Corso ITS: *ARTIFICIAL INTELLIGENCE SPECIALIST*

Modulo: **Programmazione ad oggetti in Python e librerie esterne**

Docente: *Andrea Ribuoli*

Mercoledì **26 Febbraio** 2025

09:00 - 14:00

In [1]:
```
import urllib.request
address ='https://www.comuni-italiani.it/province.html'
response = urllib.request.urlopen(address)
theBytes = response.read()
html = theBytes.decode(encoding='iso-8859-1')
from bs4 import BeautifulSoup
t1 = BeautifulSoup(html, "html.parser").table
t2 = t1.find_next("table")
t3 = t2.find_next("table")
t4 = t3.find_next("table")
riga = t4.find_next("tr")
riga = riga.find_next("tr")
import pandas as pd
province = pd.DataFrame(columns = ["sigla", "nome", "abitanti"])
i = 0
while riga != None :
    tdx  = riga.find_next("td")
    tdx  = tdx.find_next("td")
    prov = tdx.get_text()
    tdx  = tdx.find_next("td")
    abit = int(tdx.get_text().replace(".",""))
    tdx  = tdx.find_next("td")
    tdx  = tdx.find_next("td")
    tdx  = tdx.find_next("td")
    tdx  = tdx.find_next("td")
    tdx  = tdx.find_next("td")
    sigla = tdx.get_text()
    province.loc[i] = [sigla, prov, abit]
    if sigla == 'VT' : break
    i += 1
    riga = riga.find_next("tr")
province.sort_values(by="abitanti", ascending=False).to_csv("province.csv",
province.to_json("province.json")
```

In [2]:
```
## pip3 install openpyxl
province.to_excel("province.xlsx", sheet_name="Province Italiane", index=Fal
```

In [3]:
```
import pandas as pd
df = pd.DataFrame(columns = ["nome", "cognome", "anno_di_corso"])
df.to_excel("allievi.xlsx", sheet_name="Allievi", index=False)
```

In [4]:
```
from csv import reader
infile = open("province.csv")
csvReader = reader(infile)
for riga in csvReader :
    print(riga)
```

```
['sigla', 'nome', 'abitanti']
['RM', 'Roma', '4353738']
['MI', 'Milano', '3218201']
['NA', 'Napoli', '3107006']
['TO', 'Torino', '2277857']
['PA', 'Palermo', '1268217']
['BS', 'Brescia', '1262318']
['BA', 'Bari', '1260142']
['CT', 'Catania', '1113303']
['BG', 'Bergamo', '1109933']
['SA', 'Salerno', '1104731']
['FI', 'Firenze', '1014423']
['BO', 'Bologna', '1009210']
['PD', 'Padova', '936274']
['CE', 'Caserta', '924166']
['VR', 'Verona', '921557']
['VA', 'Varese', '890043']
['TV', 'Treviso', '885972']
['MB', 'Monza e della Brianza', '868859']
['VI', 'Vicenza', '865082']
['VE', 'Venezia', '854275']
['GE', 'Genova', '850071']
['LE', 'Lecce', '802082']
['CS', 'Cosenza', '711739']
['MO', 'Modena', '700862']
['PG', 'Perugia', '660690']
['ME', 'Messina', '636653']
['FG', 'Foggia', '628556']
['CO', 'Como', '600190']
['CN', 'Cuneo', '589108']
['TA', 'Taranto', '583479']
['LT', 'Latina', '574891']
['CA', 'Cagliari', '560373']
['RC', 'Reggio Calabria', '553861']
['PV', 'Pavia', '547251']
['TN', 'Trento', '538604']
['RE', 'Reggio Emilia', '532483']
['UD', 'Udine', '531466']
['BZ', 'Bolzano', '524256']
['FR', 'Frosinone', '493067']
['AN', 'Ancona', '474124']
['PR', 'Parma', '448899']
['AG', 'Agrigento', '442049']
['TP', 'Trapani', '434476']
['AL', 'Alessandria', '426658']
['AV', 'Avellino', '423506']
['PI', 'Pisa', '421851']
['MN', 'Mantova', '412610']
['SR', 'Siracusa', '402822']
['BR', 'Brindisi', '397083']
['FC', 'Forlì-Cesena', '394067']
['BT', 'Barletta-Andria-Trani', '392546']
['RA', 'Ravenna', '391414']
['LU', 'Lucca', '390042']
['CH', 'Chieti', '389169']
['PZ', 'Potenza', '370680']
```

```
['NO', 'Novara', '370143']
['CZ', 'Catanzaro', '362343']
['PU', 'Pesaro e Urbino', '360711']
['CR', 'Cremona', '359388']
['FE', 'Ferrara', '348362']
['AR', 'Arezzo', '344374']
['LC', 'Lecco', '339238']
['LI', 'Livorno', '337334']
['RN', 'Rimini', '336786']
['SS', 'Sassari', '333116']
['RG', 'Ragusa', '321359']
['PE', 'Pescara', '321309']
['VT', 'Viterbo', '319008']
['MC', 'Macerata', '318921']
['PN', 'Pordenone', '312051']
['TE', 'Teramo', '309859']
['AQ', "L'Aquila", '301910']
['PT', 'Pistoia', '291839']
['PC', 'Piacenza', '286758']
['BN', 'Benevento', '279675']
['SV', 'Savona', '279408']
['CL', 'Caltanissetta', '269710']
['SI', 'Siena', '268341']
['PO', 'Prato', '254608']
['RO', 'Rovigo', '238588']
['TS', 'Trieste', '234682']
['LO', 'Lodi', '229338']
['TR', 'Terni', '228218']
['CB', 'Campobasso', '224644']
['GR', 'Grosseto', '223045']
['SP', 'La Spezia', '220698']
['AT', 'Asti', '216677']
['IM', 'Imperia', '215130']
['AP', 'Ascoli Piceno', '209450']
['BL', 'Belluno', '205781']
['MT', 'Matera', '199685']
['MS', 'Massa-Carrara', '196580']
['SO', 'Sondrio', '181437']
['BI', 'Biella', '178551']
['KR', 'Crotone', '175566']
['FM', 'Fermo', '174849']
['VC', 'Vercelli', '173868']
['EN', 'Enna', '168052']
['VV', 'Vibo Valentia', '161619']
['OR', 'Oristano', '160746']
['OT', 'Olbia-Tempio', '160672']
['VB', 'Verbano-Cusio-Ossola', '159664']
['RI', 'Rieti', '157420']
['NU', 'Nuoro', '156096']
['GO', 'Gorizia', '139673']
['AO', 'Aosta', '126883']
['CI', 'Carbonia-Iglesias', '126324']
['VS', 'Medio Campidano', '98623']
['IS', 'Isernia', '85805']
['OG', 'Ogliastra', '57185']
```

In [5]:
```
import json
infile = open("province.json")
stringone = infile.read()
data = json.loads(stringone)
data
```

Out[5]:

```
{'sigla': {'0': 'AG',
           '1': 'AL',
           '2': 'AN',
           '3': 'AO',
           '4': 'AR',
           '5': 'AP',
           '6': 'AT',
           '7': 'AV',
           '8': 'BA',
           '9': 'BT',
           '10': 'BL',
           '11': 'BN',
           '12': 'BG',
           '13': 'BI',
           '14': 'BO',
           '15': 'BZ',
           '16': 'BS',
           '17': 'BR',
           '18': 'CA',
           '19': 'CL',
           '20': 'CB',
           '21': 'CI',
           '22': 'CE',
           '23': 'CT',
           '24': 'CZ',
           '25': 'CH',
           '26': 'CO',
           '27': 'CS',
           '28': 'CR',
           '29': 'KR',
           '30': 'CN',
           '31': 'EN',
           '32': 'FM',
           '33': 'FE',
           '34': 'FI',
           '35': 'FG',
           '36': 'FC',
           '37': 'FR',
           '38': 'GE',
           '39': 'GO',
           '40': 'GR',
           '41': 'IM',
           '42': 'IS',
           '43': 'SP',
           '44': 'AQ',
           '45': 'LT',
           '46': 'LE',
           '47': 'LC',
           '48': 'LI',
           '49': 'LO',
           '50': 'LU',
           '51': 'MC',
           '52': 'MN',
           '53': 'MS',
           '54': 'MT',
           '55': 'ME',
```

```
 '56': 'MI',
 '57': 'MO',
 '58': 'MB',
 '59': 'NA',
 '60': 'NO',
 '61': 'NU',
 '62': 'OT',
 '63': 'OR',
 '64': 'PD',
 '65': 'PA',
 '66': 'PR',
 '67': 'PV',
 '68': 'PG',
 '69': 'PU',
 '70': 'PE',
 '71': 'PC',
 '72': 'PI',
 '73': 'PT',
 '74': 'PN',
 '75': 'PZ',
 '76': 'PO',
 '77': 'RG',
 '78': 'RA',
 '79': 'RC',
 '80': 'RE',
 '81': 'RI',
 '82': 'RN',
 '83': 'RM',
 '84': 'RO',
 '85': 'SA',
 '86': 'VS',
 '87': 'SS',
 '88': 'SV',
 '89': 'SI',
 '90': 'SR',
 '91': 'SO',
 '92': 'TA',
 '93': 'TE',
 '94': 'TR',
 '95': 'TO',
 '96': 'OG',
 '97': 'TP',
 '98': 'TN',
 '99': 'TV',
 '100': 'TS',
 '101': 'UD',
 '102': 'VA',
 '103': 'VE',
 '104': 'VB',
 '105': 'VC',
 '106': 'VR',
 '107': 'VV',
 '108': 'VI',
 '109': 'VT'},
'nome': {'0': 'Agrigento',
 '1': 'Alessandria',
```

```
'2': 'Ancona',
'3': 'Aosta',
'4': 'Arezzo',
'5': 'Ascoli Piceno',
'6': 'Asti',
'7': 'Avellino',
'8': 'Bari',
'9': 'Barletta-Andria-Trani',
'10': 'Belluno',
'11': 'Benevento',
'12': 'Bergamo',
'13': 'Biella',
'14': 'Bologna',
'15': 'Bolzano',
'16': 'Brescia',
'17': 'Brindisi',
'18': 'Cagliari',
'19': 'Caltanissetta',
'20': 'Campobasso',
'21': 'Carbonia-Iglesias',
'22': 'Caserta',
'23': 'Catania',
'24': 'Catanzaro',
'25': 'Chieti',
'26': 'Como',
'27': 'Cosenza',
'28': 'Cremona',
'29': 'Crotone',
'30': 'Cuneo',
'31': 'Enna',
'32': 'Fermo',
'33': 'Ferrara',
'34': 'Firenze',
'35': 'Foggia',
'36': 'Forlì-Cesena',
'37': 'Frosinone',
'38': 'Genova',
'39': 'Gorizia',
'40': 'Grosseto',
'41': 'Imperia',
'42': 'Isernia',
'43': 'La Spezia',
'44': "L'Aquila",
'45': 'Latina',
'46': 'Lecce',
'47': 'Lecco',
'48': 'Livorno',
'49': 'Lodi',
'50': 'Lucca',
'51': 'Macerata',
'52': 'Mantova',
'53': 'Massa-Carrara',
'54': 'Matera',
'55': 'Messina',
'56': 'Milano',
'57': 'Modena',
```

```
 '58': 'Monza e della Brianza',
 '59': 'Napoli',
 '60': 'Novara',
 '61': 'Nuoro',
 '62': 'Olbia-Tempio',
 '63': 'Oristano',
 '64': 'Padova',
 '65': 'Palermo',
 '66': 'Parma',
 '67': 'Pavia',
 '68': 'Perugia',
 '69': 'Pesaro e Urbino',
 '70': 'Pescara',
 '71': 'Piacenza',
 '72': 'Pisa',
 '73': 'Pistoia',
 '74': 'Pordenone',
 '75': 'Potenza',
 '76': 'Prato',
 '77': 'Ragusa',
 '78': 'Ravenna',
 '79': 'Reggio Calabria',
 '80': 'Reggio Emilia',
 '81': 'Rieti',
 '82': 'Rimini',
 '83': 'Roma',
 '84': 'Rovigo',
 '85': 'Salerno',
 '86': 'Medio Campidano',
 '87': 'Sassari',
 '88': 'Savona',
 '89': 'Siena',
 '90': 'Siracusa',
 '91': 'Sondrio',
 '92': 'Taranto',
 '93': 'Teramo',
 '94': 'Terni',
 '95': 'Torino',
 '96': 'Ogliastra',
 '97': 'Trapani',
 '98': 'Trento',
 '99': 'Treviso',
 '100': 'Trieste',
 '101': 'Udine',
 '102': 'Varese',
 '103': 'Venezia',
 '104': 'Verbano-Cusio-Ossola',
 '105': 'Vercelli',
 '106': 'Verona',
 '107': 'Vibo Valentia',
 '108': 'Vicenza',
 '109': 'Viterbo'},
'abitanti': {'0': 442049,
 '1': 426658,
 '2': 474124,
 '3': 126883,
```

```
'4': 344374,
'5': 209450,
'6': 216677,
'7': 423506,
'8': 1260142,
'9': 392546,
'10': 205781,
'11': 279675,
'12': 1109933,
'13': 178551,
'14': 1009210,
'15': 524256,
'16': 1262318,
'17': 397083,
'18': 560373,
'19': 269710,
'20': 224644,
'21': 126324,
'22': 924166,
'23': 1113303,
'24': 362343,
'25': 389169,
'26': 600190,
'27': 711739,
'28': 359388,
'29': 175566,
'30': 589108,
'31': 168052,
'32': 174849,
'33': 348362,
'34': 1014423,
'35': 628556,
'36': 394067,
'37': 493067,
'38': 850071,
'39': 139673,
'40': 223045,
'41': 215130,
'42': 85805,
'43': 220698,
'44': 301910,
'45': 574891,
'46': 802082,
'47': 339238,
'48': 337334,
'49': 229338,
'50': 390042,
'51': 318921,
'52': 412610,
'53': 196580,
'54': 199685,
'55': 636653,
'56': 3218201,
'57': 700862,
'58': 868859,
'59': 3107006,
```

```
'60': 370143,
'61': 156096,
'62': 160672,
'63': 160746,
'64': 936274,
'65': 1268217,
'66': 448899,
'67': 547251,
'68': 660690,
'69': 360711,
'70': 321309,
'71': 286758,
'72': 421851,
'73': 291839,
'74': 312051,
'75': 370680,
'76': 254608,
'77': 321359,
'78': 391414,
'79': 553861,
'80': 532483,
'81': 157420,
'82': 336786,
'83': 4353738,
'84': 238588,
'85': 1104731,
'86': 98623,
'87': 333116,
'88': 279408,
'89': 268341,
'90': 402822,
'91': 181437,
'92': 583479,
'93': 309859,
'94': 228218,
'95': 2277857,
'96': 57185,
'97': 434476,
'98': 538604,
'99': 885972,
'100': 234682,
'101': 531466,
'102': 890043,
'103': 854275,
'104': 159664,
'105': 173868,
'106': 921557,
'107': 161619,
'108': 865082,
'109': 319008}}
```

Pag	Approfondimento
82	SymPy
123	email

Pag	Approfondimento
149	matplotlib
319	Turtle Graphics
377	SciPy
434	csv
447	os
469	urllib
534	json
650	pygame
702	BeautifulSoup

- avendo salvato nei passi precedenti un file in formato **CSV**
- il cui nome è *province.csv*
- lo apriamo con un **reader** offerto dal modulo **csv** (Python standard library)
- scorriamo i record:
- - escludendo la testata
- - stampando solo le province aventi **sigla** nella lista **marche**
- - adottiamo l'operatore di formattazione `%`
- - per incolonnare i risultati emessi

In [6]:
```
from csv import reader
infile = open("province.csv")
csvReader = reader(infile)
marche = ['PU', 'AN', 'MC', 'AP', 'FM']
formato = "%-26s %11d"
header = True
for riga in csvReader :
    if header :
        header = False
        continue
    if riga[0] in marche :
        print(formato % (riga[1], int(riga[2])) )
infile.close()
```

```
Ancona                          474124
Pesaro e Urbino                 360711
Macerata                        318921
Ascoli Piceno                   209450
Fermo                           174849
```

Novità Python in formattazione stringhe

- la versione **3.6** di Python ha introdotto le cosiddette **f-string**
- le ragioni sono descrtte nel PEP-498
- **PEP** sta per *Python Enhancement Proposals*
- il *PEP-498* ha introdotto la cosiddetta **Literal String Interpolation**
- sono ormai il metodo prederito per formattare il testo
- l'esempio precedente viene così riscritto:

In [7]:
```
from csv import reader
infile = open("province.csv")
csvReader = reader(infile)
marche = ['PU', 'AN', 'MC', 'AP', 'FM']
header = True
for riga in csvReader :
    if header :
        header = False
        continue
    if riga[0] in marche :
        provincia = riga[1]
        abitanti = int(riga[2])
        formato = f"{provincia:26.26s} {abitanti:11d}"
        print(formato)
infile.close()
```

```
Ancona                           474124
Pesaro e Urbino                  360711
Macerata                         318921
Ascoli Piceno                    209450
Fermo                            174849
```

- sono offerte molte nuove possibilità
- - come allineamento rispetto allo spazio disponibile:
- `---` `:<` (*sinistra*), `:^` (*centro*), `:>` (*destra*)
- - valore corrispondente **Unicode**: `:c`

In [8]:
```
testo = "caso di studio"
print(f"INIZIO>|{testo:<26.26s}|<FINE")
print(f"INIZIO>|{testo:^26.26s}|<FINE")
print(f"INIZIO>|{testo:>26.26s}|<FINE")
```

```
INIZIO>|caso di studio            |<FINE
INIZIO>|      caso di studio      |<FINE
INIZIO>|            caso di studio|<FINE
```

In [9]:
```
for seme in range(9824, 9832) :
    print(f"{seme:c}")
```

♠
♡
♢
♣
♤
♥
♦
♧

NumPy

- dopo **Pandas** facciamo alcuni accenni al pacchetto **NumPy**
- è dedicato al supporto per strutture dati multidimensionali
- utili nei calcoli statistici e nella analisi dei dati

- La proprità commutativa dell'AND e dell'OR è analoga a quella dell'algebra tradizionale
- la dimostrazione è immediata dall'analisi delle rispettive tabelle della verità:
- - scambiando l'ordine di A e B il risultato non cambia

In [10]:
```
import numpy as np
andMatrix = np.array([[False and False, False and True] ,
                      [True and False,  True and True ]])

print(andMatrix)
```

```
[[False False]
 [False  True]]
```

In [11]:
```
andMatrix.transpose()
```

Out[11]:
```
array([[False, False],
       [False,  True]])
```

In [12]:
```
if (andMatrix == andMatrix.transpose()).all() :
    print("La AND è commutativa")
else :
    print("La AND non è commutativa")
```

```
La AND è commutativa
```

In [13]:
```
orMatrix = np.array([[False or False, False or True] ,
                     [True or False,  True or True ]])

print(orMatrix)
```

```
[[False  True]
 [ True  True]]
```

In [14]:
```
if (orMatrix == orMatrix.transpose()).all() :
    print("La OR è commutativa")
```

```
else :
    print("La OR non è commutativa")
```

```
La OR è commutativa
```

In [15]:
```
xorMatrix = np.array([[False ^ False, False ^ True] ,
                      [True ^ False,  True ^ True ]])
print(xorMatrix)
if (xorMatrix == xorMatrix.transpose()).all() :
    print("La XOR è commutativa")
else :
    print("La XOR non è commutativa")
```

```
[[False  True]
 [ True False]]
La XOR è commutativa
```

In [16]:
```
riga = np.array([list(range(1, 11))])
colonna = riga.transpose()
matrice = riga * colonna
print(matrice)
```

```
[[  1   2   3   4   5   6   7   8   9  10]
 [  2   4   6   8  10  12  14  16  18  20]
 [  3   6   9  12  15  18  21  24  27  30]
 [  4   8  12  16  20  24  28  32  36  40]
 [  5  10  15  20  25  30  35  40  45  50]
 [  6  12  18  24  30  36  42  48  54  60]
 [  7  14  21  28  35  42  49  56  63  70]
 [  8  16  24  32  40  48  56  64  72  80]
 [  9  18  27  36  45  54  63  72  81  90]
 [ 10  20  30  40  50  60  70  80  90 100]]
```

In [17]:
```
riga = np.array([list(range(1, 13))])
colonna = riga.transpose()
matrice = riga * colonna
print(matrice)
```

```
[[  1   2   3   4   5   6   7   8   9  10  11  12]
 [  2   4   6   8  10  12  14  16  18  20  22  24]
 [  3   6   9  12  15  18  21  24  27  30  33  36]
 [  4   8  12  16  20  24  28  32  36  40  44  48]
 [  5  10  15  20  25  30  35  40  45  50  55  60]
 [  6  12  18  24  30  36  42  48  54  60  66  72]
 [  7  14  21  28  35  42  49  56  63  70  77  84]
 [  8  16  24  32  40  48  56  64  72  80  88  96]
 [  9  18  27  36  45  54  63  72  81  90  99 108]
 [ 10  20  30  40  50  60  70  80  90 100 110 120]
 [ 11  22  33  44  55  66  77  88  99 110 121 132]
 [ 12  24  36  48  60  72  84  96 108 120 132 144]]
```

In [18]:
```
matrice[5, 5]
```

Out[18]: 36

In [19]:
```
matrice * 2
```

Out[19]:
```
array([[  2,   4,   6,   8,  10,  12,  14,  16,  18,  20,  22,  24],
       [  4,   8,  12,  16,  20,  24,  28,  32,  36,  40,  44,  48],
       [  6,  12,  18,  24,  30,  36,  42,  48,  54,  60,  66,  72],
       [  8,  16,  24,  32,  40,  48,  56,  64,  72,  80,  88,  96],
       [ 10,  20,  30,  40,  50,  60,  70,  80,  90, 100, 110, 120],
       [ 12,  24,  36,  48,  60,  72,  84,  96, 108, 120, 132, 144],
       [ 14,  28,  42,  56,  70,  84,  98, 112, 126, 140, 154, 168],
       [ 16,  32,  48,  64,  80,  96, 112, 128, 144, 160, 176, 192],
       [ 18,  36,  54,  72,  90, 108, 126, 144, 162, 180, 198, 216],
       [ 20,  40,  60,  80, 100, 120, 140, 160, 180, 200, 220, 240],
       [ 22,  44,  66,  88, 110, 132, 154, 176, 198, 220, 242, 264],
       [ 24,  48,  72,  96, 120, 144, 168, 192, 216, 240, 264, 288]])
```

In [20]:
```
matrice.max()
```

Out[20]:
```
144
```

In [21]:
```
matrice.sum()
```

Out[21]:
```
6084
```

In [22]:
```
riga = np.array([[False, True]])
print(riga)
```

```
[[False  True]]
```

In [23]:
```
riga.shape
```

Out[23]:
```
(1, 2)
```

In [24]:
```
colonna = riga.transpose()
print(colonna)
```

```
[[False]
 [ True]]
```

In [25]:
```
colonna.shape
```

Out[25]:
```
(2, 1)
```

In [26]:
```
matrice = riga & colonna
print(matrice)
```

```
[[False False]
 [False  True]]
```

In [27]:
```
matrice.shape
```

Out[27]:
```
(2, 2)
```

In [28]:
```
matrice = riga | colonna
print(matrice)
```

```
[[False  True]
 [ True  True]]
```

In [29]:
```
matrice = riga ^ colonna
print(matrice)
```

```
[[False  True]
 [ True False]]
```

SPIGOLATURE

In [30]:
```
from csv import reader
marche = ['PU', 'AN', 'MC', 'AP', 'FM']
with open("province.csv", mode="r") as fh :
    csvReader = reader(fh)
    header = True
    for riga in csvReader :
        if header :
            header = False
            continue
        if riga[0] in marche :
            provincia = riga[1]
            abitanti = int(riga[2])
            formato = f"{provincia:26.26s} {abitanti:11d}"
            print(formato)
```

```
Ancona                          474124
Pesaro e Urbino                 360711
Macerata                        318921
Ascoli Piceno                   209450
Fermo                           174849
```

In [31]:
```
marche = ['PU', 'AN', 'MC', 'AP', 'FM']
listaNuova = [x for x in marche]
print(listaNuova)
```

```
['PU', 'AN', 'MC', 'AP', 'FM']
```

In [32]:
```
marche = ['PU', 'AN', 'MC', 'AP', 'FM']
listaNuova = [x * 2 for x in marche]
print(listaNuova)
```

```
['PUPU', 'ANAN', 'MCMC', 'APAP', 'FMFM']
```

In [33]:
```
interi = list(range(1, 11))
listaNuova = [x * 2 for x in interi]
print(listaNuova)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

In [34]:
```
interi = list(range(1, 11))
listaNuova = [x ** 2 for x in interi]
print(listaNuova)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

---

eliminare duplicati in una lista

In [35]:
```
listaVoti = [ 22, 29, 30, 25, 30, 22, 23 ]
listaVoti = list(set(listaVoti))
print(listaVoti)
```

```
[22, 23, 25, 29, 30]
```

---

indici assegnati da utente

In [36]:
```
import pandas as pd
df = pd.DataFrame(columns=("11/2","13/2","18/2","20/2","25/2","26/2","6/3","
                  index=("Amicucci", "Baffoni", "Bozzi", "Bramucci"),
                  dtype=bool)
df.at["Bramucci", "18/2"] = False
df.at["Bramucci", "20/2"] = False
df.at["Bozzi",    "26/2"] = False
df.at["Bramucci", "26/2"] = False
df
```

Out[36]:

	11/2	13/2	18/2	20/2	25/2	26/2	6/3	11/3	13/3	20/3	25/3	27/3
Amicucci	True	True	True	True	True	True	True	True	True	True	True	True
Baffoni	True	True	True	True	True	True	True	True	True	True	True	True
Bozzi	True	True	True	True	True	False	True	True	True	True	True	True
Bramucci	True	True	False	False	True	False	True	True	True	True	True	True

reverse.py

Sostituire ciascuna riga di un file di testo con la sua inversa

ciao come stai! !iats emoc oaic

- python reverse0.py < input.txt

reverse.py

1. chiedo a video il nome di un file
2. lo apro in modalità testo (no binary)

3. attenzione, potrebbe non esistere
4. leggo una riga alla volta ed emetto su standard output la riga al contrario

In [37]:
```
riga = "riga appena letta da standard input"
a = list(riga)
a.reverse()
print("".join(a))
```

```
tupni dradnats ad attel aneppa agir
```

Identificare la directory corrente

In [41]:
```
from os import getcwd  # Get Current Working Directory
print(getcwd())
```

```
/home/AndreaRibuoli/PythonProgram/ProgrammazioneAdOggettiInPythonELibrerieEs
terne
```

In [42]:
```
from os import getcwd, chdir
from os.path import exists
original = getcwd()
chdir("..")
fileName="province.csv"
if exists(fileName) :
    print(f"il file '{fileName}' esiste")
else :
    print(f"il file '{fileName}' non esiste\n(CWD è '{getcwd()}')")
chdir(original)
```

```
il file 'province.csv' non esiste
(CWD è '/home/AndreaRibuoli/PythonProgram')
```

In [43]:
```
from os.path import exists
fileName="province.csv"
if exists(fileName) :
    print(f"il file '{fileName}' esiste")
else :
    print(f"il file '{fileName}' non esiste")
```

```
il file 'province.csv' esiste
```

---

no_url.py

- Scrivete un programma che,
- dato un indirizzo URL di una pagina web,
- indichi i nomi e le destinazioni dei collegamenti ipertestuali **non** funzionanti.

In [ ]:
```
urllib.request.urlopen()
```

integrale.py

- immaginate gli assi cartesiani **x** (*ascisse*) e **y** (*ordinate*)
- suddividete il segmento che va da 0 a 1 sull'asse delle ascisse in 100 parti uguali
- considerate i 100 rettangoli che hanno per base i 100 segmenti e
- per altezza il valore `sqrt(1 - x ** 2)`
- dove x è la posizione iniziale di ciascuna micro-base lungo l'asse *x*
- sommate le aree dei 100 rettangoli così determinati
- dividete il valore determinato per quello di **pi greco**

- generalizzate il programma in modo da rendere parametrico
- il numero di parti in cui dividere il segmento 0-1
- quanto vale il rapporto con pi greco impostando 1000 parti?
- e 10000?
- Sapete darvi una spiegazione?