```
"""
     ESAME SEMINARIO FONDAMENTI ALGORITMI PYTHON   14 Giugno 2018


    - NON CAMBIATE IL NOME DELLE FUNZIONI
        - SE VOLETE POTETE AGGIUNGERE FUNZIONI VOSTRE, MA DOVETE SEMPRE
          RICHIAMARLE DALLE FUNZIONI CHE VI HO SCRITTO IO

    - QUANDO SCRIVO 'RITORNA', LA FUNZIONE DEVE _RITORNARE_ DEI VALORI,
      NON STAMPARLI CON LA PRINT !!!!

    - COMUNQUE, PRINT ADDIZIONALI DI DEBUG SONO CONCESSE

    - NON RIASSEGNATE I PARAMETRI IN INPUT! TRADOTTO: NON SCRIVETE _MAI_
      QUALCOSA DEL GENERE !!!

          def f(x):
             x = .....

    - OGNI FUNZIONE E' SEGUITA DAGLI ASSERT DI TEST, SE QUANDO ESEGUITE
      IL CODICE NON VEDETE ERRORI PROBABILMENTE IL CODICE E' GIUSTO
      (MA NON FIDATEVI TROPPO, I TEST NON SONO ESAUSTIVI !)


    - USARE EDITOR SPYDER.
      - Per eseguire tutto lo script: F5
      - Per eseguire solo la linea corrente o la selezione:  F9

    *** DA FARE: IMPLEMENTATE QUESTE 4 FUNZIONI: ***

      - medie
      - potenza
      - toepliz
      - quadranti
"""

'''
Esercizio 1

Dato un dizionario strutturato ad albero riguardante i voti di
uno studente in classe V e VI, RESTITUIRE un array contenente
la media di ogni materia

Esempio:

medie([
  {'id' : 1, 'subject' : 'math', 'V' : 70, 'VI' : 82},
  {'id' : 1, 'subject' : 'italian', 'V' : 73, 'VI' : 74},
  {'id' : 1, 'subject' : 'german', 'V' : 75, 'VI' : 86}
])

ritorna

[ (70+82)/2 , (73+74)/2, (75+86)/2 ]

ovvero

[ 76.0 , 73.5, 80.5 ]


'''

def medie(lista):
    # scrivi qui



# INIZIO TEST - NON TOCCARE !
import math


'''
Verifica che i numeri float in lista1 siano simili a quelli di lista2
'''
def is_list_close(lista1, lista2):
  if len(lista1) != len(lista2):
      return False

  for i in range(len(lista1)):
      if not math.isclose(lista1[i], lista2[i]):
          return False

  return True

assert is_list_close(
    medie([
            {'id' : 1, 'subject' : 'math', 'V' : 70, 'VI' : 82},
```

```
            {'id' : 1, 'subject' : 'italian', 'V' : 73, 'VI' : 74},
            {'id' : 1, 'subject' : 'german', 'V' : 75, 'VI' : 86}
            ]),
        [ 76.0 , 73.5, 80.5 ])
# FINE TEST


'''
Esercizio 2


RESTITUISCE un generatore che produce una sequenza dei primi
n elevati al quadrato

Esempio

potenza(4) resituirà

0 (0*0)
1 (1*1)
4 (2*2)
9 (3*3)
16 (4*4)


'''
def potenza(n):
  # scrivi qui


# INIZIO TEST - NON TOCCARE !

import types
assert isinstance(potenza(0), types.GeneratorType) # verifico
# che pari ritorni un generatore e non una lista

assert list(potenza(0)) == [0]
assert list(potenza(1)) == [0,1]
assert list(potenza(2)) == [0,1,4]
assert list(potenza(3)) == [0,1,4,9]
assert list(potenza(4)) == [0,1,4,9,16]

# FINE TEST



'''
Esercizio 3

ATTENZIONE: IN QUESTO ESERCIZIO NUMPY NON DEVE ESSERE UTILIZZATO

RESTITUISCE True se la matrice come lista di liste in input è Toeplitz,
mentre RESTITUISCE False se non lo è.

Una matrice è Toeplitz se e solo se tutti gli elementi su ogni diagonale
contiene gli stessi elementi.

assumiamo che la matrice contenga sempre almeno una riga di almeno
un elemento


SUGGERIMENTO: usare due for, nel primo scorrere la matrice per righe,
nel secondo per colonne

Chiedersi:
- da che riga occorre partire per la scansione? La prima è utile?
- da che colonna occorre partire per la scansione? La prima è utile?
- se scorriamo le righe dalla prima verso l'ultima e stiamo esaminando
un certo   numero ad una certa riga, che condizione deve rispettare
quel numero  affinchè la matrice sia toepliz ?

ESEMPIO:

matrix = [[1,2,3,4],[5,1,2,3],[9,5,1,2]]

1 2 3 4
5 1 2 3
9 5 1 2

Su ogni diagonale ci sono gli stessi numeri e quindi viene restituito True

1 2 3 4
5 1 4 3
9 3 1 2

Restituisce False. Ci sono due diagonali con numeri diversi:
(5,3) e (2,4,2)
```

```
'''

def toepliz(matrix):

    # scrivi qui


# INIZIO TEST - NON TOCCARE !
assert toepliz([[1]]) == True
assert toepliz([[3,7],
                [5,3]]) == True
assert toepliz([[3,7],
                [3,5]]) == False
assert toepliz([[3,7],
                [3,5]]) == False
assert toepliz([[3,7,9],
                [5,3,7]]) == True
assert toepliz([[3,7,9],
                [5,3,8]]) == False

assert toepliz([[1,2,3,4],
                [5,1,2,3],
                [9,5,1,2]]) == True

assert toepliz([[1,2,3,4],
                [5,9,2,3],
                [9,5,1,2]]) == False
# FINE TEST


'''
Esercizio 4

ATTENZIONE: IN QUESTO ESERCIZIO DEVE ESSERE UTILIZZATO NUMPY

Data una matrice 2n * 2n, dividere la matrice in 4 parti quadrate uguali
(vedi esempio per capire meglio) e RESTITUIRE una NUOVA matrice 2 * 2
contente la media di ogni quadrante

Si assume che la matrice sia sempre di dimensioni pari

SUGGERIMENTO: per dividere per 2 e ottenere un numero intero,
usare l'operatore //


Esempio:

 1, 2 , 5 , 7
 4, 1 , 8 , 0
 2, 0 , 5 , 1
 0, 2 , 1 , 1

si divide in

  1, 2 | 5 , 7
  4, 1 | 8 , 0
-----------------
  2, 0 | 5 , 1
  0, 2 | 1 , 1

e si restituisce

  (1+2+4+1)/ 4  | (5+7+8+0)/4                       2.0 , 5.0
  ------------------------------           =>       1.0 , 2.0
  (2+0+0+2)/4   | (5+1+1+1)/4


'''

import numpy as np

def quadranti(matrice):
    # scrivi qui



# INIZIO TEST - NON TOCCARE !

assert np.allclose(
    quadranti(np.array([
                          [3.0, 5.0],
                          [4.0, 9.0],
                      ])),
              np.array([
                          [3.0, 5.0],
                          [4.0, 9.0],
```

```
                        ])
      )

assert np.allclose(
    quadranti(np.array([
                          [1.0, 2.0 , 5.0 , 7.0],
                          [4.0, 1.0 , 8.0 , 0.0],
                          [2.0, 0.0 , 5.0 , 1.0],
                          [0.0, 2.0 , 1.0 , 1.0]
                        ])),
              np.array([
                          [2.0, 5.0],
                          [1.0, 2.0]
                        ]))

# FINE TEST
```