
Jupman

Raccolta esami passati

David Leoni

16 apr 2021

Copyright © 2021 by David Leoni.

Jupman 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
Ricevimento	7
1 Overview	9

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

Materiale

Il materiale presente in questa pagina è stato tutto trasferito sul sito softpython.it⁴

Eventuali link a materiale non mio li trovate nella pagina [Riferimenti su SoftPython](#)⁵

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](#)⁶ (contatto: supportostudentipovo@unitn.it)

Esami passati

.1 Esame Ven 16, Apr 2021 modulo 1

Seminari Python @Sociologia

[Download exercises](#)

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://softpython.it>

⁵ <https://it.softpython.org/references.html>

⁶ <http://datascience.unitn.it/presentation/>

.2 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
```

.3 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',
            'lampade' : 'C',
            'cavi' : 'F',
```

(continues on next page)

(continua dalla pagina precedente)

```

        'giardinaggio' : 'E'}

magazzino = {}

```

dopo il tuo codice deve risultare:

```

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

```

```

<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]: 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

```

.4 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
```

.5 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]
```

```
<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]: 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
```

.6 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
```

[]:

- 21 giugno 2019 (6 crediti) : [Risultati](#)⁷ | [Correzioni](#)⁸ | [Testo + soluzioni](#)⁹
- 5 giugno 2019 (6 crediti) : [Risultati](#)¹⁰ | [Correzioni](#)¹¹ | [Testo + soluzioni](#)¹²

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

Ricevimento

Per orari / luoghi ricevimento, [vedere qui](#)¹⁵

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

⁷ <http://davidleoni.it/etc/sps/exams/2019-06-21-public-grades.html>

⁸ https://drive.google.com/open?id=14fE1RomyIkoTaxyzd4l_dfStF1UwSYcA

⁹ <http://davidleoni.it/etc/sps/exams/2019-06-21-solutions.zip>

¹⁰ <http://davidleoni.it/etc/sps/exams/2019-06-05-public-grades.html>

¹¹ https://drive.google.com/open?id=1q_4IQRfji2WVcdi-Wlm8S7ForpgxoiwS

¹² <http://davidleoni.it/etc/sps/exams/2019-06-05-solutions.zip>

¹³ https://docs.google.com/presentation/d/1r4iGiRPjUp9SfLFWrcUznCertVpmO5V9GvxfPkhFnG0/edit#slide=id.g36a9bc8e68_0_9

¹⁴ https://docs.google.com/presentation/d/1I39iDR_F9TJ8VmnGfUtWZwnwNV4uFVCeRmqY0v1_PwE/edit#slide=id.g399504d837_0_17

¹⁵ <http://davidleoni.it/office-hours/>

CAPITOLO 1

Overview
