
SPS

Raccolta esami passati

David Leoni

11 ago 2021

Copyright © 2021 by David Leoni.

SPS is available under the Creative Commons Attribution 4.0 International License, granting you the right to copy, redistribute, modify, and sell it, so long as you attribute the original to David Leoni and identify any changes that you have made. Full terms of the license are available at:

<http://creativecommons.org/licenses/by/4.0/>

The complete book can be found online for free at:

<https://sps.davidleoni.it>

Indice

About	1
Preface	1
Materiale	1
Quando	1
Esami passati	1
Appelli	73
Istruzioni per esame	74
Ricevimento	74
1 Overview	77

About

Preface

Relatore : David Leoni david.leoni@unitn.it

Docente proponente: Ivano Bison

Edizione: 2021 Marzo/Aprile/Maggio/Giugno

Sede: Dipartimento di Sociologia e Ricerca Sociale, Università di Trento Via Giuseppe Verdi 26, Trento

Parte A: [Locandina](#)¹ | [VEDERE MOODLE](#)² (richiede account unitn)

Parte B: [Locandina](#)³ | [VEDERE MOODLE](#)⁴ (richiede account unitn)

Materiale

Il materiale presente in questa pagina è stato tutto trasferito sul sito softpython.it⁵

Eventuali link a materiale non mio li trovate nella pagina [Riferimenti su SoftPython](#)⁶

Quando

Tipicamente i seminari si svolgono in primavera, possibilmente in presenza oppure online in caso di pandemie.

Chi non potesse partecipare ai seminari, potrebbe essere interessato a iscriversi al modulo d'informatica della [summer school in data science](#)⁷ (contatto: supportostudentipovo@unitn.it)

Esami passati

.1 Esame Ven 16, Apr 2021 A

Seminari Python @Sociologia, Università di Trento

¹ <https://www.sociologia.unitn.it/alfresco/download/workspace/SpacesStore/6c44e1d2-b612-44ba-8866-bdd9d910f840/Locandina%20BISON%20Python%20A%202021.pdf>

² <https://didatticaonline.unitn.it/dol/course/view.php?id=31189>

³ <https://www.sociologia.unitn.it/alfresco/download/workspace/SpacesStore/f320e38b-350c-4da4-a028-04cb24748c6b/Locandina%20BISON%20Python%20B%202021.pdf>

⁴ <https://didatticaonline.unitn.it/dol/course/view.php?id=31865>

⁵ <https://softpython.it>

⁶ <https://it.softpython.org/references.html>

⁷ <http://datascience.unitn.it/presentation/>

.2 Scarica esercizi e soluzioni

.3 Esercizio - prendilelettere

⊗⊗ Data una frase che contiene **esattamente** 3 parole e ha **sempre** come parola centrale un numero n , scrivi del codice che STAMPA i primi n caratteri della terza parola

Esempio - data:

```
frase = "Prendi 4 lettere"
```

il tuo codice deve stampare:

```
lett
```

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">
```

```
[2]:
frase = "Prendi 4 lettere"      # lett
#frase = "Prendere 5 caratteri" # carat
#frase = "Take 10 characters"   # characters

# scrivi qui
parole = frase.split()
n = int(parole[1])
print(parole[2][:n])

lett
```

```
</div>
```

```
[2]:
frase = "Prendi 4 lettere"      # lett
#frase = "Prendere 5 caratteri" # carat
#frase = "Take 10 characters"   # characters

# scrivi qui
```

.4 Esercizio - brico

⊗⊗ Un magazzino per appassionati del fai da te dispone di un catalogo che associa tipologie di oggetti agli scaffali dove posizionarli. Ogni giorno, una lista di arrivi viene popolata con le tipologie di oggetti arrivati. Tali tipologie vanno collocate nel magazzino, un dizionario che associa ad ogni scaffale la tipologia di oggetto prescritta dal catalogo. Scrivi del codice che data la lista di arrivi e il catalogo, popola il dizionario magazzino.

Esempio - dati:

```
arrivi = ['sedie', 'lampade', 'cavi']

catalogo = {'stufe' : 'A',
            'sedie' : 'B',
            'caraffe' : 'D',
```

(continues on next page)

(continua dalla pagina precedente)

```

        'lampade' : 'C',
        'cavi' : 'F',
        'giardinaggio' : 'E'}

magazzino = {}

```

dopo il tuo codice deve risultare:

```

>>> magazzino
{'B': 'sedie', 'C': 'lampade', 'F': 'cavi'}

```

Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">

```

[3]: arrivi = ['sedie', 'lampade', 'cavi'] # magazzino diventa: {'B': 'sedie', 'C':
↳ 'lampade', 'F': 'cavi'}
#arrivi = ['caraffe', 'giardinaggio'] # magazzino diventa: {'D': 'caraffe', 'E':
↳ 'giardinaggio'}
#arrivi = ['stufe'] # magazzino diventa: {'A': 'stufe'}

catalogo = {'stufe' : 'A',
            'sedie' : 'B',
            'caraffe' : 'D',
            'lampade' : 'C',
            'cavi' : 'F',
            'giardinaggio' : 'E'}

# scrivi qui

magazzino = {}

for consegna in arrivi:
    magazzino[ catalogo[consegna] ] = consegna

magazzino

```

```

[3]: {'B': 'sedie', 'C': 'lampade', 'F': 'cavi'}

```

</div>

```

[3]: arrivi = ['sedie', 'lampade', 'cavi'] # magazzino diventa: {'B': 'sedie', 'C':
↳ 'lampade', 'F': 'cavi'}
#arrivi = ['caraffe', 'giardinaggio'] # magazzino diventa: {'D': 'caraffe', 'E':
↳ 'giardinaggio'}
#arrivi = ['stufe'] # magazzino diventa: {'A': 'stufe'}

catalogo = {'stufe' : 'A',
            'sedie' : 'B',
            'caraffe' : 'D',
            'lampade' : 'C',
            'cavi' : 'F',
            'giardinaggio' : 'E'}

# scrivi qui

```

(continues on next page)

.5 Esercizio - La parola più lunga

⊗⊗ Scrivi del codice che data una frase, stampa la **lunghezza** della parola più lunga.

- **NOTA:** vogliamo solo sapere la lunghezza della parola più lunga, non la parola stessa !

Esempio - data:

```
frase = "La strada si inerpica lungo il ciglio della montagna"
```

il tuo codice dovrà stampare

```
8
```

che è la lunghezza delle parole più lunghe che sono a parimerito inerpica e montagna

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code" style="display:none">
```

```
[4]: frase = "La strada si inerpica lungo il ciglio della montagna" # 8
#frase = "Il temibile pirata Le Chuck dominava spietatamente i mari del Sud" # 13
#frase = "Praticamente ovvio" # 12

# scrivi qui

max([len(parola) for parola in frase.split()])
```

```
[4]: 8
```

```
</div>
```

```
[4]: frase = "La strada si inerpica lungo il ciglio della montagna" # 8
#frase = "Il temibile pirata Le Chuck dominava spietatamente i mari del Sud" # 13
#frase = "Praticamente ovvio" # 12

# scrivi qui
```

.6 Esercizio - Scalinate

⊗⊗⊗ Data una lista di lunghezza dispari riempita di zeri eccetto il numero in mezzo, scrivi del codice che MODIFICA la lista per scrivere numeri che decrescano mano a mano che ci si allontana dal centro.

- la lunghezza della lista è sempre dispari
- assumi che la lista sarà sempre di lunghezza sufficiente per arrivare ad avere zero in ciascun bordo
- una lista di dimensione 1 conterrà solo uno zero

Esempio 1 - data:

```
lista = [0, 0, 0, 0, 4, 0, 0, 0, 0]
```

dopo il tuo codice, deve risultare:

```
>>> lista
[0, 1, 2, 3, 4, 3, 2, 1, 0]
```

Esempio 2 - data:

```
lista = [0, 0, 0, 3, 0, 0, 0]
```

dopo il tuo codice, deve risultare:

```
>>> lista
[0, 1, 2, 3, 2, 1, 0]
```

`Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">`

```
[5]: lista = [0, 0, 0, 0, 4, 0, 0, 0, 0] # -> [0, 1, 2, 3, 4, 3, 2, 1, 0]
#lista = [0, 0, 0, 3, 0, 0, 0] # -> [0, 1, 2, 3, 2, 1, 0]
#lista = [0, 0, 2, 0, 0] # -> [0, 1, 2, 1, 0]
#lista = [0] # -> [0]

# scrivi qui

m = len(lista) // 2

for i in range(m):
    lista[m+i] = m - i

for i in range(m):
    lista[i] = i
lista
```

```
[5]: [0, 1, 2, 3, 4, 3, 2, 1, 0]
```

`</div>`

```
[5]: lista = [0, 0, 0, 0, 4, 0, 0, 0, 0] # -> [0, 1, 2, 3, 4, 3, 2, 1, 0]
#lista = [0, 0, 0, 3, 0, 0, 0] # -> [0, 1, 2, 3, 2, 1, 0]
#lista = [0, 0, 2, 0, 0] # -> [0, 1, 2, 1, 0]
#lista = [0] # -> [0]

# scrivi qui
```

.7 Esercizio - Prima di seconda

⊗⊗⊗ Data una stringa e due caratteri car1 e car2, scrivi del codice che STAMPA True se tutte le occorrenze di car1 in stringa sono **sempre** seguite da car2.

Esempio - data:

```
stringa,car1,car2 = "accatastare la posta nella stiva", 's','t'
```

stampa True perchè tutte le occorrenze di s sono seguite da t

```
stringa,car1,car2 = "dadaista entusiasta", 's','t'
```

stampa False, perchè viene ritrovata la sequenza si dove s non è seguita da t

- **USA** un while, cerca di farlo efficiente terminandolo appena puoi
- **NON** usare **break**

Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">

```
[6]:
stringa,car1,car2 = "accatastare la posta nella stiva", 's','t' # True
#stringa,car1,car2 = "dadaista entusiasta", 's','t' # False
#stringa,car1,car2 = "barbabietole", 't','o' # True
#stringa,car1,car2 = "barbabietole", 'b','a' # False
#stringa,car1,car2 = "a", 'a','b' # False
#stringa,car1,car2 = "ab", 'a','b' # True
#stringa,car1,car2 = "aa", 'a','b' # False

# scrivi qui
i = 0

res = True

if len(stringa) == 1:
    res = False

while i + 1 < len(stringa) and res:
    if stringa[i] == car1 and stringa[i+1] != car2:
        res = False
    i += 1

res
```

[6]: True

</div>

```
[6]:
stringa,car1,car2 = "accatastare la posta nella stiva", 's','t' # True
#stringa,car1,car2 = "dadaista entusiasta", 's','t' # False
#stringa,car1,car2 = "barbabietole", 't','o' # True
#stringa,car1,car2 = "barbabietole", 'b','a' # False
#stringa,car1,car2 = "a", 'a','b' # False
#stringa,car1,car2 = "ab", 'a','b' # True
```

(continues on next page)

(continua dalla pagina precedente)

```
#stringa, car1, car2 = "aa", 'a', 'b' # False

# scrivi qui
```

[]:

.8 Esame Lun 31, Mag 2021 - A e B

Seminari Python @Sociologia, Università di Trento

.9 Scarica esercizi e soluzioni

.10 Modulo A

.11 A1 babbà

⊗⊗ Scrivi del codice che data una lettera `cerca` da trovare e una frase, produce una lista con tutte le parole contenenti quella lettera

- **USA** una list comprehension

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">
```

[1]:

```
cerca = 'à' # ['città', 'babbà']
#cerca = 'è' # ['è', 'bignè', 'caffè']

frase = "Questa città è piena di babbà , bignè e caffè"

# scrivi qui

[parola for parola in frase.split() if cerca in parola]
```

[1]: ['città', 'babbà']

</div>

[1]:

```
cerca = 'à' # ['città', 'babbà']
#cerca = 'è' # ['è', 'bignè', 'caffè']

frase = "Questa città è piena di babbà , bignè e caffè"

# scrivi qui
```

.12 A2 selnum

⊗⊗ Date una lista `la` di numeri e una `lb` di booleani, scrivi del codice che MODIFICA la lista `lc` mettendoci dentro solo i numeri di `la` per cui c'è un `True` alla corrispondente posizione di `lb`

- assumi che entrambe le liste abbiano esattamente le stesse dimensioni

Esempio - dati:

```
[2]: la = [9, 7, 6, 8, 7]
      lb = [True, False, True, True, False]
```

Dopo il tuo codice deve risultare

```
>>> print(lc)
[9, 6, 8]
```

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">
```

```
[3]: la, lb = [9,7,6,8,7], [True, False, True, True, False] # [9, 6, 8]
      #la, lb = [3,5,2,3,4,2,4], [True, True, False, True, False, True, False] # [3,5,3,2]
      lc = []

      # scrivi qui

      for i in range(len(la)):
          if lb[i]:
              lc.append(la[i])

      lc
```

```
[3]: [9, 6, 8]
```

```
</div>
```

```
[3]: la, lb = [9,7,6,8,7], [True, False, True, True, False] # [9, 6, 8]
      #la, lb = [3,5,2,3,4,2,4], [True, True, False, True, False, True, False] # [3,5,3,2]
      lc = []

      # scrivi qui
```

.13 A3 rospo

⊗⊗ Dato una stringa `parola` e una stringa `ripetizioni` contenente solo cifre, metti nella variabile `risultato` una stringa contenente tutte le lettere di `parola` ripetute per il numero di volte indicato alla posizione corrispondente in `ripetizioni`

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">
```

```
[4]: parola, ripetizioni = "rospo", "14323"      # 'roooosssppooo'
      #parola, ripetizioni = "artificio", "144232312" # 'arrrrttttiiffffiicccioo'

      # scrivi qui
      res = []

      for i in range(len(parola)):
          res.append(parola[i]*int(ripetizioni[i]))

      risultato = "".join(res)
      print(risultato)

      roooosssppooo
```

</div>

```
[4]: parola, ripetizioni = "rospo", "14323"      # 'roooosssppooo'
      #parola, ripetizioni = "artificio", "144232312" # 'arrrrttttiiffffiicccioo'

      # scrivi qui
```

.14 A4 miniera

⊗⊗ Dato un dizionario `miniera` che associa chiavi a numeri, MODIFICA il dizionario estratto associando le stesse chiavi di `miniera` a liste con le chiavi ripetute il numero di volte indicato.

Esempio - dato

```
miniera = {'ottone': 5,
           'rame'  : 8,
           'ferro' : 1}
estratto = {}
```

dopo il tuo codice deve risultare

```
>>> print(estratto)
{'ottone': ['ottone', 'ottone', 'ottone', 'ottone', 'ottone'],
 'rame'  : ['rame', 'rame', 'rame', 'rame', 'rame', 'rame', 'rame', 'rame'],
 'ferro' : ['ferro']}
```

Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">

```
[5]: miniera = {'ottone' : 5,
               'rame'  : 8,
               'ferro' : 1}

      estratto = {}

      # scrivi qui
```

(continues on next page)

(continua dalla pagina precedente)

```
for chiave in miniera:
    estratto[chiave] = [chiave] * miniera[chiave]
```

```
estratto
```

```
[5]: {'ottone': ['ottone', 'ottone', 'ottone', 'ottone', 'ottone'],
      'rame': ['rame', 'rame', 'rame', 'rame', 'rame', 'rame', 'rame', 'rame'],
      'ferro': ['ferro']}
```

```
</div>
```

```
[5]: miniera = {'ottone' : 5,
               'rame' : 8,
               'ferro' : 1}

estratto = {}

# scrivi qui
```

.15 Modulo B

.16 B1 Strutture sanitarie

⊗⊗ Scrivere una funzione che apre il dataset **SANSTRU001.CSV con pandas** (encoding UTF-8) e prende in input un codice comune e una stringa di testo, e RITORNA un dataframe con selezionate solo le righe aventi quel codice comune e che contengono la stringa nella colonna ASSISTENZA. Il dataset ritornato deve avere solo le colonne STRUTTURA, ASSISTENZA, COD_COMUNE, COMUNE. La funzione STAMPA anche il numero di righe trovate.

Fonte dati: dati.trentino.it⁸

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">
```

```
[6]: import pandas as pd      # importiamo pandas e per comodità lo rinominiamo in 'pd'
import numpy as np          # importiamo numpy e per comodità lo rinominiamo in 'np'

def strutsan(cod_comune, assistenza):

    print('***** SOLUZIONE')
    df = pd.read_csv('SANSTRUT001.csv', encoding='UTF-8')
    res = df[(df['COD_COMUNE'] == cod_comune) & df['ASSISTENZA'].str.
    ↪contains(assistenza)]

    print("Trovate", res.shape[0], "strutture")
    return res[ ['STRUTTURA', 'ASSISTENZA', 'COD_COMUNE', 'COMUNE'] ]
```

```
</div>
```

⁸ <https://dati.trentino.it/dataset/strutture-sanitarie-dell-azienda-sanitaria-e-convenzionate>

```
[6]: import pandas as pd      # importiamo pandas e per comodità lo rinominiamo in 'pd'
import numpy as np          # importiamo numpy e per comodità lo rinominiamo in 'np'
```

```
def strutsan(cod_comune, assistenza):
    raise Exception('TODO IMPLEMENT ME !')
```

```
[7]: strutsan(22050, '') # nessun filtro assistenza
```

```
***** SOLUZIONE
Trovate 6 strutture
```

```
[7]:
```

	STRUTTURA \	ASSISTENZA	COD_COMUNE	COMUNE
0	PRESIDIO OSPEDALIERO DI CAVALESE			
1	PRESIDIO OSPEDALIERO DI CAVALESE			
2	PRESIDIO OSPEDALIERO DI CAVALESE			
3	CENTRO SALUTE MENTALE CAVALESE			
4	CENTRO DIALISI CAVALESE			
5	CONSULTORIO CAVALESE			
0	ATTIVITA' CLINICA	22050	CAVALESE	
1	DIAGNOSTICA STRUMENTALE E PER IMMAGINI	22050	CAVALESE	
2	ATTIVITA' DI LABORATORIO	22050	CAVALESE	
3	ASSISTENZA PSICHIATRICA	22050	CAVALESE	
4	ATTIVITA' CLINICA	22050	CAVALESE	
5	ATTIVITA' DI CONSULTORIO MATERNO-INFANTILE	22050	CAVALESE	

```
[8]: strutsan(22205, 'CLINICA')
```

```
***** SOLUZIONE
Trovate 16 strutture
```

```
[8]:
```

	STRUTTURA	ASSISTENZA \
59	PRESIDIO OSPEDALIERO S.CHIARA	ATTIVITA' CLINICA
62	CENTRO DIALISI TRENTO	ATTIVITA' CLINICA
63	POLIAMBULATORI S.CHIARA	ATTIVITA' CLINICA
64	PRESIDIO OSPEDALIERO VILLA IGEA	ATTIVITA' CLINICA
73	OSPEDALE CLASSIFICATO S.CAMILLO	ATTIVITA' CLINICA
84	NEUROPSICHIATRIA INFANTILE - UONPI 1	ATTIVITA' CLINICA
87	CASA DI CURA VILLA BIANCA SPA	ATTIVITA' CLINICA
90	CENTRO SERVIZI SANITARI	ATTIVITA' CLINICA
93	PSICOLOGIA CLINICA	ATTIVITA' CLINICA
122	ASSOCIAZIONE TRENTINA SCLEROSI MULTIPLA, ONLUS	ATTIVITA' CLINICA
123	ANFFAS TRENTO ONLUS	ATTIVITA' CLINICA
124	COOPERATIVA SOCIALE IRIFOR DEL TRENTO ONLUS	ATTIVITA' CLINICA
126	AGSAT ASSOCIAZIONE GENITORI SOGGETTI AUTISTICI...	ATTIVITA' CLINICA
127	AZIENDA PUBBLICA SERVIZI ALLA PERSONA - RSA PO...	ATTIVITA' CLINICA
130	CST TRENTO	ATTIVITA' CLINICA
133	A.P.S.P. 'BEATO DE TSCHIDERER' - AMB. LOGO-AUD...	ATTIVITA' CLINICA
	COD_COMUNE	COMUNE
59	22205	TRENTO
62	22205	TRENTO
63	22205	TRENTO
64	22205	TRENTO
73	22205	TRENTO
84	22205	TRENTO
87	22205	TRENTO

(continues on next page)

(continua dalla pagina precedente)

```

90      22205  TRENTO
93      22205  TRENTO
122     22205  TRENTO
123     22205  TRENTO
124     22205  TRENTO
126     22205  TRENTO
127     22205  TRENTO
130     22205  TRENTO
133     22205  TRENTO

```

```
[9]: strutsan(22205, 'LABORATORIO')
```

```

***** SOLUZIONE
Trovate 5 strutture

```

```

[9]:
          STRUTTURA          ASSISTENZA  COD_COMUNE  \
61  PRESIDIO OSPEDALIERO S.CHIARA  ATTIVITA` DI LABORATORIO    22205
85          LABORATORI ADIGE SRL  ATTIVITA` DI LABORATORIO    22205
86          LABORATORIO DRUSO SRL  ATTIVITA` DI LABORATORIO    22205
89  CASA DI CURA VILLA BIANCA SPA  ATTIVITA` DI LABORATORIO    22205
92          CENTRO SERVIZI SANITARI  ATTIVITA` DI LABORATORIO    22205

          COMUNE
61  TRENTO
85  TRENTO
86  TRENTO
89  TRENTO
92  TRENTO

```

.17 B2 Strutture Comune di Trento

⊗⊗ Scrivere una funzione `selcir` che apre il dataset [2019-02-17-strutture-comune-di-trento.csv](https://dati.trentino.it/dataset/2019-02-17-strutture-comune-di-trento.csv) con un reader csv⁹ (encoding utf-8) e data una lista `filtro` di parole, seleziona solo le righe che contengono alla colonna `Circoscrizione` **almeno una** delle parole indicate, STAMPA quanti risultati sono stati trovati e RITORNA una NUOVA lista di liste riportante le colonne `Nome` e `Circoscrizione` (senza header)

- il filtro dovrebbe funzionare anche se nel testo ci sono parole con capitalizzazione diversa
- **ATTENZIONE 1:** usare punto e virgola ; come delimiter nel csv reader
- **ATTENZIONE 2:** se più parole del filtro vengono rilevate in una riga, dovresti includere la riga nell'output solo una volta!

Fonte dati: dati.trentino.it¹⁰

```

<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">

```

```

[10]: import csv

def selcir(filtro):

```

(continues on next page)

⁹ <https://it.softpython.org/formats/formats2-csv-sol.html>

¹⁰ <https://dati.trentino.it/dataset/strutture-del-comune-di-trento>

(continua dalla pagina precedente)

```

with open('2019-02-17-strutture-comune-di-trento.csv', encoding='utf-8', newline=
↪') as f:

    lettore = csv.reader(f, delimiter=';')
    next(lettore)

    ret = []
    for riga in lettore:
        #print(riga)
        nome = riga[3]
        circoscrizione = riga[16]

        tieni = False
        for el in filtro:
            if el.lower() in circoscrizione.lower():
                tieni = True
        if tieni:
            ret.append([nome, circoscrizione])

    print("Trovati", len(ret), "risultati")
    return ret

```

```
selcir(['argentario', 'gardolo'])
```

Trovati 16 risultati

```

[10]: [['Argentario', 'Circoscrizione n. 06 - Argentario'],
       ['Martignano', 'Circoscrizione n. 06 - Argentario'],
       ['Montevaccino', 'Circoscrizione n. 06 - Argentario'],
       ['Gardolo', 'Circoscrizione n. 01 - Gardolo'],
       ['Roncafort (via Caproni)', 'Circoscrizione n. 01 - Gardolo'],
       ['Il Piccolo Girasole - Marnighe', 'Circoscrizione n. 06 - Argentario'],
       ['Gardolo', 'Circoscrizione n. 01 - Gardolo'],
       ['Aquilone - Gardolo', 'Circoscrizione n. 01 - Gardolo'],
       ['Roncafort (via Caneppele)', 'Circoscrizione n. 01 - Gardolo'],
       ['Arcobaleno - Martignano', 'Circoscrizione n. 06 - Argentario'],
       ['Margit Levinson - Roncafort', 'Circoscrizione n. 01 - Gardolo'],
       ['Biancaneve - Gardolo', 'Circoscrizione n. 01 - Gardolo'],
       ['Girasole - Melta', 'Circoscrizione n. 01 - Gardolo'],
       ['Gardolo - Meano', 'Circoscrizione n. 01 - Gardolo'],
       ['Argentario', 'Circoscrizione n. 06 - Argentario'],
       ['Gardolo', 'Circoscrizione n. 01 - Gardolo']]

```

</div>

```

[10]: import csv

def selcir(filtro):
    raise Exception('TODO IMPLEMENT ME !')

```

```
selcir(['argentario', 'gardolo'])
```

Trovati 16 risultati

```
[10]: [['Argentario', 'Circoscrizione n. 06 - Argentario'],
      ['Martignano', 'Circoscrizione n. 06 - Argentario'],
      ['Montevaccino', 'Circoscrizione n. 06 - Argentario'],
      ['Gardolo', 'Circoscrizione n. 01 - Gardolo'],
      ['Roncafort (via Caproni)', 'Circoscrizione n. 01 - Gardolo'],
      ['Il Piccolo Girasole - Marnighe', 'Circoscrizione n. 06 - Argentario'],
      ['Gardolo', 'Circoscrizione n. 01 - Gardolo'],
      ['Aquilone - Gardolo', 'Circoscrizione n. 01 - Gardolo'],
      ['Roncafort (via Caneppele)', 'Circoscrizione n. 01 - Gardolo'],
      ['Arcobaleno - Martignano', 'Circoscrizione n. 06 - Argentario'],
      ['Margit Levinson - Roncafort', 'Circoscrizione n. 01 - Gardolo'],
      ['Biancaneve - Gardolo', 'Circoscrizione n. 01 - Gardolo'],
      ['Girasole - Melta', 'Circoscrizione n. 01 - Gardolo'],
      ['Gardolo - Meano', 'Circoscrizione n. 01 - Gardolo'],
      ['Argentario', 'Circoscrizione n. 06 - Argentario'],
      ['Gardolo', 'Circoscrizione n. 01 - Gardolo']]
```

```
[11]: selcir(['argentario', 'Gardolo', 'RAVINA'])
```

Trovati 22 risultati

```
[11]: [['Argentario', 'Circoscrizione n. 06 - Argentario'],
      ['Martignano', 'Circoscrizione n. 06 - Argentario'],
      ['Montevaccino', 'Circoscrizione n. 06 - Argentario'],
      ['Gardolo', 'Circoscrizione n. 01 - Gardolo'],
      ['Ravina', 'Circoscrizione n. 05 - Ravina - Romagnano'],
      ['Romagnano', 'Circoscrizione n. 05 - Ravina - Romagnano'],
      ['Roncafort (via Caproni)', 'Circoscrizione n. 01 - Gardolo'],
      ['Il Piccolo Girasole - Marnighe', 'Circoscrizione n. 06 - Argentario'],
      ['Gardolo', 'Circoscrizione n. 01 - Gardolo'],
      ['Aquilone - Gardolo', 'Circoscrizione n. 01 - Gardolo'],
      ['Ravina', 'Circoscrizione n. 05 - Ravina - Romagnano'],
      ['Roncafort (via Caneppele)', 'Circoscrizione n. 01 - Gardolo'],
      ['Arcobaleno - Martignano', 'Circoscrizione n. 06 - Argentario'],
      ['Girotondo - Ravina', 'Circoscrizione n. 05 - Ravina - Romagnano'],
      ['Margit Levinson - Roncafort', 'Circoscrizione n. 01 - Gardolo'],
      ['Biancaneve - Gardolo', 'Circoscrizione n. 01 - Gardolo'],
      ['Gli gnomi del bosco - Romagnano',
       'Circoscrizione n. 05 - Ravina - Romagnano'],
      ['Girasole - Melta', 'Circoscrizione n. 01 - Gardolo'],
      ['Gardolo - Meano', 'Circoscrizione n. 01 - Gardolo'],
      ['Argentario', 'Circoscrizione n. 06 - Argentario'],
      ['Ravina Romagnano', 'Circoscrizione n. 05 - Ravina - Romagnano'],
      ['Gardolo', 'Circoscrizione n. 01 - Gardolo']]
```

.18 B3 gradini

⊗⊗⊗ Data una matrice quadrata numpy `mat` di dimensione `n`, RITORNA un NUOVO array numpy contenente i valori recuperati dalla matrice nell'ordine seguente:

```
1,2,*,*,*
*,3,4,*,*
*,*,5,6,*
*,*,*,7,8
*,*,*,*,9
```

- se la matrice non è quadrata, lancia `ValueError`

- **NON** usare liste python!
- **SUGGERIMENTO:** quanti elementi deve avere l'array da ritornare?

Esempio:

```
>>> gradini(np.array([ [6,3,5,2,5],
                        [3,4,2,3,4],
                        [6,5,4,5,1],
                        [4,3,2,3,9],
                        [2,5,1,6,7] ] ))
array([6., 3., 4., 2., 4., 5., 3., 9., 7.] )
```

Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">

```
[12]: import numpy as np
```

```
def gradini(mat):

    #SOLUZIONE 'BASIC'
    n,m = mat.shape
    if n != m:
        raise ValueError("Richiesta una n x n, trovata invece una %s x %s" % (n,m))

    res = np.zeros(n + n - 1)

    for i in range(n):
        res[2*i] = mat[i,i]

    for i in range(n-1):
        res[2*i+1] = mat[i,i+1]

    return res

m1 = np.array([ [7] ])
assert np.allclose(gradini(m1), np.array([7]))

m2 = np.array([ [6,8],
                 [9,3] ])
assert np.allclose(gradini(m2), np.array([6,8,3]))

m3 = np.array([ [6,3,5,2,5],
                 [3,4,2,3,4],
                 [6,5,4,5,1],
                 [4,3,2,3,9],
                 [2,5,1,6,7] ])

assert np.allclose(gradini(m3), np.array([6,3,4,2,4,5,3,9,7]))
```

</div>

```
[12]: import numpy as np
```

```
def gradini(mat):
```

(continues on next page)

(continua dalla pagina precedente)

```

    raise Exception('TODO IMPLEMENT ME !')

m1 = np.array([ [7] ])
assert np.allclose(gradini(m1), np.array([7]))

m2 = np.array([ [6,8],
                [9,3] ])
assert np.allclose(gradini(m2), np.array([6,8,3]))

m3 = np.array([ [6,3,5,2,5],
                [3,4,2,3,4],
                [6,5,4,5,1],
                [4,3,2,3,9],
                [2,5,1,6,7] ])

assert np.allclose(gradini(m3), np.array([6,3,4,2,4,5,3,9,7]))

```

Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">

[13]: #SOLUZIONE 'PRO'

```

import numpy as np
def gradini_pro(mat):

    n,m = mat.shape
    if n != m:
        raise ValueError("Richiesta una n x n, trovata invece una %s x %s" % (n,m))
    a = np.diag(mat)
    b = np.diag(mat, 1)
    ret = np.zeros((1, a.shape[0] + b.shape[0]))
    ret[:, ::2] = a
    ret[:, 1::2] = b
    return ret

m1 = np.array([ [7] ])
assert np.allclose(gradini_pro(m1), np.array([7]))

m2 = np.array([ [6,8],
                [9,3] ])
assert np.allclose(gradini_pro(m2), np.array([6,8,3]))

m3 = np.array([ [6,3,5,2,5],
                [3,4,2,3,4],
                [6,5,4,5,1],
                [4,3,2,3,9],
                [2,5,1,6,7] ])

assert np.allclose(gradini_pro(m3), np.array([6,3,4,2,4,5,3,9,7]))

```

</div>

[13]: #SOLUZIONE 'PRO'

.19 B4 muro

⊗⊗⊗ Dato una lista `ripe` di ripetizioni e una matrice `n x m` `mat` come lista di liste, RITORNA una matrice **completamente NUOVA** prendendo le righe di `mat` e replicandole il numero di volte indicato nelle corrispondenti celle di `ripe`

- **NON** devono risultare puntatori dalla matrice nuova a quella vecchia!

Esempio:

```
>>> muro([3,4,1,2], [['i','a','a'],
                      ['q','r','f'],
                      ['y','e','v'],
                      ['e','g','h']])

[['i', 'a', 'a'],
 ['i', 'a', 'a'],
 ['i', 'a', 'a'],
 ['q', 'r', 'f'],
 ['q', 'r', 'f'],
 ['q', 'r', 'f'],
 ['q', 'r', 'f'],
 ['y', 'e', 'v'],
 ['e', 'g', 'h'],
 ['e', 'g', 'h']]
```

Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">

```
[14]: def muro(ripe, mat):

    res = []

    i = 0
    for i in range(len(mat)):
        riga = mat[i]
        n = ripe[i]
        for i in range(n):
            res.append(riga[:])
    return res

m1 = [['a']]
assert muro([2], m1) == [['a'],
                          ['a']]

m2 = [['a','b','c','d'],
      ['e','q','v','r']]
r2 = muro([3,2], m2)
assert r2 == [['a','b','c','d'],
              ['a','b','c','d'],
              ['a','b','c','d'],
              ['e','q','v','r'],
              ['e','q','v','r']]

r2[0][0] = 'z'
assert m2 == [['a','b','c','d'], # vogliamo una NUOVA matrice
              ['e','q','v','r']]
```

(continues on next page)

(continua dalla pagina precedente)

```

m3 = [['i', 'a', 'a'],
      ['q', 'r', 'f'],
      ['y', 'e', 'v'],
      ['e', 'g', 'h']]
r3 = muro([3,4,1,2], m3)
assert r3 == [['i', 'a', 'a'],
              ['i', 'a', 'a'],
              ['i', 'a', 'a'],
              ['q', 'r', 'f'],
              ['q', 'r', 'f'],
              ['q', 'r', 'f'],
              ['q', 'r', 'f'],
              ['y', 'e', 'v'],
              ['e', 'g', 'h'],
              ['e', 'g', 'h']]

```

</div>

```

[14]: def muro(ripe, mat):
        raise Exception('TODO IMPLEMENT ME !')

m1 = [['a']]
assert muro([2], m1) == [['a'],
                        ['a']]

m2 = [['a', 'b', 'c', 'd'],
      ['e', 'q', 'v', 'r']]
r2 = muro([3,2], m2)
assert r2 == [['a', 'b', 'c', 'd'],
              ['a', 'b', 'c', 'd'],
              ['a', 'b', 'c', 'd'],
              ['e', 'q', 'v', 'r'],
              ['e', 'q', 'v', 'r']]

r2[0][0] = 'z'
assert m2 == [['a', 'b', 'c', 'd'], # vogliamo una NUOVA matrice
              ['e', 'q', 'v', 'r']]

m3 = [['i', 'a', 'a'],
      ['q', 'r', 'f'],
      ['y', 'e', 'v'],
      ['e', 'g', 'h']]
r3 = muro([3,4,1,2], m3)
assert r3 == [['i', 'a', 'a'],
              ['i', 'a', 'a'],
              ['i', 'a', 'a'],
              ['q', 'r', 'f'],
              ['q', 'r', 'f'],
              ['q', 'r', 'f'],
              ['q', 'r', 'f'],
              ['y', 'e', 'v'],
              ['e', 'g', 'h'],
              ['e', 'g', 'h']]

```

[]:

.20 Esame Lun 14, Giu 2021 A

Seminari Python - Triennale Sociologia @Università di Trento

.21 Scarica esercizi e soluzioni

.22 A1 La gara

⊗ Una lista di `partecipanti` ha vinto un concorso a premi, e ora si vuole mostrare su un cartellone la loro posizione. Scrivi del codice che MODIFICA la lista scrivendo il numero del partecipante a fianco del nome.

Esempio - data:

```
partecipanti = ['Marta', 'Peppo', 'Elisa', 'Gioele', 'Rosa']
```

dopo il tuo codice deve risultare:

```
>>> partecipanti
['Marta-1', 'Peppo-2', 'Elisa-3', 'Gioele-4', 'Rosa-5']
```

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">
```

```
[2]: partecipanti = ['Marta', 'Peppo', 'Elisa', 'Gioele', 'Rosa']
```

```
# scrivi qui
```

```
for i in range(len(partecipanti)):
    partecipanti[i] = partecipanti[i] + '-' + str(i+1)
```

```
partecipanti
```

```
[2]: ['Marta-1', 'Peppo-2', 'Elisa-3', 'Gioele-4', 'Rosa-5']
```

```
</div>
```

```
[2]: partecipanti = ['Marta', 'Peppo', 'Elisa', 'Gioele', 'Rosa']
```

```
# scrivi qui
```

```
[2]: ['Marta-1', 'Peppo-2', 'Elisa-3', 'Gioele-4', 'Rosa-5']
```

.23 A2 ramarro

Scrivi del codice che dato un insieme cerca di caratteri da cercare, per ciascuno conta quanti ce ne sono nella stringa testo e mette il numero nel dizionario conteggi

Esempio - dati:

```
[3]: cerca = {'i', 't', 'r'}
testo = "Il ramarro orientale è un sauro della famiglia dei Lacertidi, di colore_
↪verde brillante"
conteggi = {}
```

dopo il tuo codice, deve risultare:

```
>>> conteggi
{'r': 9, 'i': 8, 't': 3}
```

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">
```

```
[4]: #jupman-ignore-output
cerca = {'i','t','r'}
testo = "Il ramarro orientale è un sauro della famiglia dei Lacertidi, di colore_
↪verde brillante"
conteggi = {}

# scrivi qui

# soluzione 1, più efficiente
for lettera in testo:
    if lettera in cerca:
        if lettera in conteggi:
            conteggi[lettera] += 1
        else:
            conteggi[lettera] = 1

print(conteggi)

# soluzione 2, meno efficiente (scansioniamo testo n volte con count)
for lettera in cerca:
    conteggi[lettera] = testo.count(lettera)

print(conteggi)

{'r': 9, 'i': 8, 't': 3}
{'r': 9, 'i': 8, 't': 3}
```

</div>

```
[4]: #jupman-ignore-output
cerca = {'i','t','r'}
testo = "Il ramarro orientale è un sauro della famiglia dei Lacertidi, di colore_
↪verde brillante"
conteggi = {}

# scrivi qui

{'r': 9, 'i': 8, 't': 3}
{'r': 9, 'i': 8, 't': 3}
```


.24 A3 hangar

Scrivi del codice che data una stringa `corsa` con un certo numero di trattini all'inizio, STAMPA la parola che segue i trattini.

Esempio - data:

```
corsa = '-----hangar'
```

il tuo codice deve stampare:

```
hangar
```

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">
```

```
[5]: corsa = '-----hangar' # hangar
#corsa = '---bimotore' # bimotore
#corsa = '----747--' # 747--
#corsa = 'aliante' # aliante
#corsa = '-----' # non stampa niente

# scrivi qui

# soluzione 1

print(corsa.lstrip('-'))

# soluzione 2, più algoritmica
i = 0
while i < len(corsa) and corsa[i] == '-':
    i += 1
print(corsa[i:])

hangar
hangar
```

```
</div>
```

```
[5]: corsa = '-----hangar' # hangar
#corsa = '---bimotore' # bimotore
#corsa = '----747--' # 747--
#corsa = 'aliante' # aliante
#corsa = '-----' # non stampa niente

# scrivi qui
```

.25 A4 deserto

Scrivi del codice che data una stringa `viaggio`, produce una lista con tutte le parole che *precedono* le virgole.

Esempio - dato:

```
[6]: viaggio = "Attraversarono deserti, guadaronο fiumi, si inerpicarono sui monti, e_
      ↳ infine arrivarono al Tempio"
```

il tuo codice deve produrre

```
['deserti', 'fiumi', 'monti']
```

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">
```

```
[7]: viaggio = "Attraversarono deserti, guadaronο fiumi, si inerpicarono sui monti, e_
      ↳ infine arrivarono al Tempio"
# ['deserti', 'fiumi', 'monti']
#viaggio = "Camminarono con fatica tra le strade,i mercati affollati, le viuzze,i_
      ↳ portici, finchè trovarono la cattedrale."
# ['strade', 'affollati', 'viuzze', 'portici']
#viaggio = "Il viaggio terminò."
# []
```

```
# scrivi qui
parole = viaggio.split(',')

[frase.split()[-1] for frase in parole[:-1]]
```

```
[7]: ['deserti', 'fiumi', 'monti']
```

</div>

```
[7]: viaggio = "Attraversarono deserti, guadaronο fiumi, si inerpicarono sui monti, e_
      ↳ infine arrivarono al Tempio"
# ['deserti', 'fiumi', 'monti']
#viaggio = "Camminarono con fatica tra le strade,i mercati affollati, le viuzze,i_
      ↳ portici, finchè trovarono la cattedrale."
# ['strade', 'affollati', 'viuzze', 'portici']
#viaggio = "Il viaggio terminò."
# []
```

```
# scrivi qui
```

```
[7]: ['deserti', 'fiumi', 'monti']
```

```
[ ]:
```

.26 Esame Lun 28, Giu 2021 B

Seminari Python - Triennale Sociologia @Università di Trento

.27 Scarica esercizi e soluzioni

.28 B1 Game of Thrones

Apri con Pandas il file `game-of-thrones.csv` che contiene gli episodi in varie annate.

- usa l'encoding UTF-8

B1.1) Ti viene fornito un dizionario `preferiti` con gli episodi preferiti di un gruppo di persone, che però non si ricordano esattamente i vari titoli che sono quindi spesso incompleti: Seleziona gli episodi preferiti da Paolo e Chiara

- assumi che la capitalizzazione in `preferiti` sia quella corretta
- **NOTA:** il dataset contiene insidiose doppie virgolette " attorno ai titoli, ma se scrivi il codice nel modo giusto questo non dovrebbe essere un problema

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code" style="display:none">
```

[2]:

```
import pandas as pd
import numpy as np    # importiamo numpy e per comodità lo rinominiamo in 'np'

preferiti = {
    "Paolo" : 'Winter Is',
    "Chiara" : 'Wolf and the Lion',
    "Anselmo" : 'Fire and',
    "Letizia" : 'Garden of'
}

# scrivi qui
df = pd.read_csv('game-of-thrones.csv', encoding='UTF-8')

titolidf = df[(df["Title"].str.contains(preferiti['Paolo'])) | (df["Title"].str.
    ↳contains(preferiti['Chiara']))]

titolidf
```

	No. overa	No. in season	Season	Title	Directed by	Written by	Novel(s) adapted	Original air date	viewers(millio	Imdb rating
0	1	1	1	"Winter Is Coming"	Tim Van Patten	David Benioff & D. B. Weiss	A Game of Thrones	17-Apr-11	2.22	9.1
4	5	5	1	"The Wolf and the Lion"	Brian Kirk	David Benioff & D. B. Weiss	A Game of Thrones	15-May-11	2.58	9.1

</div>

[2]:

```
import pandas as pd
import numpy as np    # importiamo numpy e per comodità lo rinominiamo in 'np'

preferiti = {
```

(continues on next page)

(continua dalla pagina precedente)

```
"Paolo" : 'Winter Is',
"Chiara" : 'Wolf and the Lion',
"Anselmo" : 'Fire and',
"Letizia" : 'Garden of'
}
```

```
# scrivi qui
```

No. overall	No. in season	Season	Title	Directed by	Written by	Novel(s) adapted	Original air date	viewers(millions)	imdb rating
0	1	1	"Winter Is Coming"	Tim Van Patten	David Benioff & D. B. Weiss	A Game of Thrones	17-Apr-11	2.22	9.1
4	5	5	"The Wolf and the Lion"	Brian Kirk	David Benioff & D. B. Weiss	A Game of Thrones	15-May-11	2.58	9.1

B1.2) Seleziona tutti gli episodi che sono stati mandati per la prima volta in onda in un certo anno (colonna Original air date)

- **NOTA:** anno ti viene fornito come int

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code" style="display:none">
```

```
[3]: anno = 17
```

```
# scrivi qui
annidf = df[ df['Original air date'].str[-2:] == str(anno) ]
annidf
```

No. overall		No. in season	Season	Title	Directed by	Written by	Novel(s) adapted	Original air date	viewers(millio	imdb rating
61	62	2	7	"Stormborn"	Mark Mylod	Bryan Cogman	Outline from A Dream of Spring and original content	23-Jul-17	9.27	8.9
62	63	3	7	"The Queen's Justice"	Mark Mylod	David Benioff & D. B. Weiss	Outline from A Dream of Spring and original content	30-Jul-17	9.25	9.2
63	64	4	7	"The Spoils of War"	Matt Shakman	David Benioff & D. B. Weiss	Outline from A Dream of Spring and original content	6-Aug-17	10.17	9.8
64	65	5	7	"Eastwatch"	Matt Shakman	Dave Hill	Outline from A Dream of Spring and original content	13-Aug-17	10.72	8.8
65	66	6	7	"Beyond the Wall"	Alan Taylor	David Benioff & D. B. Weiss	Outline from A Dream of Spring and original content	20-Aug-17	10.24	9.0
66	67	7	7	"The Dragon and the Wolf"	Jeremy Podeswa	David Benioff & D. B. Weiss	Outline from A Dream of Spring and original content	27-Aug-17	12.07	9.4

```
</div>
```

```
[3]: anno = 17
```

```
# scrivi qui
```

No. overall	No. in season	Season	Title	Directed by	Written by	Novel(s) adapted	Original air date	viewers(millio	imdb rating	
61	62	2	7	"Stormborn"	Mark Mylod	Bryan Cogman	Outline from A Dream of Spring and original content	23-Jul-17	9.27	8.9
62	63	3	7	"The Queen's Justice"	Mark Mylod	David Benioff & D. B. Weiss	Outline from A Dream of Spring and original content	30-Jul-17	9.25	9.2
63	64	4	7	"The Spoils of War"	Matt Shakman	David Benioff & D. B. Weiss	Outline from A Dream of Spring and original content	6-Aug-17	10.17	9.8
64	65	5	7	"Eastwatch"	Matt Shakman	Dave Hill	Outline from A Dream of Spring and original content	13-Aug-17	10.72	8.8
65	66	6	7	"Beyond the Wall"	Alan Taylor	David Benioff & D. B. Weiss	Outline from A Dream of Spring and original content	20-Aug-17	10.24	9.0
66	67	7	7	"The Dragon and the Wolf"	Jeremy Podeswa	David Benioff & D. B. Weiss	Outline from A Dream of Spring and original content	27-Aug-17	12.07	9.4

.29 B2 Punti di interesse universiadi

Scrivi una funzione che dato il file `punti-interesse.csv` dei punti di interesse di Trento individuati per le Universiadi 2013, RITORNA una lista ordinata e senza duplicati con tutti i nomi che trovi nella colonna CATEGORIA.

Sorgente dati: dati.trentino.it¹¹

- **USA un csv.reader e l'encoding latin-1**
- non includere categorie vuote nel risultato
- alcune categorie sono in realtà più di una divise da trattino, separale in categorie distinte:

Esempi:

- Banca- Bancomat-Cambiovaluta
- Centro commerciale-Grande magazzino

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">
```

```
[4]: import csv

def cercat(file_csv):

    with open(file_csv, encoding='latin-1', newline='') as f:
        lettore = csv.reader(f, delimiter=',')
        next(lettore)
        ret = set()
        for riga in lettore:
            for elem in riga[3].split('-'):
                if elem.strip() != '':
                    ret.add(elem.strip())

    return sorted(ret)

risultato = cercat('punti-interesse.csv')
print(risultato)

atteso = ['Affitta Camere', 'Agriturismo', 'Alimentari', 'Appartamento Vacanze',
          'Autostazione', 'Banca', 'Bancomat', 'Bar', 'Bed & Breakfast', 'Biblioteca',
          'Birreria', 'Bus Navetta', 'Cambiovaluta', 'Camping', 'Centro Wellness',
          'Centro commerciale', 'Corrieri', 'Discoteca', 'Editoria', 'Farmacia',
          ↪ 'Funivia',
          'Gelateria', 'Grande magazzino', 'Hotel', 'Istituzioni', 'Mercatini',
          ↪ 'Mercato',
          'Monumento', 'Museo', 'Noleggio Sci', 'Numeri utili', 'Parcheggio',
          ↪ 'Pasticceria',
          'Piscina', 'Posta', 'Prodotti tipici', 'Pub', 'Residence', 'Rifugio',
          ↪ 'Ristorante',
          'Scuola Sci', 'Sede Trentino Trasporti', 'Snow Park', 'Souvenir', 'Sport',
          ↪ 'Stadio',
          'Stadio del ghiaccio', 'Stazione dei Treni', 'Taxi', 'Teatro', 'Ufficio_
          ↪ informazioni turistiche']
#TEST
```

(continues on next page)

¹¹ <https://dati.trentino.it/dataset/poi-trento>

(continua dalla pagina precedente)

```

print()
for i in range(len(atteso)):
    if risultato[i] != atteso[i]:
        print("ERRORE ALL'ELEMENTO %s:" % i)
        print('  ATTESO:', atteso[i])
        print('  TROVATO:', risultato[i])
        break

['Affitta Camere', 'Agriturismo', 'Alimentari', 'Appartamento Vacanze', 'Autostazione
→', 'Banca', 'Bancomat', 'Bar', 'Bed & Breakfast', 'Biblioteca', 'Birreria', 'Bus
→Navetta', 'Cambiovaluta', 'Camping', 'Centro Wellness', 'Centro commerciale',
→'Corrieri', 'Discoteca', 'Editoria', 'Farmacia', 'Funivia', 'Gelateria', 'Grande
→magazzino', 'Hotel', 'Istituzioni', 'Mercatini', 'Mercato', 'Monumento', 'Museo',
→'Noleggio Sci', 'Numeri utili', 'Parcheggio', 'Pasticceria', 'Piscina', 'Posta',
→'Prodotti tipici', 'Pub', 'Residence', 'Rifugio', 'Ristorante', 'Scuola Sci', 'Sede
→Trentino Trasporti', 'Snow Park', 'Souvenir', 'Sport', 'Stadio', 'Stadio del
→ghiaccio', 'Stazione dei Treni', 'Taxi', 'Teatro', 'Ufficio informazioni turistiche
→']

```

</div>

[4]: import csv

```

def cercat(file_csv):
    raise Exception('TODO IMPLEMENT ME !')

risultato = cercat('punti-interesse.csv')
print(risultato)

atteso = ['Affitta Camere', 'Agriturismo', 'Alimentari', 'Appartamento Vacanze',
          'Autostazione', 'Banca', 'Bancomat', 'Bar', 'Bed & Breakfast', 'Biblioteca',
          'Birreria', 'Bus Navetta', 'Cambiovaluta', 'Camping', 'Centro Wellness',
          'Centro commerciale', 'Corrieri', 'Discoteca', 'Editoria', 'Farmacia',
→'Funivia',
          'Gelateria', 'Grande magazzino', 'Hotel', 'Istituzioni', 'Mercatini',
→'Mercato',
          'Monumento', 'Museo', 'Noleggio Sci', 'Numeri utili', 'Parcheggio',
→'Pasticceria',
          'Piscina', 'Posta', 'Prodotti tipici', 'Pub', 'Residence', 'Rifugio',
→'Ristorante',
          'Scuola Sci', 'Sede Trentino Trasporti', 'Snow Park', 'Souvenir', 'Sport',
→'Stadio',
          'Stadio del ghiaccio', 'Stazione dei Treni', 'Taxi', 'Teatro', 'Ufficio
→informazioni turistiche']
#TEST
print()
for i in range(len(atteso)):
    if risultato[i] != atteso[i]:
        print("ERRORE ALL'ELEMENTO %s:" % i)
        print('  ATTESO:', atteso[i])
        print('  TROVATO:', risultato[i])
        break

['Affitta Camere', 'Agriturismo', 'Alimentari', 'Appartamento Vacanze', 'Autostazione
→', 'Banca', 'Bancomat', 'Bar', 'Bed & Breakfast', 'Biblioteca', 'Birreria', 'Bus
→Navetta', 'Cambiovaluta', 'Camping', 'Centro Wellness', 'Centro commerciale',
→'Corrieri', 'Discoteca', 'Editoria', 'Farmacia', 'Funivia', 'Gelateria', 'Grande
→magazzino', 'Hotel', 'Istituzioni', 'Mercatini', 'Mercato', 'Monumento', 'Museo',
→'Noleggio Sci', 'Numeri utili', 'Parcheggio', 'Pasticceria', 'Piscina', 'Posta',
→'Prodotti tipici', 'Pub', 'Residence', 'Rifugio', 'Ristorante', 'Scuola Sci', 'Sede
→Trentino Trasporti', 'Snow Park', 'Souvenir', 'Sport', 'Stadio', 'Stadio del
→ghiaccio', 'Stazione dei Treni', 'Taxi', 'Teatro', 'Ufficio informazioni turistiche
→']

```

(continua dalla pagina precedente)

.30 B3 gratt

Il profilo di una città può essere rappresentato come una lista 2D dove gli 1 rappresentano gli edifici. Nell'esempio sotto, l'altezza dell'edificio più alto è 4 (la seconda colonna da destra)

```
[0, 0, 0, 0, 0, 0],
[0, 0, 0, 0, 1, 0],
[0, 0, 1, 0, 1, 0],
[0, 1, 1, 1, 1, 0],
[1, 1, 1, 1, 1, 1]]
```

Scrivi una funzione che prende un profilo come lista 2-D di 0 e 1 e RITORNA l'altezza del grattacielo più alto, per altri esempi vedere gli assert.

`Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">`

```
[6]:
def gratt(mat):

    n,m = len(mat), len(mat[0])
    for i in range(n):
        for j in range(m):
            if mat[i][j] == 1:
                return n-i
    return 0

assert gratt([0, 0, 0, 0, 0, 0],
              [0, 0, 0, 0, 1, 0],
              [0, 0, 1, 0, 1, 0],
              [0, 1, 1, 1, 1, 0],
              [1, 1, 1, 1, 1, 1]) == 4

assert gratt([
    [0, 0, 0, 0],
    [0, 1, 0, 0],
    [0, 1, 1, 0],
    [1, 1, 1, 1]
]) == 3

assert gratt([
    [0, 1, 0, 0],
    [0, 1, 0, 0],
    [0, 1, 1, 0],
    [1, 1, 1, 1]
]) == 4

assert gratt([
    [0, 0, 0, 0],
    [0, 0, 0, 0],
    [1, 1, 1, 0],
```

(continues on next page)

(continua dalla pagina precedente)

```
[1, 1, 1, 1]
]) == 2
```

</div>

[6]:

```
def gratt(mat):
    raise Exception('TODO IMPLEMENT ME !')

assert gratt([[0, 0, 0, 0, 0, 0],
               [0, 0, 0, 0, 1, 0],
               [0, 0, 1, 0, 1, 0],
               [0, 1, 1, 1, 1, 0],
               [1, 1, 1, 1, 1, 1]]) == 4

assert gratt([
    [0, 0, 0, 0],
    [0, 1, 0, 0],
    [0, 1, 1, 0],
    [1, 1, 1, 1]
]) == 3

assert gratt([
    [0, 1, 0, 0],
    [0, 1, 0, 0],
    [0, 1, 1, 0],
    [1, 1, 1, 1]
]) == 4

assert gratt([
    [0, 0, 0, 0],
    [0, 0, 0, 0],
    [1, 1, 1, 0],
    [1, 1, 1, 1]
]) == 2
```

.31 B4 scendisali

Scrivi una funzione che date le dimensioni di n righe e m colonne RITORNA una NUOVA matrice numpy $n \times m$ con sequenze che scendono e salgono a righe alterne come negli esempi

- se m è dispari, lancia `ValueError`

```
>>> scendisali(6,10)
array([[0., 0., 0., 0., 0., 4., 3., 2., 1., 0.],
       [0., 1., 2., 3., 4., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 4., 3., 2., 1., 0.],
       [0., 1., 2., 3., 4., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 4., 3., 2., 1., 0.],
       [0., 1., 2., 3., 4., 0., 0., 0., 0., 0.]])
```

Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">


```
[7]: import numpy as np

def scendisali(n,m):

    if m%2 == 1:
        raise ValueError("m deve essere pari, trovato %s" % m)
    mat = np.zeros((n,m))
    for i in range(0,n,2):
        for j in range(m//2):
            mat[i,j+m//2] = m//2 - j - 1
    for i in range(1,n,2):
        for j in range(m//2):
            mat[i,j] = j
    return mat

assert np.allclose(scendisali(1,2), np.array([[0., 0.],
                                              [0., 0.])))
assert type(scendisali(1,2)) == np.ndarray

assert np.allclose(scendisali(2,6), np.array([[0., 0., 0., 2., 1., 0.],
                                              [0., 1., 2., 0., 0., 0.])))

assert np.allclose(scendisali(6,10), np.array([[0., 0., 0., 0., 0., 4., 3., 2., 1., 0.
↵],
                                              [0., 1., 2., 3., 4., 0., 0., 0., 0., 0.
↵],
                                              [0., 0., 0., 0., 0., 4., 3., 2., 1., 0.
↵],
                                              [0., 1., 2., 3., 4., 0., 0., 0., 0., 0.
↵],
                                              [0., 0., 0., 0., 0., 4., 3., 2., 1., 0.
↵],
                                              [0., 1., 2., 3., 4., 0., 0., 0., 0., 0.
↵]]))
try:
    scendisali(2,3)
    raise Exception("Avrei dovuto fallire prima!")
except ValueError:
    pass
```

</div>

```
[7]: import numpy as np

def scendisali(n,m):
    raise Exception('TODO IMPLEMENT ME !')

assert np.allclose(scendisali(1,2), np.array([[0., 0.],
                                              [0., 0.])))
assert type(scendisali(1,2)) == np.ndarray

assert np.allclose(scendisali(2,6), np.array([[0., 0., 0., 2., 1., 0.],
                                              [0., 1., 2., 0., 0., 0.])))
```

(continues on next page)

(continua dalla pagina precedente)

```

assert np.allclose(scendisali(6,10), np.array([[0., 0., 0., 0., 0., 4., 3., 2., 1., 0.
↪],
                                                [0., 1., 2., 3., 4., 0., 0., 0., 0., 0.
↪],
                                                [0., 0., 0., 0., 0., 4., 3., 2., 1., 0.
↪],
                                                [0., 1., 2., 3., 4., 0., 0., 0., 0., 0.
↪],
                                                [0., 0., 0., 0., 0., 4., 3., 2., 1., 0.
↪],
                                                [0., 1., 2., 3., 4., 0., 0., 0., 0., 0.
↪]]))
try:
    scendisali(2,3)
    raise Exception("Avrei dovuto fallire prima!")
except ValueError:
    pass

```

[]:

.32 Esame Mer 11, Aug 2021 - Modulo B

Data Science Summer School @Università di Trento

All'esame sono state consegnate diverse versioni di questo foglio ciascuna con un sottoinsieme di esercizi: uno su liste di liste, uno numpy, uno su reader csv da Alberi monumentali della Campania e due su pandas da Beni culturali Regione Umbria

.33 Scarica esercizi e soluzioni

.34 Liste di liste

matriverba

Scrivi una funzione che data una matrice di caratteri, RITORNA una stringa con le parole estratte dalle colonne, mettendo in maiuscolo il primo carattere di ciascuna parola.

- per il maiuscolo usa `.upper()`

Esempio:

```

m = [['p','c','z','g','b','d'],
      ['o','a','a','i','o','e'],
      ['r','l','n','a','r','n'],
      ['t','m','n','r','s','t'],
      ['o','a','a','a','e','e']]

>>> matriverba(m)
'PortoCalmaZannaGiaraBorseDente'

```

Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">

```
[2]:
def matriverba(mat):

    ret = []

    for j in range(len(mat[0])):
        ret.append(mat[0][j].upper())
        for i in range(1, len(mat)):
            ret.append(mat[i][j])
    return ''.join(ret)

# TEST
m1 = [['a']]
assert matriverba(m1) == 'A'

m2 = [['a', 'b']]
assert matriverba(m2) == 'AB'

m3 = [['c'],
      ['b']]
assert matriverba(m3) == 'Cb'

m4 = [['c', 'e'],
      ['b', 'q']]
assert matriverba(m4) == 'CbEq'

m5 = [['p', 'c', 'z', 'g', 'b', 'd'],
      ['o', 'a', 'a', 'i', 'o', 'e'],
      ['r', 'l', 'n', 'a', 'r', 'n'],
      ['t', 'm', 'n', 'r', 's', 't'],
      ['o', 'a', 'a', 'a', 'e', 'e']]

assert matriverba(m5) == 'PortoCalmaZannaGiaraBorseDente'
```

</div>

```
[2]:
def matriverba(mat):
    raise Exception('TODO IMPLEMENT ME !')

# TEST
m1 = [['a']]
assert matriverba(m1) == 'A'

m2 = [['a', 'b']]
assert matriverba(m2) == 'AB'

m3 = [['c'],
      ['b']]
assert matriverba(m3) == 'Cb'

m4 = [['c', 'e'],
      ['b', 'q']]
```

(continues on next page)

(continua dalla pagina precedente)

```

assert matriverba(m4) == 'CbEq'

m5 = [['p','c','z','g','b','d'],
      ['o','a','a','i','o','e'],
      ['r','l','n','a','r','n'],
      ['t','m','n','r','s','t'],
      ['o','a','a','a','e','e']]

assert matriverba(m5) == 'PortoCalmaZannaGiaraBorseDente'

```

cirpillino

Data una stringa e un intero n, RITORNA una NUOVA matrice come lista di liste contenente tutte le lettere della stringa suddivise in righe da n elementi.

- se la lunghezza stringa non è esattamente divisibile per n, solleva ValueError

```

<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">

```

[3]:

```

def cirpillino(stringa, n):

    if len(stringa) % n != 0:
        raise ValueError('La stringa non è divisibile per %s' % n)
    ret = []

    for i in range(len(stringa) // n):
        ret.append(list(stringa[i*n:(i+1)*n]))
    return ret

# TEST
assert cirpillino('z', 1) == [['z']]

assert cirpillino('abc', 1) == [['a'],
                                ['b'],
                                ['c']]

assert cirpillino('abcdef', 2) == [['a','b'],
                                    ['c','d'],
                                    ['e','f']]

assert cirpillino('abcdef', 3) == [['a','b','c'],
                                    ['d','e','f']]

assert cirpillino('cirpillinozimpirelloulalimpo', 4) == [['c', 'i', 'r', 'p'],
                                                            ['i', 'l', 'l', 'i'],
                                                            ['n', 'o', 'z', 'i'],
                                                            ['m', 'p', 'i', 'r'],
                                                            ['e', 'l', 'l', 'o'],
                                                            ['u', 'l', 'a', 'l'],
                                                            ['i', 'm', 'p', 'o']]

```

(continues on next page)

(continua dalla pagina precedente)

```
try:
    cirpillino('abc', 5)
    raise Exception("Avrei dovuto fallire !")
except ValueError:
    pass
```

</div>

[3]:

```
def cirpillino(stringa, n):
    raise Exception('TODO IMPLEMENT ME !')

# TEST
assert cirpillino('z', 1) == [['z']]

assert cirpillino('abc', 1) == [['a'],
                                ['b'],
                                ['c']]

assert cirpillino('abcdef', 2) == [['a','b'],
                                   ['c','d'],
                                   ['e','f']]

assert cirpillino('abcdef', 3) == [['a','b','c'],
                                   ['d','e','f']]

assert cirpillino('cirpillinozimpirelloulalimpo', 4) == [['c', 'i', 'r', 'p'],
                                                         ['i', 'l', 'l', 'i'],
                                                         ['n', 'o', 'z', 'i'],
                                                         ['m', 'p', 'i', 'r'],
                                                         ['e', 'l', 'l', 'o'],
                                                         ['u', 'l', 'a', 'l'],
                                                         ['i', 'm', 'p', 'o']]

try:
    cirpillino('abc', 5)
    raise Exception("Avrei dovuto fallire !")
except ValueError:
    pass
```

bandiera

Dati due numeri interi n e m , con m multiplo di 3, RITORNA una matrice $n \times m$ come lista di liste avente nelle celle i numeri da 0 a 2 ripartiti in 3 fasce verticali. Per esempi vedere assert.

- se m non è un multiplo di 3, solleva `ValueError`

Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">

[4]: `def bandiera(n,m):`

```
    if m % 3 != 0:
```

(continues on next page)

(continua dalla pagina precedente)

```

    raise ValueError('Il numero di colonne non è un multiplo di 3: %s' % m)

ret = []

for i in range(n):
    riga = []
    for j in range(m):
        num = j // (m // 3)
        riga.append(num)
    ret.append(riga)
return ret

# TEST
assert bandiera(1,3) == [[0, 1, 2]]

assert bandiera(1,6) == [[0,0,1,1, 2,2]]

assert bandiera(4,6) == [[0, 0, 1, 1, 2, 2],
                        [0, 0, 1, 1, 2, 2],
                        [0, 0, 1, 1, 2, 2],
                        [0, 0, 1, 1, 2, 2]]

assert bandiera(2,9) == [[0, 0, 0, 1, 1, 1, 2, 2, 2],
                        [0, 0, 0, 1, 1, 1, 2, 2, 2]]

assert bandiera(5,12) == [[0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2],
                        [0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2],
                        [0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2],
                        [0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2],
                        [0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2]]

try:
    bandiera(3,7)
    raise Exception("Avrei dovuto fallire!")
except ValueError:
    pass

```

</div>

```

[4]: def bandiera(n,m):
    raise Exception('TODO IMPLEMENT ME !')

# TEST
assert bandiera(1,3) == [[0, 1, 2]]

assert bandiera(1,6) == [[0,0,1,1, 2,2]]

assert bandiera(4,6) == [[0, 0, 1, 1, 2, 2],
                        [0, 0, 1, 1, 2, 2],
                        [0, 0, 1, 1, 2, 2],
                        [0, 0, 1, 1, 2, 2]]

assert bandiera(2,9) == [[0, 0, 0, 1, 1, 1, 2, 2, 2],
                        [0, 0, 0, 1, 1, 1, 2, 2, 2]]

```

(continues on next page)

(continua dalla pagina precedente)

```

assert bandiera(5,12) == [[0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2],
                           [0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2],
                           [0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2],
                           [0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2],
                           [0, 0, 0, 0, 1, 1, 1, 1, 2, 2, 2, 2]]

try:
    bandiera(3,7)
    raise Exception("Avrei dovuto fallire!")
except ValueError:
    pass

```

no_diag

Data una matrice $n \times n$ come lista di liste, RITORNA una NUOVA matrice $n \times n-1$ avente le stesse celle dell'originale ECCETTO le celle della diagonale. Per esempi, vedere gli assert.

- se la matrice non è quadrata, lancia ValueError

Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">

```

[5]: def no_diag(mat):

    if len(mat) != len(mat[0]):
        raise ValueError("Matrice non quadrata: %s x %s" % (len(mat), len(mat[0])))

    ret = []
    i = 0
    for riga in mat:
        nuova_riga = riga[0:i] + riga[i+1:]
        ret.append(nuova_riga)
        i += 1

    return ret

# TEST
m1 = [[3,4],
      [8,7]]
assert no_diag(m1) == [[4],
                       [8]]
assert m1 == [[3,4], # verifica che non abbia cambiato l'originale
              [8,7]]

m2 = [[9,4,3],
      [8,5,6],
      [0,2,7]]
assert no_diag(m2) == [[4,3],
                       [8,6],
                       [0,2]]

m3 = [[8,5,3,4],
      [7,2,4,1],
      [9,8,3,5],

```

(continues on next page)

(continua dalla pagina precedente)

```

        [6,0,4,7]]
assert no_diag(m3) == [[5,3,4],
                        [7,4,1],
                        [9,8,5],
                        [6,0,4]]

try:
    no_diag([[2,3,5],
             [1,5,2]])
    raise Exception("Avrei dovuto fallire!")
except ValueError:
    pass

```

</div>

```

[5]: def no_diag(mat):
    raise Exception('TODO IMPLEMENT ME !')

# TEST
m1 = [[3,4],
      [8,7]]
assert no_diag(m1) == [[4],
                       [8]]
assert m1 == [[3,4], # verifica che non abbia cambiato l'originale
              [8,7]]

m2 = [[9,4,3],
      [8,5,6],
      [0,2,7]]
assert no_diag(m2) == [[4,3],
                       [8,6],
                       [0,2]]

m3 = [[8,5,3,4],
      [7,2,4,1],
      [9,8,3,5],
      [6,0,4,7]]
assert no_diag(m3) == [[5,3,4],
                        [7,4,1],
                        [9,8,5],
                        [6,0,4]]

try:
    no_diag([[2,3,5],
             [1,5,2]])
    raise Exception("Avrei dovuto fallire!")
except ValueError:
    pass

```


evita_diag

Data una matrice quadrata $n \times n$ come liste di liste RITORNA una NUOVA lista con la somma di tutti i numeri di ogni riga TRANNE la diagonale.

- se la matrice non è quadrata, lancia `ValueError`

`Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">`

[6]:

```
def evita_diag(mat):

    if len(mat) != len(mat[0]):
        raise ValueError("Matrice non quadrata: %s x %s" % (len(mat), len(mat[0])))
    ret = []
    i = 0
    for riga in mat:
        ret.append(sum(riga) - riga[i])
        i += 1
    return ret

assert evita_diag([[5]]) == [0]

m2 = [[5,7],
      [9,1]]
assert evita_diag(m2) == [7,9]
assert m2 == [[5,7],
              [9,1]]

assert evita_diag([ [5,6,2],
                    [4,7,9],
                    [1,9,8]]) == [8, 13, 10]

try:
    evita_diag([ [2,3,5],
                 [1,5,2]])
    raise Exception("Avrei dovuto fallire!")
except ValueError:
    pass
```

`</div>`

[6]:

```
def evita_diag(mat):
    raise Exception('TODO IMPLEMENT ME !')

assert evita_diag([[5]]) == [0]

m2 = [[5,7],
      [9,1]]
assert evita_diag(m2) == [7,9]
assert m2 == [[5,7],
              [9,1]]

assert evita_diag([ [5,6,2],
```

(continues on next page)

(continua dalla pagina precedente)

```

        [4,7,9],
        [1,9,8]]) == [8, 13, 10]

try:
    evita_diag([[2,3,5],
                [1,5,2]])
    raise Exception("Avrei dovuto fallire!")
except ValueError:
    pass

```

no_anti_diag

Data una matrice quadrata $n \times n$ *mat* come lista di liste, RITORNA una NUOVA matrice $n \times n-1$ avente le stesse celle dell'originale ECCETTO le celle della ANTI diagonale. Per esempi, vedere gli assert.

- se n non è quadrata, lancia `ValueError`

```

<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">

```

```

[7]: def no_anti_diag(mat):

    if len(mat) != len(mat[0]):
        raise ValueError("Matrice non quadrata: %s x %s" % (len(mat), len(mat[0])))
    ret = []
    for i in range(len(mat)):
        k = len(mat) - i - 1
        nuova = mat[i][:k] + mat[i][k+1:]
        ret.append(nuova)
    return ret

m1 = [[3,4],
      [8,7]]
assert no_anti_diag(m1) == [[3],
                             [7]]

assert m1 == [[3,4], # verifica che non abbia cambiato l'originale
              [8,7]]

m2 = [[9,4,3],
      [8,5,6],
      [0,2,7]]
assert no_anti_diag(m2) == [[9,4],
                             [8,6],
                             [2,7]]

m3 = [[8,5,3,4],
      [7,2,4,1],
      [9,8,3,5],
      [6,0,4,7]]
assert no_anti_diag(m3) == [[8,5,3],
                             [7,2,1],
                             [9,3,5],
                             [0,4,7]]

```

(continues on next page)

(continua dalla pagina precedente)

```

try:
    no_anti_diag([[2,3,5],
                  [1,5,2]])
    raise Exception("Avrei dovuto fallire!")
except ValueError:
    pass

```

</div>

```

[7]: def no_anti_diag(mat):
      raise Exception('TODO IMPLEMENT ME !')

m1 = [[3,4],
       [8,7]]
assert no_anti_diag(m1) == [[3],
                             [7]]

assert m1 == [[3,4], # verifica che non abbia cambiato l'originale
              [8,7]]

m2 = [[9,4,3],
       [8,5,6],
       [0,2,7]]
assert no_anti_diag(m2) == [[9,4],
                             [8,6],
                             [2,7]]

m3 = [[8,5,3,4],
       [7,2,4,1],
       [9,8,3,5],
       [6,0,4,7]]
assert no_anti_diag(m3) == [[8,5,3],
                             [7,2,1],
                             [9,3,5],
                             [0,4,7]]

try:
    no_anti_diag([[2,3,5],
                  [1,5,2]])
    raise Exception("Avrei dovuto fallire!")
except ValueError:
    pass

```

matinc

Data una matrice intera RITORNA True se tutte le righe sono strettamente crescenti da sinistra a destra, altrimenti ritorna False. Per esempi vedere i test.

Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">

```

[8]: def matinc(mat):

      for i in range(len(mat)):
          for j in range(1, len(mat[0])):
              if mat[i][j] <= mat[i][j-1]:

```

(continues on next page)

(continua dalla pagina precedente)

```

        return False
    return True

# TEST
m1 = [[5]]
assert matinc(m1) == True

m2 = [[7],
      [4]]
assert matinc(m2) == True

m3 = [[2,3],
      [3,5]]
assert matinc(m3) == True

m4 = [[9,4]]
assert matinc(m4) == False

m5 = [[5,5]]
assert matinc(m5) == False

m6 = [[1,4,6,7,9],
      [0,1,2,4,8],
      [2,6,8,9,10]]
assert matinc(m6) == True

m7 = [[0,1,3,4],
      [4,6,9,10],
      [3,7,7,15]]
assert matinc(m7) == False

m8 = [[1,4,8,7,9],
      [0,1,2,4,8]]
assert matinc(m8) == False

```

</div>

```

[8]: def matinc(mat):
    raise Exception('TODO IMPLEMENT ME !')

# TEST
m1 = [[5]]
assert matinc(m1) == True

m2 = [[7],
      [4]]
assert matinc(m2) == True

m3 = [[2,3],
      [3,5]]
assert matinc(m3) == True

m4 = [[9,4]]
assert matinc(m4) == False

m5 = [[5,5]]

```

(continues on next page)

(continua dalla pagina precedente)

```

assert matinc(m5) == False

m6 = [[1,4,6,7,9],
      [0,1,2,4,8],
      [2,6,8,9,10]]
assert matinc(m6) == True

m7 = [[0,1,3,4],
      [4,6,9,10],
      [3,7,7,15]]
assert matinc(m7) == False

m8 = [[1,4,8,7,9],
      [0,1,2,4,8]]
assert matinc(m8) == False

```

ordinul

Data una matrice come lista di liste di numeri interi, MODIFICA la matrice ordinando SOLO i numeri nell'ultima colonna

- Tutte le altre celle NON devono cambiare

Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">

```

[9]: def ordinul(mat):

    ordinata = sorted([mat[i][-1] for i in range(len(mat))])
    for i in range(len(mat)):
        mat[i][-1] = ordinata[i]

    # TEST
    m1 = [[3]]
    ordinul(m1)
    assert m1 == [[3]]

    m2 = [[9,3,7],
          [8,5,4]]
    ordinul(m2)
    assert m2 == [[9,3,4],
                  [8,5,7]]

    m3 = [[8,5,9],
          [7,2,3],
          [9,8,7]]
    ordinul(m3)
    assert m3 == [[8,5,3],
                  [7,2,7],
                  [9,8,9]]

    m4 = [[8,5,3,2,4],
          [7,2,4,1,1],
          [9,8,3,3,7],

```

(continues on next page)

(continua dalla pagina precedente)

```

        [6,0,4,2,5]]
ordinul(m4)
assert m4 == [[8, 5, 3, 2, 1],
               [7, 2, 4, 1, 4],
               [9, 8, 3, 3, 5],
               [6, 0, 4, 2, 7]]

assert ordinul([[3]]) == None

```

</div>

```

[9]: def ordinul(mat):
        raise Exception('TODO IMPLEMENT ME !')

    # TEST
    m1 = [[3]]
    ordinul(m1)
    assert m1 == [[3]]

    m2 = [[9,3,7],
          [8,5,4]]
    ordinul(m2)
    assert m2 == [[9,3,4],
                  [8,5,7]]

    m3 = [[8,5,9],
          [7,2,3],
          [9,8,7]]
    ordinul(m3)
    assert m3 == [[8,5,3],
                  [7,2,7],
                  [9,8,9]]

    m4 = [[8,5,3,2,4],
          [7,2,4,1,1],
          [9,8,3,3,7],
          [6,0,4,2,5]]
    ordinul(m4)
    assert m4 == [[8, 5, 3, 2, 1],
                  [7, 2, 4, 1, 4],
                  [9, 8, 3, 3, 5],
                  [6, 0, 4, 2, 7]]

    assert ordinul([[3]]) == None

```

.35 Numpy

colgap

Data una matrice numpy di n righe ed m colonne, RITORNA un vettore numpy di m elementi avente la differenza tra i massimi e i minimi di ciascuna colonna.

Esempio:

```
m = np.array([[5,4,2],
              [8,5,1],
              [6,7,9],
              [3,6,4],
              [4,3,7]])
>>> colgap(m)
array([5, 4, 8])
```

perchè

```
5 = 8 - 3
4 = 7 - 3
8 = 9 - 1
```

Mostra soluzione

```
[10]: import numpy as np

def colgap(mat):

    #SOLUZIONE EFFICIENTE
    mx = np.max(mat, axis=0)
    mn = np.min(mat, axis=0)
    return mx - mn

# TEST
m1 = np.array([[6]])
assert np.allclose(colgap(m1), np.array([0]))
ret = colgap(m1)
assert type(ret) == np.ndarray

m2 = np.array([[6,8]])
assert np.allclose(colgap(m2), np.array([0,0]))
m3 = np.array([[2],
               [5]])
assert np.allclose(colgap(m3), np.array([3]))
m4 = np.array([[5,7],
               [2,9]])
assert np.allclose(colgap(m4), np.array([3,2]))
m5 = np.array([[4,7],
               [4,9]])
assert np.allclose(colgap(m5), np.array([0,2]))
m6 = np.array([[5,2],
               [3,7],
               [9,0]])
assert np.allclose(colgap(m6), np.array([6,7]))
m7 = np.array([[5,4,2],
               [8,5,1],
               [6,7,9],
               [3,6,4],
               [4,3,7]])
assert np.allclose(colgap(m7), np.array([5,4,8]))
```

</div>

```
[10]: import numpy as np

def colgap(mat):
    raise Exception('TODO IMPLEMENT ME !')

# TEST
m1 = np.array([[6]])
assert np.allclose(colgap(m1), np.array([0]))
ret = colgap(m1)
assert type(ret) == np.ndarray

m2 = np.array([[6,8]])
assert np.allclose(colgap(m2), np.array([0,0]))
m3 = np.array([[2],
               [5]])
assert np.allclose(colgap(m3), np.array([3]))
m4 = np.array([[5,7],
               [2,9]])
assert np.allclose(colgap(m4), np.array([3,2]))
m5 = np.array([[4,7],
               [4,9]])
assert np.allclose(colgap(m5), np.array([0,2]))
m6 = np.array([[5,2],
               [3,7],
               [9,0]])
assert np.allclose(colgap(m6), np.array([6,7]))
m7 = np.array([[5,4,2],
               [8,5,1],
               [6,7,9],
               [3,6,4],
               [4,3,7]])
assert np.allclose(colgap(m7), np.array([5,4,8]))
```

revtriang

Data una matrice quadrata numpy, RITORNA una NUOVA matrice numpy avente le stesse dimensioni dell'originale e i numeri nelle righe della parte triangolare inferiore (diagonale esclusa) in ordine inverso

- se la matrice non è quadrata, lancia ValueError

Esempio:

```
m = np.array([[5,4,2,6,4],
              [3,5,1,0,6],
              [6,4,9,2,3],
              [5,2,8,6,1],
              [7,9,3,2,2]])

>>> revtriang(m5)
np.array([[5, 4, 2, 6, 4],
          [3, 5, 1, 0, 6],      # 3      -> 3
          [4, 6, 9, 2, 3],      # 6,4     -> 4,6
          [8, 2, 5, 6, 1],      # 5,2,8  -> 8,2,5
          [2, 3, 9, 7, 2]])     # 7,9,3,2 -> 2,3,9,7
```


Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">

```
[11]: import numpy as np

def revtriang(mat):

    n,m = mat.shape
    if n != m:
        raise ValueError("Attesa matrice quadrata, trovato invece n=%s, m=%s" % (n,m))

    ret = mat.copy()

    for i in range(1,n):
        ret[i,:i] = np.flip(mat[i,:i])
    return ret

m1 = np.array([[8]])
assert np.allclose(revtriang(m1), np.array([[8]]))

m3 = np.array([[1,5],
               [9,6]])
assert np.allclose(revtriang(m3), np.array([[1,5],
               [9,6]]))

m4 = np.array([[1,5,8],
               [9,6,2],
               [3,2,5]])
assert np.allclose(revtriang(m4), np.array([[1,5,8],
               [9,6,2],
               [2,3,5]]))

assert np.allclose(m4, np.array([[1,5,8],
               [9,6,2],
               [3,2,5]])) # non cambia l'originale

m5 = np.array([[5,4,2,6,4],
               [3,5,1,0,6],
               [6,4,9,2,3],
               [5,2,8,6,1],
               [7,9,3,2,2]])
assert np.allclose(revtriang(m5), np.array([[5, 4, 2, 6, 4],
               [3, 5, 1, 0, 6],
               [4, 6, 9, 2, 3],
               [8, 2, 5, 6, 1],
               [2, 3, 9, 7, 2]]))

try:
    revtriang(np.array([[7,1,6],
                       [5,2,4]]))
    raise Exception("Avrei dovuto fallire!")
except ValueError:
    pass
```

</div>

```
[11]: import numpy as np

def revtriang(mat):
    raise Exception('TODO IMPLEMENT ME !')

m1 = np.array([[8]])
assert np.allclose(revtriang(m1), np.array([[8]]))

m3 = np.array([[1,5],
               [9,6]])
assert np.allclose(revtriang(m3), np.array([[1,5],
                                             [9,6]]))

m4 = np.array([[1,5,8],
               [9,6,2],
               [3,2,5]])
assert np.allclose(revtriang(m4), np.array([[1,5,8],
                                             [9,6,2],
                                             [2,3,5]]))

assert np.allclose(m4, np.array([[1,5,8],
                                 [9,6,2],
                                 [3,2,5]])) # non cambia l'originale

m5 = np.array([[5,4,2,6,4],
               [3,5,1,0,6],
               [6,4,9,2,3],
               [5,2,8,6,1],
               [7,9,3,2,2]])
assert np.allclose(revtriang(m5), np.array([[5, 4, 2, 6, 4],
                                             [3, 5, 1, 0, 6],
                                             [4, 6, 9, 2, 3],
                                             [8, 2, 5, 6, 1],
                                             [2, 3, 9, 7, 2]]))

try:
    revtriang(np.array([[7,1,6],
                       [5,2,4]]))
    raise Exception("Avrei dovuto fallire!")
except ValueError:
    pass
```

compricol

Data una matrice mat $n \times 2m$ con numero di colonne pari, RITORNA una NUOVA matrice $n \times m$ in cui le colonne sono date dalle somma delle coppie di colonne corrispondenti di mat

- se mat non ha numero di colonne pari, lancia `ValueError`

Esempio:

```
m = np.array([[5,4,2,6,4,2],
              [7,5,1,0,6,1],
              [6,7,9,2,3,7],
              [5,2,4,6,1,3],
              [7,2,3,4,2,5]])
```

(continues on next page)

(continua dalla pagina precedente)

```
>>> compricol(m)
np.array([[ 9,  8,  6],
          [12,  1,  7],
          [13, 11, 10],
          [ 7, 10,  4],
          [ 9,  7,  7]])
```

perchè

```
9 = 5 + 4      8 = 2 + 6      6 = 4 + 2
12 = 7 + 5     1 = 1 + 0      7 = 6 + 1
. . .
```

Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">

```
[12]: import numpy as np

def compricol(mat):

    #SOLUZIONE EFFICIENTE
    if mat.shape[1] % 2 != 0:
        raise ValueError("Attesa matrice con numero di colonne pari, trovate invece:
→ %s" % mat.shape[1])
    n,m = mat.shape[0], mat.shape[1] // 2
    ret = mat[:,::2].copy()
    ret += mat[:,1::2]
    return ret

m1 = [[7,9]]
res = compricol(np.array(m1))
assert type(res) == np.ndarray
assert np.allclose(res, np.array([[16]]))

m2 = np.array([[5,8],
               [7,2]])
assert np.allclose(compricol(m2), np.array([[13],
                                             [9]]))
assert np.allclose(m2, np.array([[5,8],
                                  [7,2]])) # non cambia la matrice originale

m3 = np.array([[5,4,2,6,4,2],
               [7,5,1,0,6,1],
               [6,7,9,2,3,7],
               [5,2,4,6,1,3],
               [7,2,3,4,2,5]])

assert np.allclose(compricol(m3), np.array([[ 9,  8,  6],
                                             [12,  1,  7],
                                             [13, 11, 10],
                                             [ 7, 10,  4],
                                             [ 9,  7,  7]]))
```

(continues on next page)

(continua dalla pagina precedente)

```

try:
    compricol(np.array([[7,1,6],
                        [5,2,4]]))
    raise Exception("Avrei dovuto fallire!")
except ValueError:
    pass

```

</div>

```

[12]: import numpy as np

def compricol(mat):
    raise Exception('TODO IMPLEMENT ME !')

m1 = [[7,9]]
res = compricol(np.array(m1))
assert type(res) == np.ndarray
assert np.allclose(res, np.array([[16]]))

m2 = np.array([[5,8],
               [7,2]])
assert np.allclose(compricol(m2), np.array([[13],
                                             [9]]))
assert np.allclose(m2, np.array([[5,8],
                                  [7,2]])) # non cambia la matrice originale

m3 = np.array([[5,4,2,6,4,2],
               [7,5,1,0,6,1],
               [6,7,9,2,3,7],
               [5,2,4,6,1,3],
               [7,2,3,4,2,5]])

assert np.allclose(compricol(m3), np.array([[ 9, 8, 6],
                                             [12, 1, 7],
                                             [13,11,10],
                                             [ 7,10, 4],
                                             [ 9, 7, 7]]))

try:
    compricol(np.array([[7,1,6],
                        [5,2,4]]))
    raise Exception("Avrei dovuto fallire!")
except ValueError:
    pass

```

sostmax

Data una matrice numpy mat $n \times m$, MODIFICA la matrice sostituendo ogni cella con il valore massimo trovato nella colonna corrispondente.

Esempio:

```

m = np.array([[5,4,2],
              [8,5,1],
              [6,7,9],

```

(continues on next page)

(continua dalla pagina precedente)

```

        [3,6,4],
        [4,3,7]])
>>> sostmax(m)      # non ritorna niente!
>>> m
np.array([[8, 7, 9],
        [8, 7, 9],
        [8, 7, 9],
        [8, 7, 9],
        [8, 7, 9]])

```

Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">

```

[13]: import numpy as np

def sostmax(mat):

    #SOLUZIONE EFFICIENTE
    mat[:,:] = np.max(mat, axis=0)

    # TEST
    m1 = np.array([[6]])
    sostmax(m1)
    assert np.allclose(m1, np.array([6]))
    ret = sostmax(m1)
    assert ret == None # non ritorna nulla!

    m2 = np.array([[6,8]])
    sostmax(m2)
    assert np.allclose(m2, np.array([6,8]))

    m3 = np.array([[2],
                   [5]])
    sostmax(m3)
    assert np.allclose(m3, np.array([[5],
                                      [5]]))

    m4 = np.array([[5,7],
                   [2,9]])
    sostmax(m4)

    assert np.allclose(m4, np.array([[5,9],
                                      [5,9]]))

    m5 = np.array([[4,7],
                   [4,9]])
    sostmax(m5)
    assert np.allclose(m5, np.array([[4,9],
                                      [4,9]]))

    m6 = np.array([[5,2],
                   [3,7],
                   [9,0]])
    sostmax(m6)

```

(continues on next page)

(continua dalla pagina precedente)

```

assert np.allclose(m6, np.array([[9,7],
                                   [9,7],
                                   [9,7]]))

m7 = np.array([[5,4,2],
               [8,5,1],
               [6,7,9],
               [3,6,4],
               [4,3,7]])

sostmax(m7)
assert np.allclose(m7, np.array([[8, 7, 9],
                                   [8, 7, 9],
                                   [8, 7, 9],
                                   [8, 7, 9],
                                   [8, 7, 9]]))

```

</div>

```

[13]: import numpy as np

def sostmax(mat):
    raise Exception('TODO IMPLEMENT ME !')

# TEST
m1 = np.array([[6]])
sostmax(m1)
assert np.allclose(m1, np.array([6]))
ret = sostmax(m1)
assert ret == None # non ritorna nulla!

m2 = np.array([[6,8]])
sostmax(m2)
assert np.allclose(m2, np.array([6,8]))

m3 = np.array([[2],
               [5]])
sostmax(m3)
assert np.allclose(m3, np.array([[5],
                                   [5]]))

m4 = np.array([[5,7],
               [2,9]])
sostmax(m4)

assert np.allclose(m4, np.array([[5,9],
                                   [5,9]]))

m5 = np.array([[4,7],
               [4,9]])
sostmax(m5)
assert np.allclose(m5, np.array([[4,9],
                                   [4,9]]))

m6 = np.array([[5,2],
               [3,7],
               [9,0]])
sostmax(m6)

```

(continues on next page)

(continua dalla pagina precedente)

```

assert np.allclose(m6, np.array([[9,7],
                                   [9,7],
                                   [9,7]]))

m7 = np.array([[5,4,2],
                [8,5,1],
                [6,7,9],
                [3,6,4],
                [4,3,7]])

sostmax(m7)
assert np.allclose(m7, np.array([[8, 7, 9],
                                   [8, 7, 9],
                                   [8, 7, 9],
                                   [8, 7, 9],
                                   [8, 7, 9]]))

```

camminas

Data una matrice numpy $n \times m$ con m dispari, RITORNA un array numpy contenente tutti i numeri trovati lungo il percorso di una S, dal basso verso l'alto.

SUGGERIMENTO: puoi determinare a priori la dimensione dell'array risultante?

Esempio:

```

m = np.array([[5,8,2,4,6,5,7],
               [7,9,5,8,3,2,2],
               [6,1,8,3,6,6,1],
               [1,5,3,7,9,4,7],
               [1,5,3,2,9,5,4],
               [4,3,8,5,6,1,5]])

```

deve percorrere, **dal basso verso l'alto**:

```

m = np.array([[5,8,2,>,>,>,>],
               [7,9,5,^,3,2,2],
               [6,1,8,^,6,6,1],
               [1,5,3,^,9,4,7],
               [1,5,3,^,9,5,4],
               [>,>,>,>,6,1,5]])

```

Per ottenere:

```

>>> camminas(m)
array([4., 3., 8., 5., 2., 7., 3., 8., 4., 6., 5., 7.])

```

Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">

```

[14]: import numpy as np

def camminas(mat):

    #SOLUZIONE EFFICIENTE

```

(continues on next page)

(continua dalla pagina precedente)

```

n,m = mat.shape
ret = np.zeros(n + m-1)
ret[:m//2] = mat[-1,:m//2]
ret[m//2:m//2+n] = mat[::-1,m//2]
ret[-m//2:] = mat[0,m//2:]
return ret

# TEST
m1 = np.array([[7]])
assert np.allclose(camminas(m1), np.array([7]))

m2 = np.array([[7,5,2]])
assert np.allclose(camminas(m2), np.array([7,5,2]))

m3 = np.array([[9,3,5,6,0]])
assert np.allclose(camminas(m3), np.array([9,3,5,6,0]))

m4 = np.array([[7,5,2],
               [9,3,4]])
assert np.allclose(camminas(m4), np.array([9,3,5,2]))

m5 = np.array([[7,4,6],
               [8,2,1],
               [0,5,3]])
assert np.allclose(camminas(m5), np.array([0,5,2,4,6]))

m6 = np.array([[5,8,2,4,6,5,7],
               [7,9,5,8,3,2,2],
               [6,1,8,3,6,6,1],
               [1,5,3,7,9,4,7],
               [1,5,3,2,9,5,4],
               [4,3,8,5,6,1,5]])
assert np.allclose(camminas(m6), np.array([4,3,8,5,2,7,3,8,4,6,5,7]))

```

</div>

```

[14]: import numpy as np

def camminas(mat):
    raise Exception('TODO IMPLEMENT ME !')

# TEST
m1 = np.array([[7]])
assert np.allclose(camminas(m1), np.array([7]))

m2 = np.array([[7,5,2]])
assert np.allclose(camminas(m2), np.array([7,5,2]))

m3 = np.array([[9,3,5,6,0]])
assert np.allclose(camminas(m3), np.array([9,3,5,6,0]))

m4 = np.array([[7,5,2],
               [9,3,4]])
assert np.allclose(camminas(m4), np.array([9,3,5,2]))

m5 = np.array([[7,4,6],

```

(continues on next page)

(continua dalla pagina precedente)

```

        [8,2,1],
        [0,5,3]])
assert np.allclose(camminas(m5), np.array([0,5,2,4,6]))

m6 = np.array([[5,8,2,4,6,5,7],
               [7,9,5,8,3,2,2],
               [6,1,8,3,6,6,1],
               [1,5,3,7,9,4,7],
               [1,5,3,2,9,5,4],
               [4,3,8,5,6,1,5]])
assert np.allclose(camminas(m6), np.array([4,3,8,5,2,7,3,8,4,6,5,7]))

```

camminaz

Data una matrice numpy $n \times m$ con m dispari, RITORNA un array numpy contenente tutti i numeri trovati lungo il percorso di una Z, dal basso verso l'alto.

SUGGERIMENTO: puoi determinare a priori la dimensione dell'array risultante?

Esempio:

```

m = np.array([[5,8,2,4,6,5,7],
               [7,9,5,8,3,2,2],
               [6,1,8,3,6,6,1],
               [1,5,3,7,9,4,7],
               [1,5,3,2,9,5,4],
               [4,3,8,5,6,1,5]])

```

deve percorrere, **dal basso verso l'alto**:

```

m = np.array([[<, <, <, ^, 6, 5, 7],
               [7, 9, 5, ^, 3, 2, 2],
               [6, 1, 8, ^, 6, 6, 1],
               [1, 5, 3, ^, 9, 4, 7],
               [1, 5, 3, ^, 9, 5, 4],
               [4, 3, 8, ^, <, <, <]])

```

Per ottenere:

```

>>> camminaz(m)
array([5., 1., 6., 5., 2., 7., 3., 8., 4., 2., 8., 5.])

```

```

<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">

```

```
[15]: import numpy as np
```

```

def camminaz(mat):

    #SOLUZIONE EFFICIENTE
    n,m = mat.shape
    ret = np.zeros(n + m-1)
    ret[:m//2] = mat[-1,-1:m//2:-1]
    ret[m//2:m//2+n] = mat[:, -1, m//2:]

```

(continues on next page)

(continua dalla pagina precedente)

```

ret[-m//2:] = mat[0,m//2:-1]
return ret

# TEST
m1 = np.array([[7]])
assert np.allclose(camminaz(m1), np.array([7]))

m2 = np.array([[7,5,2]])
assert np.allclose(camminaz(m2), np.array([2,5,7]))

m3 = np.array([[9,3,5,6,0]])
assert np.allclose(camminaz(m3), np.array([0,6,5,3,9]))

m4 = np.array([[7,5,2],
               [9,3,4]])
assert np.allclose(camminaz(m4), np.array([4,3,5,7]))

m5 = np.array([[7,4,6],
               [8,2,1],
               [0,5,3]])
assert np.allclose(camminaz(m5), np.array([3,5,2,4,7]))

m6 = np.array([[5,8,2,4,6,5,7],
               [7,9,5,8,3,2,2],
               [6,1,8,3,6,6,1],
               [1,5,3,7,9,4,7],
               [1,5,3,2,9,5,4],
               [4,3,8,5,6,1,5]])
assert np.allclose(camminaz(m6), np.array([5,1,6,5,2,7,3,8,4,2,8,5]))

```

</div>

```

[15]: import numpy as np

def camminaz(mat):
    raise Exception('TODO IMPLEMENT ME !')

# TEST
m1 = np.array([[7]])
assert np.allclose(camminaz(m1), np.array([7]))

m2 = np.array([[7,5,2]])
assert np.allclose(camminaz(m2), np.array([2,5,7]))

m3 = np.array([[9,3,5,6,0]])
assert np.allclose(camminaz(m3), np.array([0,6,5,3,9]))

m4 = np.array([[7,5,2],
               [9,3,4]])
assert np.allclose(camminaz(m4), np.array([4,3,5,7]))

m5 = np.array([[7,4,6],
               [8,2,1],
               [0,5,3]])
assert np.allclose(camminaz(m5), np.array([3,5,2,4,7]))

```

(continues on next page)

(continua dalla pagina precedente)

```
m6 = np.array([[5,8,2,4,6,5,7],
               [7,9,5,8,3,2,2],
               [6,1,8,3,6,6,1],
               [1,5,3,7,9,4,7],
               [1,5,3,2,9,5,4],
               [4,3,8,5,6,1,5]])
assert np.allclose(camminaz(m6), np.array([5,1,6,5,2,7,3,8,4,2,8,5]))
```

.36 Alberi monumentali della Campania

albernomi

Scrivi una funzione che data una parola di ricerca carica il file `Alberi-Monumentali-Della-Campania.csv` con un csv reader (usa il parametro `delimiter=';'` ed encoding `utf-8`), STAMPA il numero di risultati ottenuti e RITORNA tutti gli alberi aventi quella parola nel nome scientifico oppure nel nome volgare.

- la ricerca deve funzionare indipendentemente dalla capitalizzazione di parola o del dataset

Il formato di output deve essere una lista di dizionari come questa:

```
>>> albernomi('tiglio')
Trovati 12 risultati

[{'nome': 'Tiglio intermedio',
  'nome_scientifico': 'Tilia vulgaris',
  'luogo': 'Collegiata della Santissima Annunziata'},
 {'nome': 'Tiglio intermedio',
  'nome_scientifico': 'Tilia vulgaris',
  'luogo': 'Petraro di Forino'},
 {'nome': 'Tiglio selvatico',
  'nome_scientifico': 'Tilia cordata',
  'luogo': 'San Barbato - Castello'},
 .
 .
 .
]
```

Sorgente dati: dati.gov.it¹²

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">
```

```
[16]: import csv

def albernomi(parola):

    with open('Alberi-Monumentali-Della-Campania.csv', encoding='utf-8', newline='')
    as f:
        lettore = csv.DictReader(f, delimiter=';')
        next(lettore)
        ret = []
        for d in lettore:
```

(continues on next page)

¹² <https://dati.gov.it/view-dataset/dataset?id=9e636fa8-8a8d-43ed-820a-09cd31c9f2b5>

(continua dalla pagina precedente)

```

    if parola.lower() in d['NOME_SCIENTIFICO'].lower() \
    or parola.lower() in d['NOME_VOLGARE'].lower():
        diz = {'nome' : d['NOME_VOLGARE'],
                'nome_scientifico' : d['NOME_SCIENTIFICO'],
                'luogo' : d['LOCALITA']}
        ret.append(diz)

    print('Trovati', len(ret), 'risultati')
    return ret

```

```

albernomi('tiglio')      # 12 risultati
#albernomi('TIGLIO')     # 12 risultati
#albernomi('tilia')      # 12 risultati
#albernomi('Tilia')      # 12 risultati
#albernomi('cordata')    # 8 risultati

```

Trovati 12 risultati

```

[16]: [{'nome': 'Tiglio intermedio',
        'nome_scientifico': 'Tilia vulgaris',
        'luogo': 'Collegiata della Santissima Annunziata'},
       {'nome': 'Tiglio intermedio',
        'nome_scientifico': 'Tilia vulgaris',
        'luogo': 'Petruro di Forino'},
       {'nome': 'Tiglio selvatico',
        'nome_scientifico': 'Tilia cordata',
        'luogo': 'San Barbato - Castello'},
       {'nome': 'Tiglio selvatico',
        'nome_scientifico': 'Tilia cordata',
        'luogo': 'Piazza San Felice'},
       {'nome': 'Tiglio selvatico',
        'nome_scientifico': 'Tilia cordata',
        'luogo': 'Casola'},
       {'nome': 'Tiglio nostrale',
        'nome_scientifico': 'Tilia platyphyllos',
        'luogo': 'Piano di Sorrento'},
       {'nome': 'Tiglio',
        'nome_scientifico': 'Tilia vulgaris',
        'luogo': 'Centro Urbano'},
       {'nome': 'Tiglio',
        'nome_scientifico': 'Tilia cordata',
        'luogo': 'Massaquano'},
       {'nome': 'Tiglio', 'nome_scientifico': 'Tilia cordata', 'luogo': 'Filetta'},
       {'nome': 'Tiglio', 'nome_scientifico': 'Tilia cordata', 'luogo': 'Campora'},
       {'nome': 'Tiglio',
        'nome_scientifico': 'Tilia cordata',
        'luogo': 'Parco Colonia montana'},
       {'nome': 'Tiglio',
        'nome_scientifico': 'Tilia cordata',
        'luogo': 'Largo Sipicciano'}]

```

</div>

```

[16]: import csv

def albernomi(parola):

```

(continues on next page)

(continua dalla pagina precedente)

```
raise Exception('TODO IMPLEMENT ME !')
```

```
albernomi('tiglio')      # 12 risultati
#albernomi('TIGLIO')     # 12 risultati
#albernomi('tilia')      # 12 risultati
#albernomi('Tilia')      # 12 risultati
#albernomi('cordata')    # 8 risultati
```

Trovati 12 risultati

```
[16]: [{'nome': 'Tiglio intermedio',
        'nome_scientifico': 'Tilia vulgaris',
        'luogo': 'Collegiata della Santissima Annunziata'},
       {'nome': 'Tiglio intermedio',
        'nome_scientifico': 'Tilia vulgaris',
        'luogo': 'Petraro di Forino'},
       {'nome': 'Tiglio selvatico',
        'nome_scientifico': 'Tilia cordata',
        'luogo': 'San Barbato - Castello'},
       {'nome': 'Tiglio selvatico',
        'nome_scientifico': 'Tilia cordata',
        'luogo': 'Piazza San Felice'},
       {'nome': 'Tiglio selvatico',
        'nome_scientifico': 'Tilia cordata',
        'luogo': 'Casola'},
       {'nome': 'Tiglio nostrale',
        'nome_scientifico': 'Tilia platyphyllos',
        'luogo': 'Piano di Sorrento'},
       {'nome': 'Tiglio',
        'nome_scientifico': 'Tilia vulgaris',
        'luogo': 'Centro Urbano'},
       {'nome': 'Tiglio',
        'nome_scientifico': 'Tilia cordata',
        'luogo': 'Massaquano'},
       {'nome': 'Tiglio', 'nome_scientifico': 'Tilia cordata', 'luogo': 'Filetta'},
       {'nome': 'Tiglio', 'nome_scientifico': 'Tilia cordata', 'luogo': 'Campora'},
       {'nome': 'Tiglio',
        'nome_scientifico': 'Tilia cordata',
        'luogo': 'Parco Colonia montana'},
       {'nome': 'Tiglio',
        'nome_scientifico': 'Tilia cordata',
        'luogo': 'Largo Sipicciano'}]
```

alberalti

Scrivi una funzione che date una altezza minima e una massima carica il file [Alberi-Monumentali-Della-Campania.csv](#) con un csv reader (usa il parametro `delimiter=';'` ed encoding utf-8), STAMPA il numero di risultati ottenuti e RITORNA tutti gli alberi aventi altezza inclusa nell'intervallo dato **estremi inclusi**.

Il formato di output deve essere una lista di dizionari come questa:

```
>>> alberalti(4,7)
```

Trovati 13 risultati

```
[{'nome': 'Tiglio selvatico',
```

(continues on next page)

(continua dalla pagina precedente)

```

    'altezza': 6,
    'località': 'San Barbato - Castello'},
{'nome': 'Sofora',
 'altezza': 5,
 'località': 'Villa Rende'},
{'nome': 'Olivo',
 'altezza': 6,
 'località': 'Via Carducci - Piazza Sabato'},
.
.
.
]
```

Sorgente dati: dati.gov.it¹³

```

<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">
```

```

[17]: import csv

def alberalti(minh, maxh):

    with open('Alberi-Monumentali-Della-Campania.csv', encoding='utf-8', newline='')
    ↪ as f:
        lettore = csv.DictReader(f, delimiter=';')
        next(lettore)
        ret = []
        for d in lettore:
            h = int(d['ALTEZZA'])
            if minh <= h and h <= maxh:
                diz = { 'nome' : d['NOME_VOLGARE'],
                        'altezza' : h,
                        'località' : d['LOCALITA']}
                ret.append(diz)

        print('Trovati', len(ret), 'risultati')
        return ret
```

```

alberalti(4,7)      # 13 risultati
#alberalti(5,8)    # 15 risultati
```

Trovati 13 risultati

```

[17]: [{ 'nome': 'Tiglio selvatico',
        'altezza': 6,
        'località': 'San Barbato - Castello'},
       { 'nome': 'Sofora', 'altezza': 5, 'località': 'Villa Rende'},
       { 'nome': 'Olivo', 'altezza': 6, 'località': 'Via Carducci - Piazza Sabato'},
       { 'nome': 'Leccio', 'altezza': 7, 'località': 'Viale della Vittoria'},
       { 'nome': 'Platano', 'altezza': 4, 'località': 'Ogliara'},
       { 'nome': 'Tiglio', 'altezza': 6, 'località': 'Centro Urbano'},
       { 'nome': 'Leccio', 'altezza': 6, 'località': 'Piazza F. Napolitano'},
       { 'nome': 'Gelso', 'altezza': 6, 'località': 'Puolo - Villa Angelina'},
       { 'nome': 'Tiglio', 'altezza': 6, 'località': 'Massaquano'},
```

(continues on next page)

¹³ <https://dati.gov.it/view-dataset/dataset?id=9e636fa8-8a8d-43ed-820a-09cd31c9f2b5>

(continua dalla pagina precedente)

```
{'nome': 'Alloro', 'altezza': 6, 'località': 'Pratillo'},
{'nome': 'Gelso', 'altezza': 4, 'località': 'Vieticala'},
{'nome': 'Tiglio', 'altezza': 5, 'località': 'Filetta'},
{'nome': 'Yucca', 'altezza': 6, 'località': "Mostra d'Oltremare"}}
```

</div>

[17]: `import csv`

```
def alberalti(minh, maxh):
    raise Exception('TODO IMPLEMENT ME !')
```

```
alberalti(4,7)    # 13 risultati
#alberalti(5,8)  # 15 risultati
```

Trovati 13 risultati

```
[17]: [{'nome': 'Tiglio selvatico',
        'altezza': 6,
        'località': 'San Barbato - Castello'},
       {'nome': 'Sofora', 'altezza': 5, 'località': 'Villa Rende'},
       {'nome': 'Oliivo', 'altezza': 6, 'località': 'Via Carducci - Piazza Sabato'},
       {'nome': 'Leccio', 'altezza': 7, 'località': 'Viale della Vittoria'},
       {'nome': 'Platano', 'altezza': 4, 'località': 'Ogliara'},
       {'nome': 'Tiglio', 'altezza': 6, 'località': 'Centro Urbano'},
       {'nome': 'Leccio', 'altezza': 6, 'località': 'Piazza F. Napolitano'},
       {'nome': 'Gelso', 'altezza': 6, 'località': 'Puolo - Villa Angelina'},
       {'nome': 'Tiglio', 'altezza': 6, 'località': 'Massaquano'},
       {'nome': 'Alloro', 'altezza': 6, 'località': 'Pratillo'},
       {'nome': 'Gelso', 'altezza': 4, 'località': 'Vieticala'},
       {'nome': 'Tiglio', 'altezza': 5, 'località': 'Filetta'},
       {'nome': 'Yucca', 'altezza': 6, 'località': "Mostra d'Oltremare"}]
```

alberi per provincia

Scrivere del codice che conta per ogni provincia quanti alberi ci sono, e visualizza un grafico a barre verdi

SUGGERIMENTO: Vedere [grafici a barre](#)¹⁴ e [xticks](#)¹⁵

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">
```

```
[18]: %matplotlib inline
import matplotlib.pyplot as plt

# scrivi qui
with open('Alberi-Monumentali-Della-Campania.csv', encoding='utf-8', newline='') as f:
    lettore = csv.DictReader(f, delimiter=',')
    next(lettore)
    ret = []
    province = {}
    for d in lettore:
```

(continues on next page)

¹⁴ <https://it.softpython.org/visualization/visualization1-sol.html#Grafici-a-barre>

¹⁵ <https://it.softpython.org/visualization/visualization1-sol.html#Le-etichette-sugli-assi>

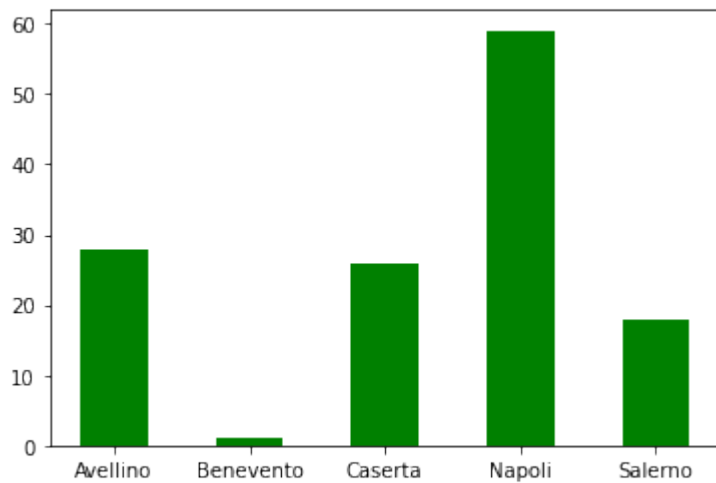
(continua dalla pagina precedente)

```
h = int(d['ALTEZZA'])
p = d['PROVINCIA']
if p not in province:
    province[p] = 1
else:
    province[p] += 1

xs = list(range(len(province)))
nomi_province = province.keys()
ys = [province[x] for x in nomi_province]
plt.xticks(xs, nomi_province)

plt.bar(xs, ys,
        0.5,
        color='green',
        align='center')

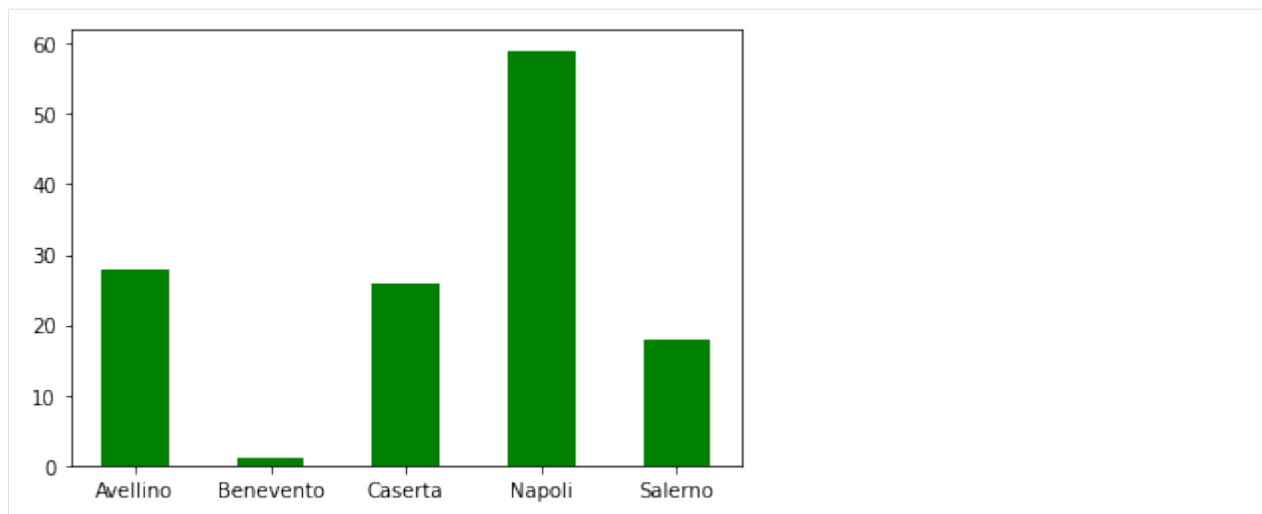
plt.show()
```



</div>

```
[18]: %matplotlib inline
import matplotlib.pyplot as plt

# scrivi qui
```

.37 Beni culturali Regione Umbria

Apri il dataset `beni-culturali-umbria.csv` con pandas (encoding UTF-8) e mostra informazioni sulle colonne

- **ATTENZIONE:** usa l'attributo `delimiter=';'`

Sorgente dati: dati.gov.it¹⁶

`Mostra soluzione<div class="jupman-sol jupman-sol-code" style="display:none">`

```
[19]: import pandas as pd
import numpy as np

# scrivi qui
df = pd.read_csv('beni-culturali-umbria.csv', encoding='UTF-8', delimiter=';')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 105 entries, 0 to 104
Data columns (total 5 columns):
#   Column                Non-Null Count  Dtype
---  -
0   NUMERO                 105 non-null   int64
1   BENEFICIARIO           105 non-null   object
2   PROGETTO               105 non-null   object
3   PROGRAMMA_OPERATIVO    105 non-null   object
4   STATO_ATTUAZIONE       105 non-null   object
dtypes: int64(1), object(4)
memory usage: 4.2+ KB
```

`</div>`

```
[19]: import pandas as pd
import numpy as np
```

(continues on next page)

¹⁶ <https://dati.gov.it/view-dataset/dataset?id=36edd544-412a-4377-b00e-f01782af90cd>

(continua dalla pagina precedente)

scrivi qui

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 105 entries, 0 to 104
Data columns (total 5 columns):
#   Column                Non-Null Count  Dtype
---  -
0   NUMERO                 105 non-null   int64
1   BENEFICIARIO           105 non-null   object
2   PROGETTO               105 non-null   object
3   PROGRAMMA_OPERATIVO    105 non-null   object
4   STATO_ATTUAZIONE       105 non-null   object
dtypes: int64(1), object(4)
memory usage: 4.2+ KB
```

[29]: df

```
[29]:
```

	NUMERO	BENEFICIARIO \
0	1	Comune di Acquasparta
1	2	Comune di Acquasparta
2	3	Comune di Acquasparta
3	4	Comune di Acquasparta
4	5	Comune di Acquasparta
..
100	101	Comune di Umbertide
101	102	Regione Umbria
102	103	Azienda pubblica 'Veralli Cortesi'
103	104	Sodalizio S. Martino Perugia
104	105	Valnestore Sviluppo

	PROGETTO \
0	Progetto per il recupero, il restauro e la tra...
1	Recupero, restauro e trasformazione in centro ...
2	Riqualificazione e valorizzazione Palazzo Cesi
3	Completamento Palazzo Cesi. Riqualificazione d...
4	Completamento delle opere di restauro e valori...
..	...
100	Lavori di completamento del Centro Socio Cultu...
101	Recupero e consolidamento della Cinta muraria ...
102	Valorizzazione arte contemporanea. Lavori di c...
103	FUSEUM Museo Brajo Fuso: completamento
104	Museo Paleontologico L. Boldrini. Completamento

	PROGRAMMA_OPERATIVO	STATO_ATTUAZIONE	sigla
0	POR FESR 2007 -2013 Attività 2.2.2	REALIZZATO	POR
1	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO	PAR
2	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO	PAR
3	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO	PAR
4	POR FESR 2014 - 2020 Azione 5.2.1	IN CORSO DI REALIZZAZIONE	POR
..
100	POR FESR 2007 -2013 Attività 2.2.2	REALIZZATO	POR
101	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO	PAR
102	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO	PAR
103	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO	PAR
104	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO	PAR

(continues on next page)

(continua dalla pagina precedente)

[105 rows x 6 columns]

stato progetti

Data il dizionario query con comune e stato, trova tutti i progetti ce soddisfino **entrambe** le condizioni

- **NON** scrivere Acquasparta o REALIZZATO nel codice !

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">
```

```
[21]: query = {'comune' : 'Acquasparta',
              'stato'  : 'REALIZZATO'
              }
#query = {'comune' : 'Spoleto', 'stato' : 'IN CORSO DI REALIZZAZIONE' }

# scrivi qui
df[(df['STATO_ATTUAZIONE'] == query['stato']) & df['BENEFICIARIO'].str.contains(query[
↪ 'comune'])]
```

```
[21]:
```

	NUMERO	BENEFICIARIO \	PROGETTO \	PROGRAMMA_OPERATIVO	STATO_ATTUAZIONE
0	1	Comune di Acquasparta	Progetto per il recupero, il restauro e la tra...	POR FESR 2007 -2013 Attività 2.2.2	REALIZZATO
1	2	Comune di Acquasparta	Recupero, restauro e trasformazione in centro ...	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO
2	3	Comune di Acquasparta	Riqualificazione e valorizzazione Palazzo Cesi	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO
3	4	Comune di Acquasparta	Completamento Palazzo Cesi. Riqualificazione d...	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO

</div>

```
[21]: query = {'comune' : 'Acquasparta',
              'stato'  : 'REALIZZATO'
              }
#query = {'comune' : 'Spoleto', 'stato' : 'IN CORSO DI REALIZZAZIONE' }

# scrivi qui
```

```
[21]:
```

	NUMERO	BENEFICIARIO \
0	1	Comune di Acquasparta
1	2	Comune di Acquasparta
2	3	Comune di Acquasparta
3	4	Comune di Acquasparta

(continues on next page)

(continua dalla pagina precedente)

```

                                PROGETTO \
0 Progetto per il recupero, il restauro e la tra...
1 Recupero, restauro e trasformazione in centro ...
2 Riqualificazione e valorizzazione Palazzo Cesi
3 Completamento Palazzo Cesi. Riqualificazione d...

PROGRAMMA_OPERATIVO STATO_ATTUAZIONE
0 POR FESR 2007 -2013 Attività 2.2.2 REALIZZATO
1 PAR FSC 2007 - 2013 Azione 3.5.2a REALIZZATO
2 PAR FSC 2007 - 2013 Azione 3.5.2a REALIZZATO
3 PAR FSC 2007 - 2013 Azione 3.5.2a REALIZZATO

```

riqualificazione

Trova tutti i progetti che prevedono riqualificazione

ATTENZIONE alle diverse capitalizzazioni! In tutto dovresti trovare 17 risultati

```

<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">

```

```

[22]: # scrivi qui
df[df['PROGETTO'].str.contains('riqualificazione') | df['PROGETTO'].str.contains(
↳ 'Riqualificazione')]

```

```

[22]:
NUMERO      BENEFICIARIO \
2         3      Comune di Acquasparta
3         4      Comune di Acquasparta
8         9      Comune di Assisi
9        10      Comune di Assisi
10       11      Comune di Bastia Umbra
15       16      Comune di Cascia
32       33      Comune di Foligno
34       35      Comune di Giano dell'Umbria
39       40      Comune di Gubbio
46       47      Comune di Montecastello di Vibio
63       64      Comune di Parrano
67       68      Comune di Perugia
68       69      Comune di Perugia
85       86      Comune di Spoleto
89       90      Comune di Terni
91       92      Comune di Todi
97       98      Comune di Trevi

                                PROGETTO \
2 Riqualificazione e valorizzazione Palazzo Cesi
3 Completamento Palazzo Cesi. Riqualificazione d...
8 Riqualificazione e adeguamento IAT Area Vasta
9 Valorizzazione degli spazi espositivi di Palaz...
10 Riqualificazione sito archeologico Via Renzini
15 Riqualificazione e adeguamento IAT Area Vasta
32 Riqualificazione e adeguamento IAT Area Vasta
34 Sistema bibliotecario - documentario. Ristrutt...
39 Valorizzazione e riqualificazione del compless...

```

(continues on next page)

(continua dalla pagina precedente)

```

46 Lavori di riqualificazione ed adeguamento impi...
63 Valorizzazione Tane del Diavolo - Riqualificaz...
67 Circuito culturale: Riqualificazione dell'impi...
68 Circuito culturale: Riqualificazione e nuove f...
85 Interventi per il potenziamento e la riqualifi...
89 Archeologia Borghi Cultura e Paesaggi. Area ar...
91 Riqualificazione e adegamento IAT Area Vasta
97 Villa Fabri: Restauro degli apparati decorativ...

```

```

                                PROGRAMMA_OPERATIVO \
2          PAR FSC 2007 - 2013 Azione 3.5.2a
3          PAR FSC 2007 - 2013 Azione 3.5.2a
8          APQ Beni culturali II Atto integrativo
9          POR FESR 2014 - 2020 Azione 5.2.1
10         POR FESR 2007 -2013 Attività 2.2.2
15         APQ Beni culturali II Atto integrativo
32         APQ Beni culturali II Atto integrativo
34         PAR FSC 2007 - 2013 Azione 3.5.2a
39         POR FESR 2007 -2013 Attività 2.2.2
46         PAR FSC 2007 - 2013 Azione 3.5.2a
63         POR FESR 2007 -2013 Attività 2.2.2
67         PAR FSC 2007 - 2013 Azione 3.5.2a
68         PAR FSC 2007 - 2013 Azione 3.5.2a
85         APQ Beni culturali II Atto integrativo
89         PAR FSC 2007 - 2013 Azione 3.5.2a
91         APQ Beni culturali II Atto integrativo
97 PAR FSC 2007 - 2013 Azione 3.5.2a e APQ Beni c...

```

```

                STATO_ATTUAZIONE
2                REALIZZATO
3                REALIZZATO
8                REALIZZATO
9  IN CORSO DI REALIZZAZIONE
10               REALIZZATO
15               REALIZZATO
32               REALIZZATO
34               REALIZZATO
39               REALIZZATO
46  IN CORSO DI REALIZZAZIONE
63               REALIZZATO
67               REALIZZATO
68               REALIZZATO
85               REALIZZATO
89               REALIZZATO
91               REALIZZATO
97               REALIZZATO

```

</div>

[22]: # scrivi qui

```

[22]:  NUMERO          BENEFICIARIO \
2      3             Comune di Acquasparta
3      4             Comune di Acquasparta
8      9             Comune di Assisi

```

(continues on next page)

(continua dalla pagina precedente)

9	10	Comune di Assisi
10	11	Comune di Bastia Umbra
15	16	Comune di Cascia
32	33	Comune di Foligno
34	35	Comune di Giano dell'Umbria
39	40	Comune di Gubbio
46	47	Comune di Montecastello di Vibio
63	64	Comune di Parrano
67	68	Comune di Perugia
68	69	Comune di Perugia
85	86	Comune di Spoleto
89	90	Comune di Terni
91	92	Comune di Todi
97	98	Comune di Trevi

PROGETTO \

2	Riqualificazione e valorizzazione Palazzo Cesi
3	Completamento Palazzo Cesi. Riqualificazione d...
8	Riqualificazione e adeguamento IAT Area Vasta
9	Valorizzazione degli spazi espositivi di Palaz...
10	Riqualificazione sito archeologico Via Renzini
15	Riqualificazione e adeguamento IAT Area Vasta
32	Riqualificazione e adeguamento IAT Area Vasta
34	Sistema bibliotecario - documentario. Ristrutt...
39	Valorizzazione e riqualificazione del compless...
46	Lavori di riqualificazione ed adeguamento impi...
63	Valorizzazione Tane del Diavolo - Riqualificaz...
67	Circuito culturale: Riqualificazione dell'impi...
68	Circuito culturale: Riqualificazione e nuove f...
85	Interventi per il potenziamento e la riqualifi...
89	Archeologia Borghi Cultura e Paesaggi. Area ar...
91	Riqualificazione e adeguamento IAT Area Vasta
97	Villa Fabri: Restauro degli apparati decorativ...

PROGRAMMA_OPERATIVO \

2	PAR FSC 2007 - 2013 Azione 3.5.2a
3	PAR FSC 2007 - 2013 Azione 3.5.2a
8	APQ Beni culturali II Atto integrativo
9	POR FESR 2014 - 2020 Azione 5.2.1
10	POR FESR 2007 -2013 Attività 2.2.2
15	APQ Beni culturali II Atto integrativo
32	APQ Beni culturali II Atto integrativo
34	PAR FSC 2007 - 2013 Azione 3.5.2a
39	POR FESR 2007 -2013 Attività 2.2.2
46	PAR FSC 2007 - 2013 Azione 3.5.2a
63	POR FESR 2007 -2013 Attività 2.2.2
67	PAR FSC 2007 - 2013 Azione 3.5.2a
68	PAR FSC 2007 - 2013 Azione 3.5.2a
85	APQ Beni culturali II Atto integrativo
89	PAR FSC 2007 - 2013 Azione 3.5.2a
91	APQ Beni culturali II Atto integrativo
97	PAR FSC 2007 - 2013 Azione 3.5.2a e APQ Beni c...

STATO_ATTUAZIONE

2	REALIZZATO
3	REALIZZATO
8	REALIZZATO

(continues on next page)

(continua dalla pagina precedente)

```

9  IN CORSO DI REALIZZAZIONE
10      REALIZZATO
15      REALIZZATO
32      REALIZZATO
34      REALIZZATO
39      REALIZZATO
46  IN CORSO DI REALIZZAZIONE
63      REALIZZATO
67      REALIZZATO
68      REALIZZATO
85      REALIZZATO
89      REALIZZATO
91      REALIZZATO
97      REALIZZATO

```

estremi

Trovare tutti i progetti aventi NUMERO incluso tra i limiti indicati nella variabile `estremi` (inclusi)

- **NON** scrivere 10 o 18 nel codice!

```

<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">

```

```

[23]: estremi = (10,18)
      #estremi = (15,21)

      # scrivi qui

df[(df['NUMERO'] >= estremi[0]) & (df['NUMERO'] <= estremi[1])]

```

```

[23]:  NUMERO      BENEFICIARIO \
9      10      Comune di Assisi
10     11      Comune di Bastia Umbra
11     12      Comune di Bettona
12     13      Comune di Bettona
13     14      Comune di Bevagna
14     15  Comune di Campello sul Clitunno
15     16      Comune di Cascia
16     17      Comune di Cascia
17     18      Comune di Cascia

      PROGETTO \
9  Valorizzazione degli spazi espositivi di Palaz...
10  Riqualificazione sito archeologico Via Renzini
11  Sistema museale di Bettona - Completamento del...
12  Lavori di movimentazione e restauro del porton...
13  Completamento Palazzo della Cultura - Allestim...
14  Valorizzazione del centro storico del Castello...
15  Riqualificazione e adeguamento IAT Area Vasta
16  Realizzazione dei servizi innovativi per la va...
17  Polo museale Santa Margherita. Completamento

      PROGRAMMA_OPERATIVO      STATO_ATTUAZIONE
9      POR FESR 2014 - 2020 Azione 5.2.1  IN CORSO DI REALIZZAZIONE

```

(continues on next page)

(continua dalla pagina precedente)

10	POR FESR 2007 -2013 Attività 2.2.2	REALIZZATO
11	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO
12	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO
13	POR FESR 2007 -2013 Attività 2.2.2	REALIZZATO
14	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO
15	APQ Beni culturali II Atto integrativo	REALIZZATO
16	Programma Parallelo al POR FESR 2007 -2013	REALIZZATO
17	PAR FSC 2007 - 2013 Azione 3.5.2a	IN CORSO DI REALIZZAZIONE

</div>

```
[23]: estremi = (10,18)
#estremi = (15,21)

# scrivi qui
```

```
[23]:          NUMERO          BENEFICIARIO \
9           10          Comune di Assisi
10          11      Comune di Bastia Umbra
11          12          Comune di Bettona
12          13          Comune di Bettona
13          14          Comune di Bevagna
14          15  Comune di Campello sul Clitunno
15          16          Comune di Cascia
16          17          Comune di Cascia
17          18          Comune di Cascia

          PROGETTO \
9  Valorizzazione degli spazi espositivi di Palaz...
10 Riqualificazione sito archeologico Via Renzini
11 Sistema museale di Bettona - Completamento del...
12 Lavori di movimentazione e restauro del porton...
13 Completamento Palazzo della Cultura - Allestim...
14 Valorizzazione del centro storico del Castello...
15 Riqualificazione e adeguamento IAT Area Vasta
16 Realizzazione dei servizi innovativi per la va...
17 Polo museale Santa Margherita. Completamento
```

	PROGRAMMA_OPERATIVO	STATO_ATTUAZIONE
9	POR FESR 2014 - 2020 Azione 5.2.1	IN CORSO DI REALIZZAZIONE
10	POR FESR 2007 -2013 Attività 2.2.2	REALIZZATO
11	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO
12	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO
13	POR FESR 2007 -2013 Attività 2.2.2	REALIZZATO
14	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO
15	APQ Beni culturali II Atto integrativo	REALIZZATO
16	Programma Parallelo al POR FESR 2007 -2013	REALIZZATO
17	PAR FSC 2007 - 2013 Azione 3.5.2a	IN CORSO DI REALIZZAZIONE

Stato attuazione PAR

Selezionare solo i progetti operativi PAR (che quindi hanno PAR in PROGETTO_OPERATIVO) e mostrarne il conteggio dello stato d'attuazione

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code" style="display:none">
```

```
[24]: # scrivi qui
adf = df[df['PROGRAMMA_OPERATIVO'].str.contains('PAR')]
adf.groupby(['STATO_ATTUAZIONE'])['STATO_ATTUAZIONE'].count()
```

```
[24]: STATO_ATTUAZIONE
IN CORSO DI REALIZZAZIONE    12
REALIZZATO                    36
Name: STATO_ATTUAZIONE, dtype: int64
```

```
</div>
```

```
[24]: # scrivi qui
```

```
[24]: STATO_ATTUAZIONE
IN CORSO DI REALIZZAZIONE    12
REALIZZATO                    36
Name: STATO_ATTUAZIONE, dtype: int64
```

progetti a Todi

Mostrare il conteggio dei progetti realizzati a Todi

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code" style="display:none">
```

```
[25]: # scrivi qui
tdf = df[df['BENEFICIARIO'] == 'Comune di Todi']
tdf.groupby(['BENEFICIARIO', 'STATO_ATTUAZIONE'])['STATO_ATTUAZIONE'].count()
```

```
[25]: BENEFICIARIO    STATO_ATTUAZIONE
Comune di Todi  IN CORSO DI REALIZZAZIONE    4
                REALIZZATO                    2
Name: STATO_ATTUAZIONE, dtype: int64
```

```
</div>
```

```
[25]: # scrivi qui
```

```
[25]: BENEFICIARIO    STATO_ATTUAZIONE
Comune di Todi  IN CORSO DI REALIZZAZIONE    4
                REALIZZATO                    2
Name: STATO_ATTUAZIONE, dtype: int64
```

Comuni beneficiari

Trovare i comuni beneficiari togliendo il prefisso “Comune di”, e senza duplicati

- **NON** usare cicli `for` o list comprehension
- **SUGGERIMENTO:** usare `pd.unique`¹⁷

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">
```

```
[26]: # scrivi qui
cdf = df[df['BENEFICIARIO'].str.contains('Comune')]

pd.unique(cdf['BENEFICIARIO'].str[10:].str.strip())

[26]: array(['Acquasparta', 'Amelia', 'Assisi', 'Bastia Umbra', 'Bettona',
'Bevagna', 'Campello sul Clitunno', 'Cascia', 'Castel Viscardo',
'Città della Pieve', 'Città di Castello', 'Corciano', 'Deruta',
'Ferentillo', 'Foligno', 'Fratta Todina', 'Giano dell'Umbria',
'Gualdo Tadino', 'Guarda', 'Gubbio', 'Marsciano',
'Montecastello di Vibio', 'Montecchio', 'Montefalco', 'Narni',
'Nocera Umbra', 'Orvieto', 'Otricoli', 'Panicale', 'Parrano',
'Perugia', 'Piegara', 'Polino', 'Preci', 'Sant'Anatolia di Narco',
'Sellano', 'Spello', 'Spoleto', 'Terni', 'Todi', 'Torgiano',
'Trevi', 'Tuoro sul Trasimeno', 'Umbertide'], dtype=object)
```

```
</div>
```

```
[26]: # scrivi qui

[26]: array(['Acquasparta', 'Amelia', 'Assisi', 'Bastia Umbra', 'Bettona',
'Bevagna', 'Campello sul Clitunno', 'Cascia', 'Castel Viscardo',
'Città della Pieve', 'Città di Castello', 'Corciano', 'Deruta',
'Ferentillo', 'Foligno', 'Fratta Todina', 'Giano dell'Umbria',
'Gualdo Tadino', 'Guarda', 'Gubbio', 'Marsciano',
'Montecastello di Vibio', 'Montecchio', 'Montefalco', 'Narni',
'Nocera Umbra', 'Orvieto', 'Otricoli', 'Panicale', 'Parrano',
'Perugia', 'Piegara', 'Polino', 'Preci', 'Sant'Anatolia di Narco',
'Sellano', 'Spello', 'Spoleto', 'Terni', 'Todi', 'Torgiano',
'Trevi', 'Tuoro sul Trasimeno', 'Umbertide'], dtype=object)
```

programma operativo

Creare una nuova colonna `sigla` con i primi 3 caratteri della colonna `PROGRAMMA_OPERATIVO`.

- se la prima parola è maggiore di 3 caratteri, impostare stringa vuota
- **SUGGERIMENTO:** servirà una `transform` con funzione

```
<a class="jupman-sol jupman-sol-toggler" onclick="jupman.toggleSolution(this);" data-jupman-show="Mostra
soluzione" data-jupman-hide="Nascondi">Mostra soluzione</a><div class="jupman-sol jupman-sol-code"
style="display:none">
```

¹⁷ <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.unique.html>

```
[27]: # scrivi qui
def trasf(stringa):
    candidato = stringa.split()[0]
    if len(candidato) == 3:
        return candidato
    else:
        return ''

df['sigla'] = df['PROGRAMMA_OPERATIVO'].transform(trasf)
```

</div>

```
[27]: # scrivi qui
```

```
[28]: df[:18] # nota la riga 16
```

```
[28]:
```

	NUMERO	BENEFICIARIO \
0	1	Comune di Acquasparta
1	2	Comune di Acquasparta
2	3	Comune di Acquasparta
3	4	Comune di Acquasparta
4	5	Comune di Acquasparta
5	6	Comune di Amelia
6	7	Comune di Assisi
7	8	Comune di Assisi
8	9	Comune di Assisi
9	10	Comune di Assisi
10	11	Comune di Bastia Umbra
11	12	Comune di Bettona
12	13	Comune di Bettona
13	14	Comune di Bevagna
14	15	Comune di Campello sul Clitunno
15	16	Comune di Cascia
16	17	Comune di Cascia
17	18	Comune di Cascia

	PROGETTO \
0	Progetto per il recupero, il restauro e la tra...
1	Recupero, restauro e trasformazione in centro ...
2	Riqualificazione e valorizzazione Palazzo Cesi
3	Completamento Palazzo Cesi. Riqualificazione d...
4	Completamento delle opere di restauro e valori...
5	Sistema bibliotecario - documentario. Risaname...
6	Allestimento di Palazzo Monte Frumentario fina...
7	Valorizzazione degli spazi espositivi di Palaz...
8	Riqualificazione e adeguamento IAT Area Vasta
9	Valorizzazione degli spazi espositivi di Palaz...
10	Riqualificazione sito archeologico Via Renzini
11	Sistema museale di Bettona - Compeltamento del...
12	Lavori di movimentazione e restauro del porton...
13	Completamento Palazzo della Cultura - Allestim...
14	Valorizzazione del centro storico del Castello...
15	Riqualificazione e adeguamento IAT Area Vasta
16	Realizzazione dei servizi innovativi per la va...
17	Polo museale Santa Margherita. Completamento

(continues on next page)

(continua dalla pagina precedente)

	PROGRAMMA_OPERATIVO	STATO_ATTUAZIONE \
0	POR FESR 2007 -2013 Attività 2.2.2	REALIZZATO
1	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO
2	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO
3	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO
4	POR FESR 2014 - 2020 Azione 5.2.1	IN CORSO DI REALIZZAZIONE
5	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO
6	POR FESR 2007 -2013 Attività 2.2.2	REALIZZATO
7	PAR FSC 2007 - 2013 Azione 3.5.2a	IN CORSO DI REALIZZAZIONE
8	APQ Beni culturali II Atto integrativo	REALIZZATO
9	POR FESR 2014 - 2020 Azione 5.2.1	IN CORSO DI REALIZZAZIONE
10	POR FESR 2007 -2013 Attività 2.2.2	REALIZZATO
11	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO
12	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO
13	POR FESR 2007 -2013 Attività 2.2.2	REALIZZATO
14	PAR FSC 2007 - 2013 Azione 3.5.2a	REALIZZATO
15	APQ Beni culturali II Atto integrativo	REALIZZATO
16	Programma Parallelo al POR FESR 2007 -2013	REALIZZATO
17	PAR FSC 2007 - 2013 Azione 3.5.2a	IN CORSO DI REALIZZAZIONE

	sigla
0	POR
1	PAR
2	PAR
3	PAR
4	POR
5	PAR
6	POR
7	PAR
8	APQ
9	POR
10	POR
11	PAR
12	PAR
13	POR
14	PAR
15	APQ
16	
17	PAR

- 21 giugno 2019 (6 crediti) : **Testo + soluzioni**¹⁸
- 5 giugno 2019 (6 crediti) : **Testo + soluzioni**¹⁹
- AA 2018/19: Vedere esami seminari **Fondamenti Python**²⁰ (3 crediti, corrisponde al primo modulo) e **Algoritmi Python 2018**²¹ (3 crediti, corrisponde al secondo modulo). Differenze con anno corrente:
 - l'esame sarà un po' più difficile
 - nel primo modulo non ci saranno funzioni nè assert
 - nel secondo modulo ci saranno anche esercizi su Numpy e Pandas

¹⁸ <http://davidleoni.it/etc/sps/exams/2019-06-21-solutions.zip>

¹⁹ <http://davidleoni.it/etc/sps/exams/2019-06-05-solutions.zip>

²⁰ https://docs.google.com/presentation/d/1r4iGiRPjUp9SfLFWrcUznCertVpmO5V9GvxfPkhFnG0/edit#slide=id.g36a9bc8e68_0_9

²¹ https://docs.google.com/presentation/d/1I39iDR_F9TJ8VmnGfUtWZwnwNV4uFVCeRmqY0v1_PwE/edit#slide=id.g399504d837_0_17

Appelli

Per ciascun seminario (modulo 1 e 2), avete a disposizione **due appelli**:

- **Venerdì 16 aprile 2021 9:00-12:00 (modulo 1)**
- **Lunedì 31 maggio 2021 15:00-18:00 (modulo 1 o 2)**
- **Lunedì 14 Giugno 2021 9:00-12:00 (modulo 1 o 2)**
- **Lunedì 28 Giugno 2021 9:00-12:00 (modulo 1 o 2)**

Ricordo che l'**ultima lezione del Modulo 2** sarà **venerdì 28 maggio 9:00-12:00**, in cui faremo anche del **testing** del sistema per scongiurare problemi tecnici.

Prenotazione: entro una settimana prima dell'appello mandate mail a david.leoni@unitn.it indicando quale parte volete dare.

Gli appelli concessi per parte sono due perchè gli studenti che mi chiedono il terzo appello di solito sono anche quelli che arrivano ai primi due e palesemente non hanno alcuna idea di come si scriva un programma. Esercizi da fare ne avete e sicuramente anche un cronometro, quindi penso potete ben valutare da voi quando è il caso di presentarsi. Per dare un'idea, mi aspetto che per ciascuno esercizio di difficoltà tre stelle su SoftPython ci mettiate max 30 min. Vedere anche sezione *esami passati* sul mio sito.

Per entrambe le parti vi chiederò di implementare del codice, per il quale riceverete un voto in base alla percentuale di correttezza. Se qualcosa non funziona in qualche linea, sentitvi liberi di metterci prima una print. Questa seconda parte sarà open book, se volete potete usare stampe del materiale e le slide del corso, più la documentazione ufficiale di Python 3.

Editor: Come editor per l'esame, useremo Jupyter.

Precondizione esame modulo 1: Per affrontarlo decentemente **dovete** aver capito la teoria. A tal proposito, in SoftPython ci sono una quantità spropositata di sezioni intitolate "Domande" (*esempio*²²) tipo «Guarda i seguenti frammenti di codice, e per ciascuno cerca di indovinare quale risultato produce (o se da errore)». Non sono lì a caso: le ho aggiunte perchè ho notato che molto spesso ci si porta all'esame dubbi che poi risultano in tempi lunghissimi passati a debuggare il codice. Il modo corretto per rispondere a quelle domande è prima scrivere (**scrivere con le dita, non pensare!**) da qualche parte quello che ritenete sia il risultato che verrà prodotto, e POI provare ad eseguire il codice per sincerarsi che il risultato pensato sia corretto. Per quanto semplici possano sembrare, vi garantisco che avrete parecchie sorprese.

Se fallite UNA volta il modulo 1, potrete ridarlo successivamente, quel giorno mi comunicherete se vorrete dare il modulo 1 o il modulo 2, vi darò test diversi a seconda della risposta. Se decidete l'esame del modulo 2 e lo passate vi riconoscerò i crediti anche per il modulo 1 (posto che foste iscritti al seminario corrispondente). **NOTA:** se non avete capito bene il materiale del modulo 1, vi garantisco che non riuscirete a passare il modulo 2!

Se fallite DUE volte il modulo 1: non riceverete alcun credito e non potrete dare l'esame per il secondo modulo.

Precondizione esame modulo 2: per affrontarlo serenamente dovrete aver capito bene il primo modulo, per cui se non avete ottenuto risultati soddisfacenti al primo appello, dovrete darvi una mossa!

Se vi prenotate ad un appello e non vi presentate: prenderete 0 per quell'appello, che verrà scalato dagli appelli disponibili. Per essere chiari, non accetto scuse: se vi è atterrato un asteroide sul condominio, vi prendete 0 lo stesso.

Appelli extra / orali / etc: se siete a corto di appelli, potete provare a supplicarmi: se siete **fortunati**, potrei concedervi l'appello extra. Se siete **sfortunati**, potrei avere altri impegni e non essere in grado di donarvi il mio tempo, al che vi consiglierò di provare a fare i due moduli d'informatica alla [summerschool](https://datascience.unitn.it/presentation/)²³ in data science questa estate. Sono tenuti dal sottoscritto, con medesimi contenuti e divisi in 3 crediti ciascuno.

²² <https://it.softpython.org/sequences/sequences-sol.html#Domande-list-comprehension>

²³ <http://datascience.unitn.it/presentation/>

Istruzioni per esame

Tecnicamente, vi conatterete da casa vostra col vostro computer ad un pc che sta nei laboratori di Povo, usando il client VMWare Horizon. Poi userete il vostro smartphone per riprendere voi e lo schermo del vostro pc.

PRIMA DELLA SESSIONE: installate il client VMWare Horizon -> [SCARICA](#)²⁴

QUANDO VI UNITE A ZOOM: seguite la procedura riportata qui sotto:

1. Mettete il VOSTRO SMARTPHONE in una posizione per visualizzare voi e lo schermo durante l'esame. NOTA: lo schermo si deve vedere bene.
2. Connettete ENTRAMBI i vostri dispositivi al seguente link zoom (disponibile 30 minuti prima dell'esame):
Il link apparirà sul Moodle del corso in alto 2 giorni prima dell'esame
3. Rinominate i vostri utenti in Zoom in modo che sia chiaro il vostro nome e che dispositivo state usando (es Leoni Desktop e Leoni Cel). Per farlo, nella pannello Partecipanti mettete il mouse sul vostro nome, apparirà un bottone blue con scritto More, cliccatelo e vedere l'opzione Rename
4. Aprite il **CLIENT** VMWare Horizon che avete installato (**NON il browser!**)
5. **Con il client** connettetevi a: <https://secureview.unitn.it> usando le credenziali universitarie.
 - **NON cercate di aprire il link nel browser**, dovete inserirlo nel client !!
6. cliccate su Esame Povo: Questo vi conatterà ad un pc del laboratorio dove potrete usare Jupyter nella finestra che si aprirà.

DURANTE LA SESSIONE:

1. Evitate di muovere lo smartphone tranne che sia strettamente necessario, o se ricevete istruzioni al riguardo. Da adesso in poi potete dimenticarvi del vostro device portatile e della connessione zoom sul device.
2. Seguite le istruzioni che vi diremo
3. Se doveste avere delle domande, usate il bottone «Alza la mano» nel pannello dei Partecipanti per richiamare la nostra attenzione (nell'applicazione Zoom sul vostro PC). Quando qualcuno sarà disponibile, una breakout room sarà creata dove vi sarà richiesto di entrare. Questo vi permetterà di temporaneamente lasciare il meeting zoom comune e raggiungere una stanza specifica dove trovare qualcuno che risponderà alla vostra questione. Nella stanza di breakout potrete in caso di necessità condividere il vostro schermo (usando il bottone Mostra schermo nel pannello inferiore di Zoom). **IMPORTANTE: QUANDO LASCIATE LA BREAKOUT ROOM CLICcate SU «LASCIA BREAKOUT ROOM» E NON su «Lascia il meeting»** altrimenti vi disconetterete dal meeting e **INVALIDERETE IL VOSTRO ESAME**. Per ulteriori domande, non esitate a contattarci.

Ricevimento

Per orari / luoghi ricevimento, [vedere qui](#)²⁵

Se per caso avete progetti in altri corsi o interesse personale per cui volete usare Python, sono disponibile a dare indicazioni.

In particolare, posso offrire aiuto per

- Installazione
- Comandi Python base
- Errori logici

²⁴ https://my.vmware.com/en/web/vmware/downloads/info/slug/desktop_end_user_computing/vmware_horizon_clients/horizon_8

²⁵ <http://davidleoni.it/office-hours/>

- Lettura / conversione dati
- Formati (CSV / JSON / XML / HTML)
- Cercare dataset
- Tutorial
- Licenze dati & software

Difficile aiutare per

- librerie particolari
- statistiche avanzate
- visualizzazioni incredibili in 3d

CAPITOLO 1

Overview
