פונקציות קסם / עומר רוזנבוים, שי סדובסקי

map

היעוד של map הוא לעבור על כל איבר ברשימה ולעשות פעולה מוגדרת.

map מקבלת רשימה ופונקציה, ומחזירה רשימה חדשה, בה כל איבר הוא התוצאה של הפונקציה על איבר מהרשימה המקורית.

```
>>> def mul_2(number):
        return number*2
>>> map(mul_2, xrange(10))
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

אמנם יכולנו לעשות את אותו דבר עם רשימות (כפי שמודגם מיד), אך לפעמים שימוש ב-map וביתר הפונקציות הקסומות שנלמד הוא נוח יותר, וכמו כן אתם עשויים להיתקל בו כשתקראו קוד.

```
>>> [x*2 for x in xrange(10)]
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

Reduce

מקבלת רשימה ופונקציה, כשהפונקציה מקבלת שני פרמטרים.

reduce מריצה את הפונקציה על זוג האיברים הראשון ברשימה המקורית. לאחר מכן, היא משתמשת בפלט בתור הארגומנט הראשון לפונקציה, ובאיבר הבא בתור הארגומנט השני, וכך הלאה.

```
>>> def add(x, y):
        return x+y
>>> reduce (add, xrange(10))
```

זוכרים את תרגיל summer? נסו לפתור אותו כעת, עם reduce. לכמה שורות קוד אתם זקוקים? ☺

Filter

מאפשרת לנו לסנן איברים מתוך רשימה. לשם כך, מקבלת רשימה ופונקציה, ומחזירה רשימה חדשה עם האיברים שעברו סינון.

```
>>> def is_even(number):
        return number%2 == 0
>>> flter(is_even, xrange(10))
[0, 2, 4, 6, 8]
```

גם את הפעולה הזו יכלנו לבצע עם רשימה בלבד, אבל בצורה פחות מגניבה:

```
>>> [x for x in xrange(10) if x%2 == 0]
[0, 2, 4, 6, 8]
```

Lambda

עד כה, עבור כל כתיבה של map, filter או reduce היינו צריכים להצהיר על פונקציה חדשה, גם כשהפונקציה הייתה מאוד פשוטה. lambda עוזרת לנו לקצר את העניין הזה. בואו ניצור פונקציה עם lambda:

```
>>> lambda x: x*2
<function <lambda> at 0x01FD21B0>
```

טוב, אבל אי אפשר לקרוא לפונקציה הזו. כעת נצליח לקרוא לה:

```
>>> mul_2 = lambda x: x*2
>>> mul_2(4)
8
```

מגניב. עכשיו אפשר לקרוא לה עם map:

```
>>> map(mul_2, xrange(10))
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

והדרך הכי אלגנטית היא להכריז על הפונקציה בתוך ה-map:

```
>>> map(lambda x: x*2, xrange(10))
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
>>> filter(lambda x: x%2==0, xrange(10))
[0, 2, 4, 6, 8]
```

זה כבר יפה ☺