מעבדה 5 (Native types, sequences: tuples, strings pairs)

משתנים מאפשרים ליצג ולתחזק מידע על העולם באמצעות כלים חישוביים שרכשנו עד כה.

תופעות בעלות משמעות פרקטית בדר"כ ייוצגו ע"י טיפוסים מורכבים - נתונים בעלי מבנה, כלומר, אוביקטים.

אובייקט הינו מבנה בעל התכונות הבאות:

1. בעל מצב (ערך) היכול להיות מורכב מכמה משתנים פרימיטיביים.
2. בעל התנהגות - לוגיקה של אינטרקציה עם אובייקטים אחרים.
3. יודע להדפיס את עצמו

```
from datatime import date
```

date היא מחלקה (class)

```
today = date(2013, 11, 9)
```

taday היא מופע (Instance) של המחלקה date

```
str(date(2013, 11, 11) - today) => '2 days, …
```

כאן - הינו אופרטור המוגדר על אוביקטים מהסוג date, ו-str הינו אופרטור בניית מחרוזת המוגדר עבור אובייקטים מסוג date.

```
today.month => 11
```

גישה למאפיינים (attributes) - ניתן לראות ע"י ctrl+space אחרי הנקודה.

```
today.strftime("%y-%m-%d %H:%M")
```

כאן ניגשנו למתודה של האובייקט הנקראת strftime ההופכת אוביקט של date למחזורת ע"פ פורמט.

מבני נתונים מובנים ב-Python

int, float, complex - numeric types

tuples, lists, strings - ordered sequences

sets, dictionaries - unordered types

יש לשים לב לעוד אבחנה חוץ ממבנים סדורים או לא סדורים והיא האבחנה של mutable לעומת immutable (ניתן לשינוי או לא ניתן לשינוי). tuples ו-strings הינם immutable objects ז"א שהם אינם ניתנים לשינוי אחרי היצירה שלהם.

דוגמאות lists and strings:

```
s = ['ab', 2, [3, 4]]
s[0] -> 'ab'
s[0][0] -> 'a'
```

מחרוזת באורך 1 זהה לתו.

```
s[2][1] -> 4
s[-1] -> [3,4]
```

אינדקסים שליליים מתחילים מהסוף (בדומה לחישוב modulo באלגברה)

דוגמאות lists and strings mutation:

```
s[0] = 'abc'
s[0][2] = 'd' -> error string is immutable
```

דוגמאות list slicing:

```
s = [7, 8, 9, 10, 11]
s[:] -> כל הרשימה
s[2:] -> [9, 10, 11]
s[:2] -> [7, 8]
s[::2] -> [7, 9, 11]
s[1:4:1] -> [8, 9, 10]
```

דוגמאות לשינוי רשימה קיימת:

```
s += 'a' -> [7, 8, 9, 10, 11, 'a'] (יוצר רשימה זמנית)
s.extend([1, 2]) -> [7, 8, 9, 10, 11, 1, 2] (יותר יעיל)
s.insert(0, 5) -> [5, 7, 8, 9, 10, 11] (יותר יקר)
del s[2, -2] -> [5, 7, 1, 2]
```

מומלץ לבדוק פונקציות בדוקומנטציה (לדוג': count, index remove, pop)

<u>n-יה (tuple) - רשימה שהינה immutable</u>

- מוגדרת ע"י () במקום []
- יותר מהירה ובטוחה
- לא ניתנת לשינוי!

הפיכה בין tuple ל-list:

```
tuple([1, 2, 3]) -> (1, 2, 3)
list((1, 2, 3)) -> [1, 2, 3]
```

אפילו עם איטרטור:

```
tuple(range(10)) -> (0, 1, 2, … , 8, 9)
```

<u>set - חייב להכיל ערכים שהינם hashable:</u>

```
{(1, 2)} -> ok
{[1, 2]} -> error

s = set(list(range(10))) -> set([0, 1, …, 9])
s[0] -> error, set is an unordered object
len(s) -> 10
```

ניתן להפעיל len על כל סוגי רשימות ומחרוזות.

```
s.add(4)
len(s) -> 10
```

הגודל של הסט לא השתנה כי אין חזרות בסט ו-4 היה חלק ממנו לפני כן.

מומלץ לבדוק פונקציות בדוקומנטציה (לדוג': update, discard, remove, pop, union, intersection, difference).

dictionary סט של זוגות מפתח-ערך:

```
s = { ‘x’ : 2, ‘y’ : [1,2], (3,4) : 5}
```

שימו לב:

- כל מפתח חייב להיות hashable
- ערכים יכולים להיות כל דבר

דוגמאות לגישה ושינוי של מילון:

```
s[(3,4)] -> 5
s[‘x’] = s[‘y’] -> {‘x’ : [1,2], ‘y’ : [1,2], (3,4) : 5}
```

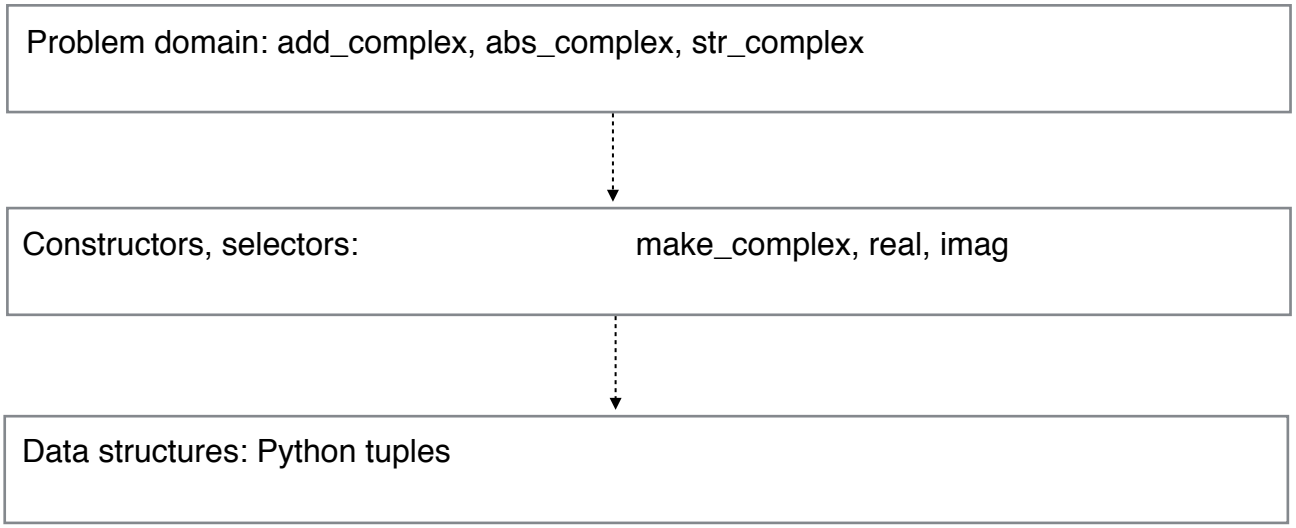
שימו לב כי המפתחות x ו-y מקושרות לאותו אובייקט.

```
s[‘x’][0] = 7
s -> { ‘x’ : [7,2], ‘y’ : [7,2], (3,4) : 5}
len(s) -> 3
‘x’ in s -> True
```

ניתן להפעיל len, in, union על כל סוגי הרשימות.

מימוש מבנה נתונים באמצעות tuples:

נממש ADT של מספרים מרוכבים (למרות שהם כבר קיימים ב-Python):




```
c = make_complex(2, 3)
str_complex(c) -> ‘(2+3i)
real(c) -> 2
imag(c) -> 3
str_complex(add_complex(c, c)) -> ‘(4 + 6i)’
abs_complex(c) -> 3.60555…
```

שינוי המימוש לשימוש ב-tuple שהינו closure עם dispatch:
נשים לב כי לאחר מימוש זוג באמצעות dispatch, יש לשנות רק ADT מרמה 1 ומעלה (בנאי ו-selectos).

```
c = make_complex(2,3)
c -> <function dispatch …>
imag(c) -> 3
str_complex(c) -> '(2+3i)'
```

<u>שימוש ברשימה רקורסיבית:</u>

```
counts = make_r_rlist(1, make_rlist(2, … (4, empty_rlist)…)
first(counts) -> 1
rest(counts) -> (2, (3, (4, None)))
len_rlist(counts) -> 4
```

תרגיל: כתוב פונקציה המקבלת רשימה רקורסיבית ויוצרת רשימה חדשה בסדר הפוך (באמצעות ה-ADT בלבד!)

- האם רשימת counts משתנה לאחר ההפעלה?

```
reverse_rlist(counts) -> (4, (3, (2, (1, None))))
```