Μεταγλωττιστές 2024
Προγραμματιστική Εργασία #2

Για την επίλυση της εργασίας θα ξεκινήσετε έχοντας ως βάση το εξής αρχείο Python:
https://mixstef.github.io/courses/compilers/calculator.py

Το αρχείο αυτό υλοποιεί ένα απλό calculator για τις 4 βασικές πράξεις (+, -, *, /) με τη μέθοδο της συντακτικής ανάλυσης αναδρομικής κατάβασης (recursive descent parsing). Η εκτέλεση των πράξεων γίνεται απευθείας κατά τη διάρκεια της συντακτικής ανάλυσης (χωρίς δημιουργία AST).

Η γραμματική που έχει χρησιμοποιηθεί είναι η ακόλουθη:

```
Expr → Term (Addop Term)*
Term → Factor (Multop Factor)*
Factor → (Expr) | number
Addop → + | -
Multop → * | /
```

Ο συντακτικός αναλυτής/interpreter συνοδεύεται από κώδικα ελέγχου με διάφορα tests που θα πρέπει να εκτελούνται επιτυχώς όταν ολοκληρώσετε το ζητούμενο της άσκησης. Δείτε τις οδηγίες για τον έλεγχο στο τέλος του κειμένου.

Ζητούμενο

Στον ήδη υπάρχοντα κώδικα του calculator θα πρέπει να προστεθούν οι τελεστές α) **μοναδιαίο μείον** (-x), β) **modulo** (x % y) και γ) **ύψωση σε δύναμη** (x ** y). Οι προτεραιότητες των τελεστών ορίζονται ως εξής (από την υψηλότερη προς τη χαμηλότερη):

Τελεστής	Σχόλιο	Προσεταιριστικότητα
**	ύψωση σε δύναμη	δεξιά
-	μοναδιαίο μείον (-x)	δεξιά
* / %		αριστερή
+ -		αριστερή

α) **Προσθέστε τους κατάλληλους κανόνες στην υπάρχουσα γραμματική** για να υλοποιήσετε τη συντακτική ανάλυση των νέων τελεστών.

Υποδείξεις:

1. Θυμηθείτε ότι όσο υψηλότερη είναι η προτεραιότητα ενός τελεστή, τόσο χαμηλότερα θα εμφανίζεται στους κανόνες της γραμματικής.

2. Όταν η προσεταιριστικότητα ενός τελεστή είναι δεξιά (δηλ. από δεξιά προς τα αριστερά), τότε η παραδοσιακή recursive descent συντακτική ανάλυση αρκεί για την υλοποίηση των κανόνων του τελεστή (δεν απαιτούνται custom τροποποιήσεις).

3. Το modulo είναι στο ίδιο επίπεδο με τα * και / και έτσι μπορείτε να το χειριστείτε μαζί με αυτά, προσθέτοντας π.χ. το % στο Multop.

β) **Ελέγξτε τις ιδιότητες της νέας γραμματικής σας** και βρείτε τα FIRST/FOLLOW sets.

Υπόδειξη:

Στο online εργαλείο http://smlweb.cpsc.ucalgary.ca/start.html ξεκινήστε εισάγοντας τη γραμματική:

```
Expr -> Term Term_tail .
Term_tail -> Addop Term Term_tail | .
Term -> Factor Factor_tail .
Factor_tail -> Multop Factor Factor_tail | .
Factor -> lpar Expr rpar | num .
Addop -> plus | minus .
Multop -> mult | div .
```

Στη συνέχεια προσθέστε τους νέους κανόνες στη σύνταξη που απαιτεί το εργαλείο.

γ) **Προσθέστε στον αρχικό κώδικα του calculator την υλοποίηση των νέων κανόνων** σύμφωνα με τα (α) και (β). Οι μέθοδοι των μη τερματικών συμβόλων θα πρέπει να εκτελούν και τις αντίστοιχες πράξεις και να επιστρέφουν το αποτέλεσμα (μελετήστε πώς γίνεται αυτό στον αρχικό κώδικα).

Υποδείξεις:

1. Μην ξεχάσετε να ορίσετε τα νέα tokens στον λεκτικό αναλυτή (tokenizer).

2. Η ύψωση σε δύναμη (x ** y) ενδεχομένως να επιστρέψει μιγαδικές τιμές για κάποια x και y. Επειδή ο calculator έχει ένα μόνο είδος τιμών (float) θα πρέπει να εντοπίζονται παρόμοιες καταστάσεις και να παράγεται το αντίστοιχο CalculatorError. Ο έλεγχος μπορεί να γίνει μετατρέποντας το αποτέλεσμα σε float μέσα σε ένα try...except statement:

```
try:
    val = float(q**qt)
except:
    raise CalculatorError(...)
```

<u>Διαδικασία ελέγχου</u>

Όταν εκτελέσετε το αρχείο, διενεργούνται οι έλεγχοι που περιλαμβάνονται ως παραδείγματα στο docstring της συνάρτησης tester(). Στην αρχική μορφή του κώδικα θα επιτύχουν 4 από τα 9 tests και στην έξοδο θα δείτε, μαζί με την περιγραφή των αποτυχημένων tests (τι αναμενόταν και τι προέκυψε), την εξής σύνοψη:

```
1 items had failures:
   5 of   9 in __main__.tester
9 tests in 2 items.
4 passed and 5 failed.
***Test Failed*** 5 failures.
```

Καθώς προσθέτετε τις ζητούμενες αλλαγές τα σφάλματα των tests θα μειώνονται. Ο στόχος είναι να επιτυγχάνουν όλα τα tests και να τυπώνεται στην έξοδο το εξής:

```
1 items passed all tests:
   9 tests in __main__.tester
9 tests in 2 items.
9 passed and 0 failed.
```

```
Test passed.
```

Υπόδειξη:

Μπορείτε να αναπτύξετε τον κώδικά σας ως ανεξάρτητο αρχείο .py (για εκτέλεση από το command line) είτε ως jupyter notebook. Στη δεύτερη περίπτωση, εάν το αποτέλεσμα του ελέγχου δεν εμφανίζεται μετά την εκτέλεση, τοποθετήστε τις 2 τελευταίες γραμμές του προγράμματος σε επόμενο κελί που θα εκτελέσετε στη συνέχεια:

```
import doctest
doctest.testmod(verbose=True,exclude_empty=True)
```

<u>Παραδοτέο</u>

Όταν έχετε ολοκληρώσει την επίλυση της εργασίας, περάστε τον τελικό κώδικα σε ένα jupyter notebook (αν δεν χρησιμοποιείτε notebook από την αρχή). Εκτελέστε τον συνολικό κώδικα. **Τα αποτελέσματα των tests θα πρέπει να εκτυπωθούν και να φαίνονται στο τελικό jupyter notebook που θα παραδώσετε**.

Ανεβάστε το τελικό σας notebook με την εκτύπωση των αποτελεσμάτων των tests στο opencourses (**Εργασία 2**) έως και τη **Δευτέρα 3/6**.

(Ίσως χρειαστεί να κάνετε zip το αρχείο ipynb για να γίνει αποδεκτό από το σύστημα)