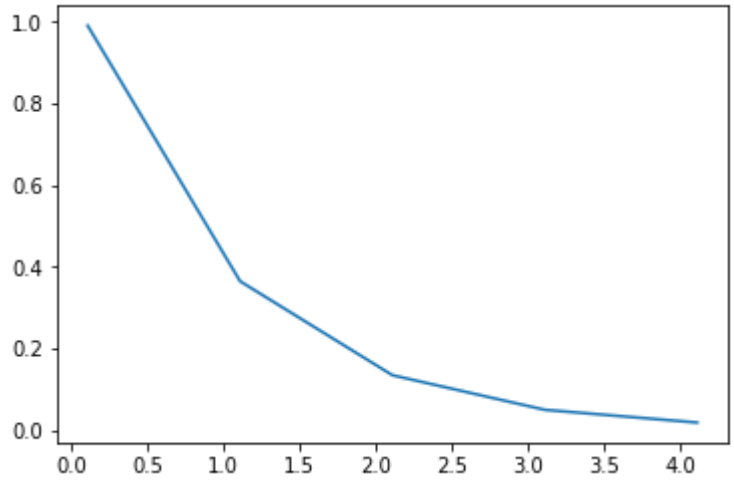
2.2

$$f_G = \nu e^{-nux}$$

2.3

In [3]:

```
nu = 0.1
y = st.expon(nu)
x = np.arange(y.ppf(0.01),y.ppf(0.99))
plt.plot(x,y.pdf(x))
plt.show()
```



2.4

$$\sqrt{(Var(G))} = \frac{1}{\nu} = E(X)$$

2.5

$$E(G) = 10$$

figura a poichè l'area sopra la curva è maggiore.

In [4]:

```
car = pd.read_csv("carsharing.csv",delimiter=";",decimal=",")
car.columns
```

Out[4]:

```
Index(['CarIdentifier', 'TimeFrame', 'RushHour', 'PremiumCustomer', 'Dista
nce',
       'Time'],
      dtype='object')
```

Esercizio 3

3.1

In [25]:

```
len(car)
```

Out[25]:

```
392
```

3.2.1

In [6]:

```
print("Qualitativo ORDINALE: {}".format(car['TimeFrame'].unique()))
```

```
Qualitativo ORDINALE: ['FRAME D' 'FRAME B' 'FRAME C' 'FRAME E' 'FRAME A']
```

3.2.2

In [7]:

```
len(car['TimeFrame'].unique())
```

Out[7]:

```
5
```

3.2.3

In [8]:

```
car['TimeFrame'].value_counts().sort_values().tail(2)
#pd.crosstab(index=car['TimeFrame'].sort_values(),columns=['Abs. Freq.'],colnames=[''])
```

Out[8]:

```
FRAME C    107
FRAME B    123
Name: TimeFrame, dtype: int64
```

3.2.4

In [9]:

```
pd.crosstab(index=car['TimeFrame'],columns=car['RushHour'],colnames=['Rush Hour'])
```

Out[9]:

Rush Hour	0	1
TimeFrame		
FRAME A	47	0
FRAME B	0	123
FRAME C	107	0
FRAME D	0	94
FRAME E	21	0

3.2.5

In [10]:

```
print("FRAME B e poi FRAME D")
```

```
FRAME B e poi FRAME D
```

3.3.1

In [26]:

```
carP = car[car['PremiumCustomer'] == 1]
len(carP)
```

Out[26]:

```
227
```

3.3.2

In [27]:

```
carP['Distance'].mean()
```

Out[27]:

```
8.437444933920705
```

3.3.3

In [28]:

```
car.PremiumCustomer.mean()
```

Out[28]:

```
0.15816326530612246
```

3.3.4

Media campionaria

3.3.5

$$P(|Tn - E(X)| < 0.05) < 1 - \frac{Var(X)}{n * (0.05)^2}$$

Non so

In [31]:

```
1-(car.PremiumCustomer.std()/((0.05**2)*len(car.PremiumCustomer)))
```

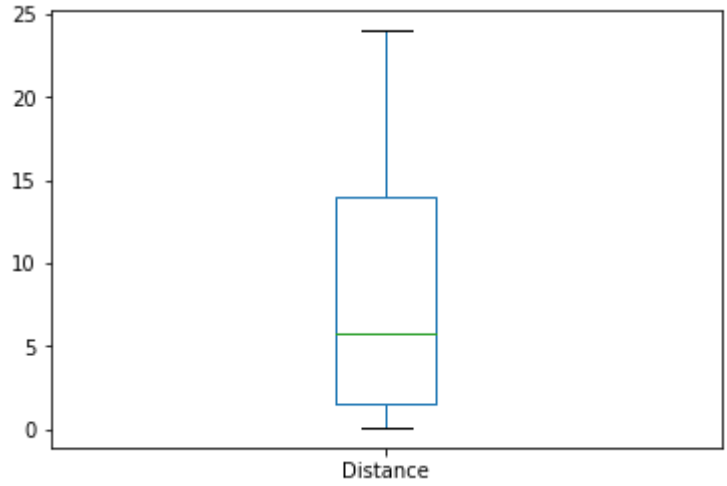
Out[31]:

```
-0.00885188191255537
```

3.4.1

In [15]:

```
car['Distance'].plot.box()
plt.show()
```



3.4.2

In [16]:

```
car['Distance'].describe()
```

Out[16]:

```
count    392.000000
mean       7.858673
std        6.805123
min        0.100000
25%        1.575000
50%        5.750000
75%       14.025000
max       24.000000
Name: Distance, dtype: float64
```

In [17]:

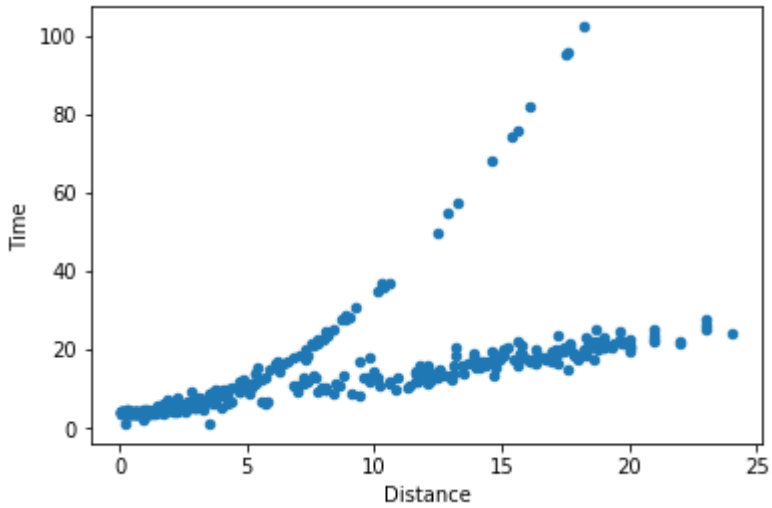
```
print('Indice di Centralità = Mediana: {}\nIndice di Dispersione = Range Interquartile
 : {}'.format(car['Distance'].quantile(0.5),(car['Distance'].quantile(0.75)-car['Distan
ce'].quantile(0.25))))
```

```
Indice di Centralità = Mediana: 5.75
Indice di Dispersione = Range Interquartile : 12.45
```

3.4.3

In [18]:

```
car.plot.scatter('Distance','Time')
plt.show()
print("Due Andamenti differenti. No relazione")
```




```
Due Andamenti differenti. No relazione
```

3.4.2

In [19]:

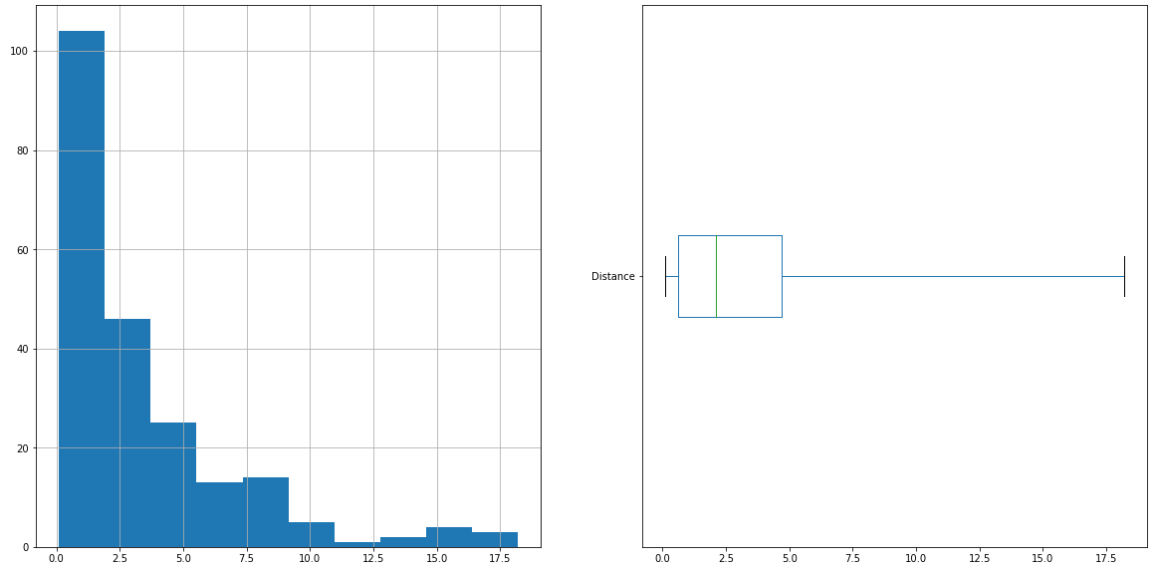
```
print("L'indice di correlazione {} conferma il fatto che non vi è una relazine di alcun
tipo".format(car['Distance'].corr(car['Time'])))
```

```
L'indice di correlazione 0.6273992247694647 conferma il fatto che non vi è
una relazine di alcun tipo
```

3.5.1

In [20]:

```
carD = car[car['RushHour'] == 1]['Distance']
plt.figure(figsize=[20,10])
plt.subplot(1,2,1)
carD.hist()
plt.subplot(1,2,2)
carD.plot.box(vert=False,whis='range')
plt.show()
```



3.5.2

In [21]:

```
print("No in quanto l'istogramma mostra che segueuna distribuzione esponenziale")
```

```
No in quanto l'istogramma mostra che segueuna distribuzione esponenziale
```

3.5.3

In [22]:

```
print(carD.mean())
print(carD.std())
```

```
3.3193548387096796
3.711106147915895
```

3.5.4

In [23]:

```
print("Esponenziale")
```

```
Esponenziale
```

3.5.5

In [24]:

```
print("Si perchè sono molto simili")
```

```
Si perchè sono molto simili
```

Febbraio 2019

Esercizio 0

Conosco P(A),P(B),P(A|B)

1)

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

$X \sim (0,1), P(X=1) = p$

2.1)

$$E(X) = p\$, \$\sigma = \sqrt{p(1-p)}$$

2.2)

$$0.3 = \sqrt{p-p^2}$$

$$0.09 = p - p^2$$

$$+p^2 - p + \frac{9}{100} = 0$$

$$(p - \frac{9}{10})(p - \frac{1}{10}) = 0 => p = 0.9 || p = 0.1$$

2.3)

$$F^{-1}(std(X)) = \frac{-2p+1}{2\sqrt{p-p^2}}$$

$$-2p + 1 = 0$$

$$p = 0.5$$

2.4) p = 0.45

Esercizio 1

$\overline{X}$

1.1) $\forall X_i = 0$ avrò

$$\sum \frac{X_i}{n} = 0$$

1.2) $\forall X_{1,2} = 1$ avrò

$$\sum \frac{X_i}{n} = \frac{2}{n}$$

1.3)$\forall X_i = 1$ avrò

$$\sum \frac{X_i}{n} = 1$$

2)

$$\{0, \frac{1}{n}, \frac{2}{n}, \dots, 1\}$$

3) è uno stimatore non distorto di p perchè la media campionaria è sempre iuno stimatore non distorto del valore atteso

$$E(\overline{X}) = E(\sum \frac{X_i}{n}) = \frac{1}{n} \sum E(X) = \frac{1}{n} n E(X) = E(X) = p$$

4) $n >> 1$ dimostrare che $P(|\bar(X) - p| \leq \epsilon) \geq 2\Phi(2\epsilon\sqrt{n}) - 1$

Standardizzo $P(|Z| \leq \frac{\epsilon\sqrt{n}}{\sigma}) \approx 2\Phi(\frac{\sqrt{n}\epsilon}{\sigma}) - 1$

Poichè $Var(X) \leq \frac{1}{4}$ allora la deviazione standard sarà max $\frac{1}{2}$ quindi avrò

$$2\Phi(2\sqrt{n}\epsilon) - 1$$

In [1]:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import statsmodels.api as sm
import scipy.stats as st
```

In [2]:

```
car = pd.read_csv("carsharing.csv",delimiter=";",decimal=",")
car[:5]
```

Out[2]:

	CarIdentifier	TimeFrame	RushHour	PremiumCustomer	Distance	Time
0	102	FRAME D	1	1	3.0	7.9
1	103	FRAME D	1	1	5.3	13.9
2	105	FRAME D	1	-1	0.4	4.1
3	110	FRAME D	1	1	2.8	5.0
4	110	FRAME B	1	-1	2.7	5.6

Esercizio 2

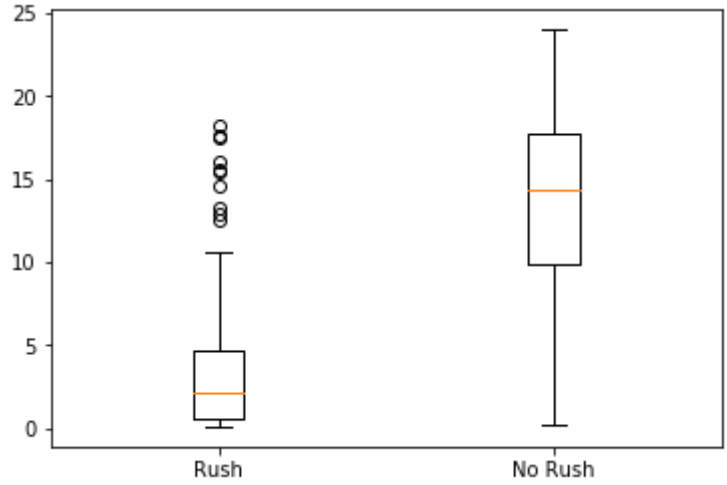
In [3]:

```
#2.1
print("Scalare: {}".format(car['Distance'].unique()))
```

```
Scalare: [ 3.   5.3  0.4  2.8  2.7 11.8  9.3  7.   4.  13.1  0.8  3.5 13.4
0.1
  1.   1.2 18.5  0.9  6.8  1.9 10.3 17.6  1.6  2.2  2.3 14.   0.2  0.3
  1.8 12.   6.3 18.  17.4 10.9 15.8  2.1 11.4  0.5 17.1  7.3 19.6 12.2
 13.5 21.  23.   7.1 12.1  1.5 16.6  1.7 16.5  0.6 15.1 14.8 15.7 13.
 14.6  4.7  4.2 17.2 16.7  3.1 19.   8.2  1.3 14.9  7.8  5.2 14.4  3.4
  7.4 12.4 13.3 20.   0.7  5.7  3.8 11.3  2.6  4.4 14.7 18.4 18.2  6.1
  6.6 19.7  3.7 10.6  3.2 13.9 11.6  5.6 15.6 16.3  3.3  7.2 16.1  7.6
  2.  19.1 17.8 16.  17.   4.9  8.8  9.7 19.4 15.4  9.4  8.4  1.1 18.8
 19.9  7.9  8.1 13.7 11.7  5.8 17.7 12.6 10.  15.9  8.   8.9 14.3 10.1
 17.5 14.1  8.7 18.3  6.2 19.3  9.1 10.4  9.8 12.5  6.4 16.2  3.6  5.4
 12.8  5.1  1.4 22.   8.5  9.  12.9 24.  10.8  3.9  9.6  8.6 15.2  5.
 16.8  4.1  5.9  2.5  7.5 13.2 10.2 15.5 11.9 18.7  7.7 14.5 18.6]
```

In [19]:

```
#2.2
carP = car[car['RushHour'] == 1]['Distance']
carNP = car[car['RushHour'] == 0]['Distance']
plt.boxplot([carP,carNP], labels=['Rush','No Rush'])
plt.show()
```



2.3

Negli orari di punta sono privilegiati gli spostamenti brevi come si può notare dal 3° quartile < 5km, al contrario per gli orari non di punta dove il 75% degli spostamenti supera i 10km

In [6]:

```
#4
print(carNP.mean())
print(carP.mean())
print("Possiamo dire che la distanza è maggiore nelle ore non di punta e che quindi abb
iamo una fascia breve di ore di punta, una grande di ore non di punta")
```

```
13.487428571428563
3.3193548387096796
Possiamo dire che la distanza è maggiore nelle ore non di punta e che quin
di abbiamo una fascia breve di ore di punta, una grande di ore non di punt
a
```

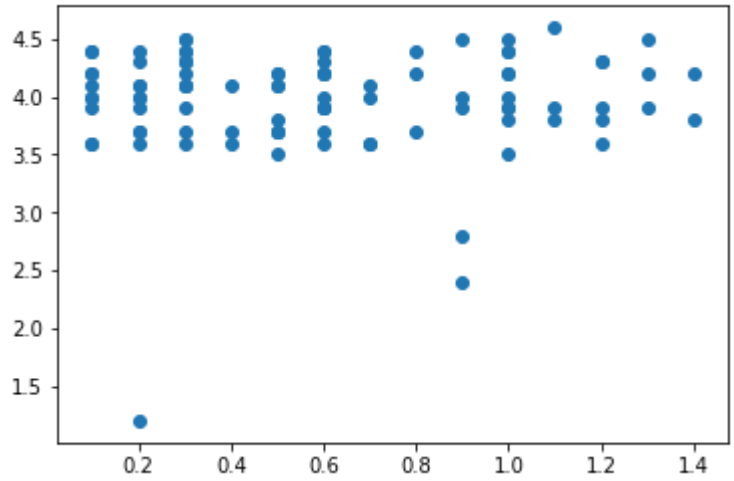
Esercizio 3

In [20]:

```
#3.1
tragittibrevi = car[car['Distance'] < 1.5]
```

In [21]:

```
#3.2
plt.scatter(tragittibrevi['Distance'], tragittibrevi['Time'])
plt.show()
```



In [22]:

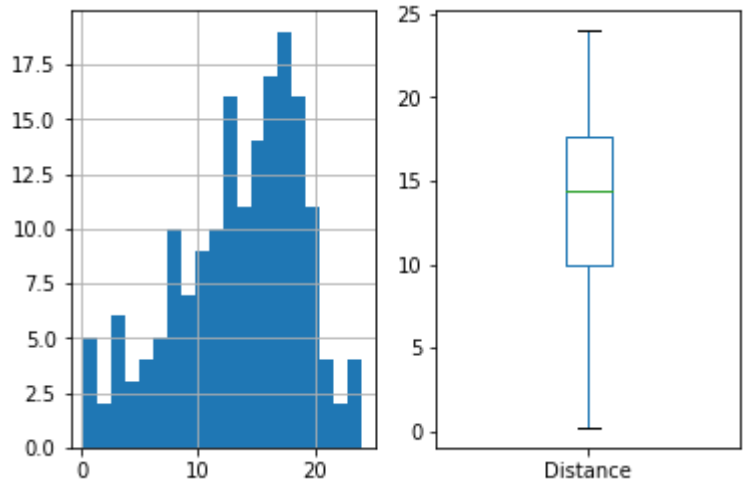
```
#3.3
print('Sia dal gragico che dal valore del coefficiente di correlazione {}, possiamo con
fermare che non vi è alcuna relazione tra i due valori presi in considerazione'.format(
tragittibrevi['Distance'].corr(tragittibrevi['Time'])))
```

```
Sia dal gragico che dal valore del coefficiente di correlazione 0.03691131
525657363, possiamo confermare che non vi è alcuna relazione tra i due val
ori presi in considerazione
```

Esercizio 4

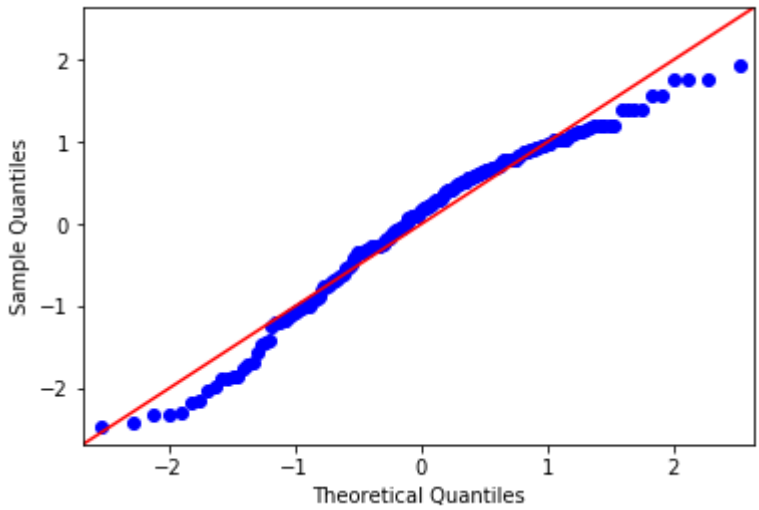
In [23]:

```
#4.1
plt.subplot(1,2,1)
carNP.hist(bins=20)
plt.subplot(1,2,2)
carNP.plot.box()
plt.show()
```



In [26]:

```
# 4.2
sm.qqplot(carNP, fit=True, line='45')
plt.show()
print(carNP.mean(), carNP.median())
print("Il grafico quasi sovrapposto alla bisettrice e la vicinanza tra media e mediana
 fanno intuire un comportamento normale")
```




```
13.487428571428563 14.4
Il grafico quasi sovrapposto alla bisettrice e la vicinanza tra media e me
diana fanno intuire un comportamento normale
```

In [11]:

```
print('Media : {}\nMediana : {}\nSia dal grafico che dal valore della Media e Mediana p
ossiamo dire che la Distanza negli orari non di punta segue un andamento approssimativa
mente normale con coda a sinistra'.format(carNP.mean(),carNP.quantile(0.5)))
```

```
Media : 13.487428571428563
Mediana : 14.4
Sia dal grafico che dal valore della Media e Mediana possiamo dire che la
Distanza negli orari non di punta segue un andamento approssimativamente n
ormale con coda a sinistra
```

Esercizio 5

In [27]:

```
#5.1
len(car[car['RushHour'] == 1])/len(car)
car['RushHour'].mean()
```

Out[27]:

```
0.5535714285714286
```

In [13]:

```
#5.2
print("Media campionaria")
```

Media

In [28]:

```
#5.3
campione = len(car.dropna())
campione
```

Out[28]:

392

5.4

$$P(|\bar{X}_n - \mu| <= 0.025)$$

$$P(-0.025 <= \bar{X}_n - \mu <= 0.025)$$

$$P(-\frac{0.025}{\frac{\sigma}{\sqrt{n}}} <= \frac{\bar{X}_n - \mu}{\frac{\sigma}{\sqrt{n}}} <= \frac{0.025}{\frac{\sigma}{\sqrt{n}}})$$

$$P(-\frac{0.025}{\frac{\sigma}{\sqrt{n}}} <= Z <= \frac{0.025}{\frac{\sigma}{\sqrt{n}}})$$

$$P(Z <= \frac{0.025}{\frac{\sigma}{\sqrt{n}}}) - P(Z <= -\frac{0.025}{\frac{\sigma}{\sqrt{n}}})$$

$$P(|\bar{X}_n - \mu| <= 0.025) = \phi(\frac{0.025}{\frac{\sigma}{\sqrt{n}}}) - \phi(-\frac{0.025}{\frac{\sigma}{\sqrt{n}}})$$

In [29]:

```
p=0.05
Z = st.norm()
dev=car['RushHour'].std()
n = len(car.dropna())
x1 = (0.025*(n)**0.5)/dev
Z.cdf(x1)-Z.cdf(-x1)
```

Out[29]:

0.6799767876150911

Esercizio 6

incidente = A = 0.15 \ orario di punta = B = 0.55

P(A|B) = 0.2

$$P(B|A) = \frac{P(B \cap A)}{P(A)}$$

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

In [16]:

```
p = 0.15
pp = 0.2
prob = (pp*(len(car[car['RushHour'] == 1])/len(car)))/p
```

In [17]:

```
print('La probabilità che una data auto oggi non è disponibile perché ieri ha subito un
incidente è : {}'.format(prob))
```

```
La probabilità che una data auto oggi non è disponibile perché ieri ha sub
ito un incidente è : 0.7380952380952381
```

In [1]:

```
import numpy as np
import pandas as pd
import scipy.stats as st
import matplotlib.pyplot as plt
import statsmodels.api as sm
import math
```

Giugno 2019

Esercizio 0

$Y \sim UnifDisc(s)$

$X \sim Bern(p)$ con p=0.8

0.1.1-0.1.2-0.2.1-0.2-2

grafici dispersione e ripartizione

0.2.3

$$E(X) = p$$
$$Var(X) = p(1-p)$$

Esercizio 1

$\overline{X_n} = \sum \frac{X_i}{n}$

1.1

$$E(\overline{X}) = p$$

1.2

$$Var(\overline{X}) = \frac{1}{n} Var(X)$$

1.3

T_n stimatore per il valore atteso

$$T_n = \overline{X_n}$$

1.4

$E(T_n) = E(\sum \frac{X_i}{n}) = \frac{1}{n} \sum E(X_i) = \frac{1}{n} n E(X) = E(X) = p$ non è distorto

1.5

U_n stimatore varianza non distorto

$$E(Var(X)) = \sum (1-p)p = np(1-p)$$

$$E(U_n) = E(nT_n(1-T_n))$$

??????????

Esercizio 2

$0 < \delta < 1, \epsilon > 0$

2.1

$P(|\overline{\overline{X_n}} - p| \leq \epsilon) \geq 1 - \delta \approx P(|X^*| \leq \frac{\epsilon}{\sqrt{\overline{p(1-p)}}}\sqrt{n})$

$P(|\frac{X_n - p}{\frac{\sigma}{\sqrt{n}}}| \leq \frac{\epsilon\sqrt{n}}{\sqrt{p(1-p)}} \geq 1 - \delta$

2.2

$P(|\overline{X} - p| \leq \epsilon) \geq 1 - \delta$

$2\Phi(2\epsilon\sqrt{n}) - 1 \geq 1 - \delta$

$\Phi(2\epsilon\sqrt{n}) \geq 1 - \frac{\delta}{2}$

2.3

$\delta = 0.05, \epsilon = 0.01$

$\Phi(2\epsilon\sqrt{n}) \geq 1 - \frac{\delta}{2} = \Phi(0.02\sqrt{n}) \geq 1 - 0.025$

$\Phi(0.02\sqrt{n}) \geq 0.975$

$\sqrt{n} \geq \Phi^-1(0.975) * 0.02$

In [2]:

```
x = st.bernoulli(0.8)
n = x.ppf(0.975)
math.sqrt(n)/0.02
```

Out[2]:

50.0

In [3]:

```
imp = pd.read_csv("impiantitermici.csv",sep=";",decimal=".",parse_dates=True)
imp[:5]
```

```
C:\Anaconda\lib\site-packages\IPython\core\interactiveshell.py:2785: Dtype
Warning: Columns (34) have mixed types. Specify dtype option on import or
set low_memory=False.
  interactivity=interactivity, compiler=compiler, result=result)
```

Out[3]:

	IDENTIFICATIVO_IMPIANTO	GENERATORI_NUMERO	POTENZA_IMPIANTC
0	2f2df97825995cbd6f840fa2889c5ff89	1	28.0
1	2e6db9dfe3d91c4d4fc41fc2086c1ff89	1	24.0
2	2e4df9b8c2895c4ddfc41fd278ec3ff89	1	24.0
3	2f9cc9d8c3b96c5dcfd40fb288dc4ff89	1	31.5
4	287ad9b9d2592c7dcfb49f82387c6ff89	1	24.0

5 rows × 39 columns

Esercizio 3

3.1

In [4]:

```
len(imp)
```

Out[4]:

189997

3.2

In [5]:

```
print("scalare")
```

scalare

3.3

In [6]:

```
len(imp[imp['GENERATORE_DATA_INST'] < '01/01/1940'])
```

Out[6]:

```
0
```

3.4

In [23]:

```
mask1 = imp['POTENZA_IMPIANTO_RISC'] > 15.0
mask2 = imp['POTENZA_IMPIANTO_RISC'] < 35.0
mask3 = imp['GENERATORE_COMBUSTIBILE'] == 'GAS NATURALE'
selezione = imp[mask1 &  mask2 & mask3]
```

3.5

In questo caso l'eterogeneità è massima perchè tutti i valori sono uguali, in quanto abbiamo posto come condizione di considerare come combustibile il gas naturale

3.6

In [9]:

```
selezione['EDIFICIO_CATEGORIA'].unique()
```

Out[9]:

```
array(['E11', 'E1', 'E8', 'E2', 'E43', 'E5', nan, 'E7', 'E3', 'E42', 'E4',
       'E13', 'E12', 'E6', 'E62', 'E41', 'E63', 'E61'], dtype=object)
```

3.7

In [25]:

```
selezione['EDIFICIO_CATEGORIA'].mode()
```

Out[25]:

```
0    E1
dtype: object
```

3.8

In [27]:

```
len(selezione[selezione['EDIFICIO_CATEGORIA'] == 'E1'])
```

Out[27]:

```
78875
```

3.9

In [28]:

```
#9
len(selezione[selezione['EDIFICIO_CATEGORIA'] == 'E1'])/len(selezione) * 100
```

Out[28]:

59.81813769357945

In [12]:

```
### Esercizio 4 non si riesce a fare il confronto fra date
```