

Esercitazione 5

06 dicembre 2016

Termine per la consegna dei lavori: **martedì 13 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 **eseguitabile** 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 consegna5
```

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: Reverse and add

Scrivere la funzione iterativa `reverse_and_add(n)` dove $n \geq 0$ intero, che ritorna il numero di passi necessari (**fissando il massimo a 10000**) per trasformare il numero n in un palindromo secondo questo semplice processo iterativo: se n è palindromo allora termina, altrimenti $n = n + \text{inverse}(n)$, dove *inverse* inverte le cifre del numero, ad esempio: $\text{inverse}(12345) = 54321$.

Esempio: partendo dal numero 59 si avrà:

$59 \rightarrow 59 + 95 = 154 \rightarrow 154 + 451 = 506 \rightarrow 506 + 605 = 1111!$, con 3 passi.

Creare le funzioni di supporto iterative `inverse(n)` e `is_palindrom(n)` dove n potete gestirlo come stringa o intero. Creare poi le stesse funzioni, compreso `reverse_and_add(n)`, ma in maniera ricorsiva!

Per entrambe le versioni stampare le coppie $(n, \#passi)$ per tutti gli interi $j \leq 1000$.

Attenzione ai seguenti valori: **196**, 295, 394, 493, 592, 689, 691, 788, 790, 879, 887, 978, 986.

Esercizio 2: Tartaglia

Scrivere la funzione ricorsiva `triangle(n)` che stampi le prime n righe (una sotto l'altra) del triangolo di Tartaglia.

Esercizio 3: Combinazioni

Creare la funzione ricorsiva `choose(n, k)` che calcoli il numero di combinazioni di k oggetti estratti da un insieme di n oggetti, che è definito dal coefficiente binomiale $\binom{n}{k}$. Creare poi la funzione ricorsiva `comb(lst, k)` che a partire dalla lista di oggetti `lst` crea tutte le combinazioni di lunghezza k e le ritorna come lista di liste. Verificare che la lunghezza della lista ritornata r sia effettivamente uguale a `choose(len(r), k)`.

Esercizio 4: Sezione Aurea e la sequenza di Lucas

Creare una funzione ricorsiva `golden(n)` che calcola il numero aureo ϕ secondo la definizione ricorsiva

$$\phi = 1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \dots}}}$$

dove n è la profondità massima che volete raggiungere nella ricorsione.

Creare poi una funzione iterativa `lucas(n)` e una ricorsiva `lucas_ric(n)` che calcola l' n -esimo numero di Lucas, dove la sequenza di Lucas è definita come $L_i = L_{i-2} + L_{i-1}$ dove $i \geq 0$ e $L_0 = 2, L_1 = 1$.

Creare una funzione `check(n)` (che usa `lucas(n)`) dove n è il valore massimo che volete controllare che esegue il seguente controllo:

$$\forall 2 \leq p \leq n : \text{round}(\phi^p) = L_p,$$

dove *round* è la funzione che arrotonda all'intero più vicino. Succede qualcosa di strano per valori di n maggiori di 65?

Creare infine la funzione `check_ric(n)` che è uguale a `check(n)` ma che usa `lucas_ric(n)`. Notate delle differenze dalla precedente?

Esercizio 5: Il gioco del 15

Link di spiegazione del gioco: http://it.wikipedia.org/wiki/Gioco_del_quindici

Tramite l'utilizzo di funzioni il programma deve:

1. Mostrare la tabellina di gioco disordinata;
2. Chiedere all'utente quale mossa vuole effettuare;
3. Verificare se effettivamente la mossa sia corretta e nel caso affermativo modificare la tabella di gioco, altrimenti ripetere la richiesta;
4. Verificare se si è raggiunto lo stato finale (tab. ordinata), altrimenti ripetere tutti i passi dal punto 1.

Consiglio: per “disordinare” in modo corretto (quindi risolvibile) la tabella iniziale del gioco del 15, si possono effettuare delle mosse scelte in modo casuale ed effettuarle solo se sono mosse corrette.