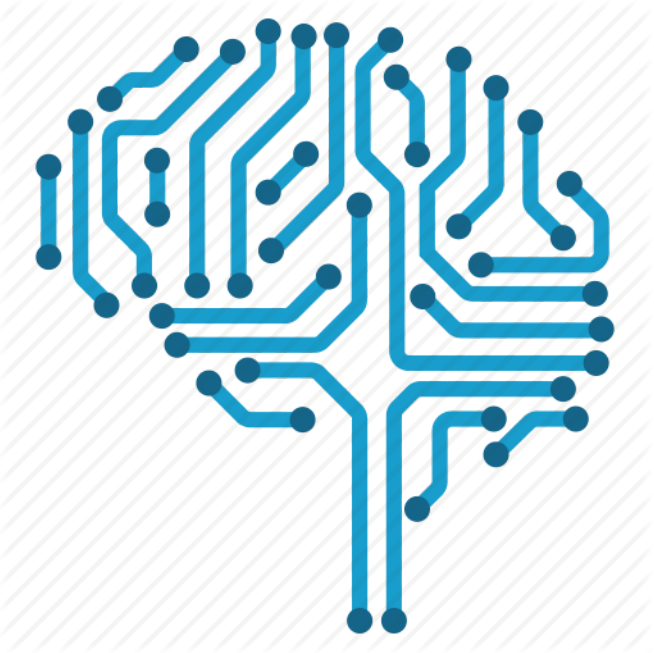
Machine Learning



make software
great again!

Giulio Angiani
I.I.S. "Blaise Pascal" - Reggio Emilia

Il linguaggio Python (3)

Python - Accesso a risorse esterne

- File di testo
- File di oggetti (serializzazione)
- Connessione a Database (MySQL e sqlite)
- Connessione a risorse di rete

Python - File di testo

- Funzione predefinita **open(filepath [,mode])**
  - filepath è il path relativo del file da aprire
  - mode può essere **r**ead (default), **w**rite, **a**ppend, **r+** (lettura e scrittura)


```
f = open("miofile.txt") # apre il file "miofile.txt" in lettura (predefinito)
# se il file non esiste genera un *FileNotFoundError*

f = open("miofile.txt", "w") # apre il file "miofile.txt" in sovrascrittura
# se il file non esiste lo crea

f = open("miofile.txt", "a") # apre il file "miofile.txt" in append
# se il file non esiste lo crea

f = open("miofile.txt", "r+") # apre il file "miofile.txt" in read/write
# se il file non esiste genera un *FileNotFoundError*
```

Python - File di testo

- scrittura e lettura
  - il metodo **read()** legge tutto il contenuto del file

PYTHON

```python
f = open("python.txt", "w")     # apro in scrittura
f.write("Un linguaggio proprio bello!\n") # scrivo una stringa (con fine riga)
f.write("Credo che lo userò a lungo!")
f.close() # chiudo il file
content = open("python.txt", "r").read()
print(content)
print("===")
print(repr(content))
```

OUTPUT

```
Un linguaggio proprio bello!
Credo che lo userò a lungo!
===
'Un linguaggio proprio bello!\nCredo che lo userò a lungo!'
```

Python - File di testo

- scrittura e lettura
  - il metodo **readlines()** legge tutto il contenuto del file splittando le righe

PYTHON

```
rows = open("python.txt", "r").readlines()
print(rows)
```

OUTPUT

```
['Un linguaggio proprio bello!\n', 'Credo che lo userò a lungo!']
```

Python - File di testo csv o tsv

- primo metodo:
  - utilizzo **readlines()** e splitto ogni riga per il carattere giusto

FILE CSV (STUDENTI.CSV)

```
matricola,cognome,nome,classe
92911,Angiani,Giulio,5C
28829,Muzzini,Alessandro,5C
22881,Catellani,Antonella,5D
```

PYTHON

```
rows = open("studenti.csv", "r").readlines()
labels = []
rownumber = 0
for r in rows:
    dati = r[0:-1] # rimuovo l'ultimo carattere di vai a capo
    info = dati.split(",")
    if rownumber == 0:
        labels = info
        print("LABELS :", info)
    else:
        print("ROW:  : ",info)
    rownumber += 1
```

OUTPUT

```
LABELS : ['matricola', 'cognome', 'nome','classe']
ROW:  :  ['92911', 'Angiani', 'Giulio','5C']
ROW:  :  ['28829', 'Muzzini', 'Alessandro','5C']
ROW:  :  ['22881', 'Catellani', 'Antonella','5D']
```

Python - File di testo csv o tsv

- secondo metodo:
  - utilizzo la libreria **csv** predefinita

PYTHON

```python
import csv
with open('studenti.csv', newline='') as csvfile:
    csv_iterator = csv.reader(csvfile, delimiter=',')
    labels = next(csv_iterator) # next legge una riga e manda avanti il puntatore di iterazione
    print("labels: ", labels)
    for row in csv_iterator:
        print(', '.join(row))
```

OUTPUT

```
labels:  ['matricola', 'cognome', 'nome','classe']
92911, Angiani, Giulio, 5C
28829, Muzzini, Alessandro, 5C
22881, Catellani, Antonella, 5D
```

Python - File di testo csv o tsv

- o meglio (se la prima riga contiene l'intestazione)

PYTHON

```python
from csv import DictReader
with open('studenti.csv', 'r') as read_obj:
    csv_dict_reader = DictReader(read_obj)
    for row in csv_dict_reader:
        print(row)

    # solo le labels
    labels = csv_dict_reader.fieldnames
    print("labels: ", labels)
```

OUTPUT

```
OrderedDict([('matricola', '92911'), ('cognome', 'Angiani'), ('nome', 'Giulio'), ('classe', '5C')])
OrderedDict([('matricola', '28829'), ('cognome', 'Muzzini'), ('nome', 'Alessandro'), ('classe', '5C')])
OrderedDict([('matricola', '22881'), ('cognome', 'Catellani'), ('nome', 'Antonella'), ('classe', '5D')])
labels:  ['matricola', 'cognome', 'nome', 'classe']
```

Python - libreria pandas

- libreria per **Data Analysis**
- permette di utilizzare file di testo come sorgente dati strutturate: il **Dataframe**
  - è una struttura dati tabellare (con righe e colonne numeriche)
  - le operazioni aritmetiche si allineano sulle etichette di riga e colonna
  - Può essere pensata come una sorta di contenitore di oggetti "Serie"
  - E' la struttura primaria della libreria pandas

- per installare pandas
- https://pandas.pydata.org/pandas-docs/stable/getting_started/install.html

Python - libreria pandas

PYTHON

```python
import pandas
dataframe = pandas.read_csv("studenti.csv")
print(dataframe)
print(dataframe.head(2))    # prime 2 righe
```

OUTPUT

```
   matricola    cognome        nome classe
0      92911    Angiani      Giulio     5C
1      28829    Muzzini  Alessandro     5C
2      22881  Catellani   Antonella     5D

   matricola    cognome        nome classe
0      92911    Angiani      Giulio     5C
1      28829    Muzzini  Alessandro     5C
```

Python - libreria pandas

PYTHON

```
dataframe.columns       # è un Index object
dataframe.columns[2]    # terzo elemento di Index
```

OUTPUT

```
Index(['matricola', 'cognome', 'nome', 'classe'], dtype='object')
'nome'
```

PYTHON

```
for col in dataframe:   # ciclare su un dataframe
    print(col, dataframe[col], dataframe[col].__class__.__name__)       # è un dict-like object
```

OUTPUT

```
matricola 0    92911
1    28829
2    22881
Name: matricola, dtype: int64 Series
cognome 0      Angiani
1      Muzzini
2    Catellani
Name: cognome, dtype: object Series
nome 0        Giulio
1    Alessandro
2     Antonella
Name: nome, dtype: object Series
classe 0    5C
1    5C
2    5D
Name: classe, dtype: object Series
```

Python - libreria pandas

PYTHON

```
print(dataframe["cognome"]) # solo la seconda colonna
print(dataframe["cognome"][1])  # solo il secondo elemento della prima colonna
```

OUTPUT

```
0      Angiani
1      Muzzini
2    Catellani
Name: cognome, dtype: object

Muzzini
```

- o più colonne contemporaneamente, anche in altro ordine

PYTHON

```
dataframe[["classe", "cognome"]]
```

OUTPUT

```
  classe    cognome
0     5C    Angiani
1     5C    Muzzini
2     5D  Catellani
```

Python - libreria pandas

- sottoinsiemi di righe

PYTHON

```
dataframe.iloc[1:2]
```

OUTPUT

```
   matricola  cognome        nome classe
1      28829  Muzzini  Alessandro     5C
```

- sottoinsiemi di righe e colonne

PYTHON

```
dataframe.iloc[1:,1:]
```

OUTPUT

```
     cognome        nome classe
1    Muzzini  Alessandro     5C
2  Catellani   Antonella     5D
```

Python - libreria pandas

- filtro righe condizionale

PYTHON

```
inquintac = dataframe['classe'] == "5C"
print(inquintac)
```

OUTPUT

```
0     True
1     True
2    False
Name: classe, dtype: bool
```

PYTHON

```
dataframe[inquintac]       # come indice un array di boolean
```

OUTPUT

```
   matricola  cognome        nome classe
0      92911  Angiani      Giulio     5C
1      28829  Muzzini  Alessandro     5C
```

Python - File di oggetti - serializzazione

- libreria **pickle** (sottaceto) per salvare oggetti su file
- due metodi
  - **dump** (dumps)
  - **load** (loads)

PYTHON

```python
oggetti = [10, 20, "trenta", {"a": 1, "b": 2}]
f = open("oggetti.dat", "wb")   # write-binary
pickle.dump(oggetti, f, protocol=0) #  protocol=0 => human readable,  protocol>0 => binary format
f.close()
```

OGGETTI.DAT

```
(lp0
I10
aI20
aVtrenta
p1
a(dp2
Va
p3
I1
sVb
p4
I2
sa.
```

Python - File di oggetti - serializzazione

- per recuperare il dato basta usare il metodo **load**

PYTHON

```
f = open("oggetti.dat", "rb")   # read-binary
oggetti = pickle.load(f)
f.close()
print(oggetti)
print(oggetti[3]["a"])
```

OUTPUT

```
[10, 20, 'trenta', {'a': 1, 'b': 2}]
1
```

Python - File di oggetti - serializzazione

- pickle funziona anche per tipi definiti da utente

PYTHON

```
class Studente:

    def __init__(self, nome, cognome, classe):
        self.__nome = nome;
        self.__cognome = cognome;
        self.__classe = classe;

    def __repr__(self):     # come toString() in Java o operator<< in C++
        return "{} {} [{}]".format(self.__nome, self.__cognome, self.__classe)

s1 = Studente("Giulio", "Angiani", "5C")
s2 = Studente("Antonella", "Catellani", "5D")
s3 = Studente("Alessandro", "Muzzini", "5C")
students = [s1, s2, s3]

print(students)

# salvo la lista in binary-mode su file "studenti.dat"
# uso il protocollo di default quindi avrò un file non huma-readable
f = open("studenti.dat", "wb")
pickle.dump(students, f)
f.close()
```

OUTPUT

```
[Giulio Angiani [5C], Antonella Catellani [5D], Alessandro Muzzini [5C]]
```

Python - File di oggetti - serializzazione

- il contenuto di studenti.dat è il seguente

STUDENTI.DAT

```
<80>^D<95>Î^@^@^@^@^@^@^@]<94>(<8c>^H__main__<94><8c>^HStudente<94><93><94>)<81><94>}<94>
(<8c>^O_Studente__nome<94><8c>^FGiulio<94><8c>^R_Studente__cognome<94><8c>^GAngiani<94>
<8c>^Q_Studente__classe<94><8c>^B5C<94>ubh^C)<81><94>}<94>(h^F<8c>
      Antonella<94>h^H<8c>    Catellani<94>h
<8c>^B5D<94>ubh^C)<81><94>}<94>(h^F<8c>
Alessandro<94>h^H<8c>^GMuzzini<94>h
h^Kube.
```

- posso recuperare i dati con

PYTHON

```
f1 = open("studenti.dat", "rb")
lista = pickle.load(f1)
f1.close()
print("Lista: ", lista)
```

OUTPUT

```
Lista:  [Giulio Angiani [5C], Antonella Catellani [5D], Alessandro Muzzini [5C]]
```

Python - Accesso a database

- python mette a disposizione librerie per ogni DBMS
  - MySQL : mysql
  - Sqlite : sqlite3
  - Oracle : cx_Oracle
  - ODBC : pyodbc
- vediamo un esempio con sqlite e uno con MySQL

```
import sqlite3  # libreria

db_filename = 'studenti.db'
conn = sqlite3.connect(db_filename)
conn.close()
```

Python e Sqlite

- lettura dati


```
import sqlite3  # libreria
db_filename = 'studenti.db'
conn = sqlite3.connect(db_filename)

sql = "select * from studenti"
cursor = conn.cursor()
cursor.execute(sql)
rows = cursor.fetchall()
for r in rows:
    print(r)
conn.close()
```

OUTPUT

```
(1, 'Giulio', 'Angiani', '5C')
(2, 'Antonella', 'Catellani', '5D')
(3, 'Alessandro', 'Muzzini', '5C')
```

Python e Sqlite

- inserimento e cancellazione dati


```
# inserimento
sql = "insert into studenti values (4, 'Barbara', 'Cattani', '5E')"
conn.executescript(sql)

# cancellazione
sql = "delete from studenti where matricola = 5"
conn.executescript(sql)
```

- righe come dizionario

PYTHON

```
conn.row_factory = sqlite3.Row  # tipo di cursore
sql = "select * from studenti"
cursor = conn.cursor()
cursor.execute(sql)
rows = cursor.fetchall()
for r in rows:
    print(dict(r))
```

OUTPUT

```
{'matricola': 1, 'nome': 'Giulio', 'cognome': 'Angiani', 'classe': '5C'}
{'matricola': 2, 'nome': 'Antonella', 'cognome': 'Catellani', 'classe': '5D'}
{'matricola': 3, 'nome': 'Alessandro', 'cognome': 'Muzzini', 'classe': '5C'}
{'matricola': 4, 'nome': 'Barbara', 'cognome': 'Cattani', 'classe': '5E'}
```

Python e MySQL

- per python3.x la libreria di riferimento è **mysql**
- per python2.x la libreria di riferimento è **MySQLdb**

- cambia solo il modo di connettersi al database perché non è un file


```
from mysql.connector import connection
conn = connection.MySQLConnection(user='scuola', password='scuola', host='127.0.0.1', database='scuola')
cursor = conn.cursor()
cursor.execute("select * from studenti")
for r in  cursor:
    print(r)
conn.close()
```

OUTPUT

```
(1, 'Giulio', 'Angiani', '5C')
(2, 'Antonella', 'Catellani', '5D')
(3, 'Alessandro', 'Muzzini', '5C')
```

Python e MySQL

- Per recuperate dati come dizionario specifico un parametro del cursore

PYTHON

```
cursor = cnx.cursor(dictionary=True)
cursor.execute("select * from studenti")
for r in  cursor:
    print(r)
```

OUTPUT

```
{'matricola': 1, 'nome': 'Giulio', 'cognome': 'Angiani', 'classe': '5C'}
{'matricola': 2, 'nome': 'Antonella', 'cognome': 'Catellani', 'classe': '5D'}
{'matricola': 3, 'nome': 'Alessandro', 'cognome': 'Muzzini', 'classe': '5C'}
```

Python - Accesso a risorse di rete

- python permette la connessione a risorse di rete tramite librerie per i vari protocolli
  - **http** : per il protocollo http/https
  - **ftplib**: per il protocollo ftp
  - **paramiko**: per il protocollo ssh
- esempio di url-retrieving

PYTHON SHELL

```
>>> h = http.client.HTTPConnection('127.0.0.1')
>>> h.request('GET', "/")
>>> r = h.getresponse()
>>> r.reason
'OK'
>>> r.status
200
>>> html = r.read()
>>> html
b'<html>\n\t<head>\n\t\t<title>AnzyWebServer : Online services</title>\n\t</head>\n\t\n\t
<body>\n\n\t<h1>Server up</h1>\n\t\n\t</body>\n</html>\n'
```

Python - Accesso a risorse di rete - FTP

```
>>> import ftplib
>>> from ftplib import FTP
>>> ftp = FTP('localhost')
>>> ftp.login(user='myuser', passwd='mypasswd')
>>> ftp.dir()
...
drwxrwxr-x   19 1000     1000         4096 Sep 02  2019 Calibre Library
-rw-rw-r--    1 1000     1000         1301 Apr 03 15:38 Diagramma1.dia
drwxr-xr-x   21 1000     1000         4096 Apr 14 09:18 Documenti
drwxr-xr-x   13 1000     1000         4096 Apr 29 12:20 Immagini
...
>>> ftp.close()
```

Giulio Angiani
I.I.S. "Blaise Pascal" - Reggio Emilia