

Corso di STATISTICA, INFORMATICA, ELABORAZIONE DELLE INFORMAZIONI

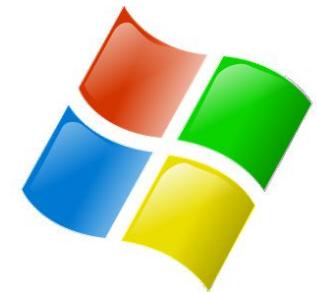
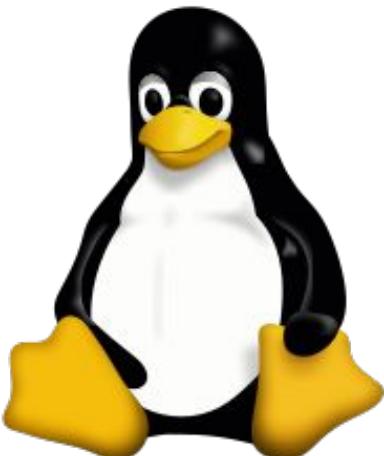
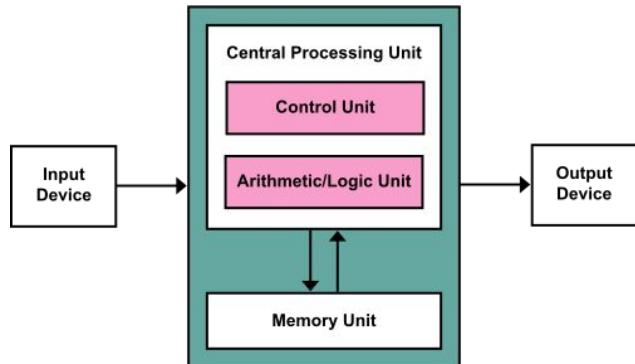
*Modulo di Sistemi di Elaborazione delle
Informazioni*



**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

Plotting

Docente:
**Domenico Daniele
Bloisi**



Domenico Daniele Bloisi

- Professore Associato

Dipartimento di Matematica, Informatica
ed Economia

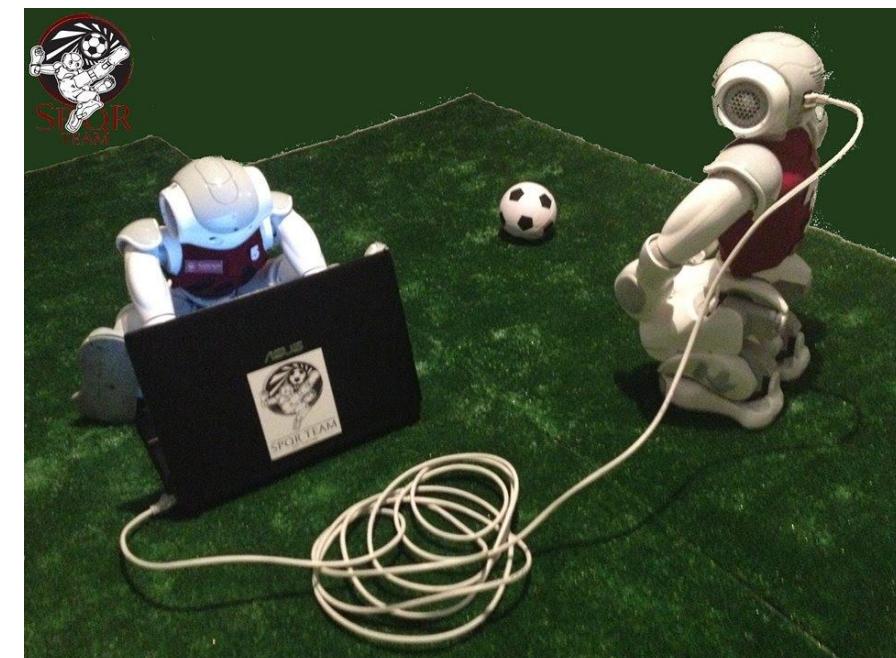
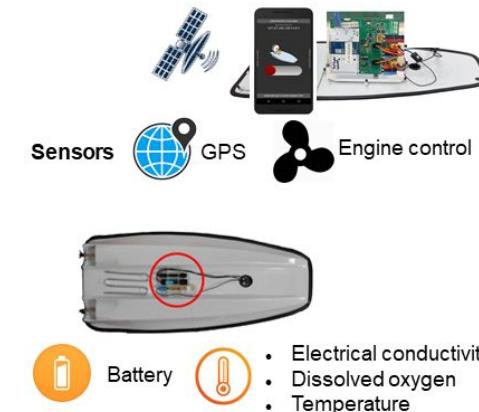
Università degli studi della Basilicata

<http://web.unibas.it/bloisi>

- SPQR Robot Soccer Team

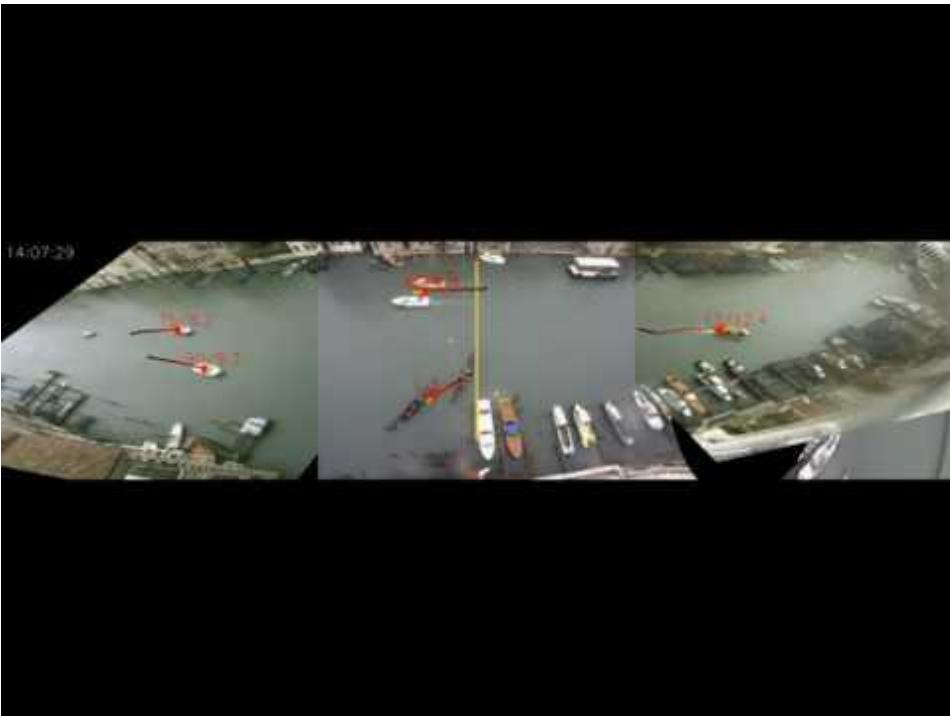
Dipartimento di Informatica, Automatica
e Gestionale Università degli studi di
Roma “La Sapienza”

<http://spqr.diag.uniroma1.it>



Interessi di ricerca

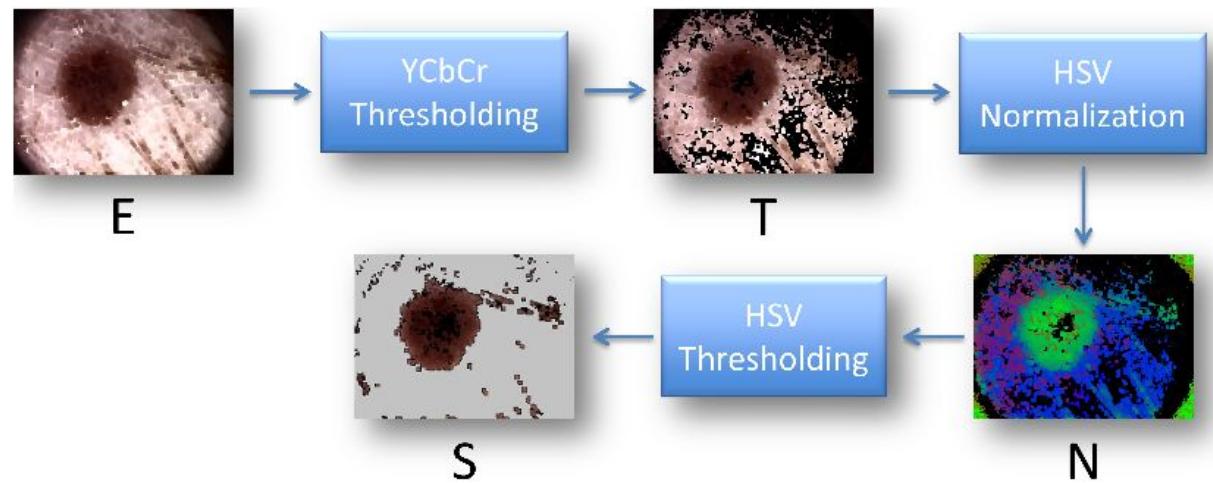
- Intelligent surveillance
- Robot vision
- Medical image analysis



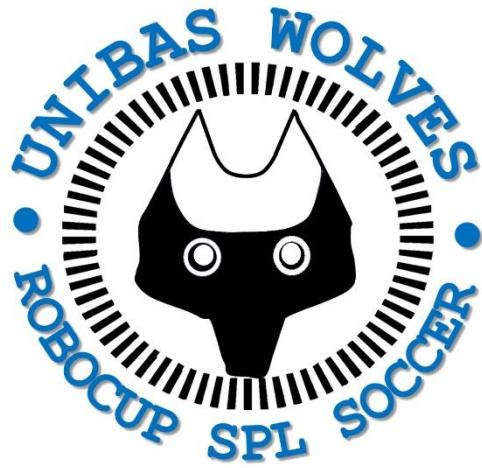
https://youtu.be/9a70Ucgbi_U



<https://youtu.be/2KHNZX7UIWQ>



UNIBAS Wolves <https://sites.google.com/unibas.it/wolves>



- UNIBAS WOLVES is the robot soccer team of the University of Basilicata. Established in 2019, it is focussed on developing software for NAO soccer robots participating in RoboCup competitions.
- UNIBAS WOLVES team is twinned with SPQR Team at Sapienza University of Rome



<https://youtu.be/ji0OmkaWh20>

Informazioni sul corso

Il corso di STATISTICA, INFORMATICA, ELABORAZIONE DELLE INFORMAZIONI

- include 3 moduli:
 - SISTEMI DI ELABORAZIONE DELLE INFORMAZIONI
(il martedì - docente: Domenico Bloisi)
 - INFORMATICA
(il mercoledì - docente: Enzo Veltri)
 - PROBABILITÀ E STATISTICA MATEMATICA
(il giovedì - docente: Antonella Iuliano)
- Periodo: I semestre ottobre 2022 – gennaio 2023

Ricevimento Bloisi

- In presenza, durante il periodo delle lezioni:
Lunedì dalle 17:00 alle 18:00
presso Edificio 3D, II piano, stanza 15
Si invitano gli studenti a controllare regolarmente la bacheca degli avvisi per eventuali variazioni
- Tramite google Meet e al di fuori del periodo delle lezioni:
da concordare con il docente tramite email

Per prenotare un appuntamento inviare
una email a
domenico.bloisi@unibas.it



Two-Dimensional Lists (1 of 3)

- Two-dimensional list: a list that contains other lists as its elements
 - Also known as nested list
 - Common to think of two-dimensional lists as having rows and columns
 - Useful for working with multiple sets of data
- To process data in a two-dimensional list need to use two indexes
- Typically use nested loops to process

```
students = [['Joe', 'Kim'], ['Sam', 'Sue'], ['Kelly', 'Kris']]  
print(students)
```

```
[['Joe', 'Kim'], ['Sam', 'Sue'], ['Kelly', 'Kris']]
```

```
print(students[1])
```

```
['Sam', 'Sue']
```

```
print(students[2][0])
```

```
Kelly
```

Two-Dimensional Lists (2 of 3)

	Column 0	Column 1
Row 0	'Joe'	'Kim'
Row 1	'Sam'	'Sue'
Row 2	'Kelly'	'Chris'

Figure 7-8 A two-dimensional list



```
# Questo programma mostra una lista bidimensionale.
def main():
    # Crea una lista bidimensionale.
    values = [[1, 2, 3],
              [10, 20, 30],
              [100, 200, 300]]
    # Visualizza gli elementi della lista.
    for row in values:
        for element in row:
            print(element)

# Chiama la funzione main.
if __name__ == '__main__':
    main()
```



```
1
2
3
10
20
30
100
200
300
```

Two-Dimensional Lists (3 of 3)

	Column 0	Column 1	Column 2
Row 0	scores[0][0]	scores[0][1]	scores[0][2]
Row 1	scores[1][0]	scores[1][1]	scores[1][2]
Row 2	scores[2][0]	scores[2][1]	scores[2][2]

Figure 7-10 Subscripts for each element of the scores list

Tuples (1 of 3)

- Tuple: an immutable sequence
 - Very similar to a list
 - Once it is created it cannot be changed
 - Format: `tuple_name = (item1, item2)`
 - Tuples support operations as lists
 - Subscript indexing for retrieving elements
 - Methods such as `index`
 - Built in functions such as `len`, `min`, `max`
 - Slicing expressions
 - The `in`, `+`, and `*` operators

```
[10] my_tuple = (1, 2, 3, 4, 5)
```

```
    print(my_tuple)
```

```
(1, 2, 3, 4, 5)
```

```
[11] print(my_tuple[2])
```

```
3
```



```
my_tuple[2] = 12
```



```
-----  
TypeError
```

```
Traceback (most recent call last)
```

```
<ipython-input-12-c823aadd157b> in <module>
```

```
----> 1 my_tuple[2] = 12
```

```
TypeError: 'tuple' object does not support item assignment
```

SEARCH STACK OVERFLOW

Tuples (2 of 3)

- Tuples do not support the methods:

- append
- remove
- insert
- reverse
- sort

ATTENZIONE



```
tupla = (1,) #tupla con un solo elemento  
valore = (1) #variabile con valore 1
```

```
print(tupla)  
print(valore)
```

```
↳ (1,  
 1
```

Tuples (3 of 3)

- Advantages for using tuples over lists:
 - Processing tuples is faster than processing lists
 - Tuples are safe
 - Some operations in Python require use of tuples
- list () function: converts tuple to list
- tuple () function: converts list to tuple

Matplotlib

Matplotlib è una libreria Python per il plotting in 2D



<https://matplotlib.org/>

Con matplotlib è possibile generare grafici, istogrammi, spettri, diagrammi a barre, grafici di dispersione e altro ancora usando una interfaccia tipo MATLAB

Plotting Data with matplotlib (1 of 4)

- The `matplotlib` package is a library for creating two-dimensional charts and graphs.
- It is not part of the standard Python library, so you will have to install it separately, after you have installed Python on your system.

Plotting Data with matplotlib (2 of 4)

- To install matplotlib on a Windows system, open a Command Prompt window and enter this command:

```
pip install matplotlib
```

- To install matplotlib on a Mac or Linux system, open a Terminal window and enter this command:

```
sudo pip3 install matplotlib
```

- See Appendix F in your textbook for more information about packages and the pip utility.

Plotting Data with matplotlib (3 of 4)

- To verify the package was installed, start IDLE and enter this command:

```
>>> import matplotlib
```

- If you don't see any error messages, you can assume the package was properly installed.

matplotlib in Colab

```
✓ 0 s ⏴ import matplotlib  
    matplotlib.__version__  
    '3.2.2'
```

Plotting Data with matplotlib (4 of 4)

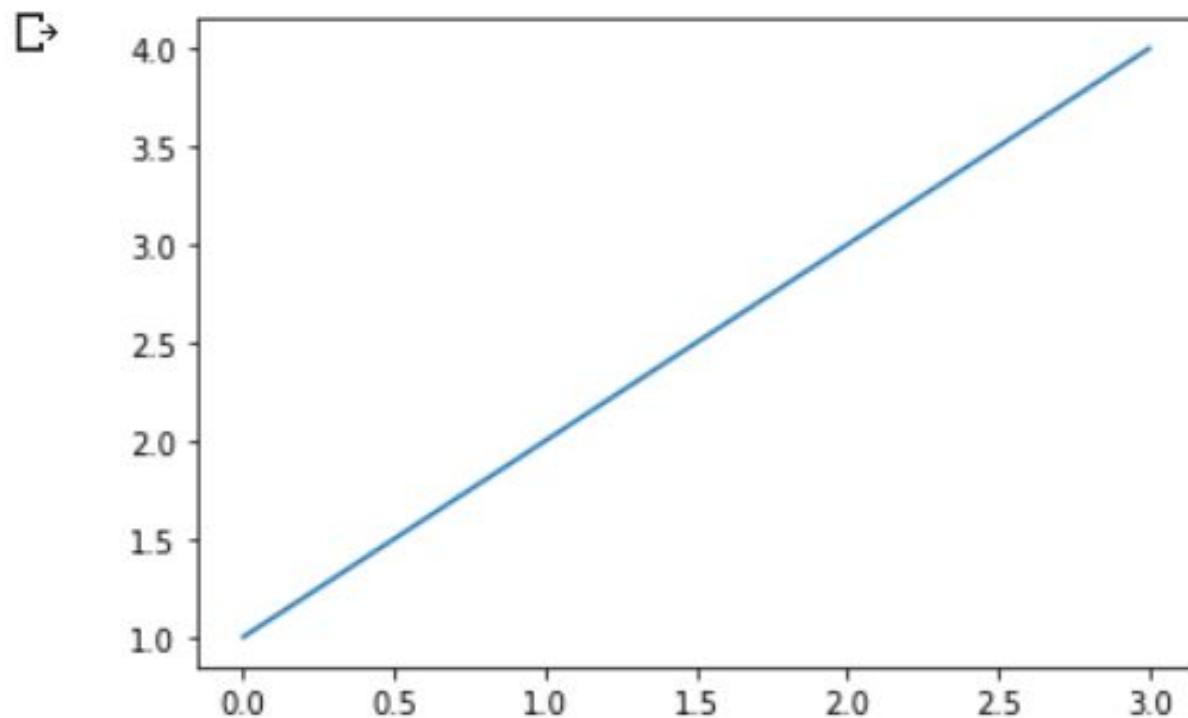
- The `matplotlib` package contains a module named `pyplot` that you will need to import.
- Use the following `import` statement to import the module and create an alias named `plt`:

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

For more information about the `import` statement, see Appendix E in your textbook.

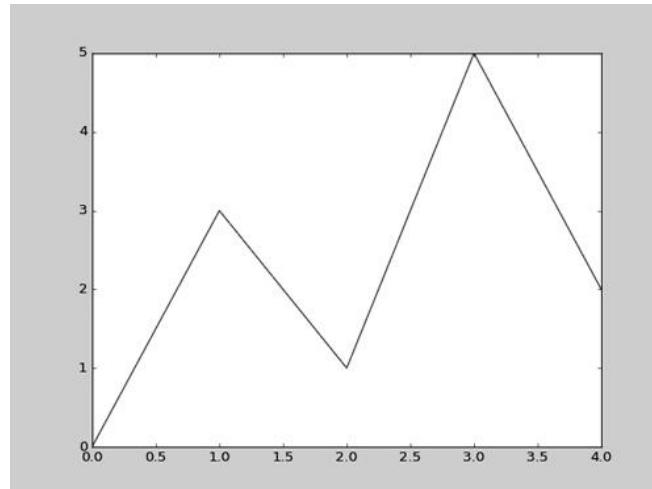
✓
0 s

```
import matplotlib.pyplot as plt  
  
plt.plot([1,2,3,4])  
  
plt.show()
```



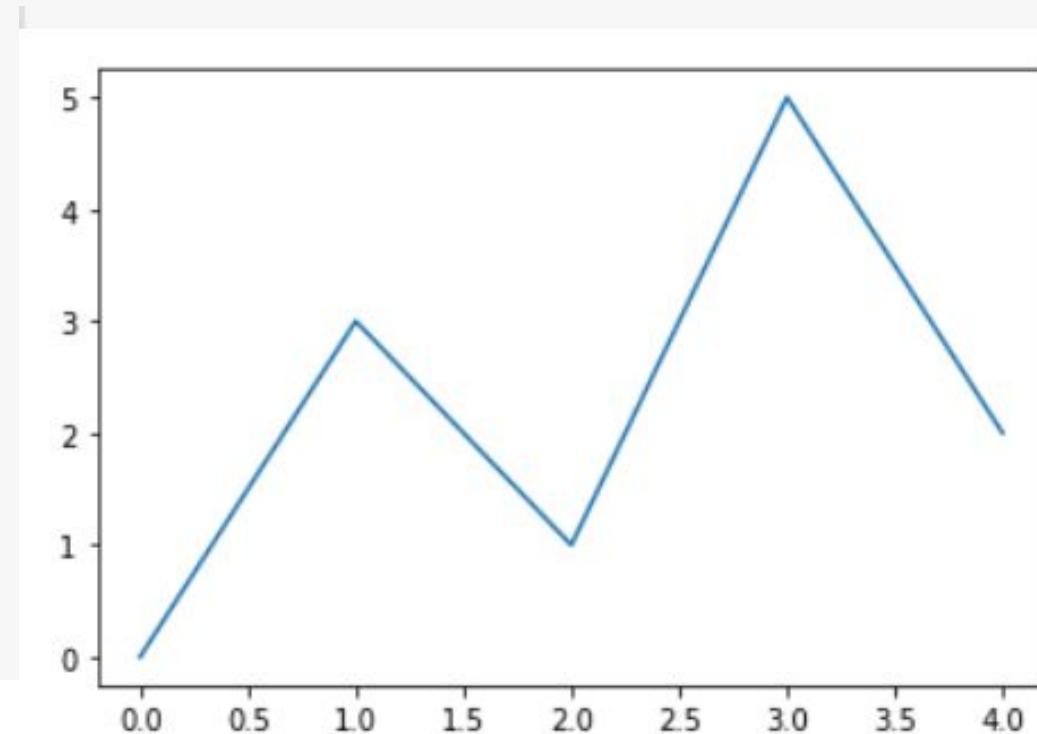
Plotting a Line Graph with the `plot` Function (1 of 4)

- Use the `plot` function to create a line graph that connects a series of points with straight lines.
- The line graph has a horizontal X axis, and a vertical Y axis.
- Each point in the graph is located at a (X, Y) coordinate.

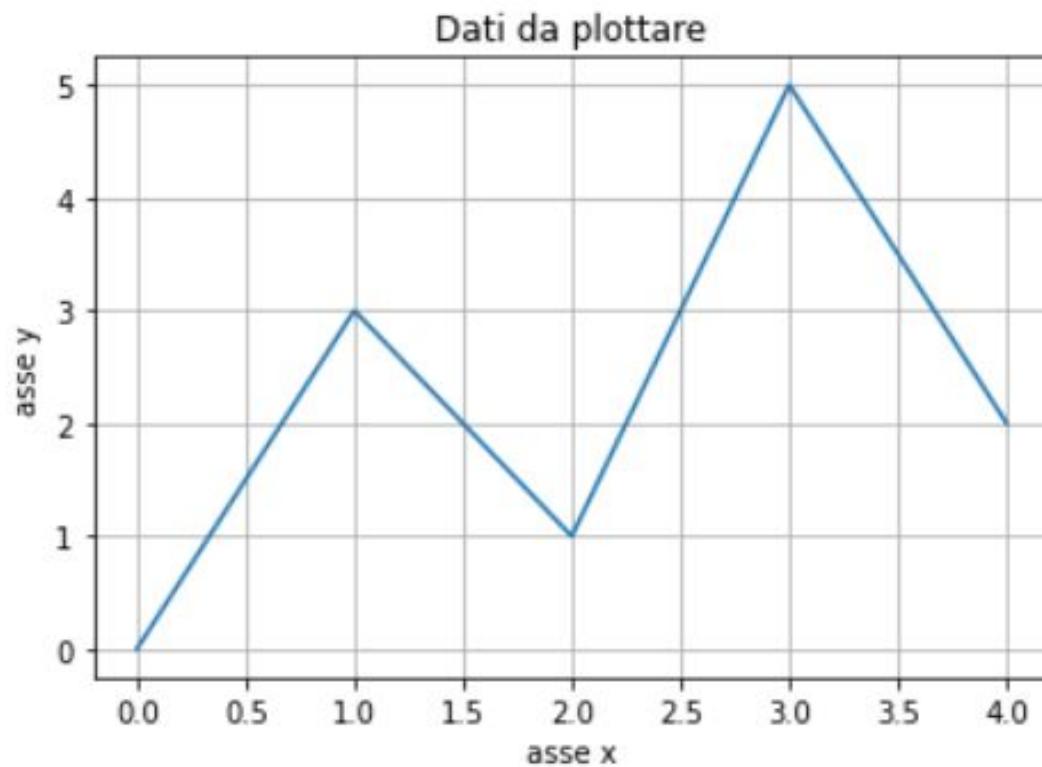




```
# Questo programma visualizza un semplice grafico a linea.  
import matplotlib.pyplot as plt  
  
def main():  
    # Crea liste con le coordinate X e Y di ogni punto dati.  
    x_coords = [0, 1, 2, 3, 4]  
    y_coords = [0, 3, 1, 5, 2]  
  
    # Costruisce il grafico a linea.  
    plt.plot(x_coords, y_coords)  
  
    # Visualizza il grafico a linea.  
    plt.show()  
  
# Chiama la funzione main.  
if __name__ == '__main__':  
    main()
```



```
# Questo programma visualizza un semplice grafico a linea.  
import matplotlib.pyplot as plt  
  
def main():  
    # Crea liste con le coordinate X e Y di ogni punto dati.  
    x_coords = [0, 1, 2, 3, 4]  
    y_coords = [0, 3, 1, 5, 2]  
  
    # Costruisce il grafico a linea.  
    plt.plot(x_coords, y_coords)  
  
    plt.title("Dati da plottare")  
    plt.xlabel("asse x")  
    plt.ylabel("asse y")  
    plt.grid(True)  
  
    # Visualizza il grafico a linea.  
    plt.show()  
  
# Chiama la funzione main.  
if __name__ == '__main__':  
    main()
```



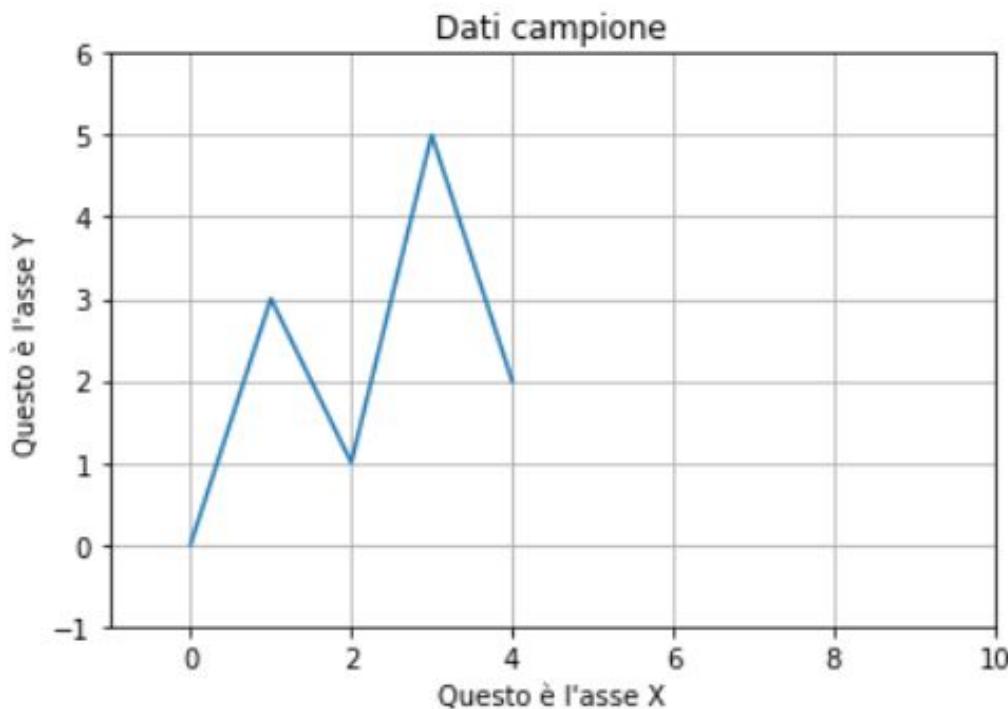
Plotting a Line Graph with the `plot` Function (3 of 4)

- You can change the lower and upper limits of the *X* and *Y* axes by calling the `xlim` and `ylim` functions. Example:

```
plt.xlim(xmin=1, xmax=100)  
plt.ylim(ymin=10, ymax=50)
```

- This code does the following:
 - Causes the *X* axis to begin at 1 and end at 100
 - Causes the *Y* axis to begin at 10 and end at 50

```
# Questo programma visualizza un semplice grafico a linea.  
import matplotlib.pyplot as plt  
  
def main():  
    # Crea liste con le coordinate X e Y di ogni punto dati.  
    x_coords = [0, 1, 2, 3, 4]  
    y_coords = [0, 3, 1, 5, 2]  
    # Costruisce il grafico a linea.  
    plt.plot(x_coords, y_coords)  
    # Aggiunge un titolo.  
    plt.title('Dati campione')  
    # Aggiunge etichette agli assi.  
    plt.xlabel("Questo è l'asse X")  
    plt.ylabel("Questo è l'asse Y")  
  
    # Imposta i limiti degli assi.  
    plt.xlim(xmin=-1, xmax=10)  
    plt.ylim(ymin=-1, ymax=6)  
  
    # Aggiunge una griglia.  
    plt.grid(True)  
    # Visualizza il grafico a linea.  
    plt.show()  
  
# Chiama la funzione main.  
if __name__ == '__main__':  
    main()
```



Plotting a Line Graph with the `plot` Function (4 of 4)

- You can customize each tick mark's label with the `xticks` and `yticks` functions.
- These functions each take two lists as arguments.
 - The first argument is a list of tick mark locations
 - The second argument is a list of labels to display at the specified locations.

```
plt.xticks([0, 1, 2, 3, 4],  
          ['2016', '2017', '2018', '2019', '2020'])  
  
plt.yticks([0, 1, 2, 3, 4, 5],  
          ['$0m', '$1m', '$2m', '$3m', '$4m', '$5m'])
```

```
import matplotlib.pyplot as plt

def main():
    # Crea liste con le coordinate X e Y di ogni punto dati.
    x_coords = [0, 1, 2, 3, 4]
    y_coords = [0, 3, 1, 5, 2]

    # Costruisce il grafico a linea.
    plt.plot(x_coords, y_coords)

    # Aggiunge un titolo.
    plt.title('Vendite per anno')

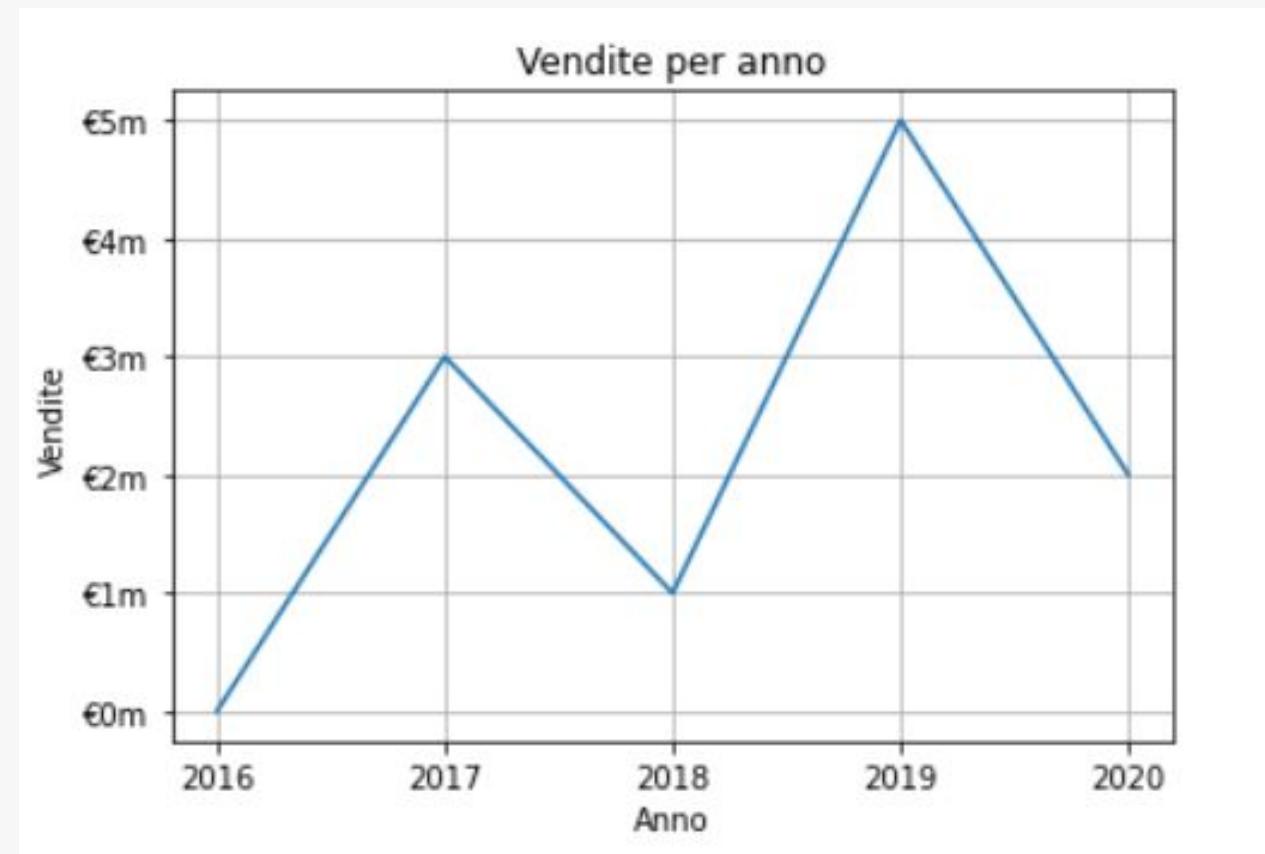
    # Aggiunge etichette agli assi.
    plt.xlabel('Anno')
    plt.ylabel('Vendite')

    # Personalizza le tacche.
    plt.xticks([0, 1, 2, 3, 4],
               ['2016', '2017', '2018', '2019', '2020'])
    plt.yticks([0, 1, 2, 3, 4, 5],
               ['€0m', '€1m', '€2m', '€3m', '€4m', '€5m'])

    # Aggiunge una griglia.
    plt.grid(True)

    # Visualizza il grafico a linea.
    plt.show()

# Chiama la funzione main.
if __name__ == '__main__':
    main()
```



Output of Program 7-24

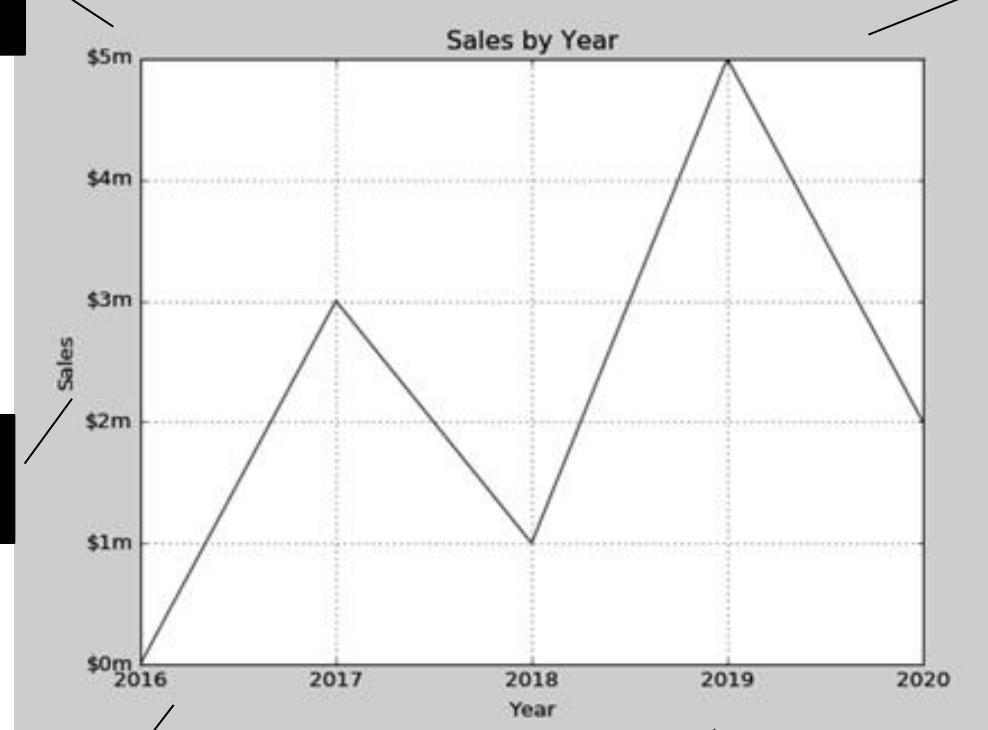
Displayed by the `yticks()` function.

Displayed by the `title()` function.

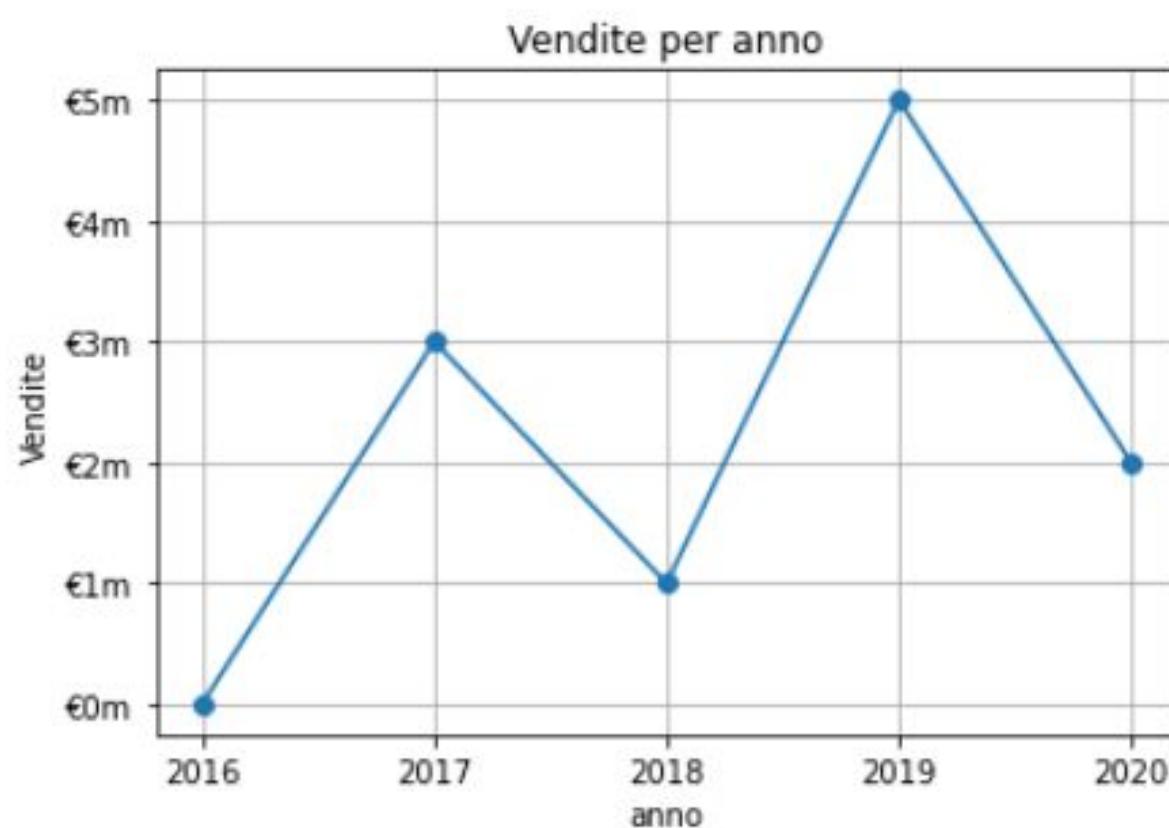
Displayed by the `ylabel()` function.

Displayed by the `xticks()` function.

Displayed by the `xlabel()` function.



```
# Questo programma visualizza un semplice grafico a linea.  
import matplotlib.pyplot as plt  
  
def main():  
    # Crea liste con le coordinate X e Y di ogni punto dati.  
    x_coords = [0, 1, 2, 3, 4]  
    y_coords = [0, 3, 1, 5, 2]  
  
    # Costruisce il grafico a linea.  
    plt.plot(x_coords, y_coords, marker='o')  
  
    # Aggiunge un titolo.  
    plt.title('Vendite per anno')  
  
    # Aggiunge etichette agli assi.  
    plt.xlabel('anno')  
    plt.ylabel('Vendite')  
  
    # Personalizza le tacche.  
    plt.xticks([0, 1, 2, 3, 4],  
              ['2016', '2017', '2018', '2019', '2020'])  
    plt.yticks([0, 1, 2, 3, 4, 5],  
              ['€0m', '€1m', '€2m', '€3m', '€4m', '€5m'])  
  
    # Aggiunge una griglia.  
    plt.grid(True)  
  
    # Visualizza il grafico a linea.  
    plt.show()  
  
# Chiama la funzione main.  
if __name__ == '__main__':  
    main()
```



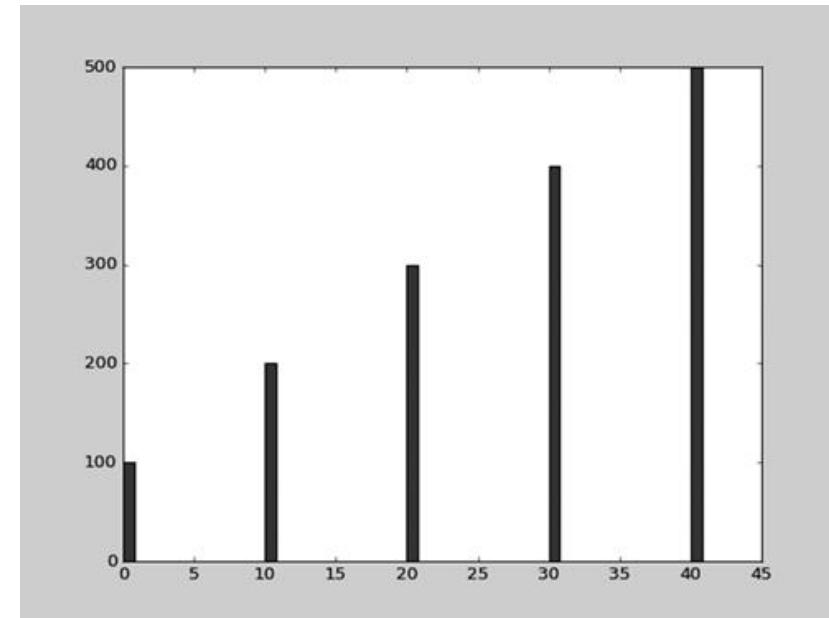
Plotting a Bar Chart (1 of 6)

- Use the `bar` function in the `matplotlib.pyplot` module to create a bar chart.
- The function needs two lists: one with the X coordinates of each bar's left edge, and another with the heights of each bar, along the Y axis.

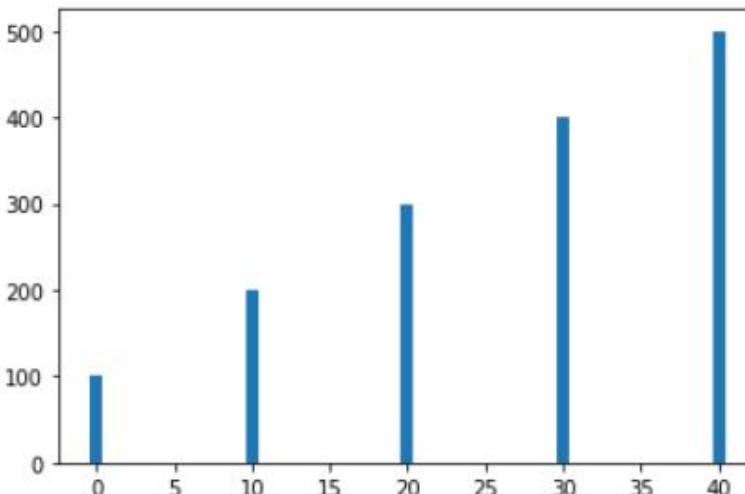
Plotting a Bar Chart (2 of 6)

```
left_edges = [0, 10, 20, 30, 40]  
heights = [100, 200, 300, 400, 500]
```

```
plt.bar(left_edges, heights)  
plt.show()
```



```
# Questo programma visualizza un semplice grafico a barre.  
import matplotlib.pyplot as plt  
  
def main():  
    # Crea una lista con le coordinate X del bordo sinistro di ogni barra.  
    left_edges = [0, 10, 20, 30, 40]  
  
    # Crea una lista con l'altezza di ogni barra.  
    heights = [100, 200, 300, 400, 500]  
  
    # Costruisce il grafico a barre.  
    plt.bar(left_edges, heights)  
  
    # Visualizza il grafico a barre.  
    plt.show()  
  
# Chiama la funzione main.  
if __name__ == '__main__':  
    main()
```

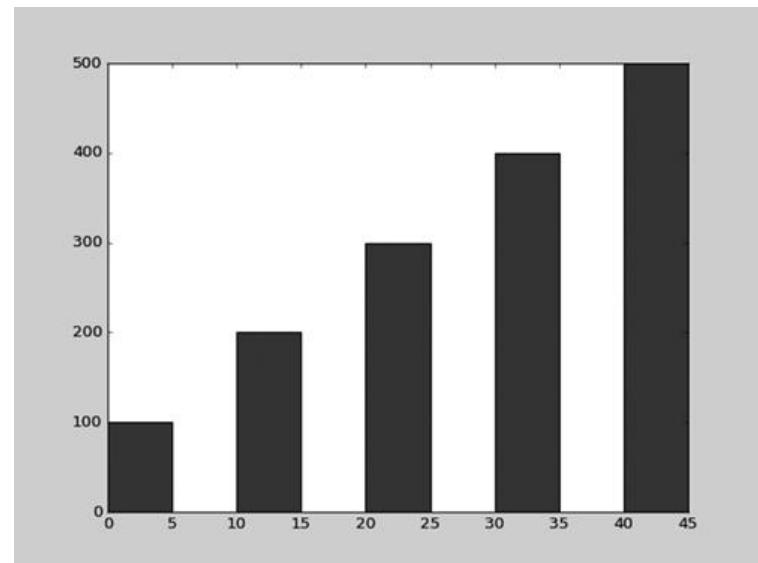


Plotting a Bar Chart (3 of 6)

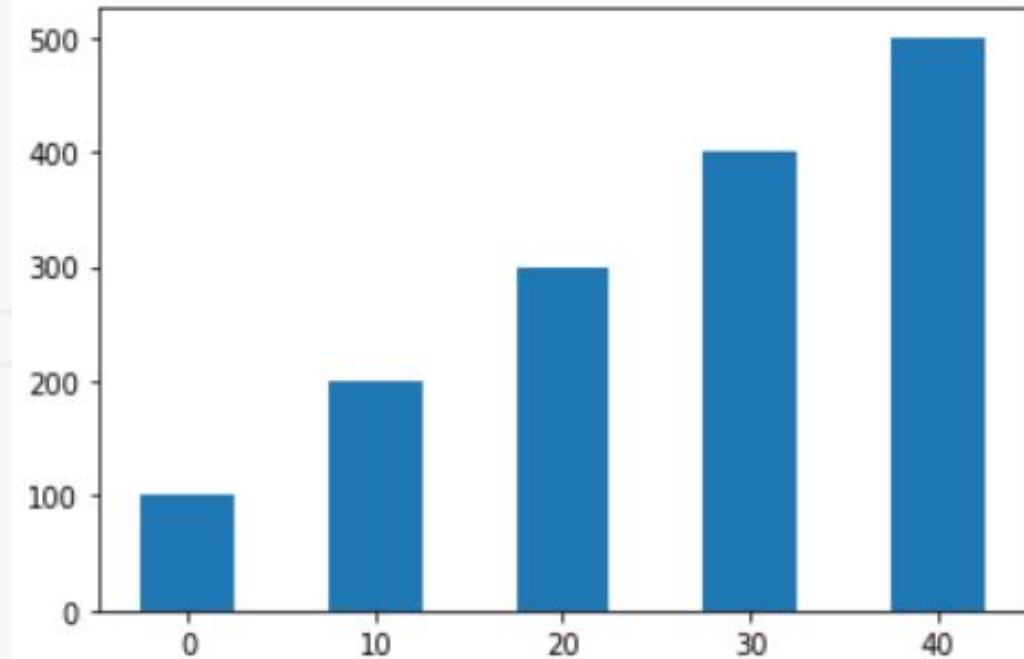
- The default width of each bar in a bar graph is 0.8 along the X axis.
- You can change the bar width by passing a third argument to the `bar` function.

```
left_edges = [0, 10, 20, 30, 40]
heights = [100, 200, 300, 400, 500]
bar_width = 5

plt.bar(left_edges, heights, bar_width)
plt.show()
```



```
# Questo programma visualizza un semplice grafico a barre.  
import matplotlib.pyplot as plt  
  
def main():  
    # Crea una lista con le coordinate X del bordo sinistro di ogni barra.  
    left_edges = [0, 10, 20, 30, 40]  
  
    # Crea una lista con l'altezza di ogni barra.  
    heights = [100, 200, 300, 400, 500]  
  
    # Crea una variabile per lo spessore delle barre.  
    bar_width = 5  
  
    # Costruisce il grafico a barre.  
    plt.bar(left_edges, heights, bar_width)  
  
    # Visualizza il grafico a barre.  
    plt.show()  
  
# Chiama la funzione main.  
if __name__ == '__main__':  
    main()
```



Plotting a Bar Chart (4 of 6)

- The `bar` function has a `color` parameter that you can use to change the colors of the bars.
- The argument that you pass into this parameter is a tuple containing a series of color codes.

Color Code	Corresponding Color
'b'	Blue
'g'	Green
'r'	Red
'c'	Cyan
'm'	Magenta
'y'	Yellow
'k'	Black
'w'	White

Plotting a Bar Chart (5 of 6)

- Example of how to pass a tuple of color codes as a keyword argument:

```
plt.bar(left_edges, heights, color=('r', 'g', 'b', 'w', 'k'))
```

- The colors of the bars in the resulting bar chart will be as follows:
 - The first bar will be red.
 - The second bar will be green.
 - The third bar will be blue.
 - The fourth bar will be white.
 - The fifth bar will be black.

Plotting a Bar Chart (6 of 6)

- Use the `xlabel` and `ylabel` functions to add labels to the `X` and `Y` axes.
- Use the `xticks` function to display custom tick mark labels along the `X` axis
- Use the `yticks` function to display custom tick mark labels along the `Y` axis.

```

# Questo programma plotta un grafico delle vendite.
import matplotlib.pyplot as plt

def main():
    # Crea una lista con le coordinate X del bordo sinistro di ogni barra.
    left_edges = [0, 10, 20, 30, 40]

    # Crea una lista con l'altezza di ogni barra.
    heights = [100, 200, 300, 400, 500]

    # Crea una variabile per lo spessore delle barre.
    bar_width = 10

    # Costruisce il grafico a barre.
    plt.bar(left_edges, heights, bar_width, color=('r', 'g', 'b', 'm', 'k'))

    # Aggiunge un titolo.
    plt.title('Vendite per anno')

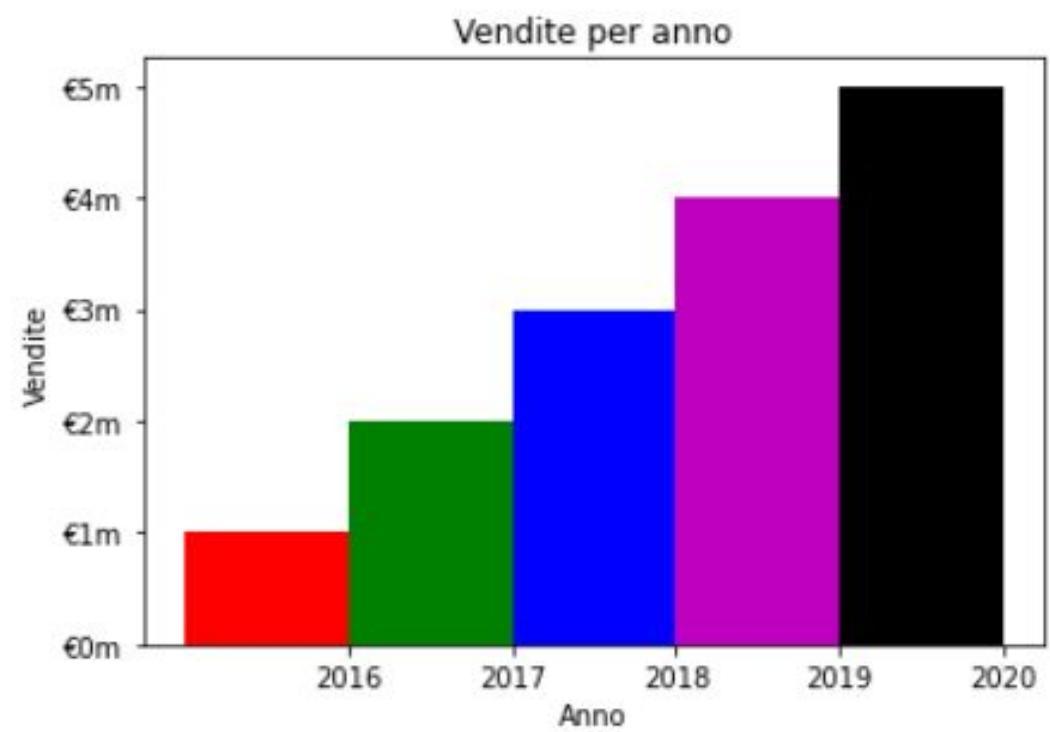
    # Aggiunge etichette agli assi.
    plt.xlabel('Anno')
    plt.ylabel('Vendite')

    # Personalizza le tacche.
    plt.xticks([5, 15, 25, 35, 45],
               ['2016', '2017', '2018', '2019', '2020'])
    plt.yticks([0, 100, 200, 300, 400, 500],
               ['€0m', '€1m', '€2m', '€3m', '€4m', '€5m'])

    # Visualizza il grafico a barre.
    plt.show()

# Chiama la funzione main.
if __name__ == '__main__':
    main()

```



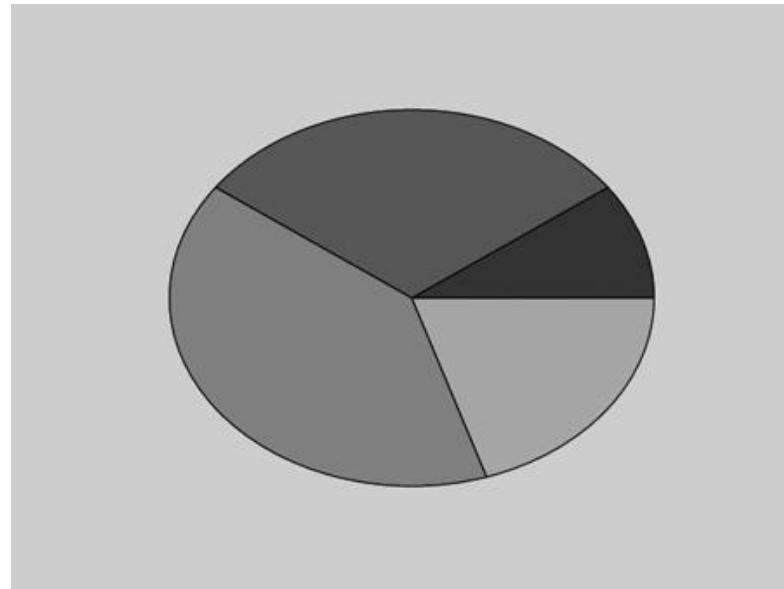
Plotting a Pie Chart (1 of 4)

- You use the `pie` function in the `matplotlib.pyplot` module to create a pie chart.
- When you call the `pie` function, you pass a list of values as an argument.
 - The sum of the values will be used as the value of the whole.
 - Each element in the list will become a slice in the pie chart.
 - The size of a slice represents that element's value as a percentage of the whole.

Plotting a Pie Chart (2 of 4)

- Example

```
values = [20, 60, 80, 40]  
plt.pie(values)  
plt.show()
```



```
# Questo programma visualizza un semplice grafico a torta
import matplotlib.pyplot as plt

def main():
    # Crea una lista di valori
    values = [20, 60, 80, 40]

    # Crea un grafico a torta partendo dai valori.
    plt.pie(values)

    # Visualizza il grafico a torta.
    plt.show()

# Chiama la funzione main.
if __name__ == '__main__':
    main()
```



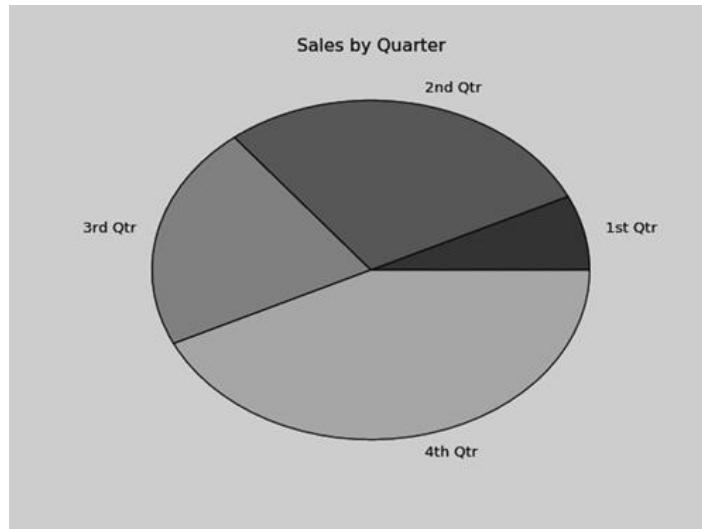
Plotting a Pie Chart (3 of 4)

- The `pie` function has a `labels` parameter that you can use to display labels for the slices in the pie chart.
- The argument that you pass into this parameter is a list containing the desired labels, as strings.

Plotting a Pie Chart (4 of 4)

- Example

```
sales = [100, 400, 300, 600]
slice_labels = ['1st Qtr', '2nd Qtr', '3rd Qtr', '4th Qtr']
plt.pie(sales, labels=slice_labels)
plt.title('Sales by Quarter')
plt.show()
```



```
# Questo programma visualizza un semplice grafico a torta
import matplotlib.pyplot as plt

def main():
    # Crea un elenco di valori delle vendite.
    sales = [100, 400, 300, 600]

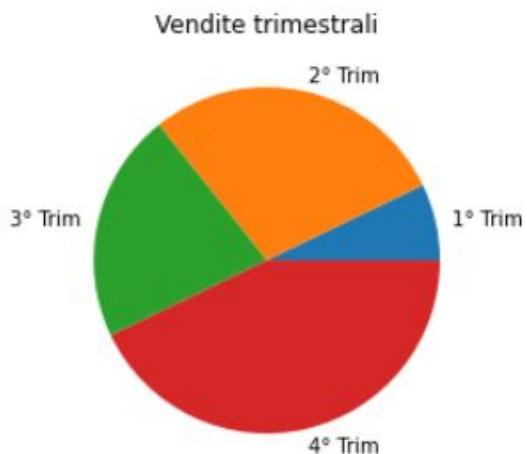
    # Crea una lista di etichette per le fette.
    slice_labels = ['1° Trim', '2° Trim', '3° Trim', '4° Trim']

    # Crea un grafico a torta partendo dai valori.
    plt.pie(sales, labels=slice_labels)

    # Aggiunge un titolo.
    plt.title('Vendite trimestrali')

    # Visualizza il grafico a torta.
    plt.show()

# Chiama la funzione main.
if __name__ == '__main__':
    main()
```



Plotting a Pie Chart

- The `pie` function automatically changes the color of the slices, in the following order:
 - blue, green, red, cyan, magenta, yellow, black, and white.
- You can specify a different set of colors, however, by passing a tuple of color codes as an argument to the `pie` **function's colors parameter**:

```
plt.pie(values, colors=('r', 'g', 'b', 'w', 'k'))
```
- When this statement executes, the colors of the slices in the resulting pie chart will be red, green, blue, white, and black.

```
# Questo programma visualizza un semplice grafico a torta
import matplotlib.pyplot as plt

def main():
    # Crea un elenco di valori delle vendite.
    sales = [100, 400, 300, 600]

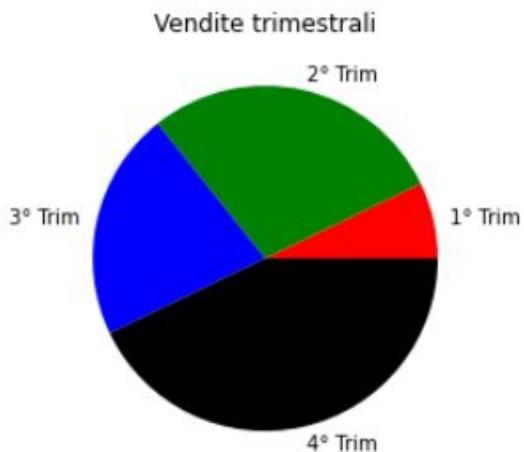
    # Crea una lista di etichette per le fette.
    slice_labels = ['1° Trim', '2° Trim', '3° Trim', '4° Trim']

    # Crea un grafico a torta partendo dai valori.
    plt.pie(sales, labels=slice_labels, colors=('r', 'g', 'b', 'k'))

    # Aggiunge un titolo.
    plt.title('Vendite trimestrali')

    # Visualizza il grafico a torta.
    plt.show()

# Chiama la funzione main.
if __name__ == '__main__':
    main()
```



Corso di STATISTICA, INFORMATICA, ELABORAZIONE DELLE INFORMAZIONI

*Modulo di Sistemi di Elaborazione delle
Informazioni*



**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

Plotting

Docente:
**Domenico Daniele
Bloisi**

