Fondamenti di Informatica

Ing. Gestionale

a.a. 2022/2023

Esercitazione del 27/04/2023

Tra parentesi trovate (se esiste) il riferimento all'esercizio di riferimento (con possibili modifiche) dal libro di testo del corso:

Horstmann, C., & Necaise, R. D. (2019). *Concetti di Informatica e Fondamenti di Python* (2ª edizione). Maggioli Editore. ISBN: 9788891635433. http://www.apogeoeducation.com/concetti-di-informatica-e-fondamenti-di-python.html

Esercizi per casa della esercitazione precedente

Esercitazione 20/04: Esercizio 2

Scrivere una funzione che data una matrice m x n (cioè con m righe e n colonne) restituisca una nuova matrice (m + 1) x (n + 1), in cui sia stata aggiunta una nuova colonna a destra che rappresenta la somma delle colonne ed una nuova riga sotto che rappresenta la somma delle righe.

Esempio:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 2 & 3 & 6 \\ 4 & 5 & 6 & 15 \\ 5 & 7 & 9 & 21 \end{pmatrix}$$

Nota: una matrice m x n può essere rappresentata come una lista di m elementi, ciascuno dei quali è a sua volta una lista di n elementi che rappresenta una riga.

Commento alla soluzione:

Possiamo rappresentare la matrice di input nel modo seguente:

```
matrice = [[1, 2, 3], [4, 5, 6]]
```

Si tratta di una lista con due elementi corrispondenti alle due righe della matrice. Queste sono state rappresentate a loro volta come delle liste, i cui elementi sono i valori su una determinata riga.

`len(matrice)` ci dà quindi il numero di righe della matrice

Assumendo che la matrice abbia almeno una riga, possiamo considerare una riga qualunque, es. quella con indice 0, e determinarne la lunghezza con `len(matrice[0])` per ottenere il numero di colonne.

Il seguente enunciato inizializza una lista formata da (n+1) zeri (che sarà utilizzata come accumulatore per determinare la somma degli elementi sulle diverse colonne)

```
somma_righe = [0] * (n + 1)
```

(funziona in modo analogo all'operatore di ripetizione delle stringhe visto in precedenza)

Si ricorda che la somma di liste in Python non funziona come la somma tra matrici in algebra. Anziché sommare le liste elemento per elemento, la somma di liste di Python le concatena.

Esercitazione 20/04: Esercizio 3

Si scriva la funzione *ruota90gradiSensoOrario* che prenda in input un'immagine fatta di n x n caratteri disposti su n righe e n colonne e ritorni la stessa immagine ruotata di 90 gradi in senso orario (non preoccuparsi di girare le lettere 😊)

Esempio:

```
immagine = "XXXXX\n      output = "....X\n
           ..X..\n                ....X\n
           ..X..\n               XXXXX\n
           ..X..\n                ....X\n
           ..X..\n"               ....X\n"
```

n = 5

Ricordarsi: l'accapo vale un carattere '\n' e, per semplicità, l'ultima riga ha un accapo.

Commento alla soluzione:

L'immagine di output può essere costruita carattere per carattere, iterando sulle colonne dell'immagine di input e per ciascuna di esse, scorrendola dal basso verso l'alto.

Per implementare questa operazione, è utile trasformare la immagine di input da semplice stringa in una sorta di matrice. A tale scopo, possiamo usare il metodo *split* delle stringhe.

Senza argomenti, il metodo *split* suddivide la stringa in corrispondenza di qualsia spazio e ignora eventuali stringhe vuote prodotte.

```
" a   b c  ".split()
```

produce

```
['a', 'b', 'c']
```

È possible passare un delimitatore differente ma in questo caso potrebbero essere generate stringhe vuote, e.g. due delimitatori adiacenti o delimitatore all'inizio/fine della stringa.

```
"a,b,c".split(",")
```

Produce

```
['a', 'b', 'c']
```

```
",a,,b,c,".split(",")
```

Produce

```
['', 'a', '', 'b', 'c', '']
```

Il metodo *join* è il "duale" del metodo *split*:

```
lista_stringhe = ["a", "b", "c"]

",".join(lista_stringhe)
```

Produce

```
'a,b,c'
```

Esercitazione 20/04: Esercizio 4

Scrivere una funzione che converta una stringa contenente una data dal formato "gg/mm/aaaa" al formato "aaaa/mm/gg".

Commento alla soluzione:

Se proviamo a invertire una stringa che rappresenta una data

```
"27/04/2023"[::-1]
```

otteniamo una cosa tipo questa

```
'3202/40/72'
```

Abbiamo effettivamente messo anno, mese e giorno nella posizione corretta, ma abbiamo corrotto i loro valori.

La soluzione corretta è prima suddividere la stringa (usando il metodo *split*) nelle componenti della data, e quindi invertire questa lista.

Esercitazione 20/04: Esercizio 5

Scrivere una funzione che data in input una lista di interi restituisca sia il numero di pari sia il numero di dispari.

Commento alla soluzione:

L'aspetto interessante di questo esercizio è la possibilità di restituire più valori da una funzione:

```
return numero_pari, numero_dispari
```

al chiamante viene restituita una tupla contenente i valori indicati.

Nota:

Le tuple sono simili alle liste, ma a differenza di quest'ultime sono immutabili.

Le tuple possono essere scritte usando le parentesi tonde (al posto delle parantesi quadre usate invece per le liste).

```
tupla = (1, 2, 3, 4)
```

Possiamo accedere ai singoli elementi della tupla attraverso il loro indice. Usando la tupla appena vista, l'espressione *tupla[2]* restituisce il valore 3

Un enunciato come il seguente

tupla[2]=2

solleva un'eccezione perché le tuple sono immutabili

Traceback (most recent call last):

File "C:\[…] \Python\Python310\lib\code.py", line 90, in runcode

```
  exec(code, self.locals)
File "<input>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Esercitazione 20/04: Esercizio 6

Scrivere una funzione che data una lista restituisce True se e solo se la lista è ordinata in senso crescente (stretto)

Alcuni esempi:

[] → True

[1] → True

[1, 2] → True

[1, 2, 3] → True

[2, 1, 3] → False

[1, 1, 2] → False

Commento alla soluzione:

Occorre confrontare elementi in posizioni adiacenti.

Esercitazione 20/04: Esercizio 7

Scrivere una funzione che data una lista di interi rimuova da essa (in place, senza farne una copia) tutti i numeri pari.

Commento alla soluzione:

Leggere commenti nel codice.

Esercitazione 20/04: Esercizio 8

Scrivere una funzione che data una lista di interi restituisca True se e solo se ci sono solo numeri pari. In questo caso, a differenza di altre funzioni standard di Python viste in precedenza, si vuole che una lista vuota restituisca True.

Commento alla soluzione:

Un possibile approccio è scorrere tutta lista e contare i numeri pari. Al termine della visita, si restituirà *True* se e solo se il contatore è uguale alla lunghezza della lista.

Un modo alternativo e potenzialmente più efficiente è considerare che *una lista è formata da soli pari se e solo se non c'è alcun dispari*. In questo modo:

- Inizializziamo una variabile di controllo a *True*
- Finché la variabile di controllo ha valore *True* e non abbiamo terminato di scorrere la lista, analizziamo ciascun elemento della lista, impostando la variabile di controllo a *False* se incontriamo un dispari.

Esercitazione 20/04: Esercizio 9

Scrivere una funzione che data una lista di interi restituisca True se e solo se c'è almeno un numero pari.

Commento alla soluzione:

A differenza dell'esempio precedente, stiamo andando alla ricerca all'interno del ciclo di un elemento pari per restituire *True*.