Εργαστήριο 7

Εισαγωγή στον Προγραμματισμό Υπολογιστών

Περιεχόμενα

Χειρισμός λιστών

```
>>> ls = list(map(_____________, range(10)))
>>> ls
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

- Κατασκευάστε την ίδια λίστα με *list comprehension*:

```
>>> ls = [_____________________________]
>>> ls
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

- Κατασκευάστε την ίδια λίστα με επαναληπτική εισαγωγή:

```
>>> ls = []
>>> for _____________:
        ls._____________
>>> ls
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Χειρισμός λιστών

```
>>> ls = list(map(lambda x: x*x, range(10)))
>>> ls
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

- Κατασκευάστε την ίδια λίστα με *list comprehension*:

```
>>> ls = [x*x for x in range(10)]
>>> ls
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

- Κατασκευάστε την ίδια λίστα με επαναληπτική εισαγωγή:

```
>>> ls = []
>>> for x in range(10):
        ls.append(x * x)
>>> ls
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Χειρισμός λιστών

- Εισαγωγή πολλών στοιχείων

```
>>> kids = ['Huey']
>>> kids._____(_____)
>>> kids
['Huey', 'Louie', 'Dewey']
```

Χειρισμός λιστών

- Εισαγωγή πολλών στοιχείων

```
>>> kids = ['Huey']
>>> kids.extend(['Louie', 'Dewey'])
>>> kids
['Huey', 'Louie', 'Dewey']
```

Χειρισμός λιστών

```
>>> original_ls = [2, 39, -3, 8, 16, 4, 9]
>>> ls = original_ls
```

- Αφαιρέστε τους αριθμούς 39, -3, 9 από τη λίστα `original_ls`, χωρίς να χρησιμοποιήσετε το όνομα `original_ls`

```
>>> _______________
>>> _______________
>>> _______________
>>> original_ls
[2, 8, 16, 4]
```

Χειρισμός λιστών

```
>>> original_ls = [2, 39, -3, 8, 16, 4, 9]
>>> ls = original_ls
```

- Αφαιρέστε τους αριθμούς 39, -3, 9 από τη λίστα `original_ls`, χωρίς να χρησιμοποιήσετε το όνομα `original_ls`

```
>>> ls.remove(39)
>>> ls.remove(-3)
>>> ls.remove(9)
>>> original_ls
[2, 8, 16, 4]
>>> ls is original_ls
True
```

Χειρισμός λιστών

- Αφαίρεση *όλων* των εμφανίσεων μιας τιμής απο λίστα

```
>>> def remove_all(ls, x):
        for y in ls:
            if y == x:
                ls.remove(y)

>>> ls = [2, 4, 4, -2, 'world']
>>> remove_all(ls, 4)
>>> ls
[2, 4, -2, 'world']
```

➢ Διορθώστε τον **κώδικα** ώστε να δίδεται το σωστό αποτέλεσμα:

```
>>> ls
[2, -2, 'world']
```

Χειρισμός λιστών

- Αφαίρεση *όλων* των εμφανίσεων μιας τιμής απο λίστα

```
>>> def remove_all(ls, x):
        for y in ls[:]:
            if y == x:
                ls.remove(y)

>>> ls = [2, 4, 4, -2, 'world']
>>> remove_all(ls, 4)
>>> ls
[2, -2, 'world']
```

Χειρισμός λιστών

- Αφαίρεση *όλων* των εμφανίσεων μιας τιμής απο λίστα

```
>>> def remove_all(ls, x):
        _____ = _____________________________ # άλλη λύση, σε μια γραμμή;
```

```
>>> ls = [2, 4, 4, -2, 'world']
>>> remove_all(ls, 4)
>>> ls
[2, -2, 'world']
```

Χειρισμός λιστών

- Αφαίρεση *όλων* των εμφανίσεων μιας τιμής απο λίστα

```
>>> def remove_all(ls, x):
        ls[:] = [y for y in ls if y != x] # άλλη λύση, σε μια γραμμή;
```

```
>>> ls = [2, 4, 4, -2, 'world']
>>> remove_all(ls, 4)
>>> ls
[2, -2, 'world']
```

Χειρισμός λιστών

- Κατασκευή νέας λίστας χωρίς μια συγκεκριμένη τιμή

```
>>> def remove_all_immutable(ls, x):
        _____________________________________ # μη λάβετε υπόψη το πλήθος γραμμών
        _____________________________________

>>> ls = [2, 4, 4, -2, 'world']
>>> newls = remove_all_immutable(ls, 4)
>>> newls
[2, -2, 'world']
>>> ls
[2, 4, 4, -2, 'world']
```

Χειρισμός λιστών

- Κατασκευή νέας λίστας χωρίς μια συγκεκριμένη τιμή

```
>>> def remove_all_immutable(ls, x):
        return list(filter(lambda y: y != x, ls))


>>> ls = [2, 4, 4, -2, 'world']
>>> newls = remove_all_immutable(ls, 4)
>>> newls
[2, -2, 'world']
>>> ls
[2, 4, 4, -2, 'world']
```

Χειρισμός λιστών

- Απεικόνιση τιμών *σε νέα* λίστα

```
>>> def map_immutable(func, ls):
        _____________________________ # μη λάβετε υπόψη το πλήθος γραμμών
        _____________________________

>>> ls = [2, 4, 9, -2]
>>> newls = map_immutable(lambda x: 2 * x, ls)
>>> newls
[4, 8, 18, -4]
>>> ls
[2, 4, 9, -2]
```

Χειρισμός λιστών

- Απεικόνιση τιμών *σε νέα* λίστα

```
>>> def map_immutable(func, ls):
        return [func(x) for x in ls]


>>> ls = [2, 4, 9, -2]
>>> newls = map_immutable(lambda x: 2 * x, ls)
>>> newls
[4, 8, 18, -4]
>>> ls
[2, 4, 9, -2]
```

Χειρισμός λιστών

- Απεικόνιση τιμών *στην ίδια* λίστα

```
>>> def map_mutable(func, ls):
        _____________________________ # μη λάβετε υπόψη το πλήθος γραμμών
        _____________________________
        _____________________________

>>> ls = [2, 4, 9, -2]
>>> map_mutable(lambda x: 2 * x, ls)
>>> ls
[4, 8, 18, -4]
```

Χειρισμός λιστών

- Απεικόνιση τιμών *στην ίδια* λίστα

```
>>> def map_mutable(func, ls):
        for i in range(len(ls)):
            ls[i] = func(ls[i])
```

```
>>> ls = [2, 4, 9, -2]
>>> map_mutable(lambda x: 2 * x, ls)
>>> ls
[4, 8, 18, -4]
```

Χειρισμός λιστών

- Νέα λίστα με στοιχεία σε αντίστροφη σειρά

```
>>> def invert_immutable(ls):
        _____________________________ # μη λάβετε υπόψη το πλήθος γραμμών
        _____________________________

>>> ls = [1, 2, 3, 4]
>>> newls = invert_immutable(ls)
>>> newls
[4, 3, 2, 1]
>>> ls
[1, 2, 3, 4]
```

Χειρισμός λιστών

- Νέα λίστα με στοιχεία σε αντίστροφη σειρά

```
>>> def invert_immutable(ls):
        return ls[::-1]


>>> ls = [1, 2, 3, 4]
>>> newls = invert_immutable(ls)
>>> newls
[4, 3, 2, 1]
>>> ls
[1, 2, 3, 4]
```

Χειρισμός λιστών

- Αντιστροφή σειράς στοιχείων στην *ίδια* λίστα

```
>>> def invert_mutable(ls):
        _____________________________ # μη λάβετε υπόψη το πλήθος γραμμών
        _____________________________

>>> ls = [1, 2, 3, 4]
>>> invert_mutable(ls)
>>> ls
[4, 3, 2, 1]
```

Χειρισμός λιστών

- Αντιστροφή σειράς στοιχείων στην *ίδια* λίστα

```
>>> def invert_mutable(ls):
        for i in range(len(ls) // 2):
            ls[i], ls[-i - 1] = ls[-i - 1], ls[i]

>>> ls = [1, 2, 3, 4]
>>> invert_mutable(ls)
>>> ls
[4, 3, 2, 1]
```

Χειρισμός λιστών

- Αντιστροφή σειράς στοιχείων στην *ίδια* λίστα

```
>>> def invert_mutable(ls):
        _____ = _________ # άλλη λύση, σε μια γραμμή;
```

```
>>> ls = [1, 2, 3, 4]
>>> invert_mutable(ls)
>>> ls
[4, 3, 2, 1]
```

Χειρισμός λιστών

- Αντιστροφή σειράς στοιχείων στην *ίδια* λίστα

```
>>> def invert_mutable(ls):
        ls[:] = ls[::-1] # άλλη λύση, σε μια γραμμή;


>>> ls = [1, 2, 3, 4]
>>> invert_mutable(ls)
>>> ls
[4, 3, 2, 1]
```

Χειρισμός συμβολοσειρών

- Επιλέξτε τη σωστή μέθοδο/τελεστή από τη δεξιά στήλη

```
>>> s = 'hello world'
>>> _______
'dlrow olleh'
>>> _______
['hello', 'world']
>>> _______
'hello-world'
>>> _______
'hello'
>>> _______
3
>>> _______
'Help'
>>> _______
'Hello world'
```

1. `>>> s.split()`
2. `>>> 'Hello'.replace('lo','p')`
3. `>>> 'Hello'.lower()`
4. `>>> '  Hello world  '.strip()`
5. `>>> s[::-1]`
6. `>>> 'Hello'.find('lo')`
7. `>>> '-'.join(['hello', 'world'])`

Χειρισμός συμβολοσειρών

- Επιλέξτε τη σωστή μέθοδο/τελεστή από τη δεξιά στήλη

```
>>> s = 'hello world'
>>> ____5____
'dlrow olleh'
>>> ____1____
['hello', 'world']
>>> ____7____
'hello-world'
>>> ____3____
'hello'
>>> ____6____
3
>>> ____2____
'Help'
>>> ____4____
'Hello world'
```

1. `>>> s.split()`
2. `>>> 'Hello'.replace('lo','p')`
3. `>>> 'Hello'.lower()`
4. `>>> '  Hello world  '.strip()`
5. `>>> s[::-1]`
6. `>>> 'Hello'.find('lo')`
7. `>>> '-'.join(['hello', 'world'])`

Χειρισμός συμβολοσειρών

- Υλοποιήστε συνάρτηση `count_unique(s)` η οποία επιστρέφει *το πλήθος των διαφορετικών λέξεων* στη συμβολοσειρά `s`

- Παράδειγμα:

```
>>> count_unique('to be or not to be')
4
>>> count_unique('The first second was alright but the second second was tough')
7
```

Χειρισμός συμβολοσειρών

- Υλοποιήστε συνάρτηση `count_unique(s)` η οποία επιστρέφει *το πλήθος των διαφορετικών λέξεων* στη συμβολοσειρά `s`

- Παράδειγμα:

```
>>> count_unique('to be or not to be')
4
>>> count_unique('The first second was alright but the second second was tough')
7
```

➢ Λύση (χωρίς αντικείμενα `set` που θα δείτε στις παρακάτω διαλέξεις)

```
>>> def count_unique(s):
        unique = []
        for word in s.lower().split():
            if word not in unique:
                unique.append(word)
        return len(unique)
```

Tuples

- Tuple: *immutable* λίστα

```
>>> tup = ('hello', 4, 'world', 5)
>>> tup[____]
('hello', 'world')
>>> tup._____('world') # δείκτης στοιχείου 'world'
2
>>> tup[0] = 'goodbye'
_____
```

Tuples

- Tuple: *immutable* λίστα

```
>>> tup = ('hello', 4, 'world', 5)
>>> tup[::2]
('hello', 'world')
>>> tup.index('world')
2
>>> tup[0] = 'goodbye'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

Tuples

- Εξηγήστε τη διαφορά μεταξύ `tuple` και `list`:

```
>>> tup = (1, 3, 2, 9)
>>> sup = tup
>>> tup += (4, 4)
>>> tup
(1, 3, 2, 9, 4, 4)
>>> sup
(1, 3, 2, 9)
```

```
>>> ls = [1, 3, 2, 9]
>>> ms = ls
>>> ls += [4, 4]
>>> ls
[1, 3, 2, 9, 4, 4]
>>> ms
[1, 3, 2, 9, 4, 4]
```

Tuples

- Εξηγήστε τη διαφορά μεταξύ `tuple` και `list`:

```
>>> tup = (1, 3, 2, 9)
>>> sup = tup
>>> tup += (4, 4)
>>> tup
(1, 3, 2, 9, 4, 4)
>>> sup
(1, 3, 2, 9)

>>> sup is tup
False
```

```
>>> ls = [1, 3, 2, 9]
>>> ms = ls
>>> ls += [4, 4]
>>> ls
[1, 3, 2, 9, 4, 4]
>>> ms
[1, 3, 2, 9, 4, 4]

>>> ms is ls
True
```

Λεξικά

- Λεξικό (τύπος `dict`): δεδομένα όπου χρησιμοποιείται δείκτης οποιουδήποτε τύπου (αντί μόνο `int` όπως στις λίστες)

```
>>> contacts = {'Maria': 44444, 'Georgia': 33333, 'Manos': 11111,\
                'Dina': 22222, 'Nikos':99999}

>>> contacts[______]
44444
>>> contacts[______]
99999
>>> 'Dina' in contacts

______
>>> contacts[__________] = ______
>>> contacts
{'Maria': 44444, 'Georgia': 33333, 'Manos': 11111, 'Dina': 22222,
'Nikos':99999, 'Christina': 55555}
```

Λεξικά

- Λεξικό (τύπος `dict`): δεδομένα όπου χρησιμοποιείται δείκτης οποιουδήποτε τύπου (αντί μόνο `int` όπως στις λίστες)

```
>>> contacts = {'Maria': 44444, 'Georgia': 33333, 'Manos': 11111,\
                'Dina': 22222, 'Nikos':99999}

>>> contacts['Maria']
44444
>>> contacts['Nikos']
99999
>>> 'Dina' in contacts
True
>>> contacts['Christina'] = 55555
>>> contacts
{'Maria': 44444, 'Georgia': 33333, 'Manos': 11111, 'Dina': 22222,
'Nikos':99999, 'Christina': 55555}
```

Παράδειγμα: τηλεφωνικές επαφές

- Να ορίσετε τις συναρτήσεις `call` και `show_contacts` που λειτουργούν ως εξής:

```
>>> contacts = {'Maria': 44444, 'Georgia': 33333, 'Manos': 11111, \
                'Dina': 22222, 'Nikos': 99999}

>>> call('Georgia', contacts)
Calling Georgia (33333)...
>>> call('Eleni', contacts)
Unknown name

>>> show_contacts(contacts)
Name: Maria, Number: 44444
Name: Georgia, Number: 33333
Name: Manos, Number: 11111
Name: Dina, Number: 22222
Name: Nikos, Number: 99999
```

Παράδειγμα: τηλεφωνικές επαφές

```
def call(name, contacts):
    if name in contacts:
        print('Calling', name, '(' + str(contacts[name]) +')...')
    else:
        print('Unknown name')


def show_contacts(contacts):
    sorted_names = list(contacts.keys())
    sorted_names.sort()
    for name in sorted_names:
        print('Name:', name, 'Number:', str(contacts[name]))
```

Παράδειγμα: τηλεφωνικές επαφές

- Να ορίσετε συναρτήσεις `add_contact(name, number, contacts)` και `remove_contact(name, contacts)` οι οποίες προσθέτουν και αφαιρούν επαφές αντίστοιχα

- Παραδείγματα:

```
>>> add_contact('Eleni', 66666, contacts)
>>> call('Eleni', contacts)
Calling Eleni (66666)...

>>> remove_contact('Eleni', contacts)
>>> call('Eleni', contacts)
Unknown name
>>> remove_contact('Michalis', contacts)
Unknown name
```

Παράδειγμα: τηλεφωνικές επαφές

```
def add_contact(name, number, contacts):
    contacts[name] = number


def remove_contact(name, contacts):
    if name in contacts:
        del contacts[name]
    else:
        print('Unknown name')
```