חשיבה מחשובית ותכנות בשפת פייתון
תרגיל בית 11

הנחיות כלליות:

- קראו **היטב** את השאלות והקפידו שהתכניות שלכם פועלות בהתאם לנדרש.
- את התרגיל יש לפתור לבד!
- הקפידו על הוראות ההגשה המפורסמות במודל. בפרט, יש להגיש את כל השאלות יחד בקובץ ex#_012345678.py המצורף לתרגיל, לאחר החלפת ה# במספר התרגיל והחלפת הספרות 012345678 במספר ת.ז שלכם, כל 9 הספרות כולל ספרת הביקורת. למשל, אם מספר תעודת הזהות שלי הינו 112233445 ואני מגיש את תרגיל מספר 1, שם הקובץ שאגיש יהיה ex1_112233445.py.
- מועד אחרון להגשה: כמפורסם באתר.
- 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 וכי התוכנית אינה קורסת).
- היות ובדיקת התרגילים עשויה להיות אוטומטית, **יש להקפיד על פלטים מדויקים על פי הדוגמאות (כן כן, עד לרמת הרווח).**
- אופן ביצוע התרגיל: בתרגיל זה עליכם להשלים את הקוד בקובץ המצורף.
- מומלץ להתעדכן בפורום לגבי שאלות של סטודנטים אחרים וכמובן, במידה ועדיין משהו לא ברור, לשאול בעצמכם.
- החתימה של כל אחת מהפונקציות שעליכם לממש מופיעה כבר בשלד התרגיל ואין לשנותה. עליכם לממש את גוף הפונקציה בלבד. יש למחוק את הפקודה pass (היא נמצאת שם רק כדי שתוכלו להריץ חלקי קוד בלי לשים בהערה את המימושים החסרים) ולהחליף אותה במימוש המתאים.
- שימו לב 2, בשאלה בה נתבקשתם לממש פונקציה, אין לעשות שום דבר מעבר לכך. ניתן ואף רצוי לקרוא לפונקציה שכתבתם עם מגוון קלטים על מנת לוודא את תקינותה אך אין להשאיר את הקריאות הללו בהגשת התרגיל.
- שימו לב 3, אף פונקציה לא מדפיסה דבר אלא רק מחזירה ערך כלשהו.

רשת המזון המצליחה "ארנב" החליטה לפתוח סניף חדש במרכז הבינתחומי ושכרה את שירותיכם על מנת "למחשב" לה את מוצריה. בתרגיל זה נממש 3 מחלקות לייצוג פריטי מזון, משקאות, וארוחות שניתן להזמין ברשת המזון "ארנב". מומלץ בחום לקרוא תחילה את כל התרגיל ולראות כיצד ניתן לתכנן את הפונקציות בהתאם (אולי אפילו לנסות לשרטט דיאגרמה שמתארת את מחלקות התרגיל והקשר בינן). בשאלות בהן ניתן חופש פעולה לגבי צורת המימוש של המחלקות – כל פתרון שעובד על-פי הדרישות יתקבל. ניתן להוסיף מתודות מחלקה נוספות כרצונכם. **ניתן להניח שערכי הקלט בכל אחת מהמתודות <u>תקינים</u> למעט מקומות בהן נכתב במפורש אחרת. שאלות המסומנות בכוכביות דורשות חומר אשר יילמד בשיעור השני של OOP** (תוכלו להמתין עם סעיפים אלה להרצאה הבאה ו/או, רחמנא ליצלן, לקרוא באתר הזה הנחמד הזה, נו איך קוראים לו... אהה גוגל!).

<u>שאלה 1</u> – בשאלה זו נממש את המחלקה Dish שתייצג פריט מזון שניתן להזמין במסעדה. כל אובייקט מסוג Dish יכיל את השדות (attributes) הבאים:

1. name - שדה מסוג string המייצג את שם פריט המזון.
2. price - שדה מסוג float המייצג את מחיר פריט המזון.
3. calories - שדה מסוג int המייצג את מספר הקלוריות בפריט המזון.
4. ingredients - שדה מסוג list באורך כלשהו המכיל strings המציינים את שמות המרכיבים בפריט המזון.

א. ממשו את בנאי המחלקה __init__(self, name, price, calories, ingredients) הבנאי מקבל את הנתונים הדרושים ושומר אותם בשדות הרלוונטיים של האובייקט החדש שהוא מייצר. ניתן להניח כי המחרוזות (השם ואלה שברשימת המרכיבים) אינן ריקות ומכילות אותיות קטנות בלבד (לצורך הפשטות). יש לוודא שהמשתנים של price וcalories גדולים מ-0, ואם לא, יש <u>להקריס</u> (crash) את התוכנית ע"י העלאת חריגה (למשל, באמצעות הפקודה raise ValueError) עם הודעת שגיאה מתאימה <u>לבחירתכם</u>.

*ב. ממשו את מתודת ה- __repr__(self) של המחלקה אשר <u>תחזיר</u> מחרוזת המייצגת אובייקט של המחלקה כך שבמידה וננסה לבצע הדפסה של אובייקט מסוג Dish, תודפס בדיוק המחרוזת הבאה (שתי שורות):

<name> costs <price> NIS and contains: <ingredients>
(Only <calories> calories!)

כאשר יש להחליף את <> והמילה שביניהם בערך המתאים, וכן, החלק של כמות הקלוריות יופיע בשורה נפרדת (ראו דוגמאות הרצה בהמשך).

*ג. נרצה להגדיר יחס סדר בין המוצרים ע"פ מחיר. נגדיר שפריט מזון אחד "גדול" יותר מפריט מזון אחר ע"פ המחיר. במידה ושני פריטי המזון הם בעלי אותו מחיר, כמות המרכיבים היא זו שתכריע (יותר מרכיבים -> יותר גדול). במידה ושניהם בעלי אותו מחיר ומכילים את אותו מספר פריטים, אזי הם שווים בגודלם. הוסיפו למחלקה Dish את 3 המתודות הבאות:

- המתודה __eq__(self, other) (קיצור של equal) כך שאם פריט מזון x שווה לפריט מזון y (לפי ההגדרה לעיל) אזי עבור הביטוי x==y יוחזר True , אחרת יוחזר False.

- המתודה __lt__(self, other) (קיצור של lower than) כך שאם פריט מזון x קטן יותר מפריט מזון y (לפי ההגדרה לעיל) אזי עבור x<y יוחזר True, אחרת יוחזר False.

- המתודה __le__(self, other) (קיצור של lower or equal) כך שאם פריט מזון x קטן או שווה לפריט מזון y (לפי ההגדרה לעיל) אזי עבור x<=y יוחזר True , אחרת יוחזר False. שימו לב, אין צורך לממש את אותה הלוגיקה שוב עבור __le__. ניתן ואף רצוי להשתמש בעובדה שכבר מימשתם את __eq__ ו-__lt__ (איך?).

ד. מפעם לפעם, בעלי המסעדה רוצים לבצע שינויים בפריט מזון מסוים ולהוסיף\להחסיר מרכיבים מסוימים. ממשו את 2 המתודות הבאות:

- המתודה add_ingredient(self, ingredient, calories) אשר מקבלת 2 ערכים: ingredient מסוג string ו- calories מסוג int. המתודה תוסיף את המרכיב לרשימת המרכיבים ותוסיף את מספר הקלוריות שבמרכיב לסך הקלוריות של פריט המזון.

- המתודה remove_ingredient(self, ingredient, calories) אשר מקבלת 2 ערכים: ingredient מסוג string ו- calories מסוג int. המתודה תסיר את המרכיב מרשימת המרכיבים ותחסר את מספר הקלוריות שבמרכיב מסך הקלוריות של פריט המזון.

שימו לב! במידה ובוצע ניסיון להוריד מרכיב שאינו קיים, או ניסיון להוסיף מרכיב שכבר קיים, יש להקריס את התוכנית ע"י העלאת חריגה (למשל,raise ValueError) עם הודעת שגיאה מתאימה לבחירתכם.

להלן דוגמאות הרצה למחלקה Dish:

```
>>> hamburger = Dish('hamburger', 30.0, 500, ['bun', 'meat', 'Tomato'])
>>> print(hamburger)
hamburger costs 30.0 NIS and contains: ['bun', 'meat', 'Tomato']
(Only 500 calories!)
>>> cheese_burger = Dish('chesse_burger', 40.0, 600, ['bun', 'meat',
'cheese'])
>>> print(cheese_burger)
chesse_burger costs 40.0 NIS and contains: ['bun', 'meat', 'cheese']
(Only 600 calories!)
>>> hamburger < cheese_burger
True
>>> pizza = Dish('pizza', 40.0, 500, ['dough', 'cheese', 'tomato sauce',
'pepperoni'])
>>> print(pizza)
pizza costs 40.0 NIS and contains: ['dough', 'cheese', 'tomato sauce',
'pepperoni']
(Only 500 calories!)
>>> cheese_burger < pizza
True
>>> pizza.remove_ingredient('pepperoni', 100)
>>> print(pizza)
pizza costs 40.0 NIS and contains: ['dough', 'cheese', 'tomato sauce']
(Only 400 calories!)
>>> cheese_burger < pizza
False
>>> pizza == cheese_burger
True
```

הסבר: ראשית מייצרים פריט מזון בשם המבורגר ופריט מזון בשם צ'יזבורגר ומדפיסים את שתי המנות, כיוון שצ'יזבורגר יקר יותר מהמבורגר רגיל, נקבל שמתקיים אי השוויון הבא: hamburger < cheese_burger. לאחר מכן יוצרים פריט מזון בשם פיצה, מחירה זהה למחיר של צ'יזבורגר אך היא מכילה יותר מרכיבים ולכן נקבל ש: cheese_burger < pizza. לבסוף משנים את מנת הפיצה ומוציאים את מרכיב הפפרוני, כעת מחירה ומחיר הצ'יזבורגר זהה, והם גם מכילים אותו מספר פריטים ולכן פיצה "שווה" לצ'יזבורגר. שימו לב שבהדפסת הפיצה כעת לא מופיעה מרכיב הפפרוני וגם השתנתה כמות הקלוריות...

<u>שאלה 2</u> – בשאלה זו נממש את המחלקה Beverage שתייצג משקה שניתן להזמין במסעדה. כל אובייקט מסוג Beverage יכיל את השדות (attributes) הבאים:

1. name - שדה מסוג string המייצג את שם המשקה.
2. price - שדה מסוג float המייצג את מחיר המשקה.
3. is_diet - שדה מסוג bool המכיל True האם המשקה הוא דיאטטי ו-False אחרת.

א. ממשו את בנאי המחלקה __init__(self, name, price, is_diet)
יש לוודא שהמשתנה של price גדול מ- 0 ואם לא, יש <u>להקריס</u> את התוכנית ע"י העלאת חריגה (למשל, raise ValueError) עם הודעת שגיאה מתאימה <u>לבחירתכם</u>.

*ב. ממשו את מתודת ה- __repr__(self) של המחלקה אשר תחזיר מחרוזת המייצגת אובייקט של המחלקה כך שבמידה וננסה לבצע הדפסה של אובייקט מסוג Beverage, תודפס המחרוזת הבאה במידה והמשקה אינו דיאטטי:

<name> costs <price> NIS

והמחרוזת הבאה במידה והוא כן דיאטטי:

<name> costs <price> NIS (diet)

כאשר יש להחליף את <> והמילה שביניהם בערך המתאים (ראו דוגמאות הרצה בהמשך).

ג. ממשו את מתודת get_price(self, size) אשר תחזיר את מחיר המשקה כתלות בגודל המבוקש, המשתנה size הינו משתנה מסוג string המציין את גודל המשקה ערך חוקי למשתנה זה יהיה אחת מהמחרוזות הבאות: 'big', 'small', 'normal'. במידה וגודל המשקה המצוין הוא normal יוחזר המחיר הרגיל (כפי שהוא שמור בשדה price), עבור big יוחזר 120% מהמחיר הרגיל ועבור small יוחזר 80% מהמחיר הרגיל, בכל מקרה יש להחזיר את המחיר כ-float. במידה והמשתנה size אינו מכיל את אחד הערכים החוקיים לעיל, יש <u>להקריס</u> את התוכנית ע"י העלאת חריגה (למשל, raise ValueError) עם הודעת שגיאה מתאימה <u>לבחירתכם</u>. נרצה לתמוך גם באפשרות לקרוא למתודה זו ללא ציון גודל המשקה. במקרה כזה ערכו של המשתנה size יהיה normal, כלומר קריאה למתודה ללא כל ערך תתנהג כאילו נקראה עם הערך normal. לצורך כך **אתם רשאים לשנות את חתימת המתודה הנ"ל בקובץ השלד, אך לא את השם שלה**.

להלן דוגמאות הרצה למחלקה Beverage:

```
>>> coke = Beverage('coke', 5.0, False)
```

```
>>> print(coke)
coke costs 5.0 NIS
>>> diet_coke = Beverage('diet coke', 5.0, True)
>>> print(diet_coke)
diet coke costs 5.0 NIS (diet)
>>> print(coke.get_price())
5.0
>>> print(coke.get_price('big'))
6.0
>>> print(coke.get_price('huge'))
Traceback (most recent call last):
File "<input>", line 1, in <module>
File "<input>", line 62, in get_price
ValueError: invalid size
```

הסבר : המחיר הרגיל של קולה הינו 5.0, ולכן coke.get_price() יחזיר 5.0, זה שקול לקריאה ל-coke.get_price("normal"). בקריאה האחרונה נשלח ערך לא חוקי לפרמטר size ולכן הועלתה חריגת ValueError עם הודעת השגיאה "invalid size".

<u>שאלה 3</u> – בשאלה זו נממש את המחלקה Meal שתייצג ארוחה במסעדה. כל אובייקט מסוג Meal יכיל את השדות (attributes) הבאים:

1. name - שדה מסוג string המייצג את שם הארוחה.
2. beverage - שדה מסוג Beverage המייצג את המשקה שכלול בארוחה.
3. dishes - שדה מסוג list המציין את רשימת פריטי המזון הכלולים בארוחה. כל איבר ברשימה זו הוא אובייקט מסוג Dish.
4. is_diet - שדה מסוג bool המציין האם הארוחה דיאטטית או לא.

א. ממשו את בנאי המחלקה __init__(self, name, beverage, dishes)
הבנאי מקבל את הנתונים הדרושים ושומר אותם בשדות הרלוונטיים של האובייקט החדש שהוא מייצר. ניתן להניח שטיפוסי הקלט תקינים ואינם ריקים. שימו לב, **אינכם מקבלים** ערך עבור is_diet בחותמת הבנאי אלא עליכם להסיק אותו בעצמכם- ארוחה הינה דיאטטית אם המשקה של דיאטטי וגם סך הקלוריות של כלל פריטי המזון בארוחה קטן מ- 800.

*ב. ממשו את מתודת __repr__(self) של המחלקה אשר תחזיר מחרוזת המתארת אובייקט של המחלקה כך שבמידה וננסה לבצע הדפסה של אובייקט מסוג Meal, אם הוא מייצגת ארוחה לא דיאטטית תודפס המחרוזת הבאה:

<name> meal costs <price> NIS

במידה והארוחה דיאטטית תודפס המחרוזת הבאה:

<name> meal costs <price> NIS (healthy!)

כאשר יש להחליף את <> והמילה שביניהם בערך המתאים (ראו דוגמאות הרצה בהמשך).

*ג. ממשו את המתודה get_price(self) שתחזיר את מחיר הארוחה כ- float. מחיר הארוחה יורכב ממחיר המשקה בגודל <u>קטן</u> בתוספת סכום מחירי כל פריטי המזון בארוחה, פרט למחיר פריט המזון הכי "קטן" (על פי ההגדרה בשאלה 1 סעיף ג), ניתן להניח שאכן קיים פריט מזון אחד שיהיה בעל מחיר נמוך יותר מכל השאר.

רמז: כיוון שdishes היא רשימת ערכים מסוג Dish , אשר מממשת את הפונקציות __lt__, __eq__, __le__ ניתן לחשב את מחיר כל פריטי המזון ואז להשתמש בפונקציית ה-min המובנית של python על הרשימה על מנת למצוא את פריט המזון הזול ביותר ולהחסיר את מחירו מהמחיר של כלל המנות.

להלן דוגמאות הרצה למחלקה Meal:

```
>>> hamburger = Dish('hamburger', 40.0, 500, ['bun', 'meat', 'Tomato'])
>>> fries = Dish('fries', 10.0, 600, ['potato', 'salt'])
>>> salad = Dish('salad', 40.0, 200, ['tomato', 'lettuce', 'cucumber'])
>>> bread = Dish('bread', 10.0, 100, ['dough', 'butter'])
>>> coke = Beverage('coke', 5.0, False)
>>> water = Beverage('water', 3.0, True)
>>> burger_meal = Meal('burger', coke, [hamburger, fries])
>>> salad_meal = Meal('salad', water, [salad, bread])
>>> print(burger_meal)
burger meal costs 44.0 NIS
>>> print(salad_meal)
salad meal costs 42.4 NIS (healthy!)
```

הסבר: ארוחת המבורגר אינה ארוחה דיאטטית גם כיוון שסך הקלוריות שלה גדול מ 800 וגם המשקה שלה אינו דיאטטי, ארוחת סלט לעומת זאת היא דיאטטית כיוון שהמשקה שלה דיאטטי, וסך הקלוריות בארוחה הוא 300. המחיר של ארוחת ההמבורגר הינו מחיר של קולה קטנה (4 שקלים) ועוד סך כל מחירי המנות (50 שקלים) פחות מחיר הפריט מזון הזול ביותר בארוחה שזה הצ'יפס (10 שקלים) לכן סה"כ 44 שקלים. המחיר של ארוחת הסלט הינו המחיר של מים קטנים (2.4 שקלים) ועוד מחירי המנות (50 שקלים) פחות מחיר פריט המזון הכי זול שהוא הלחם (10 שקלים) לכן סה"כ 42.4 שקלים.

<u>שאלת בונוס</u> (5 נק׳): נרצה לספק אופציה לבדוק בצורה נוחה האם מרכיב כלשהו נמצא בארוחה מסוימת או לא (למשל, עבור טבעונים, אוכלי כשר, פירותניים, וכו...). באותו אופן כמו שאנחנו כבר מכירים ממבני נתונים אחרים כגון רשימות, מילונים וכו׳. כלומר נרצה לאפשר לבצע פעולות בדיקה באמצעות הפעולה in. למשל, אם ארוחת ילדים מכילה עגבנייה, הביטוי הבא: 'tomato' in kids_meal יחזיר True (כמובן בהנחה שkids_meal הינו אובייקט תקין מטיפוס Meal), ואם אין עגבנייה, יוחזר False. ראינו בכיתה שקיימות מתודות מיוחדות שניתן לממש במחלקות כדי לספק פונקציונליות סטנדרטית (כמו למשל __lt__ שמומש בשאלה 1 או __add__ במחלקה Rational). חפשו איזו מתודה מתאימה לפעולה הנ״ל וממשו אותה עבור המחלקה Meal כך שנקבל את הפונקציונליות המבוקשת (אין עבורה חתימה בשלד, הוסיפו אותה בעצמכם..). העזרו בסעיף 3.3.7 בדוקומנטציה של השפה בלינק הבא: https://docs.python.org/3/reference/datamodel.html.

```
>>> 'meat' in burger_meal
True
>>> 'meat' in salad_meal
False
```

הסבר: בשר (meat) נמצא בתוך פריט המזון ההמבורגר שהוא חלק מארוחת הבורגר (שהוגדרה בדוגמת ההרצה הקודמת), ולכן יוחזר True. לעומת זאת בשר לא נמצא באף אחד מרכיבי המזון בארוחת הסלט ולכן יוחזר False.