משימות למעבדה מס' 9

(Mutable data, Dispatch function)

1. (2014 מועד א' שאלה 2)

עליכם להשלים את הפונקציה **text_preprocessing** - פונקציה היוצרת **pipeline** לעיבוד טקסט וסופרת תדירות של מילים בטקסט. ניתן להניח כי הטקסט מגיע ללא סימני פיסוק ומכיל אותיות קטנות וגדולות, מספרים, וכן רווחים בין המילים. תהליך העיבוד מורכב מהשלבים הבאים:

1) הפרדה בין המילים.
2) סינון מילות עצירה (**stopwords**) ומספרים.
3) חישוב תדירות מילים.

הפונקציה שומרת את התוצאות של **שלב 2** (מילים אחרי סינון, **<2>**,**<1>**) במשתנה **filtered** ולאחר מכן מחשבת תדירות **(<3>)** לכל אחת מהמילים.

```
def text_preprocessing(text, stopwords):
        bag_words = {}
        filtered = filter(<1>, map(<2>, text.split()))
        <3>
        return bag_words
```

דוגמת הרצה:

```
>>> stop_list = ('is', 'it', 'a', 'the', 'my', 'and')
>>> text_preprocessing ('My cat is 10 and it is a very fat cat', stop_list)
{'fat': 1, 'very': 1, 'cat': 2}
```

להלן חתימות הפונקציות שניתן להיעזר בהן(שתיהן שייכות לממשק של **str**):

isnumeric() - מחזירה True כאשר מופעלת על מחרוזת המורכבת מתווים מספריים בלבד.

lower() - מחזירה עותק של מחרוזת עם כל התווים מומרים לאותיות קטנות.

.2 עליכם לממש מערכת לניהול חשבון בנק (בעזרת פונקציית **dispatch**), התומכת בפעולות הבאות:

- יצירת חשבון בעל מאזן התחלתי של **0**.
- פעולת **get** המחזירה את הסכום בחשבון.
- פעולת שינוי הסכום בחשבון (ע"י הוספה או גריעה של סכום נקוב).
- פעולת העברת כספים לחשבון אחר.

על התוכנה שלכם לתמוך ב-**Driver** הבא:

Driver

```
a1 = make_account()
print (a1)
a2 = make_account()
print (a1('change')(20))
print (a1('get'))
print (a1('change')(-25))
print (a1('move')(a2, 7))
print (a2('move')(a1, 2))
print (a1('move')(a2, 30))
print (a1('move')(a2, -30))
```

Output

```
<function make_account.<locals>.dispatch at 0x03552390>

20
20
out of funds during change
(13, 7)
(5, 15)
out of funds during move
Negative transaction amount
```

.3 (2013 מועד א' שאלה 2)

ממש\י פייפ ליין לחישוב מכפלה קרטזית (**cartesian product**) של אלמנטים "תקינים" בשתי רשימות. בכל סעיף יש להשתמש בסעיף הקודם

1) השלימו כך שבהינתן רצף וערך כלשהו, נוצר **tuple** של המכפלה הקרטזית בין הערך והרצף:

make_pairs = lambda el, lst: tuple(**<1>**)

דוגמת הרצה:

```
>>> make_pairs(5, (1,2,3))
((5, 1), (5, 2), (5, 3))
```

2) השלימו כך שבהינתן שני רצפים נוצר **tuple** (לא שטוח - מורכב מ-**tuples** של זוגות) של המכפלה הקרטזית שלהם (השתמשו ב-**make_pairs** מסעיף קודם)

c_prod = lambda lst1, lst2: tuple(**<2>**)

דוגמת הרצה:

```
>>> c_prod((1, 2), (3, 4))
(((1, 3), (1, 4)), ((2, 3), (2, 4)))
```

3) השלימו כך שבהינתן שני רצפים נוצר **tuple** (**שטוח** - מורכב אך ורק מזוגות) של מכפלה קרטזית שלהם (השתמשו בסעיף הקודם):

flat_c_prod = lambda lst1, lst2: reduce(**<3>**, **<4>**,())

דוגמת הרצה:

```
>>> flat_c_prod((1, 2), (3, 4))
((1, 3), (1, 4), (2, 3), (2, 4))
```

4) השלימו כך שבהינתן שני רצפים ופרדיקט **p** נוצר **tuple** (שטוח) של המכפלה הקרטזית של האלמנטים ה"תקינים" המקיימים את התנאי של **p** (השתמשו בסעיף הקודם):

cond_c_prod = lambda p, lst1, lst2: **<5>**

דוגמת הרצה:

```
>>> cond_c_prod (lambda x: type(x)== int, (1, 2, 3.5), (3, 'a', 4))
((1, 3), (1, 4), (2, 3), (2, 4))
```

4. (2013 מועד ב' שאלה 2)

אתם מתבקשים לכתוב סימולציה של כרטיס חניה אלקטרוני ארצי **שמיזי פארק**. שימוש בכרטיס הזה מאפשר תשלום חנייה רק עבור משך החנייה בפועל (בשעות). משתמש יכול לקנות כרטיס חדש (ריק), להטעין אותו ולהפעיל בכל פעם כאשר הוא חונה בחנייה עירונית.

א) השלימו את הקוד בצורת **dispatch** מבוסס הודעות (ללא מילון, עם טיפול בהודעות שגויות):

```
def shmeasy_park(fee):          # fee – תעריף לשעה
    <1>
    def charge(amount):         #amount – מילוי יתרה (בשקלים)
        <2>
    def park(time):             # time – משך חניה (בשעות)
        <3>
    def dispatch( <4> ):
        <5>
    return <6>
```

דוגמת הרצה(מחייבת):

```
>>> k = shmeasy_park(5)
>>> k('charge', 100)
>>> k('park', 10)
balance left: 50.0
>>> k('add', 20)
unknown message: add
```

בהצלחה !