Σύνολα

Μιχάλης Δρακόπουλος

Απρίλιος 2020

ΣΥΝΟΛΑ

Οπως και στα Μαθηματικα:

- Συλλογή στοιχείων *διακριτών* μεταξύ τους χωρίς να είναι διατεταγμένα ως προς τη θέση τους.

Επιπλέον στην Python τα σύνολα:

- Μπορεί να περιέχουν στοιχεία *διαφορετικού* τύπου
- **Πρέπει** να έχουν στοιχεία *αμετάβλητου* τύπου, π.χ. `int`, `float`, `str`, `tuple`
- Τηρούν το *πρωτόκολλο επανάληψης*
- Μπορεί να τροποποιηθούν με προσθήκη/αφαίρεση στοιχείων
- Εχουν τον τύπο `set`

Δημιουργία συνόλων

- Αμεσα μέσα σε *άγκιστρα*

[1]:
```
S = {42, 'spam', 3.1415, (1.5, 2.0)}
print(S)
```

```
{(1.5, 2.0), 42, 3.1415, 'spam'}
```

> **Παρατήρηση:** Η αρχική διάταξη δημιουργίας μπορεί τελικά να αλλάξει για λόγους *αποτελεσματικότητας*!

[2]:
```
{42, 'spam', 3.1415, [1.5, 2.0]}
```

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

TypeError                                 Traceback (most recent call last)

<ipython-input-2-0e8ae38145ed> in <module>() ----> 1 {42, 'spam', 3.1415, [1.
↪5, 2.0]}


TypeError: unhashable type: 'list'
```

Παρατήρηση: Οι λίστες δεν μπορεί να είναι στοιχεία συνόλου γιατί είναι *μεταβλητός* τύπος!

- με μετατροπή άλλων αντικειμένων (που τηρούν το *πρωτόκολλο επανάληψης*) με τη συνάρτηση `set`

[3]:
```
set('abcdef'), set([10, 20, 30, 40]), set(range(5))
```

[3]:
```
({'a', 'b', 'c', 'd', 'e', 'f'}, {10, 20, 30, 40}, {0, 1, 2, 3, 4})
```

- Το κενό σύνολο δημιουργείται *αποκλειστικά* με τη συνάρτηση `set` και όχι με άγκιστρα

[4]:
```
E = set()
print(E)
```

```
set()
```

Συναρτήσεις για σύνολα

[5]:
```
S = set(range(-5, 5))
S
```

[5]:
```
{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4}
```

[6]:
```
len(S), min(S), max(S), sum(S)
```

[6]:
```
(10, -5, 4, -5)
```

[7]:
```
sorted(S)
```

[7]:
```
[-5, -4, -3, -2, -1, 0, 1, 2, 3, 4]
```

Μέθοδοι για σύνολα

[8]:
```
dir(set)
```

[8]:
```
['__and__',
 '__class__',
 '__contains__',
 '__delattr__',
 '__dir__',
 '__doc__',
 '__eq__',
 '__format__',
 '__ge__',
 '__getattribute__',
 '__gt__',
 '__hash__',
 '__iand__',
 '__init__',
```

```
'__init_subclass__',
'__ior__',
'__isub__',
'__iter__',
'__ixor__',
'__le__',
'__len__',
'__lt__',
'__ne__',
'__new__',
'__or__',
'__rand__',
'__reduce__',
'__reduce_ex__',
'__repr__',
'__ror__',
'__rsub__',
'__rxor__',
'__setattr__',
'__sizeof__',
'__str__',
'__sub__',
'__subclasshook__',
'__xor__',
'add',
'clear',
'copy',
'difference',
'difference_update',
'discard',
'intersection',
'intersection_update',
'isdisjoint',
'issubset',
'issuperset',
'pop',
'remove',
'symmetric_difference',
'symmetric_difference_update',
'union',
'update']
```

Θα δούμε κάποιες από τις εξής:

[9]:
```
dir(set)[dir(set).index('add'):]
```

[9]:
```
['add',
 'clear',
 'copy',
 'difference',
 'difference_update',
 'discard',
 'intersection',
 'intersection_update',
 'isdisjoint',
 'issubset',
 'issuperset',
 'pop',
 'remove',
 'symmetric_difference',
 'symmetric_difference_update',
 'union',
 'update']
```

- Προσθήκη στοιχείου `add`

[10]:
```
S = {1, 2, 3, 'spam', 42, 13.4}
S.add('abc')
S
```

[10]:
```
{1, 13.4, 2, 3, 42, 'abc', 'spam'}
```

Το στοιχείο πρέπει να είναι *αμετάβλητου* τύπου

[11]:
```
S.add([999, -999])
```

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

TypeError                                 Traceback (most recent call last)

<ipython-input-11-edc21cfe052b> in <module>() ----> 1 S.add([999, -999])


TypeError: unhashable type: 'list'
```

[12]:
```
S.add((999, -999))
S
```

[12]:
```
{(999, -999), 1, 13.4, 2, 3, 42, 'abc', 'spam'}
```

- Αφαίρεση στοιχείου `discard`, `remove`

[13]:
```
S.discard('spam')
S
```

[13]:
```
{(999, -999), 1, 13.4, 2, 3, 42, 'abc'}
```

[14]:
```
S.remove(1)
S
```

[14]:
```
{(999, -999), 13.4, 2, 3, 42, 'abc'}
```

Αν δεν υπάρχει το στοιχείο, η `remove` προκαλεί σφάλμα (εξαίρεση) ενώ η `discard` όχι

- Αφαίρεση και επιστροφή ενός αυθαίρετου στοιχείου `pop`

[15]:
```
el = S.pop()
print(el)
```

```
2
```

- Αντίγραφο συνόλου `copy`

[16]:
```
print(S)
T = S.copy()
print(T)
T is S
```

```
{3, 42, 13.4, (999, -999), 'abc'}
{3, 'abc', 42, 13.4, (999, -999)}
```

[16]:
```
False
```

- Διαγραφή όλων των στοιχείων `clear`

[17]:
```
T.clear()
T
```

[17]:
```
set()
```

Πράξεις συνόλων

Λογικές πράξεις Το αποτέλεσμα είναι `True` ή `False`

[18]:
```
P = {10, 20, 30}
Q = {10, 20, 30, 50}
R = {30, 10, 20}
```

- Ελεγχος μέλους `in`

[19]:
```
10 in P
```

[19]: True

- Ισότητα

[20]:
```
P == Q, P == R
```

[20]: (False, True)

- Σύνολα ξένα μεταξύ τους `isdisjoint`

[21]:
```
{1, 2, 3}.isdisjoint(Q)
```

[21]: True

[22]:
```
P.isdisjoint(Q)
```

[22]: False

- Υποσύνολα, υπερσύνολα, γνήσια υποσύνολα και γνήσια υπερσύνολα με χρήση *τελεστών*

[23]:
```
P < Q, P <= Q, P < R
```

[23]: (True, True, False)

[24]:
```
Q > R, Q >= P, Q > Q
```

[24]: (True, True, False)

- Υποσύνολα, υπερσύνολα με χρήση μεθόδων

[25]:
```
P.issubset(Q), Q.issuperset(Q)
```

[25]: (True, True)

Παρατήρηση Η παράμετρος των μεθόδων `isdisjoint`, `issubset` και `issuperset` μπορεί να είναι οποιοδήποτε αντικείμενο που τηρεί το *πρωτόκολλο επανάληψης* (μετατρέπεται αυτόματα σε σύνολο).

[26]:
```
P.issuperset([10,30]),  Q.issubset(range(10,100)), R.isdisjoint(['a','b','c'])
```

[26]: (True, True, True)

Πράξεις συνολοθεωρίας Το αποτέλεσμα είναι σύνολο

[27]:
```
cs1 = {16101, 17222, 18102, 19103}
cs2 = {19250, 17222}
```

- Ενωση

[28]:
```
cs1 | cs2
```

[28]:
```
{16101, 17222, 18102, 19103, 19250}
```

- Τομή

[29]:
```
cs1 & cs2
```

[29]:
```
{17222}
```

- Διαφορά

[30]:
```
cs1 - cs2
```

[30]:
```
{16101, 18102, 19103}
```

- Συμμετροδιαφορά

[31]:
```
cs1 ^ cs2
```

[31]:
```
{16101, 18102, 19103, 19250}
```

Οι παραπάνω πράξεις υλοποιούνται και με χρήση μεθόδων

Πράξη	Τελεστής	Μέθοδος χωρίς ενημέρωση	Μέθοδος με ενημέρωση
Ενωση	`|`	`union`	`update`
Τομή	`&`	`intersection`	`intersection_update`
Διαφορά	`-`	`difference`	`difference_update`
Συμμετροδιαφορά	`^`	`symmetric_difference`	`symmetric_difference_update`

Στις πράξεις με χρήση τελεστών (2η στήλη) και οι δύο όροι πρέπει να είναι σύνολα και το αποτέλεσμα είναι ένα νέο σύνολο.

[32]:
```
all = cs1 | cs2
print(cs1)
print(cs2)
print(all)
```

```
{18102, 16101, 17222, 19103}
{19250, 17222}
{19250, 16101, 18102, 17222, 19103}
```

Στις πράξεις με χρήση μεθόδων χωρίς ενημέρωση (3η στήλη), η παράμετρος, εκτός από σύνολο, μπορεί να είναι και οποιοδήποτε *άλλο αντικείμενο που τηρεί το πρωτόκολλο επανάληψης* (μετατρέπεται αυτόματα σε σύνολο)

[33]:
```
new_cs2 = cs2.union([19001, 19002])
print(cs2)
```

```
print(new_cs2)
```

```
{19250, 17222}
{19001, 19250, 19002, 17222}
```

Οι μέθοδοι της 3ης στήλης, επιπλέον, δεν δημιουργούν νέο σύνολο αλλά αλλάζουν (ενημερώνουν) το ίδιο το σύνολο στο οποίο αναφέρονται.

[34]:
```
new_cs2.difference_update((17222, 19250))
print(new_cs2)
```

```
{19001, 19002}
```

Χρήση συνόλων σε γενικότερες προγραμματιστικές εφαρμογές

- "Φιλτράρισμα" επαναλαμβανόμενων τιμών σε μια αλληλουχία

[35]:
```
L = [10, 2, 1, 10, -1, 2, 10]
L
```

[35]:
```
[10, 2, 1, 10, -1, 2, 10]
```

[36]:
```
L = list(set(L))
L
```

[36]:
```
[1, 10, 2, -1]
```

Η διάταξη όμως *αλλάζει*!

- Διαφορές σε συλλογές

[37]:
```
set('spam') - set('ham')
```

[37]:
```
{'p', 's'}
```

- Ελεγχος ισότητας ανεξάρτητα από τη διάταξη

[38]:
```
set('spam') == set('mpas')
```

[38]:
```
True
```

Επαναληπτική επεξεργασία συνόλων

Προφανώς χωρίς δείκτες!

Παράδειγμα Δημιουργία συνόλου με είσοδο προκαθορισμένου αριθμού στοιχείων τύπου `str`

[39]:
```
n = int(input('How many elements? '))
S = set()
for i in range(n):
```

```
    x = input('Next element? ')
    S.add(x)
print(S)
```

```
How many elements?  4
Next element?  Java
Next element?  Python
Next element?  C++
Next element?  Fortran77
```

```
{'Python', 'Fortran77', 'Java', 'C++'}
```

Παράδειγμα Εμφάνιση όλων των στοιχείων με μήκος < 5

[40]:
```
for x in S:
    if len(x) < 5:
        print(x)
```

```
Java
C++
```