

תרגיל 2: פונקציות

תאריך פרסום:	7.11.2024
תאריך הגשה:	3.2.2025
מתרגלת אחראית:	עומרי אביטל
משקל תרגיל:	2 נקודות
מטרות העבודה:	שימוש בפונקציות ועבודה עם מחרוזות
דגשים לתרגיל 2	

- במימוש תרגיל זה עליכם להשתמש אך ורק בפקודות הבאות, כלומר אין להשתמש בפקודות אחרות, אפילו אם הן נלמדו בשיעור.
- תנאים: if, elif, else, in, וכל פעולות ההשוואה שנלמדו בכיתה.
- לולאות: for, while, continue, break, in, range, וכו'.
- ניתן להשתמש במשתנים מספריים, בוליאניים, תווים, מחרוזות ורשימות. אין להשתמש בסוגי משתנים אחרים (כגון מילונים, tuple, סטים וכו').
- הפונקציות האריתמטיות (+, -, *, **) וכו' והלוגיות (and, or) וכו' של פייתון.
- מתוך הפונקציות המובנות של פייתון ניתן להשתמש רק באלו: len, chr, str, ord, int, bool, float, list, pow, type, range, min, max, print, input, sum.
- כל השיטות של String (אלא אם כן נאמר אחרת).
- השיטות pop, insert, index, append של List.
- ניתן להוסיף פונקציות עזר.
- חובה לכתוב docstring (כפי שלמדתם בתרגול) לכל פונקציה.

שאלה 1:

נגדיר ערימה של פנקייקים כערימה בה הפנקייקים מסודרים אחד מעל השני בסדר אקראי. נגדיר ערימת פנקייקים מושלמת כערימה בה הפנקייקים מסודרים בסדר יורד כך שהקטן ביותר נמצא למעלה והגדול ביותר נמצא למטה.

נגדיר ערימה של פנקייקים כרשימה של מספרים טבעיים, המספרים ברשימה מייצגים את גודלם של הפנקייקים. במקום הראשון של הרשימה (באינדקס 0) יש את הפנקייק העליון בערימה, ובמקום האחרון יש את הפנקייק התחתון בערימה.

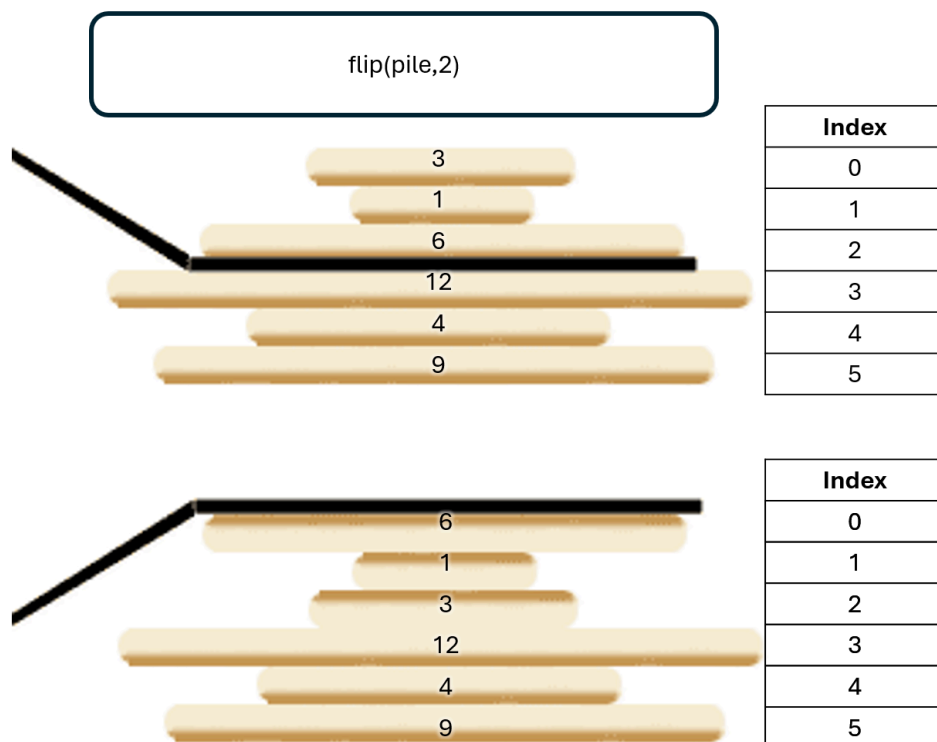
הדרך היחידה לשנות את מיקום הפנקייקים בערימה הוא להכניס את המרית בין שני פנקייקים ולהפוך את כל מה שנמצא מעליה.

סעיף א':

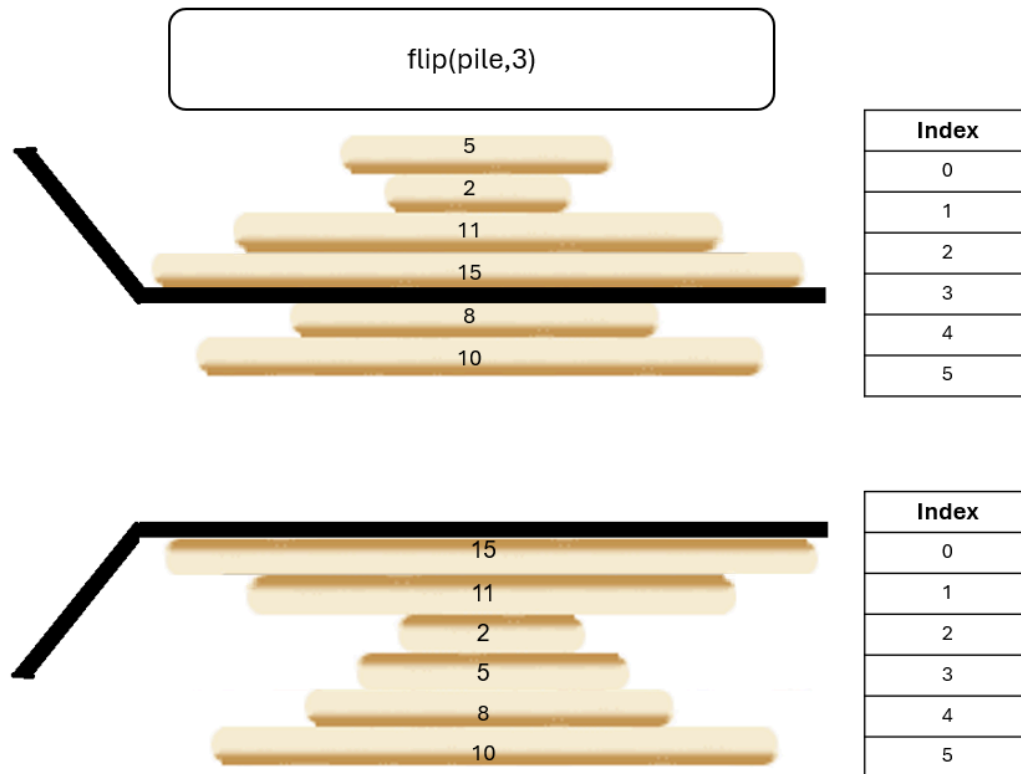
ממשו פונקציה בשם `flip(pile,index)`

הפונקציה מקבלת כקלט רשימה המייצגת ערימת פנקייקים המיוצגים כמספרים טבעיים ואינדקס (`index`) שמתחילת הערימה (אינדקס 0) ועד אליו יש להפוך את הערימה. הפונקציה מחזירה רשימה בה בוצע היפוך של תת-הערימה מתחילת הערימה ועד האינדקס שצוין (כולל).

לפניכם דוגמה של ביצוע פעולת `flip`:



דוגמה נוספת:

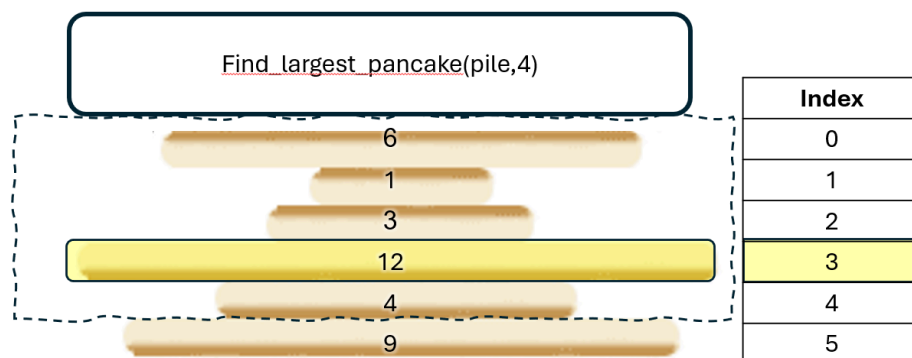


סעיף ב':

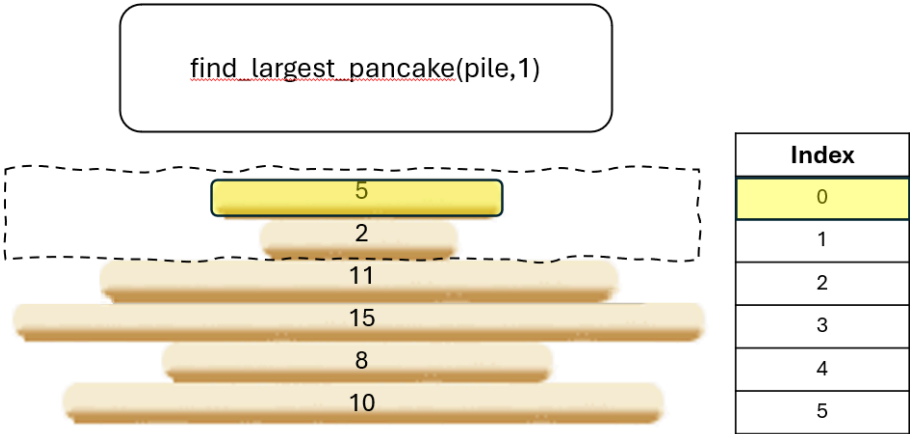
ממשו פונקציה בשם `find_largest_pancake(pile, index)`

הפונקציה מקבלת כקלט רשימה (`pile`) המייצגת ערימת פנקייקים ומספר טבעי המייצג אינדקס ברשימה. הפונקציה מחזירה מספר טבעי המייצג את האינדקס של הפנקייק הכי גדול בתת רשימה המורכבת מהפנקייקים מתחילת הערימה ועד האינדקס המצוין (`index`) (כולל אותו).

לפניכם דוגמה של ביצוע פעולת `find_largest_pancake`, בצהוב מסומנים הפנקייק הכי גדול בתת הרשימה המוקפת והאינדקס שלו:



הפעולה במקרה הזה תעבוד על תת הרשימה המוקפת ותחזיר את הערך 3.



ובמקרה הזה תעבוד הפונקציה על תת הרשימה המוקפת ותחזיר את הערך 0.

סעיף ג':

ממשו את הפונקציה pancake_sort(pile)

הפונקציה מקבלת ערימת פנקייקים המיוצגת על ידי רשימת מספרים לא ממוינת.

השתמשו בפונקציות שיצרתם בסעיפים א' וב' על מנת להפוך את הערימה לערימה מושלמת - כלומר לסדר את ערימת הפנקייקים בסדר יורד כפי שתואר בשאלה.

אין להשתמש בפונקציות מובנות של פייתון או לבצע שינויי מיקום בערימה שלא באמצעות פונקציית flip.



שאלה 2

נרצה לחלק את הפנקייקים בין סועדים בצורה שווה. נגדיר את החלוקה באופן הבא:

- כל סועד מקבל שני פנקייקים.
- כל הסועדים יקבלו פנקייקים שגודלם המצטבר שווה.

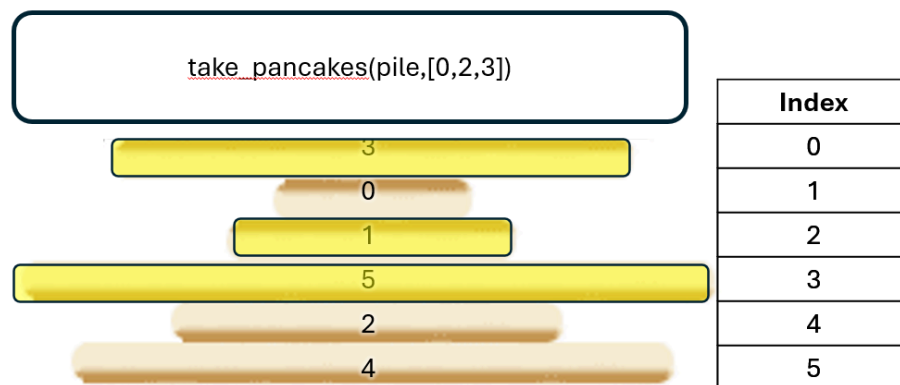
סעיף א'

ממשו את הפונקציה `take_pancakes(pile, index_list)`

הפונקציה מקבלת ערימה של פנקייקים (`pile`) כרשימת מספרים, ורשימה של מספרים טבעיים המייצגים אינדקסים (`index_list`). הפונקציה נעזרת ברשימת האינדקסים (`index_list`), מוציאה את הפנקייקים הנמצאים במקומות המתאימים מתוך `pile`, ומחזירה רשימה חדשה המכילה את גדלי הפנקייקים שהוצאו.

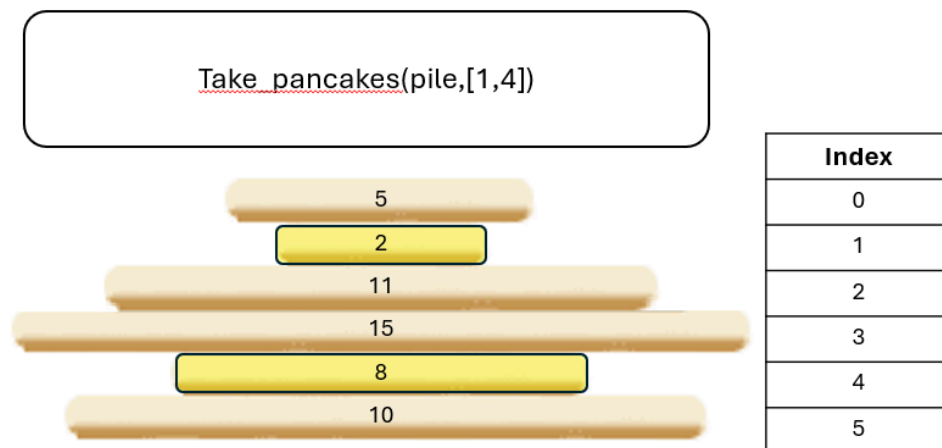
רמז: היעזרו בשיטה `pop` של רשימה

לפניכם דוגמה של שימוש בפונקציה, בצהוב מסומנים הפנקייקים באינדקסים המתאימים לאינדקסים בפרמטר `index_list` והם הפנקייקים שיילקחו מהערימה כתוצאה מהפעלת הפונקציה:



הערך שיוחזר: `[3, 1, 5]`

הערימה לאחר הפעלת הפעולה: `[0, 2, 4]`



הערך שיוחזר: [2,8]
 הערימה לאחר הפעלת הפעולה: [5,11,15,10]

סעיף ב'

ממשו את הפונקציה `insert_pancakes(pile, pancakes)`

הפונקציה מקבלת כקלט שתי רשימות של מספרים טבעיים המייצגות שתי ערימות של פנקייקים כאשר ידוע שהערימה `pile` היא ערימה מושלמת (ממוינת מהקטן לגדול).
 על הפונקציה להכניס את הפנקייקים מהערימה `pancakes` לערימה הראשית `pile` תוך שמירה על המיון בסדר יורד של הערימה הראשית `pile`.
 יש להכניס כל פנקייק למקום המתאים לו בערימה `pile`.
אין למיין את הערימה בתהליך – פתרון שממין את הערימה בתהליך ייפסל (בין אם בעזרת פונקציה מובנית של פייתון או בעזרת פונקציות שנכתבו בסעיפים קודמים)
רמז: היעזרו בשיטה `insert` של רשימה

לדוגמה עבור הרשימה

```
pile = [2,3,4,7,12]
```

והפעלת הפעולה:

```
insert_pancakes(pile,[1,6,13])
```

תתקבל התוצאה:

```
[1, 2, 3, 4, 6, 7, 12, 13]
```

סעיף ג'

ממשו את הפונקציה `divide_pancakes(pile, total_size)`

הפונקציה מקבלת כקלט `pile` רשימת מספרים טבעיים המייצגת ערימת פנקייקים ומספר טבעי `total_size` המייצג את סכום הגדלים אותו נרצה לחלק לכל סועד. על הפונקציה להחזיר רשימה מקוננת של רשימות **באורך 2** (כלומר רשימה המכילה רשימות שבכל אחת נמצאים שני פנקייקים בדיוק) כך שכל רשימה מוכלת היא ערימת פנקייקים שסכום הגדלים שלה שווה ל-`total_size`.

לדוגמה עבור הערימה:

```
pile = [2,3,4,4,5,5,6,7,8,9,12]
```

והפעלת הפונקציה עם הפרמטר 10:

```
divide_pancakes(pile,10)
```

יתקבל הפלט הבא:

```
[[7, 3], [8, 2], [6, 4], [5, 5]]
```

עבור הפעלת הפונקציה עם הפרמטר 12:

```
divide_pancakes(pile,12)
```

יתקבל הפלט הבא:

```
[[9, 3], [8, 4], [7, 5]]
```

אין משמעות לסדר של הפנקייקים בפלט.