Μεταγλωττιστές 2023-24

Λεκτική ανάλυση με κανονικές εκφράσεις
Παραδείγματα με την κλάση Tokenizer

Λεκτικός Αναλυτής (Scanner/Tokenizer)

- Μετατρέπει το κείμενο εισόδου σε σύμβολα
  - tokens, μαζί με το κείμενο που αντιστοιχεί (lexeme) και προαιρετικά με άλλη πληροφορία, π.χ. η θέση στο συνολικό κείμενο (αριθμός γραμμής/χαρακτήρα)
- Διαφέρει από άλλες χρήσεις των εργαλείων αναγνώρισης κειμένου, όπως
  - η αναζήτηση ταιριασμάτων μέσα σε ένα γενικότερο κείμενο,
  - η αναγνώριση εγκυρότητας εισόδου χρήστη, κλπ

Μέθοδοι χρήσης λεκτικού αναλυτή

- Μέθοδος 1: ο λεκτικός αναλυτής είναι το κυρίως πρόγραμμα. Σε ένα επαναληπτικό loop αναλύει την είσοδο σύμβολο προς σύμβολο και εκτελεί τις κατάλληλες λειτουργίες.
- Μέθοδος 2: ο λεκτικός αναλυτής αντιμετωπίζεται ως συνάρτηση, η οποία καλείται επαναληπτικά από το κυρίως πρόγραμμα. Επιστρέφει σε κάθε κλήση ένα σύμβολο που το χειρίζεται στη συνέχεια το κυρίως πρόγραμμα.

Απαιτήσεις από τον λεκτικό αναλυτή

- Θα πρέπει να αναγνωρίζει μια σειρά διαφορετικών tokens
  - π.χ. keywords, identifiers, literals, operators και λοιπά στοιχεία σύνταξης μιας γλώσσας προγραμματισμού
- Θα πρέπει να αναλύει (=αναγνωρίζει) όλο το κείμενο εισόδου (π.χ. πηγαίο κώδικα) σε επαναλαμβανόμενα βήματα
  - Και να παρέχει ένδειξη για το τι αναγνωρίστηκε κάθε φορά
- Θα πρέπει να επιλέγει το ταίριασμα με το μεγαλύτερο μήκος
  - π.χ. όλο το // (αρχή σχολίου) κι όχι μόνο το πρώτο / (διά)
- Σε περίπτωση που το ίδιο κείμενο μπορεί να αποδοθεί με δύο διαφορετικά tokens, θα πρέπει να παρέχεται ένας μηχανισμός επιλογής, με βάση την προτεραιότητα που θέτει ο χρήστης
  - π.χ. το int είναι όνομα μεταβλητής ή keyword; πώς θα το ορίσει ο χρήστης;

Η κλάση Tokenizer

- θα τη βρείτε στο module compilerlabs
  - Βεβαιωθείτε ότι έχετε την πιο πρόσφατη έκδοση
- Επιτρέπει την κατασκευή λεκτικού αναλυτή βασισμένου στις κανονικές εκφράσεις της Python
  - Μπορείτε να χρησιμοποιήσετε όλα τα στοιχεία κανονικών εκφράσεων που έχουμε συναντήσει
- Η υλοποίηση χρησιμοποιεί την εναλλαγή (|) για να συνδέσει patterns που αναγνωρίζουν tokens σε μια μεγάλη κανονική έκφραση
  - Προσοχή λοιπόν στις ιδιαιτερότητες της εναλλαγής!

Χρήση της κλάσης Tokenizer (σύνοψη)

- Εισαγωγή κλάσης και βοηθητικών αντικειμένων

```
from compilerlabs import Tokenizer,TokenAction,TokenizerError
```

- Δημιουργία tokenizer

```
t = Tokenizer()
```

- Προσθήκη pattern/token (μία φορά για κάθε pattern)

```
t.pattern('[a-z]+','word')
```

- Λήψη tokens

```
text = "abcd efgh"
for s in t.scan(text):
    print(s)
```

Η μέθοδος pattern()

- *pattern(regex, token, keywords=None)*
  - *regex* η κανονική έκφραση για το συγκεκριμένο pattern
  - *token* θα επιστραφεί όταν αναγνωριστεί το pattern
    - Ειδικές τιμές token:
      - TokenAction.IGNORE → το token δεν επιστρέφεται
      - TokenAction.TEXT → ό,τι αναγνωρίστηκε (lexeme) επιστρέφεται ως token
      - TokenAction.ERROR → δημιουργείται σφάλμα TokenizerError
  - *keywords* ακολουθία ή λεξικό τιμών. Αν αναγνωριστεί κάποια από τις τιμές της ακολουθίας ή τα κλειδιά του λεξικού δεν επιστρέφεται το token αλλά η τιμή ως έχει (για ακολουθία) ή η τιμή που αντιστοιχεί στο κλειδί (για λεξικό)

Προτεραιότητα ταιριάσματος

- Λόγω της χρήσης της εναλλαγής προσθέστε το μεγαλύτερο pattern πρώτα

```
t.pattern('//[^\n]*',TokenAction.IGNORE)

t.pattern('[-+*/]','OPERATOR')
```

- Αν υπάρχουν strings που πρέπει να τα χειριστείτε με ειδικό τρόπο χρησιμοποιήστε το όρισμα keywords

```
t.pattern('[a-z]+','word',('print','int','def'))
```

- Εναλλακτικά

```
t.pattern('[a-z]+','word',
{'print':'KW_PRINT','int':'KW_INT','def':'KW_DEF'})
```

Χειρισμός μη έγκυρης εισόδου

- Ως τελευταίο pattern προσθέστε

```
t.pattern('.',TokenAction.ERROR)
```

- Χειρισμός σφάλματος

```
try:
    for s in t.scan(text):
        print(s)
except TokenizerError as e:
    print(e)
```

Η μέθοδος scan()

- *scan(text, flags=0, eot=True)*
  - *text* το κείμενο (string) που θα αναλυθεί
  - *flags* (προαιρετικό όρισμα) οποιοσδήποτε συνδυασμός από flags του module re, αν υπάρχει θα χρησιμοποιηθεί στο ταίριασμα
  - *eot* (προαιρετικό όρισμα) εάν είναι αληθές ο tokenizer θα επιστρέψει ένα τελευταίο token ίσο με None για να σηματοδοτήσει το τέλος του κειμένου εισόδου (end-of-text, EOT).

Τι επιστρέφει η scan()

- *Symbol(token, lexeme, lineno, charpos)*
  - *token* αντιστοιχεί στο κείμενο που αναγνωρίστηκε
  - *lexeme* το κείμενο που αναγνωρίστηκε
  - *lineno, charpos* η θέση του κειμένου που αναγνωρίστηκε (γραμμή/χαρακτήρας) μέσα στο συνολικό text εισόδου