Corso ITS:

PROGETTISTA E SVILUPPATORE SOFTWARE:

FULL STACK DEVELOPER E CLOUD SPECIALIST

Modulo: **Programmazione in Python**

Docente: *Andrea Ribuoli*

Mercoledì **30 Aprile** 2025

09:00 - 13:00

13:30 - 16:30

In [ ]:
```python
a = False
b = False
print(f"a = {a}, b = {b}, a and b = {a and b}")
a = True
b = False
print(f"a = {a}, b = {b}, a and b = {a and b}")
a = False
b = True
print(f"a = {a}, b = {b}, a and b = {a and b}")
a = True
b = True
print(f"a = {a}, b = {b}, a and b = {a and b}")
```

a	b	a and b
0	0	0
0	1	0
1	0	0
1	1	1

In [ ]:
```
a = False
b = False
print(f"a = {a}, b = {b}, not (a or b) = {not (a or b)}")
a = True
b = False
print(f"a = {a}, b = {b}, not (a or b) = {not (a or b)}")
a = False
b = True
print(f"a = {a}, b = {b}, not (a or b) = {not (a or b)}")
a = True
b = True
print(f"a = {a}, b = {b}, not (a or b) = {not (a or b)}")
```

a	b	a or b
0	0	0
0	1	1
1	0	1
1	1	1

In [ ]:
```
a = False
b = False
print(f"a = {a}, b = {b}, (not a) and (not b) = {(not a) and (not b)}")
a = True
b = False
print(f"a = {a}, b = {b}, (not a) and (not b) = {(not a) and (not b)}")
a = False
b = True
print(f"a = {a}, b = {b}, (not a) and (not b) = {(not a) and (not b)}")
a = True
b = True
print(f"a = {a}, b = {b}, (not a) and (not b) = {(not a) and (not b)}")
```

In [ ]:
```
a = False
b = False
print(f"a = {a}, b = {b}, not(a or b) = {not (a or b)}")
a = True
b = False
print(f"a = {a}, b = {b}, not(a or b) = {not (a or b)}")
a = False
b = True
print(f"a = {a}, b = {b}, not(a or b) = {not (a or b)}")
a = True
b = True
print(f"a = {a}, b = {b}, not(a or b) = {not (a or b)}")
```

In [ ]:
```
print("not(a or b) <== identiche ==> (not a) and (not b)")
print("not(a and b) <== identiche ==> (not a) or (not b)")
```

leggi di MORGAN

In [ ]:
```
import urllib.request
url = "https://www.andrearibuoli.it"
risultato = urllib.request.urlopen(url)
theBytes = risultato.read()
text = theBytes.decode()
import bs4
doc = bs4.BeautifulSoup(text)
print(doc.prettify())
```

spigolatura.py

pip install bs4

In [ ]:
```
!python3 resoconto.py spigolatura
```

In [ ]:
```
lista = dir(bs4.element.Tag)
for met in lista:
    if not met.startswith('__'):
        print(met)
```

In [ ]:
```
html = """<html>
 <head>
  <title>
    CORSO PYTHON
  </title>
 </head>
 <body>
  <p>
    Facciamo un primo esempio
  </p>
  <p>
    di pagina HTML remota
    <a href="https://www.andrearibuoli.it">
      link
    </a>
  </p>
  <p>
    composta di paragrafi.
  </p>
 </body>
</html>
"""
```

In [ ]:
```
print(html)
```

In [ ]:
```
html_compatto = "".join([x.strip() for x in html.split("\n")])
```

In [ ]:
```
print(html_compatto)
```

In [ ]:
```
import bs4
doc = bs4.BeautifulSoup(html_compatto)
```

In [ ]:
```
def naviga2(tag, indent) :
    print(indent + tag.name.upper())
    if tag.name.upper() == "A":
        print(tag.get("href"))
    for stag in tag.contents:
        if type(stag) == bs4.element.Tag :
            naviga2(stag, indent + " ")
```

In [ ]:
```
def naviga3(tag) :
    if tag.name.upper() == "A":
        print(tag.get("href"))
    for stag in tag.contents:
        if type(stag) == bs4.element.Tag :
            naviga3(stag)
```

In [ ]:
```
def naviga4(tag) :
    if tag.name.upper() == "A":
        print(tag["href"])
    for stag in tag.contents:
        if type(stag) == bs4.element.Tag :
            naviga4(stag)
```

In [ ]:
```
naviga4(doc.contents[0])
```

dipendenze.py

In [ ]:
```
import urllib.request
import bs4

def main():
    url = input("URL della pagina di partenza: ")
    visited = []
    depth = 4
    crawl(url, depth, visited)

def crawl(url, depth, visited) :
    if depth == 0:
        return
    response = urllib.request.urlopen(url)
    doc = bs4.BeautifulSoup(response)
    print(f"Sto visitando il percorso: '{url}'")
    try:
        for link in doc.find_all("a"):
            href = link["href"]
            if href[0:4] == "http" and href not in visited :
```

```
                visited.append(href)
                crawl(href, depth - 1, visited)
    except:
        return

main()
```

In [ ]:
```
import urllib.request
url = "https://www.comuni-italiani.it/province.html"
response = urllib.request.urlopen(url)
theBytes = response.read()
```

In [ ]:
```
text = theBytes.decode()
```

In [ ]:
```
text = theBytes.decode(encoding="iso-8859-1")
```

In [ ]:
```
print(text[0:600])
```

In [ ]:
```
import bs4
doc = bs4.BeautifulSoup(text)
elems = doc.find_all("table")
```

In [ ]:
```
len(elems)
```

In [ ]:
```
type(elems[6])
```

In [ ]:
```
print(elems[3])
```

In [ ]:
```
table = elems[3]
```

In [ ]:
```
import urllib.request
url = "https://www.comuni-italiani.it/province.html"
response = urllib.request.urlopen(url)
theBytes = response.read()
text = theBytes.decode(encoding="iso-8859-1")

import bs4
doc = bs4.BeautifulSoup(text)
elems = doc.find_all("table")
table = elems[3]

def naviga2(tag, indent) :
    print(indent + tag.name.upper())
    for stag in tag.contents:
        if type(stag) == bs4.element.Tag :
            naviga2(stag, indent + " ")

naviga2(table, "")
```

In [ ]:
```
import urllib.request
url = "https://www.comuni-italiani.it/province.html"
response = urllib.request.urlopen(url)
theBytes = response.read()
```

```
text = theBytes.decode(encoding="iso-8859-1")

import bs4
doc = bs4.BeautifulSoup(text)
elems = doc.find_all("table")
table = elems[3]
for tr in table.contents[2:-1]:
    if type(tr) == bs4.element.Tag :
        tds = tr.contents
        print(tds[7])
```

In [9]:
```
import urllib.request
url = "https://www.comuni-italiani.it/province.html"
response = urllib.request.urlopen(url)
theBytes = response.read()
text = theBytes.decode(encoding="iso-8859-1")

import bs4
doc = bs4.BeautifulSoup(text)
elems = doc.find_all("table")
table = elems[3]
for tr in table.contents[2:-2]:
    if type(tr) == bs4.element.Tag :
        tds = tr.contents
#        sequ = tds[0].get_text()
#        prov = tds[1].get_text()
#        resi = tds[2].get_text()
#        sigl = tds[7].get_text()
#        print(sequ, prov, resi, sigl, sep=" - ")
        sequ = int(tds[0].get_text())
        prov =     tds[1].get_text()
        resi = int(tds[2].get_text().replace(".",""))
        sigl =     tds[7].get_text()
        print(f"{sequ:3d} {prov:25s} {resi:9d} {sigl}")
```

1	Agrigento	442049	AG
2	Alessandria	426658	AL
3	Ancona	474124	AN
4	Aosta	126883	AO
5	Arezzo	344374	AR
6	Ascoli Piceno	209450	AP
7	Asti	216677	AT
8	Avellino	423506	AV
9	Bari	1260142	BA
10	Barletta-Andria-Trani	392546	BT
11	Belluno	205781	BL
12	Benevento	279675	BN
13	Bergamo	1109933	BG
14	Biella	178551	BI
15	Bologna	1009210	BO
16	Bolzano	524256	BZ
17	Brescia	1262318	BS
18	Brindisi	397083	BR
19	Cagliari	560373	CA
20	Caltanissetta	269710	CL
21	Campobasso	224644	CB
22	Carbonia-Iglesias	126324	CI
23	Caserta	924166	CE
24	Catania	1113303	CT
25	Catanzaro	362343	CZ
26	Chieti	389169	CH
27	Como	600190	CO
28	Cosenza	711739	CS
29	Cremona	359388	CR
30	Crotone	175566	KR
31	Cuneo	589108	CN
32	Enna	168052	EN
33	Fermo	174849	FM
34	Ferrara	348362	FE
35	Firenze	1014423	FI
36	Foggia	628556	FG
37	Forlì-Cesena	394067	FC
38	Frosinone	493067	FR
39	Genova	850071	GE
40	Gorizia	139673	GO
41	Grosseto	223045	GR
42	Imperia	215130	IM
43	Isernia	85805	IS
44	La Spezia	220698	SP
45	L'Aquila	301910	AQ
46	Latina	574891	LT
47	Lecce	802082	LE
48	Lecco	339238	LC
49	Livorno	337334	LI
50	Lodi	229338	LO
51	Lucca	390042	LU
52	Macerata	318921	MC
53	Mantova	412610	MN
54	Massa-Carrara	196580	MS
55	Matera	199685	MT
56	Messina	636653	ME

```
 57 Milano                    3218201 MI
 58 Modena                     700862 MO
 59 Monza e della Brianza      868859 MB
 60 Napoli                    3107006 NA
 61 Novara                     370143 NO
 62 Nuoro                      156096 NU
 63 Olbia-Tempio               160672 OT
 64 Oristano                   160746 OR
 65 Padova                     936274 PD
 66 Palermo                   1268217 PA
 67 Parma                      448899 PR
 68 Pavia                      547251 PV
 69 Perugia                    660690 PG
 70 Pesaro e Urbino            360711 PU
 71 Pescara                    321309 PE
 72 Piacenza                   286758 PC
 73 Pisa                       421851 PI
 74 Pistoia                    291839 PT
 75 Pordenone                  312051 PN
 76 Potenza                    370680 PZ
 77 Prato                      254608 PO
 78 Ragusa                     321359 RG
 79 Ravenna                    391414 RA
 80 Reggio Calabria            553861 RC
 81 Reggio Emilia              532483 RE
 82 Rieti                      157420 RI
 83 Rimini                     336786 RN
 84 Roma                      4353738 RM
 85 Rovigo                     238588 RO
 86 Salerno                   1104731 SA
 87 Medio Campidano             98623 VS
 88 Sassari                    333116 SS
 89 Savona                     279408 SV
 90 Siena                      268341 SI
 91 Siracusa                   402822 SR
 92 Sondrio                    181437 SO
 93 Taranto                    583479 TA
 94 Teramo                     309859 TE
 95 Terni                      228218 TR
 96 Torino                    2277857 TO
 97 Ogliastra                   57185 OG
 98 Trapani                    434476 TP
 99 Trento                     538604 TN
100 Treviso                    885972 TV
101 Trieste                    234682 TS
102 Udine                      531466 UD
103 Varese                     890043 VA
104 Venezia                    854275 VE
105 Verbano-Cusio-Ossola       159664 VB
106 Vercelli                   173868 VC
107 Verona                     921557 VR
108 Vibo Valentia              161619 VV
109 Vicenza                    865082 VI
110 Viterbo                    319008 VT
```

estrai.py modificare il programma (su PDF) in moda da: * non estrarre più il numero di sequenza * emettere in ordine: sigla, nome provincia, abitanti, kmq * ricalcolare la densità per kmq e confrontarla col valore in tabella segnalando eventali divergenze

In [11]:
```
!python3 resoconto.py estrai
```

```
 1  Mirco  Azzolini
 2  Wallace  Bezerra Beretta
 3  Alexandru Razvan  Brasovianu
 4  Edoardo  Caprini
 5  Maryuri  Catozzi
 6  Federico  De Grandis
 7  Maikol  Freddari
 8  Sofia  Gaona
 9  Alessia  Gasparini
10  Enrico  Giorgi
11  Andrea  Kanakciu
12  Francesco  Marinelli
13  Filippo  Martino
14  Eleonora  Moroni
15  Norman  Muzi
16  Mattia  Roberti
17  Alessandro  Rovinelli
18  Davide  Sambughi
19  Maximiliano  Serafini
20  Giovanni  Sperandini
21  Alessio  Stomeo
22  Lesly Pierina  Vera Castillejo
```

---

utilizzo del supposto standard del formato CSV

In [ ]:
```
infile = open("province.csv", "r")
from csv import reader
csvReader = reader(infile)
for row in csvReader:
    print(row)
infile.close()
```

In [33]:
```
outfile = open("province.csv", "w")
from csv import writer
csvWriter = writer(outfile)
csvWriter.writerow(['sigla', 'provincia', 'residenti', 'kmq'])
csvWriter.writerow(['PU', 'Pesaro e Urbino', 100000.0, 1000.0])
outfile.close()
```

In [38]:
```
import urllib.request
url = "https://www.comuni-italiani.it/province.html"
response = urllib.request.urlopen(url)
theBytes = response.read()
text = theBytes.decode(encoding="iso-8859-1")
outfile = open("province.csv", "w")
from csv import writer
csvWriter = writer(outfile)
csvWriter.writerow(['sigla', 'provincia', 'residenti', 'kmq'])
import bs4
doc = bs4.BeautifulSoup(text)
```

```
elems = doc.find_all("table")
table = elems[3]
for tr in table.contents[2:-2]:
    if type(tr) == bs4.element.Tag :
        tds = tr.contents
        sequ = int(tds[0].get_text())
        prov = tds[1].get_text()
        resi = int(tds[2].get_text().replace(".",""))
        sigl = tds[7].get_text()
        kmq = int(tds[4].get_text().replace(".",""))
        denso = float(tds[5].get_text().replace(".","").replace(",","."))
        densc = round(resi / kmq, 1)
        csvWriter.writerow([sigl, prov, resi, kmq])
outfile.close()
```

---

utilizzo di archiviazione nativa con **pickle**

In [41]:
```
import pickle
lista = [1,2,3,4]
backup = open("province", "wb")
pickle.dump(lista, backup)
backup.close()
```

In [43]:
```
import pickle
lista = [1,2,3,4,5]
with open("province", "wb") as backup :
    pickle.dump(lista, backup)
```

In [44]:
```
import pickle
backup = open("province", "rb")
lista = pickle.load(backup)
backup.close()
print(lista)
```

```
[1, 2, 3, 4, 5]
```

In [45]:
```
import pickle
with open("province", "rb") as backup:
    lista = pickle.load(backup)
    print(lista)
```

```
[1, 2, 3, 4, 5]
```

In [46]:
```
import urllib.request
url = "https://www.comuni-italiani.it/province.html"
response = urllib.request.urlopen(url)
theBytes = response.read()
text = theBytes.decode(encoding="iso-8859-1")
lista = []
import bs4
doc = bs4.BeautifulSoup(text)
elems = doc.find_all("table")
table = elems[3]
```

```
for tr in table.contents[2:-2]:
    if type(tr) == bs4.element.Tag :
        tds = tr.contents
        sequ = int(tds[0].get_text())
        prov = tds[1].get_text()
        resi = int(tds[2].get_text().replace(".",""))
        sigl = tds[7].get_text()
        kmq = int(tds[4].get_text().replace(".",""))
        denso = float(tds[5].get_text().replace(".","").replace(",","."))
        densc = round(resi / kmq, 1)
        lista.append([sigl, prov, resi, kmq])

import pickle
with open("province", "wb") as backup :
    pickle.dump(lista, backup)
```

In [47]:
```
import pickle
with open("province", "rb") as backup:
    lista = pickle.load(backup)
    print(lista)
```

```
[['AG', 'Agrigento', 442049, 3042], ['AL', 'Alessandria', 426658, 3562], ['A
N', 'Ancona', 474124, 1940], ['AO', 'Aosta', 126883, 3263], ['AR', 'Arezzo',
344374, 3235], ['AP', 'Ascoli Piceno', 209450, 1228], ['AT', 'Asti', 216677,
1511], ['AV', 'Avellino', 423506, 2792], ['BA', 'Bari', 1260142, 3825], ['B
T', 'Barletta-Andria-Trani', 392546, 1539], ['BL', 'Belluno', 205781, 3678],
['BN', 'Benevento', 279675, 2071], ['BG', 'Bergamo', 1109933, 2723], ['BI',
'Biella', 178551, 914], ['BO', 'Bologna', 1009210, 3703], ['BZ', 'Bolzano',
524256, 7400], ['BS', 'Brescia', 1262318, 4784], ['BR', 'Brindisi', 397083,
1839], ['CA', 'Cagliari', 560373, 4570], ['CL', 'Caltanissetta', 269710, 212
5], ['CB', 'Campobasso', 224644, 2909], ['CI', 'Carbonia-Iglesias', 126324,
1495], ['CE', 'Caserta', 924166, 2639], ['CT', 'Catania', 1113303, 3552],
['CZ', 'Catanzaro', 362343, 2391], ['CH', 'Chieti', 389169, 2588], ['CO', 'C
omo', 600190, 1288], ['CS', 'Cosenza', 711739, 6650], ['CR', 'Cremona', 3593
88, 1771], ['KR', 'Crotone', 175566, 1717], ['CN', 'Cuneo', 589108, 6903],
['EN', 'Enna', 168052, 2562], ['FM', 'Fermo', 174849, 860], ['FE', 'Ferrara'
, 348362, 2631], ['FI', 'Firenze', 1014423, 3514], ['FG', 'Foggia', 628556,
6971], ['FC', 'Forlì-Cesena', 394067, 2377], ['FR', 'Frosinone', 493067, 324
4], ['GE', 'Genova', 850071, 1839], ['GO', 'Gorizia', 139673, 466], ['GR', '
Grosseto', 223045, 4504], ['IM', 'Imperia', 215130, 1156], ['IS', 'Isernia',
85805, 1524], ['SP', 'La Spezia', 220698, 881], ['AQ', "L'Aquila", 301910, 5
034], ['LT', 'Latina', 574891, 2251], ['LE', 'Lecce', 802082, 2759], ['LC',
'Lecco', 339238, 816], ['LI', 'Livorno', 337334, 1211], ['LO', 'Lodi', 22933
8, 782], ['LU', 'Lucca', 390042, 1773], ['MC', 'Macerata', 318921, 2774],
['MN', 'Mantova', 412610, 2339], ['MS', 'Massa-Carrara', 196580, 1156], ['M
T', 'Matera', 199685, 3446], ['ME', 'Messina', 636653, 3247], ['MI', 'Milano
', 3218201, 1579], ['MO', 'Modena', 700862, 2689], ['MB', 'Monza e della Bri
anza', 868859, 405], ['NA', 'Napoli', 3107006, 1171], ['NO', 'Novara', 37014
3, 1339], ['NU', 'Nuoro', 156096, 3934], ['OT', 'Olbia-Tempio', 160672, 339
9], ['OR', 'Oristano', 160746, 3040], ['PD', 'Padova', 936274, 2142], ['PA',
'Palermo', 1268217, 4992], ['PR', 'Parma', 448899, 3449], ['PV', 'Pavia', 54
7251, 2965], ['PG', 'Perugia', 660690, 6334], ['PU', 'Pesaro e Urbino', 3607
11, 2564], ['PE', 'Pescara', 321309, 1225], ['PC', 'Piacenza', 286758, 258
9], ['PI', 'Pisa', 421851, 2444], ['PT', 'Pistoia', 291839, 965], ['PN', 'Po
rdenone', 312051, 2273], ['PZ', 'Potenza', 370680, 6548], ['PO', 'Prato', 25
4608, 365], ['RG', 'Ragusa', 321359, 1614], ['RA', 'Ravenna', 391414, 1858],
['RC', 'Reggio Calabria', 553861, 3183], ['RE', 'Reggio Emilia', 532483, 229
3], ['RI', 'Rieti', 157420, 2749], ['RN', 'Rimini', 336786, 867], ['RM', 'Ro
ma', 4353738, 5352], ['RO', 'Rovigo', 238588, 1790], ['SA', 'Salerno', 11047
31, 4917], ['VS', 'Medio Campidano', 98623, 1516], ['SS', 'Sassari', 333116,
4282], ['SV', 'Savona', 279408, 1545], ['SI', 'Siena', 268341, 3821], ['SR',
'Siracusa', 402822, 2109], ['SO', 'Sondrio', 181437, 3212], ['TA', 'Taranto'
, 583479, 2437], ['TE', 'Teramo', 309859, 1948], ['TR', 'Terni', 228218, 212
2], ['TO', 'Torino', 2277857, 6831], ['OG', 'Ogliastra', 57185, 1854], ['T
P', 'Trapani', 434476, 2460], ['TN', 'Trento', 538604, 6207], ['TV', 'Trevis
o', 885972, 2477], ['TS', 'Trieste', 234682, 212], ['UD', 'Udine', 531466, 4
904], ['VA', 'Varese', 890043, 1199], ['VE', 'Venezia', 854275, 2462], ['V
B', 'Verbano-Cusio-Ossola', 159664, 2255], ['VC', 'Vercelli', 173868, 2088],
['VR', 'Verona', 921557, 3121], ['VV', 'Vibo Valentia', 161619, 1139], ['V
I', 'Vicenza', 865082, 2723], ['VT', 'Viterbo', 319008, 3612]]
```

---

utilizzo di archiviazione tramite **Pandas**

In [62]:
```
import pandas as pd
```

```
df = pd.read_csv("province.csv")
```

In [63]:
```
df
```

Out[63]:

	Unnamed: 0	sigla	provincia	residenti	kmq
0	0	AG	Agrigento	442049	3042
1	1	AL	Alessandria	426658	3562
2	2	AN	Ancona	474124	1940
3	3	AO	Aosta	126883	3263
4	4	AR	Arezzo	344374	3235
...	...	...	...	...	...
105	105	VC	Vercelli	173868	2088
106	106	VR	Verona	921557	3121
107	107	VV	Vibo Valentia	161619	1139
108	108	VI	Vicenza	865082	2723
109	109	VT	Viterbo	319008	3612

110 rows × 5 columns

In [57]:
```
df[["provincia", "sigla"]]
```

Out[57]:

	provincia	sigla
0	Agrigento	AG
1	Alessandria	AL
2	Ancona	AN
3	Aosta	AO
4	Arezzo	AR
...	...	...
105	Vercelli	VC
106	Verona	VR
107	Vibo Valentia	VV
108	Vicenza	VI
109	Viterbo	VT

110 rows × 2 columns

---

utilizzo di archiviazione tramite **json**

---

utilizzo di archiviazione tramite **xml**

In [ ]:
```
BeautifulSoup(text, "html.parser")
```

---

utilizzo di archiviazione tramite database

MySQL

In [52]:
```
!grep "2" test_grep.txt
```

```
come 23 o
25
```

In [54]:
```
from re import split
with open("test_grep.txt") as infile:
    for riga in infile.readlines():
        print(split("[0-9]+", riga))
```

```
['Oggi voglio dimostrare\n']
['la possibilità di identificare\n']
['le righe di ', ' file\n']
['che contengono numeri\n']
['come ', ' o \n']
['', '\n']
['indicando le righe\n']
['in cui ciò avviene']
```