Pattern Recognition
and Applications Lab

La Programmazione ad Oggetti in Python

Docente: Ambra Demontis

Anno Accademico: 2024 - 2025

University of Cagliari, Italy

Department of Electrical and Electronic Engineering

Corso di Laurea in Ingegneria Elettronica, Informatica e delle Telecomunicazioni

La Programmazione ad Oggetti in Python

Interpretare i dati:

- la libreria per la visualizzazione scientifica matplotlib
- la libreria per il calcolo statistico scipy

Visualizzare i Dati

Per trarre delle informazioni utili da dei dati è spesso utile visualizzarli.

In Python, questo si può fare con la libreria **matplotlib**.

In particolare, si utilizza il modulo fornito da questa libreria chiamato **pyplot**, che fornisce funzioni utili per creare i grafici.

Visualizzare i Dati

Prima di tutto importiamo quindi questo modulo.

Poichè il nome del modulo è lungo, per importarlo utilizziamo la sintassi:
Import <modulo> as <nome>
Che ci permetterà di riferirci al modulo usando <nome> invece che il nome del modulo.

```
import matplotlib.pyplot as plt
```

La classe Figura

Per poter creare un qualsiasi tipo di grafico, bisogna prima di tutto creare un oggetto di tipo figura.

Per far questo, dobbiamo richiamare le funzione ***figure*** del modulo pyplot, eventualmente passandogli come argomento una tupla contenente larghezza e altezza (in pollici*) che la figura deve avere.

plt.figure(figsize=(<altezza>,<larghezza>))

* 1 pollice = 2.54 centimetri

La Funzione Subplot

Creiamo poi uno spazio per inserire un grafico nella nostra figura utilizzando la funzione subplot.

ax = plt.subplot()

Questa funzione volendo ci permette di inserire anche più grafici nella stessa figura.

ax = plt.subplot(nrows, ncols, index)

Plot

Supponiamo ora di essere dei programmatori di una multinazionale e che ci venga chiesto di creare un grafico che mostra l'andamento delle vendite di uno dei nostri prodotti durante il 2020.

Vogliamo quindi **visualizzare come varia una variabile** (il numero delle vendite) **al crescere di un'altra variabile** (il tempo).

Per far questo possiamo creare, dentro la figura, un grafico di tipo **plot**.
plt.plot(<valori_variabile_x>, <valori_variabile_y>)

Questa funzione (come tutte le altre) agiscono sulla figura attiva.

Le Funzioni Savefig, Show e Close

Una volta che abbiamo creato il grafico possiamo salvare, visualizzare e poi chiudere la figura.

Per **salvarla**:
plt.savefig(<percorso>)

Per **visualizzarla** si utilizza la funzione:
plt.show()

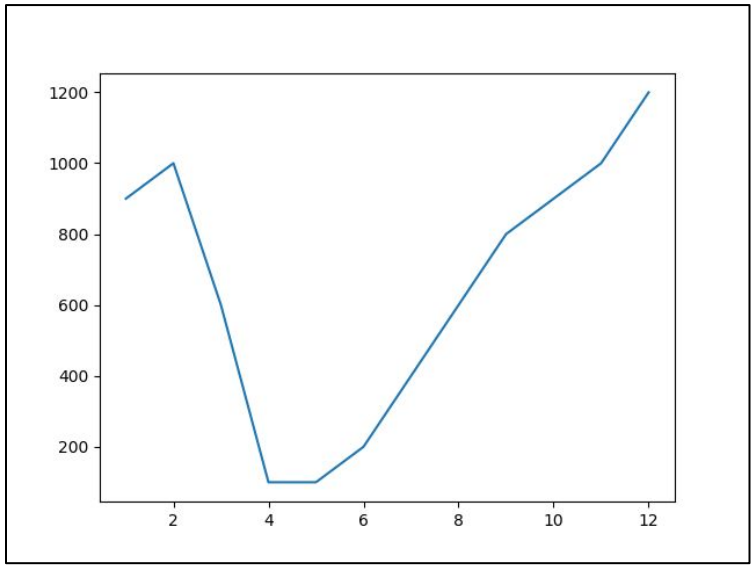
Per **chiuderla** si utilizza la funzione:
plt.close()

Plot

```
import matplotlib.pyplot as plt

vendite = [900, 1000, 600, 100, 100, 200, 400, 600, 800, 900, 1000, 1200]
mesi = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

plt.figure()
ax = plt.subplot()
plt.plot(mesi, vendite)
plt.savefig("grafico_vendite.png")
plt.show()

plt.close()
```

Plot

Il codice mostrato nella slide precedente produce il grafico:



Questo grafico mostra la curva ma ha un aspetto poco leggibile e professionale..

Funzioni per Modificare la Grafica dei Plot.

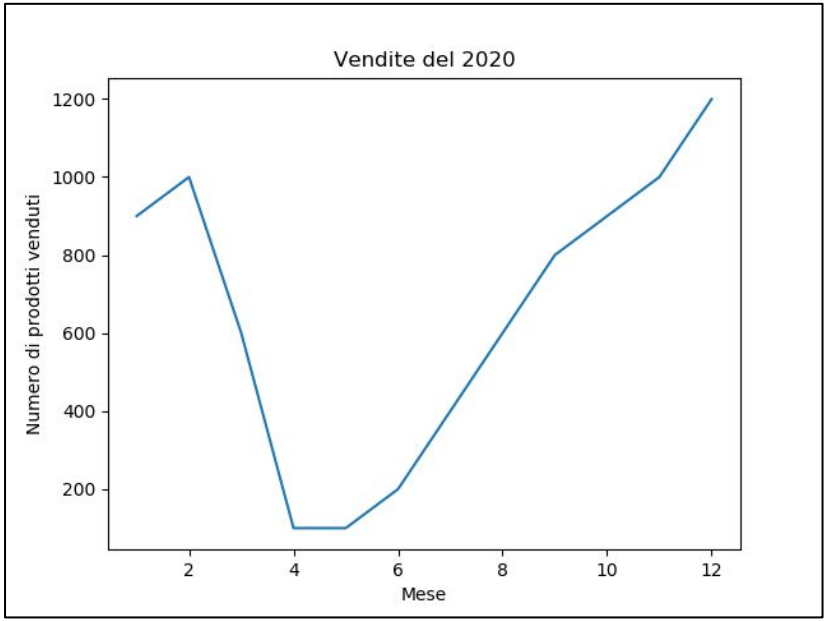
Matplotlib mette a disposizione diverse funzioni per modificare l'aspetto dei plot.

Prima di tutto cerchiamo di rendere più chiaro cosa questo grafico mostra, aggiungendo:

- Un titolo (title)
- Il nome della variabile i cui valori sono mostrati sull'asse x (xlabel)
- Il nome della variabile i cui valori sono mostrati sull'asse y (ylabel)

Funzioni per Modificare la Grafica dei Plot.

```
import matplotlib.pyplot as plt

vendite = [900, 1000, 600, 100, 100, 200, 400, 600, 800, 900, 1000, 1200]
mesi = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

plt.figure()
plt.plot(mesi, vendite)
ax = plt.subplot()

plt.title('Vendite del 2020')
plt.xlabel('Mese')
plt.ylabel('Numero di prodotti venduti')

plt.savefig("grafico_vendite.png")
plt.show()
plt.close()
```



Funzioni per Modificare la Grafica dei Plot.

La grafica può ancora essere migliorata...

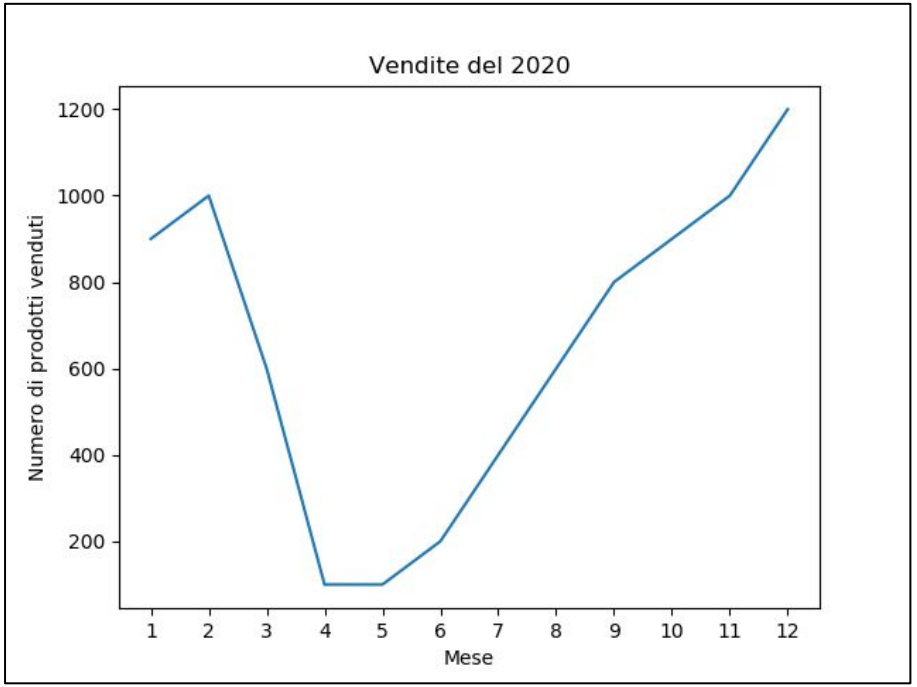
Prima di tutto invece di mostrare un mese si e un mese nò, vogliamo mostrare tutti i mesi.

Utilizziamo la funzione **xticks** che riceve come argomento la lista dei tick da mostrare sull'asse x.

In pratica determina in quanti "settori" deve essere suddivisa l'asse x.

Funzioni per Modificare la Grafica dei Plot.

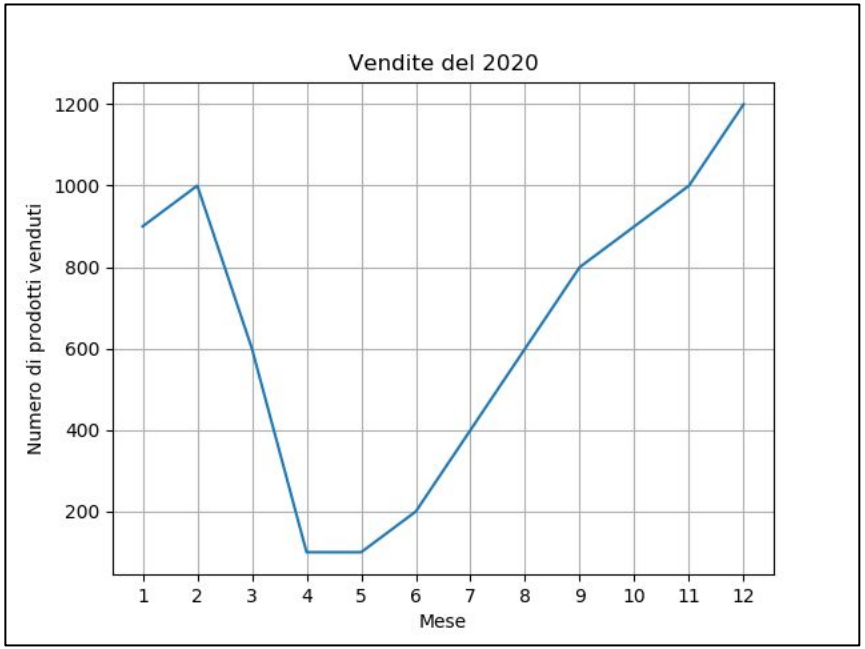
Aggiungiamo nel codice:
plt.xticks(mesi)



Funzioni per Modificare la Grafica dei Plot.

Per rendere più semplice a chi guarda il grafico capire qual'è stato il numero di prodotti venduti in un determinato mese, aggiungiamo una griglia utilizzando la funzione grid.

plt.grid()



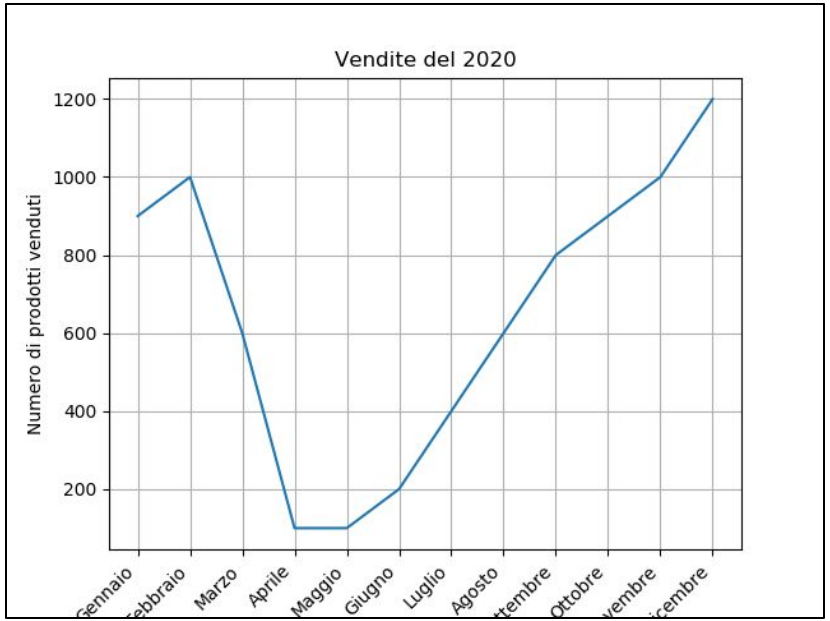
Funzioni per Modificare la Grafica dei Plot.

Sostituiamo all'indice del mese il suo nome settando il nome dei ticks.

```
nomi_mesi = ["Gennaio", "Febbraio", "Marzo", "Aprile", "Maggio", "Giugno","Luglio", "Agosto",
"Settembre", "Ottobre", "Novembre", "Dicembre" ]
ax.set_xticklabels(nomi_mesi, rotation = 45, ha="right")
```

Funzioni per Modificare la Grafica dei Plot.

Il grafico che otteniamo è questo:



Alcuni nomi dei mesi risultano tagliati e c'è tanto spazio vuoto in alto e in basso..

Funzioni per Modificare la Grafica dei Plot.

Per posizionare il grafico all'interno della figura possiamo utilizzare la funzione **subplots_adjust**.

Questa funzione determina la distanza che deve avere il grafico (il rettangolo che mostra la curva) dallo spazio assegnato al plot all'interno della figura.
Nel caso in cui (come nel caso di esempio) abbiamo un solo grafico, tutto lo spazio della figura sarà assegnato al grafico.

Come argomenti riceve, in ordine, la distanza che deve avere dal margine:
Sinistro, inferiore, destro, superiore.

Funzioni per Modificare la Grafica dei Plot.

Partiamo dai valori di default:
plt.subplots_adjust(0.125, 0.1, 0.9, 0.9)

E li sistemiamo in modo da eliminare lo spazio bianco e far sì che tutte le scritte vengano mostrate correttamente nel grafico.
plt.subplots_adjust(0.125, 0.2, 0.98, 0.95)

Funzioni per Modificare la Grafica dei Plot.

Facciamo si che venga mostrata solo la parte del grafico di interesse (quella che contiene la curva, evitando che ci sia spazio vuoto a sinistra e a destra).

```
plt.xlim(1,12)
```

Funzioni per Modificare la Grafica dei Plot.

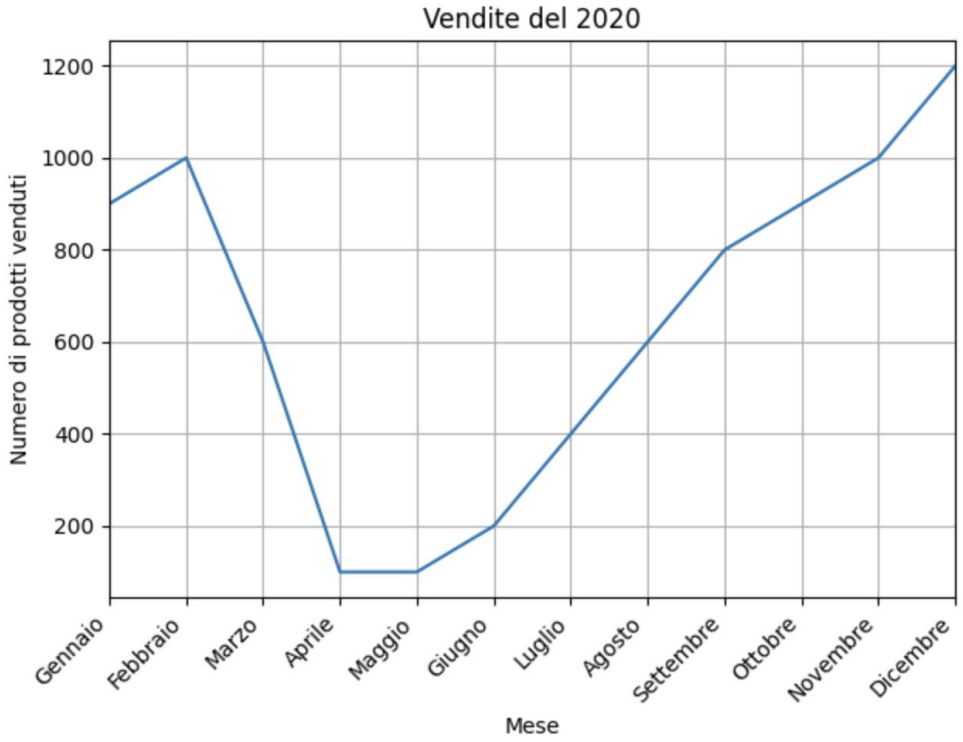
Il codice finale è:

```
import matplotlib.pyplot as plt

vendite = [900, 1000, 600, 100, 100, 200, 400, 600, 800, 900, 1000, 1200]
mesi = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

plt.figure()
ax = plt.subplot()
plt.plot(mesi, vendite)

plt.title('Vendite del 2020')
plt.xlabel('Mese')
plt.ylabel('Numero di prodotti venduti')
plt.xticks(mesi)
```

Funzioni per Modificare la Grafica dei Plot.

```
plt.grid()

nomi_mesi = ["Gennaio", "Febbraio", "Marzo", "Aprile", "Maggio", "Giugno",
    "Luglio", "Agosto", "Settembre", "Ottobre", "Novembre", "Dicembre" ]
ax.set_xticklabels(nomi_mesi, rotation = 45, ha="right")

plt.xlim(1,12)

plt.subplots_adjust(0.125, 0.2, 0.98, 0.95)

plt.savefig("grafico_vendite.png")
plt.show()
plt.close()
```

Funzioni per Modificare la Grafica dei Plot.



Esercizio sulla Creazione di Plot

Una startup, fondata a giugno del 2020 vi chiede di creare un grafico che mostra il suo fatturato nel 2020.

Il suo fatturato è stato il seguente.
Giugno: 0
Luglio: 0
Agosto: 200 euro
Settembre: 400 euro
Ottobre: 300 euro
Novembre: 800 euro
Dicembre: 1600 euro

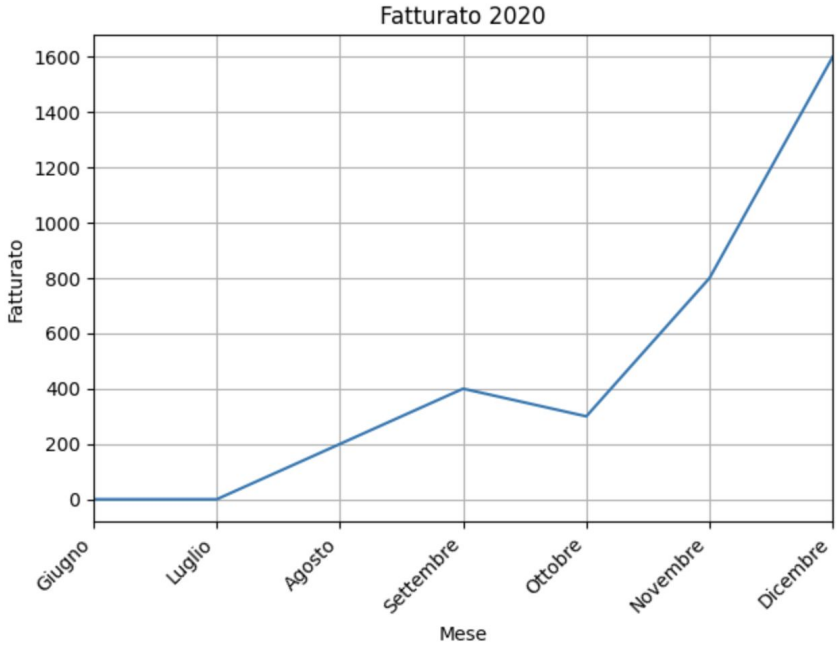
Esercizio sulla Creazione di Plot

```
import matplotlib.pyplot as plt

vendite = [0, 0, 200, 400, 300, 800, 1600]
mesi = [6, 7, 8, 9, 10, 11, 12]

plt.figure()
ax = plt.subplot()
plt.plot(mesi, vendite)

plt.title('Fatturato 2020')
plt.xlabel('Mese')
plt.ylabel('Fatturato')
plt.xticks(mesi)
```

Esercizio sulla Creazione di Plot

```
plt.grid()

nomi_mesi = ["Giugno",
    "Luglio", "Agosto", "Settembre", "Ottobre", "Novembre", "Dicembre" ]
ax.set_xticklabels(nomi_mesi, rotation = 45, ha="right")

plt.xlim(6,12)

plt.subplots_adjust(0.125, 0.2, 0.98, 0.95)

plt.savefig("grafico_vendite.png")
plt.show()
plt.close()
```

Esercizio sulla Creazione di Plot



Scatter Plot

Un’altra cosa che capita frequentemente è di avere, per diversi campioni, i valori assunti da due variabili e di voler capire se queste due variabili sono correlate (se c’è tra esse una relazione).

Supponete di voler capire se in Italia, il reddito delle persone cresce al crescere del loro grado di istruzione. Per capirlo, dovremmo avere, per un numero significativo di persone, la loro età e il loro grado di istruzione.

Possiamo poi utilizzare uno **scatter plot** per visualizzare la relazione tra le due variabili.

Scatter Plot

Supponiamo le misure a nostra disposizione siano:
reddito = [2400, 1850, 1900, 1750, 1500, 1550, 1100, 800, 500, 400,450, 300]
grado_istruzione = [6, 6, 5, 5, 4, 4, 3, 3, 2, 2, 1, 1]

Dove per il grado di istruzione:
1 rappresenta le scuole elementari
2 rappresenta le scuole medie
3 rappresenta le scuole superiori
4 rappresenta la triennale
5 rappresenta la magistrale
6 rappresenta il dottorato

Scatter Plot

Possiamo creare uno **scatter plot** (un grafico che invece di una curva mostra dei puntini).

Per far questo possiamo utilizzare la funzione di matplotlib chiamata ***scatter.***

plt.scatter(<valori_variabile_x>, <valori_variabile_y)

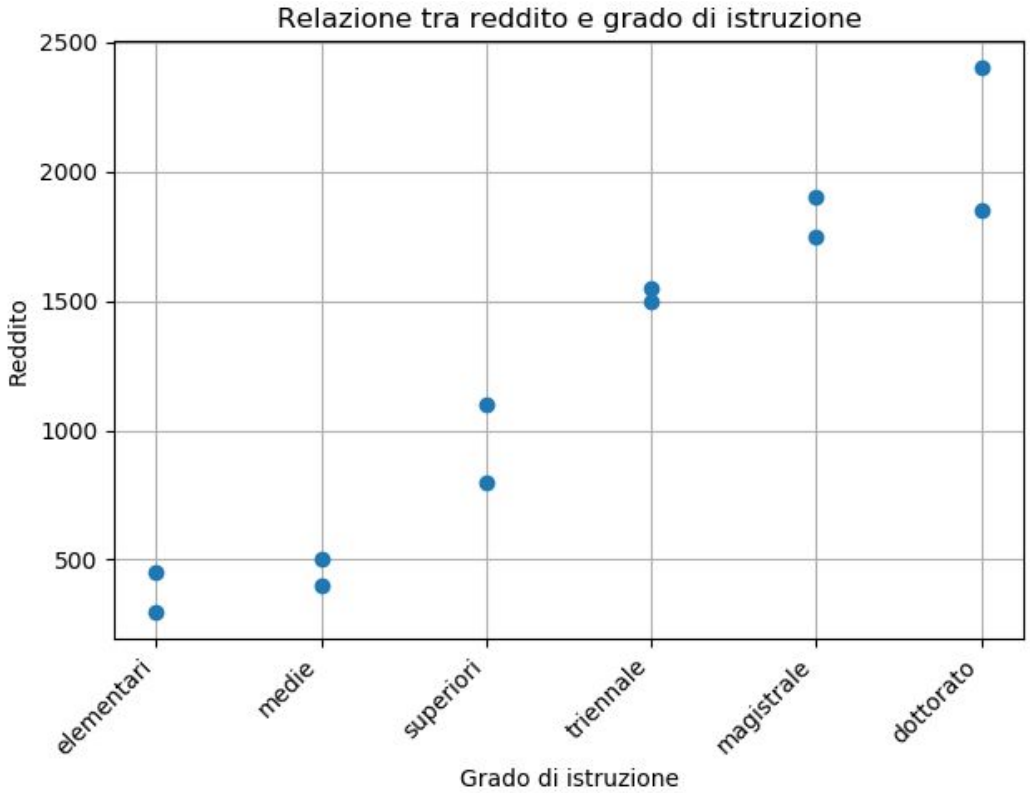
Scatter Plot

```
import matplotlib.pyplot as plt

reddito = [2400, 1850, 1900, 1750, 1500, 1550, 1100, 800, 500, 400,
       450, 300]
grado_istruzione = [6, 6, 5, 5, 4, 4, 3, 3, 2, 2, 1, 1]

plt.figure()
ax = plt.subplot()
plt.scatter(grado_istruzione, reddito)

plt.title('Relazione tra reddito e grado di istruzione')
plt.xlabel('Grado di istruzione')
plt.ylabel('Reddito')
plt.xticks([1,2,3,4,5,6])
```

Scatter Plot

```
plt.grid()
ax.set_axisbelow(True) # fa si che la griglia venga mostrata dietro i punti

nomi_grado_istr = ["elementari", "medie", "superiori", "triennale",
                   "magistrale", "dottorato"]
ax.set_xticklabels(nomi_grado_istr, rotation = 45, ha="right")

plt.subplots_adjust(0.125, 0.2, 0.98, 0.95)

plt.savefig("grafico_reddito_istruzione.png")
plt.show()
plt.close()
```

Scatter Plot



Calcolo della Correlazione

Nel grafico mostrato è ovvio che c'è una correlazione positiva (cioè che il reddito aumenta al variare del grado di istruzione).

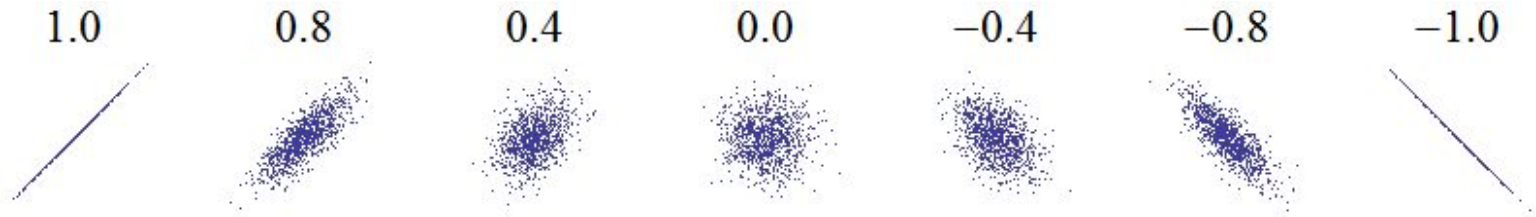
Tuttavia in altri casi potrebbe non essere così evidente.

In quei casi, possiamo calcolare una misura statistica, il coefficiente di correlazione, utilizzando la libreria scipy.

Calcolo della Correlazione

Ad esempio possiamo utilizzare la funzione **pearsonr** del modulo di scipy chiamato **stats** che **calcola il coefficiente di correlazione di Pearson.**

Questo coefficiente misura la correlazione lineare:



Calcolo della Correlazione

La funzione **pearsonr** fornisce in output due numeri:

- Il coefficiente di correlazione lineare
- Il pvalue: un numero che ci indica quanto il coefficiente di correlazione lineare è significativo statisticamente (affidabile), più è basso, meglio è. La correlazione è statisticamente significativa se il pvalue è < 0.05.

Calcolo della Correlazione

Proviamo a calcolarlo per l'esempio che abbiamo visto:

```
from scipy.stats import pearsonr

p_coef, pval = pearsonr(grado_istruzione, reddito)

print("Il coefficiente di correlazione di Pearson è: ", p_coef)
print("Il pvalue è : ", pval)
```

Stamperà:
Il coefficiente di correlazione di Pearson è: 0.9659753537324693
Il pvalue è : 3.3917980104823314e-07

Esercizio sulla Creazione di Scatter Plot

Si vuole capire se in Italia c'è una correlazione tra il reddito e l'età delle persone.
I dati che vi vengono forniti sono:
Età: [50, 30, 47, 28, 26, 22, 50, 20, 33]
Reddito: [2400, 1800, 1400, 1450, 1200, 800, 850, 750, 1500]

Creare uno scatter plot e calcolare il coefficiente di correlazione di Pearson.

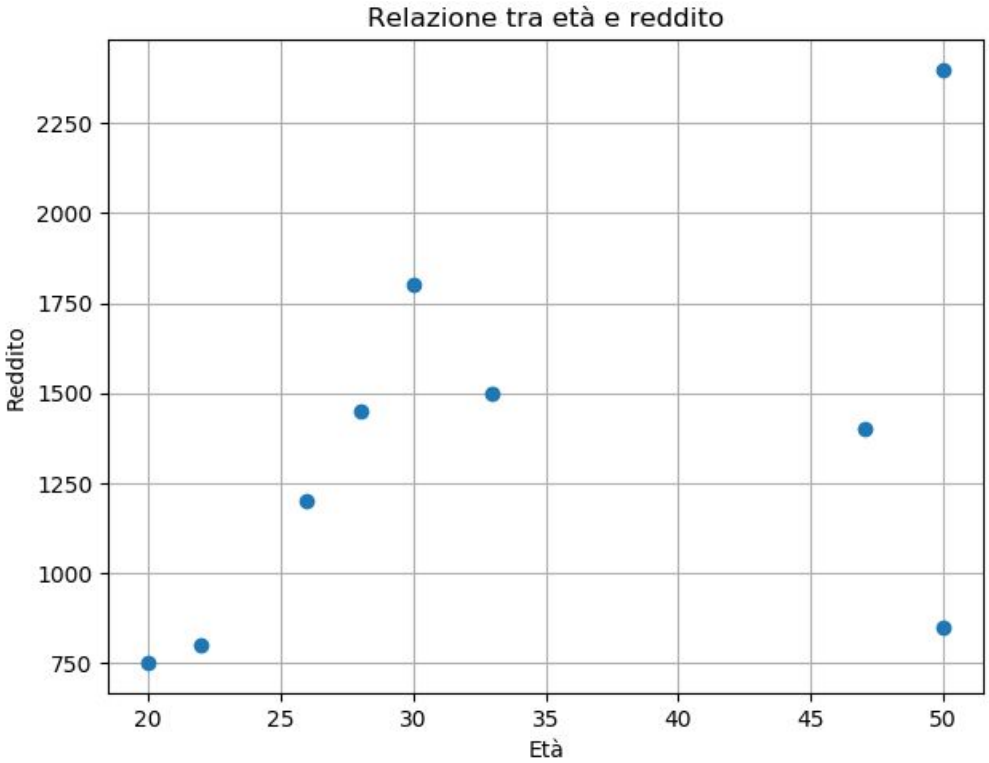
Esercizio sulla Creazione di Scatter Plot

```
import matplotlib.pyplot as plt
eta= [50, 30, 47, 28, 26, 22, 50, 20, 33]
reddito= [2400, 1800, 1400, 1450, 1200, 800, 850, 750, 1500 ]

plt.figure()
ax = plt.subplot()

plt.grid()
ax.set_axisbelow(True) # fa si che la griglia venga mostrata dietro i punti

plt.scatter(eta, reddito)

plt.title('Relazione tra età e reddito')
plt.xlabel('Età')
plt.ylabel('Reddito')
```

Esercizio sulla Creazione di Scatter Plot

```
plt.subplots_adjust(0.125, 0.1, 0.98, 0.95)

plt.savefig("grafico_reddito_eta.png")
plt.show()
plt.close()

from scipy.stats import pearsonr

p_coef, pval = pearsonr(eta, reddito)

print("Il coefficiente di correlazione di Pearson è: ", p_coef)
print("Il pvalue è : ", pval)
```

Esercizio sulla Creazione di Scatter Plot



Esercizio sulla Creazione di Scatter Plot

Il coefficiente di correlazione di Pearson è: 0.4539940367146339
Il pvalue è : 0.2196208129609393

Il pvalue è maggiore di 0.05 quindi la correlazione calcolata con il coefficiente di correlazione di Pearson non è statisticamente significativa.