מס טופס:_____מספר ת"ז:_____

אביב, מועד ב, 06.10.2024

# בחינה בקורס תכנות בפייתון למהנדסים 0509-1820

חגי לוי, שחר צרפתי, רותם פלח

## הנחיות כלליות

- משך הבחינה שלוש שעות.
- חומר עזר מותר: שני דפי A4 כתובים משני הצדדים.
- במבחן שלוש שאלות הכוללות סעיפי משנה. במבחן 105 נקודות, כל ציון מעל 100 יעוגל ל-100. ניקוד כל סעיף מצוין לידו. אין בהכרח קשר בין ניקוד הסעיף ובין רמת הקושי שלו.
- את הפתרונות יש לכתוב במסגרות המסומנות לכל שאלה בטופס הבחינה.
- במחברת הטיוטא יש להשתמש כטיוטא בלבד. בבדיקת המבחן, מחברות הטיוטא לא ייבדקו.
- מומלץ לכתוב פתרון מלא במחברת הטיוטא ורק לאחר מכן להעתיק את הפתרון לטופס הבחינה.
- בכל סעיף ניתן להשתמש בקוד שהתבקשתם לכתוב בסעיפים קודמים, גם אם לא כתבתם אותו.
- ניתן להוסיף פונקציות עזר אם לא צוין אחרת בשאלה.
- ניתן להניח שהקלט לקוד שתכתבו תקין, אלא אם נכתב אחרת בשאלה.
- על כל סעיף ניתן לענות "איני יודע/ת את התשובה"; על סעיף זה יינתנו 20% מהנקודות. במקרה זה אין להוסיף הסבר.
- אין להשתמש בחבילות או מודולים, אלא אם נאמר במפורש. כאשר אתם מתבקשים להשתמש בחבילה אין צורך לבצע import.
- יש לרשום את מספר תעודת הזהות ואת מספר הטופס בכל עמוד למקרה שחלק מהדפים יתלשו.
- מומלץ לקרוא כל שאלה עד סופה, על כל סעיפיה, לפני תחילת הפתרון.
- בכל מקום בו תהיה חריגה מכמות השורות המותרת יורדו עד 2 נקודות או 2/3 מהסעיף – המינימלי מביניהם.

בהצלחה!

***לשימוש הבודקים:***

| | |
|---|---|
| תעודת זהות | |
| מספר טופס | |
| שאלה 1 | |
| שאלה 2 | |
| שאלה 3 | |
| ציון | |

## שאלה 1 (אלגוריתמים ורקורסיה) – 35 נקודות.

**בסעיפים בהם נדרש פתרון רקורסיבי תינתן הנחיה מפורשת.**

במדינת "פייתוניה" ישנן מספר מפלגות: "pandas", "numpys", "recursives", "loopers" ועוד. כל מפלגה קיבלה מספר מסוים של מנדטים בבחירות, כך שמספר המנדטים מסתכם ל-120. חלק מהמפלגות לא מוכנות לשבת יחד בממשלה. בסעיפים הבאים תעזרו להרכיב ממשלה יציבה.

**סעיף א (7 נקודות)**

ועדת הבחירות המרכזית של "פייתוניה" שלחה לכם את הרשימות הבאות:

- רשימת **המנדטים** שכל מפלגה קיבלה בבחירות. מדובר ברשימה של tuples, כך שהאיבר הראשון בכל tuple הוא שם המפלגה (מטיפוס str), והאיבר השני הוא כמות המנדטים (מטיפוס int). לדוגמא:

```
mandates_list = [
    ("loopers", 30), ("recursives", 40), ("numpys", 25), ("mutables", 10), ("immutables", 10), ("pandas", 5)
                ]
```

- רשימת של זוגות של **מפלגות יריבות**, שלא מוכנות בשום פנים ואופן לשבת זו עם זו בממשלה. מדובר ברשימה של tuples, כך שהאיבר הראשון והשני בכל tuple הם שמות של מפלגות יריבות. לדוגמא:

```
rivalries_list = [
        ("loopers", "recursives"), ("mutables", "immutables"), ("loopers", "numpys")
                ]
```

<u>ממשו פונקציה</u> המקבלת את שתי הרשימות הנ"ל (רשימת מנדטים ורשימת יריבויות), ומחזירה מילון, שבו המפתחות הם שמות המפלגות, והערכים הם מילונים המכילים את כמות המנדטים של המפלגה ורשימת המפלגות היריבות שלה. לדוגמא , בהינתן 2 הרשימות לעיל כקלט, הפלט יהיה:

```
{
"mutables":     {"mandates": 10, "rivals": ["immutables"]},
"immutables":  {"mandates": 10, "rivals": ["mutables"]},
"loopers":       {"mandates": 30, "rivals": ["recursives", "numpys"]},
"recursives":   {"mandates": 40, "rivals": ["loopers"]},
"numpys":       {"mandates": 25, "rivals": ["loopers"]},
"pandas":       {"mandates": 5,   "rivals": []}}
}
```

```
def tuples_to_dict(mandates_list, rivalries_list):
```

**סעיף ב (7 נקודות)**

סטודנטית מימשה את הפונקציה list_without_rivals, המקבלת שלושה פרמטרים: (1) elections_results – מילון הפלט של סעיף א', (2) current_party - מחרוזת המייצגת שם של מפלגה כלשהי, (3) other_parties – רשימת מחרוזות המייצגות **שמות של מפלגות** אחרות. הפונקציה מחזירה **רשימה חדשה** המכילה את שמות המפלגות מ-other_parties **שאינן יריבות** של המפלגה current_party.

לדוגמא, בהינתן תוצאות הבחירות מסעיף א׳:

(i) עבור המפלגה הנוכחית "mutables", ועבור רשימת המפלגות ["loopers", "immutables", "recursives"] , הפונקציה תחזיר את רשימת המפלגות החדשה ["loopers", "recursives"].

(ii) עבור המפלגה הנוכחית "pandas", ועבור רשימת המפלגות ["immutables", "recursives"] , הפונקציה תחזיר את רשימת המפלגות החדשה ["immutables", "recursives"].

ניתן להניח שהמפלגה הנוכחית לא נמצאת ברשימת "המפלגות האחרות".

```
def list_without_rivals(elections_results, other_parties, current_party):
    non_rivals = ()
    for party in other_parties:
        if current_party not in election_results[party]["rivals"]:
            non_rivals = non_rivals.append(party)
    return non_rivals
```

במימוש זה קיימות שתי טעויות (באגים). סמנו **בבירור ובמדויק** בקוד היכן נפלו וענו בשתי שורות לכל היותר כיצד יש לתקנן:_____

_____

**סעיף ג (7 נקודות). יש לפתור סעיף זה בצורה רקורסיבית.**

**בסעיף זה אין להשתמש בלולאות. ניתן להשתמש בפונקציות מהסעיפים הקודמים ולהניח כי הן תקינות.**

נגדיר **קואליציה אפשרית** כרשימת מפלגות שסכום המנדטים שלהן הוא **לפחות 61**, ואף אחת מהן אינה יריבה של מפלגה אחרת באותה רשימה.

ממשו פונקציה **רקורסיבית** שמקבלת את הקלטים הבאים:

- elections_results - מילון הפלט מהפונקציה שמימשתם בסעיף א׳.
- parties_list - רשימה של מחרוזות המייצגות את שמות המפלגות מהן נרצה להרכיב ממשלה.
- קלט נוסף לבחירתכם (לא חובה).

הפונקציה תחזיר את מספר תת הקבוצות בparties_list המהוות קואליציות האפשריות.

לדוגמא:

עבור תוצאות הבחירות מהדוגמא בסעיף א׳, ורשימת המפלגות [ "pandas", "numpys", "recursives", "loopers"] הפונקציה תחזיר 2, מכיוון ש-["numpys", "recursives"] ו- ["pandas", "numpys", "recursives"] הן 2 הקואליציות האפשריות היחידות.

לעומת זאת, עבור תוצאות הבחירות מהדוגמא בסעיף א׳, ורשימת המפלגות [ "pandas", "immutables", "mutables", "numpys", "loopers"], הפונקציה תחזיר 0, מכיוון שאין אף תת רשימה שמהווה קואליציה אפשרית.

מס טופס:_____מספר ת"ז:_____

```
def num_coalition(elections_results, parties_list, _____):
```

**סעיף ד (7 נקודות)**

משרד החוץ שכר את שירותי צוות הקורס כדי לכתוב פונקציה שתעזור להרכיב משלחת דיפלומטית למדינת "ג'אווה". הצוות כתב פונקציה **רקורסיבית** שמקבלת את גודל המשלחת (מטיפוס int) ומספר חברי פרלמנט (מטיפוס int), ומחזירה את מספר המשלחות השונות שניתן להרכיב. למשל, עבור שלושה חברי פרלמנט שמתמודדים (שחר, חגי ורותם), ניתן להרכיב 3 משלחות שונות בגודל 1: (שחר), (חגי), (רותם). באופן דומה, עבור אותה קבוצה של חברי פרלמנט, ניתן להרכיב 3 משלחות שונות בגודל 2: (שחר, חגי), (שחר, רותם), (חגי, רותם), ורק קבוצה אחת בגודל 3: (שחר, חגי, רותם). **שימו לב כי אין חשיבות לסדר חברי המשלחת.** להלן מספר קלטים ופלטים נוספים של הפונקציה:

```
delegations(1, 3)→ 3
delegations(2, 4)→ 6
delegations(2, 5)→ 10
```

להלן הפתרון שהציע הצוות:

```
def delegations(delegation_size, members_of_parliament):
    if delegation_size == 0 or delegation_size == members_of_parliament:
        return 1
    if members_of_parliament < delegation_size:
        return 0
    op1=delegations(delegation_size - 1, members_of_parliament)
    op2=delegations(delegation_size, members_of_parliament - 1)
    return op1+op2
```

האם המימוש יעבוד כמצופה? אם כן, הסבירו מדוע. אחרת, סמנו את החלק/ים הדורשים תיקון וכתבו כיצד יש לתקנם:

_____

_____

**סעיף ה (7 נקודות)**

בתורת המספרים, **מחלק משותף מרבי** (**Greatest Common Divisor**) של שני מספרים שלמים הוא המספר השלם הגדול ביותר שמחלק את שניהם ללא שארית. למשל, המחלק המשותף המרבי של 12 ו־18 הוא 6.
להלן הנוסחא לחישוב המחלק המשותף המירבי בין 2 מספרים a וb:

$$\gcd(a, b) = \gcd(b, a \ modulo \ b)$$

ממשו את הפונקציה gcd המקבלת 2 מספרים שלמים ומחזירה את המחלק המשותף המירבי.

```
def gcd(a, b):
```

## שאלה 2 - תכנות מונחה עצמים (35 נקודות)

**בשאלה זו יש לבצע בדיקות תקינות על הקלט רק במקומות בהם נאמר זאת במפורש.**

**ניקוד מלא יינתן לפתרונות שיעשו שימוש נכון בתכנות מונחה עצמים לצורך צמצום שכפול קוד.**

במדינת "פייתוניה" החליטו שמעתה ואילך ינוהלו תוצאות הבחירות במערכת של תכנות מונחה עצמים. לצורך כך, הם שכרו את צוות הקורס המיומן לממש קוד שייצג את הפרלמנט הפייתוני. בסעיפים הבאים תעזרו לצוות הקורס לממש חלק מן המחלקות הדרושות למערכת.

**סעיף א (6 נקודות)**

1. צוות הקורס החל לממש את המחלקה Representative המייצגת נציג בפרלמנט, ובפרט סיפק לכם את בנאי המחלקה, לפי המפרט הבא:

חתימת הבנאי: __init__(self, name, charisma, truthfulness) .
מתודת הבנאי (constructor) תאתחל אובייקט חדש הכולל את השדות הבאים:

- name: מחרוזת המייצגת את שם הנציג.
- תכונות הנציג, עבורן **ניתן להניח שהקלט מספר שלם (int), אך יש לוודא שאינו קטן מ-0 ואינו גדול מ-10. במקרה של קלט שחורג מהטווח, יש להדפיס הודעת שגיאה מתאימה (כפי שיוצג בהמשך). בנוסף, אם המספר גדול מ-10, יש לקבוע את ערך השדה כ-10, ואם המספר קטן מ-0 יש לקבוע את ערך השדה כ-0.**
  i. charisma: מספר שלם (int) בין 0 ל-10, המייצג את מידת הכריזמה של הנציג.
  ii. truthfulness: מספר שלם (int) בין 0 ל-10, המייצג את מידת היושרה של הנציג.

להלן המימוש (השאלה בעמוד הבא):

```
def __init__(self, name, charisma, truthfulness):
        self.name = name

        if not 0 <= charisma <= 10 or not 0 <= truthfulness <= 10:
            print("Warning: all attributes must be between 0 and 10.")

        min(10, max(0, self.charisma))
        min(10, max(0, self.truthfulness))
```

האם הקוד יעבוד כמצופה? אם כן, הסבירו מדוע. אחרת, כתבו כיצד יש לתקנו:_____

_____

2. ממשו את המתודה (self)__repr__, המחזירה מחרוזת המייצגת את האובייקט בפורמט על-פי הדוגמא המוצגת בהמשך. **יש לממש מתודה זו בשורת אחת בלבד.**

דוגמה לשימוש במחלקה:

```
>>> rep1 = Representative("Harry Potter", 11, 8)
error: all attributes must be between 0 and 10.
>>> print(rep1)
Harry Potter: charisma=10, truthfulness=8
>>> rep2 = Representative("Hermione Granger", 8, 10)
>>> print(rep2)
Hermione Granger: charisma=8, truthfulness=10
>>> rep2.had_great_speech()
>>> print(rep2)
Hermione Granger: charisma=9, truthfulness=10
```

```
class Representative():
    ...
    def __repr__(self):
```

**סעיף ב (5 נקודות). יש לממש סעיף זה ב3 שורות לכל היותר.**

**תזכורת:** *"ערכו של <u>הממוצע המשוקלל</u> הוא סכום המכפלות של כל ערך במשקל שלו, מחולק בסכום המשקלים."*

הוסיפו למחלקה Representative את המתודות הבאות:

1. score(self, charisma_factor=1, truthfulness_factor=1). **יש לממש מתודה זו ב3 שורות לכל היותר.**
המתודה **תחזיר** את הממוצע המשוקלל של 2 התכונות של הנציג, כאשר הפרמטרים הנתונים כקלט הם המשקלים הרלוונטיים מטיפוס int. ערך ההחזר הוא מטיפוס float **ומעוגל ל-2 ספרות עשרוניות אחרי הנקודה.**
לדוגמא, עבור ערכי השדות charisma=10, truthfulness =8, והקלטים charisma_factor=2, truthfulness_factor=3, הפלט יחושב אופן הבא:

$$\frac{10 \cdot 2 + 8 \cdot 3}{2 + 3} = 8.8$$

2. (self, other)__lt__. **יש לממש מתודה זו בשורה אחת בלבד.**
מתודה זו תחזיר True אם הפלט של המתודה score עם ערכים דיפולטיביים (ברירת המחדל) של האובייקט ממנו הופעלה הפונקציה קטן מזה של other.

ניתן להניח כי אובייקטי ה- Representative שמהם/עליהם מופעלות המתודות תקינים.

דוגמת הרצה (בהמשך לדוגמא הקודמת):

```
>>> rep1 = Representative("Harry Potter", 10, 8)
>>> rep2 = Representative("Hermione Granger", 8, 10)
>>> print(rep1.score(2, 3))
8.8
>>> print(rep2.score(4, 1))
8.4
>>> print(rep2 < rep1)
False
```

```
class Representative():
    . . .
    def score(self, charisma_factor=1, truthfulness_factor=1):









    def __lt__(self, other):
```

**סעיף ג (9 נקודות)**

המשימה הבאה היא לממש את המחלקה Party המייצגת מפלגה.

בנאי המחלקה (self, name, representatives, has_primaries)__init__ יאתחל אובייקט חדש הכולל את השדות הבאים:

- name – מחרוזת המכילה את שם המפלגה.
- has_primaries – שדה בוליאני שמעיד אם קיימות בחירות פנימיות במפלגה.
- representatives – רשימת אובייקטים מטיפוס Representatives. אם קיימות בחירות פנימיות במפלגה, הרשימה השמורה בשדה צריכה להיות מסודרת לפי פלט הפונקציה score של האובייקטים עם הערכים הדיפולטיבים בסדר יורד. אחרת, הרשימה תישאר בסדר שבה היא התקבלה. **ניקוד מלא יתקבל על מימוש ללא שכפול קוד.**

<u>דוגמה לשימוש במחלקה:</u>

```
>>> rep1 = Representative("Harry Potter", 10, 5)
>>> rep2 = Representative("Hermione Granger", 8, 10)
>>> rep3 = Representative("Ron Weasley", 6, 8)
>>> griffindor = Party("Griffindor", [rep1, rep2, rep3], has_primaries=True)
>>> print([rep.name for rep in griffindor.representatives])
['Hermione Granger', 'Harry Potter', 'Ron Weasley']
```

בשלב זה צוות הקורס החליט לכתוב בעצמו גם את מתודת הבנאי של מחלקה זו. להלן המימוש:

```
class Party:
    def __init__(self, name, representatives, has_primaries):
        self.name = name
        self.has_primaries = has_primaries
        if has_primaries:
            self.representatives = representatives.sort()
        else:
            self.representatives = representatives[::-1]
```

האם הקוד יעבוד כמצופה? אם כן, הסבירו מדוע. אחרת, כתבו כיצד יש לתקנו:_____

_____

**סעיף ד (5 נקודות). יש לממש מתודה זו ב8 שורות לכל היותר.**
הוסיפו למחלקה Party את המתודה __add__(self, other), אשר תחזיר אובייקט **חדש** מסוג Party שמהווה איחוד בין המפלגות, כאשר:

- שם המפלגה החדשה מורכב מאיחוד שמות המפלגות, באופן שיוצג בהמשך.
- רשימת הנציגים מורכבת מאיחוד רשימות הנציגים של שתי המפלגות – self ו-other.
- רשימת הנציגים תורכב בשיטה של **ריץ' רץ** – הנציג הראשון במפלגה החדשה יהיה הנציג הראשון במפלגה ממנה הופעלה המתודה __add__, הנציג השני במפלגה החדשה יהיה הנציג הראשון במפלגת other, הנציג השלישי במפלגה החדשה יהיה הנציג השני במפלגה ממנה הופעלה המתודה __add__ וכו'. אם גודל המפלגות אינו זהה, השארית של המפלגה הגדולה מביניהן תתווסף לסוף הרשימה החדשה, לפי הסדר המקורי שלה.

במפלגה החדשה לא יתקיימו בחירות פנימיות.

<u>דוגמה לשימוש במחלקה:</u>

```
>>> rep1 = Representative("Harry Potter", 10, 5)
>>> rep2 = Representative("Hermione Granger", 8, 10)
>>> rep3 = Representative("Ron Weasley", 6, 8)
>>> griffindor1_house = Party("Grif", [rep1, rep2], has_primaries=True)
>>> griffindor2_house = Party("findor", [rep3], has_primaries=False)
>>> united_house = griffindor1_house + griffindor2_house
>>> print([rep.name for rep in united_house.representatives])
[Hermione Granger, Ron Weasley, Harry Potter]
>>> print(united_house.name)
Grif & findor
>>> print(united_house.has_primaries)
False
```

```
class Party:
    ...
    def __add__(self, other):
```

**סעיף ה (5 נקודות)**

**יש לממש סעיף זה בשורה אחת בלבד.** הוסיפו למחלקה Party את המתודה (self,other)__sub__, המחזירה אובייקט חדש מסוג Party שמהווה חיסור בין המפלגות:

- שם המפלגה והשדה has_primaries יישארו זהים לאלו המוגדרים באובייקט שממנו הופעלה המתודה __sub__.
- רשימת הנציגים במפלגה החדשה ,תכיל את אלו הקיימים ברשימה של האובייקט ממנו הופעלה המתודה __sub__ למעט אלו הקיימים ברשימה של other.
- סדר הנציגים ברשימה של האובייקט החדש יהיה כפי שהוגדר ברשימה של האובייקט ממנו הופעלה המתודה __sub__.

<u>דוגמה לשימוש במחלקה:</u>

```
>>> rep1 = Representative("Draco Malfoy", 8, 2)
>>> rep2 = Representative("Thomas Riddle", 10, 0)
>>> rep3 = Representative("Severus Snape", 3, 10)
>>> slytherin_house = Party("Slytherin", [rep1, rep2, rep3], False)
>>> rep4 = Representative("Lord Voldemort", 10, 0)
>>> rep5 = Representative("Bellatrix Lestrange", 9, 1)
>>> evil_party = Party("Evil", [rep2, rep4, rep5], False)
>>> slytherin_without_evil = slytherin_house - evil_party
>>> print(slytherin_without_evil.representatives)
   [Draco Malfoy: charisma=8, truthfulness=2,
   Severus Snape: charisma=3, truthfulness=10]
```

```
class Party:
    ...
    def __sub__(self, other):
```

## <u>שאלה 3 (Numpy & Image Processing, Pandas) – 35 נקודות</u>

**NUMPY**

**ניתן להניח כי לפני סעיפי החלק של Numpy הריצו את השורה**

```
import numpy as np
```

בשאלה זו נממש פעולות שונות בעיבוד תמונה.

**סעיף א (6 נקודות)**

כאשר מגדילים תמונה, יש למלא בצורה כלשהי את הפיקסלים החדשים, בעזרת הפיקסלים הישנים, כך שההגדלה תהיה טבעית ככל האפשר. אחת השיטות הבסיסיות לעשות זאת היא באמצעות עיקרון "השכן הקרוב". ערך הפיקסל בתמונה המוגדלת יקבל את ערך הפיקסל הקרוב ביותר לפיקסל הנ"ל בתמונה המקורית.

הנוסחא לפיקסל בתמונה המוגדלת תהיה:

$$big[i,j] = small\left[\lfloor\left(i \times \frac{ySmallSize}{yBigSize}\right)\rfloor, \lfloor\left(j \times \frac{xSmallSize}{xBigSize}\right)\rfloor\right]$$

כאשר small, big הן מטריצות ה-numpy של התמונה הקטנה והגדולה, ו-xBigSize למשל, הוא גודל התמונה הגדולה בציר ה-x. i,j הם האינדקסים של הפיקסלים בתמונה המוגדלת. האינדקסים יעוגלו כלפי מטה לערך השלם.

ממשו את הפונקציה nearest_enlarge(img, a) המקבלת שני פרמטרים: תמונה img (מטיפוס ndarray), וערך הגדלה a גדול מ-1 (int). הפונקציה מחזירה תמונה (מערך ndarray דו-מימדי) מוגדלת כאשר מספר הפיקסלים בכל אחד ממימדי התמונה מוגדל פי a.

דוגמת הרצה:

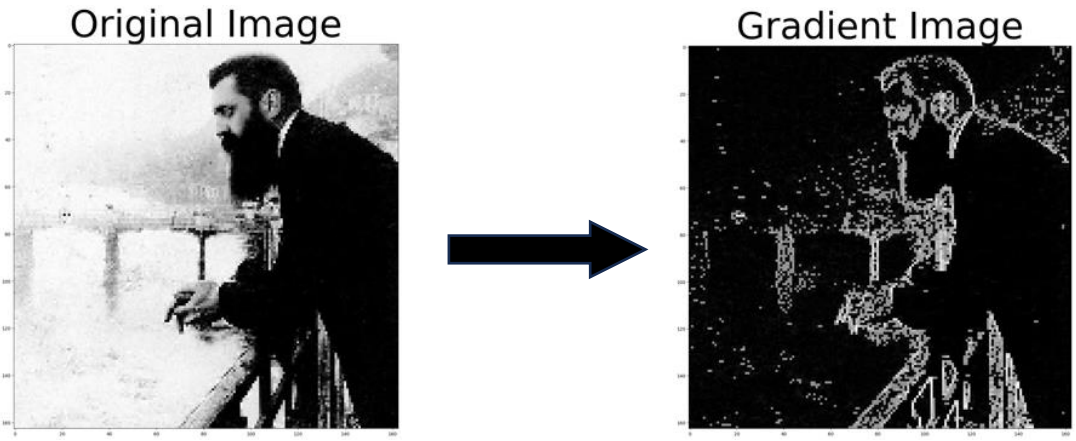
```
>>> im=np.array([[1,2],[3,4]])
nearest_enlarge(im, 2)
array([[1, 1, 2, 2],
       [1, 1, 2, 2],
       [3, 3, 4, 4],
       [3, 3, 4, 4]])
```

הסבר (לחלק מהדוגמא): ארבעת הפיקסלים השמאליים העליונים במטריצת הפלט (שערכם 1) קיבלו את ערכם מהפיקסל העליון השמאלי במטריצת הקלט.

```
def nearest_enlarge(img, a):
```

**סעיף ב (6 נקודות). ניקוד מלא יתקבל על מימוש עם לולאה אחת לכל היותר, ובנייה של כמות מערכים קבועה (שאינה תלויה ב-k).**

*תזכורת מהתרגול: בהינתן תמונה בגווני אפור, בתמונת הגרדיאנט כל פיקסל מייצג את ההפרש בין הפיקסל המקורי באותו מיקום לבין הפיקסל הבא אחריו בכיוון מסוים.*



בדומה לחישוב שראינו בתרגול, נרצה למצוא את תמונת הגרדיאנטים, אך הפעם אנחנו מעוניינים רק ב"גרדיאנטים k-יציבים" **בציר ה-x**. גרדיאנט k-יציב עבור ציר מסוים מוגדר כהפרש בין ערך הפיקסל, לבין הממוצע של k הפיקסלים שאחריו (בציר המדובר) **מעוגלים כלפי מטה**. כלומר:

$$kStableGradientX[i,j] = \left| img[i,j] - \lfloor \frac{\sum_{d=1}^{k} img[i,j+d]}{k} \rfloor \right|$$

כאשר עבור מקרי הקצה של העמודות האחרונות, כלומר $j+d \geq X\ size$ נניח שהעמודות ה"נוספות" זהות לעמודה האחרונה, כלומר:

$$img[i,j+d] = img[i,x\ size-1]$$

ממשו את הפונקציה k_stable_gradient(img) המקבלת שני פרמטרים: תמונה img (מטיפוס ndarray) ומספר טבעי k גדול מאפס (מטיפוס int), ומחזירה את תמונת הגרדיאנט היציב עבור k כלשהו.

דוגמת הרצה:

```
>>> im=np.array([[1,4,2,7],
                 [7,3,2,7],
                 [10,20,40,7]])
>>> k_stable_gradient(im,2)
array([[ 2,  0,  5,  0],
       [ 5,  1,  5,  0],
       [20,  3, 33,  0]])
```

הסבר (לחלק מהדוגמא):

- הפיקסל השמאלי העליון במטריצת הפלט (2) קיבל את ערכו מהחישוב $\left|1 - \lfloor\left(\frac{4+2}{2}\right)\rfloor\right| = 2$.
- הפיקסל השני משמאל בשורה הראשונה (0) קיבל את ערכו מהחישוב $\left|4 - \lfloor\left(\frac{7+2}{2}\right)\rfloor\right| = 0$.
- הפיקסל הימני תחתון (0) קיבל את ערכו מהחישוב $\left|7 - \lfloor\left(\frac{7+7}{2}\right)\rfloor\right| = 0$.

```
def k_stable_gradient(img, k):
```

מס טופס:_____מספר ת"ז:_____

**סעיף ג (5 נקודות)**

בהנתן הקוד להלן:

```
>>> arr = np.array([100, 150, 200], dtype=np.uint8)
>>> result = arr + 250
>>> result2 = result + 250
>>> print(max(np.max(result), np.max(result2)))
```

מה תדפיס השורה האחרונה? הסבר:_____

_____

**PANDAS**

**בכל הסעיפים הבאים אין להשתמש בלולאות. ניתן להניח כי לפני סעיפי החלק של Pandas הריצו את השורה**

```
import pandas as pd
```

גראלט מסתובב בכל רחבי היבשת במצוד אחר מפלצות המטרידות את תושבי הממלכות. הוא בוחר את יעדיו לפי חישוב של עלות מול תועלת.

במהלך השאלה, נעזר בקובץ משימות בשם missions.csv המכיל את כלל היעדים והמפלצות אותם יש לחסל. הקובץ בפורמט CSV ומכיל מידע בפורמט הבא (לדוגמא):

| Kingdom | Bounty | Ruler | Expenses | Duration | Danger level |
|---|---|---|---|---|---|
| Temeria | 1000 | Foltest | 250 | 5 | 7 |
| Redania | 1500 | Radovid | 500 | 3 | 10 |
| Kaedwen | 500 | Henslet | 100 | 7 | 3 |
| Cintra | 2500 | Calanthe | 2000 | 3 | 2 |
| Temeria | 1050 | Foltest | 50 | 1 | 7 |
| Cintra | 750 | Calanthe | 2000 | 3 | 2 |
| Kaedwen | 550 | Henslet | 200 | 3 | 3 |

העמודות Kingdom ו-Ruler מכילות מחרוזות. שאר העמודות מכילות מספרים מטיפוס int

**סעיף ד (3 נקודות). יש לממש סעיף זה בשורה אחת בלבד.**

ממשו את הפונקציה read_missions_file(file_name), המקבלת את **שם קובץ המשימות** (מטיפוס str). הפונקציה תחזיר טבלה (dataframe) עם השינויים הבאים:

- הטבלה תכיל רק את 100 השורות הראשונות בקובץ
- הטבלה לא תכיל את העמודות המייצגות את שליט הנוכחי בממלכה (Ruler) ורמת הסיכון (Danger level)

```
def read_missions_file(file_name):

```

**סעיף ה**

**1. (3 נקודות). יש לממש תת-סעיף זה בשורה אחת בלבד.**

ממשו את הפונקציה add_daily_gain_col(missions), המקבלת טבלת משימות כפי שהוגדר הפלט בסעיף ה׳, ומוסיפה **לטבלת הקלט המקורית** עמודה חדשה שתיקרא "Daily gain", המכילה את הרווח היומי הנקי בכל ממלכה. רווח יומי נקי נמדד לפי תגמול (Bounty) בניכוי ההוצאות (Expenses) חלקי מספר הימים שלוקח לבצעה (Duration), לדוגמא עבור השורה הראשונה בטבלה שבדוגמא לעיל הרווח הנקי הוא:

$$\frac{1000-250}{5}=150$$

הפונקציה לא צריכה להחזיר כלום.

```
def add_daily_gain_col(missions):

```

**2. (3 נקודות). יש לממש תת-סעיף זה בשורה אחת בלבד**.

ממשו את הפונקציה get_best_daily_gain(missions), המקבלת טבלת משימות לאחר שהופעלה עליה הפונקציה add_daily_gain_col. הפונקציה תחזיר טבלה המכילה את עמודה של שם הממלכה, ושל הBounty של המשימה בעלת ה"Daily gain" הגבוה ביותר **עבור אותה הממלכה.**

```
def get_best_daily_gain(missions):
```

**סעיף ו (3 נקודות). יש לממש סעיף זה בשתי שורות לכל היותר.**

ממשו את הפונקציה fuse_missions(dfs), המקבלת רשימה של טבלאות משימות **בפורמט שהוגדר לטבלת הפלט בסעיף ה'**. הפונקציה תחזיר טבלת משימות חדשה בפורמט זהה שמכילה את השורות של כלל הטבלאות. ניתן להניח כי לא קיימות משימות זהות בין הטבלאות. על האינדקסים בטבלה החדשה להיות יחודיים (ללא חזרות), והערכים של הטבלה יהיו ממיונים לפי גודל ה-bounty, **בסדר יורד**. ניתן להניח כי לא קיימות 2 שורות בעלי ערך Bountry זהה.

```
def fuse_missions(dfs):
```

**סעיף ז (3 נקודות)**

מה תחזיר השורה להלן? הסבר:

```
>>> df2=read_missions_file('missions.csv')
>>> df2.loc[~(df2['Duration']==df2['Duration'].max()), 'Kingdom']
```

_____

_____

מס טופס:_____מספר ת"ז:_____

מסגרת חירום 1

מס טופס:_____מספר ת"ז:_____

מסגרת חירום 2