SPS

Raccolta esami passati

David Leoni

14 giu 2021

Indice

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.it[5]

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 modulo 1

Seminari Python @Sociologia, Università di Trento

[1] https://www.sociologia.unitn.it/alfresco/download/workspace/SpacesStore/6c44e1d2-b612-44ba-8866-bdd9d910f840/Locandina%20%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 - prendilettere

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

Esempio - data:

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

il tuo codice deve stampare:

```
lett
```

Mostra soluzione

[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
```

[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

[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'}
```

[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)

(continua dalla pagina precedente)

```
```

.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`

Mostra soluzione

[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
```

[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

[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]
```

[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

[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
```

[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

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 contenti quella lettera

- **USA** una list comprehension

Mostra soluzione

[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']
```

[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]
```

Mostra soluzione

[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]
```

[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`

Mostra soluzione

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

# scrivi qui
res = []

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

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

```
roooossppooo
```

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

# 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(db)
{'ottone': ['ottone', 'ottone', 'ottone', 'ottone', 'ottone'],
 'rame'  : ['rame', 'rame', 'rame', 'rame', 'rame', 'rame', 'rame', 'rame'],
 'ferro' : ['ferro']}
```

Mostra soluzione

[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']}
```

[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.it[8]

Mostra soluzione

[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']  ]
```

[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  \
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

                                ASSISTENZA  COD_COMUNE    COMUNE
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 TRENTINO ONLUS  ATTIVITA` CLINICA
124      COOPERATIVA SOCIALE IRIFOR DEL TRENTINO 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 con un reader csv[9] (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.it[10]

Mostra soluzione

[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']]
```

[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

[17]:
```
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]))
```

[17]:
```
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

[19]:
```
#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]))
```

[19]:
```
#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

[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']]
```

[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

Seminari Python - Triennale Sociologia @Università di Trento

.21 Modulo A

.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']
```

Mostra soluzione

[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']
```

[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}
```

Mostra soluzione

[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 efficente
for lettera in testo:
    if lettera in cerca:
        if lettera in conteggi:
            conteggi[lettera] += 1
        else:
            conteggi[lettera] = 1

print(conteggi)

# soluzione 2, meno efficente (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}
```

[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
```

Mostra soluzione

[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
```

[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, guadarono fiumi, si inerpicarono sui monti, e␣
↪infine arrivarono al Tempio"
```

il tuo codice deve produrre

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

Mostra soluzione

[7]:
```
viaggio = "Attraversarono deserti, guadarono 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']
```

[7]:
```
viaggio = "Attraversarono deserti, guadarono 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']
```

[ ]:
```

```

- 21 giugno 2019 (6 crediti) : Testo + soluzioni[11]
- 5 giugno 2019 (6 crediti) : Testo + soluzioni[12]

[11] http://davidleoni.it/etc/sps/exams/2019-06-21-solutions.zip
[12] http://davidleoni.it/etc/sps/exams/2019-06-05-solutions.zip

- AA 2018/19: Vedere esami seminari Fondamenti Python[13] (3 crediti, corrisponde al primo modulo) e Algoritmi Python 2018[14] (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

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, sentitivi 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[15]) 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 testi 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!

[13] https://docs.google.com/presentation/d/1r4iGiRPjUp9SfLFWrcUznCertVpmO5V9GvxfPkhFnG0/edit#slide=id.g36a9bc8e68_0_9
[14] https://docs.google.com/presentation/d/1I39iDR_F9TJ8VmnGfUtWZwnwNV4uFVCeRmqY0vl_PwE/edit#slide=id.g399504d837_0_17
[15] https://it.softpython.org/sequences/sequences-sol.html#Domande-list-comprehension

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[16] in data science questa estate. Sono tenuti dal sottoscritto, con medesimi contenuti e divisi in 3 crediti ciascuno.

Istruzioni per esame

Tecnicamente, vi connetterete 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[17]

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 connetterà 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 trovere qualcuno che risponderà alla vostra questione. Nella stanza di breakout potrete in caso di necessitàcondividere 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 disconnetterete dal meeting e INVALIDERETE IL VOSTRO ESAME. Per ulteriori domande, non esitate a contattarci.

[16] http://datascience.unitn.it/presentation/
[17] https://my.vmware.com/en/web/vmware/downloads/info/slug/desktop_end_user_computing/vmware_horizon_clients/horizon_8

Ricevimento

Per orari / luoghi ricevimento, vedere qui[18]

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
- 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

[18] http://davidleoni.it/office-hours/

CAPITOLO 1

Overview