Indovina il numero

Un gioco in Python

Di che tipo di gioco si tratta?

Il programma estrae un numero a caso tra 1 e 50 e chiede al giocatore di indovinarlo. Per ogni tentativo inserito il programma indica se il numero è troppo alto o troppo basso, oppure se il giocatore ha indovinato.

Cos'è "Python"?

Python è un linguaggio di programmazione che è stato creato agli inizi degli anni 90 da Guido van Rossum al Centro di Matematica di Stichting nei Paesi Bassi. Da allora il linguaggio si è evoluto ed è stato utilizzato in molti ambiti, dalla matematica alla programmazione dei giochi.

Uno dei vantaggi di Python è il fatto che lo stesso programma può essere eseguito su diversi tipi di Sistemi Operativi: Linux, Windows, MacOS, e ora anche iOS e Android. Basta avere installato l'**interprete** che esegue il programma.

Inteprete

E' come l'interprete di una lingua straniera: capisce il linguaggio Python ed esegue le istruzioni del tuo programma.

Per avviarlo basta scrivere:

```
python
```

alla console. Prova.

Se non sai come aprire una console (chiamata anche "terminale") fatti aiutare da un mentor.

Dovrebbe comparire qualcosa di simile a questo:

```
Python 2.7.6 (default, Jun 22 2015, 17:58:13)
[GCC 4.8.2] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Dopo i tre segni di ">" l'inteprete rimane in attesa di istruzioni. Prova a scrivere

```
1 + 1
```

e a premere Invio. La risposta dovrebbe essere "2". Se è così siamo a buon punto: sa fare i conti.

Quella che hai appena sperimentato è la cosiddetta "modalità interattiva", quando l'interprete rimane in attesa ed esegue immediatamente le istruzioni che inserisci. E' molto comoda per fare esperimenti e per chiedere aiuto. Prova con (ci servirà dopo):

```
help ( raw_input )
```

Compare a video la spiegazione della funzione "raw_input". Per farla sparire devi premere il tasto "q".

Ovviamente i programmi enormi non vengono inseriti ed eseguiti tramite la modalità interattiva. Spesso questi programmi sono divisi in più file diversi collegati tra loro, ed esiste un file principale da cui parte tutto e che viene passato all'interprete per essere eseguito:

```
python nome_del_file_da_eseguire.py
```

Nota che di solito il nome di un file che contiene un programma Python finisce con l'estensione ".py"

Il gioco

Passiamo ora al nostro gioco.

Per ogni riga di codice del programma questo tutorial contiene le istruzioni per scriverla e una slide separata con maggiori informazioni riguardanti il linguaggio. Se hai fretta puoi saltare questa seconda slide a una prima lettura, ma è molto importante leggerla almeno una volta dopo aver completato il gioco perché fornisce informazioni essenziali per capire com'è fatto Python e per imparare a programmare.

Intestazione di un file Python

```
#!/usr/bin/env python
# -*- coding: utf8 -*-
```

Per ora considera queste righe come una specie di "magia" (ricordati: la spiegazione lunga è nella slide sotto).

La prima riga indica al computer che questo è un file contenente delle istruzioni che devono essere passate all'interprete Python. Se le passa a qualcun altro o se le apre con, per esempio, LibreOffice, non ottiene l'effetto desiderato.

La seconda riga indica all'interprete Python come sono scritti i caratteri all'interno del programma. Eh, sì! Purtroppo agli occhi di un computer i caratteri, soprattutto quelli accentati, non compaiono sempre uguali, anche se a te lo sembrano. Lì c'è scritto che per scriverli utilizziamo la codifica più generale possibile, ovvero l'UTF8.

Importazione di una funzione dalla libreria

```
from random import randint
```

Dal modulo "random" importiamo la funzione "randint" che serve per generare un numero casuale. Sarà il numero che l'utente dovrà indovinare.

Il linguaggio Python porta con sé, oltre all'interprete, anche una cosiddetta "libreria standard" contenente un sacco di funzioni interessanti per fare qualunque cosa: matematica, leggere e scrivere file, scaricare file da Internet, ecc. Queste funzioni

sono raggruppate in "moduli" per argomento. Per esempio il modulo "random" contiene varie funzioni che si occupano di generare numeri casuali.

L'istruzione "import" ha anche altre forme oltre a quella che hai utilizzato. Per esempio:

```
import random  # importa tutto il modulo
from random import *  # importa tutte le funzioni del modulo
```

Estrazione di un numero casuale

```
estratto = randint ( 1, 50 )
```

Con la funzione "randint" estraiamo un numero casuale tra 1 e 50 e lo copiamo nella variabile "estratto".

- le funzioni si eseguono utilizzando le parentesi tonde;
- tra le parentesi tonde si mettono, separati da virgola, i parametri della funzione;
- le variabili si assegnano con il simbolo "=".

Come faccio a sapere quali sono i parametri della funzione "randint" (la sua "sintassi")? Leggo la documentazione su Internet (http://www.python.org) oppure uso la modalità interattiva (che, però, a volte è un po' troppo sintetica):

```
>>> from random import randint
>>> help(randint)
```

Richiesta di input all'utente

```
numero = input ( "Indovina il numero: " )
```

Con la funzione "input" si chiede all'utente di scrivere qualcosa. Il testo passato come parametro alla funzione compare a video e l'inteprete rimane in attesa. Quello che viene digitato dall'utente viene restituito dalla funzione e noi lo memorizziamo nella variabile "numero".

Una serie di caratteri in un programma, per esempio il testo racchiuso tra virgolette passato alla funzione "input", prende il nome di "stringa di caratteri" o semplicemente "stringa".

Python permette di racchiudere le stringhe di caratteri sia con le virgolette doppie che con le virgolette singole.

Ciclo fino a quando la condizione è vera

```
while numero != estratto:
```

L'istruzione "while" ripete tutto quello che aggiungerai sotto fino a quando la condizione sarà vera. La condizione dice che il numero inserito dall'utente è diverso dal numero estratto dal programma. Quindi fino a quando l'utente non indovinerà il programma ripeterà le istruzioni che andrai ad aggiungere nel ciclo.

- "while" è un'istruzione che inizia un ciclo ("loop" in inglese). C'è anche "for";
- l'operatore != significa "diverso". Altri operatori sono: +, -, *, **, /, //, %, <<, >>, &, |, ^, ~, <, >, <=, >=, ==, !=, <>

Attenzione: l'istruzione "while", come molte altre in Python, finisce con un due punti ":" perché introduce un blocco di istruzioni.

In python i rientri sono importanti

A differenza degli altri linguaggi, in Python i blocchi di codice non sono delimitati da parentesi (graffe o quadre), ma sono riconoscibili dal fatto che le linee di codice rientrano di qualche spazio (o tabulazione) rispetto alle precedenti.

Così, dopo aver aperto il ciclo con l'istruzione "while", per indicare a Python quali sono le istruzioni da ripetere nel ciclo occorre indentare (far rientrare) le righe di codice:

```
while (condizione):
        riga 1
        riga 2
        ...

riga fuori dal ciclo
```

All'interno del ciclo cosa dobbiamo fare?

- Controllare **se** il numero inserito è minore di quello estratto e scriverlo a video;
- controllare **se** il numero inserito è maggiore di quello estratto e scriverlo a video;
- chiedere all'utente un nuovo numero.

Controllo se una condizione è vera

```
if numero < estratto:
        print "Il tuo numero è minore."
```

L'istruzione "if" controlla se la condizione è vera e, in tal caso, esegue il blocco di codice sotto (che deve essere rigorosamente rientrato di una tabulazione).

L'istruzione "print" stampa un valore (stringa, numero, ecc.) sul terminale.

La seconda condizione è simile alla prima ma la disuguaglianza è al contrario:

```
if numero > estratto:
        print "Il tuo numero è maggiore."
```

Ricordati che queste istruzioni sono all'interno del ciclo "while".

Nuova richiesta di input

Per chiudere il ciclo, visto che a quanto pare l'utente non ha indovinato, gli chiedo di inserire un altro numero:

```
numero = input ( "Indovina il numero: " )
```

Dopodiché il ciclo "while" riparte controllando se la sua condizione (numero != estratto) è vera, e così via...

Faccio i complimenti all'utente ed esco

Se la condizione del ciclo "while" è falsa il ciclo finisce e nel nostro caso significa che l'utente ha indovinato il numero, così il programma lo scrive a video ed esce.

```
print "Bravo! Hai indovinato!"
```

Il programma completo

```
#!/usr/bin/env python
# -*- coding: utf8 -*-

from random import randint

estratto = randint ( 1, 50 )

numero = input ( "Indovina il numero: " )

while numero != estratto:
	if numero < estratto:
		print "Il tuo numero è minore."

	if numero > estratto:
		print "Il tuo numero è maggiore."

	numero = input ( "Indovina il numero: " )

print "Bravo! Hai indovinato!"
```

Possibili estensioni

- Chiedere all'utente come si chiama e salutarlo prima dell'inizio del gioco. Suggerimento: per unire due stringhe si usa l'operatore "+".
- Il programma genera un numero casuale da 1 a 50 fisso, ma si potrebbe chiedere all'utente quale deve essere il numero maggiore da indovinare invece di 50.
- Quando l'utente indovina il numero stampare, assieme alle congratulazioni, il numero di tentativi effettuati per indovinarlo. Suggerimento: per incrementare di 1 il valore di una variabile si riassegna la variabile così a = a + 1.

E poi?

Il tuo salvagente è sempre la documentazione ufficiale https://docs.python.org/2.7, soprattutto il Tutorial e la Library Reference.

Del tutorial esiste anche una versione italiana che, anche se vecchiotta, va benissimo per imparare il linguaggio: http://docs.python.it/html/tut/tut.html