Table of Contents

Fondamenti di Programmazione

Andrea Sterbini

lezione 11 - 10 novembre 2022

RECAP: Files

- **`with open(<filename>, mode=<mode>, encoding='utf8') as F:`**
  `...`
- **`F.read()`** legge tutto il contenuto
- **`F.readline()`** legge UNA riga (compreso **`'\n'`**)
- **`F.readlines()`** legge TUTTE le righe (compresi **`'\n'`**)
- **`F.write(<string>)`** scrive un testo nel file (dovete aggiungere voi '\n')
- **`print( ..., file=F)`** stampa nel file (molto comodo)

- `F.seek(0)` torna alla posizione 0 (inizio)

Ricerca delle parole più importanti in un gruppo di documenti

- per ogni documento
  - carichiamo le parole
  - le contiamo
  - ne calcoliamo la frequenza percentuale

Ricerca del file più importante dato un gruppo di parole

Le parole più frequenti potrebbero essere troppo comuni e poco significative (con poco contenuto semantico)

Quindi una parola sarà "interessante" se **appare spesso in un file** ma **appare in pochi file**

Vedi https://en.wikipedia.org/wiki/tf-idf (https://en.wikipedia.org/wiki/tf-idf)

Ci serve la frequenza di una parola in ciascun file (tf = term frequency)

- per ciascun file
  - contiamo le parole
  - dividiamo per il numero totale di parole

In [1]:
```
# ho un elenco di files con il loro encoding, ottenuto unzippando files.zip
files = {
    'files/holmes.txt'       : 'utf-8-sig',
    'files/alice.txt'        : 'utf-8-sig',
    'files/frankenstein.txt' : 'utf-8-sig',
    'files/alice_it.txt'     : 'latin',
    'files/prince.txt'       : 'utf-8-sig'
    }
```

In [3]:

```python
# per estrarre le parole da un file che contiene anche segni di interpunzione
def estrai_parole(filename, encoding):
    # apro il file e ne leggo il contenuto
    with open(filename, encoding=encoding) as FILE:
        text = FILE.read()
        # se voglio lo converto in lowercase (ma potrei preferire di no)
        text = text.lower()
        # calcolo l'insieme dei caratteri presenti nel testo
        caratteri = set(text)
        # ne estraggo solo i caratteri non-alpha
        nonalfa = [ c for c in caratteri if not c.isalpha() ]
        # li rimpiazzo con spazio
        for c in nonalfa:
            text = text.replace(c, ' ')
        # uso split sul testo in cui ho eliminato tutti i non-apha
        return text.split()

# esempio: parole del libro Alice nel paese delle meraviglie
parole = estrai_parole('files/alice_it.txt', 'latin')

### come usare meno memoria? lavorando riga per riga
parole[:30]
```

Out[3]:

```
['charles',
 'lutwidge',
 'dodgson',
 'alice',
 'nel',
 'paese',
 'delle',
 'meraviglie',
 'questo',
 'e',
 'book',
 'è',
 'stato',
 'realizzato',
 'anche',
 'grazie',
 'al',
 'sostegno',
 'di',
```

In [10]:

```
# per contare quante sono le parole
def conta_parole(parole):
    # trovo le parole uniche usando un insieme
    parole_uniche = set(parole)
    # e costruisco il dizionario parola:conteggio usando count  ### INEFFICIENTE!!!
    N = len(parole)
    return { parola: parole.count(parola)/N for parola in parole_uniche }

# rifaccio i conti sulle parole di Alice
conteggi = conta_parole(parole)

#print(list(conteggi.items())[:20])

# le 50 parole più presenti in Alice sono:
più_presenti = sorted(conteggi, reverse=True,
                      key=lambda K: conteggi[K])[:50]
list(più_presenti)
```

Out[10]:

```
['e', 'il', 'la', 'di', 'che', 'non', 'un', 'alice', 'a', 'si', 'disse', 'in', 'ma', 'con
', 'le', 'per', 'è', 'una', 'se', 'era', 'i', 'l', 'più', 'come', 'rispose', 'd', 'da', '
del', 'perchè', 'mi', 'gli', 'al', 'regina', 'della', 'così', 'lo', 'quando', 'poi', 're
', 'cosa', 'o', 'aveva', 'io', 'sono', 'questo', 'tu', 'ne', 'tutti', 'qualche', 'cappell
aio']
```

In [11]:

```
# frequenza delle parole nel file iesimo
#    - tf_i  = # presenze / # parole del file
# per tutti i file costruiamo il dizionario del numero di files in cui appaiono
frequenze = {}      # conteggio delle parole nei file: frequenze[name][parola] -> % d

# conto le frequenze file per file
for filename, encoding in files.items():
    print(filename)
    parole = estrai_parole(filename, encoding)
    conteggio = conta_parole(parole)
    frequenze[filename] = conteggio
```

```
files/holmes.txt
files/alice.txt
files/frankenstein.txt
files/alice_it.txt
files/prince.txt
```

e l'inverso della presenza della parola nei file (idf = inverse document frequency)

In [13]:
```
# conto i file che contengono una parola
# presenze[parola] -> # di file che la contengono
presenze  = {}
for filename in files:
    for parola in frequenze[filename]:
        presenze[parola] = presenze.get(parola, 0) + 1

# divido per il numero di file per avere la presenza %
Nfile = len(files)
for parola in presenze:
    presenze[parola] /= Nfile

list(presenze.items())[:30]
```

Out[13]:
```
[('acute', 0.2), ('expense', 0.8), ('instant', 0.4), ('definitely', 0.2), ('displaying',
0.8), ('respects', 0.6), ('measure', 0.8), ('jersey', 0.2), ('disadvantages', 0.2), ('bes
t', 0.8), ('put', 0.8), ('foil', 0.2), ('dazed', 0.2), ('wayward', 0.2), ('arrest', 0.6),
('ferret', 0.2), ('clergyman', 0.2), ('agonies', 0.4), ('carriage', 0.4), ('figured', 0.
2), ('feathers', 0.4), ('sherlock', 0.2), ('crest', 0.2), ('gaunt', 0.4), ('introduction
', 0.6), ('stage', 0.4), ('imprudence', 0.4), ('gleaming', 0.2), ('fanciful', 0.4), ('cri
nkled', 0.2)]
```

In [15]:
```
from math import log

# log dell'inverso della frequenza nei documenti
#    - idf   = log( # di file / # di file che contengono la parola )
idf = { parola :  log(1/quante)
                  for parola,quante in presenze.items() }

list(idf.items())[:30]
# NOTA: ogni parola appare in almeno UN file quindi 1/N è sempre calcolabile
```

Out[15]:
```
[('acute', 1.6094379124341003), ('expense', 0.22314355131420976), ('instant', 0.916290731
8741551), ('definitely', 1.6094379124341003), ('displaying', 0.22314355131420976), ('resp
ects', 0.5108256237659907), ('measure', 0.22314355131420976), ('jersey', 1.60943791243410
03), ('disadvantages', 1.6094379124341003), ('best', 0.22314355131420976), ('put', 0.2231
4355131420976), ('foil', 1.6094379124341003), ('dazed', 1.6094379124341003), ('wayward',
1.6094379124341003), ('arrest', 0.5108256237659907), ('ferret', 1.6094379124341003), ('cl
ergyman', 1.6094379124341003), ('agonies', 0.9162907318741551), ('carriage', 0.9162907318
741551), ('figured', 1.6094379124341003), ('feathers', 0.9162907318741551), ('sherlock',
1.6094379124341003), ('crest', 1.6094379124341003), ('gaunt', 0.9162907318741551), ('intr
oduction', 0.5108256237659907), ('stage', 0.9162907318741551), ('imprudence', 0.916290731
8741551), ('gleaming', 1.6094379124341003), ('fanciful', 0.9162907318741551), ('crinkled
', 1.6094379124341003)]
```

In [18]:
```python
# e finalmente posso vedere quale file "pesa di più" rispetto alle parole della query

# date le parole di una query
# per ciascun file
    # ne conto la frequenza nel file
    # la moltiplico per idf
    # sommando ottendo il "peso" del file per quella query

query = [ 'turtle', 'alice']

pesi = {}
for file in frequenze:
    peso = 0
    for p in query:
        # aggiungo il peso della parola (0 se non presente)
        peso += frequenze[file].get(p,0) * idf[p]
    pesi[file] = peso
    print(f"{file:30} {peso:00.3}")

# ordino i file per peso decrescente
sorted(pesi.items(), reverse=True, key=lambda coppia: coppia[1])
```

```
files/holmes.txt               5.62e-05
files/alice.txt                0.00989
files/frankenstein.txt         0.0
files/alice_it.txt             0.00889
files/prince.txt               0.0
```

Out[18]:
```
[('files/alice.txt', 0.009889199619031072), ('files/alice_it.txt', 0.008886242041680276),
('files/holmes.txt', 5.6241811189737675e-05), ('files/frankenstein.txt', 0.0), ('files/pr
ince.txt', 0.0)]
```

Altri tipi di file: JSON (https://docs.python.org/3/library/json.html)

File di testo con sintassi semplificata per rappresentare:

- dizionari
- liste
- interi, stringhe, float, bool, None

Molto usati nelle applicazioni WEB

In [19]:

```python
## Esempio
import json

with open('api.github.com.json') as F:
    api = json.load(F)
!cat api.github.com.json
```

```
{
  "current_user_url": "https://api.github.com/user",
  "current_user_authorizations_html_url": "https://github.com/settings/connections/applic
ations{/client_id}",
  "authorizations_url": "https://api.github.com/authorizations",
  "code_search_url": "https://api.github.com/search/code?q={query}{&page,per_page,sort,or
der}",
  "commit_search_url": "https://api.github.com/search/commits?q={query}{&page,per_page,so
rt,order}",
  "emails_url": "https://api.github.com/user/emails",
  "emojis_url": "https://api.github.com/emojis",
  "events_url": "https://api.github.com/events",
  "feeds_url": "https://api.github.com/feeds",
  "followers_url": "https://api.github.com/user/followers",
  "following_url": "https://api.github.com/user/following{/target}",
  "gists_url": "https://api.github.com/gists{/gist_id}",
  "hub_url": "https://api.github.com/hub",
  "issue_search_url": "https://api.github.com/search/issues?q={query}{&page,per_page,sor
t,order}",
```

Attenzione:

- NON si possono inserire commenti
- NO virgola aggiuntiva in fondo a lista o dizionario
- SOLO chiavi semplici (NO tuple, SI int, float, str, none, bool)
- SOLO doppi apici **"** (NO singoli apici)

Abbastanza facili da usare in python

Conversione automatica per alcuni tipi di dati

- **json.load(<file>) -> dict | list | tipo_base**
- **json.dump(<obj>, <file>)**

Personalizzabile anche per altri tipi di oggetti (non ovvio)

In [28]:
```
# posso anche leggere da una stringa in formato JSON
XX = json.loads('''
[{"nome": "Minnie", "cognome": "Mouse", "telefono": "555-54321",
  "indirizzo": "via di M.me Curie 1", "città": "Topolinia"},
{"nome": "Pippo", "cognome": "de' Pippis", "telefono": "555-33333",
 "indirizzo": "via dei Pioppi 1", "città": "Topolinia"}]
''')
print(XX)
# oppure produrre la stringa in formato JSON da una struttura Python
print(json.dumps(XX))
```

```
[{'nome': 'Minnie', 'cognome': 'Mouse', 'telefono': '555-54321', 'indirizzo': 'via di M.m
e Curie 1', 'città': 'Topolinia'}, {'nome': 'Pippo', 'cognome': "de' Pippis", 'telefono':
'555-33333', 'indirizzo': 'via dei Pioppi 1', 'città': 'Topolinia'}]
[{"nome": "Minnie", "cognome": "Mouse", "telefono": "555-54321", "indirizzo": "via di M.m
e Curie 1", "citt\u00e0": "Topolinia"}, {"nome": "Pippo", "cognome": "de' Pippis", "telef
ono": "555-33333", "indirizzo": "via dei Pioppi 1", "citt\u00e0": "Topolinia"}]
```

In [122]:
```
# o produrre una stringa
json.dumps(XX)
```

Out[122]:
```
'[{"nome": "Minnie", "cognome": "Mouse", "telefono": "555-54321", "indirizzo": "via di M.
me Curie 1", "citt\\u00e0": "Topolinia"}, {"nome": "Pippo", "cognome": "de\' Pippis", "te
lefono": "555-33333", "indirizzo": "via dei Pioppi 1", "citt\\u00e0": "Topolinia"}]'
```

In [30]:
```
# Un file json può contenere valori semplici (int, float, str, True=true, False=false,
print(json.dumps(None))

print(json.dumps([False, True]))

## NOTA: le stringhe DEVONO essere racchiuse da doppi apici
print(json.dumps('Pape"rino'))
```

```
null
[false, true]
"Pape\"rino"
```

In [31]:
```
# Esempio: data una tabella come lista di dizionari
agenda = [
    {'nome': 'Paperino','cognome':'Paolino',      'telefono':'555-1313',  'indirizzo'
    {'nome': 'Gastone', 'cognome':'Paperone',     'telefono':'555-1717',  'indirizzo'
    {'nome': 'Paperon', 'cognome':"de' Paperoni", 'telefono':'555-99999', 'indirizzo'
    {'nome': 'Archimede','cognome':'Pitagorico',  'telefono':'555-11235', 'indirizzo'
    {'nome': 'Pietro',  'cognome':'Gambadilegno', 'telefono':'555-66666', 'indirizzo'
    {'nome': 'Trudy',   'cognome':'Gambadilegno', 'telefono':'555-66666', 'indirizzo'
    {'nome': 'Topolino','cognome':'Mouse',        'telefono':'555-12345', 'indirizzo'
    {'nome': 'Minnie',  'cognome':'Mouse',        'telefono':'555-54321', 'indirizzo'
    {'nome': 'Pippo',   'cognome':"de' Pippis",   'telefono':'555-33333', 'indirizzo'
    ]
```

In [32]:
```
import json
# prima la salvo nel file
with open('agenda.json', encoding='utf8', mode='w') as F:
    json.dump(agenda, F)

!cat agenda.json
```

```
[{"nome": "Paperino", "cognome": "Paolino", "telefono": "555-1313", "indirizzo": "via dei
Peri 113", "citt\u00e0": "Paperopoli"}, {"nome": "Gastone", "cognome": "Paperone", "telef
ono": "555-1717", "indirizzo": "via dei Baobab 42", "citt\u00e0": "Paperopoli"}, {"nome":
"Paperon", "cognome": "de' Paperoni", "telefono": "555-99999", "indirizzo": "colle Papero
1", "citt\u00e0": "Paperopoli"}, {"nome": "Archimede", "cognome": "Pitagorico", "telefon
o": "555-11235", "indirizzo": "colle degli Inventori 1", "citt\u00e0": "Paperopoli"}, {"n
ome": "Pietro", "cognome": "Gambadilegno", "telefono": "555-66666", "indirizzo": "via dei
Ladri 13", "citt\u00e0": "Topolinia"}, {"nome": "Trudy", "cognome": "Gambadilegno", "tele
fono": "555-66666", "indirizzo": "via dei Ladri 13", "citt\u00e0": "Topolinia"}, {"nome":
"Topolino", "cognome": "Mouse", "telefono": "555-12345", "indirizzo": "via degli Investig
atori 1", "citt\u00e0": "Topolinia"}, {"nome": "Minnie", "cognome": "Mouse", "telefono":
"555-54321", "indirizzo": "via di M.me Curie 1", "citt\u00e0": "Topolinia"}, {"nome": "Pi
ppo", "cognome": "de' Pippis", "telefono": "555-33333", "indirizzo": "via dei Pioppi 1",
"citt\u00e0": "Topolinia"}]
```

In [33]:
```
# poi la ricarico
with open('agenda.json', encoding='utf8') as F:
    L1 = json.load(F)
L1[:2]  # e ne mostro i primi 2 elementi
```

Out[33]:
```
[{'nome': 'Paperino', 'cognome': 'Paolino', 'telefono': '555-1313', 'indirizzo': 'via dei
Peri 113', 'città': 'Paperopoli'}, {'nome': 'Gastone', 'cognome': 'Paperone', 'telefono':
'555-1717', 'indirizzo': 'via dei Baobab 42', 'città': 'Paperopoli'}]
```

File YAML (https://pyyaml.org/wiki/PyYAMLDocumentation) (un altro modo di rappresentare dati annidati in testo semplice)

- dizionari (una **chiave : valore** per ciascuna riga)
- liste (valori su righe diverse, preceduti da `-`)
- dati semplici (str senza apici se possibile, True, False, null=None, interi, float)
- documenti multipli
- generici oggetti Python (advanced)

Per annidare le strutture si usa l'**indentazione**

In [34]:
```
import yaml
with open('agenda.yaml', mode='w', encoding='utf8') as F:
    yaml.dump(agenda[:2], F)
!cat agenda.yaml
```

```
- "citt\xE0": Paperopoli
  cognome: Paolino
  indirizzo: via dei Peri 113
  nome: Paperino
  telefono: 555-1313
- "citt\xE0": Paperopoli
  cognome: Paperone
  indirizzo: via dei Baobab 42
  nome: Gastone
  telefono: 555-1717
```

leggere/scrivere file YAML

- **`yaml.dump(<oggetto>, FILE)`** salva l'oggetto nel file aperto con open
- **`yaml.safe_load(FILE) -> <oggetto>`** legge l'oggetto dal file aperto con open

In [35]:
```yaml
## posso decodificare direttamente del testo
testo = """
  none: [~, null]
  bool: [true, false, on, off]
  int: 42
  float: 3.14159
  list:
    - LITE
    - RES_ACID
    - SUS_DEXT
  dict:
    hp: 13
    sp: 5
"""
yaml.safe_load(testo)
```

Out[35]:
```
{'none': [None, None], 'bool': [True, False, True, False], 'int': 42, 'float': 3.14159, '
list': ['LITE', 'RES_ACID', 'SUS_DEXT'], 'dict': {'hp': 13, 'sp': 5}}
```

In [37]:
```yaml
with open('esempio.yaml', mode='w', encoding='utf8') as F:
    print(testo, file=F)

!cat esempio.yaml
# oppure leggerlo direttamente da file
with open('esempio.yaml') as F:
    EX = yaml.safe_load(F)
EX
```

```
none: [~, null]
bool: [true, false, on, off]
int: 42
float: 3.14159
list:
  - LITE
  - RES_ACID
  - SUS_DEXT
dict:
  hp: 13
  sp: 5
```

Out[37]:
```
{'none': [None, None], 'bool': [True, False, True, False], 'int': 42, 'float': 3.14159, '
list': ['LITE', 'RES_ACID', 'SUS_DEXT'], 'dict': {'hp': 13, 'sp': 5}}
```

Leggere pagine o file da Internet

Ci sono molte librerie, la più comune è [requests (https://requests.readthedocs.io/en/latest/user/quickstart/)](https://requests.readthedocs.io/en/latest/user/quickstart/)

- permette di eseguire sia **GET** che **POST**
- con parametri
- torna un codice di errore/OK
- e decodifica il testo della pagina html o json

Molto potente, permette anche streaming, sessioni, ...

In [39]:
```
# importo la libreria
import requests

# leggo la pagina di python.org
pagina = requests.get('https://python.org')

# lo status code ci dice se tutto è andato bene
status = pagina.status_code

# se è un testo nell'attributo text trovo il testo decodificato
contenuto = pagina.text

status, pagina.encoding, contenuto[:100]
```

Out[39]:
```
(200, 'utf-8', '<!doctype html>\n<!--[if lt IE 7]>   <html class="no-js ie6 lt-ie7 lt-ie8
lt-ie9">   <![endif]-->\n<!-')
```

le pagine JSON vengono automaticamente decodificate

In [40]:
```
# leggo da internet una pagina JSON
pagina_json = requests.get('https://api.github.com')

# la decodifica avviene automaticamente col metodo json()
risultato = pagina_json.json()
risultato
```

Out[40]:

```
{'current_user_url': 'https://api.github.com/user', 'current_user_authorizations_html_url
': 'https://github.com/settings/connections/applications{/client_id}', 'authorizations_ur
l': 'https://api.github.com/authorizations', 'code_search_url': 'https://api.github.com/s
earch/code?q={query}{&page,per_page,sort,order}', 'commit_search_url': 'https://api.githu
b.com/search/commits?q={query}{&page,per_page,sort,order}', 'emails_url': 'https://api.gi
thub.com/user/emails', 'emojis_url': 'https://api.github.com/emojis', 'events_url': 'http
s://api.github.com/events', 'feeds_url': 'https://api.github.com/feeds', 'followers_url':
'https://api.github.com/user/followers', 'following_url': 'https://api.github.com/user/fo
llowing{/target}', 'gists_url': 'https://api.github.com/gists{/gist_id}', 'hub_url': 'htt
ps://api.github.com/hub', 'issue_search_url': 'https://api.github.com/search/issues?q={qu
ery}{&page,per_page,sort,order}', 'issues_url': 'https://api.github.com/issues', 'keys_ur
l': 'https://api.github.com/user/keys', 'label_search_url': 'https://api.github.com/searc
h/labels?q={query}&repository_id={repository_id}{&page,per_page}', 'notifications_url': '
https://api.github.com/notifications', 'organization_url': 'https://api.github.com/orgs
/{org}', 'organization_repositories_url': 'https://api.github.com/orgs/{org}/repos{?type,
page,per_page,sort}', 'organization_teams_url': 'https://api.github.com/orgs/{org}/teams
', 'public_gists_url': 'https://api.github.com/gists/public', 'rate_limit_url': 'https://
api.github.com/rate_limit', 'repository_url': 'https://api.github.com/repos/{owner}/{repo
o}', 'repository_search_url': 'https://api.github.com/search/repositories?q={query}{&pag
e,per_page,sort,order}', 'current_user_repositories_url': 'https://api.github.com/user/re
pos{?type,page,per_page,sort}', 'starred_url': 'https://api.github.com/user/starred{/owne
r}{/repo}', 'starred_gists_url': 'https://api.github.com/gists/starred', 'topic_search_ur
l': 'https://api.github.com/search/topics?q={query}{&page,per_page}', 'user_url': 'http
s://api.github.com/users/{user}', 'user_organizations_url': 'https://api.github.com/user/
orgs', 'user_repositories_url': 'https://api.github.com/users/{user}/repos{?type,page,per
_page,sort}', 'user_search_url': 'https://api.github.com/search/users?q={query}{&page,per
_page,sort,order}'}
```


Scaricare file binari

- il contenuto "raw" è nell'attributo **`content`** della risposta

In [42]:

```python
## possiamo anche scaricare file binari
logo = requests.get('https://www.python.org/static/img/python-logo@2x.png')

# posso estrarre dagli headers le dimensioni e il tipo del file ()
print(logo.headers['Content-Type'], logo.headers['Content-Length'])

# se si tratta di un file di dati (immagine/audio/film ...)
# il contenuto lo trovo nell'attributo content
dati = logo.content

# per salvarlo devo aprire il file in scrittura ('w') e in modo binario ('b')
with open('logo.png', mode='wb') as F:
    F.write(dati)
```

```
image/png 15770
```

Passare parametri ad una GET

Si usa l'argomento **params**

- **parametri={ chiave: valore, ...}**
- **requests.get(<URL>, params=parametri)**

In [44]:
```python
# Search GitHub's repositories for requests
response = requests.get(
    #'https://api.github.com/search/repositories',
    'https://google.com',
    params={'q': 'requests+language:python'},
)
# response.json()
response.text[:1000]
```

Out[44]:
```
'<!DOCTYPE html><html lang="it" dir="ltr"><head><style nonce="7TONh04u7T6gIjcLn2ZHNQ">\n
a, a:link, a:visited, a:active, a:hover {\n  color: #1a73e8;\n  text-decoration: none;\
n}\nbody {\n  font-family: Roboto,RobotoDraft,Helvetica,Arial,sans-serif;\n  text-align:
center;\n  -ms-text-size-adjust: 100%;\n  -moz-text-size-adjust: 100%;\n  -webkit-text-si
ze-adjust: 100%;\n}\n.box {\n  border: 1px solid #dadce0;\n  box-sizing: border-box;\n  b
order-radius: 8px;\n  margin: 24px auto 5px auto;\n  max-width: 800px;\n  padding: 24px;\
n}\nh1 {\n  color: #2c2c2c;\n  font-size: 24px;\n  hyphens: auto;\n  margin: 24px 0;\n}\
n.icaCallout {\n  background-color: #f8f9fa;\n  padding: 12px 16px;\n  border-radius: 10p
x;\n  margin-bottom: 10px;\n}\np, .sub, .contentText, .icaCallout {\n  color: #5f6368;;\n
font-size: 14px;\n  line-height: 20px;\n  letter-spacing: 0.2px;\n  text-align: left;\n}\
n.signin {\n  text-align: right;\n}\n.saveButtonContainer,\n.saveButtonContainerNarrowScr
een {\n  width: 100%;\n  margin-top: 12px;\n}\n.customButtonContainer {'
```

Passare parametri ad altri metodi HTTP

Si usa l'argomento **`data=<dizionario>`**

```python
requests.post(   'https://httpbin.org/post',
              data={'key':'value'})
requests.put(    'https://httpbin.org/put',
              data={'key':'value'})
requests.delete( 'https://httpbin.org/delete')
requests.head(   'https://httpbin.org/get')
requests.patch(  'https://httpbin.org/patch',
               data={'key':'value'})
requests.options('https://httpbin.org/get')
```