Λεξικά

Μιχάλης Δρακόπουλος

Μάιος 2020

ΛΕΞΙΚΑ

Είναι μια **μη διατεταγμένη** συλλογή αντικειμένων

Ιδιότητες λεξικών

- Η αποθήκευση γίνεται με **κλειδί** αντί θέσης: Η **τιμή (value)** κάθε αντικειμένου στο λεξικό σχετίζεται με ένα **κλειδί (key)**.

- Το κλειδί χρησιμοποιείται για την ανάκτηση του αποθηκευμένου αντικειμένου.
- Τα στοιχεία του λεξικού είναι ζεύγη **κλειδιών/τιμών**.

[ ]:
```
phone = {"John": '697-1234567', 'Mary': '694-4445566', 'Helen': '698-3141592'}
```

- Το όνομα του τύπου τους είναι `dict`

[ ]:
```
type(phone)
```

- Τα λεξικά είναι **απεικονίσεις (mappings)**: απεικονίζουν κλειδιά σε τιμές.

Επιλογή στοιχείου:

[3]:
```
phone['Helen']
```

[3]:
```
'698-3141592'
```

Η αναζήτηση τιμών ενός λεξικού είναι πιο γρήγορη απ' ότι η αναζήτηση με θέση, όπως π.χ στις λίστες:

[4]:
```
names = ['John', 'Mary', 'Helen']
number = ['697-1234567', '694-4445566', '698-3141592']
```

Το τηλέφωνο της Helen προκύπτει με σύνθετη αναζήτηση:

[5]:
```
number[names.index('Helen')]
```

[5]:
```
'698-3141592'
```

- Τα στοιχεία ενός λεξικού μπορεί να είναι διαφορετικού τύπου:

[6]:
```
student = {'name': 'Mary Pap', 'age': 20, 'mesos_oros': 8.2}
```

Το λεξικό student θα μπορούσε για παράδειγμα να είναι μια εγγραφή (record) σε μια βάση δεδομένων.

- Τα στοιχεία ενός λεξικού αποθηκεύονται με τη σειρά εισαγωγής τους (από Python 3.7 και μετά)

[7]:
```
student
```

[7]:
```
{'name': 'Mary Pap', 'age': 20, 'mesos_oros': 8.2}
```

Λόγω της μη-διάταξης των στοιχείων τους, τα λεξικά δεν υποστηρίζουν πράξεις **τεμαχισμού** και **συνένωσης**.

- Είναι **μεταβλητός** τύπος: οι **τιμές** των στοιχείων τους μπορούν να **αλλάξουν**

[8]:
```
student['mesos_oros'] = 9.5
student
```

[8]:
```
{'name': 'Mary Pap', 'age': 20, 'mesos_oros': 9.5}
```

- Είναι **μεταβλητού μήκους**: μπορούμε να **εισάγουμε** νέα στοιχεία

[9]:
```
student['telephone'] = phone['Mary']
student
```

[9]:
```
{'name': 'Mary Pap', 'age': 20, 'mesos_oros': 9.5, 'telephone': '694-4445566'}
```

(σε αντίθεση με τις λίστες, μπορούμε να εκχωρήσουμε τιμές σε κλειδιά που δεν υπάρχουν, *χωρίς να προκύψει σφάλμα*: απλά δημιουργείται νέο στοιχείο)

- Είναι **μεταβλητού μήκους**: μπορούμε να **διαγράψουμε** στοιχεία

[10]:
```
del student['age']
student
```

[10]:
```
{'name': 'Mary Pap', 'mesos_oros': 9.5, 'telephone': '694-4445566'}
```

- Τα **κλειδιά** λεξικού πρέπει να είναι **αμετάβλητου** τύπου, π.χ. `str`, `int`, `float`, `tuple`

[11]:
```
spring = {3: 'Mar', 4: 'Apr', 5: 'May'}
```

[12]:
```
for month in range(3,6):
    print('Spring month ', month-2, ' is ', spring[month], sep = '')
```

```
Spring month 1 is Mar
Spring month 2 is Apr
Spring month 3 is May
```

- Αντίθετα δεν υπάρχουν περιορισμοί για τις **τιμές**

Συναρτήσεις και τελεστές για λεξικά

[13]:
```
student
```

[13]:
```
{'name': 'Mary Pap', 'mesos_oros': 9.5, 'telephone': '694-4445566'}
```

- Μήκος: συνάρτηση `len`

[14]:
```
len(student)
```

[14]:
```
3
```

- Έλεγχος ισότητας `==`, `!=`

Δυο λεξικά είναι ίσα αν και μόνο αν έχουν ακριβώς τα ίδια ζεύγη κλειδιών/τιμών ανεξάρτητα από σειρά αποθήκευσης τους

[15]:
```
D1 = {'a': ord('a'), 'b': ord('b')}
D1
```

[15]:
```
{'a': 97, 'b': 98}
```

[16]:
```
D2 = {'b': 98, 'a': 97}
D2
```

[16]:
```
{'b': 98, 'a': 97}
```

[17]:
```
D1 == D2
```

[17]:
```
True
```

- Έλεγχος μέλους με χρήση του **κλειδιού**: τελεστές `in` και `not in`

[18]:
```
'name' in student
```

[18]:
```
True
```

[19]:
```
'name' not in student
```

[19]:
```
False
```

Δεν δουλεύει με **τιμές**

[20]:
```
'Mary Pap' in student
```

[20]:
```
False
```

Άλλοι τρόποι δημιουργίας λεξικών

- Από αλληλουχίες (λίστες, πλειάδες) με ζεύγη τιμών/κλειδιών

Λίστα με λίστες ή πλειάδες κλειδιών/τιμών

[21]:
```
movie = dict( [ ['title', 'The Life of Brian'], ('year', 1979) ] )
movie
```

[21]:
```
{'title': 'The Life of Brian', 'year': 1979}
```

Πλειάδα με λίστες ή πλειάδες κλειδιών/τιμών

[22]:
```
book = dict( ( ('title', 'Elements'), ['author', 'Euclid'] ) )
book
```

[22]:
```
{'title': 'Elements', 'author': 'Euclid'}
```

- Με λιγότερη πληκτρολόγηση, αν όλα τα κλειδιά είναι **συμβολοσειρές**

[23]:
```
textbook = dict(title = 'Starting out with Python', author = 'Tony Gaddis',␣
↪pages = 774)
textbook
```

[23]:
```
{'title': 'Starting out with Python', 'author': 'Tony Gaddis', 'pages': 774}
```

- Σταδιακά, ξεκινώντας από το **κενό** λεξικό

[24]:
```
squares = {}
for x in range(10, 15):
    squares[x] = x**2
squares
```

[24]:
```
{10: 100, 11: 121, 12: 144, 13: 169, 14: 196}
```

- Με τη μέθοδο `fromkeys`

Δημιουργεί / αρχικοποιεί λεξικό με κλειδιά από αντικείμενο που τηρεί το πρωτόκολλο επανάληψης, και με αρχικές τιμές ίσες μεταξύ τους

[25]:
```
D = dict.fromkeys(['name', 'job', 'age'])
D
```

[25]:
```
{'name': None, 'job': None, 'age': None}
```

[26]:
```
D = dict.fromkeys('abcde', 0)
D
```

[26]:
```
{'a': 0, 'b': 0, 'c': 0, 'd': 0, 'e': 0}
```

Μέθοδοι για λεξικά

[27]:
```
dir(dict)[-11:]
```

[27]:
```
['clear',
 'copy',
 'fromkeys',
 'get',
 'items',
 'keys',
 'pop',
 'popitem',
 'setdefault',
 'update',
 'values']
```

- Αντιγραφή `copy`, διαγραφή `clear`

[28]:
```
squares
```

[28]:
```
{10: 100, 11: 121, 12: 144, 13: 169, 14: 196}
```

[29]:
```
squares2 = squares.copy()
squares == squares2, squares2 is squares
```

[29]:
```
(True, False)
```

[30]:
```
squares.clear()
print(squares)
print(squares2)
```

```
{}
{10: 100, 11: 121, 12: 144, 13: 169, 14: 196}
```

- Κλειδιά `keys`, τιμές `values`, ζεύγη κλειδιών/τιμών `items`

[31]:
```
textbook
```

[31]:
```
{'title': 'Starting out with Python', 'author': 'Tony Gaddis', 'pages': 774}
```

[32]:
```
k = textbook.keys()
v = textbook.values()
both = textbook.items()
```

Επιστρέφουν *αντικείμενα* που έχουν ειδικούς τύπους, τις προβολές λεξικού (dictionary views):

[33]:
```
type(k), type(v), type(both)
```

[33]:
```
(dict_keys, dict_values, dict_items)
```

τα οποία δεν είναι άμεσα διαχειρίσιμα:

[34]:
```
k
```

[34]:
```
dict_keys(['title', 'author', 'pages'])
```

[35]:
```
v
```

[35]:
```
dict_values(['Starting out with Python', 'Tony Gaddis', 774])
```

[36]:
```
both
```

[36]:
```
dict_items([('title', 'Starting out with Python'), ('author', 'Tony Gaddis'),
('pages', 774)])
```

Τηρούν όμως το **πρωτόκολλο επανάληψης** και μπορούμε να τα διαχειριστούμε μετατρέποντάς τα σε άλλους τύπους, ή με εντολές επανάληψης.

[37]:
```
LK = list(k)
LK
```

[37]:
```
['title', 'author', 'pages']
```

[38]:
```
TV = tuple(v)
TV
```

[38]:
```
('Starting out with Python', 'Tony Gaddis', 774)
```

[39]:
```
LB = list(both)
LB
```

[39]:
```
[('title', 'Starting out with Python'),
 ('author', 'Tony Gaddis'),
 ('pages', 774)]
```

[40]:
```
for key in k:
    print(key.upper())
```

```
TITLE
AUTHOR
PAGES
```

- Τιμή από κλειδί get

[41]:
```
n = textbook.get('author')
n
```

[41]:
```
'Tony Gaddis'
```

Είναι το ίδιο με:

[42]:
```
m = textbook['author']
m
```

[42]:
```
'Tony Gaddis'
```

αλλά όταν το ζητούμενο κλειδί δεν υπάρχει στο λεξικό:

[43]:
```
textbook['publisher']
```

```
---------------------------------------------------------------------------

KeyError                                  Traceback (most recent call last)

<ipython-input-43-2e9b1057ab25> in <module>() ----> 1 textbook['publisher']


KeyError: 'publisher'
```

[44]:
```
p = textbook.get('Publisher')
print(p)
```

```
None
```

[45]:
```
p = textbook.get('Publisher', 'daVinci')
p
```

[45]:
```
'daVinci'
```

[46]:
```
textbook
```

[46]:
```
{'title': 'Starting out with Python', 'author': 'Tony Gaddis', 'pages': 774}
```

- Τιμή από κλειδί και αφαίρεση του στοιχείου από το λεξικό pop

[47]:
```
npages = textbook.pop('pages')
print(npages)
```

```
774
```

[48]:
```
textbook
```

[48]:
```
{'title': 'Starting out with Python', 'author': 'Tony Gaddis'}
```

- Πλειάδα με στοιχείο (κλειδί / τιμή) popitem

Αφαιρεί το στοιχείο που προστέθηκε τελευταία (Python >= 3.7) LIFO!!!

[49]:
```
student
```

[49]:
```
{'name': 'Mary Pap', 'mesos_oros': 9.5, 'telephone': '694-4445566'}
```

[50]:
```
a, b = student.popitem()
print('Removed element with key =', a, 'and value =', b)
```

```
Removed element with key = telephone and value = 694-4445566
```

[51]:
```
a, b = student.popitem()
print('Removed element with key =', a, 'and value =', b)
```

```
Removed element with key = mesos_oros and value = 9.5
```

[52]:
```
student
```

[52]:
```
{'name': 'Mary Pap'}
```

- Ενημέρωση λεξικού (συνένωση με άλλο λεξικό) update

[53]:
```
employee = {'name': 'Bob', 'salary': 1000, 'age': 28}
```

[54]:
```
employee
```

[54]:
```
{'name': 'Bob', 'salary': 1000, 'age': 28}
```

[55]:
```
info = {'company': 'ACME', 'job': 'Programmer'}
employee.update(info)
employee
```

[55]:
```
{'name': 'Bob',
 'salary': 1000,
 'age': 28,
 'company': 'ACME',
 'job': 'Programmer'}
```

Επανάληψη με λεξικά

[56]:
```
ages = dict(Bob = 20, Mary = 31, Ann = 20, John = 42)
ages
```

[56]:
```
{'Bob': 20, 'Mary': 31, 'Ann': 20, 'John': 42}
```

Επανάληψη με τα κλειδιά 1ος τρόπος

[57]:
```
for k in ages.keys():
    print(k, '->', ages[k])
```

```
Bob -> 20
Mary -> 31
Ann -> 20
John -> 42
```

2ος τρόπος (προτιμητέος)

[58]:
```
for k in ages:
    print(k, '->', ages[k])
```

```
Bob -> 20
Mary -> 31
Ann -> 20
John -> 42
```

Εφαρμογή Εμφάνιση στοιχείων λεξικού με αλφαβητική σειρά των κλειδιών του

[59]:
```
for k in sorted(ages):
    print(k, '->', ages[k])
```

```
Ann -> 20
Bob -> 20
John -> 42
Mary -> 31
```

Παράλληλη επανάληψη με κλειδιά και τιμές

[60]:
```
for (name, age) in ages.items():
    print(name, 'is', age, 'years old')
```

```
Bob is 20 years old
Mary is 31 years old
Ann is 20 years old
John is 42 years old
```

Λεξικά με στοιχεία αλληλουχίες, αλληλουχίες με στοιχεία λεξικά, λεξικά με στοιχεία λεξικά

[61]:
```
employee
```

[61]:
```
{'name': 'Bob',
 'salary': 1000,
 'age': 28,
 'company': 'ACME',
 'job': 'Programmer'}
```

[62]:
```
employee['job'] = ['Programmer', 'Technician']
employee
```

[62]:
```
{'name': 'Bob',
 'salary': 1000,
 'age': 28,
 'company': 'ACME',
 'job': ['Programmer', 'Technician']}
```

[63]:
```
print(employee['name'], 'has', len(employee['job']), 'roles:',␣
↪employee['job'][0], 'and', employee['job'][1])
```

```
Bob has 2 roles: Programmer and Technician
```

Παράδειγμα: Βάση δεδομένων (λίστα) με στοιχεία φοιτητών

[64]:
```
student_db = []
for i in range(2):
    rec = {}
    rec['name'] = input('Student name? ')
    info = {}
    info['year'] = input('Year of study? ')
    info['AM'] = input('AM? ')
    rec['details'] = info
    student_db.append(rec)
```

```
Student name?  Mary
Year of study?  1
AM?  201900123
Student name?  John
Year of study?  3
AM?  201600320
```

[65]:
```
student_db
```

[65]:
```
[{'name': 'Mary', 'details': {'year': '1', 'AM': '201900123'}},
 {'name': 'John', 'details': {'year': '3', 'AM': '201600320'}}]
```

1ος τρόπος: επανάληψη με δείκτες

[66]:
```
for i in range(len(student_db)):
    print(i+1, ')', student_db[i]['name'], 'has AM',␣
↪student_db[i]['details']['AM'])
```

```
1 ) Mary has AM 201900123
2 ) John has AM 201600320
```

2ος τρόπος: επανάληψη χωρίς δείκτες

[67]:
```
for student in student_db:
    print(student['name'], 'has AM', student['details']['AM'])
```

```
Mary has AM 201900123
John has AM 201600320
```

Παράδειγμα: Βάση δεδομένων (λεξικό) με στοιχεία φοιτητών

[68]:
```
student_db = {}
for i in range(2):
    AM = input('AM? ')
    info = {}
    info['name'] = input('Student name? ')
    info['year'] = input('Year of study? ')
    student_db[AM] = info
```

```
AM?  201000012
Student name?  George
Year of study?  9
AM?  200600314
Student name?  Georgia
Year of study?  13
```

[69]:
```
student_db
```

[69]:
```
{'201000012': {'name': 'George', 'year': '9'},
 '200600314': {'name': 'Georgia', 'year': '13'}}
```

[70]:
```
for key in student_db:
    print(student_db[key]['name'], 'has AM', key)
```

```
George has AM 201000012
Georgia has AM 200600314
```

Σύγκριση με λίστες

Τα λεξικά προτιμώνται σε σχέση με τις λίστες για: - δεδομένα που προσδιορίζονται με κάποιο όνομα ή ετικέτα - στην περίπτωση αυτή η αναζήτηση τιμής με βάση το όνομα είναι *γρηγορότερη* από γραμμική αναζήτηση με βάση τη θέση. - για αραιές δομές (βλ. παρακάτω) - για συλλογές δεδομένων που αναπτύσσονται σε τυχαίες θέσεις (βλ. παρακάτω)

Παραδείγματα

1. Αραιοί πίνακες = πίνακες με πολλά μηδενικά στοιχεία - Τα μηδενικά στοιχεία *δεν αποθηκεύονται*

Υπολογισμός γινομένου αραιού πίνακα με διάνυσμα, $y = Ax$

$$y = Ax = \begin{pmatrix} 2 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 4 & 5 \end{pmatrix} \begin{pmatrix} 10 \\ 12 \\ 15 \\ 8 \end{pmatrix} = \begin{pmatrix} y_0 \\ y_1 \\ y_2 \end{pmatrix}$$

Ο *αραιός* πίνακας A αποθηκεύεται ως λεξικό με κλειδιά πλειάδες

[71]:
```
A = {(0,0): 2, (0,3): 1, (1,1): 1, (2,3): 5, (2,2): 4}
x = [10, 12, 15, 8]
y = [0]*3
for ij in A:
    y[ij[0]] += A[ij]*x[ij[1]]
print(y)
```

```
[28, 12, 100]
```

2. Λίστες με στοιχεία μόνο σε συγκεκριμένες θέσεις

Εστω ότι έχουμε μια σειρά δεδομένων, π.χ. μέγιστη θερμοκρασία που σημειώθηκε σε ένα τόπο, για κάποια (ενδεχομένως όχι όλα) έτη από το 1900 μέχρι σήμερα (2020). Θέλουμε να καταχωρήσουμε τα δεδομένα αυτά και να τα επεξεργαστούμε με ένα πρόγραμμα. Π.χ. να βρούμε τον μέσο όρο

- Με λίστα

[72]:
```
data = [0]*2020
nyears = 0
while True:
    year = int(input('Year? '))
    if year < 0:
        break
    nyears += 1
    temperature = float(input('Temperature? '))
    data[year] = temperature
mo = sum(data)/nyears
print('MO =', mo)
```

```
Year?  1917
Temperature?  31.8
Year?  2012
Temperature?  38.4
Year?  2018
Temperature?  37.7
Year?  -10
```

```
MO = 35.96666666666667
```

Δεσμεύουμε 2020 θέσεις μνήμης για τη λίστα `data`, ενώ θα χρησιμοποιήσουμε μόνο λίγες από αυτές.

- Με λεξικό

[73]:
```
data = {}
while True:
    year = int(input('Year? '))
    if year < 0:
        break
    temperature = float(input('Temperature? '))
    data[year] = temperature
mo = sum(data.values())/len(data)
print('MO =', mo)
```

```
Year?  2001
Temperature?  28.5
Year?  1924
Temperature?  30
Year?  1976
Temperature?  34.3
Year?  -1
```

```
MO = 30.933333333333334
```

Δεσμεύουμε μόνο τις θέσεις μνήμης που θα χρειαστούμε

3. Εναλλαγή τιμών κλειδιών

Υπάρχουν 2 θέματα:

1) Οι τιμές πρέπει να είναι *αμετάβλητου* τύπου για να μπορούν να χρησιμοποιηθούν ως κλειδιά σε λεξικό.
2) Οι τιμές σε ένα λεξικό μπορεί να μην είναι μοναδικές, ενώ τα κλειδιά πρέπει να είναι. Τα κλειδιά που αντιστοιχούν σε ίδιες τιμές αποθηκεύονται στο νέο λεξικό ως λίστες.
    - Για παράδειγμα το `{'a': 1, 'b': 1}` γίνεται `{1: ['a', 'b']}`

[74]:
```
old = {'a': 1, 'b': 2, 'c': 3, 'A': 1, 'B': 2, 'C': -1, 'xyz': 'abc'}
new = {}
for k, v in old.items():
    if v not in new:
        new[v] = [k]
    else:
        new[v].append(k)
print(new)
```

```
{1: ['a', 'A'], 2: ['b', 'B'], 3: ['c'], -1: ['C'], 'abc': ['xyz']}
```