Progetto “Notazione Polacca Inversa”

Appello del 10 febbario 2022

Descrizione del progetto

Lo scopo del progetto è quello di realizzare un programma che, data una sequenza di stringhe e numeri, implementi una valutazione della sequenza come delle istruzioni in notazione polacca inversa.

Partendo dall'esempio `[4, 3, '+', 5 '*']` le operazioni svolte su uno stack `S` inizialmente vuoto sono le seguenti:

- `4` viene inserito in cima a `S`
- `3` viene inserito in cima a `S`
- Dato che `+` è una operazioni che richiede due argomenti, questi vengo estratti dalla cima dello stack (quindi `3` e `4` sono estratti), viene calcolato `3 + 4` e il risultato (`7`) viene inserito in cima allo stack
- `5` viene inserito in cima allo stack
- Dato che `*` è una operazione che richiede due argomenti, questi vengono estratti dalla cima dello stack (quindi `5` e `7` sono estratti) e il risultato di `5 * 7`, ovvero `35` viene messo in cima allo stack.

In generale l'interpretazione di una sequenza di simboli in notazione polacca inversa è fatta come segue: - Se il simbolo è un numero allora viene messo in cima allo stack - Se il simbolo rappresenta una operazione allora gli argomenti sono rimossi dalla cima dello stack, l'operazione viene eseguita e il risultato inserito in cima allo stack.

Si richiede di creare una classe `InterpreteStack` che abbia i seguenti metodi:

- Creazione di un oggetto di tipo `InterpreteStack` senza argomenti.
- Un metodo `interpreta` che prende come argomento una lista di numeri e stringhe e ritorna una lista contenente il contenuto dello stack al termine dell'interpretazione della lista fornita come argomento. La cima dello stack deve essere l'ultimo elemento della lista.

Si richiede che il sistema supporti numeri interi e le seguenti operazioni:

- `+`, somma di due numeri - `*`, prodotto di due numeri
- `dup`, copia il valore in cima allo stack
- `drop`, rimuove il valore in cima allo stack
- `print`, stampa (rimuovendolo) il valore in cima allo stack

Codice

Viene fornito uno scheletro del codice che deve essere implementato, con i metodi che devono essere scritti:

```
class InterpreteStack:

    def __init__(self):
        #...

    def interpreta(self, sequenza);
        #...
```

Nel progetto è consentito avere funzioni aggiuntive che testano il buon funzionamento delle funzionalità richieste. È anche consentito avere metodi aggiuntivi.

Si ricorda di commentare adeguatamente il codice, approfittandone per spiegare le scelte implementative effettuate.

Esempi d'uso

Il seguente frammento di codice chiama tutti i metodi la cui implementazione è richiesta dal progetto. Come commento è indicato il valore atteso in una implementazione funzionante:

```
def test():
    vm = InterpreteStack()
    vm.interpreta([3, 4, '+']) # ritorna [7]
    vm.interpreta([5, 'dup', '*']) # ritorna [25]
    vm.interpreta([3, 2, '*', 1, 4, '+', '+']) # ritorna [11]
    vm.interpreta([4, 5, 5, '+', 'print']) # ritorna [4] e stampa 10
    vm.interpreta([4, 5, 6, '+', 'dup']) # ritorna [4, 11, 11]
    vm.interpreta([5, 6, 7, 'drop', 'print', 'drop']) # ritorna [] e stampa 6
```

Indicazioni

Il progetto deve essere svolto **individualmente**. La consegna dovrà avvenire entro le ore 23:59 del giorno 04/02/2021 secondo le seguenti modalità:

- Invio di una email a `lmanzoni@units.it` dal vostro account email istituzionale con oggetto *[informatica] consegna progetto appello del 10/02/2022*.
- L'email deve avere come allegato il progetto in un singolo file in codice sorgente Python versione 3, dal nome `Nome_Cognome_matricola.py`, quindi, per esempio Mario Rossi di matricola 12345 consegnerà un file dal nome `Mario_Rossi_12345.py`.
- Il file deve contenere sotto forma di commento le seguenti linee indicanti nome, cognome e numero di matricola: `python # Nome: Mario # Cognome: Rossi # Matricola: 12345`