תרגיל בית 8

הנחיות כלליות:

- קראו **בעיון** את השאלות והקפידו שהתכניות שלכם פועלות בהתאם לנדרש.
- את התרגיל יש לפתור לבד!
- הקפידו על כללי ההגשה המפורסמים באתר. בפרט, יש להגיש את כל הפתרונות לשאלות יחד בקובץ ex8_012345678.py המצורף לתרגיל, לאחר החלפת הספרות 012345678 שבשם הקובץ במספר תעודת הזהות שלכם, כל 9 הספרות כולל ספרת ביקורת.
- אופן ביצוע התרגיל: שימו לב, בתרגיל זה עליכם להשלים את הקוד בקובץ המצורף.
- **אין לשנות את שמות המחלקות, הפונקציות, המתודות והמשתנים שכבר מופיעים בקובץ השלד של התרגיל.**
- **אין למחוק את ההערות שמופיעות בקובץ השלד.**
- היות ובדיקת התרגילים עשויה להיות אוטומטית, **יש להקפיד על פלטים מדויקים על פי הדוגמאות (עד לרמת הרווח).**
- בדיקה עצמית: כדי לוודא את נכונותן ואת עמידותן של התוכניות לקלטים שגויים, בכל שאלה הריצו את תוכניתכם עם מגוון קלטים שונים, אלה שהופיעו כדוגמאות בתרגיל וקלטים נוספים עליהם חשבתם (וודאו כי הפלט נכון).
- אין להשתמש בספריות חיצוניות (ובפונקציות שלהן) מעבר לאלו שסופקו יחד עם שלד התרגיל. כלומר, אין להשתמש בפקודת import. כל פונקציה שלא דורשת פקודה זו מותרת לשימוש (כלומר, זו פונקציה שהמתרגם (interpreter) מכיר ללא פקודה זו)
- **ניתן להניח שהקלט תקין בהתאם להערות המפורטות בהוראות כל שאלה, אלא אם מצויין אחרת.**
- מועד אחרון להגשה: כמפורסם באתר.

בתרגיל זה תכתבו תוכנה לניהול חדרי בית מלון ואורחיהם. התוכנה כוללת מחלקה המייצגת חדר במלון ומחלקה המייצגת את המלון. בנוסף מחלקה המייצגת מיניבר.

הערות כלליות חשובות:

- מומלץ לקרוא תחילה את כל התרגיל, להבין את הוראות התרגיל ולתכנן את המחלקות השונות והיחסים ביניהן בהתאם.
- במימוש המחלקות השונות יש להימנע משכפול קוד ככל שניתן!

- בכל השאלות, ניתן להוסיף שדות ומתודות נוספים לאלה הנדרשים בשאלה, אם הדבר מסייע לכם במימוש הפיתרון.

השלימו את המחלקות בקובץ התרגיל בהתאם לדרישות המופיעות בשאלות שלהלן.

שאלה 1

בשאלה זו יש להמיר את כל המחרוזות לlowercase. לדוגמא, יש להתייחס למחרוזות "ABc" ו- "abC" כשוות, כיוון שבפורמט lowercase שתיהן הופכות למחרוזת "abc".

א. ממשו את המחלקה Minibar. לכל מיניבר יש את התכונות (Attributes) הבאות:

שם	תיאור	סוג	הערות
drinks	המשקאות הנמצאים במיניבר.	מילון המכיל את השמות של המשקאות כמפתחות (keys) ואת מחירי המשקאות כערכים (values). המחירים הם מספרים אי שליליים מטיפוס int.	יכול להיות מילון ריק.
snacks	החטיפים הנמצאים במיניבר.	מילון המכיל את השמות של החטיפים כמפתחות (keys) ואת מחירי החטיפים כערכים (values). המחירים הם מספרים אי שליליים מטיפוס int.	יכול להיות מילון ריק.
bill	החשבון של המיניבר.	מספר שלם מטיפוס int.	יש לאתחל כ- 0.

התחילו במימוש בנאי המחלקה על פי החתימה הבאה:

```
def __init__(self, drinks, snacks):
```

ניתן להניח שכל הקלטים תקינים, כמתואר בטבלה. אין צורך לבצע בדיקות של טיפוסים או תקינות הקלט.

ב. ממשו את המתודות eat(self, snack) ו- drink(self,drink).

מתודה eat(self, snack) מקבלת שם של חטיף כמחרוזת ומסירה אותו מהמילון snacks. בנוסף, היא מוסיפה את המחיר של החטיף לחשבון (bill) של המיניבר. המתודה כשלעצמה לא מחזירה כלום.

מתודה drink(self,drink) מקבלת שם של משקה כמחרוזת ומסירה אותו מהמילון drinks. בנוסף, היא מוסיפה את המחיר של המשקה לחשבון (bill) של המיניבר. המתודה כשלעצמה לא מחזירה כלום.

במידה והערך (שם החטיף או המשקה) אינו נמצא במילון המתאים, יש להעלות (raise) שגיאה מסוג ValueError עם ההודעה הבאה:

'The snack {snack_name} was not found' עבור חטיף

ו- 'The drink {drink_name} was not found' עבור משקה

כאשר {snack_name} ו{drink_name} הם שמות החטיף והמשקה שהועברו כקלט בהתאמה.

ג. ממשו את המתודה __repr__(self) שמחזירה מחרוזת המתארת אובייקט מסוג Minibar, על פי הדוגמה:

קלט:

```
drinks1 = {'coke': 12, 'rum': 25}
snacks1 = {'m&m': 10, 'cake': 30}

m = Minibar(drinks1, snacks1)
print(m)
```

פלט:

Drinks – (coke, rum). Snacks – (m&m, cake). No bill yet.

במידה החשבון לא ריק (כלומר, bill>0) תודפס ההודעה על פי הדוגמה הבאה:

```
drinks1 = {'coke': 12, 'rum': 25}
snacks1 = {'m&m': 10, 'cake': 30}
m = Minibar(drinks1, snacks1)
m.drink('coke')
m.eat('m&m')
print(m)
```

פלט:

Drinks – (rum). Snacks – (cake). Bill – 22.

הערות:

- יש להדפיס את הערכים בפורמט הכיתוב כמובא בדוגמא **במדויק**.
- אין צורך לכלול \n לאחר התכונה האחרונה.

דוגמת הרצה עבור קלט שגוי:

קלט:

```
drinks1 = {'coke': 12, 'rum': 25}
snacks1 = {'m&m': 10, 'cake': 30}

m = Minibar(drinks1, snacks1)
print(m)

m.drink('beer')
print(m)
```

פלט:

שגיאה מסוג ValueError עם הכיתוב:

The drink beer was not found

שאלה 2

א. ממשו את המחלקה **Room**, אשר מייצגת חדר בבית מלון. לכל חדר יש את התכונות (Attributes) הבאות:

שם	תיאור	סוג	הערות
minibar	המיניבר הממוקם בחדר	minibar שמומש בשאלה 1	מיניבר המכיל משקאות וחטיפים
floor	מספר הקומה בה נמצא החדר	מספר שלם int גדול או שווה לאפס	

number	מספר החדר	מספר שלם int חיובי ממש	
guests	רשימת שמות אורחי החדר הנוכחיים	רשימה של מחרוזות. המחרוזות ברשימה מכילות אך ורק אותיות קטנות ורווחים	תיתכן רשימת אורחים ריקה (חדר ריק)
clean_level	רמת הניקיון של החדר בין 1 (מלוכלך) ל-15 (מבריק)	מספר שלם int בין 1 ל-15	
is_suite	האם החדר הוא סוויטה. True אם הוא סוויטה ו False אחרת	בוליאני	
satisfaction	רמת שביעות הרצון של אורחי החדר בין 0 (נמוכה ביותר) ל-1 (גבוהה ביותר)	מספר ממשי (float) בין 0.0 ל-1.0	* רמת שביעות הרצון בברירת מחדל (default) היא 0.5. * במידה והמשתמש הזין רמה שביעות רצון מסוג int, יש להמירה לfloat בעת שמירתה כתכונה באובייקט.

התחילו במימוש בנאי המחלקה על פי החתימה הבאה:

__init__(self, minibar, floor, number, guests, clean_level, is_suite, satisfaction=0.5)

ניתן להניח את תקינות סוגי וערכי הקלטים כפי שמתואר בטבלה למעלה, פרט למקרים המתוארים להלן, שבהם יש לוודא תקינות:

- יש להתחיל בבדיקה של הטיפוסים. במקרה ואחד הטיפוסים לא תקין, יש להעלות (raise) חריג (Exception) מסוג TypeError. יש להתייחס למקרים הבאים:
  - רמת הנקיון (clean_level) הינה מסוג int.
  - האם החדר סוויטה (is_suite) הינו מסוג bool.
  - רמת שביעות הרצון (satisfaction) הינה מסוג float. <u>שימו לב:</u> המשתמש יכול להעביר רמת שביעות רצון שלמה. בפרט, ייתכן שהיא תינתן כארגומנט מסוג int. למשל, בתור הארגומנט 1 אשר מייצג רמת שביעות רצון 1.0.
  - רשימת האורחים הינה רשימה של מחרוזות. במידה והמחרוזות מכילות אותיות גדולות (uppercase), תבוצע המרה שלהן לlowecase.
  - אין צורך לבדוק את טיפוס המיניבר, מספר החדר. ניתן להניח שהם תקינים.
- לאחר בדיקת הטיפוסים נעבור לבדיקת ערכים. במקרה ואחד מהערכים אינו תקין (<u>למרות שהטיפוס תקין</u>), יש להעלות חריג מסוג ValueError. יש להתייחס למקרים הבאים:
  - רמת הניקיון הינה בין 1 ל-15 (כולל קצוות הטווח).
  - רמת שביעות הרצון הינה בין 0.0 ל-1.0 (כולל קצוות הטווח).

<u>הערות:</u>

- ניתן לבחור כרצונכם את ההודעה שבכל TypeError ובכל ValueError.
- במקרה שיש מספר ארגומנטים שונים שאינם תקינים, אין חשיבות לזהות הארגומנט הבעייתי שבעקבותיו יועלה החריג.

ב. ממשו את המתודה __repr__(self) שמחזירה מחרוזת המתארת אובייקט מסוג Room, על פי דוגמת הפלט שבהמשך.

- המחרוזת תכלול שורה נפרדת עבור כל אחת מתכונות החדר בפורמט "שם התכונה:<רווח אחד>ערך התכונה". שם התכונה יתחיל באות גדולה וקו תחתי (underscore) יוחלף ברווח. ערך התכונה של minibar הוא לפי המימוש שכתבתם עבור __repr__ של Minibar.
- סדר הופעת התכונות יהיה זהה לסדר הופעתן בטבלה למעלה.
- עבור התכונה guests, ערך התכונה יהיה שמות רשימת האורחים **בסדר לקסיקוגרפי עולה**, כאשר הינם מופרדים ב<<פסיק><רווח אחד>>. **אם רשימת האורחים ריקה, ערך התכונה יהיה המילה empty**.
- מכיוון שתכונת satisfaction הינה מסוג float, היא תודפס למסך עד רמת דיוק של 2 ספרות לאחר הנקודה (עשו שימוש בround. בדוגמת ההרצאה ניתן לראות את הפורמט המבוקש)
- **אין לכלול \n לאחר התכונה האחרונה**

דוגמאות הרצה:

'm' מייצג את מחלקת המיניבר משאלה 1 בכל דוגמאות ההרצה

קלט:

```
m = Minibar({'coke': 10, 'lemonade': 7}, {'bamba': 8,
'mars': 12})
print(Room(m, 12, 101, ["Shir", "Ronen"], 6, True))
```

פלט:

Minibar: Drinks - (coke, lemonade). Snacks – (bamba, mars). No bill yet.

Floor: 12.

Number: 101.

Guests: ronen, shir.

Clean level: 6.

Is suite: True.

Satisfaction: 0.5.

ג. הוסיפו את מימוש המתודות הבאות למחלקה Room:

חתימת המתודה	תיאור
is_occupied(self)	מחזירה True אם החדר תפוס, כלומר, יש בחדר אורחים, ואחרת – False.
clean(self)	מבצעת פעולת ניקיון של החדר, שאיכותה ומידת השפעתה על רמת הניקיון עולה עם דרגת החדר (סוויטה או לא סוויטה). פעולת ניקיון אחת מעלה את רמת הניקיון של החדר $clean\_level$ ל-$\min(15, clean\_level + 1 + \text{is\_suite})$, כאשר is_suite מומר ל0 במידה והוא False ול1 במידה והוא True.
better_than(self, other)	משווה בין רמתם של שני חדרים, ומחזירה True אם self הוא חדר "יותר טוב" מהחדר other, ואחרת – False. • החדר self נחשב ל"יותר טוב" מהחדר other אם $(self.is\_suite, self.clean\_level, self.floor) > (other.is\_suite, other.clean\_level, other.floor)$ כאשר הסדר $>$ **הוא כפי שהוא מוגדר על tuples.** • אם other אינו מסוג Room, יש להעלות חריג מסוג TypeError עם ההודעה: "Other must be an instance of Room".
check_in(self, guests)	מכניסה אורחים לחדר. • המתודה תכניס את רשימת האורחים guests לחדר self אם הוא ריק, ובנוסף, תאתחל את רמת שביעות הרצון ל-0.5. • אם החדר תפוס, לא ניתן לבצע פעולה זו, ולכן המתודה תעלה חריג מסוג RoomError עם ההודעה: "Can't check-in new guests to an occupied room". • ניתן להניח ש-guests היא רשימת מחרוזות <u>לא ריקה</u> עם שמות חוקיים (כלומר, שכוללים רק אותיות אנגליות ורווחים), ואותיות שיכולות להיות בפורמט case גדול או קטן. • <u>הערה:</u> על המתודה להכניס את שמות האורחים לשדה self.guests בפורמט lowercase בלבד.
check_out(self)	מבצעת צ'ק-אאוט, כלומר, מפנה את האורחים השוהים כעת בחדר. • אם רשימת האורחים של החדר (self.guests) <u>אינה</u> ריקה, אז הפינוי יהפוך אותה לרשימה ריקה. • אחרת, אם self.guests ריקה, יש להעלות חריג מסוג RoomError עם ההודעה: "Cannot check-out an empty room".
move_to(self, other)	מעבירה את אורחי החדר self לחדר other אם הוא ריק. • אם החדר self ריק, אין אורחים להעביר, ולכן המתודה תעלה חריג מסוג RoomError עם ההודעה: "Cannot move guests from an empty room". • אם החדר other תפוס, לא ניתן לבצע את העברת האורחים, ולכן המתודה תעלה חריג מסוג RoomError עם ההודעה: "Can't move guests into an occupied room". • אם self ריק וגם other תפוס, אז יש להעלות את החריג אודות self.

	<u>פעולת העברת האורחים מהחדר self כוללת את הצעדים הבאים:</u> א. העברת שמות האורחים מ-self לרשימה המתאימה ב-other. ב. אם other הוא חדר "יותר טוב" מ-self (כפי שהוגדר למעלה במתודה better_than), אז רמת שביעות הרצון של האורחים שעברו ל-other משתפרת, ולכן כעת $other.satisfaction = \min(1.0, self.satisfaction + 0.05)$. <u>אחרת</u>, רמת שביעות הרצון ב-other הופכת לזו שב-self בעת ביצוע ההעברה. ג. לבסוף, מחיקת איברי רשימת האורחים של self, כך שהיא הופכת לריקה. • ניתן להניח ש-other הוא אובייקט תקין מסוג Room. • ניתן להניח ש-self ו-other מייצגים חדרים שונים.

<u>הערה:</u> ניתן בפייתון להעלות חריגים (Exceptions) מיוחדים שאותם הגדרנו בעצמנו לפי הצרכים שלנו בתוכנית מסויימת. דוגמה לכך היא **סוג החריג RoomError, שמוגדר בקובץ התרגיל, וניתן להעלות אותו עם raise** כפי שעשינו עד כה עם כל חריג סטנדרטי.

<u>דוגמת הרצה (וודאו שהנכם מבינים היטב את מהלכה):</u>

```
>>> r1 = Room(m, 2, 23, ["Dana", "Ron"], 5, False)
>>> r_better = Room(m, 6, 57, [], 4, True)
>>> r_better.better_than(r1)
True
>>> r_better.check_in(["Amir"])
>>> r_better.clean()
>>> r_better.clean_level
6
>>> r1.check_in(["Avi", "Hadar"])
Traceback (most recent call last):
…
RoomError: Cannot check-in new guests to an occupied room
>>> r1.is_occupied()
True
>>> r1.check_out() ## note: None is returned, and so nothing is printed
>>> r1.is_occupied()
False
>>> r_better.move_to(r1)
>>> r1.satisfaction
0.5
>>> r1.guests
['amir']
>>> r1.move_to(r_better)
>>> r1.is_occupied()
False
>>> r_better.satisfaction
0.55
>>> r_better.guests
['amir']
```

שאלה 3

כעת נממש את המחלקה Hotel המייצגת בית מלון.

א. ממשו את הבנאי __init__(self, name, rooms) המקבל מחרוזת name המציינת את שם המלון, ורשימת חדרים rooms, שאין לבדוק את תקינותם.

הערות:

- ניתן להניח ש-name הינו מחרוזת המייצגת שם מלון חוקי, שמכילה רווחים, ספרות ואותיות אנגליות בלבד (ב- uppercase ו\או ב-lowercase).
- ניתן להניח שהרשימה rooms לא ריקה, כאשר כל איבריה הינם חדרים תקינים (מסוג Room) ושונים זה מזה (כלומר, אין אובייקט חדר המופיע פעמיים, ואין אובייקטים שונים עם אותו מספר חדר וגם אותו מספר קומה). ניתן להניח ששמות האורחים שונים זה מזה גם באותו החדר וגם בחדרים השונים.
- הרשימה rooms יכולה להכיל חדרים תפוסים ו\או פנויים.
- ניתן לשמור את החדרים כשדה של אובייקט המלון תוך שימוש בכל מבנה נתונים בפייתון שהנכם רואים לנכון. בפרט, אין הכרח להשתמש ברשימה במימוש הפנימי.
- ניתן להוסיף שדות ו/או מתודות נוספים שיכולים לסייע לכם במימוש המחלקה.

ב. ממשו את המתודה __repr__(self) אשר מחזירה מחרוזת שמתארת את אובייקט המלון על פי הפורמט הבא:

```
<self.name><רווח בודד>Hotel has: <רווח בודד><number of occupied rooms>/<Total
number of rooms (occupied and not occupied together)><רווח בודד>occupied
rooms.
```

דוגמת הרצה:

```
>>> h = Hotel("Best",[Room(m, 15, 140, [], 5, 1), Room(m, 1, 2, ["Liat"], 7,
3)])
>>> h
Best Hotel has: 1/2 occupied rooms.
```

הערה: בכתיבת הקוד של __repr__, חישוב מספר החדרים יכול להתבצע בכל אופן שהנכם רואים לנכון.

בפרט, ניתן להשתמש במתודות עזר.

ג. על אובייקט מלון לתמוך במתודות הבאות:

שם וחתימה	תיאור

check_in(self, guests, is_suite)	מנסה לבצע צ'ק-אין לרשימת שמות האורחים guests (רשימת מחרוזות) לחדר אחד (כלשהו) מחדרי המלון, שהוא סוויטה אם is_suite הוא True ולא סוויטה אם הוא False. • במידה ונמצא חדר פנוי ומתאים בדרגתו (סוויטה או לא סוויטה), המתודה תבצע ל-guests צ'ק-אין אליו, ותחזיר את (אובייקט) החדר שנמצא, ואחרת – None. • ניתן להניח ששמות האורחים ב-guests לא מתנגשים עם שמות אורחים אחרים שכבר שוהים במלון. • ניתן להניח ש-guests היא רשימת מחרוזות לא ריקה עם שמות חוקיים (כלומר, שכוללים רק אותיות אנגליות ורווחים), ואותיות שיכולות להיות בפורמט case גדול או קטן.
check_out(self, guest)	מנסה לבצע צ'ק–אאוט לאורח בשם guest (מחרוזת) יחד עם האורחים הנוספים השוהים עמו בחדר (אם יש כאלה). • במידה ונמצא החדר בו הוא שוהה, יש לבצע את הצ'ק-אאוט בהצלחה, ולהחזיר את אובייקט החדר בו שהה האורח. • אחרת – לא ניתן לבצע את הצ'ק אאוט, ויש להחזיר None. • בחיפוש החדר בו guest שוהה יש להתעלם מפורמט ה-case. למשל, נתייחס ל-UZI כ-uzi וכ-Uzi.
upgrade(self, guest)	מנסה לבצע "שדרוג" חדר לאורח בשם guest (מחרוזת), במידה ו-guest שוהה בחדר בבית המלון, ויש חדר פנוי שניתן "לשדרג" אליו. • פעולת ה"שדרוג" כוללת את העברת האורח יחד עם האורחים הנוספים השוהים עמו בחדר (אם יש כאלה) לחדר אחר במלון שהינו פנוי ו"טוב יותר" (כפי שהוגדר בשאלה 1). • במידה וה"שדרוג" מתבצע בהצלחה, יש להחזיר את (אובייקט) החדר שאליו הועברו האורחים. • אחרת, אם האורח לא שוהה במלון או פעולת ה"שדרוג" לא ניתנת לביצוע, יש להחזיר None. • בחיפוש החדר בו guest שוהה יש להתעלם אותיות או גדולות. • במידה וישנם מספר חדרים הניתנים לשדרוג יש לשדרג לאחד מהם

הערות:

- על כל המתודות תמיד לסיים לרוץ ללא העלאת חריגים כלשהם מסוג RoomError
- ניתן להניח את תקינות (סוג וערך) הארגומנטים בכל המתודות. בפרט, ניתן להניח שכל מחרוזת בקלט מייצגת שם תקין של אורח ב-case גדול ו/או קטן.

דוגמת הרצה:

- קוד הרצה לדוגמה מצורף לשלד התרגיל בפונקציה **test_hotel**.
- הקובץ **test_hotel_output.txt (המצורף לתרגיל)** כולל את ההדפסות שנוצרות במהלך ריצת הפונקציה הנ"ל, ונועד לאפשר לכם לבדוק שהדפסות המימוש שלכם זהות.

- שימו לב, ישנן פקודות להן יתכן יותר מערך אחד תקין (למשל ישנן מספר אפשרויות לשדרוג החדר של liat), במקרה כזה יתכן שיתקבל פלט השונה מזה שבדוגמה (תלוי מימוש).

שאלה 4

במהלך מסעותיו, גילה ארכיאולוג תחריט רומאי ועליו נוסחא לכוחו האינסופי של האל זאוס (הנקרא גם Reggie). על מנת לפענח את הנוסחא, נדרש היה לתרגמה מכתיב רומי. לצורך כך, החליט הארכיאולוג לממש מחלקה התומכת בייצוג דואלי של מספרים בכתיב רגיל ובכתיב רומי, ובפעולות אריתמטיות בסיסיות על מספרים אלה.

בכתיב רומי, הספרות מיוצגים על ידי צירופי אותיות מהאלפבית הלטיני. לקריאה נוספת:
https://en.wikipedia.org/wiki/Roman_numerals

הספרות הרומיות הבסיסיות הן:

'I': 1, 'V': 5, 'X': 10, 'L': 50, 'C': 100, 'D': 500, 'M': 1000

1. ממשו את המחלקה Roman אשר מאותחלת באמצעות מספר (int) אשר מכיל את הערך המספרי שנרצה לייצג באמצעות ספרות רומיות, או באמצעות מחרוזת אשר מכילה מספר הכתוב בספרות רומיות.
התחילו במימוש של הבנאי (constructor):

def __init__(self, input_value):

הארגומנט input_value מכיל מספר שלם או מחרוזת המכילה מספר הכתוב בספרות רומיות. הבנאי מאתחל שלוש תכונות (attributes):

int_value – שומרת את הערך המספרי בתור מספר שלם (int)
roman_value – שומרת את הייצוג של המספר בספרות רומיות (str)
is_neg – שומרת ערך True אם המספר הוא שלילי (קטן מ 0), ו False אחרת (bool).

שלד התרגיל מכיל מימוש של שתי מתודות: get_roman_from_int ו get_int_from_roman על מנת להמיר מספר שלם ליצוג בספרות רומיות, ולהיפך.

הניחו כי הקלט חוקי ויכול להכיל רק מספר שלם או מחרוזת ב upper-case המכילה יצוג תקין באותיות רומיות. בנוסף, הניחו כי הקלט אינו 0, שכן לא ניתן לייצג 0 בספרות רומיות.

2. כעת נרצה לממש את המתודות __str__ ו __repr__ על מנת לאפשר ייצוג "יפה" של אובייקטים מסוג Roman. הפורמט המבוקש הוא:

- The method of __ str__:
  >>> print(Roman('V'))
  V
  >>> print(Roman(50))
  L
  >>> print(Roman('XL'))
  XL
- The method of __repr__
  >>>Roman('L')
  int: 50; Roman Numeral: 'L'

```
>>> Roman(50)
int: 50; Roman Numeral: 'L'
>>>Roman('XL')
int: 40; Roman Numeral: 'XL'
>>> Roman('-V')
int: -5; Roman Numeral:'-V'
>>> Roman(-5)
int -5; Roman Numeral: '-V'
```

שימו לב לייצוג של מספרים שליליים כפי שהוא מוצג בשתי הדוגמאות האחרונות.

3. נרצה לתמוך בפעולה שלילה (negation). לצורך כך, עליכם לממש את השירות __neg__.
דוגמא:

```
>>>-Roman(50)
-L
>>>-Roman('V')
-V
```

4. כעת נרצה לממש את המתודות המתאימות לפעולות של סכימה (המתודה __add__) והשוואה (המתודות __lt__ ו __gt__) של ספרות רומיות (עליכם לתמוך אך ורק בפעולות שבהן ה Roman מופיע מצד שמאל לאופרטור). על הפעולות לתמוך בסכימה\השוואה של Roman עם אובייקט מסוג Roman או int.

```
Method: __add__
>>> Roman(5) + Roman(6)
XI
>>> Roman('V') + 6
XI
>>> Roman(5) + Roman(-6)
-I
>>> Roman('V') + (-6)
-I
```

```
Method: __lt__ and __gt__
>>> Roman(5) < Roman(6)
True
>>> Roman(6) < Roman(5)
False
>>> Roman(5) > 6
False
>>> 5 < Roman(6)
True
```

שימו לב למקרה המיוחד שבו תוצאת החיבור נותנת 0. מכיוון שאין יצוג ל 0 בספרות רומיות, עליכם לזרוק ValueError עם מלל לבחירתכם אם תוצאת הסכימה היא 0.

5. מכיוון שאנחנו עובדים רק עם מספרים שלמים, נרצה לתמוך בפעולה של חלוקה ללא שארית (האופרטור // ולא האופרטור /). גם במקרה זה, נתמוך רק בפעולות שבהן Roman מופיע בצידו השמאלי של האופרטור. תוצאת החלוקה היא Roman.
מקרים מיוחדים:
   א. הניחו כי לא תתבצע חלוקה ב 0.
   ב. עליכם לזרוק ValueError עם מלל לבחירתכם אם תוצאת החלוקה ללא שארית היא 0.
פעולת החילוק ללא שארית תעבוד באופן הבא:

```
>>> Roman(6) // Roman(5)
```

```
int: 1; Roman Numeral: 'I'
>>> Roman('VI') // 5
int: 1; Roman Numeral: 'I'
>>> Roman(6) // Roman('-V')
int: -2; Roman Numeral: '-II'
>>> Roman(6) // -5
int: -2; Roman Numeral: '-II'
>>> Roman(2) // Roman(3)    # this should raise a ValueError!
```

פלט רצוי מהרצת שלד התרגיל:

```
 ====Q1: Basic tests/output====
II
II
int: 2; Roman Numeral: 'II'
====
-IV
====
V
V
int: 5; Roman Numeral: 'V'
int: -2; Roman Numeral: '-II'
```

הבהרות:

- שימו לב שהעיגול של תוצאת החילוק הוא כלפי מטה ולכן הערך של התוצאה בדוגמה האחרונה הוא אכן 2- ולא 1- בגלל ש2- יותר קטן, בדומה להתנהגות של // על int:

```
>>> 6//-5     #  This is the behavior with the type int
-2
>>> Roman(6) // -5   # This is the behavior with type Roman implementation
int: -2; Roman Numeral: '-II'
```

- בפעולות **חיבור וחילוק** ניתן להניח כי המימוש בRoman **חייב להופיע בצידו השמאלי של הביטוי.** למעשה, תנאי זה הוא חובה על מנת להפעיל את הפונק׳ שהתבקשתם לממש. מצד ימין יכול להופיע Roman או ,int כפי שמוצג בדוגמאות.
- בפעולות **השוואה** (מגדול מ.. <, קטן מ... >) ייצוג הroman או הint יכול להופיע **בכל אחד** מצידי האופרטור
- את הערכים המספריים (int) שהועברו בבונה יש לשמור כמו שהם ובנוסף לאתחל את התכונה של השליליות

- לא צריך לבדוק תרחיש של השוואת Roman עם 0.
- במתודה __neg__ יש להחזיר **אובייקט Roman חדש** (כפי שקורה באובייקטים מסוג int)
- דרך נוספת לבדוק את המימושים שלכם היא לשמור אותו בתור משנה ואז להפעיל את המימושים את מימושי הוואז לנסות להדפיס אותו בשני האופנים שראיתם.
- הפלט של פונקציות __str__ ו__repr__ הוא **בדיוק** כפי שרשום בתרגיל (עם גרש סביב הייצוג הרומי)
- אין להשתמש בחבילות פייתון מעבר לאלו המיובאות לכם בשלד התרגיל

בהצלחה!