<u>הערה לסעיפים א' ו-ב':</u>
יש לטפל בחריגות הרלוונטיות לפעולות אלו.

חלק 6: שאלות תיאורטיות

א) במערכת אובייקטים שבנינו בכיתה (של **Shmython**) פונקציה **get** <u>של אובייקט</u> תמיד מחזירה מתודה כשמקבלת שם המאפיין שהוא פונקציה.

ב) במימוש מחשבון יש מקרים (בריצות מסוימות), כאשר פונקציה רקורסיבית **calc_eval** לא מבצעת חישוב רקורסיבי.

ג) בשימוש בפונקציה גנרית **coerce_apply** ניתן להמיר טיפוס אחד לטיפוסים שונים לפי הצורך.

ד) פונקציה שמפילה פונקציה אחרת שעלולה לעלות חריגה חייבת לטפל באותה חריגה ע"י תפיסתה או העברתה לפונקציה המפעילה.

ה) בשימוש OOP (Python) לכל האובייקטים של אותה מחלקה יש בדיוק אותם שדות (תכונות) ולא יתכן מצב שלאובייקט אחד יהיה שדה שלא מופיע אצל אובייקטים אחרים.

א) אובייקטים במערכת של Synthon פונקציה get של אובייקט צו גישה למאפיינים של המופע (תיפוש רקורסיבי שבתחילת המופע ומסתיים במחלקה) כאשר הקריאה ל get היא תחליר זו פונקציה!

ב) נבנה יש מקרים בריצות מסוימות פונקציה calc_eval לא מבצעת חישוב רקורסיבי כאשר אנחנו נוקים שהוא גיסם פרימיטיבי!

ג) יתרון המשמעות של הפונקציה גנרית coerce_apply זה שלא תמיד ניתן להמיר מטיפוס אחד לטיפוס אחר. למשל, לא ניתן להמיר מס מורכב לשבר (מתמטיים).

ד) נבנה, אז שהפונקציה עלולה לעלות חריגה אמורה לתפוס אן הפונקציה שקרא לה צריכה לתפוס את החריגה אחרת אין אז מי שיתפוס זה תהיה שגיאה!

ה) לא נכון, כן האובייקטים שבאותחלים יש הסיכוי אותם שמות ותכונות באותן אך לאחר האתחול ניתן להוסיף לכל אובייקט שמות אחרים כך שבאובייקט אחד יהיה לו שדה שלא יופיע באובייקט אחר.

```
class check()
    def __init__(self, a)
```

self.a = a

A = check(5) -> A.a => 5

B = check(6) -> B.a => 6

B.b = 3 => B.b = 3

A.b => <u>Error</u>