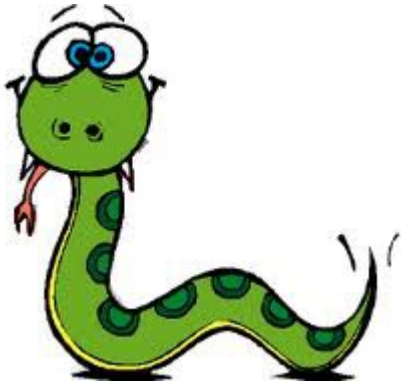
Esercitazione 7

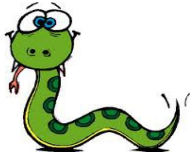


Termine consegna lavori:
Da Mercoledì 11 dicembre ore 00:01
A Martedì 17 dicembre ore 23:59

I lavori dovranno essere salvati all'interno di una cartella che dovrà contenere solo ciò che volete venga consegnato.
Da dentro questa cartella (in modalità terminal) dovrete digitare il comando:

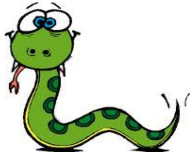
consegna consegna7

Dopo aver digitato tale comando e battuto invio, vi verrà visualizzata la lista di tutto ciò che avete inviato.
Potete fare invii multipli però verrà da noi verrà visto solo l'ultimo effettuato.
E' obbligatorio che all'interno di ogni file sia riportato il vostro nome, cognome e numero di matricola.



Sia data la seguente classe:

```
class Punto:
    def __init__(self, x=0, y=0):
        self.x = x
        self.y = y
    def __str__(self):
        return '('+str(self.x)+','+str(self.y)+')'
    def __eq__(self,altropunto):
        return (self.x == altropunto.x and self.y ==
altropunto.y)
    def __add__(self,altropunto):
        return Punto(self.x+altropunto.x,self.y+altropunto.y)
```



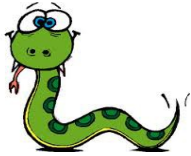
Estendere la classe definita precedentemente aggiungendo:

1) un metodo distanza(self, altropunto=Punto(0,0))
che restituisce la distanza euclidea del punto da un altro punto passato come parametro, oppure la distanza dall'origine se non viene passato nessun punto

2) il metodo speciale __lt__(self,altropunto)
che definisce l'operatore '<' tra punti, e restituisce True se la distanza del punto dall'origine è più piccola di quella relativa all'altro punto

3) Una nuova classe segmento, contenente due punti come estremi, e dotata di un metodo lunghezza(self) che restituisce la lunghezza del segmento, evitando di ricalcolarla ad ogni nuova invocazione. Per la classe segmento definire anche un metodo interseca(self,altroseg) che verifica se due segmenti si intersecano.

4) Provare il funzionamento del programma calcolando la distanza del punto P1(3,4) dall'origine verificando se P1 è "minore" del punto P2(1,5) definendo un segmento con estremi uguali a P1 e P2, e stampandole la lunghezza.



5) Definire poi una classe MatriceQ, che rappresenti una matrice quadrata

- Definire un metodo stampa(), che stampi la matrice

Esempio:

```
>>> m = MatriceQ([[1,2,3],[4,5,6],[7,8,9]])
>>> m.stampa()
 1 2 3
 4 5 6
 7 8 9
```

- Definire poi gli operatori + e *