Esercitazione 7

16 dicembre 2014

Termine per la consegna dei lavori: **martedì 30 dicembre** ore **23.59** (due settimane di tempo).

Istruzioni

I lavori dovranno essere salvati in una cartella che deve contenere tutto e solo ciò che volete venga consegnato e valutato, dovrà essere **obbligatoriamente** un file **eseguibile** con estensione `.py` per **ognuno** degli esercizi. 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 consegna7
```

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

Definire la classe `Matrice` che rappresenta una matrice di dimensione $n \times m$. In particolare la classe deve contenere:

1. un costruttore (`__init__`) con un parametro `values` che come valore di default vale `[[0]]` (matrice nulla 1×1) e in generale è una lista di liste rappresentanti le righe della matrice, ad esempio deve essere possibile istanziare una matrice con l'istruzione `M = Matrice([[1,2,3],[4,5,6]]);`

2. la definizione del metodo speciale `__repr__(self)` che ritorna una rappresentazione per la matrice, ad esempio quella sottostante;

   ```
   1 2 3
   4 5 6
   ```

3. la definizione dei metodi speciali:

   - `__add__(self, M)` che, data la matrice `M`, restituisce `None` se le matrici non hanno la stessa dimensione, altrimenti la matrice somma delle due.
   - `__sub__(self, M)` che, data la matrice `M`, restituisce `None` se le matrici non hanno la stessa dimensione, altrimenti la matrice differenza delle due.
   - `__mul__(self, M)` che, data la matrice `M`, restituisce `None` se la seconda matrice ha un numero di righe diverso dal numero di colonne della prima, altrimenti la matrice prodotto delle due.
   - `__pow__(self, e)` che, dato un naturale `e`, restituisce `None` se la matrice non sia quadrata, altrimenti la matrice potenza secondo l'esponente `e`.

Per ognuno di questi metodi dare almeno un esempio di invocazione. È possibile eseguire più operazioni nella stessa istruzione?
Nota: i metodi `__add__`, `__sub__`, `__mul__` e `__pow__` vengono invocati, rispettivamente, con gli operatori `+`, `-`, `*` e `**`. Mentre l'operatore `__repr__` è invocato quando un oggetto di tipo `Matrice` viene stampato con `print`.

Esercizio 2

Definire una classe `Polinomio` che rappresenta un polinomio di grado n

$$c_n x^n + c_{n-1} x^{n-1} + ... + c_1 x + c_0.$$

In particolare la classe deve contenere:

1. un costruttore (`__init__`) con un parametro `coeff` che come valore di default vale `[0]` (polinomio nullo) e in generale è la lista dei coefficienti del polinomio in ordine crescente `[c_0, c_1, ..., c_n]`, ad esempio deve essere possibile istanziare un polinomio con l'istruzione `p = Polinomio([4, 4, 1])`;

2. la definizione del metodo speciale `__repr__(self)` che ritorna una rappresentazione per il polinomio, ad esempio quella sottostante;

   ```
   x^2 + 4x + 4
   ```

3. un metodo `valuta(self, x)` che, dato un reale `x`, restituisce il valore del polinomio valutato in `x`.

4. la definizione dei metodi speciali:

   - `__add__(self, p)` che, dato il polinomio `p`, restituisce il polinomio somma dei due.
   - `__sub__(self, p)` che, dato il polinomio `p`, restituisce il polinomio differenza dei due.
   - `__mul__(self, p)` che, dato il polinomio `p`, restituisce il polinomio prodotto dei due.
   - `diviso(self, p)` che, dato il polinomio `p`, stampi i polinomi quoziente e resto della divisione per `p`.

Per ognuno di questi metodi dare almeno un esempio di invocazione.
Nota: i metodi `__add__`, `__sub__` e `__mul__` vengono invocati, rispettivamente, con gli operatori `+`, `-` e `*`. Mentre l'operatore `__repr__` è invocato quando un oggetto di tipo `Polinomio` viene stampato con `print`.