Esercitazione 6

3 dicembre 2015

Termine per la consegna dei lavori: **martedì 10 dicembre** ore **23.59**.

Istruzioni

I lavori dovranno essere salvati in una cartella che deve contenere tutto e solo ciò che volete venga consegnato e valutato, d'ora in poi sarà **obbligatoriamente** un file **eseguibile** con estensione `.py` per ognuno degli esercizi. Solo per questa esercitazione, la consegna può essere effettuata su un unico file `.py`. Controllate che l'esecuzione del comando:

```
python <nome_file>.py
```

per ognuno degli esercizi produca l'output desiderato.
Per consegnare gli elaborati dovete raggiungere la cartella contenente i file da inviare in modalità terminale (`cd path_della_cartella`) e quindi eseguire il comando:

```
consegna consegna6
```

verrà visualizzata la lista di tutto ciò che è stato inviato.
Consegne successive (entro il termine per la consegna) sovrascriveranno le precedenti, verrà valutata solo l'ultima consegna sottomessa.
È obbligatorio che all'interno di ogni file sia riportato il vostro nome, cognome e numero di matricola (riportati all'interno di una riga commento all'inizio del file, es: `#Mario Rossi 1234567`).

ATTENZIONE!

Gli unici moduli importabili ammessi in questa esercitazione sono i moduli `math` e `random`. Esercizi risolti utilizzando altri moduli importati riceveranno il punteggio minimo. Python contiene delle **built-in functions**, che potete utilizzare e la cui lista si può trovare a questo indirizzo: `https://docs.python.org/2/library/functions.html`.

Esercizio 1

Creare la classe `Point3D` che rappresenta un punto in uno spazio euclideo tridimensionale. In particolare la classe deve contenere:

1. un costruttore (`__init__`) con tre parametri (x, y, z) che come valori di default valgono 0;
2. la definizione del metodo `distance(self, point)` che restituisce la distanza del punto da `point`;
3. la definizione del metodo speciale `__repr__(self)` che ritorna una stringa di rappresentazione per il punto, ad esempio `"Point3D(x,y,z)"` dove x, y, z sono le coordinate del punto;
4. la definizione dei metodi speciali `__eq__(self, point)`, `__lt__(self, point)` e `__gt__(self, point)`, i quali dato il punto `point` restituiscono un booleano che rappresenta rispettivamente, se i due punti sono uguali, se il punto è inferiore o maggiore di `point`. Un punto p_1 è maggiore di un punto p_2 se la distanza di p_1 dall'origine è maggiore di quella di p_2.

Per ognuno di questi metodi dare almeno un esempio di invocazione.
Nota: i metodi `__eq__`, `__lt__` e `__gt__` vengono invocati si effettuano confronti, rispettivamente, con gli operatori ==, < e >. Mentre l'operatore `__repr__` è invocato quando un oggetto di tipo `Point3D` viene stampato con `print`.

Esercizio 2

Creare una classe `Sphere3D` che rappresenta una sfera in uno spazio euclideo tridimensionale. In particolare la classe deve contenere:

1. un costruttore (`__init__`) con due parametri `(center, radius)` dove `center` è un `Point3D` che rappresenta il centro, con valore di default l'origine, e `radius` è il valore del raggio, con valore di default 1. All'interno del costruttore calcolare anche superficie e volume della sfera;
2. la definizione del metodo speciale `__repr__(self)`, che funziona analogamente al corrispondente metodo della classe `Point3D`;
3. la definizione dei metodi speciali `__eq__(self, sphere)`, `__lt__(self, sphere)` e `__gt__(self, sphere)`, i quali data la sfera `sphere` restituiscono un booleano che rappresenta rispettivamente se, le due sfere sono uguali, se la sfera ha un volume minore o maggiore di `sphere`;
4. la definizione del metodo `contains(self, point)` il quale ritorna un booleano che indica se il punto `point` è contenuto nella sfera;
5. la definizione del metodo `intersect(self, sphere)` il quale ritorna un booleano che indica se la sfera è intersecata dalla sfera `sphere`.

Per ognuno di questi metodi dare almeno un esempio di invocazione.

Esercizio 3

Definire un main, che sfrutta le classi `Point3D` e `Sphere3D`, il quale deve generare casualmente 20 sfere, con raggio compreso nell'intervallo reale $[1, 3]$ e con centro di coordinate intere $x, y, z \in [0, 10]$, e 40 punti con con coordinate intere $x, y, z \in [0, 10]$.
Una volta generate le sfere e i punti ricercare:

1. la sfera che contiene più punti: in caso di *ex aequo* restituire la sfera più piccola, e in caso si abbiano sfere con la stessa dimensione restituire la più vicina all'origine;

2. la sfera che interseca più sfere: in caso di *ex aequo* seguire le stesse istruzioni del punto precedente;

3. il punto contenuto in più sfere: in caso di *ex aequo* restituire il punto maggiore, secondo la definizione del metodo `__gt__`.