תרגיל מחלקות וירושה

שאלה 1

א. Shape

כתבו מחלקה **מופשטת** בשם Shape שתייצג צורה במישור. המחלקה לא תכיל תכונות ובנאים, ותכיל את השיטות המופשטות area() ו-perimeter() המחזירות את שטח והיקף הצורה בהתאמה.

- הוסיפו אופרטור השוואה > בין צורה לצורה על פי שטח הצורה.

ב. Circle

כתבו מחלקה בשם Circle היורשת מ-Shape. המחלקה תכיל את התכונה **הפרטית** radius המייצגת את רדיוס המעגל.

- המחלקה תכיל את __init__ המקבלת את אורך הרדיוס ומציבה אותו בתכונה.
  אם הערך שהתקבל אינו חיובי, יוצב ברדיוס הערך 1.
- הוסיפו למחלקה את הפונקציות הדרושות על מנת להציב ולקבל את הרדיוס מחוץ למחלקה.
- הוסיפו את __str__ כדי לייצג את המעגל בפורמט הבא:

Circle: radius = [radius]

כאשר במקום הערך [radius] יופיע ערכו של הרדיוס.

- ממשו במחלקה את השיטות המופשטות שמוגדרות ב-Shape.

ג. Rectangle

כתבו מחלקה בשם Rectangle היורשת מ-Shape. המחלקה תכיל את התכונות **הפרטיות**:

width המייצג את רוחב המלבן ו-height המייצג את גובה המלבן.

- המחלקה תכיל את __init__ המקבלת את ממדי המלבן ומציבה אותם בתכונות. אם אחד מהאורכים אינו חיובי, יוצב הערך 1 במקומו.
- הוסיפו למחלקה את הפונקציות הדרושות על מנת להציב ולקבל את אורכי המלבן מחוץ למחלקה.
- הוסיפו את __str__ כדי לייצג את המלבן בפורמט הבא:

Rectangle: width = [width], height = [height]

כאשר במקום הערכים [width] ו-[height] יופיע ערכיהן של התכונות.

- ממשו במחלקה את השיטות המופשטות שמוגדרות ב-Shape.

ד. Square

כתבו מחלקה בשם Square היורשת מ-Rectangle. המחלקה לא תכיל תכונות נוספות.

- המחלקה תכיל את __ init__ המקבלת את אורך צלע הריבוע ומציבה אותה בתכונות.
- הוסיפו את __str__ כדי לייצג את המעגל בפורמט הבא:

Square: length = [length]

כאשר במקום הערך [length] יופיע אורך הצלע.

ה. ShapesCollection

כתבו מחלקה בשם **ShapesCollection** שתייצג מאגר של צורות. המחלקה תכיל את התכונה **הפרטית** הבאה: shapes – רשימה שתכיל את הצורות במאגר. **הצורות יהיו ממויינות בסדר עולה לפי השטח שלהן. כלומר, הצורה ששטחה הקטן ביותר תהיה בתא הראשון.** השתמשו לצורך זה באופרטור השוואה שהגדרתם.

הוסיפו למחלקה את __ init__ שתאתחל את התכונה.

הוסיפו למחלקה את השיטות הבאות:

- ממשו את __len()__– השיטה תחזיר את מספר הצורות שקיימות במאגר.
- ממשו את אופרטור [] כדי שאפשר יהיה לגשת לרשימה במקום ה-i
- insert(self, s) – השיטה תקבל כפרמטר אובייקט, תבדוק שהוא מסוג Shape ותוסיף אותו לרשימה במקום הנכון, כך שהרשימה תישאר ממוינת בסדר עולה לפי שטח הצורות.
- הוסיפו את __str__ – השיטה תחזיר מחרוזת שמכילה את פרטי כל הצורות שבמאגר. בתחילה תכיל המחרוזת את המשפט: Shapes in collection: ואז תכיל המחרוזת את פרטי כל הצורות, כל צורה בשורה נפרדת.
- biggest_perimeter_diff() – השיטה תחזיר את הפרש הגדול ביותר הקיים בין היקפיהן של שתי צורות כלשהן.
- same_area_as(s) – השיטה תקבל כפרמטר אובייקט מסוג Shape ותחזיר רשימה או מבנה נתונים אחר שמכיל את כל הצורות מהמאגר ששטחן שווה לשטחו של s.
- How_many_quadrilaterals() – השיטה תחזיר כמה מרובעים (מלבנים וריבועים) קיימים במאגר.

שאלה 2

א. Person

כתבו מחלקה בשם Person השומרת את התכונות הבאות:

1. שם פרטי,
2. שם משפחה,
3. כתובת,

4. מספר ת.ז.

- כל התכונות יהיו **פרטיות**. את השם הפרטי ואת ת.ז. לא ניתן יהיה לשנות. את שם המשפחה ואת הכתובת ניתן יהיה לשנות.
- יש לבדוק שהשם הפרטי והשם משפחה מורכבים רק מאותיות ולא מספרות. בנוסף, יש לבדוק שת.ז. מורכבת רק מספרות. אם הבדיקה נכשלת (בשמות או בת.ז.), יש להציב ערך דיפולטיבי. לשם פרטי: Avi, לשם משפחה: Cohen ולת"ז 300010000.

ב. Bank_account

כתבו מחלקה בשם Bank_account השומרת את התכונות הבאות:

1. מספר חשבון,
2. פרטי לקוח (יישמר כטיפוס Person),
3. יתרה בחשבון

ג. Student_account

כתבו מחלקה בשם Student_account היורשת מBank_account ושומרת תכונה נוספת: מוסד לימודים.

- לכל המחלקות יש לכתוב פונקציית __init__מתאימה המאתחלת את כל תכונות האובייקט.
- עבור מחלקת Bank_account יש לממש את __str__ אשר תאפשר להדפיס אובייקט מהמחלקה באופן רגיל ולקבל את הפלט:

Account: [account number]

Name: [first name] [last name]

Balance: [balance]

כאשר במקומות שבהם יש [] יופיעו הערכים המתאימים

- עבור חשבון סטודנט יודפס הפלט:

Student Account: [account number]

Name: [first name] [last name]

Balance: [balance]

כאשר במקומות שבהם יש [] יופיעו הערכים המתאימים

ד. ניהול החשבון

כתבו תכנית המאפשרת למשתמש לבצע פעולות על חשבון הבנק.

בתחילת התוכנית המשתמש יתבקש להכניס פרטים לצורך יצירת חשבון חדש: שם פרטי, שם משפחה, כתובת, מספר ת.ז, יתרה, סוג לקוח (רגיל או סטודנט), מוסד לימודים (אם מסוג סטודנט).
התוכנית תבחר אקראית מספר עבור מספר החשבון בטווח 0 עד 1000.
כעת, התוכנית תאפשר לבצע את הפעולות הבאות:

1. הפקדה לחשבון – המשתמש יכניס את הסכום אותו ירצה להוסיף לחשבון
2. משיכה – המשתמש יכניס את הסכום אותו ירצה למשוך. התוכנית תוודא שיש מספיק כסף בחשבון, אחרת תדפיס הודעה שהפעולה נכשלה.
3. הדפסה – התוכנית תזמן את פונקציית ההדפסה של המחלקה.
4. הדפסת היסטוריה – ייפתח לקריאה קובץ ההיסטוריה ויודפס התוכן שלו.
5. סיום.

- עבור פעולות הפקדה ומשיכה שהמשתמש מבצע מופחתת מהחשבון עמלה של 5 שקלים ללקוח רגיל ו-3 שקלים לסטודנט.
- כמו כן, התוכנית תשמור קובץ עם מידע על היסטוריית הפעולות שביצע המשתמש. קובץ זה ייפתח בתחילת התוכנית וישמור מידע על כל פעולות ההפקדה והמשיכה שנעשו בחשבון לאורך ריצת התוכנית.

תוכן הקובץ צריך להיראות פחות או יותר כך:

```
New account, balance: <100>
Deposit <20>, balance: <115>
Withdraw <15>, balance: <95>
Deposit <50>, balance: <140>
Deposit <40>, balance: <175>
Withdraw <30>, balance: <140>
```

הנחיות כלליות:

- הגישו הכול באותו קובץ.
- תעדו את הקוד בדרך המקובלת.
- בתיעוד בתחילת הקובץ רשמו את שם המגיש/ה/ים ואת ת"ז.