



POLITECNICO
MILANO 1863



SLIDES
DURANTE IL
LABORATORIO

Fondamenti di Comunicazioni e Internet

Antonio Capone, Matteo Cesana, Guido
Maier, Francesco Musumeci



POLITECNICO
MILANO 1863



Lab 1: Scripting con Python

**Antonio Capone, Matteo Cesana, Guido
Maier, Francesco Musumeci**

Programma del laboratorio

1) Programmazione Python

- Operazioni base con HTTP
- Socket programming (TCP e UDP)

2) Cisco Packet Tracer

- Configurazione di switch/routers
- Protocolli di routing interno (RIP e OSPF)

Materiale didattico

Dashboard - I miei corsi - 054303 - FONDAMENTI DI COMUNI... 765752 - Materiali - MATERIALE DIDATTICO 2021-2022

054303 - FONDAMENTI DI COMUNICAZIONI E INTERNET (MAIER GUIDO ALBERTO)

MATERIALE DIDATTICO 2021-2022

Materiale dell'anno accademico in corso.



Esercitazioni



Laboratorio



Lab 1



Pre-Lab Video Introduttivi.pdf



Setup_Lab_Python.pdf



Lezioni



1a-Introduzione_P1_v05.pdf



1b-Introduzione_P2_v05.pdf

Struttura delle lezioni

1) Pre-Lab

- Slide condivise in doveroso anticipo
- Materiale da studiare PRIMA della lezione

2) Durante-Lab

- Mix teoria+esempi/esercizi
- **Bisogna essere PC-muniti**

3) Post-Lab

- “Compito a casa” introdotto a fine lezione
- **Lo scopo è di autovalutazione:** scrivete a me o al tutor se vi bloccate!
- La soluzione sarà pubblicata in qualche giorno

Obiettivi della lezione

1) Capire intuitivamente:

- cosa sia un **server HTTP**
- cosa sia una **HTTP GET request**

2) Misurare il **ritardo** a livello applicativo

3) Automatizzare e visualizzare alcune misure

- HTTP verrà spiegato nel dettaglio dal prof. Maier nelle lezioni future!

Informazioni: Cisco CCNA

- Gli studenti di FCI possono iscriversi al Cisco CCNA
- Il corso rilascia una certificazione ufficiale Cisco
- Iscrizione e maggiori info:
<https://www.netacad.com/portal/web/self-enroll/m/course-1149792>
- Corso di introduzione a Cisco Packet Tracer
- Si può scaricare il software per la seconda metà del Lab nel caso non possiate/vogliate utilizzare la VM
- Maggiori info: <https://www.netacad.com/portal/web/self-enroll/m/course-1149792>

Attività di laboratorio: Versioni software



Gli esempi mostrati a lezione utilizzano:

- Python versione 3.9
- PyCharm IDE

Sono entrambi pre-installati nella macchina virtuale

In alternativa potete installare Python e PyCharm sul vostro computer (+ librerie requests e matplotlib)

Tempo di risposta di un server HTTP

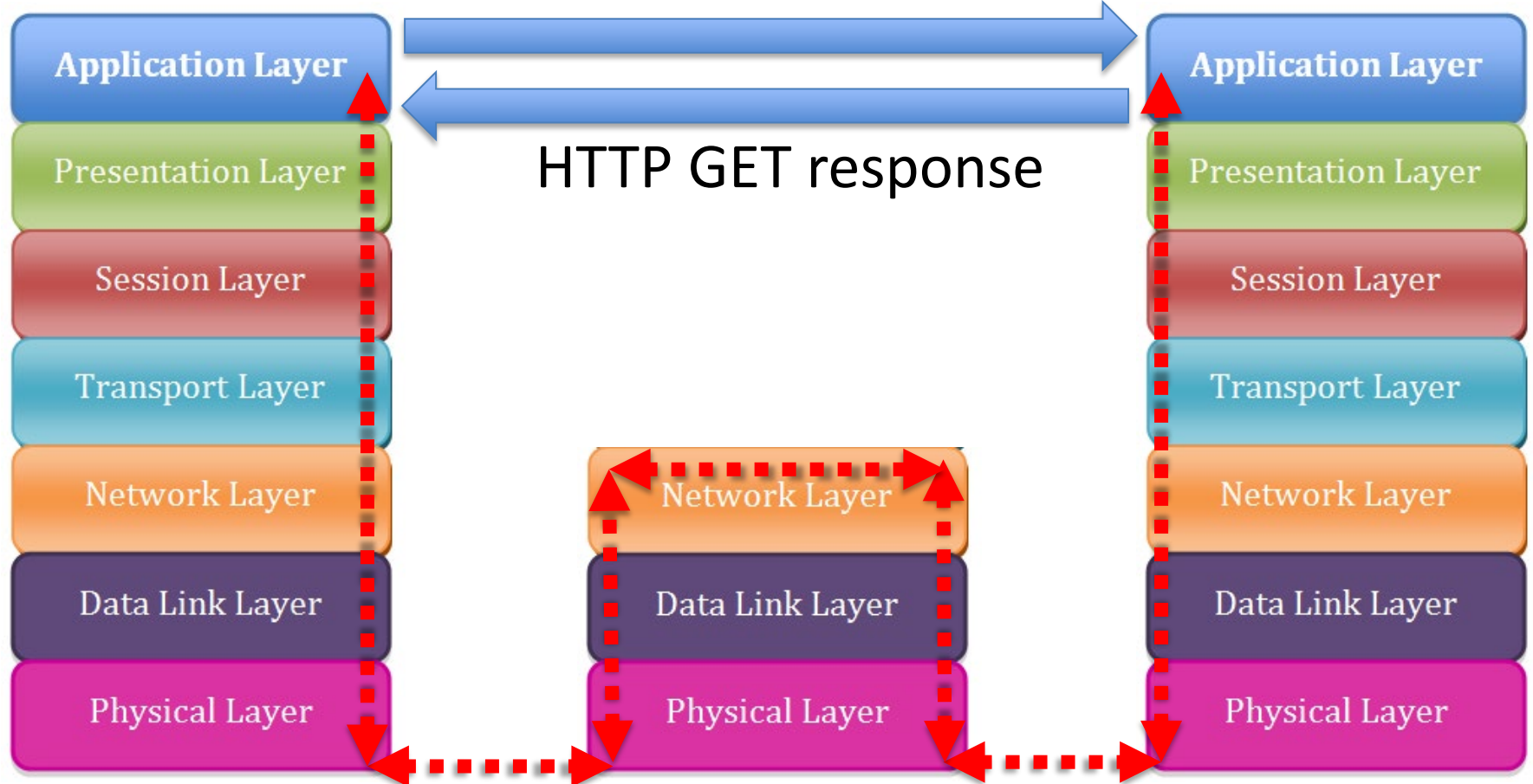
- Valutare il tempo di risposta del server di Google per scaricare l'home page

Modulo requests

```
1 import requests
2
3 r = requests.get('http://www.google.com')
4 print('Tempo di Risposta:', r.elapsed.microseconds / 1000, 'ms')
```

Cosa sta succedendo?

HTTP GET request
“Dammi il contenuto della pagina web”



Il nostro PC

Google Server

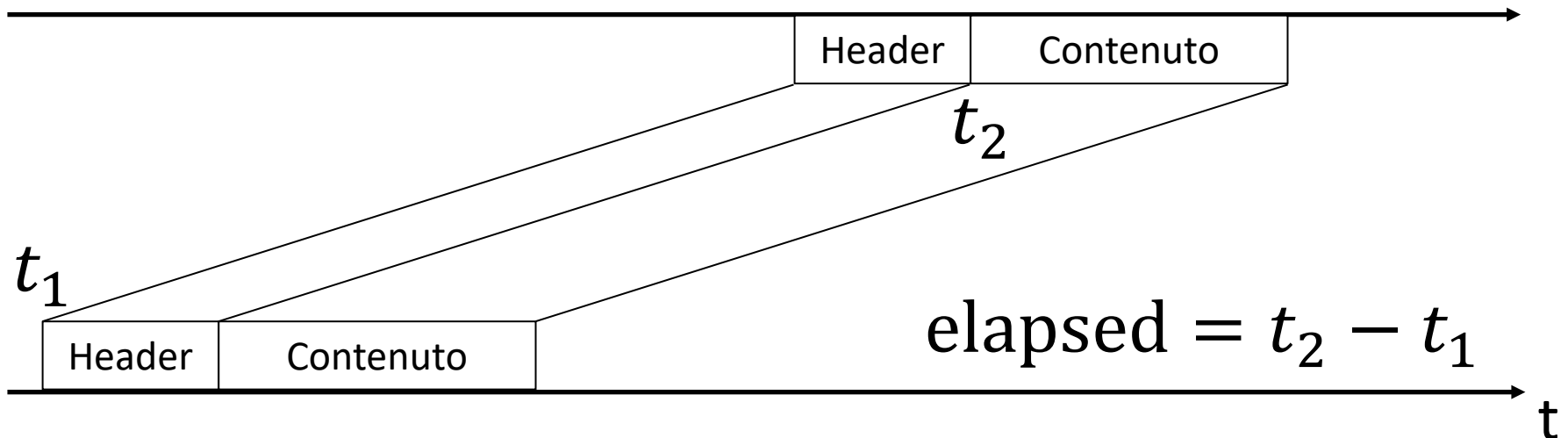
Cosa stiamo misurando?

- Al livello applicativo non è scontato definire il ritardo
- Nel dubbio, riferiamoci alla documentazione!

elapsed

The amount of time elapsed between sending the request and the arrival of the response (as a `timedelta`). This property specifically measures the time taken between sending the first byte of the request and finishing parsing the headers. It is therefore unaffected by consuming the response content or the value of the `stream` keyword argument.

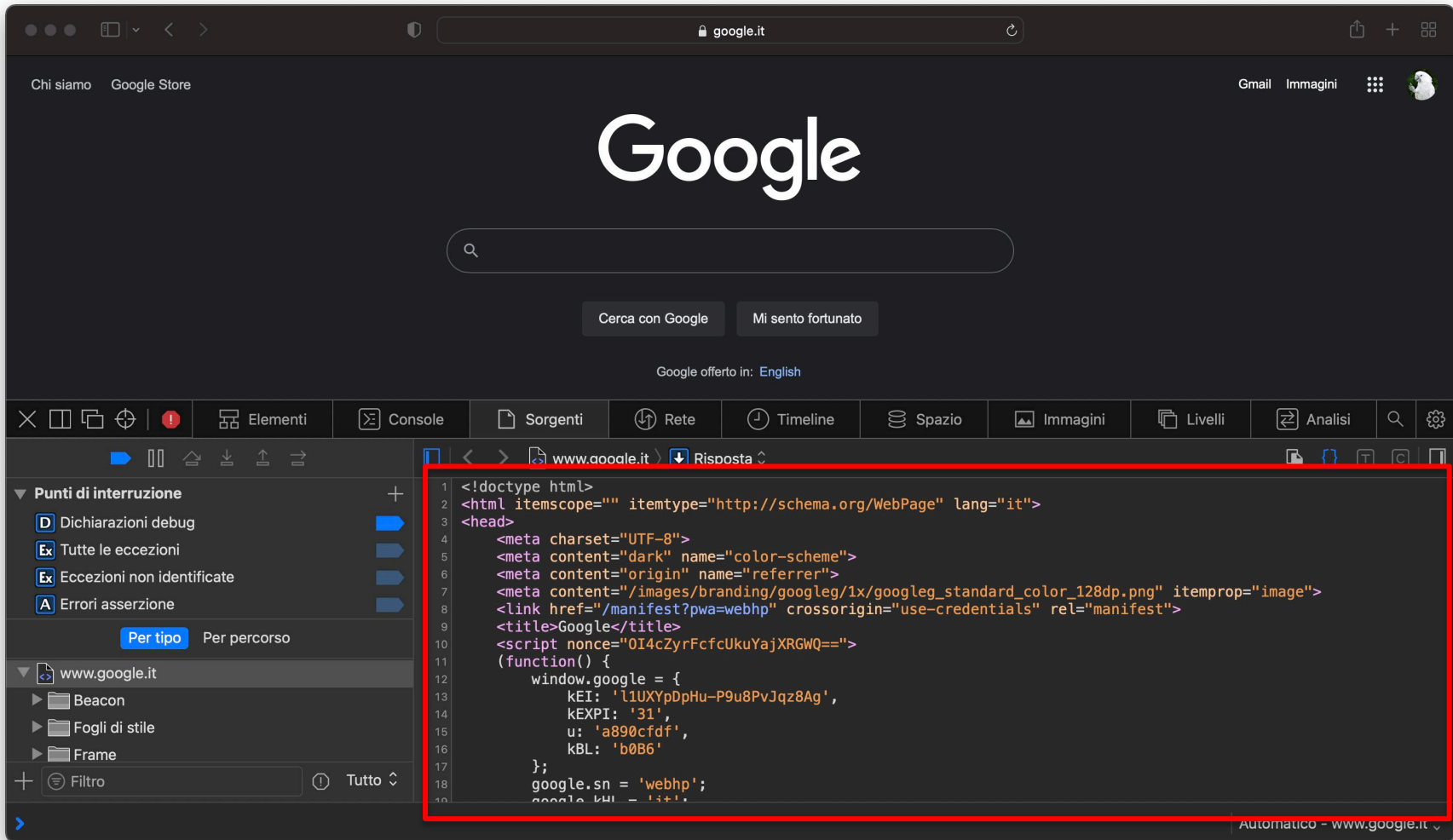
<https://docs.python-requests.org/en/latest/api/#requests.Response>



Cosa c'è nella GET Response?

```
>>> r = requests.get('https://www.google.com')
>>> print(r.content)

b'<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage" lang="it"><head><meta cont
ent="text/html; charset=UTF-8" http-equiv="Content-Type"><meta content="/images/branding/googleg/1x
```



Tempo di risposta di un server HTTP

- Valutare il tempo di risposta del server di Google per scaricare l'home page

```
1 import requests
2
3 r = requests.get('http://www.google.com')
4 print('Tempo di Risposta:', r.elapsed.microseconds / 1000, 'ms')
```

- Ripetere la misura 10 volte

```
1 import requests
2
3 for _ in range(10):
4     r = requests.get('http://www.google.com')
5     print('Tempo di Risposta:', r.elapsed.microseconds / 1000, 'ms')
```

Tempo di risposta di un server HTTP

- Valutare il tempo di risposta del server di Google per scaricare l'home page

```
1 import requests
2
3 r = requests.get('http://www.google.com')
4 print('Tempo di Risposta:', r.elapsed.microseconds / 1000, 'ms')
```

- Ripetere la misura 10 volte

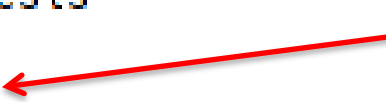
```
1 import requests
2
3 for _ in range(10):
4     r = requests.get('http://www.google.com')
5     print('Tempo di Risposta:', r.elapsed.microseconds / 1000, 'ms')
```

- Come possiamo calcolare minimo, media e massimo?

Calcolo minimo, media e massimo

```
1 import requests
2
3 tempi = []
4 for _ in range(10):
5     r = requests.get('http://www.google.com')
6     tempi.append(r.elapsed.microseconds / 1000)
7
8 print('Tempo di Risposta - MIN:', min(tempi))
9 print('Tempo di Risposta - AVG:', sum(tempi)/len(tempi))
10 print('max', max(tempi))
```

Una lista per i valori



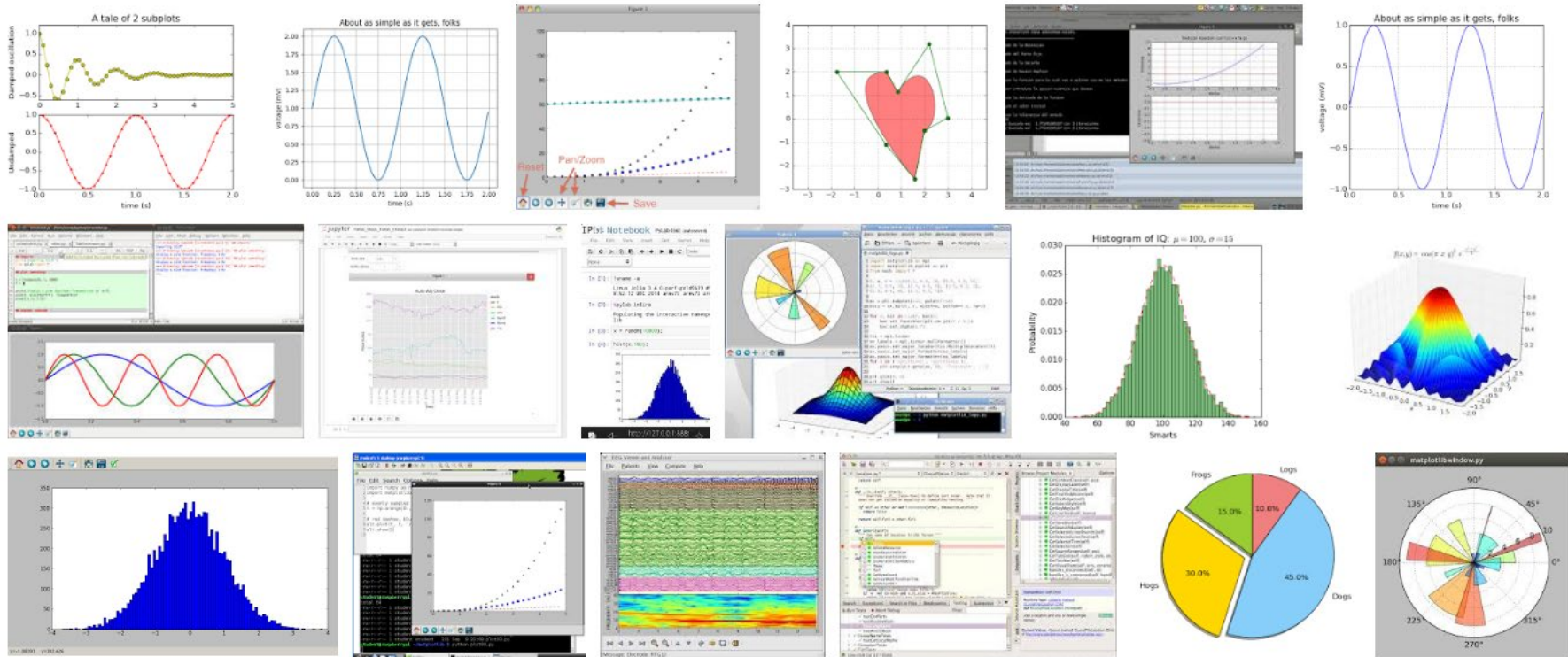
- **Altre funzioni per calcolare la media:**
 - statistics.mean() -> lento ma molto preciso
 - statistics è dentro Python 3.4+
 - numpy.mean() -> veloce e mediamente preciso

Calcolo minimo, media e massimo

```
1  import requests
2
3  tempi = []
4  for _ in range(10):
5      r = requests.get('http://www.google.com')
6      tempi.append(r.elapsed.microseconds / 1000)
7
8  print('Tempo di Risposta - MIN:', min(tempi))
9  print('Tempo di Risposta - AVG:', sum(tempi)/len(tempi))
10 print('max', max(tempi))
```

- Come possiamo rappresentare graficamente i risultati?

matplotlib

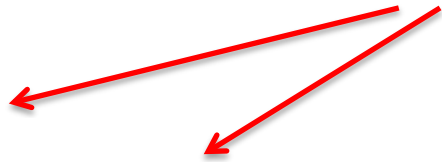


<http://matplotlib.org/examples/>

Grafici con Python (1)

```
1 import requests
2 import matplotlib
3
4 import matplotlib.pyplot as plt
5
```

Matplotlib



Grafici con Python (1)

```
1 import requests
2 import matplotlib
3
4 import matplotlib.pyplot as plt
5
6 tempi = []
7 for _ in range(10):
8     r = requests.get('http://www.google.com')
9     tempi.append(r.elapsed.microseconds / 1000)
10
```

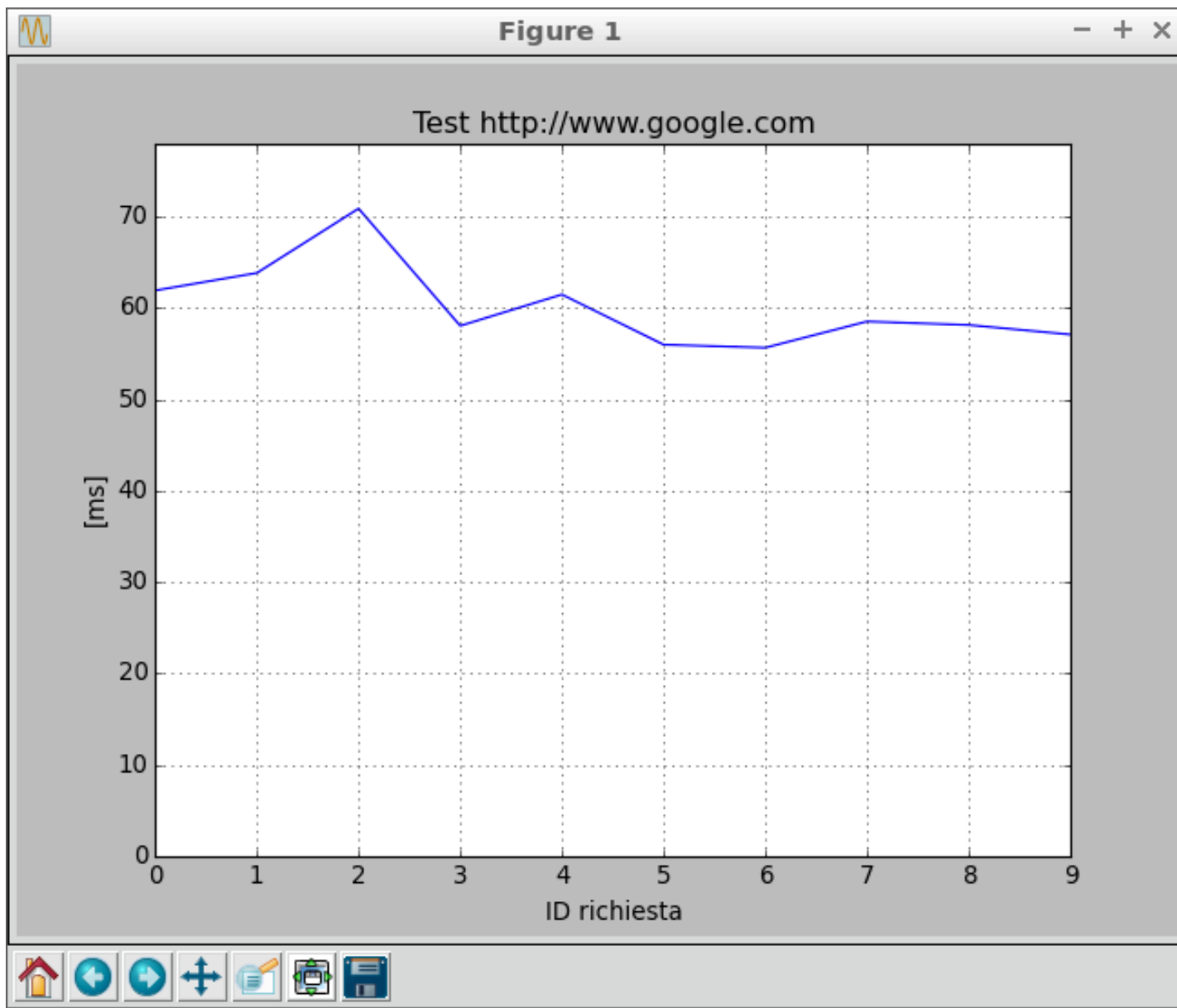
Grafici con Python (1)

```
1 import requests
2 import matplotlib
3
4 import matplotlib.pyplot as plt
5
6 tempi = []
7 for _ in range(10):
8     r = requests.get('http://www.google.com')
9     tempi.append(r.elapsed.microseconds / 1000)
10
11 print('Tempo di Risposta - MIN:', min(tempi))
12 print('Tempo di Risposta - AVG:', sum(tempi)/len(tempi))
13 print('Tempo di Risposta - MAX:', max(tempi))
14
15 plt.figure()
16 plt.plot(tempi)
17 plt.ylim([0, max(tempi)])
18 plt.show()
```

Grafici con Python (2)

```
1 import requests
2 import matplotlib.pyplot as plt
3 tempi = []
4 for _ in range(10):
5     r = requests.get('http://www.google.com')
6     tempi.append(r.elapsed.microseconds / 1000)
7
8 print('Tempo di Risposta - MIN:', min(tempi))
9 print('Tempo di Risposta - AVG:', sum(tempi)/len(tempi))
10 print('Tempo di Risposta - MAX', max(tempi))
11
12 plt.figure()
13 plt.plot(tempi)
14 plt.ylim([0, 1.1*max(tempi)])
15 plt.xlabel('ID richiesta')
16 plt.ylabel('[ms]')
17 plt.title('Test http://www.google.com')
18 plt.grid()
19 plt.show()
```

Grafici con Python (3)



Tempo di risposta di server HTTP multipli

```
1 import requests
2
3 siti = ['http://www.gazzetta.it', 'http://www.netflix.com', 'http://www.facebook.com']
```

Tempo di risposta di server HTTP multipli

```
1 import requests
2
3 siti = ['http://www.gazzetta.it', 'http://www.netflix.com', 'http://www.facebook.com']
4 for url in siti:
5     print('Test', url)
6     tempi = []
```

Blocco for per i siti

Tempo di risposta di server HTTP multipli

```
1 import requests
2
3 siti = ['http://www.gazzetta.it', 'http://www.netflix.com', 'http://www.facebook.com']
4 for url in siti:
5     print('Test', url)
6     tempi = []
7     for _ in range(10):
8         r = requests.get(url)
9         tempi.append(r.elapsed.microseconds/1000)
```

Blocco for per i siti

*Blocco for range(10)
per 10 requests/sito*

Tempo di risposta di server HTTP multipli

```
1 import requests
2
3 siti = ['http://www.gazzetta.it', 'http://www.netflix.com', 'http://www.facebook.com']
4 for url in siti:
5     print('Test', url)
6     tempi = []
7     for _ in range(10):
8         r = requests.get(url)
9         tempi.append(r.elapsed.microseconds/1000)
10    print('Tempo di Risposta - MIN:', min(tempi))
11    print('Tempo di Risposta - AVG:', sum(tempi)/len(tempi))
12    print('Tempo di Risposta - MAX', max(tempi))
```

Blocco for per i siti

*Blocco for range(10)
per 10 requests/sito*

Tempo di risposta di server HTTP multipli

```
1 import requests
2
3 siti = ['http://www.gazzetta.it', 'http://www.netflix.com', 'http://www.facebook.com']
4 for url in siti:
5     print('Test', url)
6     tempi = []
7     for _ in range(10):
8         r = requests.get(url)
9         tempi.append(r.elapsed.microseconds/1000)
10    print('Tempo di Risposta - MIN:', min(tempi))
11    print('Tempo di Risposta - AVG:', sum(tempi)/len(tempi))
12    print('Tempo di Risposta - MAX', max(tempi))
```

```
for ID_url in range(len(siti)):
    print('test sito #', ID_url)
    r = requests.get(siti[ID_url])
```

```
for ID_url, url in enumerate(siti):
    print('Test sito #', ID_url)
    r = requests.get(url)
```

VS

```
for url in siti:
    r = requests.get(url)
```

Grafici con server HTTP multipli

Matplotlib

Per calcolare il massimo

```
1 import requests
2 import matplotlib.pyplot as plt
3 m = 0 # massimo tra i massimi
4 plt.figure()
5 siti = ['http://www.gazzetta.it', 'http://www.netflix.com', 'http://www.facebook.com']
6 for url in siti:
7     print('Test', url)
8     tempi = []
9     for _ in range(10):
10         r = requests.get(url)
11         tempi.append(r.elapsed.microseconds/1000)
```

Grafici con server HTTP multipli

```
1 import requests
2 import matplotlib.pyplot as plt
3 m = 0 # massimo tra i massimi
4 plt.figure()
5 siti = ['http://www.gazzetta.it', 'http://www.netflix.com', 'http://www.facebook.com']
6 for url in siti:
7     print('Test', url)
8     tempi = []
9     for _ in range(10):
10         r = requests.get(url)
11         tempi.append(r.elapsed.microseconds/1000)
12     plt.plot(tempi, label=url)
13     print('Tempo di Risposta - MIN:', min(tempi))
14     print('Tempo di Risposta - AVG:', sum(tempi)/len(tempi))
15     print('Tempo di Risposta - MAX', max(tempi))
16     m = max([m, max(tempi)]) # ricalcolo il massimo tra i massimi
```

Matplotlib

Per calcolare il massimo

*plt.plot per graficare i
valori in tempi*

Massimo tra i massimi

Grafici con server HTTP multipli

```
1 import requests
2 import matplotlib.pyplot as plt
3 m = 0 # massimo tra i massimi
4 plt.figure()
5 siti = ['http://www.gazzetta.it', 'http://www.netflix.com', 'http://www.facebook.com']
6 for url in siti:
7     print('Test', url)
8     tempi = []
9     for _ in range(10):
10         r = requests.get(url)
11         tempi.append(r.elapsed.microseconds/1000)
12     plt.plot(tempi, label=url)
13     print('Tempo di Risposta - MIN:', min(tempi))
14     print('Tempo di Risposta - AVG:', sum(tempi)/len(tempi))
15     print('Tempo di Risposta - MAX', max(tempi))
16     m = max([m, max(tempi)]) # ricalcolo il massimo tra i massimi
17
18 plt.ylim([0, 1.1*m])
19 plt.xlabel('ID richiesta')
20 plt.ylabel('[ms]')
21 plt.title('Test tempi di risposta')
22 plt.legend(loc='lower right', fontsize=8)
23 plt.grid()
24 plt.show()
```

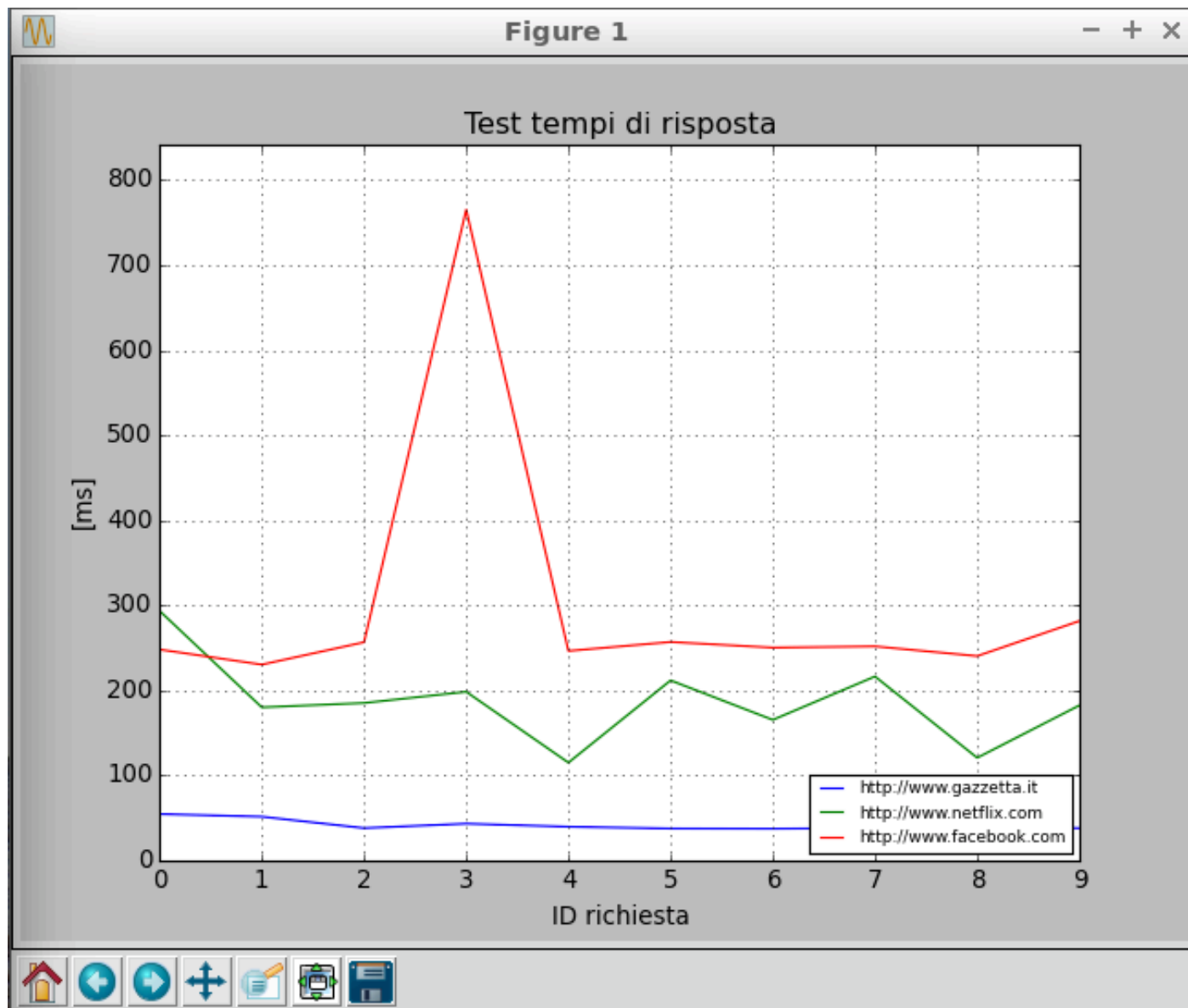
Matplotlib

Per calcolare il massimo

plt.plot per i valori in tempi

Massimo tra i massimi

Grafici con server HTTP multipli (2)



Esercizio 1.1

Scrivere uno script che stampi il nome della pagina col **miglior tempo di risposta medio** tra **2** siti Internet.

- Numero di richieste = 5
- Siti internet:
 1. `http://www.google.com`
 2. `http://www.youtube.com`

Soluzione esercizio 1.1

```
1 import requests
2
3 tempi1 = []
4 for _ in range(5):
5     r = requests.get('http://www.google.com')
6     tempi1.append(r.elapsed.microseconds/1000)
7 avg1 = sum(tempi1)/len(tempi1)
8
9 tempi2 = []
10 for _ in range(5):
11     r = requests.get('http://www.youtube.com')
12     tempi2.append(r.elapsed.microseconds/1000)
13 avg2 = sum(tempi2)/len(tempi2)
14
15 if avg1 < avg2:
16     print('http://www.google.com')
17 else:
18     print('http://www.youtube.com')
```

Ora tocca a voi!

Esercizio 1.2

Scrivere uno script che stampi il nome della pagina col **miglior tempo di risposta medio** tra **6** siti Internet. Per il calcolo del tempo medio, si definisca la funzione *media(list)* che ritorna la media dei valori contenuti in *list*.

- Numero di richieste = 10
- Siti internet:
 1. `http://www.google.com`
 2. `http://www.youtube.com`
 3. `http://www.polimi.it`
 4. `http://www.wikipedia.org`
 5. `http://www.amazon.com`
 6. `http://www.twitter.com`

Ora tocca a voi!

Esercizio 1.2

Hint:

`LIST.index(x)` ritorna la **posizione dell'elemento `x`** nella lista *LIST*.

È l'inverso dell'accesso alla lista tramite posizione

```
L = [1, 5, 20, 4]          L = [1, 5, 20, 4]
print(L.index(5)) # STAMPA 1 print(L[1]) # STAMPA 5
```

NB: gli indici in Python iniziano da 0!