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).
- **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 תכלול מחלקות המייצגות את הסוגים השונים של חדרי המלון, וכן מחלקה המייצגת את המלון.

הערות כלליות חשובות:

- מומלץ לקרוא תחילה את כל התרגיל, להבין את הוראות התרגיל ולתכנן את המחלקות השונות והיחסים ביניהם בהתאם.
- במימוש המחלקות השונות יש להשתמש בירושה על מנת להימנע משכפול קוד ככל שניתן!
- השוואת מחרוזות בכל התרגיל תיעשה לפי lowercase בלבד. לדוגמה, יש להתייחס למחרוזות "ABc" ו-"abC" כשוות, כיוון שבפורמט lowercase שתיהן הופכות למחרוזת "abc".
- 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.

<u>שאלה 1</u>

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

שם	תיאור	סוג	הערות
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 (גדולות או קטנות) ורווחים	תיתכן רשימת אורחים ריקה (חדר ריק)
clean_level	רמת הניקיון של החדר בין 1 (מלוכלך) ל-10 (מבריק)	מספר שלם int בין 1 ל-10	
rank	דרגת החדר המייצגת את יוקרתיות החדר. ישנן 3 דרגות: 1 (בסיסית -Basic), 2 (סטנדרטית – Standard), או 3 (יוקרתית - Luxury)	מספר שלם int בין 1 ל-3	
satisfaction	רמת שביעות הרצון של אורחי החדר בין 1.0 (נמוכה ביותר) ל-5.0 (גבוהה ביותר)	מספר ממשי (float) בין 1.0 ל-5.0	* רמת שביעות הרצון בברירת מחדל (default) היא 1.0. * תיתכן רמת שביעות רצון שאינה שלמה, למשל: 4.5. * במידה והמשתמש הזין רמה שביעות רצון מסוג int, יש להמירה לfloat בעת שמירת כתכונה באובייקט.

התחילו במימוש בנאי המחלקה על פי החתימה הבאה:

__init__(self, floor, number, guests, clean_level, rank, satisfaction=1.0)

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 אינה מסוג int.
  - דרגת החדר אינה מסוג int.

- רמת שביעות הרצון אינה מספר שלם מסוג int וגם אינה מספר ממשי מסוג float. שימו לב: המשתמש יכול לבנות חדר עם רמת שביעות רצון שלמה. בפרט, ייתכן שהיא תינתן כארגומנט מסוג int. למשל, בתור הארגומנט 4 אשר מייצג רמת שביעות רצון 4.0.

- לאחר בדיקת הטיפוסים נעבור לבדיקת ערכים. במקרה ואחד מהערכים אינו תקין (למרות שהטיפוס תקין), יש להעלות חריג מסוג ValueError. יש להתייחס למקרים הבאים:
  - רמת הניקיון אינה בין 1 ל-10 (כולל קצוות הטווח).
  - דרגת החדר אינה בין 1 ל-3 (כולל קצוות הטווח).
  - רמת שביעות הרצון אינה בין 1 ל-5 (כולל קצוות הטווח).

הערות:

- ניתן לבחור כרצונכם את ההודעה שבכל TypeError ובכל ValueError.
- במקרה שיש מספר ארגומנטים שונים שאינם תקינים, אין חשיבות מהו הארגומנט הבעייתי שבעקבותיו יועלה החריג.
- יש להמיר את כל האותיות בשמות האורחים ברשימה שמתקבלת כקלט ביצירת אובייקט חדש לאותיות קטנות (lowercase). אופן מימוש זה יקל על מימוש הסעיפים שבהמשך השאלה.

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".
- סדר הופעת התכונות יהיה זהה לסדר הופעתן בטבלה למעלה.
- עבור התכונה guests, ערך התכונה יהיה שמות רשימת האורחים ב-lowercase (בסדר כלשהו), כאשר הינם מופרדים ב<<פסיק><רווח אחד>>. אם רשימת האורחים ריקה, ערך התכונה יהיה המילה empty (ראו דוגמה שנייה בדוגמאות הרצה).
- מכוון שתכונת satisfaction הינה מסוג float, היא תודפס למסך עד רמת דיוק של ספרה אחת אחרי הנקודה (עשו שימוש בround, ראו דוגמא)
- אין צורך לכלול \n לאחר התכונה האחרונה

דוגמאות הרצה:

```
>>> guests = ["Roni", "Danny"]
>>> r1 = Room(3, 21, guests, 5, 1)
>>> r1
floor: 3
number: 21
guests: roni, danny
clean_level: 5
rank: 1
satisfaction: 1.0
>>> r2 = Room(4, 28, [], 5, 1)
>>> r2
floor: 4
number: 28
```

```
guests: empty
clean_level: 5
rank: 1
satisfaction: 1.0
```

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.
can_clean(self)	מדווחת האם ניתן לנקות את החדר ע"י החזרת ערך בוליאני. • כברירת מחדל, נגדיר מתודה זו כך שתמיד תחזיר True • למרות שזה עשוי להיראות מוזר שהמטודה מחזירה תמיד True, שימו לב שהמימוש במחלקות היורשות (ראו שאלות הבאות) עשוי להיות שונה. המשמעות היא שחדר מסוג ROOM תמיד ניתן לנקות (אך לא כך בהכרח עבור מחלקה היורשת מROOM)
clean(self)	מבצעת פעולת ניקיון של החדר, שאיכותה ומידת השפעתה על רמת הניקיון עולה עם דרגת החדר. • במידה וניתן לנקות את החדר, מנקה את החדר ע"י פעולת ניקיון אחת שמעלה את רמת הניקיון של החדר $clean\_level$ ל- $\min(10, clean\_level + rank)$, כאשר $rank$ הוא דרגת החדר. • אם לא ניתן לנקות את החדר, יש להעלות חריג מסוג RoomError (ראו הערה בהמשך) עם ההודעה: "Room cannot be cleaned".
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 $(self.rank, self.floor, self.clean\_level) > (other.rank, other.floor, other.clean\_level)$ כאשר הסדר $>$ **הוא כפי שהוא מוגדר על tuples.** • אם other אינו מסוג Room וגם אינו מסוג היורש מ-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-1.0. • אם החדר תפוס, לא ניתן לבצע זאת, ולכן המתודה תעלה חריג מסוג RoomError עם ההודעה: "Cannot 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 כולל את הפיכתה 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 self ריק, אין אורחים להעביר, ולכן המתודה תעלה חריג מסוג RoomError עם ההודעה: "Cannot move guests from an empty room". • אם other תפוס, לא ניתן לבצע את העברת האורחים, ולכן המתודה תעלה חריג מסוג RoomError עם ההודעה: "Cannot 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 $other.satisfaction = \min(5.0, self.satisfaction + 1.0)$. <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 או היורש מ-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(2, 23, ["Dana", "Ron"], 5, 2)
>>> r_better = Room(6, 57, [], 4, 3)
>>> r_better.better_than(r1)
True
>>> r_better.check_in(["Amir"])
>>> r_better.clean()
>>> r_better.clean_level
7
>>> 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
1.0
>>> r1.guests
['amir']
>>> r1.move_to(r_better)
>>> r1.is_occupied()
False
>>> r_better.satisfaction
2.0
>>> r_better.guests
['amir']
```

שאלה 2

כעת נממש את המחלקות **BudgetRoom**, ו-**LegacyRoom** בהנחה שיש כבר בידינו את המחלקה Room שהגדרנו בסעיף הקודם. עליהן לייצג סוגי חדרי מלון אשר מכילים את כל התכונות והפעולות של חדר מלון Room, אך עם שינויים ותוספות הייחודיים לכל סוג חדר, כמפורט בסעיפים שלהלן.

א. בנאים (ראו דוגמת הרצה בסעיף ב'):

א.1. ממשו את בנאי המחלקה BudgetRoom (המייצגת חדר חֶסֶך) על פי החתימה:

__init__(self, floor, number, guests, clean_level, rank=1, satisfaction=1.0, clean_stock=0)

- שימו לב, Rank מקבל ערך ברירת מחדל 1.
- לחדר חסך נוספת התכונה **clean_stock**:
  - מייצגת את מספר פעולות הנקיון הכולל שניתן לבצע בחדר במהלך שהות האורחים בחדר.
  - מסוג מספר שלם int גדול או שווה לאפס.
  - מקבלת ערך ברירת מחדל 0.
  - ניתן להניח את תקינות הסוג והערך של התכונה, ואין לבדוק את תקינותם.

א.2. ממשו את בנאי המחלקה LegacyRoom (המייצגת חדר "שרות מלא") על פי החתימה:

__init__(self, floor, number, guests, clean_level, rank=2, satisfaction=1.0, minibar_drinks=2, minibar_snacks=2)

- שימו לב, rank מקבל ערך ברירת מחדל 2.
- לחדר LegacyRoom נוספות התכונות **minibar_drinks** ו-**minibar_snacks**:
  - מייצגות את מספר המשקאות והחטיפים שזמינים במיני-בר עם כניסת האורחים לחדר, בהתאמה.
  - שתיהן מסוג מספר שלם int גדול או שווה לאפס.

- שתיהן מקבלות ערך ברירת מחדל 2.
- ניתן להניח את תקינות הסוג והערך של שתיהן, ואין צורך לבדוק את תקינותם.

הערה:

- כחלק מכתיבת קוד הבנאי של כל מחלקה, יש לקרוא לבנאי של Room עם הארגומנטים המתאימים.

ב. המתודה __repr__: מחזירה מחרוזת המתארת את האובייקט באופן הדומה לזה של מחלקת האב Room, אך בתוספת תיאורי השדות הייחודיים לסוג החדר הספציפי, בהתאם לדוגמאות הבאות:

```
>>> br1 = BudgetRoom(1, 12, ["Loren", "Or"], 5)
>>> br1
floor: 1
number: 12
guests: loren, or
clean_level: 5
rank: 1
satisfaction: 1.0
type: BudgetRoom
clean_stock: 0
>>> lr1 = LegacyRoom(5, 94, ["Ronen", "Dror", "Liat", "Smadar"], 5)
>>> lr1
floor: 5
number: 94
guests: ronen, dror, liat, smadar
clean_level: 5
rank: 2
satisfaction: 1.0
type: LegacyRoom
minibar_drinks: 2
minibar_snacks: 2
```

הערות:

- לפני תיאור השדות הייחודיים, על כל מחלקה להוסיף את השורה: "שם המחלקה<רווח אחד>:type".
- על תיאורי השדות הייחודיים להופיע לפי סדר הופעת השדות בסעיף א' למעלה.
- כחלק מכתיבת קוד מתודה זו, יש להשתמש במתודה __repr__ של המחלקה Room.

ג. פעולות: ממשו את המתודות הנתמכות ע"י BudgetRoom ו-LegacyRoom, תוך הקפדה על:

- כתיבה יעילה - אין לכתוב פעמיים את אותו המימוש של מתודה, אלא להשתמש בהורשה.
- תכנון נכון - ייתכן ותצטרכו לממש את אותה המתודה יותר מפעם אחת על-מנת להתאים אותה לכל מחלקה בהתאם לדרישות. עם זאת, יש לשים לב שהנכם לא משכפלים קוד. בפרט, יש לשים לב אם ומתי ניתן לקרוא למתודה שכבר מימשתם במחלקת האב בתוך המימושים השונים שלה במחלקות היורשות.

ג.1. על המחלקה BudgetRoom לתמוך במתודות הבאות:

שם וחתימה	תיאור
is_occupied(self)	זהה לתיאור של מתודה זו במחלקה Room.
can_clean(self)	מדווחת האם ניתן לנקות את החדר ע"י החזרת ערך בוליאני. • חדר חסך ניתן לניקוי אם מספר פעולות הנקיון שניתן לבצע בחדר (clean_stock) חיובי ממש.
clean(self)	פועלת כפי שהוגדר עבור מתודה זו במחלקה Room, ובנוסף, במידה וניתן לנקות את החדר, מקטינה ב-1 את מספר פעולות הנקיון שניתן לבצע בחדר. • במימוש המתודה יש להשתמש בקריאה למתודה המקבילה במחלקת האב (Room).
better_than(self, other)	זהה לתיאור של מתודה זו במחלקה Room.
check_in(self, guests)	פועלת כפי שהוגדר עבור מתודה זו במחלקה Room, ובנוסף, אם החדר self ריק, מאתחלת את $self.clean\_stock$ לערך 0. • במימוש המתודה יש להשתמש בקריאה למתודה המקבילה במחלקת האב. • 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)	זהה לתיאור של מתודה זו במחלקה Room.
move_to(self, other)	פועלת כפי שהוגדר עבור מתודה זו במחלקה Room, ובנוסף, אם פעולת העברת האורחים יכולה להתבצע (כלומר, אם החדר self לא ריק, והחדר other ריק), ובמידה ו-other הוא אובייקט של BudgetRoom, משנה את הערך של $other.clean\_stock$ כך שיהיה זהה לזה של $self.clean\_stock$. • במימוש המתודה יש להשתמש בקריאה למתודה המקבילה במחלקת האב.
grant_clean(self)	מעניקה פעולת נקיון במתנה לאורחי החדר ע"י הגדלת clean_stock ב-1. • כתוצאה מכך, רמת שביעות הרצון של אורחי החדר $satisfaction$ משתפרת ל- $\min(5.0, satisfaction + 0.5)$. • אם החדר ריק, לא ניתן להעניק את המתנה, ולכן יש להעלות חריג מסוג RoomError עם ההודעה: "Cannot grant an empty room".
grant_snack(self)	מעניקה לאורחי החדר חטיף "על חשבון הבית". • החטיף משפר את רמת שביעות הרצון של אורחי החדר $satisfaction$ ל-$\min(5.0, satisfaction + 0.8)$. • עם זאת, החטיף מלכלך, וגם מדרדר את רמת הניקיון $clean\_level$ ל-$\max(1, clean\_level - 1)$. • אם החדר ריק, לא ניתן להעניק את המתנה, ולכן יש להעלות חריג מסוג RoomError עם ההודעה: "Cannot grant an empty room".

דוגמת הרצה (וודאו שהנכם מבינים היטב את מהלכה):

```
>>> br1 = BudgetRoom(1, 12, ["loren", "or"], 5)
>>> br1.clean_stock
0
>>> br1.satisfaction
1.0
```

```
>>> br1.clean()
Traceback (most recent call last):
  …
RoomError: Room cannot be cleaned
>>> br1.grant_clean()
>>> br1.clean_stock
1
>>> br1.satisfaction
1.5
>>> br1.clean()
>>> br1.clean_stock
0
>>> br1.clean_level
6
>>> br1.grant_snack()
>>> br1.clean_level
5
>>> br1.satisfaction
2.3
>>> br2 = BudgetRoom(2, 23, [], 6)
>>> br2.better_than(br1)
True
>>> br1.grant_clean()
>>> br1.move_to(br2)
>>> br2
floor: 2
number: 23
guests: loren, or
clean_level: 6
rank: 1
satisfaction: 3.8
type: BudgetRoom
clean_stock: 1
```

ג.2. על המחלקה LegacyRoom לתמוך במתודות הבאות:

שם וחתימה	תיאור
is_occupied(self)	זהה לתיאור של מתודה זו במחלקה Room.
can_clean(self)	מדווחת האם ניתן לנקות את החדר ע"י החזרת ערך בוליאני. • חדר LegacyRoom תמיד ניתן לניקוי.
clean(self)	זהה לתיאור של מתודה זו במחלקה Room.
better_than(self, other)	זהה לתיאור של מתודה זו במחלקה Room.
check_in(self, guests)	פועלת כפי שהוגדר עבור מתודה זו במחלקה Room, ובנוסף, אם החדר self ריק, מאתחלת את $self.minibar\_drinks$ ו- $self.minibar\_snacks$ ל-2. • במימוש המתודה יש להשתמש בקריאה למתודה המקבילה במחלקת האב. • 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)	זהה לתיאור של מתודה זו במחלקה Room.
move_to(self, other)	זהה לתיאור של מתודה זו במחלקה Room.

add_drinks(self, quantity)	מגדילה את מספר המשקאות שבמיני-בר ב-quantity, שהינו מספר שלם int חיובי ממש. • כתוצאה מכך, שביעות הרצון של אורחי החדר $satisfaction$ משתפרת ל-$\min(5.0, satisfaction + 0.2 * quantity)$. • ניתן להניח את תקינות הסוג והערך של quantity. • ניתן להניח כי ישנם אורחים בחדר בעת הקריאה למטודה
add_snacks(self, quantity)	מגדילה את מספר החטיפים שבמיני-בר ב-quantity, שהינו מספר שלם int חיובי ממש. • כתוצאה מכך, שביעות הרצון של אורחי החדר $satisfaction$ משתפרת ל-$\min(5.0, satisfaction + 0.3 * quantity)$. • ניתן להניח כי ישנם אורחים בחדר בעת הקריאה למטודה • עם זאת, החטיפים מלכלכים, וגם מדרדרים את רמת הניקיון $clean\_level$ ל-$\max(1, clean\_level - 1)$. • ניתן להניח את תקינות הסוג והערך של quantity.

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

```
>>> lr1 = LegacyRoom(5, 94, ["Ronen", "Dror", "Liat", "Smadar"], 5)
>>> lr1.satisfaction
1.0
>>> lr1.add_drinks(3)
>>> lr1.minibar_drinks
5
>>> lr1.satisfaction
1.6
>>> lr1.clean()
>>> lr1.clean_level
7
>>> lr1.add_snacks(2)
>>> lr1.minibar_snacks
4
>>> lr1.satisfaction
2.2
>>> lr1.clean_level
6
>>> lr1.check_out() ## note: None is returned, and so nothing is printed
>>> lr1.is_occupied()
False
>>> lr1.check_out()
Traceback (most recent call last):
  …
RoomError: Cannot check-out an empty room
```

<u>שאלה 3</u>

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 או היורש מ-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:\n
<number of BudgetRoom objects><רווח בודד>BudgetRooms\n
<number of LegacyRooms objects><רווח בודד>LegacyRooms\n
<number of Room objects that are not instances of BudgetRoom or
LegacyRooms><רווח בודד>other room types\n
<number of occupied rooms><רווח בודד>occupied rooms"
```

דוגמת הרצה:

```
>>> h = Hotel("Best",[BudgetRoom(15, 140, [], 5), BudgetRoom(1, 2,
["Liat"], 7)])
>>> h
Best hotel has:
2 BudgetRooms
0 LegacyRooms
0 other room types
1  occupied rooms
```

הערה: בכתיבת הקוד של __repr__, חישוב מספר החדרים השונים 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, rank)	מנסה לבצע צ'ק-אין לרשימת שמות האורחים guests (רשימת מחרוזות) לחדר אחד (כלשהו) מחדרי המלון, שדרגתו היא rank (מספר שלם).

	• במידה ונמצא חדר פנוי ומתאים בדרגתו, 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.
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 מה-case. • 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 הדומה לזה של הדוגמה למטה ממומש בפונקציה **test_hotel** בקובץ התרגיל.
- 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).

```
>>> rooms = [BudgetRoom(15, 140, [], 5), LegacyRoom(12, 101, ["Ronen", "Shir"],
6), BudgetRoom(1, 2, ["Liat"], 7), Room(2, 23, [], 6, 3)]
>>> h = Hotel("Dan",rooms)
```

```
>>> h.upgrade("Liat")
floor: 15
number: 140
guests: liat
clean_level: 5
rank: 1
satisfaction: 2.0
type: BudgetRoom
clean_stock: 0
>>> h.check_out("Ronen")
floor: 12
number: 101
guests: empty
clean_level: 6
rank: 2
satisfaction: 1.0
type: LegacyRoom
minibar_drinks: 2
minibar_snacks: 2
>>> h.check_in(["Alice", "Wonder"], 2)
floor: 12
number: 101
guests: alice, wonder
clean_level: 6
rank: 2
satisfaction: 1.0
type: LegacyRoom
minibar_drinks: 2
minibar_snacks: 2
>>> h.check_in(["Alex"], 3)
floor: 2
number: 23
guests: alex
clean_level: 6
rank: 3
satisfaction: 1.0
>>> h
Dan hotel has:
2 BudgetRooms
1 LegacyRooms
1 other room types
3 occupied rooms
>>> h.check_in(["Oded", "Shani"], 3)
>>> h.check_in(["Oded", "Shani"], 1)
floor: 1
number: 2
guests: oded, shani
clean_level: 7
rank: 1
satisfaction: 1.0
type: BudgetRoom
clean_stock: 0
>>> h.check_out("Liat")
floor: 15
number: 140
guests: empty
clean_level: 5
```

```
rank: 1
satisfaction: 2.0
type: BudgetRoom
clean_stock: 0
>>> h.check_out("Liat")
>>> h
Dan hotel has:
2 BudgetRooms
1 LegacyRooms
1 other room types
3 occupied rooms
```

בהצלחה!