עקרונות שפות תוכנה

תשע"ו, סמסטר א'

<u>עבודת הגשה מס' 3</u>

הנחיות:

- יש להגיש את העבודה עד **20/12/2015**
- על כל יום איחור בהגשה, ללא הצדקה, ירדו 3% מהציון. לא ניתן להגיש <u>כלל</u> באיחור של מעל שבוע
- מותר להכין את העבודה בזוגות או בודדים בלבד.
- חובה להשתמש בשמות הפונקציות המוגדרות.
- על הקובץ להכיל רק את הפונקציות ללא הרצה.
- מימוש המשימות חייב להיות מתואם לדוגמאות הרצה.
- יש לתעד (docstrings) בתוך כל פונקציה. המטרה היא שאם משתמש מפעיל help(func) אז הוא יקבל את כל המידע הדרוש להבנת הפונקציה.
- ההגשה היא באתר moodle – רק סטודנט/ית אחד מגיש - צריכים לעלות קובץ עם אותו השם שהוא צירוף ת"ז שלהם עם קו תחתון וקידומת מספר עבודה. לדוגמה:

  HW2_123456789_123456789.zip
- יש להגיש קובץ PY עבור החלק התכנותי וקובץ PDF עבור החלק התאורטי מכווצים יחד בקובץ ZIP. לכל 3 קבצים חייב להיות אותו שם (לדוגמה: HW2_123456789_123456789.zip/py/pdf)

בהצלחה!

כל שאלה ופניה בנוגע לתרגיל יש להפנות אך ורק לאחראית על התרגיל **מרינה ליטבק**
באימייל: marinal@sce.ac.il
פניות בכל בדרך אחרת לא יענו!

<u>חלק 1: Data abstraction</u>

1. יש לעדכן מימוש של מספר רציונאלי (rational number) שנעשה בשיעור (#4) כך שלא יהיה שימוש בטיפוסי נתונים מובנים של Python כלל (למשל, המימוש הנוכחי משתמש ב-tuple). יש להחליף שימוש ב-tuple בשימוש ב-pair שמימשנו לבד.האם יצא לכם טיפוס (מספר רציונאלי) שהוא mutable או immutable? הסבר.

<u>דוגמת הרצה</u>:

```
>>> third = make_rat(1,3)
>>> third
<function make_pair.<locals>.dispatch at 0x02A880C0>
>>>numer(third)
1
```

חלק 2: Conventional Interface, pipeline

עיבוד מוקדם של נתונים (data preprocessing) הוא מרכיב חשוב מאוד של תהליך כריית מידע (data mining). הוא אחראי על ניקוי, נירמול והשלמת נתונים מנותחים. במשימה הזאת אתם מתבקשים לבנות מנוע לעיבוד מוקדם של נתונים שהתקבלו מחיישן המודד טמפרטורה כל 30 דקות. התוצאות נרשמות בבסיס נתונים. יתכן שעקב תקלת חיישן מתקבלת תוצאה שגויה, או שעקב תקלה בשמירת נתונים לא תירשם תוצאה כלל. המנוע מקבל נתונים בצורה של מחרוזת ארוכה שמכילה רצף של מספרים המופרד בפסיקים. במקרה של ערך חסר לא יהיה כלום בין הפסיקים.
לדוגמה, המחרוזת הבאה "1,1,,100,3, 5,5,5,,1,2,3" מכילה 10 ערכים מספריים ו-2 ערכים חסרים. בהינתן ערך מקסימאלי של 10 לטמפרטורה אפשרית, אחד הערכים הוא חריג (100) וכנראה שגוי.

2. יש לבנות מנוע לעיבוד מוקדם של נתונים תוך שימוש בממשק קונבנציונאלי. יש לממש את המנוע כפונקציה בשם data_preprocessing_*** ולבנות אותו כ- pipeline של שלבים הבאים (לפי סדר):
   a. **enumerate** – קריאה ומינוי ערכים. יש להחזיר רצף של ערכים כפלט של השלב הזה.
   b. **clean (noise removal)** – הסרת ערכים שגויים . ניתן לזהות ערכים כאלו ע"י השוותם מול טמפרטורה מקסימאלית ומינימאלית (שני ערכים אלו צריכים להיקבע כארגומנטים של המנוע). כל ערך שהוא מחוץ לתווך האפשרי, יש להסיר.
   c. **complete missing values** – השלמת ערכים חסרים. יש להחליף כל ערך חסר ע"י ממוצע של שני ערכים הצמודים אליו ברצף: טמפרטורה שנמדדה לפניו ומפרטורה שנמדדה אחריו.
   d. **accumulate** – יש לממש כמה סוגים של חישוב סטטיסטיקות:
      i. **build histogram** – לבנות היסטוגרמה עבור הנתונים המנורמלים. מטרתה של היסטוגרמה להראות התפלגות של ערכים מספריים. אתם צריכים לחשב תדירות של כל ערך יחודי ולייצר רצף של צמדים (ניתן להשתמש ב-tuple לצורך זה), כאשר כל צמד מורכב מהערך עצמו והתדירות שלו. שימו לב שהרצף לא אמור להכיל כפיליות – חישבו על מבנה נתונים מתאים לאיחסון של הצמדים האלו! יש לקרוא לפונקצית מנוע שבונה היסטוגרם בשם data_preprocessing_histogram.
      ii. **get range** – לחשב תווך של ערכים. ערך ההחזרה של הפונקציה הוא שלישיה (tuple) של הערך המינימאלי, ממוצע הערכים, והערך המקסימאלי. יש לקרוא לפונקצית מנוע שמחשבת תווך ערכים בשם data_preprocessing_range

הערה: אין להשתמש בלולאות ופוקציות עזר, אלא רק בפונקציות של ממשק קונבנציונאלי ו-lambda. מותר להשתמש בפעולות השמה ע"מ לשמור תוצאות ביניים.

הערה: יש להניח לגבי קלט שערכים חסרים לא יכולים להופיע בהתחלה ו/או בסוף של המחרוזת, ולא יתכן שיופיעו 2 או יותר ערכים חסרים ברצף.

דוגמת הרצה:

```
>>> data = "1,1,,100,3, 5,5,5,,1,2,3"
>>> min_val,max_val = 0,10
>>> data_preprocessing_histogram(data,min_val,max_val)
{(1, 3), (3, 3), (5, 3), (2, 2)}
>>> data_preprocessing_range(data,min_val,max_val)
(1, 2.8, 5)
```

חלק 3: Mutable data, message passing, dispatch function, dispatch dictionary

3. יש לממש טיפוס נתונים חדש בשם **coordinate** שמייצג קאורדינאטה ב-3 מיימדים: X, Y, Z תוך שימוש ב-dispatch function ו-message passing. יש לממש את הפעולות הבאות:
   a. גישה לערך של מיימד מסוים (X, Y, Z). ההודעה המתאימה: 'get_value'
   b. עידכון של ערך במיימד מסוים (X, Y, Z). ההודעות המתאימות: 'set_value'
   c. ייצוג טקסטואלי. יש לייצג את הקאורדינאטה בדומה לייצוג של tuple בן 3 מספרים (ראה דוגמת הרצה). ההודעה המתאימה - 'str'

<u>הערה</u>: אין להשתמש בטיפוסים מובנים של Python!!!

<u>דוגמת הרצה:</u>

```
>>> p = make_coordinate(1,2,3)
>>> p('get_value')('x')
1
>>> p('get_value')('y')
2
>>> p('set_value')('y',0)
>>> p('get_value')('y')
0
>>> p('str')
'(1, 0, 3)'
```

4. בשאלה זו אתם מתבקשים לממש טיפוס נתונים חדש בשם **filter_iterator** . ניתן ליצור אובייקט של filter_iterator על רצף ***s*** ופונקציה ***p*** (פרדיקט של ארגומנט אחד) על מנת לעבור על ערכים של הרצף החדש שמתקבל על ידי סינון ערכים של s שלא מקיימים את p.

יש לממש פונקציה **get_filter_iterator** היוצרת אוביקט של filter_iterator לפי שיטת dispatch dictionary . שתי פעולות מוגדרות על הטיפוס:

a. **next** - פעולה מחזירה את האלמנט הבא של הרצף המתקבל

b. **has_more** - פעולה מחזירה – TRUE כל עוד יש אלמנטים ברצף המתקבל

<u>הערה</u>: במידה ופונקציה לא קיבלה את הארגומנט השני (p) אז היא תחזיר איטראטור על ערכים של הרצף הארגומנט (s) ללא שינוי.

<u>הערה</u>: אין לטפל בחריגות ש Python מעלה במקרים של חישובים כושלים כגון חלוקה ב-0 וכד' (תטפלו בחריגות בעבודה הבאה).

<u>דוגמת הרצה:</u>

```
>>> it = get_filter_iterator((1,0,6), lambda x: x%2==0)
>>> while it['has_more']():
        print(it['next']())
0
6
>>> it = get_filter_iterator((1,0,6))
>>> for _ in range(1,6):
        print(it['next']())
1
0
6
no more items
no more items
```

5. שאלה זאת מתייחסת למימוש של רשימה שנעשה בשיעור (make_mutable_rlist) בשיטת message passing ו-dispatch dictionary. יש להשלים\לעדכן את המימוש ע"י שינויים הבאים:

a. פעולה לבנית ייצוג טקסטואלי (**str**) אמורה להחזיר מחרוזת המייצגת רשימה בדומה לייצוג של רשימה ב-Python (למשל, רשימה שמכילה 1, 2 ו-3 תוצג כ-"[1, 2, 3]").
b. יש להוסיף פעולה חדשה בשם **slice** שמחזירה תת-רשימה (בדומה לאופרטור slicing). הפעולה תקבל 2 ארגומנטים: אינדקס התחלה (כולל) ואינקס אחרון (לא כולל) ותחזיר אוביקט חדש של mutable_list
c. יש להוסיף פעולה חדשה בשם **insert** שמאפשרת להכניס אלמנט חדש למקום מסויים ברשימה (בדומה ל-insert של list).
d. לאפשר הפעלת בנאי מעתיק (copy constructor), כלומר להעתיק איברים מרשימת-ארגומנט לפונקציה make_mutable_list לרשימה החדשה.
e. פעולה בשם **get_iterator** שמחזירה iterator על תוכן הרשימה.

<u>דוגמת הרצה:</u>

```
>>> my_list = make_mutable_rlist()
>>> for x in range(4)
            my_list['push_first'](x)
>>> my_list['str']()
[3, 2, 1, 0]
>>> my_list['insert'](1,5)
>>> my_list['str']()
[3, 5, 2, 1, 0]
>>> my_list['slice'](0,2)['str']()
[3, 5]
>>> your_list = make_mutable_rlist(my_list)
>>> your_list['str']()
[3, 5, 2, 1, 0]
>>> it = my_list['get_iterator']()
>>> while it['hasNext']():
          print(it['next']())
3
5
2
1
0
```

<u>חלק 4: שאלות תאורטיות</u>

6. סמנו אילו מהטענות נכונות והסבירו בקצרה לכל טענה:
   a. ניתן להחזיר מפונקציה ולהעביר לפונקציה כארגומנט פונקציה מסדר גבוה (high-order function).
   b. ב Python 3 -משתמשים בהצהרה nonlocal על מנת לעדכן קשירה של משתנה במסגרת גלובאלית.
   c. בשפות עם Lexical Scoping ניתן לממש טיפוסי נתונים חדשים שהם mutable תוך שימוש בפונקציות והשמה לא לוקאלית (nonlocal).