בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

מבוא למדעי המחשב 67101

תרגיל 3 - לולאות

להגשה בתאריך 15/11/2017 בשעה 22:00

הקדמה

בתרגיל זה נתרגל שימוש בלולאות ומשתנים. <u>דגשים לתרגיל:</u>

- יש לכתוב את כל הקוד בתוך הקובץ ex3.py.
- **חתימות הפונקציות** (שם הפונקציה והפרמטרים שלה) בקובץ המוגש צריכות להיות זהות במדויק לחתימות המתוארת במשימות. ניתן לשנות את חתימות הפונקציות על ידי הוספת ערכים דיפולטיביים לפרמטרים של פונקציות (במקומות בהם השימוש מתאים).
- לנוחותכם, מסופק **קובץ שלד** הכולל את כל חתימות הפונקציות שיש לממש (ex3.py).
- ניתן להוסיף לקובץ המוגש **פונקציות נוספות** במידת הצורך וניתן להשתמש בשאלות מתקדמות בפונקציות שמומשו בסעיפים קודמים.
- **סגנון**: הקפידו על תיעוד נאות ובחרו שמות משתנים משמעותיים. הקפידו להשתמש בקבועים (שמות משתנים באותיות גדולות) על פי הצורך.
- שימוש ב**כלים שלא נלמדו בקורס** – ניתן להשתמש בכלים אשר לא נלמדו בקורס כל עוד הם לא מייתרים את השאלה. לדוגמה, אם המשימה באחת השאלות הייתה לממש סכום של איברים ברשימה באמצעות לולאות, ניתן לייתר את השאלה על ידי שימוש בפונקציה sum המובנית של Python כדי להשיג את התוצאה הנ"ל. במקרה של ספק, יש לשאול בפורום בצורה קונקרטית.
  a. בפתרון התרגיל **אין להשתמש ב**פונקציות – count, index, sum, במודולים של numpy וב – itertools.
- **רשימה ריקה** – היא רשימה אשר אינה מכילה איברים []. היא עדיין נחשבת רשימה ומספר האיברים בה הוא 0.
- **מחרוזת ריקה** – היא מחרוזת אשר אינה מכילה תווים: "" ו – ''. היא עדיין נחשב מחרוזת ואורכה הוא 0. מחרוזת המכילה רוח אחד " " או מספר רווחים " " היא <u>אינה</u> מחרוזת ריקה.
- שימו לב מתי יש לקבל **קלט מהמשתמש** בעזרת הפונקציה input (השאלה הראשונה בלבד) ומתי הקלט מתקבל כארגומנט בעת קריאה לפונקציה.
  a. כאשר הפונקציה לא מקבלת קלט מהמשתמש בעזרת פונקציית input, מומלץ לכתוב קטעי קוד אשר קוראים לפונקציות עם ארגומנטים מתאימים כדי לבדוק שהן עובדות (אין לכלול את קטעי קוד הבדיקה בהגשה הסופית).
- בכל שאלה מפורט **מה ניתן להניח על הקלט** - אין צורך לבצע בדיקות תקינות נוספות מעבר למפורט (כלומר, ניתן להניח כי הקלט תקין).
- בכל הסעיפים, כאשר **פונקציה צריכה להחזיר ערך**, הכוונה היא לשימוש במילה השמורה **return**, בפרט הכוונה אינה להדפיס למסך בעזרת הפונקציה print. שימו לב, **אין להדפיס למסך** כל הודעה מלבד ההודעות הנדרשות במפורש.
- בפונקציות המקבלות רשימות כקלט – **אין לשנות את רשימות הקלט**.

בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

חלק א' - קבלת קלט מהמשתמש

1. קבלת רשימה כקלט

הקדמה

הפונקציה `input` מקבלת שורת קלט מהמשתמש וממירה אותה למחרוזת אותה ניתן לשמור לתוך משתנה. לדוגמה, בעת הרצת קטע הקוד הבא התכנה תמתין לקלט מהמשתמש והקלט אותו יכניס המשתמש (לדוגמה המחרוזת Hello) יישמר לתוך המשתנה `string_from_user`. במשתנה זה ניתן להשתמש, לדוגמה על מנת להדפיס את ערכו (למסך תודפס המחרוזת Hello).

```
string_from_user = input()
print(string_from_user)
```

מימוש

- כתבו פונקציה אשר מקבלת מחרוזות רבות מהמשתמש.
- הפונקציה תקבל מחרוזות מהמשתמש עד אשר יכניס המשתמש מחרוזת ריקה.
  - שימו לב - רווח יחיד או מספר רווחים הם שונים ממחרוזת ריקה.
- הפונקציה תחזיר רשימה המכילה את כל הערכים שהמשתמש הכניס כקלט.
  - כל תא ברשימה יכיל קלט אחד בדיוק של המשתמש.
  - הקלט האחרון (המחרוזת הריקה) לא צריך להיות איבר ברשימה.
  - סידור הערכים ברשימה יהיה על פי סדר קבלתם כך שקלט המשתמש הראשון יימצא בתא ה – 0 ברשימה והקלט ה – n יימצא בתא ה – n-1 ברשימה.
- במקרה והקלט הראשון הוא מחרוזת ריקה, תחזיר הפונקציה רשימה ריקה.
- חתימת הפונקציה:

```
def create_list():
```

- בעת מימוש הפונקציה ניתן לעשות שימוש בפונקציה append.
- יש לקרוא לפונקציה input ללא מחרוזת (כלומר אין להדפיס למסך הודעה לבקשת קלט).

דוגמה לשימוש בפונקציה

עבור הקלט (*הטקסט בירוק הוא הסבר לתרגיל ולא מופיע בקלט המשתמש*):

```
Hello world
What
 # 1 Space
a nice day
3
# Enter with no input
```

הפונקציה תחזיר את הרשימה:

```
['Hello world', 'What', ' ', 'a nice day','3']
```

2. שרשור אברי רשימה למחרוזת אחת

מימוש

- כתבו פונקציה המקבלת כקלט רשימה של מחרוזות ומחזירה את שרשור (concatenation) איברי רשימה כמחרוזת אחת.
- הנחות על הקלט:
  - ניתן להניח כי קלט הפונקציה הוא רשימה.
  - לא ניתן להניח כי הרשימה מכילה ערכים.
  - ניתן להניח כי כל איבר ברשימה הוא מטיפוס str.
- חתימת הפונקציה:

```
def concat_list(str_lst):
```

- ניתן לשרשר שתי מחרוזות ע"י האופרטור +.
- לדוגמה, שימוש באופרטור + על שתי המחרוזות באופן הבא 'or'+'ange' יניב את המחרוזת 'orange'.
- עבור רשימה ריקה, תחזיר הפונקציה מחרוזת ריקה.
- במימוש אין להשתמש בפונקציה join.
- הקלט לפונקציה נשלח כפרמטר (אין להשתמש בפונקציית input כדי לקבל קלט מהמשתמש).

דוגמה לשימוש בפונקציה

עבור הקריאה (רשימת מחרוזות הנשלחת לפונקציה):

```
concat_list (['Hello world', 'What', ' ', 'a nice day','3'])
```

הפונקציה תחזיר את המחרוזת:

```
'Hello worldWhat a nice day3'
```

3. חישוב ממוצע של רשימת מספרים

מימוש

- כתבו פונקציה המקבלת רשימה של מספרים ממשיים (float) ומחזירה את הממוצע שלהם.
- חתימת הפונקציה:

```
def average(num_list):
```

- הנחות על הקלט:
  - ניתן להניח כי אם קיימים איברים ברשימה הם כולם מספרים (float או int).
  - לא ניתן להניח כי ברשימה קיימים איברים, עבור רשימה ריקה על הפונקציה להחזיר None.
- במימוש אין להשתמש בפונקציית sum.

דוגמה לשימוש בפונקציה

1. עבור הקלט:

```
[2.7, 5.3, 2.5, 0.0, 6.5]
```

פלט הפונקציה הוא:

```
3.4
```

מכיוון ש:

(2.7+5.3+2.5+0+6.5)/5=3.4

2. עבור הקלט:

[2,4]

פלט הפונקציה הוא:

3.0

3. עבור הקלט:

[]

פלט הפונקציה הוא:

None

בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

חלק ב' – לולאות ורשימות

4. הסטה מעגלית

הקדמה

תהא רשימה באורך k ו – m מספר שלם אי שלילי כלשהו. הסטה מעגלית של הרשימה ב - m מקומות, היא הזזת כל האיברים m מקומות ימינה כך שאיברים שחורגים מסוף הרשימה מושמים חזרה בתחילתה. כלומר, איבר עובר מאינדקס i לאינדקס (i+m)%k.

לדוגמה הסטה של m=1 עבור הרשימה ['a','b', 'c', 'd']:

['a','b', 'c', 'd'] → ['d','a','b', 'c']

כיוון שהאינדקסים ברשימה המקורית ממופים לאינדקסים ברשימה החדשה באופן הבא:

[0 -> 1, 1 -> 2, 2 -> 3, 3 -> 0]

פרמוטציה ציקלית של רשימה היא הסטה מעגלית (ציקלית) של רשימה ב- m מקומות עבור m כלשהו.

לדוגמה לרשימה בגודל 3 ['a', 'b', 'c'] קיימות 3 פרמוטציות ציקליות:

(1) ['a','b','c'], (2) ['c','a','b'], (3) ['b','c','a']

מימוש

- כתבו פונקציה המקבלת שתי רשימות ומחזירה True אם רשימה אחת היא פרמוטציה ציקלית של השנייה, אחרת הפונקציה מחזירה False.
- במימוש הפתרון אין לשנות את הרשימות הניתנות כקלט.
- חתימת הפונקציה:

```
def cyclic(lst1,lst2):
```

- הנחות על הקלט:
  - lst1 ו – lst2 הן רשימות, לא ניתן להניח כי הרשימות מכילות איברים.
  - לא ניתן להניח דבר על תוכן הרשימות. בפרט, טיפוס הנתונים לא ידוע מראש ולא חייב להיות זהה לכל איבר ברשימה. איברי הרשימה עשויים לחזור על עצמם.
- אם שתי הרשימות אינן באורך זהה הפונקציה מחזירה False.
- שתי רשימות ריקות נחשבות כהסטה ציקלית אחת של השנייה.

בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

דוגמה לשימוש בפונקציה

cyclic([1],[1, 2]) → False	cyclic([1, 2],[1,2] → True
cyclic([1, 2, 3],[1, 3, 2]) → False	cyclic([1, 2],[2, 1] → True
cyclic([1, 2, 3],[4, 2, 3]) → False	cyclic([1, 2, 3],[2, 3, 1] → True
cyclic([1],[5]) → False	cyclic([],[]) → True
cyclic(['dog'],['gdo']) → False	cyclic(['dog', 1, 1.5],[1.5, 'dog', 1]) → True
cyclic(['a', 'a', 'b', 'b'],['a', b', 'a', 'b']) → False	cyclic(['a', 'a', 'b'],['b', 'a', 'a']) → True

5. היסטוגרמה

הקדמה

יהא מספר n ותהא רשימת מספרים L המכילה רק מספרים **שלמים** בטווח R=[0 ... n-1]. היסטוגרמה של הרשימה היא מספור של הפעמים שכל איבר בטווח המספרים הופיע ברשימה. כלומר לכל אחד מהערכים R, כמה פעמים הוא הופיע ב – L. לפיכך, עבור n, היסטוגרמה היא רשימה באורך n (ללא תלות בתוכן של L) אשר בתא ה – i שלה מופיע מספר המציין את מספר הפעמים בו הופיע המספר i בתוך L.

מימוש

- כתבו פונקציה המקבלת רשימה של מספרים שלמים ומספר שלם n, ומחזירה את ההיסטוגרמה של הרשימה. ערך ההיסטוגרמה עבור מספרים שלא מופיעים ברשימה הוא אפס.
- חתימת הפונקציה:

```
def histogram(n, num_list):
```

- הנחות על הקלט:
  - n הוא מספר שלם (int) חיובי.
  - אין ב - list_num איברים שאינם מטיפוס int.
  - איברים ב - list_num הם אי שליליים הקטנים מ – n.
  - לא ניתן להניח כי -list_num מכילה איברים.
- במימוש הפונקציה אין להשתמש ב: מודול numpy, בפונקציה count וב - collections.Cunter.

דוגמה לשימוש בפונקציה

```
histogram (3, []) → [0, 0, 0]
histogram (3, [0]) → [1, 0, 0]
histogram (3, [1]) → [0, 1, 0]
histogram (3, [1, 2]) → [0, 1, 1]
histogram (3, [1, 2, 2]) → [0, 1, 2]
histogram (4, [1, 2, 2, 3, 3, 3]) → [0, 1, 2, 3]
histogram (4, [3, 2, 3, 1, 3, 2]) → [0, 1, 2, 3]
histogram (4, [3]) → [0, 0, 0, 1]
histogram (5, [3]) → [0, 0, 0, 1, 0]
```

בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

חלק ג' – לולאות מקוננות

השאלות בפרק זה ניתנות לפתרון על ידי שימוש בלולאות מקוננות. עם זאת, אין חובה לקודד לולאות מקוננות "באופן מפורש" ובפרט אפשר מתוך לולאה לקרוא לפונקציה אחרת אשר מריצה בעצמה לולאה.

6. פירוק לגורמים

הקדמה

על פי המשפט היסודי של האריתמטיקה לכל מספר טבעי קיים פירוק יחיד למספרים ראשוניים. כלומר כל מספר טבעי יכול להיכתב כמכפלה ייחודית של מספרים ראשוניים (ייחודית, עד כדי שינוי סדר הגורמים). לדוגמה הגורמים הראשוניים של המספר 12 הם [2, 2, 3].

דרך אחת למציאת הגורמים הראשוניים של מספר נתון – n היא על ידי מעבר סדרתי על כל המחלקים הפוטנציאליים (2 עד n) ועבור כל מחלק פוטנציאלי לבדוק כמה פעמים ניתן לחלק את n במספר זה ללא שארית.

[שאלה למחשבה (ללא ניקוד): האם ניתן להקטין את כמות המספרים הנבדקים כך שתהיה קטנה מ-n-2? איך ניתן לייעל את מציאת הגורמים?]

מימוש

- כתבו פונקציה המקבלת מספר שלם חיובי - n ומחזירה את רשימת כל הגורמים הראשוניים שלו (מהקטן לגדול).
- חתימת הפונקציה:

```
def prime_factors(n):
```

- הנחות על הקלט – ניתן להניח כי n הוא מספר שלם (int) גדול או שווה ל – 1.
- 1 מחלק את כל המספרים ואין לכלול אותו ברשימות הפלט.
  - רשימת המחלקים של המספר 1 היא רשימה ריקה.
- עבור מספרים ראשוניים – יש להחזיר את המספר עצמו.
- במקרה של ריבוי שורשים, כלומר אם מספר מסוים מחלק את n יותר מפעם אחת, יש לכלול אותו ברשימת הפלט כמספר הפעמים אותה הוא מחלק את המספר.

דוגמה לשימוש בפונקציה

```
prime_factors(1)→ []
prime_factors(2)→ [2]
prime_factors(17)→ [17]
prime_factors(6)→ [2, 3]
prime_factors(8)→ [2, 2, 2]
prime_factors(15)→ [3, 5]
prime_factors(75)→ [3, 5, 5]
```

7. מכפלה קרטזית

הקדמה

מכפלה קרטזית של שתי קבוצות היא אוסף כל הזוגות בהם איבר אחד הוא מהקבוצה הראשונה והאיבר השני מקבוצה השנייה.

המכפלה הקרטזית של כל רשימה עם רשימה ריקה, היא רשימה ריקה.

מימוש

- כתבו פונקציה אשר מקבלת שתי רשימות ומחזירה את המכפלה הקרטזית שלהן.
- חתימת הפונקציה:

```
def cartesian(lst1, lst2):
```

- הנחות על הקלט:
  - איברים ברשימה יכולים להיות מכל טיפוס, הטיפוס עשוי להיות שונה בתוך הרשימה ובין רשימות.
  - הרשימות לא חייבות להיות באותו האורך.
  - הרשימות עשויות להיות ריקות.
  - הרשימות עשויות להכיל איברים זהים בתוך הרשימה ובין הרשימות.
- פלט הפונקציה הוא רשימה של כל הזוגות האפשריים בהם האיבר הראשון הוא איבר מ – lst1 והאיבר השני הוא איבר מ – lst2.
  - פלט הפונקציה הוא רשימה של tuples באורך 2- רשימה של זוגות. tuple הוא גם אובייקט מסוג sequence בפייתון, כמו רשימות, רק שלא ניתן לשנות בו את האיברים לאחר השמה והוא משתמש בסוגריים מעוגלים: ("a", 1, "b"). ניתן לקרוא עוד בלינק.
  - סדר האיברים ברשימה החיצונית (כלומר סדר הזוגות) אינו חשוב. אך הסדר בתוך זוג כן חשוב, האיבר הראשון צריך להיות מתוך lst1 והשני מתוך lst2.

דוגמה לשימוש בפונקציה

- `cartesian([], []) →[]`
- `cartesian(['a'], []) →[]`
- `cartesian(['a'], ['x'])→[('a', 'x')]`
- `cartesian(['a', 'b'], ['x']) →[('a', 'x'), ('b', 'x')]`
- `cartesian(['a', 'b'], ['x', 'y']) →[('a', 'x'), ('b', 'x'), ('a', 'y'), ('b', 'y')]`
- `cartesian (['a', 'b', 'c'], ['x', 'y']) → [('a', 'x'), ('b', 'x'), ('a', 'y'), ('b', 'y'), ('c', 'x'), ('c', 'y')]`
- `cartesian(['Hello', 'World'], ['dog', 'cat'])→ [('Hello', 'dog'), (' Hello ', 'cat'), ('World', 'dog'), (' World ', 'cat')]`
- `cartesian(['Hello', 'World', 'dog', 'cat'], []) → []`
- `cartesian(['Hello', 1], ['Hello']) → [(1, 'Hello'), ('Hello', 'Hello')]`

- `cartesian([1, 1], [2]) → [(1, 2), (1, 2)]`

8. זוגות שווי סכום

מימוש

- כתבו פונקציה המקבלת רשימה של מספרים שלמים num_list, ומספר שלם n, ומחזירה את רשימת כל הזוגות (רשימות באורך 2) שסכומם שווה בדיוק ל – n.
- חתימת הפונקציה:

```
def pairs(n, num_list):
```

- הנחות על הקלט:
  - n הוא מספר שלם (int).
  - num_list היא רשימה של מספרים שלמים (int) השונים זה מזה (כל מספר מופיע לכל היותר פעם אחת ברשימה).
  - num_list עשויה להיות ריקה.
- במידה ואין אף זוג שסכומו n תחזיר הפונקציה רשימה ריקה.
- אין חשיבות לסדר הזוגות ברשימת הפלט ואין חשיבות לסדר האיברים בתוך כל זוג.
- אי אפשר לחבר מספר לעצמו.

דוגמה לשימוש בפונקציה

```
pairs(5, []) → []
pairs(5, [4]) → []
pairs(5, [5]) → []
pairs(4, [2]) → []
pairs(5, [4, 1]) → [[1, 4]]
pairs(5, [4, 2]) → []
pairs(5, [4, 2, 7, 10, 0]) → []
pairs(5, [4, 1, 2]) → [[1, 4]]
pairs(5, [4, 1, 5]) → [[1, 4]]
pairs(5, [4, 1, 5, 0]) → [[1, 4], [5, 0]]
pairs(6, [-1, 4, 1, 7, 2, 5]) → [[2, 4], [5,1], [7, -1]]
```

בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

חלק ד' – שאלות תאורטיות

על השאלות הבאות יש לענות באופן מילולי (באנגלית, לא בקוד) בקובץ ה – README.

הסבירו מה יחזיר **הקוד שלכם** עבור הקריאות הבאות ומדוע.

שימו לב:

- מצוינים להלן גם מקרים בהם לא נדרשתם לטפל, לכן אם התכנה קורסת או מחזירה תשובה לא נכונה זה מצב תקין.
- אין צורך להשיב מה עשוי להחזיר כל מימוש שהוא לבעיה אלא מה יחזיר הקוד אותו כתבתם אתם.

```
1. cyclic('abcd','bcda')

2. histogram(3,[1,2,3,4])

3. prime_factors(0)

4. pairs(2,[0,0,1,1,2,2])
```

בית הספר להנדסה ומדעי המחשב ע"ש רחל וסלים בנין

הוראות הגשה

עליכם להגיש את הקובץ ex3.zip (בלבד) בקישור ההגשה של תרגיל 3 דרך אתר הקורס על ידי לחיצה על "Upload file".

ex3.zip צריך לכלול את הקבצים:

1. ex3.py
2. README (כמפורט בנהלי הקורס)

הנחיות כלליות בנוגע להגשה

- הנכם רשאים להגיש תרגילים דרך מערכת ההגשות באתר הקורס מספר רב של פעמים. ההגשה האחרונה בלבד היא זו שקובעת ושתיבדק.
- לאחר הגשת התרגיל, ניתן ומומלץ להוריד את התרגיל המוגש ולוודא כי הקבצים המוגשים הם אלו שהתכוונתם להגיש וכי הקוד עובד על פי ציפיותיכם.
  - באחריותכם לוודא כי – PDF הבדיקות נראה כמו שצריך.
- קראו היטב את קובץ נהלי הקורס לגבי הנחיות נוספות להגשת התרגילים.
- שימו לב - יש להגיש את התרגילים בזמן!

בהצלחה!