SPS

Raccolta esami passati

David Leoni

26 mag 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

Download exercises

[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 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


```

.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'}
```

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
```

.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`

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


```

.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]
```

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
```

.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

[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


```

```
[ ]:
```

- 21 giugno 2019 (6 crediti) : Risultati[8] | Correzioni[9] | Testo + soluzioni[10]
- 5 giugno 2019 (6 crediti) : Risultati[11] | Correzioni[12] | Testo + soluzioni[13]

AA 2018/19: Vedere esami seminari Fondamenti Python[14] (3 crediti, corrisponde al primo modulo) e Algoritmi Python 2018[15] (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)**

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[16]) 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

[8] http://davidleoni.it/etc/sps/exams/2019-06-21-public-grades.html
[9] https://drive.google.com/open?id=14fE1RomyIkoTaxyzd4l_dfStF1UwSYcA
[10] http://davidleoni.it/etc/sps/exams/2019-06-21-solutions.zip
[11] http://davidleoni.it/etc/sps/exams/2019-06-05-public-grades.html
[12] https://drive.google.com/open?id=1q_4IQRFji2WVcdi-Wlm8S7ForpgxoiwS
[13] http://davidleoni.it/etc/sps/exams/2019-06-05-solutions.zip
[14] https://docs.google.com/presentation/d/1r4iGiRPjUp9SfLFWrcUznCertVpmO5V9GvxfPkhFnG0/edit#slide=id.g36a9bc8e68_0_9
[15] https://docs.google.com/presentation/d/1I39iDR_F9TJ8VmnGfUtWZwnwNV4uFVCeRmqY0vl_PwE/edit#slide=id.g399504d837_0_17
[16] https://it.softpython.org/sequences/sequences-sol.html#Domande-list-comprehension

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!

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[17] 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[18]

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 della simulazione / 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

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

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.

Ricevimento

Per orari / luoghi ricevimento, vedere qui[19]

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

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

CAPITOLO 1

Overview