מס' ת.ז. 205586399

מס' מחברת 38

205586399 38
03511100011

סמסטר ב' תשע"ח
מועד א', תאריך 28.06.2017

בחינה בקורס תכנות לכמאים בשפת פייתון

דביר נתנאלי, דניאל קצן

הנחיות כלליות

- משך הבחינה שלוש שעות, נצלו את הזמן ביעילות !
- חומר עזר מותר : דף A4 כתוב בשני הצדדים.
- במבחן שלוש שאלות הכוללות סעיפי משנה. כדי לקבל ציון 100 במבחן יש לענות נכון על כל השאלות. 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 הבחינה יש להשתמש כטיוטה בלבד. מחברות הבחינה לא תיבדקנה.
- יש לרשום את מספר תעודת הזהות ואת מספר הטופס בכל עמוד למקרה שחלק מהדפים יתלשו.
- מומלץ לכתוב פתרון מלא במחברת הבחינה 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.
- ניתן להניח שהקלט לקוד שתכתבו תקין, אלא אם נכתב אחרת בשאלה.
- אין להשתמש בחבילות או במודולים, אלא אם נאמר במפורש. כאשר אתם מתבקשים להשתמש בחבילה אין צורך לבצע import.

בהצלחה!

לשימוש הבודקים:

תעודת זהות	
מספר מחברת	
שאלה 1	32
שאלה 2	34
שאלה 3	30
ציון	96 כל הכבוד !

כל הזכויות שמורות ©
מבלי לפגוע באמור לעיל, אין להעתיק, לצלם, להקלים, לשדר, לאחסן במאגר מידע, בכל דרך שהיא, בין מכנית ובין אלקטרונית או בכל דרך אחרת כל חלק שהוא מטופס הבחינה.

מס' ת.ז. 205586399
מס' מחברת 38

שאלה 1 – אלגוריתמים (34 נקודות)

סעיף א' (8 נקודות)

כתבו בכל אחד מהמלבנים הבאים מה יודפס למסך בעקבות הרצת קטע הקוד שלפניו, והסבירו במשפט אחד קצר למה הודפס מה שהודפס.

1. (4 נקודות)

```
lst1 = range(5)
lst2 = lst1[::-1]
for n in lst1:
    if n != lst2[n]:
        print n,
```

יודפס באותו שורה בלי הנסיון

-1

0
1
3
4

הסבר: lst1 הוא: [0,1,2,3,4], lst2 כולל את כל אבר. lst2 בסדר הפוך, כלומר: [4,3,2,1,0]
כל איבר n שמגיע בנשמה lst2 בנקודה n, מובטח שלא כל אבר חוץ מ-2.

2. (4 נקודות)

```
def swap1(a, b):
    c = a
    a = b
    b = c

def swap2(lst, i, j):
    c = lst[i]
    lst[i] = lst[j]
    lst[j] = c

lst = [1, 2, 3, 4, 5]
swap1(lst[1], lst[2])
swap2(lst, 3, 4)
print lst
```

✓

[1,2,3,5,4]

הסבר: הפונקציה 1 swap מופעלת ע"י (2,3) כ- ארגומנט ולכן לא כלום לא מבוצע שום שינוי, הפונקציה swap פועלת ע"ל המשתנה שהינו אלמנטים לא בלבד שלהם.

מס' ת.ז. 205586399
מס' מחברת 38

סעיף ב' (10 נקודות)

ממשו את הפונקציה **calc_atom_freq (atoms)** המקבלת רשימה של מחרוזות בשם atoms המייצגת את סימולי האטומים המרכיבים מולקולה כלשהי. הפונקציה תספור כמה פעמים מופיע כל סימול אטום ברשימה ותחזיר רשימה של סימולי האטום השונים (ללא חזרות) אשר הופיעו לפחות פעמיים.

שימו לב: רשימת סימולי האטום המוחזרת תהיה ממוינת בסדר יורד לפי מספר הפעמים שהופיע כל סימול אטום.

דוגמת הרצה:

```
>> atoms = ['C','O','C','H','C','O','C','H','H','Ca','Na']

>> calc_atom_freq(atoms)

['C', 'H', 'O']
```

הסבר: ברשימה המוחזרת יופיעו סימולי האטומים אשר הופיעו לפחות פעמיים ברשימת הקלט, כאשר הם ממוינים בסדר יורד על פי מספר ההופעות שלהם. בדוגמא זו C הופיע 4 פעמים ברשימת הקלט ולכן יופיע ראשון ברשימה המוחזרת, ו-O הופיע פעמיים ולכן יופיע אחרון. Ca ו-Na הופיעו רק פעם אחת ברשימת הקלט ולא לא יכללו ברשימה המוחזרת.

```
def calc_atom_freq (atoms):
    d={}
    for atom in atoms:
        if atom not in d.keys():
            d[atom]=1
        else:
            d[atom]+=1
    res=sorted(d.keys(),key=d.get,reverse=True)
    for atom in res:
        if d[atom]<2:
            res.remove(atom)
    return res
```

לא משנים מילון בזמן ריצה עליו

צריך להרוס את מיון הלולאה

-1

מס' ת.ז. 205786377
מס' מחברת 38

סעיף ג' (16 נקודות)

1. (6 נק') ממשו את הפונקציה **unzip(lst)** שמקבלת רשימה "מכווצת" של מספרים ופורשת אותה.

רשימת הקלט מורכבת מזוגות של מספרים שלמים וחיוביים (tuples באורך 2) כאשר האיבר הראשון מציין ערך כלשהו, והאיבר השני מציין כמה פעמים על הערך לחזור על עצמו ברצף ברשימה הפרושה.

דוגמאות הרצה:

```
>> unzip([(1, 4), (7, 2), (9, 3)])
[1, 1, 1, 1, 7, 7, 9, 9, 9]
>> unzip([(7, 1), (5, 1), (7, 1)])
[7, 5, 7]
>> unzip([])
[]
```

```
def unzip(lst):
    res=[]
    for tup in lst:
        for i in range(tup[1]):
            res.append(tup[0])
    return res
```

מס' ת.ז. 205586317
מס' מחברת 38

2. (10 נק') ממשו את הפונקציה **zip(lst)** שמקבלת רשימת מספרים ומחזירה רשימה "מכווצת" בה כל רצף של ערכים זהים ברשימת המקור מוחלף בזוג (tuple) בו המספר הראשון מציין את הערך והמספר השני מציין את מספר החזרות בו הופיע הערך ברצף ברשימת הקלט.

דוגמאות הרצה:

```
>> zip([1,1,1,1,7,7,9,9,9])
[(1,4),(7,2),(9,3)]
>> zip([7,5,7])
[(7,1),(5,1),(7,1)]
>> zip([])
[]
```

```
def zip(lst):
    if len(lst)==0:
        return lst
    else:
        res=[]
        prev_num=lst[0]
        counter=0
        for num in lst:
            if num==prev_num
                counter+=1
            else:
                res.append((prev_num,counter))
                counter=1
                prev_num=num
        res.append((prev_num,counter))
        return res
```

מס' ת.ז. 205586300
מס' מחברת 38

שאלה 2 – OOP (34 נקודות)

בשאלה זו נממש חלק ממערכת לניהול הזמנות עבור בית דפוס.
ניקוד מלא יינתן לפתרונות שיכללו תכנון נכון של המחלקות הנדרשות ויעשו שימוש נכון בתכנות מונחה עצמים לצורך צמצום שכפול קוד.

סעיף א' – 24 נקודות

בית הדפוס מאפשר לבצע הזמנה של שני סוגי הדפסות: פוסטר (Poster) ומכתב (Letter).

ממשו את המחלקות PosterOrder ו-LetterOrder על פי הפירוט הבא, **ותוך שימוש במחלקת בסיס משותפת (שיהיה עליכם להגדיר בעצמיכם)** שתאפשר מניעת שכפול קוד ככל שניתן.

PosterOrder

המחלקה **PosterOrder** תייצג הזמנה של הדפסת **פוסטר**. המחלקה תכלול את המתודות הבאות:

1. בנאי המחלקה __init__(self, client_name, copies, size) יקבל את הפרמטרים הבאים וישמור כל אחד מהם לשדה:
   1.1. client_name – מחרוזת המייצגת את שם הלקוח.
   1.2. copies – מספר שלם המציין את מספר העותקים.
   1.3. size – זוג מספרים שלמים (tuple) המציינים את אורך ורוחב הפוסטר במטרים.

   טיפול בשגיאות (מלבד בדיקות אלו ניתן להניח שהקלט תקין):
   - יש להעלות שגיאה מסוג ValueError (עם ההודעה "Invalid input") אם המחרוזת המציינת את שם הלקוח ריקה או אם מספר העותקים קטן מ-1.
   - יש להעלות שגיאה מסוג ValueError (עם ההודעה "Invalid poster size") אם הפרמטר size אינו באורך 2 או אם אחד מאיבריו קטן מ-1.
2. המתודה __repr__(self) תחזיר מחרוזת המייצגת את פרטי האובייקט על פי דוגמת הפלט שבהמשך.
3. המתודה calc_cost(self) תחזיר את עלות עבודת ההדפסה בשקלים על פי הנוסחה הבאה: מספר העותקים * שטח הפוסטר (אורך* רוחב) * 30 ש"ח.

LetterOrder

המחלקה **LetterOrder** תייצג הזמנה של הדפסת **מכתב**. המחלקה תכלול את המתודות הבאות:

1. בנאי המחלקה __init__(self, client_name, paper_type, paper_prices) יקבל את הפרמטרים הבאים וישמור כל אחד מהם לשדה:
   1.1. client_name – מחרוזת המייצגת את שם הלקוח.
   1.2. copies – מספר שלם המציין את מספר העותקים.
   1.3. paper_type – מחרוזת המייצגת את סוג הדף.
   1.4. paper_prices – מילון הממפה לכל סוג דף (מחרוזת) את מחיר הדף (מספר שלם).

טיפול בשגיאות (מלבד בדיקות אלו ניתן להניח שהקלט תקין):

- יש להעלות שגיאה מסוג ValueError (עם ההודעה "Invalid input") אם המחרוזת המציינת את שם הלקוח ריקה או אם מספר העותקים קטן מ-1.
- יש להעלות שגיאה מסוג ValueError (עם ההודעה "Invalid paper type") אם המחרוזת paper_type אינה כלולה במפתחות המילון paper_prices.

2. המתודה __repr__(self) תחזיר מחרוזת המייצגת את פרטי האובייקט לפי דוגמת הפלט שבהמשך.
3. המתודה calc_cost(self) תחזיר את עלות עבודה ההדפסה בשקלים על פי הנוסחה הבאה: מספר העותקים * המחיר לדף על פי סוג הדף (כפי שמוגדר במילון paper_prices).

דוגמאות הרצה:

```
poster1 = PosterOrder ("Dvir", 1, (1, 1))
print poster1
print poster1.calc_cost()
```

פלט:

```
Poster order: Client name: Dvir, Copies: 1, Size: (1, 1)
30
```

```
paper_prices = {"A4 80gr": 3, "A5 100gr": 5}
letter1 = LetterOrder ("Evgeny", 1, "A4 80gr", paper_prices)
print letter1
print letter1.calc_cost()
letter2 = LetterOrder ("Dor", 1, "A5 100gr", paper_prices)
print letter2
print letter2.calc_cost()
letter3 = LetterOrder ("Dor", 5, "A5 100gr", paper_prices)
print letter3
print letter3.calc_cost()
```

פלט:

```
Letter order: Client name: Evgeny, Copies: 1, Paper type: A4 80gr
3
Letter order: Client name: Dor, Copies: 1, Paper type: A5 100gr
5
Letter order: Client name: Dor, Copies: 5, Paper type: A5 100gr
25
```

<u>שימו לב:</u> בשני העמודים הבאים עליכם לכתוב מימוש למחלקות **PosterOrder** ו-**LetterOrder** כך שיפעלו בדיוק על פי ההוראות ודוגמאות ההרצה שהובאו לעיל.

התחילו במימוש על פי שיקול דעתכם של מחלקת בסיס בשם **PrintOrder** אשר לא תשמש אותנו ליצירה של אובייקטים בצורה ישירה (ולכן אינה חייבת לממש את כל המתודות שהוזכרו), אבל תאפשר מימוש של המחלקות הנ"ל תוך שימוש בירושה לצורך מניעת שכפול קוד <u>ככל שניתן</u>.

מס' ת.ז. 325858339

מס' מחברת 38


```
class PrintOrder:
    def __init__(self,client_name, copies):
        if client_name=="" or copies<1:
            raise ValueError('Invalid input')

        self.client_name=client_name
        self.copies=copies
    def __repr__(self):
        r=' order: Client name: '+self.client_name+', Copies: '
        r+=str(self.copies)+', '
        return r


class PosterOrder(PrintOrder):
    def __init__(self,client_name,copies,size):
        if len(size)==2:
            if size[0]<1 or size[1]<1:
                raise ValueError('Invalid poster size')

        else:
            raise ValueError('Invalid poster size')
        PrintOrder.__init__(self, client_name, copies)
        self.size = size
    def __repr__(self):
        r= 'Poster'+PrintOrder.__repr__(self)+'Size: '+str(self.size)
        return r
    def calc_cost(self):
        r= self.copies*self.size[0]*self.size[1]*30
        return r
```

עדיף להגדיר מ PrintOrder את הפונקציה calc_cost(self) שתחזיר את מספר העותקים כפול
כי גם חלק מהשדות הרלוונטיים בתוך calc_cost בין שזו המלצה חזקה עם להשקעה בה אבל מחלקה
להתנהג את תלוית בפונקציה מהPrintOrder כמובן ומה אם מגיעים שהמחיר שונה כי מאז לעתיד
אין צורך בהגדרה מחדש כלו כי PrintOrder קל מאוד בשינוי קוד מאוחר.

מס' ת.ז. 201586377
מס' מחברת 38


```
class LetterOrder(PrintOrder):
    def __init__(self, client_name, copies, paper_type, paper_prices):
        if paper_type not in paper_prices.keys():
            raise ValueError('Invalid paper type')
        PrintOrder.__init__(self,client_name, copies)
        self.paper_type=paper_type
        self.paper_prices= paper_prices

    def __repr__(self):
        r='Letter'+PrintOrder.__repr__(self)+'Paper type:'
        r+=self.paper_type
        return r
    def calc_cost(self):
        r=self.copies*self.paper_prices[self.paper_type]
        return r
```

24

מס' ת.ז. 205586338
מס' מחברת 38

סעיף ב' – 10 נקודות

ממשו את המחלקה **PrintShop** שתייצג בית דפוס על פי הפירוט הבא:

1. בנאי המחלקה (self)__init__ לא יקבל פרמטרים כלשהם, אך יאתחל את השדות הבאים:
   1.1. orders – רשימה ריקה שתחזיק אובייקטים מסוג PosterOrder או LetterOrder ותייצג את תור עבודות ההדפסה הממתינות לביצוע.
   1.2. revenues – שדה מסוג מספר שלם המאותחל ל-0 ומייצג את הכנסות בית הדפוס החל מרגע אתחול האובייקט.
2. המתודה add_order(self,order) תקבל אובייקט בשם order המייצג עבודת הדפסה ותוסיף אותו לסוף רשימת ההזמנות הממתינות לביצוע בשדה orders.
3. המתודה print_next_order(self) מבצעת את עבודת ההדפסה הבאה בתור (זו שהוכנסה הכי מוקדם לרשימת ההזמנות) על ידי הוספת עלות ההזמנה לשדה revenues והסרת ההזמנה מרשימת ההזמנות שמאוחסנת בשדה orders.
4. המתודה (self)__repr__ תחזיר מחרוזת המייצגת את פרטי אובייקט בית הדפוס כפי שמוצג בדוגמת ההרצה שבהמשך.

שימו לב: המחרוזת המוחזרת תפרט כמה הזמנות סה"כ ממתינות לביצוע ברשימה שבשדה orders <u>עבור כל לקוח</u> (אין חשיבות לסדר בו יופיעו שמות הלקוחות). היעזרו במילון!

דוגמת הרצה תוך שימוש באובייקטים שנוצרו בדוגמא של סעיף א':

```
print_shop = PrintShop()
print_shop.add_order(poster1)
print_shop.add_order(letter1)
print_shop.add_order(letter2)
print_shop.add_order(letter3)

print print_shop

print_shop.print_next_order() #prints poster1 whose cost is 30
print print_shop
```

הפלט:

```
Print shop orders:
Evgeny : 1 orders
Dor : 2 orders
Dvir : 1 orders
Revenues: 0

Print shop orders:
Evgeny : 1 orders
Dor : 2 orders
Revenues: 30
```

מס' ת.ז. 205586355

מס' מחברת 38


```
class PrintShop:
    def __init__(self):
        self.orders=[]
        self.revenues=0
    def add_order(self, order):
        self.orders.append(order)
    def print_next_order(self):
        self.revenues += self.orders[0].calc_cost()
        self.orders.pop(0)
    def __repr__(self):
        r='Print shop orders:\n'
        d={}
        for order in self.orders:
            if order.client_name in d.keys():
                d[order.client_name]+=1
            else:
                d[order.client_name]=1
        for client in d.keys():
            r+= client+' : '+str(d[client])+' orders\n'
        r+='Revenues: '+str(self.revenues)
        return r
```

10

מס' ת.ז. 205586311
מס' מחברת 38

שאלה 3 – IO ו- NUMPY (32 נקודות)

סעיף א' (16 נקודות)

במהלך ניסוי שנערך במעבדה לתרמודינמיקה, העבירו זרם חשמלי בו-זמנית במספר מבחנות המכילות תרכובות מסוגים שונים ולאחר מכן מדדו פעם בשעה את הטמפרטורה של כל אחת מהתרכובות בעזרת חיישנים דיגיטליים.

תוצאות הניסוי נשמרו בקובץ בשם compound_temps.csv שהינו קובץ CSV (השדות בכל שורה מופרדים על ידי פסיק) שנראה כך:

```
Compound#,T1,T2,T3,T4,T5,T6,T7\n
Compound1,35,30,25,20,15,10,10\n
Compound2,34,33,32,31,30\n
Compound3,28,27,26\n
Compound4,30,30,30,30,30\n
Compound5,25,30,25,20,25\n
Compound6,25,25,25,25,25\n
```

תיאור פורמט הקובץ:

- השורה הראשונה בקובץ (החל מהשדה השני) כוללת את שמות נקודות הזמן. לדוגמא 'T1' הוא שמה של נקודת הזמן הראשונה שהתקיימה שעה אחת לאחר העברת הזרם החשמלי.
- השדה הראשון בכל שורה (החל מהשורה השנייה) מציין את שם התרכובת.
- הערכים המספריים בטבלה מציינים את הטמפרטורה במעלות צלזיוס של תרכובת מסוימת בנקודת זמן מסוימת. למשל – הטמפרטורה של התרכובת Compound3 בנקודת הזמן T2 (שעתיים לאחר העברת הזרם) הייתה 27 מעלות צלזיוס.

<u>שימו לב</u>: **עבור כל תרכובת עשוי להופיע בקובץ מספר שונה של מדידות טמפרטורה**, אך עבור כל תרכובת תופענה בהכרח לפחות 3 מדידות (המדידות תופענה תמיד ברצף החל מ-T1). שורת הכותרת תכיל את שמות נקודות הזמן עבור המספר המקסימלי של נקודות זמן שנמדדו עברו תרכובת כלשהי.

ממשו את הפונקציה **load_temps(filename, n)** אשר מקבלת מחרוזת filename אשר מציינת את שם קובץ הנתונים, ומספר שלם n <u>הגדול מ-3</u> ומציין את מספר מדידות הטמפרטורה שנדרש עבור כל תרכובת כדי שתיכלל במערכים המוחזרים על ידי הפונקציה.
הפונקציה תקרא את קובץ הקלט תוך שהיא מתעלמת משורות בהן יש מספר מדידות טמפרטורה השונה מ-n, ותחזיר 3 מערכי Numpy על פי הפירוט הבא:

1. מערך חד-מימדי של מחרוזות המכיל את n שמות נקודות הזמן הראשונות כפי שהן מופיעות בשורה הראשונה בקובץ (החל מהשדה השני).
2. מערך חד-מימדי של מחרוזות המכיל את שמות התרכובות על פי הסדר בו הן מופיעות בעמודה הראשונה בקובץ (החל מהשורה השניה). **יש לכלול רק שמות של תרכובות שעבורן מופיעות בקובץ בדיוק n מדידות טמפרטורה**.
3. מערך דו-מימדי (מטריצה) של שלמים המכיל את הנתונים המספריים בקובץ שמציינים את נתוני מדידות הטמפרטורה. במטריצה זו יהיו n עמודות בדיוק המייצגות את n נקודות הזמן

מס' ת.ז. 205586345
מס' מחברת 38

הראשונות בקובץ. מספר השורות תלוי במספר התרכובות בקובץ עבורן יש בדיוק n מדידות טמפרטורה.

- ✓ ניתן להניח שבקובץ תהיה לפחות שורה אחת בעלת n מדידות טמפרטורה.
- ✓ ניתן להניח שפורמט הקובץ תקין, אבל במקרה של שגיאה מסוג IOError יש לסגור את הקובץ כנדרש, להדפיס הודעת שגיאה כלשהי למסך ולהחזיר None.

דוגמת הרצה על קובץ הנתונים שניתן כדוגמא לעיל:

```
compounds, timepoints, data = load_temps('compound_temps.csv',5)

print compounds
print timepoints
print data
```

פלט:

```
['Compound2' 'Compound4' 'Compound5' 'Compound6']

['T1' 'T2' 'T3' 'T4' 'T5']

[[34 33 32 31 30]
 [30 30 30 30 30]
 [25 30 25 20 25]
 [25 25 25 25 25]]
```

מס' ת.ז. 205586377

מס' מחברת 38


```
import numpy as np
def load_temps(filename, n):
    f=None
    try:
        f=open(filename,'r')
        t=f.read()
        lines = t.rstrip('\n').split('\n')        -1
        first_line=lines[0].split(',')
        time_points=np.array(first_line[1:n+1])
        data=[]
        comps=[]
        for row in lines[1:]:
            line=row.split(',')
            if len(line)==(n+1):
                data.append([])
                comps.append(line[0])
                for num in line[1:]:
                    data[-1].append(int(num))  X  -1
        data_mat=np.array(data)
        companies = np.array(comps)
        return time_points, companies, data_mat
    except IOError:
        print('Error!')
        return None
    finally:
        if f!=None:
            f.close()
```

14

מס' ת.ז. 205576309
מס' מחברת 38

סעיף ב' (16 נקודות)

קראו בקפידה את ההנחיות הבאות והשלימו את הקוד לחישוב של חיתוכים שונים על נתוני הניסוי (לאחר השמטת התרכובות שעבורן אין בדיוק n מדידות, כמתואר בסעיף א'):

- עליכם להשתמש במערכי Numpy הבאים בלבד (ללא המרה לרשימות), כפי שהוחזרו ע"י הפונקציה שמימשתם בסעיף א': compounds, timepoints, data.
- יש לכתוב קוד כללי שיפעל עבור מספר כלשהו של תרכובות או נקודות זמן שעשויים להופיע במערכים.
- בסעיף זה אין להשתמש בלולאות או ברקורסיה.

1. **(4 נק') הטמפרטורה הנמוכה ביותר שנמדדה במהלך הניסוי** (הערך המינימלי בטבלה).

פלט הקוד על בסיס הדוגמא לעיל:

```
print "Lowest measured temperature:"


print data.min()
```

```
Lowest measured temperature:
20
```

2. **(4 נק') מספר המדידות בהן הטמפרטורה היתה נמוכה מ-25 מעלות.**

פלט הקוד על בסיס הדוגמא לעיל:

```
print "Number of times a temperature lower than 25 was
measured:"


print (data<25).sum()
```

```
Number of times a temperature lower than 25 was measured:
1
```

3. **(4 נק') שמה של נקודת הזמן בה ממוצע הטמפרטורות על פני כל התרכובות היה מקסימלי.**

ניתן להניח שיש רק נקודת זמן אחת בעלת טמפרטורה ממוצעת שהינה מקסימלית.

```
print "Name of timepoint having the highest
temperature average over all compounds:"


print timepoints[data.mean(axis=0).argmax()]
```

פלט הקוד על בסיס הדוגמא לעיל:

```
Name of timepoint having the highest temperature average
over all compounds:
T2
```

16

מס' ת.ז. 205586[illegible]
מס' מחברת 38

4. (4 נק') האם קיימת בטבלה מדידת טמפרטורה שערכה נמוך מ-20 ?

```
print "Is there a measurement lower than 20 in the
table?"

print (data<20).any()
```

פלט הקוד על בסיס הדוגמא לעיל:

```
Is there a measurement lower than 20 in the table?
False
```

מס' ת.ז. 207558039
מס' מחברת 38

מסגרת חירום