Esercitazione 7 - BIS

30 dicembre 2017

Termine per la consegna dei lavori: **martedì 9 gennaio 2018** ore **23.55**.

Istruzioni

Ogni esercizio dovrà esser salvato in un file con estensione `.py` e consegnato tramite Moodle su sito `https://elearning.unipd.it/math`, sezione "**Matematica: Laurea Triennale**" corso di "**Laboratorio di Programmazione**" cliccando su "**Esecitazione7b**".
È obbligatorio che all'interno di ogni file sia riportato il vostro nome, cognome e numero di matricola (potete riportarli all'interno di una riga commento all'inizio del file, es: `#Mario Rossi 1234567`).
Controllate che l'esecuzione del comando:

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

per ognuno degli esercizi produca l'output desiderato.

ATTENZIONE!

Per questa consegna non è previsto l'utilizzo di alcun modulo aggiuntivo. Inoltre, come sarà chiaro dalla consegna, **non è possibile usare nessuna funzionalità della classe string di python**!

Esercizio 1

Creare la classe `Stringa` che rappresenta una qualsiasi stringa. Ognuna delle funzionalità richieste **non** dovrà utilizzare nessuna utility della classe string offerta da python. Potete ovviamente avvalervi dell'utilizzo di liste e singoli caratteri per "simulare" le stringhe. In particolare la classe deve contenere i seguenti metodi:

1. costruttore (`__init__`) con un parametro `stringa` che come valore di default vale `[]` che rappresenta una stringa vuota;
2. `is_digit(self)` che ritorna vero se la stringa rappresenta un intero;
3. `is_float(self)` che ritorna vero se la stringa rappresenta un float (numero con virgola) ma non è un intero;
4. `to_lower(self)` ritorna una nuova stringa uguale alla corrente avente tutti i caratteri alfabetici minuscoli;
5. `to_upper(self)` ritorna una nuova stringa uguale alla corrente avente tutti i caratteri alfabetici maiuscoli;
6. `replace(self, s1, s2)` che ritorna una nuova stringa uguale alla corrente nella quale tutte le occorrenze di `s1` sono sostituite con `s2`. Notare che `s1` e `s2` si assumono di tipo `Stringa`;
7. `__len__(self)` ritorna la lunghezza della stringa;
8. `__eq__(self, stringa)` ritorna vero se la stringa corrente e `stringa` sono uguali. `stringa` si assume di tipo `Stringa`;
9. `count(self, stringa)` che ritorna quante volte `stringa` è contenuta nella stringa corrente. `stringa` si assume di tipo `Stringa`;
10. `split(self, delimiter)` che ritorna una lista di stringhe che rappresentano le parti della stringa corrente divisa ad ogni occorrenza di `delimiter`. `delimiter` si assume di tipo `Stringa`;
11. `join(self, seq)` che ritorna un'unica stringa che rappresenta la concatenazione delle stringhe contenute in `seq` intervallate dalla stringa corrente. Sostanzialmente `join` esegue l'operazione opposta di `split`;
12. `substring(self, start, end)` ritorna una sottostringa della stringa corrente delimitata dagli indici `start` e `end` (escluso);
13. `center(self, width)` crea una nuova stringa che rappresenta la centratura della stringa corrente in uno spazio contenente `width` caratteri. La centratura viene eseguita aggiungendo spazi a destra e a sinistra. In caso si debba aggiungere un numero di spazi dispari aggiungere uno spazio in più a sinistra della stringa;
14. `strip(self)` ritorna una nuova stringa uguale alla corrente nella quale sono stati rimossi tutti gli spazi in testa e in coda;

15. `find(self, stringa, start=None, end=None)` ritorna l'indice della prima occorrenza della stringa `stringa` nella stringa corrente o in una sua sottostringa delimitata dagli indici `start` e `end`. In caso non siano specificati (=None) si assume `start` = 0 e `end` la lunghezza della stringa.

16. `__repr__(self)` che ritorna la rappresentazione in tipo *string* di python. In questo **unico** metodo, per ovvi motivi, è consentito usare le stringhe di python.

Per ognuno dei metodi dare almeno un esempio di utilizzo non banale.