

תרגיל 6: תכנות מונחה עצמים

תאריך פרסום: 8.12.2022

תאריך הגשה: 25.12.2022 בשעה 23:59

מתרגלת אחראית: תום משיח

משקל תרגיל: 4 נקודות



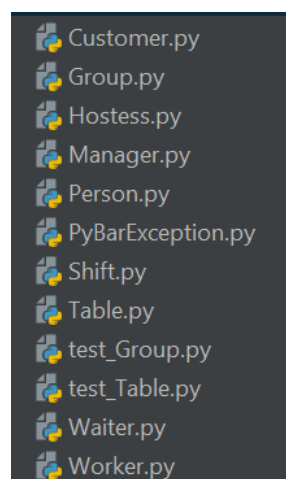
מנהל הבר "PyBar" ביקש ממך, מפתח תוכנה צעיר, לפתח עבורו תוכנה לניהול משמרות. להלן הדרישות הכלליות לאפיון התכנה. בהמשך התרגיל תמצאו הדרכה יותר מפורטת לגבי המימוש. התוכנה תומכת בעובדים ובלקוחות, כאשר העובדים מתחלקים למנהלים, מלצרים ומארחים. על התוכנה לאפשר לקבוצת לקוחות (המכילה לכל הפחות 2 לקוחות) להזמין שולחן. כל שולחן הוא בעל קיבולת מסוימת ויכול להכיל לכל היותר קבוצה אחת של לקוחות. התיאור של שולחן כולל את קבוצת הלקוחות שיושבת בשולחן, את מספר השולחן ואת ההזמנות והחשבון של הקבוצה. משמרת כוללת את מספר המשמרת, השולחנות בבר, העובדים במשמרת (מנהל, מלצר ומארח), התפריט במשמרת, רשימה של הזמנות - קבוצות שהזמינו שולחן למהלך המשמרת, את סך הכסף במשמרת ואת סך הטיפים במשמרת. המשמרת מתנהלת לפי הסדר הבא:

1. הושבה. המארח מקבל את פני הקבוצות ומושיב אותם בשולחנות בהתאם למספר המקומות בשולחן. ישנה עדיפות להושיב קודם כל שולחנות כמה שיותר גדולים. במידה ואין מקום הוא מתנצל בנימוס והקבוצה עוזבת את המסעדה (צוות הקורס לא ממליץ לנהוג באופן הזה...).
2. הזמנה. המלצר לוקח הזמנה מכל השולחנות בבר. במידה והם רוצים להזמין מצר שאין בתפריט יש להתנצל בנימוס. לאחר לקיחת ההזמנה המלצר מדפיס את ההזמנות והחשבון.
3. תשלום. המנהל עובר בין כל השולחנות וגובה מהם את התשלום. כאשר כל הקבוצות סיימו לשלם הוא מחשב את ההכנסה מהמשמרת וכמה ממנה היא טיפים.



דגשים:

1. מומלץ לקרוא את כל התרגיל **לפני** המימוש. מדובר בתרגיל ארוך, אך לא מורכב מבחינה אלגוריתמית. האתגר בתרגיל הוא תכנון נכון של המחלקות הנדרשות, מימוש מסודר ובדיקה יסודית של המימוש.
 2. בתרגיל **לא מצוין** באופן מפורש איך עליכם לממש את הפתרון. לדוגמא, לא יצוין מתי עליכם לדרוס שיטה, להשתמש בשיטת האב, להגדיר מחלקה\שיטה אבסטרקטית, להגדיר שדה פרטי, לבצע shallow copy או deep copy וכדומה. עליכם יהיה להחליט על פרטי המימוש **תוך עמידה בדרישות התרגיל ובעקרונות שלמדתם בקורס** (לדוגמא, להמנע משכפול קוד).
 3. בניגוד לתרגילים קודמים, **בתרגיל זה אין להניח שהקלט תקין!** עבור כל שיטה / פונקציה מוגדרות הדרישות לגבי הקלט. הנחות על קלטים מסוימים יצוינו באופן מפורש. ובמידה והקלט לא תקין עליכם לזרוק חריגה מתאימה.
 4. דוגמאות הרצה לחלק מהשיטות מצויות בקובץ נפרד בשם Tests.py.
 5. מכיוון שהעבודה עמוסה מבחינת כמות הקוד שעליכם לממש וכדי להקל עליכם - אינכם נדרשים לכתוב docstring בעבודה זו.
 6. נא לקרוא את כל העבודה **לפני** שאתם מתחילים לכתוב את הקוד (זו הפעם השניה של הסעיף הזה ולא במקרה..).
- שימו לב:** עליכם ליצור ולהגיש בעבודה זו 12 קבצים. יש לקרוא לקבצים בדיוק כפי שכתוב עבור כל מחלקה. בנוסף יש לשמור ב-pycharm את כל הקבצים באותה תיקייה על מנת שתוכלו לייבא (import) אותם אחד לשני במידת הצורך על ידי שמם בלבד ללא נתיב לקובץ.
- לדוגמא: `from Table import Table`
- רשימת הקבצים שיש להגיש:



חלק א':

בחלק זה תצרו את המחלקות השונות הבונות את מערכת הבר. שימו לב כי כל מחלקה נמצאת בקובץ נפרד כך ששם הקובץ הוא שם המחלקה.

חריגות (PyBarException)

עליכם לממש מספר מחלקות של חריגות שישמשו אתכם בהמשך התרגיל בקובץ בשם `PyBarExceptions.py`. עליכם יהיה לזרוק את החריגות המתאימות בהתאם לדרישות שיפורטו ביתר המחלקות.

`InvalidInputException`

החריגה תיזרק כאשר הקלט לא תואם את הדרישות.

`OccupiedTableException`

החריגה תיזרק כאשר מנסים להוסיב קבוצה בשולחן תפוס.

`TooSmallTableException`

החריגה תיזרק כאשר מנסים להוסיב קבוצה בעלת מספר לקוחות גבוה יותר מהקיבולת המקסימלית של השולחן.

`EmptyTableException`

החריגה תיזרק כאשר מנסים לקחת הזמנה או תשלום בשולחן ריק.

`AccessDeniedException`

החריגה תיזרק כאשר עובד שהוא לא מטיפוס מנהל מנסה לגשת לכסף במשמרת / הטיפ במשמרת.

בחריגה זו בלבד עליכם לדרוס את השיטה `__str__` של המחלקה `AccessDeniedException`:

```
def __str__(self):
```

השיטה תחזיר מחרוזת המייצגת תיאור של החריגה.

פלט:

○ `[-str]` - מחרוזת המייצגת את השגיאה. מתחיל במילה `ERROR:` ולאחר מכן הודעת השגיאה.

`ERROR: message`

דוגמא:

```
try:
    raise AccessDeniedException("Only a manager can access the
money")
```

```
except Exception as e:
    print(e)
```

יודפס:

'ERROR: Only a manager can access the money'

אדם (Person):

מחלקה אשר מייצגת אדם. במערכת הבר לא ניתן ליצור אובייקט מסוג אדם. אדם מתואר על ידי השדות הבאים אליהם לא ניתן לשנות ולגשת מחוץ למחלקת אדם:

🌀 **-name** שם מטיפוס מחרוזת ולא ריקה.

🌀 **-age** גיל מטיפוס int, מספר שלם הגדול שווה ל-18.

ממשו את מחלקת אדם:

```
def __init__(self, name, age):
```

בנאי המאתחל את שדות המחלקה.

במידה ואחד השדות קיבל ערך לא תקין יש לזרוק חריגה מתאימה.

```
def __str__(self):
```

תחזיר מחרוזת המייצגת אדם בפורמט:

Name:name,**Age:**age

כאשר אדום מסמן את התווים הקבועים ושחור את ערכי השדות המתאימים (ללא רווחים).

לדוגמא:

Name:Avi,Age:20

```
def get_name(self):
```

תחזיר מחרוזת שהינה שם האדם.

```
def get_age(self):
```

תחזיר מספר שלם שהינו גיל האדם.

לקוח (Customer):

מחלקה אשר מייצגת לקוח. לקוח הינו אדם (Person) שבנוסף מוגדר לו הטיפ הקבוע אותו הוא נוהג להשאיר.

tip – השבר העשרוני של הטיפ אותו הלקוח ירצה להוסיף. על מספר זה להיות גדול שווה מ-0.

לדוגמא, לקוח שהטיפ שהוא משאיר הינו 0.17, במידה והוא צריך לשלם 100 שקלים הוא ישלם 117 שקלים.

ממשו את מחלקת לקוח:

```
def __init__(self, name, age, tip):
```

בנאי המאתחל את שדות המחלקה.

במידה ואחד השדות קיבל ערך לא תקין יש לזרוק חריגה מתאימה.

```
def __str__(self):
```

תחזיר מחרוזת המייצגת לקוח בפורמט:

Name:name, Age:age, Tip:tip%

כאשר אדום מסמן את התווים הקבועים ושחור את ערכי השדות המתאימים (ללא רווחים).

לדוגמא:

Name:Avi, Age:20, Tip:12%

עובד (Worker):

מחלקה אשר מייצגת עובד. עובד הינו אדם (Person). במערכת הבר לא ניתן ליצור אובייקט מסוג עובד.

ממשו את מחלקת עובד:

```
def __str__(self):
```

תחזיר מחרוזת המייצגת עובד בפורמט

Name:name, Age:age, Job:

כאשר אדום מסמן את התווים הקבועים ושחור את ערכי השדות המתאימים (ללא רווחים).

לדוגמא:

Name:Avi, Age:20, Job:

```
def work(self, shift):
```

המטרה של השיטה work היא לבצע את פעולות העובד במשמרת מסויימת.

אין צורך לבצע בדיקת קלט על shift.

בהמשך נבנה 3 מחלקות היורשות מ-Worker: Manager, Waiter, Hostess כך שלכל אחת מהן יהיה מימוש אחר לשיטה work.

קבוצה (Group):

מחלקה אשר מייצגת קבוצה. קבוצה מתוארת על ידי השדות הבאים:

🌀 **customer_list** - רשימה של הלקוחות המרכיבים את הקבוצה הערך חייב להיות

רשימה באורך של לפחות 2. ניתן להניח שבמידה וזו רשימה האיברים בתוכה הינם

מטיפוס של Customer.

🌀 **order_dict** - מילון של הפריטים והמספר מכל פריט אותם הקבוצה תהיה מעוניינת

להזמין. במידה ואכן הערך שהתקבל הינו מילון ניתן להניח כי המפתחות הם מטיפוס

string והערכים הם מטיפוס int. שימו לב כי לא ניתן לגשת לשדה זה מחוץ למחלקה.

דוגמא למילון הזמנה:

```
order_dict_example = {"Goldstar": 3, "Red Wine": 1, "French fries": 1}
```

ממשו את מחלקת קבוצה:

```
def __init__(self, customer_list, order_dict):
```

בנאי המאתחל את שדות המחלקה.

במידה ואחד השדות קיבל ערך לא תקין יש לזרוק חריגה מתאימה.

```
def get_order(self):
```

בשיטה זו עליכם להחזיר את מילון ההזמנה של הקבוצה.

זכרו כי אנו לא מעוניינים ששדה זה יהיה ניתן לשינוי מחוץ למחלקה.

```
def __str__(self):
```

תחזיר מחרוזת המייצגת קבוצה בפורמט:

The group has group_len members, their order: order_dict

כאשר אדום מסמן את התווים הקבועים ושחור את ערכי השדות המתאימים.

לדוגמא:

The group has 5 members, their order: {"Negev": 3, "Coke":2 "Pizza": 2}

```
def __repr__(self):
```

השיטה תחזיר מחרוזת המייצגת את הקבוצה כפי שהוגדר בשיטה str.

בנוסף, יש לדרוס את השיטות הבאות:

1. שיטה המאפשרת את השימוש ב-`len` ומחזירה את מספר הלקוחות בקבוצה. דוגמא לשימוש: `len(group)`.
2. שיטה המחזירה האם הקבוצה קטנה מקבוצה אחרת. `**` בסעיף 2 גודל הקבוצה נקבע כפי שהוסבר בסעיף 1.

`def get_customers_string(self):`

שיטה זו תחזיר מחרוזת עם שמות הלקוחות בקבוצה כאשר שמותיהם מופרדים בפסיק ולפני השם האחרון יש את המילה "and" כפי שניתן לראות בדוגמא הבא:

Assaf, Noy, Essra and Tom

מימוש שיטה זו נתון לכם ותוכלו להשתמש בו:

```
def get_customers_string(self):
    name_list = [customer.get_name() for customer in self.customers_list]
    return_string = ", ".join(name_list[:-1]) + " and " + name_list[-1]
    return return_string
```

שולחן (Table):

מחלקה המייצגת שולחן. שולחן מוגדר על ידי השדות הבאים:

- 🌀 **number** - מספר סידורי של השולחן. מספר שלם הגדול מ-0.
- 🌀 **max capacity** - מספר המקומות בשולחן, כלומר גודל הקבוצה המקסימלי היכולה לשבת בו. מספר שלם הגדול מ-0.
- 🌀 **group [Group]** - הקבוצה היושבת בשולחן (None אם השולחן ריק)
- 🌀 **bill [dict]** - מילון המייצג את חשבון השולחן. המפתחות הינם שם הפריט והערכים הינם סך כל התשלום על הפריט הנ"ל. (למשל אם גולדסטאר עולה 30 והשולחן הזמין 2 גולדסטאר הערך של המפתח גולדסטאר יהיה 60).

ממשו את מחלקת שולחן:

`def __init__(self, number, max_capacity):`

בנאי המאתחל את שדות המחלקה.

במידה ואחד השדות קיבל ערך לא תקין יש לזרוק חריגה מתאימה.

`def __str__(self):`

תחזיר מחרוזת המייצגת שולחן בפורמט:

Table number number **has** max_capacity **seats**.

כאשר אדום מסמן את התווים הקבועים ושחור את ערכי השדות המתאימים.

לדוגמא:

Table number 1 has 4 seats.

`def __repr__(self):`

השיטה תחזיר מחרוזת המייצגת את השולחן כפי שהוגדר בשיטה `str`.

בנוסף, יש לדרוס את המטודות הבאות:

1. שיטה המחזירה את קיבולת השולחן. ניתן להשתמש בה באופן הבא: `len(table)`

קיבולת השולחן הינו מספר המקומות שיש בשולחן (`max_capacity`).

2. שיטה המחזירה האם שולחן קטן משולחן אחר.

****** בסעיף 2 גודל השולחן נקבע כפי שהוסבר בסעיף 1.

`def is_empty(self):`

שיטה המחזירה ערך בולאני המייצג האם השולחן ריק או שיש בו קבוצה.

פלט:

○ `[bool]-True` אם השולחן ריק ואין בו קבוצה, אחרת `False`.

עליכם לתמוך בשלוש פעולות של המחלקה:

`def seat(self, group):`

הושבת קבוצה לשולחן. השיטה תכניס את הקבוצה לערך השדה `group`. במידה והשולחן תפוס

(כלומר יש קבוצה היושבת בו) או במידה וקיבולת השולחן קטנה מכמות הלקוחות בקבוצה יש

לזרוק חריגות מתאימות. במידה ונזרקה חריגה כלשהי, אין להושיב את הקבוצה בשולחן.

קלט:

○ `group [Group]` - אובייקט מטיפוס קבוצה. אם האובייקט הוא לא מטיפוס קבוצה יש

לזרוק חריגה בהתאם.

`def order(self, menu):`

השיטה לוקחת הזמנה משולחן. במידה והשולחן ריק תיזרק שגיאה מתאימה.

עבור כל פריט במילון ההזמנות של הקבוצה היושבת בשולחן ייבדק האם הפריט קיים בתפריט.

במידה ולא תודפס ההודעה עם הפורמט הבא:

Sorry we don't have product.

לדוגמא:

Sorry we don't have Goldstar.

במידה והפריט קיים בתפריט יש להוסיף את הפריט ואת סך המחיר שיש לשלם עליו למילון החשבון בשולחן.

לאחר סיום לקיחת ההזמנה יש להדפיס למסך את ההודעה הבאה:

Your bill is:

ולאחר מכן יש להדפיס את החשבון באופן הבא:

product1.....money1

product2.....money2

product3.....money3

(ישנן 10 נקודות בין שם הפריט לסך המחיר שלו ואין רווחים).

קלט:

○ **menu [dict]** - מילון אשר המפתחות הינם פריטים המצויים בתפריט והערכים הינם

המחיר בשקלים של כל פריט בהתאם. ניתן להניח כי הקלט תקין.

לדוגמא:

```
menu = {"Goldstar": 32, "Mojito": 48, "White wine": 29, "Hamburger": 55}
```

דוגמאת הרצה:

התפריט שהועבר כקלט הינו התפריט מהדוגמא לעיל.

הזמנת הקבוצה היושב בשולחן הינה:

```
{'Goldstar': 1, 'Mojito': 2, 'Pizza': 1}
```

לאחר הפעלת השיטה יודפס:

Sorry we don't have Pizza.

Your bill is:

Goldstar.....32

Mojito.....96

def pay(self):

השיטה משלמת את חשבון השולחן. במידה והשולחן ריק תיזרק חריגה מתאימה.

התשלום בבר עובד באופן הבא: כל חברי הקבוצה מתחלקים בסכום שיש לשלם באופן שווה. כל

אחד מוסיף לחלק שלו את הטיפ שהוא נוהג להשאיר. השיטה תחזיר את סך החשבון של השולחן

ואת סך הטיפ שהשולחן השאיר למלצר.

פלט:

○ **[int]** - סך הכסף בשקלים של חשבון השולחן (לא כולל טיפ)

○ **[float]** - סך הטיפ שהשאר השולחן.

לדוגמא:

אם נסתכל על השולחן בו דנו קודם, נניח כי בשולחן יש 3 לקוחות כאשר הראשון שם 10% טיפ, השני 15% והשלישי 20%. סך הכל השולחן צריך לשלם 128 שקלים. כל אחד צריך לשלם 42.66 שקלים. הלקוח הראשון ישלם 46.9, הלקוח השני ישלם 49 והלקוח השלישי ישלם 51.2. סך הכל הטיפ שהם השאירו הינו 19.1 שקלים. השיטה תחזיר: 128 ו-19.1.

משמרת (Shift):

מחלקה המייצגת משמרת בבר. משמרת מאופיינת על ידי השדות הבאים:

🌀 **shift_number** - מספר שלם הגדול מ-0 המייצג את מספר המשמרת.

🌀 **table_list** - רשימה של השולחנות הזמינים במשמרת. במידה ואכן הוכנסה רשימה ניתן להניח כי כל האיברים בה הינם מטיפוס שולחן.

🌀 **group_list** - רשימה של קבוצת המעוניינות לשבת במשמרת. במידה ואכן הוכנסה רשימה ניתן להניח כי כל האיברים בה הינם מטיפוס קבוצה.

🌀 **workers_list** - רשימה של העובדים במשמרת לפי הסדר הבא: מארח, מלצר, מנהל. שימו לב כי יש חשיבות לסדר של האיברים ברשימה ושהרשימה תכיל בדיוק את שלושת האיברים הללו.

🌀 **menu** - מילון המייצג את התפריט במשמרת. במידה ואכן הוכנס מילון ניתן להניח כי טיפוס המפתחות והערכים במילון הינם תקינים.

דוגמא למילון:

```
{"Goldstar": 32, "Mojito": 48, "White wine": 29, "Hamburger": 55}
```

🌀 **total_money** - סך כל הכסף שהכניסה המשמרת (לא כולל שירות). שימו לב שלשדה זה אין גישה מחוץ למחלקה.

🌀 **total_tip** - סך כל הטיפ שהכניסה המשמרת. שימו לב שלשדה זה אין גישה מחוץ למחלקה.

ממשו את מחלקת משמרת:

```
def __init__(self, shift_number, table_list, groups_list, workers_list, menu):
```

בנאי המאתחל את שדות המחלקה.

במידה ואחד השדות קיבל ערך לא תקין יש לזרוק חריגה מתאימה.

```
def __str__(self):
```

תחזיר מחרוזת המייצגת משמרת בפורמט:

Shift number number.

כאשר אדום מסמן את התווים הקבועים ושחור את ערכי השדות המתאימים.

לדוגמא:

Shift number 1.

def __repr__(self):

השיטה תחזיר מחרוזת המייצגת את המשמרת כפי שהוגדר בשיטה str.

def add_money(self, money):

השיטה תוסיף כסף לכלל הכסף של המשמרת.

קלט:

- money [int] - מספר חיובי שלם המתאר את הכסף בשקלים שיש להוסיף למשמרת. ניתן להניח כי הערך שהוכנס הינו מספר שלם אך במידה והוכנס ערך שלילי יש לזרוק חריגה מתאימה.

def add_tip(self, tip):

השיטה תוסיף טיפ לכלל הטיפים של המשמרת.

קלט:

- tip [float] - מספר חיובי המתאר את הטיפ בשקלים שיש להוסיף למשמרת. ניתן להניח כי הערך שהוכנס הינו מספר עשרוני אך במידה והוכנס ערך שלילי יש לזרוק חריגה מתאימה.

def get_money(self, manager):

השיטה תחזיר את הכסף שיש במשמרת. רק למנהל יש גישה לכסף ולכן יש להעביר כארגומנט אובייקט מסוג מנהל.

קלט:

- manager [Manager] - אובייקט מסוג מנהל. במידה והאובייקט שהועבר אינו מטיפוס מנהל יש לזרוק חריגה מתאימה עם הודעת השגיאה הבאה:

Only a manager can access the money

def get_tip(self, manager):

השיטה תחזיר את הטיפים שיש במשמרת. רק למנהל יש גישה לטיפים ולכן יש להעביר כארגומנט אובייקט מסוג מנהל.

קלט:

○ [Manager] manager - אובייקט מסוג מנהל. במידה והאובייקט שהועבר אינו מטיפוס

מנהל יש לזרוק חריגה מתאימה עם הודעת השגיאה הבאה:

Only a manager can access the tip

`def shift_day(self):`

שיטה המבצעת את יום המשמרת.

השיטה משתמשת בעיקרון הפולימורפיזם על עובדי המשמרת. על כל עובד לפי הסדר ברשימה

לבצע את השיטה `work`. בין עובד לעובד יש להדפיס 40 קווים תחתונים.

ניתן לראות דוגמאת הרצה בקובץ `Test.py`.

מנהל (Manager):

מחלקה המייצגת מנהל. מנהל הינו עובד (`Worker`).

ממשו את מחלקת מנהל:

`def __str__(self):`

תחזיר מחרוזת המייצגת משמרת בפורמט:

Name:name, Age:age, Job:Manager

כאשר אדום מסמן את התווים הקבועים ושחור את ערכי השדות המתאימים (ללא רווחים).

לדוגמא:

Name:Avi, Age:20, Job:Manager

`def work(self, shift):`

שיטה זו עוברת על כל השולחנות ברשימת השולחנות של המשמרת ומבצעת תשלום עבור כל שולחן.

עבור כל שולחן המנהל מוסיף את הכסף והטיפ שהתקבל לכסף הכללי והטיפ הכללי של המשמרת.

לאחר שהשולחן סיים לשלם מתבצעת הדפסה למסך:

Thank you name, name,... and name! You paid total pay shekels. See you next time!

שימו לב שיש להדפיס ב-`total money` את סך הכסף שהשולחן שילם, כלומר כולל את הטיפ.

לדוגמא:

Thank you Tal and Adi! You paid 146.56 shekels. See you next time!

לאחר שכל השולחנות סיימו לשלם, יש לבצע הדפסה למסך אשר אומרת כמה בסך המשמרת
הכניסה מהמכירות ומהטיפ:

This is the end of the shift:

shift

Total money: money, **total tip:** tip

לדוגמא:

This is the end of the shift:

Shift number: 1.

Total money: 128, total tip: 18.560000000000002

קלט:

○ [Shift] -shift - המשמרת בה המנהל עובד. ניתן להניח כי הקלט תקין.

מלצר (Waiter)

מחלקה המייצגת מלצר. מלצר הינו עובד (Worker).

ממשו את מחלקת מלצר:

def __str__(self):

תחזיר מחרוזת המייצגת משמרת בפורמט:

Name:name, **Age:**age, **Job:**Waiter

כאשר אדום מסמן את התווים הקבועים ושחור את ערכי השדות המתאימים (ללא רווחים).

לדוגמא:

Name:Avi, Age:20, Job:Waiter

def work(self, shift):

קלט:

○ [Shift] -shift - המשמרת בה המלצר עובד. ניתן להניח כי הקלט תקין.

שיטה זו עוברת על כל השולחנות ברשימת השולחנות של המשמרת ומבצעת הזמנה עבור כל שולחן.

עבור כל שולחן טרם ביצוע ההזמנה ישנה הדפסה למסך:

Hey name, name,... **and** name! **My name is** name **and I'm your waiter.**

לדוגמא:

Hey Tal, Alon and Adi! My name is Noy and I'm your waiter.

לאחר מכן מתבצעת ההזמנה מהשולחן.

בין הזמנה של שולחן לשולחן ישנה שורה ריקה.

מארח (Hostess)

המחלקה מייצגת מארח. מארח הינו עובד (Worker).

ממשו את מחלקת מארח:

```
def __str__(self):
```

תחזיר מחרוזת המייצגת משמרת בפורמט:

Name:name, Age:age, Job:Hostess

כאשר אדום מסמן את התווים הקבועים ושחור את ערכי השדות המתאימים (ללא רווחים).

לדוגמא:

Name:Avi, Age:20, Job:Hostess

```
def work(self, shift):
```

שיטה זו תושיב את הקבוצות הנמצאות ברשימת הקבוצות המעוניינות לשבת במשמרת בשולחנות

של המשמרת. ישנה עדיפות להושבת קבוצות גדולות על פני קבוצות קטנות.

שימו לב, במימוש שיטה זו עליכם להשתמש בפונקציה המובנת sorted אשר תקבל רשימה

ותחזיר את הרשימה בסדר עולה.

המארח יעבור על קבוצה אחר קבוצה (מהגדולה לקטנה) ועבור כל קבוצה תבדוק האם קיים שולחן

ריק שהקיבולת שלו גדולה או שווה ממספר האנשים בקבוצה. במידה וכן המארח יושיב את

הקבוצה בשולחן, תודפס ההודעה הבאה ולאחריה המארח יעבור לקבוצה הבאה:

name, name,... and name **you can seat on table number please.**

לדוגמא:

Tal and Adi you can seat on table 2 please.

במידה והמארח עבר על כל השולחנות ולא מצאה שולחן פנוי המתאים לגודל הקבוצה תודפס

ההודעה הבאה:

Sorry name, name,... and name, **we don't have place for** number **people.**

לדוגמא:

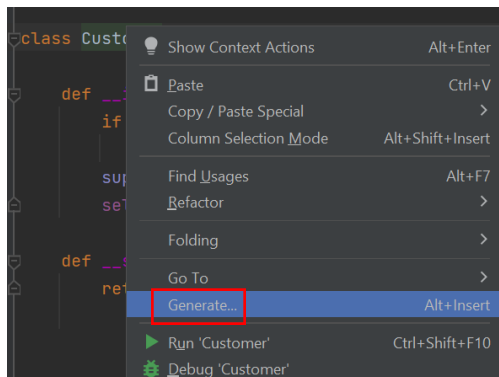
Sorry Avi, Bruno and Shaked, we don't have place for 3 people.

חלק ב' - Unit test

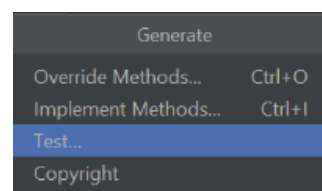
זאת הפעם הראשונה שהינכם מתמודדים עם מספר רב של קבצים, שיטות ומחלקות. על מנת שתוכלו לבדוק את התרגיל שלכם בצורה קלה ונוחה תשתמשו בכלי בדיקות תכנה שנקרא **unit test**, שישמש אתכם בהמשך דרככם המקצועית.

שליבים לפתיחת קובץ Test

על מנת לפתוח קובץ test בצורה נוחה בצעו את השלבים הבאים:
לחצן ימיני בעבר על שם המחלקה ובחירה ב-
Generate. לאחר מכן, בחרו test.

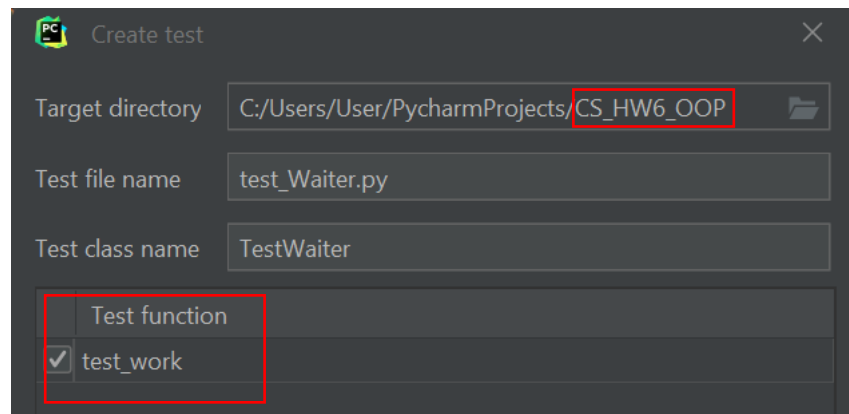


יפתח החלון הבא:



יש לבחור ב-Test...

בתיקיית היעד (target directory) הגדירו את התיקייה בה כתבתם את קבצי התרגיל.
שם הקובץ ושם המחלקה נתון לבחירתכם.
סמנו ב-V את כל השיטות המופיעות תחת קטגוריות test function על מנת ליצור טסטים מובנים.



לאחר לחיצה על OK יוצר הקובץ הבא:

```
from unittest import TestCase

class TestWaiter(TestCase):
    def test_work(self):
        self.fail()
```

שימו לב כי לא נוצרו טסטים אוטומטים לשיטות שדרסתם (override) כמו `__str__` ועליכם ליצר להם טסטים בצורה ידנית.

בנוסף, על מנת לקרוא למחלקה אותה אנו רוצים לבדוק, יש לייבא את הקובץ שלה על ידי הפקודה:
`from <fileName> import <className>`

שליבים להגדרת test חדש

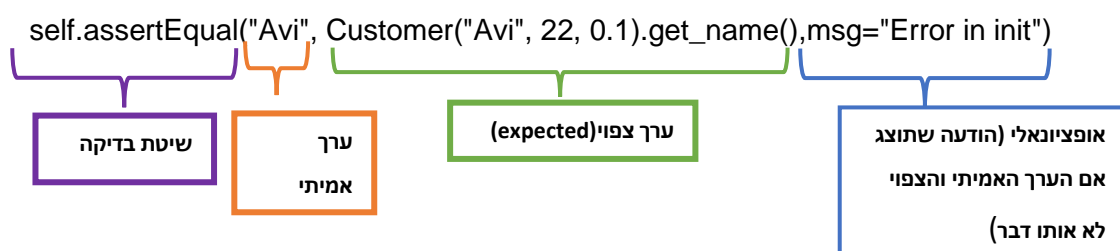
כדי ליצור טסט חדש נגדיר פונקציה כך ששם הפונקציה מתחיל עם המילה `test`. למשל:

```
def test_str(self):
```

בתוך הפונקציה נבצע תרחיש מסוים שבודק שיטה בקוד. למשל, בדוגמה אנו בודקים כי הבנאי של המחלקה `Customer` מבוצע כהלכה.

על מנת להריץ את ה-`test`, עלינו להשתמש באובייקט `self` ובשיטת בדיקה.

למשל, בדוגמה:

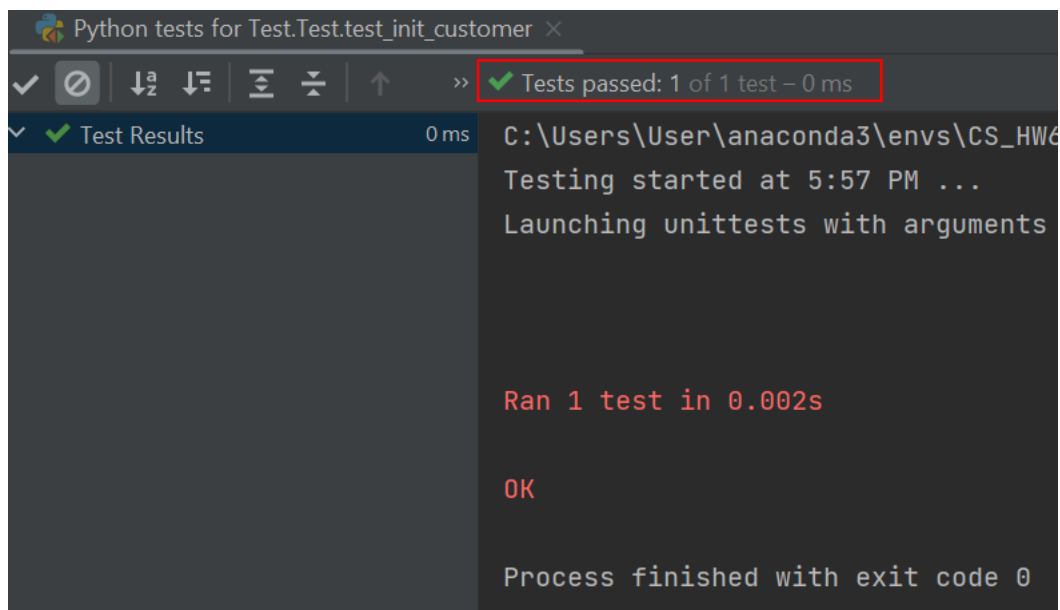


מוזמנים לקרוא על כל המתודות להשוואה [בקישור](#).

ניתן להריץ טסט בודד על ידי לחיצה על החץ הירוק בצד שמאל ליד שם הפונקציה:

```
▶ class Test(TestCase):  
▶ def test_init_customer(self):  
    noa = Customer("Noa", 22, 0.15)  
    self.assertEqual("Noa", noa.get_name())  
    self.assertRaises(InvalidInputException, Customer, 12, 20, 0.12)
```

לאחר ההרצה נקבל פלט כי הטסט עבר:



Python tests for Test.Test.test_init_customer

✓ Tests passed: 1 of 1 test – 0 ms

✓ Test Results 0 ms

C:\Users\User\anaconda3\envs\CS_HW6
Testing started at 5:57 PM ...
Launching unittests with arguments

Ran 1 test in 0.002s

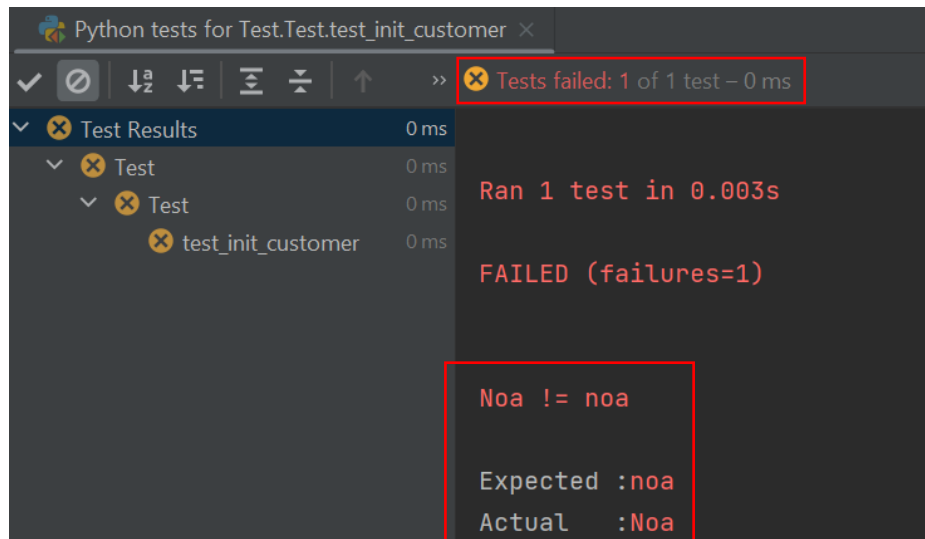
OK

Process finished with exit code 0

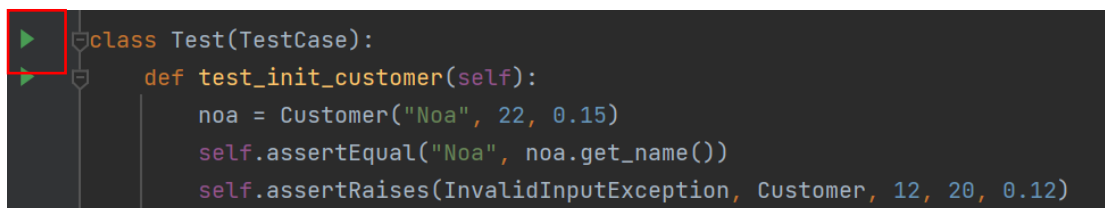
אם למשל נשנה את הטסט לטסט שגוי:

```
▶ class Test(TestCase):  
▶ def test_init_customer(self):  
    noa = Customer("Noa", 22, 0.15)  
    self.assertEqual("noa", noa.get_name())  
    self.assertRaises(InvalidInputException, Customer, 12, 20, 0.12)
```

כאשר נריץ נקבל הודעה כי הטסט נכשל:



על מנת להריץ את כל הטסטים לחצו על החץ הירוק ליד שם המחלקה:



מחלקת Test לדוגמה

סיפקנו לכם מחלקה בשם Tests.py שמכילה דוגמאות לטסטים שבהם תוכלו לבדוק מספר שיטות שונות. שימו לב, המחלקה תוכל לרוץ רק לאחר שסיימתם לממש את כל הדרישות של העבודה. יש לשמור קובץ זה בתיקייה בה נמצאים כל הקבצים שיצרתם.

דרישות התרגיל

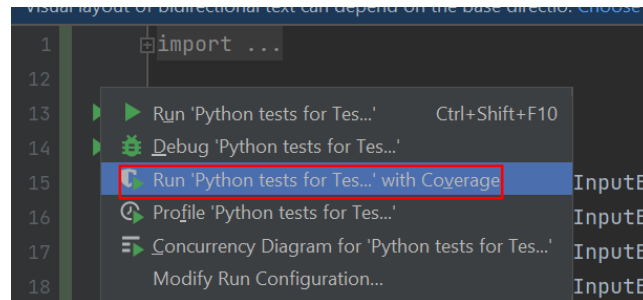
כחלק מדרישות העבודה עליכם לכתוב שני קבצי test עבור שתי מחלקות: **קבוצה ושולחן**. שימו לב כי עבור המחלקה "שולחן" בדיקת השיטה **order** נתונה לכם בקובץ ה-Test. את יתר השיטות עליכם לממש אליהם בדיקה בעצמכם.

חישבו: כיצד אתם יכולים לכסות את כל מקרי הקצה שיש בשיטה אותה אתם בודקים.

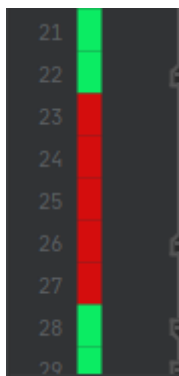
המלצה חמה: לכל מחלקה נסו לכתוב טסטים לפני שאתם כותבים את הקוד על פי השיטה TDD (test driven development) שהוצגה בתרגול 7. באופן הזה אתם כותבים את הקוד כדי שיעבור את הטסטים ולא להיפך. זוהי שיטה מקובלת לפיתוח תכנה.

המלצה חמה חמה: דרישות התרגיל הינם לממש unit test עבור שתי מחלקות בלבד. אנו ממליצים לכם לממש עבור כל המחלקות שיצרתם.

כדי שתוכל לדעת האם הטסטים שלכם מכסים את מרבית הקוד שכתבתם תוכלו להשתמש באופציית ה-coverage. לחצן ימני על החץ הירוק ובחירה "run with coverage".



לאחר ההרצה תוכל לראות את השורות שהטסטים כיסו ליד שם המחלקה. למשל:



בנוסף, כאשר תיכנסו לקובץ של המחלקה תוכלו לראות את הצבעים הבאים:

- ירוק – הטסט עבר בשורה זאת ולכן השורה נבדקה.
 - אדום – הטסט לא עבר בשורה זאת ולכן השורה לא נבדקה.
- לקריאה נוספת על ה-coverage מוזמנים לקרוא [בא](#).

עליכם לכתוב עבור המחלקות שולחן וקבוצה קבצי טסט שיכסו לפחות 95% משורות כל מחלקה בהתאם.

שימו לב כי אסור לשתף טסטים בין סטודנטים.

בהצלחה ועבודה מהנה !

