```
import random

class Matrice():

    #Crea una matrice di r righe e c colonne
    #tutti i valori sono impostati di default a 0

    def __init__(self,r,c):
        self.row=r
        self.col=c
        self.n=[]
        for i in range(self.row):
            m=[]
            for j in range(self.col):
                m.append(0)
            self.n.append(m)

    #Stampa tutti gli elementi di una matrice, disposti per riga

    def stampa(self):
        for righe in self.n:
            print righe

    #Imposta casualmente i valori contenuti in una matrice

    def riempi(self):
        for i in range(self.row):
            for j in range(self.col):
                self.randVal(i,j)

    #Restituisce il valore alla posizione (r;c)

    def getVal(self,r,c):
        return self.n[r-1][c-1]

    #Imposta un valore casuale alla posizione (r;c)

    def randVal(self,r,c):
        self.n[r-1][c-1]=random.randint(0,3)

    #Imposta un valore v alla posizione (r;c)

    def setVal(self,r,c,v):
        self.n[r-1][c-1]=v
        #!!!
        #!!!
        #self.n[r-1][c-1]=round(float(v),2)?????
        #!!!
        #!!!

    #Restituisce la matrice somma di due matrici a,b passate in
input

    @staticmethod
```

```
    def somma(a,b):
        if (a.row==b.row and a.col==b.col):
            Terza=Matrice(a.row,b.row)
            for r in range(Terza.row):
                for c in range(Terza.col):
                    Terza.setVal(r,c,a.getVal(r,c)+b.getVal(r,c))
            return Terza
        else:
            print "matrici non compatibili"

    #Restituisce la matrice prodotto riga-colonna tra due matrici
a,b passate in input

    @staticmethod
    def prodottoRC(a,b):
        if (a.col==b.row):
            Terza=Matrice(a.row, b.col)
            for i in range(Terza.row):
                for j in range(Terza.col):
                    v=0
                    for k in range(a.col):
                        v+=a.getVal(i,k)*b.getVal(k,j)
                    Terza.setVal(i,j,v)
            return Terza
        else:
            print "matrici non compatibili"

    #Restituisce la matrice prodotto tra una matrice a e un valore k

    @staticmethod
    def prodottoS(a,k):
        Terza=Matrice(a.row, a.col)
        for i in range(Terza.row):
            for j in range(Terza.col):
                Terza.setVal(i,j,a.getVal(i,j)*k)
        return Terza

    #Restituisce la sottomatrice della matrice su cui viene invocata
e a cui vengono tolte la riga r e la colonna c

    def sotto(self,r,c):
        if (self.row == 1 or self.col == 1):
            print "Impossibile creare una matrice"
        else:
            Sotto=Matrice(self.row-1, self.col-1)
            ii=0
            jj=0
            for i in range(self.row):
                for j in range(self.col):
                    if (i != r and j != c):
                        if (i > r):
                            ii=i-1
                        else:
                            ii=i
```

```
                        if (j > c):
                            jj=j-1
                        else:
                            jj=j
                        Sotto.setVal(ii, jj, self.getVal(i,j))
            return Sotto

    #Restituisce la matrice trasposta della matrice su cui e'
invocata

    def trasposta(self):
        Trasposta=Matrice(self.row, self.col)
        for i in range(self.row):
            for j in range(self.col):
                Trasposta.setVal(j, i, self.getVal(i,j))
        return Trasposta

    #Restituisce il determinante della matricesu cui e' invocata

    def determinante(self):
        d = 0
        if (self.row == 0 or self.col == 0 or self.row != self.col):
            return d
        elif (self.row == 1):
            return self.getVal(1,1)
        else:
            for j in range(self.col):
                d += ((-1) ** (j+1)) * self.getVal(1,j) *
self.sotto(1,j).determinante()
        return d

    #Restituisce la matrice inversa di quella su cui e' invocata

    def inversa(self):
        det = self.determinante()
        if (det == 0):
            print "Matrice singolare"
        else:
            Inversa=Matrice(self.row, self.col)
            for i in range(self.row):
                for j in range(self.col):
                    v=float(((-1) ** (i+j)) *
self.sotto(i,j).determinante() / det)
                    Inversa.setVal(i,j,v)
            Inversa=Inversa.trasposta()
            return Inversa

    #Verifica se la matrice b e' l'inversa della matrice a

    #@staticmethod
    #def ifInversa(a,b):
    #    if (a.row==b.row and a.col==b.row and a.row==a.col):
    #        v=Matrice(a.row,b.row)
    #        bool=True
```

```
    #        for i in range(v.row):
    #            for j in range(v.col):
    #                if((i==j and v.getVal(i,j)==1.00)or(i!=j and
v.getVal(i,j)==0.00)):
    #                    bool=True
    #                else:
    #                    bool=False
    #                    break
    #        return bool


#Debugging
n=raw_input("quante righe?")
m=raw_input("quante colonne?")
n=int(n)
m=int(m)
Prima=Matrice(n,m)
Prima.stampa()
print "Matrice vuota"
Prima.riempi()
Prima.stampa()
print "+++++++"
Seconda=Matrice(n,m)
Seconda.riempi()
Seconda.stampa()
print "======="
Matrice.somma(Prima, Seconda).stampa()
print ".-.-.-.-.-.-.-.-.-.-.-.-.-."
Prima.stampa()
print "********"
Seconda.stampa()
print "========"
Matrice.prodottoRC(Prima, Seconda).stampa()
print ".-.-.-.-.-.-.-.-.-.-.-.-.-."
Prima.stampa()
print "********"
print(10)
print "========"
Matrice.prodottoS(Prima,10).stampa()
print ".-.-.-.-.-.-.-.-.-.-.-.-.-."
Prima.stampa()
x=2
y=2
print "Tolte la riga 2 e la colonna 2:"
Prima.sotto(x,y).stampa()
print ".-.-.-.-.-.-.-.-.-.-.-.-.-."
Prima.stampa()
print "Trasposta:"
Prima.trasposta().stampa()
print ".-.-.-.-.-.-.-.-.-.-.-.-.-."
Prima.stampa()
print "Il determinante e':" + str(Prima.determinante())
print ".-.-.-.-.-.-.-.-.-.-.-.-.-."
Prima.stampa()
```

```
print "********"
Prima.inversa().stampa()
print "========"
Matrice.prodottoRC(Prima,Prima.inversa()).stampa()
#print Matrice.ifInversa(Prima,Prima.inversa()):
```