

Si consideri una matrice di numeri interi di dimensione $r \times c$, contenente l'esito degli esami per i c corsi del primo anno di Ingegneria Industriale per r studenti. I valori ammessi nella matrice sono interi compresi tra 18 e 30 (inclusi), se l'esame è stato superato, e 0, altrimenti (cioè se l'esame non è stato ancora superato). Per semplicità, non viene considerato il caso della lode. L'ultima colonna rappresenta il corso di Fondamenti di Informatica.

Si consideri il seguente esempio, relativo ad una matrice 10×8 (10 studenti, 8 esami).

24	18	0	22	20	0	0	0
29	18	20	24	0	22	0	0
20	21	27	0	21	0	22	0
21	26	25	0	26	0	0	0
25	27	26	19	0	0	0	0
30	25	0	23	0	0	0	0
20	22	20	20	25	23	0	0
23	21	0	27	0	25	0	0
22	27	20	25	20	0	0	0
20	19	24	18	0	0	0	0

La matrice può essere rappresentata in Python come una lista di liste di interi.

Si consideri inoltre che ogni corso è associato ad un numero di CFU (Crediti Formativi Universitari). I crediti associati a c corsi possono essere rappresentati in Python come una lista di c interi. Con riferimento all'esempio precedente, la lista può contenere ad esempio i seguenti valori:

```
crediti = [9, 9, 6, 9, 12, 9, 6, 9]
```

I dati dell'esempio sono da interpretare come segue: il primo studente ha preso 24 nell'esame del primo corso (da 9 CFU), ha preso 18 nell'esame del secondo corso (da 9 CFU), non ha ancora superato l'esame del terzo corso (da 6 CFU), ha preso 22 nell'esame del quarto corso (da 9 CFU). E così via.

1. Definire la funzione `stampa_matrice`

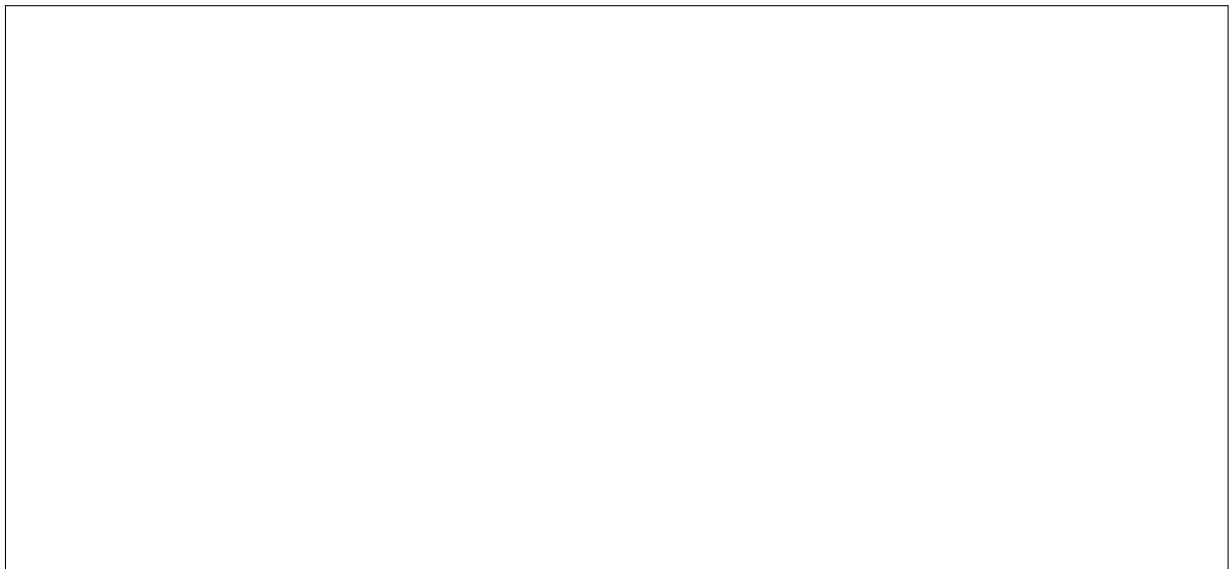
- **Parametri di ingresso:**

- `matrice_studenti`: lista di liste di interi.

- **Restituisce:** non restituisce alcunché.

- **Descrizione:** la funzione permette di stampare riga per riga la matrice fornita in ingresso, separando gli elementi con uno spazio e riservando due caratteri per la rappresentazione di ogni valore.

- **Esempio:** La tabella riportata sopra è un esempio di stampa di una matrice 10×8 .



2. Definire la funzione `inserisci_esiti_fondamenti`

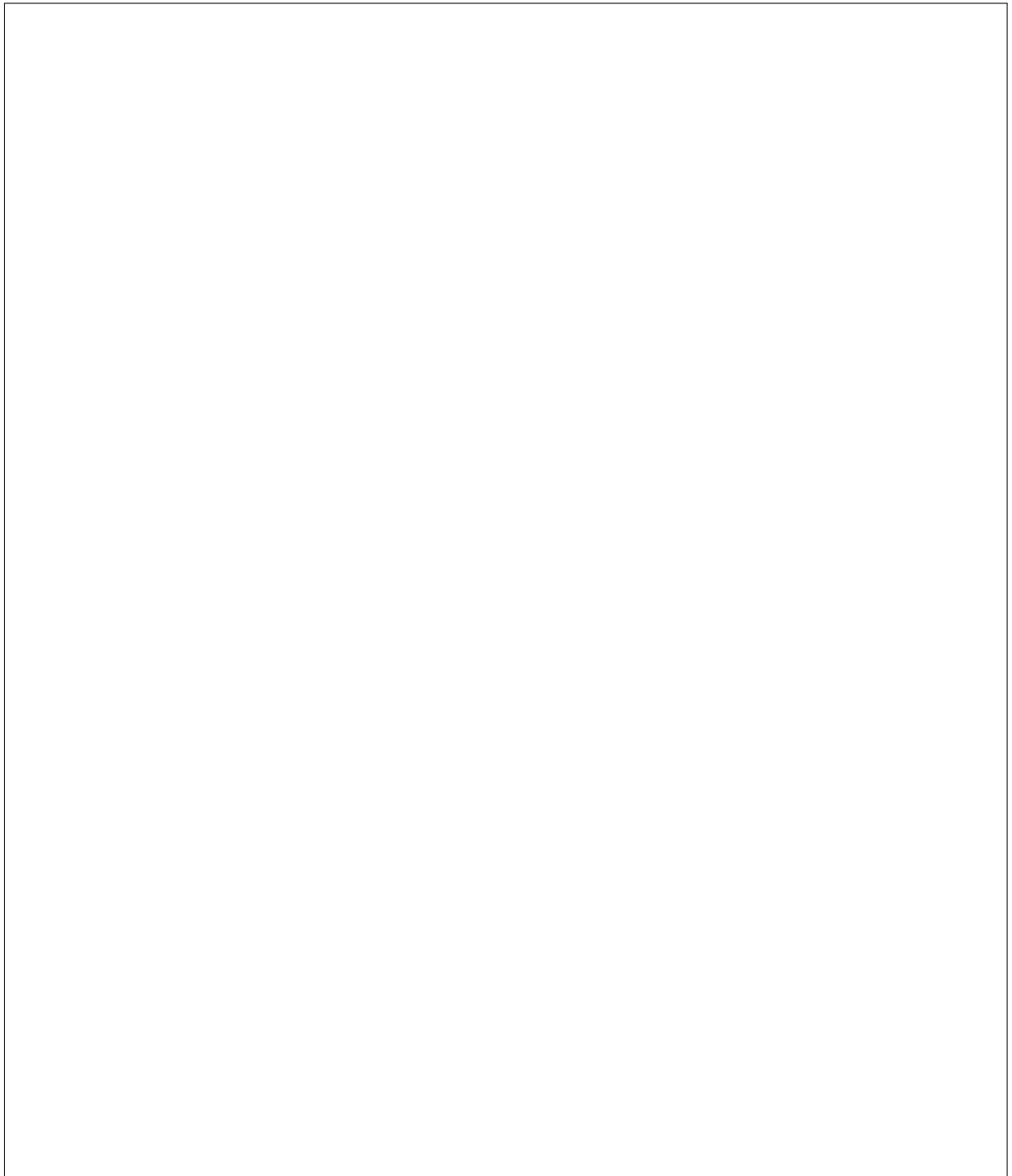
- **Parametri di ingresso:**
 - `matrice_studenti`: lista di liste di interi.
- **Restituisce:** non restituisce alcunché.
- **Descrizione:** la funzione permette di acquisire da tastiera un numero di valori pari al numero di righe di `matrice_studenti`, di controllare la validità dei valori acquisiti, e di modificare `matrice_studenti` memorizzando i valori validi acquisiti nell'ultima colonna della matrice. Un valore è considerato valido se è un intero compreso tra 18 e 30 (inclusi) oppure pari a 0. In caso di valore non valido, la funzione deve richiedere nuovamente l'inserimento fino a quando non viene fornito un valore valido.

3. Definire la funzione `calcola_media_pesata`

- **Parametri di ingresso:**
 - `voti_studente`: lista di interi.
 - `cfu_corsi`: lista di interi.
- **Restituisce:** un float.
- **Descrizione:** la funzione calcola la media pesata dei valori non nulli in `voti_studente`. Ciascun valore viene pesato in base all'intero presente nella lista `cfu_corsi` alla medesima posizione. I valori di `voti_studente` pari a 0 non contribuiscono al calcolo della media. Si assuma che le due liste abbiano la stessa lunghezza.
- **Esempio:** dati gli esempi di input `voti_studente = [24, 30, 0, 0]` e `cfu_corsi = [6, 12, 9, 6]`, la funzione restituisce 28.0.

4. Definire la funzione `trova_esame_media_massima`

- **Parametri di ingresso:**
 - `matrice_studenti`: lista di liste di interi.
- **Restituisce:** una tupla composta da un intero e da un float.
- **Descrizione:** la funzione calcola, per ciascuna colonna della matrice `matrice_studenti`, la media aritmetica dei valori non nulli presenti nella colonna. I valori pari a 0 non contribuiscono al calcolo della media. Se una colonna contiene esclusivamente valori nulli, la sua media è considerata pari a 0. Tra tutte le colonne, individua quella con media massima e restituisce la media calcolata e l'indice della colonna corrispondente.
- **Esempio:** con riferimento alla matrice riportata nella pagina introduttiva, la colonna con la media più alta (23.4) è la prima. La funzione restituisce dunque la tupla (23.4, 0).



Legenda

s, **s1**: stringa **a**, **b**: numero **i**, **j**, **k**, **n**: intero **x**: elemento generico
l, **l1**: lista **d**, **d1**: dizionario **r**, **r1**: set **t**, **t1**: tupla

Principali funzioni built-in (ordine alfabetico)

abs(a): restituisce il valore assoluto di un numero
enumerate(iterable): restituisce un iteratore i cui elementi sono tuple contenenti il conteggio e il valore ottenuto dall'iterazione su iterable
input(s): scrive il prompt **s** sullo standard output e restituisce stringa acquisita
isinstance(object, classinfo): restituisce **True** se **object** è del tipo **classinfo** o di una sottoclasse di **classinfo**. Altrimenti, restituisce **False**
len(x): restituisce la lunghezza (numero di elementi) di **x**. **x** può essere una sequenza (ad es.: string, list, tuple, range) o un contenitore (ad es.: dict, set)
max(iterable, *, key=None), max(arg1, arg2, *args, key=None): restituisce il massimo nell'iterabile, o il massimo tra due o più argomenti. La funzione **key** serve a calcolare il valore da usare per confrontare gli elementi

min: analogo a **max**
print(*x, sep=' ', end='\n'): stampa oggetti, separati da **sep** e seguiti infine da **end**
range(j), **range(i, j, k)**: restituisce una sequenza immutabile per progressione aritmetica
reversed(seq): restituisce un iteratore che percorre **seq** in ordine inverso
round(a, n): arrotonda il valore di **a** all'intero più vicino o ad **n** cifre decimali
sorted(iterable, key=None, reverse=False): restituisce una nuova lista ordinata degli elementi di **iterable**. La funzione **key** serve a calcolare il valore da usare per confrontare gli elementi. **reverse=True** inverte l'ordine
sum(iterable): restituisce la somma dei valori dell'iterabile
zip(*iterables): itera in parallelo su diversi iterabili e restituisce un iteratore costituito da tuple. La *i*-esima tupla è costituita dall'*i*-esimo elemento di ciascuno degli iterabili

Indexing e Slicing

seq[i]: restituisce l'elemento di indice **i** della sequenza
seq[i:j]: restituisce una sottosequenza di **seq** dall'indice **i** all'indice **j** (escluso)
seq[i:j:k]: restituisce una sottosequenza di **seq** da indice **i** a indice **j** (escluso) con step **k**

Modulo Random

choice(seq): restituisce un elemento casuale dalla sequenza **seq**
choices(seq, k=1): restituisce una lista di **k** elementi scelti da **seq** con reinserimento
random(): restituisce un float casuale in $[0,1)$
randint(i, j): restituisce un intero casuale tra **i** e **j** (inclusi)
sample(seq, k): restituisce una lista di **k** elementi unici scelti da **seq** senza reinserimento
shuffle(seq): rimescolamento della sequenza **seq** in-place
uniform(a, b): restituisce un float casuale tra **a** e **b** (inclusi)

Principali funzioni del modulo math

math.cos(a), **math.sin(a)**, **math.exp(a)**, **math.log(a)**, **math.log2(a)**, **math.sqrt(a)**

Stringhe e principali metodi

Trasformazioni (non in-place): **s.lower()**, **s.upper()**, **s.replace(s1,s2)**, **s.strip()**
Controlli: **s.startswith(s1)**, **s.endswith(s1)**, **s.islower()**, **s.isupper()**, **s.isdigit()**
Da lista a stringa: **s.join(l1)**, Da stringa a lista: **s.split(sep)**
Altro: **s.count(s1)**, **s.index(s1)**
Concatenazione: **s1 + s2** Replicazione: **s1 * n** Membership: **s in s1**

Liste e principali metodi

Modifiche (in-place): **l.append(x)**, **l.extend(l1)**, **l.insert(i, x)**, **l.pop(i)**, **l.remove(x)**, **l.reverse()**, **l.sort(key=None, reverse=False)**
Altro: **l.count(x)**, **l.index(x)**
List comprehension: **l = [expression for item in iterable]**
Concatenazione: **l1 + l2** Replicazione: **l1 * n** Membership: **x in l**

Tuple e principali metodi

Altro: **t.count(x)**, **t.index(x)**
Concatenazione: **t1 + t2** Replicazione: **t1 * n** Membership: **x in t**

Insiemi e principali metodi

Modifiche (in-place): **r.add(x)**, **r.remove(x)**, **r.discard(x)**, **r.clear()**
Operazioni: **r.difference(r1)** (oppure **r - r1**), **r.symmetric_difference(r1)** (oppure **r^r1**), **r.union(r1)** (oppure **r|r1**), **r.intersection(r1)** (oppure **r&r1**)
Controlli: **r.issubset(r1)**, **r.issuperset(r1)**, **r.isdisjoint(r1)**
Membership: **x in r1**

Dizionari e principali metodi

Modifiche (in-place): **d1.update(d2)**, **d.pop(key)**
Viste: **d.keys()**, **d.values()**, **d.items()**
Accesso: **d[key]**, **d.get(key, x=None)**
Aggiunta o modifica: **d[key] = val**
Membership: **key in d1**
Dict comprehension: **d = {key:val for item in iterable}**

f-string

Esempio numeri (cifre decimali): **f"{3.14:.1f}"** → **'3.1'**
Esempio stringa con allineamento (<, >, ^): **f"{'ciao':<10}"** → **'ciao.....'**
f"{'Trieste':<10} {'10:>3}" → **'Trieste... 10'**
f"{'Udine':<10} {'7:>3}" → **'Udine.....7'**

Principali eccezioni

FileNotFoundError, **IndexError**, **KeyError**, **ValueError**, **Exception** (eccezione generica)