

תכנות בסיסי חוברת להסבר ותרגול פיתון

מסמך זה נוצר במקור ע"י ד"ר אהרון ידן ועודכן ע"י ד"ר רמי רשקוביץ

הערת מבוא

החוברת נוסחה בלשון זכר מטעמי נוחיות בלבד – עמכן הסליחה!

כל ההתייחסויות לשפת הפיתוח פיתון נעשו בלשון נקבה.

חוברת זו נכתבה כדי להקל על לימוד הקורס **תכנות בסיסי** (בעיקר עבור הסטודנטים אשר נחשפים לחומר זה בפעם הראשונה). החוברת כוללת מגוון רחב של תרגילים בנושאי תכנות בשפת פיתון. במקומות רבים, בפירוט פקודות (קוד ופרמטרים) התווספו רווחים לצורך שיפור הקריאות.

פיתון – מדריך ללימוד עצמי

1. [מהו מחשב](#)
2. [שיטות מספריות](#)
3. [הסביבה האינטראקטיבית – IDLE](#)
4. [פיתון כמחשבון](#)
5. [סוגי נתונים מספריים](#)
6. [דוגמת תצרוכת דלק](#)
7. [שמות ערכים](#)
8. [משתנים](#)
9. [מחרוזות](#)
10. [עוד על סוגי נתונים](#)
11. [פונקציות מתמטיות נוספות](#)
12. [טיפול במחרוזות](#)
13. [הדפסה](#)
14. [עוד על הדפסה](#)
15. [לולאות](#)
16. [ביטויים בוליאניים](#)
17. [קלט](#)
18. [תנאים](#)
19. [רשימות](#)
20. [לולאות while](#)
21. [פונקציות](#)
22. [עוד על פונקציות](#)
23. [טיפול ברשימות](#)
24. [פעולות בוליאניות](#)
25. [פונקציות לטיפול במחרוזות](#)
26. [עיצוב מחרוזות](#)
27. [פקודות break, continue ו- else](#)
28. [טיפול בקבצים](#)
29. [הבנת תוכניות](#)
30. [מצא את החסר](#)
31. [כתיבת תוכניות](#)
32. [כתיבת פונקציות](#)
33. [רקורסיה](#)
34. [תרגילי תכנות \(ללא פיתון\)](#)

תשובות לתרגילים

1. [מהו מחשב](#)
2. [שיטות מספריות](#)
3. [הסביבה האינטראקטיבית – IDLE](#)
4. [פיתון כמחשבון](#)
5. [סוגי נתונים מספריים](#)
6. [דוגמת תצרוכת דלק](#)
7. [שמות ערכים](#)
8. [משתנים](#)
9. [מחרוזות](#)
10. [עוד על סוגי נתונים](#)
11. [פונקציות מתמטיות נוספות](#)
12. [טיפול במחרוזות](#)
13. [הדפסה](#)
14. [עוד על הדפסה](#)
15. [לולאות](#)
16. [ביטויים בוליאניים](#)
17. [קלט](#)
18. [תנאים](#)
19. [רשימות](#)
20. [לולאות while](#)
21. [פונקציות](#)
22. [עוד על פונקציות](#)
23. [טיפול ברשימות](#)
24. [פעולות בוליאניות](#)
25. [פונקציות לטיפול במחרוזות](#)
26. [עיצוב מחרוזות](#)
27. [פקודות break, continue ו- else](#)
28. [הבנת תוכניות](#)
29. [מצא את החסר](#)
30. [כתיבת תוכניות](#)
31. [כתיבת פונקציות](#)
32. [רקורסיה](#)

מהו מחשב

השלבים הראשונים בהבנת ארגון המחשב ותכנותו מחייבים הבנה של מהו מחשב. קיימות הגדרות רבות למחשבים, כמו:

- "יחידה אלקטרונית לאחסנה ועיבוד של נתונים"¹, או,
- "מכונה מתוכנתת הקולטת, מעבדת ופולטת נתונים"², וכן
- "יחידה או מערכת אשר מסוגלת לבצע רצף של פקודות באופן שהוגדר מראש. הפעולות הן לעיתים חישובים מספריים, או עיבודי נתונים, אך הן גם כוללות אפשרויות לביצוע קלט ופלט"³.

ללא כל קשר להגדרה שבשימוש, מחשב הוא מכונה אשר נועדה לבצע רצף של פעולות, מעין "מתכון" מסוים, על אוסף נתונים שגם הוא מוגדר ב"מתכון". המתכון הוא לא יותר מאשר האלגוריתם, או התוכנית שנכתבה לפתרון הבעיה. את התוכנית כותבים תוך שימוש בפקודות אשר יחד מבצעות את מה שצריך לבצע (האלגוריתם). המעגלים האלקטרוניים במחשב נבנו כך שיכירו את אוסף הפקודות ויהיו מסוגלים לבצען. פקודות המכונה הבסיסיות (פקודות סף), הן בדרך כלל פשוטות מאוד ומשמשות לביצוע פעולות פשוטות (אטומיות) כמו למשל:

- השוואה בין שני מספרים,
- בדיקה אם מספר כלשהו קטן מאפס,
- חיבור שני מספרים,
- הכפלת שני מספרים, וכד'

הפקודות הפשוטות שתוארו לעיל, הן חלק משפת תכנות בסיסית, שפת המכונה. שפה זו, שהינה ייחודית לכל מעבד, מורכבת מאוסף של סיביות (ספרות בינאריות) המגדירות את הפעולה שיש לבצע. כאמור, לכל מעבד שפת מכונה משלו, ותוכנית שנכתבה תוך שימוש בשפת המכונה למעבד אחד, לא תוכל לרוץ על מעבד אחר (אלא אם כן, המעבדים שייכים לאותה משפחה – כדוגמת מעבדי משפחת ה-PC).. השימוש בפקודות פשוטות נובע מהרצון ומהצורך לתכנן ופתוח של מחשבים שהם פשוטים ובעיקר זולים מאוד. פשטות המחשב (המעבד) והוזלת עלותו מאפשרות להניע את ה"מחזור הכלכלי", אשר אחראי במידה רבה לפיתוח המואץ בתחומי המחשוב השונים ואשר בעטיו כיום רוב המכשירים המוכרים לאדם עושים שימוש, לעיתים נרחב מאוד, במחשבים. "מחזור כלכלי" זה, מאופיין ע"י מספר שלבים המשפיעים האחד על השני. המחזור יכול להתחיל בהופעה של טכנולוגיה חדשה וזולה, המאפשרת פיתוח מחשבים/מעבדים חדשים, אמינים וזולים. תוך שימוש במחשבים/מעבדים חדשים אלה, ניתן לפתח מוצרים חדשים, ו/או יישומים חדשים. מוצרים ויישומים חדשים אלה מספקים פתרון טוב יותר לצרכים בשווקים קיימים וכן מאפשרים לפנות לפלחי שוק חדשים. הפניה לקהל חדש, יוצרת הזדמנויות שיווקיות וגורמת לפתיחה של חברות חדשות, הקמות כדי להתחרות בשווקים החדשים, ועל קהל היעד החדש, אך תוך יישום רעיונות חדשים. רעיונות חדשים לעיתים דוחפים ליצירת טכנולוגיה חדשה והמחזור הכלכלי חוזר על עצמו פעם נוספת. הכוח המניע את כל המחזור הוא כמובן כסף ולכן הוא מכונה מחזור כלכלי. דוגמה להמחשת האמור לעיל, ניתן למצוא בעולם התקשורת הסלולארית. כל הטלפונים מבוססי מעבדים, אולם הפיתוח המאסיבי, אשר לוה גם בשיווק אגרסיבי, החל רק לאחר שהטכנולוגיה הבשילה וניתן היה ליצר מעבדים שהם זולים, אמינים, בעל הספק נמוך וקטנים מספיק כדי שיוכלו להיכלל במכשיר הטלפון הנייד. ברגע שהמחזור החל, אנו עדים למרוץ מהיר מאוד של שכלולים, מכשירים חדשים בעלי יכולות חדשות, חברות חדשות וחוזר חלילה.

כאמור, עקב התפתחות הטכנולוגיה, כיום, רוב המכשירים הינם מבוססי מחשב, אפילו מכשירים ביתיים "פשוטים" כמו טלוויזיה, תנור, מקרר, מכונת כביסה, מערכת אזעקה וכד'. לעיתים, אף מדובר במכשירים זולים במיוחד, כמו למשל שעונים מבוססי מחשב שנמכרים במחיר של דולרים בודדים. השימוש הנרחב במעבדים ככוח המניע את המכשירים הרבים, מתאפשר רק בגלל העובדה שהם זמינים ובעלויות שהינן נמוכות מספיק. זו הסיבה שהמעבדים נדרשים לתמוך בפקודות פשוטות שיישומן בחומרה לא מעלה את מחיר המעבד יתר על המידה.

¹ www.micro2000uk.co.uk/hardware_glossary.htm

² education.jlab.org/beansactivity/6thgrade/vocabulary/

³ www.mcc.commnet.edu/HermesWebApp/pageFactory

תכנות בסיסי

שפת המכונה, הכוללת רצף סיביות בינארי, קשה מאוד לשימוש ע"י בני אדם ולכן פותחה שפת הסף שהינה תרגום ישיר (1:1) לשפת המכונה. במקום להשתמש בערכים מספריים לייצוג פקודות, שפת הסף מגדירה פקודות בעלות שם משמעותי.

לדוגמה:

הערך הבינארי

1011 000 0110 0100

הוא פקודה בשפת מכונה במחשב אישי רגיל (מבוסס מעבד אינטל ממשפחת ה - X86).

פקודה זו כאשר מתורגמת לשפת סף היא: MOV AL, 100

הפקודה מכניסה את הערך 100 לתוך אוגר AL שהינו חלק מהמעבד (הדוגמה תהייה ברורה הרבה יותר בהמשך, אך היא נמצאת כאן רק כדי להגיש את העובדה ששפת מכונה הינה בינארית, קשה לכתיבה והבנה ולכן פותחה שפת הסף שהינה הרבה יותר ידידותית למפתח).

הפשטות הממומשת בשפת הסף, מצד אחד, אמנם תורמת את חלקה בהורדת עלויות וקידום המחזור הכלכלי, אך מצד שני שימוש בשפת סף, עבור מפתחי תוכנה, שהינו הרבה יותר קל בהשוואה לשפת מכונה, הינו עדיין קשה ומעייף. כדי להתגבר על בעיה זו, עולם המחשוב מממש מספר רמות של הפשטה. כל רמה נבנית ומתבססת על הרמה מתחתיה, ומוסיפה יכולות ו/או הפשטה של תהליך פיתוח התוכנה. בצורה זו ניתן לגשר בין פשטות תכנון ופיתוח החומרה (כדי להוריד עלויות), לבין הצורך הגובר בכלי פיתוח מתוחכמים, שיעזרו למפתחי התוכנה לצמצם את הפער ההולך וגדל, בין צרכי המשתמשים ובין יכולות הפיתוח.

שיטות מספריות

כבר משחר ההיסטוריה, האדם נזקק למערכת שתאפשר להעריך כמויות. ואכן עם הזמן פותחו שיטות שונות שנועדו לתת מענה לבעיה של מדידת והערכת כמויות. חלק משיטות אלה היו פשוטות ואחרות מורכבות, חלקן התייחסו רק לכמויות מוחשיות (למשל הספרה אפס לא הייתה אצל המצרים הקדמונים), וחלק אחר התייחס למושג כמות, גם כאשר היא לא קיימת (הספרה אפס הייתה מוגדרת כחלק מהסימנים שבשימוש בני המאיה בדרום אמריקה). ההבנה לגבי מאפייני שיטות המספר התפתחה עם הזמן.

כל שיטת מספור חייבת להשתמש בסימנים מוסכמים ומוגדרים לתיאור הגודל (ביטוי למספרים השונים). רצוי, כמובן, שכמות הסימנים תהיה קטנה ככל האפשר, כדי להקל על בני האנוש להשתמש בהם בקלות (אך לא קטנה מדי – כפי שנראה בפרק הדין במספרים בינאריים). אוסף המספרים, כמובן, חייב לכסות את כל אפשרויות המספרים.

שיטת המספור הרומית למשל, הנהוגה לעיתים עד ימינו, הגדירה סימנים מיוחדים, כאשר לכל סימן ערך קבוע. ערך המספר מחושב מתוך ערכי הסימנים המרכיבים אותו. יתרה מזאת, סימנים זהים יכולים להתייחס לערכים שונים רק בגלל מיקום שונה. למשל IV שונה מ – VI, למרות שבשני המספרים צירוף זה של סימנים. בכל מקרה הסימן I מציין את הערך אחד והמיקום יכול להגדיר אם מדובר ההוספה של אחד או בחיסור. בניגוד ובהרחבה לשיטת המספור הרומית, שיטת המספור הנפוצה כיום מבוססת על עשרה סימנים (ספרות), כאשר ערך המספר נקבע גם על פי מיקום הספרה ולא רק על פי ערכה.

השיטה העשרונית

השיטה העשרונית (או הדצימלית), כאמור מבוססת על עשרה סימנים (ספרות). מספר עשרוני מוגדר ע"י מחרוזת של ספרות. לכל מיקום יש משקל שונה וערך המספר נקבע ע"י סיכום מכפלת המשקלות והספרות השונות.

לדוגמה:

הערך של 573_{10} מחושב ע"י:

$$5 * 10^2 + 7 * 10^1 + 3 * 10^0 = 573$$

ובאופן כללי:

$$\text{Number} = \sum_{i=0}^{p-1} d_i * b^i$$

כאשר:

- p – מגדיר את מיקום הספרה
- b – מגדיר את בסיס השיטה המספרית (במקרה זה $b = 10$)
- d – מגדיר את הספרה שמדובר בה. ($d = \{0,1,2,3,4,5,6,7,8,9\}$)

השיטה העשרונית היא הטבעית ביותר לאדם, אך קיימות שיטות נוספות המבוססות על שימוש בבסיסים אחרים. חלק מהשיטות הינן תיאורטיות בלבד, ואחרות פותחו מתוך כוונה לעזור ולהקל בעיקר בעבודה מול מחשבים.

כאמור, קיימות שיטות נוספות וגם עבורן מתאימה הנוסחה הכללית שתוארה בסעיף הקודם. יש כמובן, להחליף את הבסיס ואז מתקבל מספר אחר.

לדוגמה:

הערך של 573_8 (573 בבסיס 8 או בסיס אוקטלי) מחושב ע"י:

תכנות בסיסי

$$5 \cdot 8^2 + 7 \cdot 8^1 + 3 \cdot 8^0 = 379_{10}$$

במקרה זה 573 בבסיס אוקטלי ערכו 379 עשרוני.

ובאופן כללי:

$$\text{Number} = \sum_{i=0}^{p-1} d_i \cdot b^i$$

כאשר:

- p – מגדיר את מיקום הספרה,
- b – מגדיר את בסיס השיטה המספרית (במקרה זה $b = 8$),
- d – מגדיר את הספרה שמדובר בה. ($d = \{0,1,2,3,4,5,6,7\}$).

יש לציין שבבסיס אוקטלי למשל, מספר הסימנים השונים (הספרות) הוא רק שמונה ולכן הספרות המשתתפות במספרים אוקטליים הן: 0,1,2,3,4,5,6,7 בלבד.

באופן דומה ניתן להגדיר מספרים לפי כל בסיס נתון.

השיטה הבינארית

השיטה הבינארית (על בסיס 2) היא השיטה הפשוטה ביותר מבין כל השיטות מבוססות המקום (כפי שהוגדר בסעיף 2, שיטות בהן הערך נקבע לא רק לפי הספרה, אלא גם עפ"י מיקומה). שיטה זו גם מעניינת בכל הקשור למחשבים. אמנם המחשבים הראשונים היו מבוססים מספרים עשרוניים, אך העובדה שמדובר במעגלים אלקטרוניים הנמצאים בשני מצבים - דולק/כבוי (On/Off), גרמה לשינוי וכיום כל המחשבים המודרניים מבוססי מספרים בינאריים. השיטה הבינארית, למעשה אינה שונה משאר השיטות. גם כאן הנוסחה זהה, אלא שאם בשיטה העשרונית הבסיס הוא 10, כאן, הבסיס הוא 2 והספרות האפשריות הן רק 0 ו-1. ספרות בינאריות נקראות סיביות (סיבית – ספרה בינארית) או באנגלית (Bit - Binary Digit).

לדוגמה:

הערך של 11011_2 מחושב ע"י:

$$1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 27_{10}$$

במקרה זה 11011_2 בינארי ערכו 27 עשרוני.

ובאופן כללי:

$$\text{Number} = \sum_{i=0}^{p-1} d_i \cdot b^i$$

כאשר:

- p – מגדיר את מיקום הספרה,
- b – מגדיר את בסיס השיטה המספרית (במקרה זה $b = 2$),
- d – מגדיר את הספרה שמדובר בה. ($d = \{0,1\}$).

תכנות בסיסי

כאמור השיטה הבינארית היא השיטה הנהוגה בעולם המיחשוב, אך הבנה של מספרים גדולים תוך שימוש בשיטה הבינארית קשה עד בלתי אפשרית. השימוש בשתי ספרות בלבד גורם לכך שנדרשות הרבה ספרות כדי לבטא אפילו מספר שאינו גדול במיוחד.

למשל:

$$255_{10} = 11111111_2$$

עבור ייצוג של מספרים גדולים במיוחד, הופכת השיטה הבינארית לבלתי ישימה לחלוטין.

לדוגמה:

$$3,212,580_{10} = 11000100000010100100100_2$$

גם אם נכניס רווחים מלאכותיים (שאינם חלק מהמספר ונמצאים רק כדי להקל על העין האנושית והמוח האנושי), עדיין קשה מאוד לזכור את המספר הבינארי.

$$3,212,580_{10} = 1\ 1000\ 1000\ 00101\ 0010\ 0100_2$$

הדרך לעקוף את הבעיה היא ע"י שימוש בשיטה האוקטלית (בסיס 8), או השיטה ההקסהדצימלית (בסיס 16). השיטה ההקסהדצימלית משתמשת ב - 16 סימנים (ספרות) הכוללים את: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F. שתי השיטות בהיותן מבוססות בסיס שהינו חזקה של שתיים, מאפשרות המרה קלה ע"י קיבוץ ספרות (ראה תרשים 2-1). מספר הספרות שיש לקבץ הוא החזקה המתאימה של שתיים. (כל שלוש ספרות בינאריות הופכות לספרה אוקטלית אחת, וכל ארבע ספרות בינאריות הופכות לספרה ההקסהדצימלית אחת). הקיבוץ מתבצע תמיד מצד ימין!

לכן:

$$255_{10} = 11111111_2 = 377_8 = FF_{16}$$

ובאופן דומה:

$$3,212,580_{10} = 11000100000010100100100_2$$

$$3,212,580_{10} = 14202444_8 = 310524_{16}$$

תהליך ההמרה בין מספרים אוקטליים והקסהדצימליים לבינאריים הוא כמובן דו סטרי. כדי להפוך מספרים בינאריים יש לקבץ מספר ספרות וכדי להפוך מספרים אוקטליים או הקסהדצימליים יש להפוך כל ספרה לערך הבינארי המתאים.

תכנות בסיסי

הקסהדצימלי (16)	אוקטלי (8)	בסיס 4 (4)	דצימלי (10)	בינארי (2)
1	1	1	1	0001
2	2	2	2	0010
3	3	3	3	0011
4	4	10	4	0100
5	5	11	5	0101
6	6	12	6	0110
7	7	13	7	0111
8	10	20	8	1000
9	11	21	9	1001
A	12	22	10	1010
B	13	23	11	1011
C	14	30	12	1100
D	15	31	13	1101
E	16	32	14	1110
F	17	33	15	1111

תרשים 2-1: טבלת המרה מספרים בינאריים

שברים

בסעיפים הקודמים דובר על מספרים שלמים. שברים מחושבים באופן דומה, אלא שכאן המשקלות הן חזקות שליליות של הבסיס. המספר במקרה זה מחושב ע"י הנוסחה:

$$\text{Number} = \sum_{i=-n}^{p-1} d_i \cdot b^i$$

לדוגמה:

הערך של 675.438 מחושב ע"י

$$6 \cdot 8^2 + 7 \cdot 8^1 + 5 \cdot 8^0 + 4 \cdot 8^{-1} + 3 \cdot 8^{-2} = 445.546875_{10}$$

המרת מספרים שלמים

כאמור, המרות מבסיס בינארי לבסיס אחר שהוא חזקה של שתיים (כמו בסיס אוקטלי או הקסהדצימלי) מתבצעות בקלות ע"י קיבוץ מספר ספרות בינאריות (סיביות או ביטים). מספר הספרות שיש לקבץ הוא החזקה של הבסיס החדש. המרה הפוכה מתבצעת ע"י תרגום הספרה האוקטלית או ההקסהדצימלית לספרות הבינאריות המתאימות (ראה תרשים 2-1). ספרה אוקטלית נרשמת כשלוש סיביות ואילו ספרה הקסהדצימלית נרשמת כארבע סיביות (החזקות המתאימות של שתיים).

לדוגמה:

$$55646_8 = 101\ 101\ 110\ 100\ 110_2$$

תכנות בסיסי

או

$$\begin{aligned} 5BA6_{16} &= 0101\ 1011\ 1010\ 0110_2 \\ &= 101101110100110_2 \end{aligned}$$

המרות מכל בסיס לבסיס עשרוני (או במילים אחרות, קביעת הערך של המספר) מתבצעות על פי הנוסחה הכללית, ע"י הכפלת הספרה המתאימה בחזקת הבסיס המתאימה.

$$\text{Number} = \sum_{i=-n}^{p-1} d_i \cdot b^i$$

המרות של מספרים עשרוניים שלמים לכל בסיס מבוצעות ע"י חלוקה חוזרת של המספר העשרוני בבסיס, ואיסוף השאריות בסדר הפוך, כל זאת עד אשר התוצאה המתקבלת היא אפס.

למשל:

המרה של 37_{10} למספר בינארי מבוצעת ע"י:

$37/2 = 18$	remainder:	1	
$18/2 = 9$	remainder:	0	
$9/2 = 4$	remainder:	1	
$4/2 = 2$	remainder:	0	
$2/2 = 1$	remainder:	0	
$1/2 = 0$	remainder:	1	

The binary number is: 1 0 0 1 0 1

המרה של מספרים עשרוניים לבסיסים אחרים מתבצעת באופן דומה.

למשל:

המרה של 122_{10} לבסיס 7

$122/7 = 17$	remainder:	3
$17/7 = 2$	remainder:	3
$2/7 = 0$	remainder:	2

ולכן,

$$122_{10} = 233_7$$

תכנות בסיסי

המרה מבסיס כלשהו n , לבסיס כלשהו m (כאשר $n - m$ אינם בסיסים עשרוניים, בינאריים או חזקה של שתיים), מתבצעת ע"י המרה כפולה. בשלב ראשון, המרה לבסיס עשרוני ולאחר מכן המרה נוספת מבסיס עשרוני לבסיס m . (ניתן גם להמיר ישירות, אך המרה כפולה זו, קלה יותר).

המרת שברים

המרות של שברים מבסיס כלשהו לבסיס עשרוני (קביעת הערך של השבר) מבוצעות ע"י שימוש בנוסחה (באופן דומה להמרות של מספרים שלמים). הכפלת הספרה במקום המתאים בחזקה המתאימה.

למשל:

הערך של 0.321_4 מחושב ע"י:

$$3 \cdot 4^{-1} + 2 \cdot 4^{-2} + 1 \cdot 4^{-3} = 0.890625_{10}$$

המרות של שברים מבסיס עשרוני לבסיס כלשהו מבוצעות ע"י הכפלת השבר בבסיס, איסוף השלמים, הכפלה נוספת (רק של השבר) וחוזר חלילה. כל זאת עד אשר התוצאה המתקבלת היא 1.0 (שלם ללא שבר).

למשל:

הפיכת 0.375_{10} למספר בינארי:

$$\begin{aligned} 0.375 \cdot 2 &= 0.75 \\ 0.75 \cdot 2 &= 1.5 \\ 0.5 \cdot 2 &= 1.0 \end{aligned}$$

לכן

$$0.375_{10} = 0.011_2$$

חשוב להדגיש שיש מצבים בהם שבר עשרוני אינו ניתן להמרה פשוטה למספר בינארי ולכן, התוצאה תהיה שבר מחזורי אינסופי.

לדוגמה:

הפיכת 0.3_{10} למספר בינארי:

$$\begin{aligned} 0.3 \cdot 2 &= 0.6 \\ 0.6 \cdot 2 &= 1.2 \\ 0.2 \cdot 2 &= 0.4 \\ 0.4 \cdot 2 &= 0.8 \\ 0.8 \cdot 2 &= 1.6 \\ 0.6 \cdot 2 &= 1.2 \\ 0.4 \cdot 2 &= 0.8 \end{aligned}$$

לכן

$$0.3_{10} = 0.010011001100110011\dots_2$$

תכנות בסיסי

שאלות חזרה

1. הפוך מבסיס דצימלי (עשרוני) לבינארי

1. 765_{10} = _____₂
2. 1234_{10} = _____₂
3. 5739_{10} = _____₂
4. 1001_{10} = _____₂
5. 9173_{10} = _____₂
6. 1024_{10} = _____₂
7. 2047_{10} = _____₂
8. 15365_{10} = _____₂
9. 999_{10} = _____₂
10. 1010_{10} = _____₂

2. הפוך מבסיס בינארי לדצימלי (עשרוני)

1. 1100011_2 = _____₁₀
2. 11000111_2 = _____₁₀
3. 100101011_2 = _____₁₀
4. 100011_2 = _____₁₀
5. 1101111_2 = _____₁₀
6. 10011010010_2 = _____₁₀
7. 1011000_2 = _____₁₀
8. 10101111_2 = _____₁₀
9. 10010000_2 = _____₁₀
10. 100100000_2 = _____₁₀

3. הפוך מדצימלי לבסיס המתאים

1. 4759_{10} = _____₁₆
2. 4759_{10} = _____₈
3. 918273_{10} = _____₁₆
4. 918273_{10} = _____₈
5. 117_{10} = _____₄
6. 555_{10} = _____₄
7. 555_{10} = _____₈
8. 555_{10} = _____₁₆
9. 777_{10} = _____₇
10. 999_{10} = _____₉
11. 123456_{10} = _____₅
12. 199_{10} = _____₃
13. 9876_{10} = _____₆
14. 998877_{10} = _____₁₁

תכנות בסיסי

15. $9753_{10} = \underline{\hspace{2cm}}_7$

4. הפוך מבסיס לבסיס על פי הרשום

1. $78_{10} = \underline{\hspace{2cm}}_7$

2. $99_{10} = \underline{\hspace{2cm}}_6$

3. $AA_{16} = \underline{\hspace{2cm}}_9$

4. $88_9 = \underline{\hspace{2cm}}_{10}$

5. $232_7 = \underline{\hspace{2cm}}_5$

6. $888_9 = \underline{\hspace{2cm}}_3$

7. $6543_7 = \underline{\hspace{2cm}}_8$

8. $1265_9 = \underline{\hspace{2cm}}_6$

9. $4343_5 = \underline{\hspace{2cm}}_9$

10. $5555_6 = \underline{\hspace{2cm}}_7$

5. השלם את הטבלה

בינארי	בסיס 3	בסיס 5	בסיס 8	עשרוני	הקסהדצימלי
				219	65
			174		
		2010			
	100200				
10101010					
	21120				
			307		
					49

6. המספרים הבאים הינם מספרים עשרוניים, בטא אותם בכל הבסיסים שלפניהם (2-9).

1. 100

2. 64

3. 77

4. 25

5. 83

7. תרגם את המספרים (נקודה קבועה)

1. $1110.1011_2 = \underline{\hspace{2cm}}_{10}$

2. $1001.0110_2 = \underline{\hspace{2cm}}_{10}$

תכנות בסיסי

3. $0110.0001_2 = \underline{\hspace{2cm}}_{10}$
4. $1111.1111_2 = \underline{\hspace{2cm}}_{10}$
5. $0.000001_2 = \underline{\hspace{2cm}}_{10}$
6. $3.5625_{10} = \underline{\hspace{2cm}}_2$
7. $9.9375_{10} = \underline{\hspace{2cm}}_2$
8. $11.55_{10} = \underline{\hspace{2cm}}_2$
9. $0.7_{10} = \underline{\hspace{2cm}}_2$
10. $14.0625_{10} = \underline{\hspace{2cm}}_2$

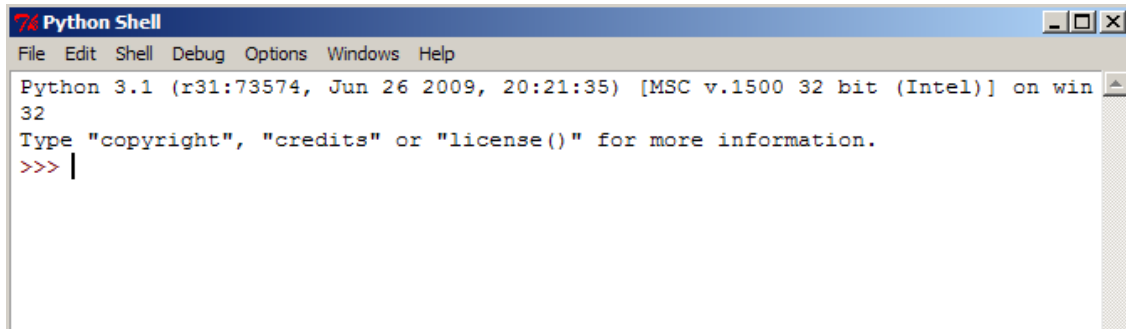
[תשובות](#)

[חזור לתוכן](#)

הסביבה האינטראקטיבית – IDLE

סביבת הפיתוח של פיתון יכולה לעבוד כ – interpreter (מפענח), משמעות הדבר שהיא מבצעת פקודות בצורה אינטראקטיבית ומציגה את התוצאה מייד לאחר הקשת הפקודה. כל התרגולים בחוברת זו מבוצעים תוך שימוש בסביבה אינטראקטיבית זו.

לכניסה לסביבה, הפעל את פיתון. ייווצר מסך דמוי:



שורת הכותרת מציינת פרטים לגבי הסביבה (כמו למשל בדוגמה זו המהדורה של שפת התכנות היא 3.1, תאריך השחרור שלה הוא יוני 2009) ועוד.

הסימון ">>>" מציין שפיתון מוכן לקבל את הפקודה הבאה שלנו.

כדי לסגור את הסביבה, אפשר ללחוץ על ה – X בקצה הימני (כמו סגירת כל חלון), או לחילופין ללחוץ על Cntrl D.

בהכנסת פקודות, רווח הוא תו בלתי נראה, כך ששפת פיתון מתעלמת ממנו.

[חזור לתוכן](#)

פיתון כמחשבון

עקב העובדה שפיתון עובדת גם ב – interpreter אפשר להשתמש בה גם כמחשבון. פירוש הדבר שאפשר להקיש שני מספרים וביניהם סימן הפעולה הנדרשת ומיידית נקבל את התשובה. סימני החשבון הבסיסיים הם: + , - , * , /. למשל:

```
>>> 2 + 7
```

```
9
```

```
>>> 3 * 5
```

```
15
```

```
>>> 16 / 4
```

```
4
```

```
>>> 12 - 9
```

```
3
```

```
>>>
```

בוודאי שמת לב לעובדה שתרגילים המתייחסים לפקודות חיבור חיסור וכפל, התשובה נרשמת כמספר שלם ואילו כאשר אנו מחלקים התשובה נרשמת כשבר. זו תכונה חדשה שהוכנסה במהדורה 3.1 של פיתון והיא נועדה בעיקר למנוע איבוד מידע, כאשר אנו מחלקים. אנו נתייחס לנקודה זו ברחבה גם בהמשך. בכל פעולה חשבונית בין שבר למספר שלם, התוצאה תהיה שבר (כדי למנוע איבוד מידע).

שפת פיתון מאפשרת גם לחשב חזקה (נרשמת ע"י **), כדוגמה

```
>>> 5 ** 2
```

```
25
```

```
>>> 2 ** 10
```

```
1024
```

```
>>>
```

סדר ביצוע הפעולות החשבוניות הוא הסדר הרגיל: קודם חזקות, אחר כך כפל ו/או חילוק ובסוף חיבור ו/או חיסור. ניתן כמובן להשתמש בסוגריים לשינוי הסדר על פי הנדרש:

```
>>> 3 + 2 ** 10 / 2 + 3
```

```
518.0
```

```
>>> 3 + 2 ** 10 / ( 2 + 3 )
```

```
207.8
```

```
>>>
```

```
>>> 3 + 2 ** ( 10 / ( 2 + 3 ) )
```

```
7.0
```

```
>>>
```

שים לב לשינוי התוצאה הנגרם ע"י השימוש בסוגריים, אשר למעשה מבצע פעולת חשבונית שונה. בנוסף, בגלל שהפעולה החשבונית כוללת בתוכה חלוקה וכפי שהוסבר לעיל, חלוקה גוררת תשובה שפורמט של שבר, כל התשובות שהתקבלו בתרגיל זה הינן שברים גם כן.

אם אנחנו רוצים לחלק שלמים, או במילים אחרות לשאול כמה פעמים ערך אחד נכנס בערך אחר (כך שהתשובה תהיה מספר שלם), ניתן להשתמש בסימן //. למשל:

```
>>> 16 // 5
```

תכנות בסיסי

3

>>>

זזה כמובן בניגוד לפעולת החילוק הרגילה:

>>> 16 / 5

3.2

>>>

אם ברצוננו לייצג מספר בפורמט של שבר, כל שיש לעשות זה להוסיף לו נקודה עשרונית. לכן, המספר 3 מבטא מספר שלם ואילו המספר 3.0 מבטא שבר. בפעולות בין מספרים שלמים התוצאה תהיה מספר שלם, כאמור למעט במקרה של חילוק. בפעולות בין שברים, או שאחד האיברים בפעולה הוא שבר התוצאה תהיה שבר.

לכן,

>>> 15 / 5

3.0

>>> 15 / 5.

3.0

>>> 15 / 5.0

3.0

>>> 15 * 5

75

>>> 15.0 * 5

75.0

>>> 15 * 5.

75.0

>>> 15. * 5.0

75.0

>>> 3 + 4.0

7.0

>>> 19 - 3.99

15.01

>>>

תרגילי חזרה

נסה להעריך את התשובה (כולל הפורמט שלה, במילים אחרות אם התשובה היא מספר שלם או שבר) לפני שתנסה את התרגיל במחשב:

1. $2 + 3 + 4 - 5 - 6 - 7 + 8 + 9$
2. $2 * 3 * 4 / 2 + 3 * 2 * 2 - 4 / 2 + 9 / 3$
3. $(2 + 2 - 3) * (4 * 3 - 2)$
4. $(2 + 2) - 3 * 4 * 3 - 2$
5. $2 + (2 - 3) * 4 * (3 - 2)$
6. $64 / 4 + 4 * 2$
7. $64 / (4 + 4) * 2$
8. $64 / ((4 + 4) * 2)$
9. $3 * 3 + 3 * 3 + 4 * 4$
10. $3 * (3 + 3) * (3 + 4) * 4$
11. $3 ** 2 + 4 / 2$
12. $3 ** (2 + 4) / 2$
13. $3 ** (2 + 4 / 2)$
14. $3 / 3 * 17 // 5$
15. $3 / 3 * 17. // 5$
16. $(1 + 2 + 3 + 4 + 5 + 6) ** 2 // 12.$
17. $(1 + 2 + 3 + 4 + 5 + 6) ** 2 / 12.$
18. $31 // 3. * 17 / 12. * 1.5$
19. $81 / ((29 // 9) * 9.)$
20. $6.25 - (1.25 * 3) // 6 + 6.25 // 1.25$
21. $125 // (25 // 5) ** 2$
22. $125 // 25 // 5 ** 2$
23. $(125 // 25 // 5) ** 2$
24. $2 ** 8 / 2 ** 6 // 2$
25. $2 ** 8 / 2 ** (6 // 2)$
26. $2 ** 8 / (2 ** 6 // 2)$
27. $((32 - 2 ** 4) ** (21 // 10)) ** (5 / 10.)$
28. $2 ** 2 ** 2 // 256 // 6$
29. $2 ** 2 ** (2 // 256 // 6)$
30. $2 ** (2 ** 2 // 256 // 6)$

[תשובות](#)

[חזור לתוכן](#)

סוגי נתונים

בפרק הקודם התייחסנו לשני סוגי נתונים מספריים, מספרים שלמים ושברים. שפת פיתון תומכת במגוון רחב של סוגי נתונים (כולל פקודות לטיפול בנתונים אלה). בפרק זה נדון אך ורק בנתונים מספריים, אולם סוגי נתונים נוספים יפורטו בהמשך. הטבלה הרצ"ב מפרטת את סוגי הנתונים המספריים הקיימים:

סוג הנתון (Data Type)	פירוט
מספרים שלמים (Integer)	מספר שלם (עם או בלי סימן)
שברים (Float)	שבר (חיובי/שלילי) בגודל 64 סיביות. מיוצג ע"י שבר עשרוני דוגמת: 2.57, או ייצוג הנדסי: 1.23×10^{-15} (אשר משמעותו הינה 1.23×10^{-15})
שלמים בבסיס בינארי	מספר שלם המוגדר ע"י אפס מוביל ואחריו האות האנגלית b. למשל 0b1001001
שלמים בבסיס אוקטלי	מספר שלם המוגדר ע"י אפס מוביל ואחריו האות האנגלית o. למשל 0o56746
שלמים בבסיס הקסהדצימאלי	מספר שלם המוגדר ע"י הקידומת 0x, למשל 0xACDF
מספרים מרוכבים	החלק הממשי והחלק המדומה נכתבים ע"י: $2+5j$, או לחילופין: $1.35 - 0.006j$

כאמור, כדי לעבוד עם שברים, גם כאשר המספרים הינם שלמים, יש לרשום את המספר בתוספת נקודה עשרונית, או שבר עשרוני (אפילו מאופס). המספר 7 מייצג מספר שלם ואילו המספר 7.0 מייצג שבר. כל הפקודות החשבונאיות, עובדות כמובן על כל סוגי הנתונים ועל כל צירוף שלהם, כפי שניתן לראות בדוגמאות הרצ"ב:

```
>>> 1.25 * 7
8.75
>>> -325.98 * 12.3 / 34.8
-115.21706896551727
>>> 0.5 ** 4
0.0625
>>> 0.5 ** ( 4 * 4 )
1.52587890625e-05
>>> 0b1010 + 0b1111
25
>>> 0b1100+35
47
>>> 0xff + 23
278
>>> 0xab + 123 + 0b10110
316
>>> 2 + 3j + 3 + 4j
(5+7j)
```

תכנות בסיסי

```
>>> 15.0 / 0b11
3.0
>>> 31. / 0x1f
1.0
>>> 0b11 ** (0x10 - 14)
9
>>> 0b11 ** (0x10 - 14)
9.0
```

שים לב שמספרים המבוטאים בצורה הנדסית הם שברים.

```
>>> 2e2*3e3
600000.0
>>> 0x77+0o77
182
>>> 0x77
119
>>> 0o77
63
>>> 0b11+0o11+0x11
29
>>> 0x77/0o77
1.8888888888888888
>>> 0x77//0o77
1
>>>>> 0x11/0b11
5.666666666666667
>>> 0x12*0o12
180
>>> 0b1111111
127
>>> 0b1111111 + 0o177 -0x80
126
```

תרגילי חזרה

נסה להעריך את התשובה לפני שתנסה את התרגיל במחשב. יש להגדיר את התשובה, כולל סוג הנתונים שלה (שלם או שבר):

1. $0b111 + 0b110011$
2. $0x12FF + 0xFF12$
3. $99 + 0b10011 + 0x77$
4. $0x66 - 66$
5. $0b110110 + 66$
6. $12e2 + 12e4$
7. $13e-3 + 13e-1$
8. $3e4 * 4e3$
9. $11e4 / 55 + 55$
10. $11e4 / (55 + 55)$
11. $7e3 * 0b1000$
12. $0.3e2 * 0x12$
13. $0.4e2 / 0.4e-2$
14. $2 * 1.5 / 3 * 1.23e-2$
15. $2e-001 * 2e-01 * 2e003$
16. $0.2 * 0.2 * 2000$
17. $12 * 12 * 2.0$

[תשובות](#)

חשוב להזכיר שפעולות המבוצעות בין מספרים שלמים גוררות תוצאה שהיא מספר שלם (פרט למקרה של חילוק שבו חילוק אפילו בין שני שלמים יגרור תוצאה שהיא שבר) ואילו פעולות בין שברים (או שברים ומספרים שלמים) גוררות תוצאה שהיא שבר.

לדוגמה:

```
>>> 13 * 15
195                מספר שלם
>>> 13.0 * 15
195.0             שבר
>>> 13 * 15.0
195.0             שבר
>>> 13.0 * 15.0
195.0             שבר

>>> 15 * 3
5.0               שבר
```

אבל,

בגלל שבפיתון לעיתים ערכים מסוימים חייבים להיות שלמים, יש לשים לב לפורמט התוצאה. אם אנו מחלקים, אפילו אם הפרמטרים הם מספרים שלמים, התוצאה תהיה שבר ולפני שנוכל להשתמש בה

תכנות בסיסי

בפקודות המשתמשות במספרים שלמים בלבד, נצטרך להמיר את השבר ולהפכו למספר שלם. עקב חשיבותו, הנושא יורחב בהמשך החוברת.

נסה להעריך את התשובה לפני שתנסה את התרגיל במחשב:

18. $5 / 3$
19. $8 / 4$
20. $8 / 4.0$
21. $15 / 4$
22. $15 / 4.0$
23. $1.0 * 15 / 5$
24. $1 * 15 / 5.0$
25. $1 / 3 * 20$
26. $1 / 3 * 20.0$
27. $2 ** 5 / 64 * 2.$
28. $2. ** 5 / 64 * 2$
29. $(0b1001 ** 2 // 10 + (2.5 * 8)) // 10$
30. $2 ** (((0x28 + 0x20) / 8 + 3) / 6)$
31. $3 + 0o11 + 0b11 + 0x11$
32. $(0o21 + 0b11) / 0x4$
33. $0b1001 + 0b1111 + 11$
34. $0x11 - 0o11 - 0b11$
35. $0b11 * 0o10 / 0x6$
36. $(25 - 0x15) ** 0b100$
37. $2 ** (0b10 + 0x1 + 0x1)$
38. $0x21 // 0o21$
39. $0x30 // 0o30$
40. $(0x13 - 0o13) / 0b100 + 3$

[תשובות](#)

[חזור לתוכן](#)

דוגמת תצורת דלק

נניח שפלוני מבצע רישום של תצורת הדלק במכוניתו. לאחר שמלא במכונית 49 ליטרים, המכונית נסעה 669 ק"מ. תוך שימוש במחשבון של פיתון הוא חישב את מספר הקילומטרים לליטר:

```
>>> 669 / 49.
```

```
13.653061224489797
```

עכשיו אם ברצונו לגלות כמה עולה לו כל ק"מ (הכוונה כאן להוצאות ישירות בלבד), עליו לחלק את מחירו של ליטר דלק בתוצאה שהתקבלה, פירוש הדבר להעתיק את התוצאה שהתקבלה. פלוני הינו דייקן מאוד ולכן עליו להעתיק את כל 15 הספרות שמימין לנקודה העשרונית. כדי לחסוך את ההעתקה, סביבת הפיתוח של פיתון שומרת את תוצאת הפעולה האחרונה והיא נגישה ע"י הסימן "_" (קו תחתי). לכן אם נניח שמחיר ליטר דלק הינו 4.80 ₪, אזי הפעולה היא:

```
>>> 4.8 / _
```

```
0.35156950672645737
```

וממנה עולה שמחיר הדלק עבור כל קילומטר הינו 35 אגורות. אם פלוני רוצה לחשב כמה תעלה הנסיעה מעפולה לאילת (מרחק של 470 ק"מ), עליו לכפול את התוצאה האחרונה ב – 470:

```
>>> _ * 470
```

```
165.23766816143495
```

ואז יסתבר שהדלק לנסיעה יעלה 165 ₪.

תרגילי חזרה

התרגילים קשורים (תוצאת תרגיל אחד מוזנת לתרגיל העוקב). נסה להעריך את התשובה (כולל סוג הנתונים שלה) לפני שתנסה את התרגיל במחשב:

1. $0b10011 + 0x3f$
2. $_ * 2$
3. $_ / 16.0$
4. $_ - 8.75$
5. $-3.0 * _$
6. $9 + _$
7. $_ * 5 / 2$
8. $0x77 * 2$
9. $140 - _$
10. $_ + 100$
11. $7 ** _$
12. $_ / 7$
13. $_ - 0b1001$
14. $0xf * _$
15. $_ / (_ + _)$
16. $_ + 5.0$
17. $_ / (_ + _)$
18. $_ ** (_ * 4)$
19. $_ ** (-4 * _)$
20. $_ ** (_ / 2)$
21. $0b1001 + 0o11$
22. $_ * 0b11$
23. $_ // 0x12$
24. $_ ** 0b11$
25. $_ / 0xc$
26. $_ * 0o4$
27. $(_ + 1) * 0x2$
28. $(_ / 0b10) ** 0o2$
29. $(2 * _) // 0o67$
30. $_ ** _$
31. $_ / 3 + _ / 9$
32. $_ ** (_ / 0b110)$
33. $_ // 0xf$
34. $_ / 3 + _ // 3$
35. $_ ** (_ // 0b11)$
36. $(_ / 0o14) ** (_ / 0x12)$
37. $(_ // 0o4) + _ // 0o5$
38. $_ ** (_ - 0x1) - _ ** (_ - 0b1)$
39. $0xff ** _$
40. $0x10 ** _ + 0o10 ** _ + 0b10 ** _$
41. $_ - (_ // 10 + _ // 0b10 + _ // 0o10)$
42. $_ * _ - (_ / 0b10) * (_ / 0x2)$
43. $(_ / 0b11) / (0b10 ** 0b10)$
44. $>>> 0x2 ** _ - 0b10 ** (_ / 0b10)$

תכנות בסיסי

45. >>> _ + _ * (_ // 0o6)

[תשובות](#)

[חזור לתוכן](#)

שמות ערכים

הקו התחתון בפיתון מאפשר שמירה של הערך האחרון שחושב. אולם לפעמים עלינו לשמור יותר מערך אחד (על מנת שלא נצטרך להקלידו כל פעם מחדש). למשל אם מחיר ליטר דלק משתנה, נצטרך להעתיק את מספר הקילומטרים שהמכונית גומעת בליטר, כדי להשתמש בו בחישוב החדש. פתרון טוב היה יכול להיות קו תחתי לכל ערך שחישבנו....

סביבת הפיתוח פיתון אכן מאפשרת מנגנון כזה, ע"י נתינת שם לכל ערך שחושב (literal או סימין בעברית). זו מעין הרחבה של המחשבון שתואר קודם לכן. קיימת רשימה של מילים שמורות (ראה באתר python.org), אשר לא ניתן להשתמש בהן כשמות (למשל: and, del, is, for וכד'). נניח שנקרא לערך שחושב kpl (Kilometer per liter). נתינת שם מבוצעת ע"י הגדרת שם, סימן "=" (שווה) והערך אותו השם מייצג (ערך זה יכול גם להיות תוצאת חישוב כלשהו):

```
>>> kpl=669 / 49.
```

עכשיו אם נרצה להשתמש בערך שחושב, נוכל פשוט להשתמש בשם kpl. אם נרצה לראות את ערכו, פשוט נקליד kpl.

```
>>> kpl
13.653061224489797
```

השם kpl ממיצג את הערך שחושב והוא קיים כל זמן שאנחנו נמצאים בסביבת העבודה. חשוב לציין שבפיתון אותיות קטנות וגדולות הן בעלות משמעות שונה. לכן, KPL למשל אינו kpl, אלא שם שונה (אשר במקרה שלנו לא מוגדר). באופן דומה, אנחנו יכולים להגדיר שם עבור המרחק בין עפולה לאילת, למשל distance ואז נצטרך לשייך לשם גם ערך ע"י:

```
>>> distance = 470
```

באופן דומה גם נשייך את מחיר ליטר דלק לשם price, ע"י:

```
>>> price = 4.8
```

את כל החישובים שעשינו קודם, אפשר לעשות עכשיו, אלא שבמקום להשתמש בערכים, נוכל להשתמש בשמות שלהם. (כאמור שמות שרירותיים שאנחנו קבענו). מחיר הדלק לנסיעה של קילומטר אחד יחושב ע"י:

```
>>> price / kpl
0.35156950672645737
```

ובאופן דומה, מחיר הנסיעה לאילת תחושב ע"י:

```
>>> distance * price / kpl
165.23766816143498
```

תרגילי חזרה

התרגילים קשורים (תוצאת תרגיל אחד מוזנת לתרגיל העוקב). נסה להעריך את התשובה לפני שתנסה את התרגיל במחשב.

עבור עשרה התרגילים הראשונים הנח ש: $a = 1.5$, $b = 0b1010$, $c = 0.5$, $d = 5$, $e = 0x12$

1. $b * e$
2. $b * c$
3. $d + e$
4. $d * a$
5. $a + e$
6. $d * b$
7. $a * e$
8. $a * c$
9. $(a + c) * d / b$
10. $(b + e) / (a + c + d)$

עבור עשרה התרגילים הבאים הנח ש: $a = 3$, $b = 0x10$, $c = 0o21$, $d = 2.5$, $e = 0b1111$

11. $a * b$
12. e / a
13. $e * c$
14. $(a + b) // c$
15. $(a + c + c) / d$
16. $d * b / e$
17. $a ** (e // d)$
18. $(c - e) ** a$
19. $a ** (c - b)$
20. $b + e - c$

חלק ב'

חזור על תרגילים 11-20 והפעם הנח ש: $a = 0b111$, $b = 7$, $c = 5.0$, $d = 0x5$, $e = 0o21$

[תשובות](#)

[חזור לתוכן](#)

משתנים

ניתן להקצות שם לכל דבר, ולא רק לערכים. יתרה מזאת, ניתן לשנות גם הערכים המיוצגים ע"י השמות ולכן שמות אלה נקראים משתנים. הם מייצגים "משהו" שיכול להיות מספר, מחרוזת טקסט, רשימה ועוד. משתנים הם עקרון בסיסי וחשוב בפיתוח תוכנה. בדוגמה הקודמת הגדרנו שמות לערכים, אך למעשה אלה היו משתנים שכן במקרה אחר, הערכים המיוצגים ע"י משתנים אלה יכולים להשתנות. למשל אם מחיר הדלק משתנה, או מספר הקילומטרים שהמכונית נוסעת לליטר דלק וכד'. אין מגבלה על מספר המשתנים שברצוננו להגדיר, או מספר הפעמים שאנחנו משנים את ערכו של כל משתנה. בכל פעם שמשנים את ערכו של משתנה, הערך הקודם נעלם ורק הערך החדש נשמר.

אם המשתנה הוגדר ואנחנו מקלידים אותו, סביבת הפיתוח תדפיס את ערכו. לעומת זאת, הקלדה של שם שלא הוגדר קודם לכן, תגרור הודעת שגיאה שכן המשתנה לא מוגדר.

```
>>> a32 = 123 * 2 / 4 + 23
>>> a32
84
>>> a33
```

Traceback (most recent call last):

File "<pyshell#119>", line 1, in <module>

a33

NameError: name 'a33' is not defined

בדוגמה לעיל, הגדרנו משתנה חדש a32 והכנסנו לתוכו ערך שהינו תוצאת החישוב. הקלדת שם המשתנה, גורמת לפיתוח להדפיס את ערכו. לעומת זאת, כאשר ניסינו לקבל את ערכו של המשתנה a33, התקבלה הודעת שגיאה (a33 לא מוגדר).

משתנים מאפשרים רמת הפשטה בסיסית ע"י שימוש בשם כללי במקום ערכים אמיתיים. כדי להקל על ההפשטה, רצוי מאוד לבחור שמות בעלי משמעות.

תרגילי חזרה

חשב את ערכם של המשתנים: e, f, g, h, i, j, k, l, m, n, o, p. לפני שתנסה את התרגיל במחשב.
הנח ש: a = 1, b = 2, c = 3, d = 4

1. $e = a + d$
2. $f = b * c$
3. $g = d / b$
4. $h = a / c$
5. $i = b / d$
6. $j = (a + c) / (b + d)$
7. $k = (a - c) / (b - d)$
8. $l = b ** (c + a) / d$
9. $m = (a + d) * c / b$
10. $n = (a + b + c) / d$
11. $o = (a + d) ** c / b$
12. $p = (a + d) ** (b / d)$

[תשובות](#)

חזור על התרגילים, אלא שעכשיו הנח ש: a = 1.0, b = 2, c = 3.0, d = 4
חזור על התרגילים פעם נוספת, אלא שעכשיו הנח ש: a = 1, b = 2.0, c = 3, d = 4.0

[חזור לתוכן](#)

מחרוזות

בנוסף לסוגי הנתונים המספריים, שפת פיתון גם תומכת בסוגים נוספים המיועדים למלל. מחרוזות (רצף של תווים) מוגדרת בפיתון ע"י הכללתה בין גרשיים (גרשיים כפולים, משולשים או בודדים). פרט לעובדה שלא מדובר על ערכים מספריים, כל שנאמר לעיל מתקיים גם עבור מחרוזות. מטבע הדברים, פעולות חשבונאיות לא מתבצעות על מחרוזות, אם כי חלק מהפעולות מתבצעות בצורה שונה (כפי שיורחב בהמשך). אולם המשתנים שהוגדרו לעיל, יכולים לשמש כמשתנים מספריים וגם כמשתנים המכילים מחרוזות:

```
>>> "chips and fish"
'chips and fish'
>>> 'chips and fish'
'chips and fish'
>>> t = "chips and fish"
>>> t
'chips and fish'
```

אחת הסיבות לשימוש במספר סוגי הגרשיים היא כדי לאפשר הכללת הגרשיים כחלק מהמחרוזות. למשל:

```
>> "He doesn't know"
"He doesn't know"
>>> 'He said: "yes!"'
'He said: "yes!"'
```

אולם פיתון מאפשרת גם הכללת הגרש או הגרשיים בחלק מהמחרוזות ע"י הכללת הלוכסן ההפוך לפני הגרשיים:

```
>>> st = 'He doesn't know'
>>> st
"He doesn't know"
>>> st1 = "He said: \"No!\""
>>> st1
'He said: "No!"'
```

הלוכסן ההפוך מאפשר המשך המחרוזות בשורה הבאה (כאשר המחרוזת ארוכה ואינה נכנסת לשורה אחת). למשל:

```
>>> st2 = "This is an example \
of a long string"
>>> st2
'This is an example of a long string'
```

אם ברצוננו להדפיס את תוכנו של משתנה, אזי בסביבה האינטראקטיבית אפשר פשוט לכתוב את שמו של המשתנה ואז פיתון מבינה שהכוונה להדפסתו. בתוכנית (והדבר אפשרי גם בסביבה האינטראקטיבית) יש להשתמש בפקודת הדפסה – print. הפקודה תוסבר בהמשך, אולם לצורך התרגילים בסעיף זה, יש לזכור שזו אינה פקודה רגילה. היא מומשה בצורה של פונקציה (קטע קוד שניתן לקרוא לו) ולכן היא מקבלת פרמטרים. במילים אחרות, את כל מה שאנחנו מבקשים להדפיס, יש להכניס לסוגריים.

למשל:

```
>>> st = 'He does\'t know'
>>> print (st)
"He does't know"
>>> st1 = "He said: \"No!"
>>> print (st1)
'He said: "No!"
```


תרגילי חזרה

הגדר את הפקודות הבונות את המחרוזות: (בדוק את תשובתך במחשב)

1. Hello world
2. Hi people. "How are you?"
3. 1. "abc","cde" and 2. "fgh","ijk"
4. "one" and "two"
5. "a","b","c","d"
6. " a" "
7. ' ~"~"~"~" '
8. "(*)(*)(*)" "
9. "^^">>>"<<<"
10. "0123456789"

[תשובות](#)

[חזור לתוכן](#)

עוד על סוגי נתונים

כאמור, בנוסף לסוגי הנתונים המספריים, גם מחרוזות הן סוג נתונים (וקיימים סוגים נוספים שיפורטו בהמשך). פיתון מספקת אפשרות להמרה של סוגי נתונים וכן אפשרות לבדוק את סוג המשתנה. את בדיקת המשתנה מבצעים ע"י פקודת type אשר גם היא מומשה כפונקציה (פירוש הדבר את הפרמטר לפקודה יש לספק בתוך סוגריים). הפקודה מקבלת פרמטר אחד ומחזירה את סוגו.

```
>>> a = 1
>>> b = 2.0
>>> c = 0b1
>>> d = 0x1
>>> e = "Hello"
>>> type ( a )
<class 'int'>
>>> type ( b )
<class 'float'>
>>> type ( c )
<class 'int'>
>>> type ( d )
<type 'int'>
>>> type ( e )
<class 'str'>
```

ניתן לראות שלגבי חלק מסוגי הנתונים, פיתון שומרת את הסוג (למשל עבור המשתנים a, b, e). משתנים אחרים (b, c) הוגדרו כמשתנים בבסיסים אחרים, אלא שלאחר שהמספר הומר לבסיס עשרוני, המשתנה הפך עשרוני גם כן.

כשם שפיתון מאפשרת לגלות את סוג המשתנה, היא כוללת גם פונקציות ספריה אשר מאפשרות להמיר משתנה מסוג אחד לסוג אחר:

```
>>> a , b = 1 , 2.0
>>> type ( a ) ; type ( b )
<class 'int'>
<class 'float'>
>>> d = float ( a )
>>> e = int ( b )
>>> type ( d ) ; type ( e )
<class 'float'>
<class 'int'>
>>> a ; b ; d ; e
1
2.0
1.0
2
```

תכנות בסיסי

בדוגמה מופיעות יכולות נוספות של פיתון. בפקודה הראשונה מודגמת השמה מרובה. ניתן להגדיר מספר משתנים, סימן שווה ואחריו מספר ביטויים, אשר מגדירים את הערכים של המשתנים השונים בהתאמה. מספר הביטויים חייב להיות זהה למספר הביטויים שהוגדרו.

אפשרות נוספת המופיעה בדוגמה לעיל היא היכולת להגדיר מספר פקודות בשורה אחת (הפקודות מופרדות ביניהן ע"י ";"). חשוב לציין שאלה פקודות שונות, אשר רק רשומות באותה שורה.

פונקציות ההמרה המתוארות הן:

- Int – אשר ממירה את המשתנה לסוג שלם (Integer)
- Float – אשר ממירה את המשתנה לשבר (Floating point)

אם ברצוננו להמיר מספרים לערכם הבינארי, אפשר להשתמש בפונקציה bin(). למשל:

```
>>> a=0b111111
>>> a
63
>>> bin(a)
'0b111111'
>>> b=bin(a)
>>> b
'0b111111'
>>> b
'0b111111'
>>> print (b)
0b111111
>>>
```

ניתן להתרשם שהערכים לא השתנו בעקבות ההמרה, אלא רק סוג הנתונים.

חשוב לציין שהפונקציות שתוארו מסוגלות גם להמיר מחרוזות למספרים (בהנחה, כמובן שהערכים השמורים במחרוזת מייצגים מספרים). בנוסף, גם קיימת הפונקציה ההפוכה הממירה ערכים מספריים למחרוזות: str.

```
>>> a , b = 2 , 3.0
>>> s1, s2 = str ( a ) , str ( b )
>>> s1 ; s2
'2'
'3.0'
```

וההמרה ההפוכה:

```
>>> d , e = int ( s1 ) , float ( s2 )
>>> d ; e
2
3.0
```

תרגילי חזרה

הנח שהפקודה הראשונה היא: `a, b, c, d, e = 2.0, 2, 0b11, 3e2, "4"`

כתוב את התשובה שתתקבל מפיתון לאחר הקלדת כל אחת מהפקודות:

1. `a ; b ; c ; d ; e`
2. `type (a + b)`
3. `type (b + c)`
4. `int (b + a)`
5. `int (a + a)`
6. `float (b / c)`
7. `float (int (a / b) / float (a / d))`
8. `a + float (e)`
9. `float (a + int (e) * d)`
10. `int (int (d) / int (e) * a)`
11. `int (d) / int (e) * a`
12. `str (int (e) / (int (a) / b))`
13. `str (a + b + c + d)`
14. `str (float (c / b))`
15. `str (int (d) / int (-a) * b * c * int (e))`
16. `str (int (d) / (int (a) * b * c * int (e)))`
17. `str (int (str (int (e))))`
18. `str (int (e) - int (a) - b)`
19. `str (int (e) - a - b)`
20. `str ((c + c) ** a)`
21. `bin (int (e))`
22. `bin (int (a * b / c))`
23. `int (a * b / c)`
24. `bin (int (d / a ** (b + c)))`
25. `str (bin (int (d)))`
26. `str(a+b-c)`
27. `str (int (a + b + c))`
28. `str (int (a + b - c))`
29. `float (int (e) * a)`
30. `str (d - (a + b) ** (b + c))`

[תשובות](#)

[חזור לתוכן](#)

פונקציות מתמטיות נוספות

בנוסף לפעולות החשבונאיות הרגילות פיתון מאפשרת פעולות נוספות:
 $\text{abs}(x)$ – אשר מחזירה ערך מוחלט של המשתנה
 $x \% y$ – מודולו או שארית חילוקו במספר אחר (שהוא בסיס המודול או מודולו). למשל $5 \% 2 = 1$
 משמעותו לחלק את המספר 5 ב-2 ולשמור את השארית.

דוגמאות:

```
>>> a, b, c, d, e = -1, 3, 7, 5.0, 6.5
>>> a
-1
>>> abs(a)
1
>>> a * e
-6.5
>>> abs(a * e)
6.5
>>> b % c
3
>>> c % b
1
>>> e % d
1.5
>>> d % e
5.0
```

בנוסף לפעולות החשבונאיות ה"רגילות", פיתון כמו שפות פיתוח רבות כוללת גם ספרייה מתמטית המרחיבה את יכולות השפה. הספרייה מכילה פונקציות מתמטיות רבות, אשר זמינות למפתח התוכנה. כדי שאפשר יהיה להשתמש בפונקציות אלה, יש לכלול בתחילת התוכנית פקודה:

```
Import math
```

המשמעות היא שאנחנו מבקשים ל"ייבא" לתוכנית שלנו את היכולות המוגדרות בספרייה המתמטית. כדי לפנות לפונקציה מסוימת המוגדרת בספרייה עלינו להוסיף את הקידומת `math`. למשל אם ברצוננו לקרוא לפונקציה המחשבת שורש ריבועי של המשתנה `a` נכתוב: `math.sqrt(a)`

הספרייה מכילה פונקציות רבות:

1. `ceil` – מחזירה את המספר השלם הקטן ביותר שהינו גדול או שווה לפרמטר שמועבר לה.
 למשל:

```
>>> math.ceil(3.7)
4
>>> math.ceil(4)
4
>>> math.ceil(-12.45)
-12
```

2. `copysign` – מקבלת שני מספרים כפרמטרים ומחזירה תשובה אחת שהיא הפרמטר הראשון (בפורמט של שבר), כאשר סימנו הוא סימן המספר שהועבר כפרמטר השני:

```
>>> math.copysign(3, -4)
-3.0
>>> math.copysign(-3.75, 1.2)
3.75
```

תכנות בסיסי

```
>>> math.copysign ( 5, 3 )
5.0
>>> math.copysign ( -3.1, -4.2 )
-3.1
```

3. factorial – מקבלת פרמטר אחד שהוא מספר שלם וחיובי ומחזירה את העצרת שלו:

```
>>> math.factorial ( 5 )
120
>>> math.factorial ( 5.5 ) <<<<<<<<<< דוגמת שבר
Traceback (most recent call last):
  File "<pyshell#13>", line 1, in <module>
    math.factorial(5.5)
ValueError: factorial() only accepts integral values
>>> math.factorial ( -3 ) <<<<<<<<<< דוגמת מספר שלילי
Traceback (most recent call last):
  File "<pyshell#14>", line 1, in <module>
    math.factorial(-3)
ValueError: factorial() not defined for negative values
```

4. floor – מחזירה את המספר השלם הגדול ביותר שהוא עדיין קטן או שווה לפרמטר שהועבר לפונקציה:

```
>>> math.floor(3.4)
3
>>> math.floor(-4.2)
-5
>>> math.floor(-4)
-4
>>> math.floor(176)
176
```

5. exp – מקבלת פרמטר אחד ומחזירה את e בחזקת הפרמטר שהועבר לה:

```
>>> math.exp ( 2 )
7.38905609893065
>>> math.exp ( 2.4 )
11.023176380641601
>>> math.exp ( -4.5 )
0.011108996538242306
```

6. log10 – מקבלת פרמטר אחד (גדול מאפס וחיובי) ומחזירה את הלוגריתם בבסיס 10 של הפרמטר שהועבר לה:

```
>>> math.log10 ( 0.1 )
-1.0
>>> math.log10 ( 123 )
2.089905111439398
>>> math.log10 ( 1000 )
3.0
```

תכנות בסיסי

```
>>> math.log10 ( 0 )
Traceback (most recent call last):
  File "<pyshell#29>", line 1, in <module>
    math.log10(0)
ValueError: math domain error
```

7. pow – מקבלת שני פרמטרים ומחזירה את הפרמטר הראשון בחזקת הפרמטר השני:

```
>>> math.pow ( 2, 4 )
16.0
>>> math.pow ( -2, 5 )
-32.0
>>> math.pow ( 0.3, 4 )
0.0081
>>> math.pow ( 12, 1.24 )
21.786363482330792
>>> math.pow ( 2, 0 )
1.0
>>> math.pow ( 2, -1 )
0.5
```

8. sqrt – מחזירה את השורש הריבועי של הפרמטר שהועבר לה (חייב להיות אפס או חיובי):

```
>>> math.sqrt ( 144 )
12.0
>>> math.sqrt ( 12.66 )
3.558089374931439
>>> math.sqrt ( 0 )
0.0
>>> math.sqrt ( -1 )
Traceback (most recent call last):
  File "<pyshell#42>", line 1, in <module>
    math.sqrt(-1)
ValueError: math domain error
```

9. פונקציות טריגונומטריות (sin, cos, tan) המקבלות פרמטר ברדיאנים וכמו כן שתי פונקציות שנועדו להמרה בין מעלות לרדיאנים (radians – degrees):

```
>>> math.degrees ( 0.5 )
28.64788975654116
>>> math.radians ( _ )
0.5
>>> math.sin ( _ )
0.479425538604203
>>> math.cos ( _ )
0.8872600507176526
>>> math.tan ( _ )
1.2277064808170115
```

10. קבועים – e ו- π :

```
>>> math.e
2.718281828459045
>>> math.pi
3.141592653589793
```


תרגילי חזרה

הנח שהפקודה הראשונה היא: $a, b, c, d, e = -2.5, 3, 7, 9, 5$

כתוב את התשובה שתתקבל מפיתון לאחר הקלדת כל אחת מהפקודות:

1. $\text{abs}(a * b)$
2. $\text{abs}(a * a)$
3. $\text{abs}(a / b)$
4. $a \% b$
5. $c \% b$
6. $c \% (b + d)$
7. $(b + d) \% c$
8. $(a + b) \% (c + d)$
9. $(c / e) \% a$
10. $(c + e) \% b$
11. $e \% (b + c)$
12. $e \% e$
13. $(e + 1) \% (e + 3)$
14. $(-1 / a) \% d$
15. $(-e / a) \% d$
16. $(2 * d) \% (a * -e)$
17. $b \% c$
18. $b \% c \% d$
19. $d \% b \% c$
20. $d \% c \% b$

חלק ב'

חזור על תרגילים 1 – 20, אלא שהפעם הנח: $a, b, c, d, e = -3.5, 0b101, 0x9, 0o13, 7$

[תשובות](#)

[חזור לתוכן](#)

טיפול במחרוזות

כאמור, מחרוזת היא רצף תווים (טקסט) והיא מוגדרת בין גרשיים (בדדים, כפולים או משולשים). פיתון מספקת פונקציות מיוחדות המאפשרות לטפל במחרוזות, לשלוף מהן נתונים וגם לאחד (לחבר) מחרוזות.

בניגוד לערכים מספריים, השמורים במחשב בצורה בינארית, המחרוזת שמורה כמלל ולכן ע"י שימוש במפתח (אינדקס) הנמצא בסוגריים מרובעים וצמוד למחרוזת, ניתן להתייחס לכל אחד מהתווים שבמחרוזת. מיקום התווים מתחיל בצד שמאל עם המספר 0 וממשיך ימינה עד $n-1$, כאשר n מייצג את אורך המצגת. למשל, נגדיר מחרוזת a המכילה את המלל Shalom. במחרוזת מיוצגת בזיכרון המחשב כרצף תווים כמתואר בטבלה הרצ"ב:

S	H	a	l	o	m	תו
0	1	2	3	4	5	מיקום

בדוגמה, נשלוף מהמחרוזת a את התו הראשון (שכאמור מספרו אפס) ואת התו השני שמספרו 1:

```
>>> a = "Shalom"
>>> a
'Shalom'
>>> a[0]
'S'
>>> a[1]
'h'
```

כדי להקל על העבודה עם מחרוזות בגודל משתנה, פיתון מאפשרת גישה לתווים שנמצאים בצד ימין. הדבר מתבצע ע"י שימוש במפתח שלילי (-1 הוא התו הימני ביותר, -2 התו שלשמאלו וכן הלאה), כפי שניתן לראות בדוגמה:

```
>>> a[-1]
'm'
>>> a[-2]
'o'
```

לאור דוגמה זו, הטבלה שמתארת את מיקום התווים משתנה במקצת.

S	H	a	l	o	m	תו
0	1	2	3	4	5	מיקום 1
-6	-5	-4	-3	-2	-1	מיקום 2

כפי שניתן גם לראות בדוגמה:

```
>>> a[0]; a[-6]
'S'
'S'
>>> a[5]; a[-1]
'm'
'm'
>>> a[3]; a[-3]
'l'
'l'
```

המפתוח שתואר לעיל אפשר לשלוף תו אחד בודד. לעיתים יש צורך לשלוף מספר תווים (תת-מחרוזת). פיתון מאפשרת את השליפה ע"י הרחבת מנגנון המפתוח ע"י שימוש בשני פרמטרים – מיקום התו הראשון לשליפה ומיקום התו האחרון (פלוס אחד) המופרדים ע"י נקודתיים. המיקומים חייבים להיות משפרים שלמים. דוגמה:

```
>>> a = "Programming is fun!!"
>>> a [ 0 : 11 ]
'Programming'
>>> a [ 0 : 7 ]
'Program'
>>> a [ 3 : 7 ]
'gram'
>>> a [ -5 : -1 ]
'fun!'
```

כמו במקרים אחרים שראינו, המפתחות (כתובת ההתחלה והסיום) יכולים להיות ביטויים הכוללים נוסחאות מתמטיות שונות (ובלבד שתוצאת הנוסחה תהיה מספר שלם. אין גם חובה לציין את שני הביטויים. אם הם חסרים, פיתון תשלים את כתובות הקצה. כלומר אם הביטוי הראשון חסר, הוא יוחלף באפס ואילו אם הביטוי השני חסר, משמעות הדבר עד התו האחרון במחרוזת. כפי שניתן לראות בדוגמה:

```
>> a [ : ]
'Programming is fun!!'
>>> a [ 15 : ]
'fun!!'
>>> a [ : 11 ]
'Programming'
```

בפיתון קיימות עוד מספר פונקציות שנועדו לעזור בטיפול במחרוזות:
 + (חיבור או concatenation) שמשמעותה להצמיד שתיים (או יותר) מחרוזות כדי ליצור מחרוזת גדולה.
 * שכפול המחרוזת מספר פעמים
 len (x) – מחזירה את אורכה של המחרוזת.

דוגמאות:

```
>>> a = "Programming is fun!!!"
>>> b = " with Python"
>>> c = "more "
>>> d = a [ : 11 ] + b + a [ 11 : 15 ] + c + a [ 15 : ]
>>> d
'Programming with Python is more fun!!!'
>>> len ( d )
38
>>> len ( a ) ; len ( b ) ; len ( c )
21
12
5
>>> a = "abc"
```

```
>>> b = 2 * a
>>> b
'abcabc'
>>> b = b + " "
>>> b * 2
'abcabc abcabc '
```

תרגילי חזרה

רשום את התשובה שתתקבל מפיתון לאחר הקלדת כל אחת מהפקודות:

1. word = "Information Systems"
2. word [5 : 12]
3. word [: 5]
4. word [-7 : -1]
5. word [:]
6. word
7. word [-7 : -4] + " "
8. (word [-7 : -4] + " ") * 3
9. "<" + word [: 3] + ">"
10. word [-2 :]
11. word [-0]
12. word [15 : 100]
13. word [: 4] + word [4 :]
14. word + " class"
15. word [: 11] + "+" + word [12 : 19] + "=" + word [: 19]
16. word [int(len (word) / 2) :]
17. len ('Supercalifragilisticexpialidocious')
18. word [int(len (word) / 3) : int(len (word) / 2) + 2]
19. word [: int(len (word) / 4)] * (int(len (word) / 4))
20. len (word [1 : int(len (word) / 4)])
21. word [int (float (23 / 7)) : int (float (17 / 3))]
22. word [: -8] + " for " + word [-7 :]
23. word [-7 : -4] * 3 + " "
24. [word [-7 : -3] + " "] * 3
25. word[-4:-1]+word[12:15]
26. 3 * [word [-7 : -4] + word [: 2]]
27. 2 * [2 * word [-11 : -9] + 3 * word [5 : 7]]
28. 2 * word [5 : 7] + word [-2] + word [-11] + word [6]
29. word [(int(len (word) / 4)) : (int(len (word) / 2))]
30. word [-1 : 5]

[תשובות](#)

[חזור לתוכן](#)

הדפסה

פקודות הדפסה הוזכרו כבר בסעיף קודם. כאן מובאת הרחבה נוספת.

בסביבה האינטראקטיבית, הקלדת שם משתנה גורמת להדפסת ערכו (הצגתו על גבי המסך), אולם כאשר כותבים תוכנית, יש צורך בפקודה מפורשת להדפסה. פקודה זו היא פקודת ה- `print`.

בצורתה הבסיסית הפקודה מדפיסה את הפרמטרים שקיבלה (אפס או יותר). אם לא מספקים פרמטרים היא תדפיס שורה ריקה (למעשה תקדם את הסמן לשורה הבאה). אם הפרמטרים הם משתנים, פיתון תדפיס אותם בהתאם לסוג. לשברים תתווסף הנקודה העשרונית. אולם אם הביטוי בפקודה הוא לא משתנה, אלא ערך, פיתון תדפיס אותו, כולל ביצוע הפעולות הנדרשות. למשל במספרים, בעלי פעולה אריתמטית, פיתון תדאג לבצעה ותדפיס את התשובה.

נקודה חשובה שיש לזכור היא העובדה שפקודת ההדפסה בפיתון מומשה כפונקציה ולכן את הפרמטרים שלה יש להכניס לסוגריים. יתרה מזאת, גם אם רוצים "להדפיס" שורה ריקה יש להשתמש בפקודת ההדפסה, כאשר היא מלוות בסוגריים ריקים: `print ()`

דוגמאות לשימוש בפקודת ההדפסה:

```
>>> a, b, c, d = 1, 0x2, 3., "abc"
>>> print (a, b, c, d)
1 2 3.0 abc
>>> print (a + b)
3
>>> print (a + c)
4.0
>>> print (2 * 3 / 5)
1.2
>>> print ("Hello")
Hello
>>> print ("a"); print (); print ("b")
a

b
```

קדימויות הפעילויות המתמטיות הן כפי שהוסבר בפרק הדין במחשבון.

תרגילי חזרה

רשום את התשובה שתתקבל מפיתון לאחר הקלדת כל אחת מהפקודות:

1. `print ("12 * 8 = ", 12 * 8)`
2. `print ("This is an example")`
3. `print ("3 * fun = ", 3 * "fun ")`
4. `print ("Hi" , "everybody" , " , " , "how are you?")`
5. `a="Computer"; print (a , a [: 3] , a [3 : 5] , a [5 :])`
6. `a= "This is a test"; print (' ' ' , a , ' ' ' , "is" , len (a) , "character long")`
7. `MyVar = 100 ; MyString = "MyVar=" ; print (MyString , MyVar)`
8. `x = "Hello" ; y = "World" ; z = x + y ; print (z)`
9. `print ("One Megabytes is 2^20 bytes, or : " , 2 ** 20 , "bytes")`
10. `print ("V" , " , " , "V" , " , " , "V")`
11. `print ("~~" , " , " , "~~" , " , " , "~~")`
12. `print ("a,b,c" , " , " , "d,e" , " , " , "f")`
13. `print (",,, " * 3 , " , " , " , " , " , " * 3)`
14. `print (" , " + " , " + " , ")`
15. `print ("#,#" , " , " , "#,#") ##`
16. `print ("3*5" , " , " , "3" * 5)`
17. `print ((5 * "abc") [: 5])`
18. `print (5 * "abc" [: 5])`
19. `print (" print " * 3)`
20. `print (3 * "3* " + 2 * "2*")`

[תשובות](#)

[חזור לתוכן](#)

עוד על הדפסה

בהמשך לפקודות המגדירות מחרוזות ולפקודת ההדפסה, עולה לעיתים הצורך לשלב את תו הסיום (גרש או גרשיים) בתוך המחרוזת. לצורך כך הוגדר תו מיוחד Escape sequence שמאפשר יכולות נוספות:

```
>>> a = "abcd\"efg"
>>> print (a)
abcd"efg
```

לאחר שהוגדר התו, הוא מאפשר יכולות נוספות, כמו הוספת מעבר שורה באמצע המחרוזת (`\n`), הוספת TAB (`\t`) והגדרת יכולת להוסיף כל תו (אפילו כאלה שלא ניתנים להדפסה). שימוש נוסף הוא הוספה של התו `\` כחלק מהמחרוזת.

דוגמאות:

```
>>> print ("abc\ndef\nghi")
abc
def
ghi
>>> print ("ab\tcd\tef")
ab    cd    ef
>>> print ("ab\\cd")
ab\cd
```


[illegible]

חזור לתוכן

לולאות

לפעמים בתוכנית עלינו לבצע רצף פקודות בצורה מחזורית. הדבר נקרא לולאה. פיתון תומך בשני סוגי לולאות ובפרק זה נדון ונתרגל את פקודת ה- `for`.

פורמט הפקודה הוא: `for <var> in <sequence>:`
`<body>`

- `<var>` מייצג משתנה כלשהו, הנקרא גם מונה הלולאה, אשר מתקדם עם ביצוע הלולאה ובכל מחזור ערכו משתנה, עד אשר הלולאה מסתיימת.
- `<sequence>` מייצג רצף כלשהו, אזור מגדיר את מספר מחזורי הביצוע ואת ערכי מונה הלולאה.
- `<body>` מייצג את קצף הפקודות שיש לבצע בכל מחזור של הלולאה.

את פקודות ה- `for` מסיימים בנקודתיים ושורה חדשה, ואחריהם סביבת הפיתוח מבצעת הזחה וזה המקום להקליד את גוף הלולאה (רצף הפקודות לביצוע). במקרה שגוף הלולאה כולל פקודה אחת בלבד, ניתן להקלידה לאחר הנקודתיים באותה שורה.

חשוב לציין שבעת ביצוע לולאת `for`, פיתון יודעת את מספר המחזורים שעליה לבצע.

דוגמה:

```
>>> for k in [ 0, 2, 5, 9, 3, 7 ] :
      print (k)
```

```
0
2
5
9
3
7
```

בדוגמה זו, `k` הוא מונה הלולאה, המספרים בסוגריים (נתונים מסוג רשימה, שתוסבר בהמשך) קובעים את הערכים של מונה הלולאה. גוף הלולאה במקרה זה כולל רק את פקודת ההדפסה של מונה הלולאה. ניתן לראות את ערכו של מונה הלולאה בכל מחזור. ערכים אלה חופפים את הערכים אשר נמצאים ברשימה.

כדי לתמוך בפעילויות שונות ובין היתר גם בלולאות, הוגדרה בפיתון פונקציה עזר `range`. הפונקציה מקבלת פרמטר מספרי אחד (`n`) ומחזירה את רצף המספרים `[0,1,2,...n-1]`. לכן אם נשנה מעט את הלולאה לעיל, ונשתמש בפונקציה `range`, נקבל לולאה אחרת:

```
>>> for l in range ( 7 ) :
      print (l)
```

```
0
1
2
3
4
5
6
```

תכנות בסיסי

למעשה, לפונקציה range עוד שני פרמטרים (הניתנים להשמטה), אך במקרים רבים יכולים להיות שימושיים. range(start,end,step)

- start – מציין את הערך ההתחלתי של רצף המספרים
- end – מציין את הערך העליון של רצף המספרים, אולם יש לשים לב שהרצף לא יכלול את המספר end, אלא אחד לפניו.
- step – מציין את המרווח בין מספרים עוקבים.

ברירת המחדל עבור start היא אפס וברירת המחדל עבור step היא 1.

דוגמאות לפונקציה:

```
>>> range ( 10 )
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range ( 0 , 10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range ( 0 , 10 , 1 )
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range ( 2 , 15 , 4 )
[2, 6, 10, 14]
>>> range ( 5 , 55 , 22 )
[5, 27, 49]
>>> range ( -5 , 5 , 2 )
[-5, -3, -1, 1, 3]
>>> range ( -3 , 3 )
[-3, -2, -1, 0, 1, 2]
```

דוגמה ללולאה מבוססת range:

```
>>> for i in range ( 3 , 31 , 7 ) :
    print (i)

3
10
17
24
```

במקרה זה, הרצף החל במספר 3 ובכל מחזור התקדם בשבע. לכן במחזור השני ערכו של מונה הלולאה היה עשר, במחזור השלישי שבעה עשר, במחזור הרביעי עשרים וארבעה ובגלל שבמחזור הבא הוא היה צריך להיות שלושים ואחד, מחזור זה לא התבצע.

אם במהלך הדפסת ערכים בלולאה, אנו רוצים שלא לרדת לשורה הבאה, אפשר להוסיף פרמטר לפקודות ההדפסה. (" " end=) print (num , end=) הפרמטר end שבמקרה זה ערכו רווח, מציין שבסוף ההדפסה יש להוסיף את הרווח במקום לרדת לשורה הבאה.

תרגילי חזרה

רשום את התשובה שתתקבל מפיתון לאחר הקלדת כל אחת מהפקודות (בדוק את תשובתך). בגלל שהלולאות רשומות בשורה אחת, יש צורך בהקשת Enter נוסף, כדי שפיתון תריץ את הקוד. חלק מהתרגילים, בעיקר האחרונים שבהם כוללים התייחסות לפקודות שיופיעו בפרקים הבאים בחוברת.

1. for i in range (1, 12, 4) : print (i)
2. for i in range (2, 12, 1) : print (i); print (i+1)
3. for k in range (2, -12, -2) : print (k)
4. for l in range (15, 3) : print (l)
5. for j in range (5, 6) : print (j)
6. for g in [0, 2, 3, 4, 8, 3, 6, 4] : print (g * g)
7. for k in [2, 4, 5, 3, 6, 8, 7, 9] : print (float (k * (k-1)))
8. for t in [3, 2, 5, 4, 7, 6, 9, 8] : print (float (t / 2))
9. for t in [3, 2, 5, 4, 7, 6, 9, 8] : print (float (t) / 2)
10. for r in "abcdefg": print (r)
11. for r in " a b c d e f g ": print (r)
12. for r in "ABC": print (r+"a")
13. for e in "12345": print (int (e) + 3)
14. for e in "12345": print (int (e) + eval (e))
15. for e in "12345": print (str (int (e)) + "9")
16. for v in [2, 3, 4, 5, 6, 7] : print (v % 2)
17. for v in [2, 3, 4, 5, 6, 7] : print (v % v)
18. for v in [2, 3, 4, 5, 6, 7] : print (v % v + 1)
19. for v in [2, 3, 4, 5, 6, 7] : print (v % (v + 1))
20. for v in [2, 3, 4, 5, 6, 7] : print ((v + 1) % v)
21. for v in [2, 3, 4, 5, 6, 7] : print (2 * v % (v + 1))
22. for v in [2, 3, 4, 5, 6, 7] : print (3 * v % 2 * v)
23. for f in [2, 3, 4, 5, 6, 7] : print (str (f) * f)
24. for f in [2, 3, 4, 5, 6, 7] : print (str (f) + "\t" + str (f))
25. for f in range (10) : print ("*" * f)
26. for f in range (10) : print ("*" * int ((f / 2)))
27. for f in range (1, 5) : print ("*" * f)
28. for t in "1 , 2 , 3 , 4 " : print (t)
29. for r in [1, 2, 3, 4] : print (int (r))
30. for j in range (10, 0, -1) : print (j)
31. for j in range (10, 0, -1) : print (abs (j))
32. for item in ["This", "is", "an", "example"] : print ("Current item in list is:" , item)

לולאות בעלות מספר שורות.

33.

```
oten = range ( 1, 10 )
for i in oten :
    print ( i , i*i , i*i*i )
```
34.

```
sum = 0.
for i in range ( 7 ) :
    sum = sum + i
    print ( sum / ( i + 1 ) )
```

```
35. sum = 0.
    for i in range ( 7 ) :
        sum = sum + i * i
    print ( sum / ( i + 1 ) )
```

```
36. sum = 0.
    for j in [ 2, 3, 5, 8, 12, 13, 12, 14 ] :
        sum = sum - j
    print (abs ( sum / ( j + 1 ) ) )
```

```
37. sum = 0.
    for j in [ 7, 3, 5 ] :
        sum = sum * j
    print ( sum % j )
```

```
38. for j in range( -32, -21 ) :
    print ( j+30 )
```

שילוב וקיבון לולאות (רשום מה קטע הקוד מבצע והערך את התשובה).

```
39. for j in range ( 10 ) :
    for k in range ( j ) :
        print ( k, end = " " )
    print ( )
```

```
40. for j in range ( 9 ) :
    for k in range ( j ) :
        print ( ""*k, end = " " )
    print ( )
```

```
41. for j in range ( 7 ) :
    for k in range ( j ) :
        print ( "outer loop = ", j , "inner loop = : " , k )
    print ( )
```

```
42. for j in range ( 2, 20 ) :
    for k in range ( 2, j ) :
        if j % k == 0 :
            print ( j, k )
    print ( )
```

```

43. a = [ 1, 2, 3, 3, 4, 2 ]
    sum = 0.
    more, less, equal = 0 , 0 , 0
    for j in a:
        sum = sum + j
    avg = sum / len ( a )
    for k in range ( len ( a ) ) :
        if a [ k ] > avg :
            more = more + 1
        elif a [ k ] == avg :
            equal =equal + 1
        else :
            less = less + 1
    print ( "For:", a )
    print ( "Above average:", more )
    print ( "Average:", equal )
    print ( "Less than average:", less )


44. for multiplier in range ( 2 , 7 ) :
    for j in range ( 1 , 7 ) :
        print ( "%d x %d = %d" % ( j , multiplier , j * multiplier ) )
    print ( "----- " )


45. n=9+1
    for row in range (2, n):
        print ( row, "|", end=" " )
        for col in range ( 2, n ) :
            print ( "%3d" % (row * col), end = " " )
        print ( )


46. width = 8
    height = 7
    n = 1
    for x in range ( width ) :
        print ( )
        for y in range ( height ) :
            if n < 10 :
                print ( "", end = " " )
            print ( n, end = " " )
            n = n + 1
    print ( )

```

```

47. for k in range ( 5 ) :
    for j in range ( 5 ) :
        print ( " " * k , "*" * j )

48. for k in range ( 1 , 5 ) :
    for j in range ( 1 , 4 ) :
        print ( "k = " , k , " | j =" , j )
    print ( "Sof!" )

49. n = 5
    cnt = 0
    for a in range ( n ):
        for b in range ( n ):
            for c in range ( n ) :
                for d in range ( n ) :
                    cnt = cnt + 1
    print ( cnt )

50. for j in range ( 1 , 7 ) :
    for k in range ( 1 , j + 1 ) :
        print ( j , k , j * k )
    print ( "-----" )

51. for s in [ 'ab' , 'c' ] :
    for n in [ 1 , 3 ] :
        print ( s * n , end = " " )
    print ( )

52. outer = ['AB' , 'CD' , 'EF' ]
    inner = ['ab' , 'cd' , 'ef' ]
    for m in outer :
        for n in inner :
            print ( m + n )

53. text = "ABCDE"
    for line in range ( len ( text ) ) :
        print ( text [ line ] * ( line + 1 ) )

54. text = "ABCDE"
    for line in range ( len ( text ) ) :
        print ( text [ line ] * ( line + 1 ) , end = " " )

55. text = "ABCDE"
    for line in range ( len(text) ) :
        for col in range ( line + 1 ) :
            print ( text [ col ] * ( col + 1 ) , end = " " )
    print ( )

```

```
56. for o1 in range (2):
    for o2 in range (2):
        for o3 in range (2):
            for o4 in range (2):
                for o5 in range (2):
                    print (o1,o2,o3,o4,o5)
```

```
57. for ch in "Word":
    print (ch)
```

```
58. list = [ 1, 2, 3, 4 ]
    for index in range ( len ( list ) ) :
        list [ index ] += 1
    print ( mylist )
```

```
59. for x in [ 1, 2, 3, 4]:
    for y in [ 10, 15, 3, 22 ] :
        if x*y > 25:
            print ( x, y )
```

```
60. largest = 0
    for value in [ 4, 31, 21, 16, 44, 19, 40, 36 ] :
        if value > largest:
            largest = value
    print ("largest value :", largest )
```

```
61. largest = 0
    for value in [ 4, 31, 21, 16, 44, 19, 40, 36 ] :
        if value == 16:
            break
        if value > largest:
            largest = value
    print ("largest value :", largest )
```

```
62. largest = 0
    for value in [ 4, 31, 21, 16, 44, 19, 40, 36 ] :
        if value == 44:
            continue
        if value > largest:
            largest = value
    print ("largest value :", largest )
```


תכנות בסיסי

```
63. total = 0
    for index1 in range(0,3):
        for index2 in range(0,4):
            for index3 in range(0,5):
                total += 1
    print (total)
```

[תשובות](#)

[חזור לתוכן](#)

ביטויים בוליאניים

מספרים בוליאניים הם סוג נתונים נוסף (מעבר לאלה שתוארו בפרקים קודמים), אשר מאופיינים בעובדה שהם בעלי שני מצבים בלבד. אמת או שקר, 1 או 0 (True, False). משתנה כזה מוגדר ע"י Bool. לדוגמה:

```
>>> b = True
>>> type ( b )
<type 'bool'>
>>> a = False
>>> type ( a )
<type 'bool'>
```

כדי לבדוק תנאים בפיתון משתמשים בסימנים מוכרים (חלקם) על פי הפירוט:

משמעות	תנאי מתמטי	תנאי בפיתון
קטן	$<$	$<$
קטן שווה	\leq	$<=$
שווה	$=$	$==$
גדול או שווה	\geq	$>=$
גדול	$>$	$>$
שונה	\neq	$!=$

דוגמאות:

```
>>> a , b = 4 , 7
>>> a == 4
True
>>> b == 6
False
>>> a + b == 11
True
>>> a + b != 11
False
>>> a > b
False
>>> b <= a
False
>>> b > a
True
>>> a + b > b + a
False
>>> ( a % b ) == ( b - a )
False
>>> ( b % a ) == ( b - a )
```

תכנות בסיסי

True

>>> a / b == b // a

False

תרגילי חזרה

רשום את התשובה שתתקבל מפיתון לאחר הקלדת כל אחת מהפקודות (בדוק את תשובתך).

1. "True" == "True"
2. 1 == 1
3. 13 == 31
4. 2 > 3
5. 2 * 3 > 3 * 2
6. 2 ** 3 > 3 ** 2
7. 13 == "13"
8. "Hi" == "Hello"
9. 25 >= 5 * 2
10. 0x19 == 25
11. 25.0 == 25
12. "abc" > "abc"
13. "bcd" < "bcd"
14. len (range (-2 , 10)) == len (range (12))
15. 1 + 2 + 3 + 4 == 4 + 2 + 3 + 1
16. 4 ** 2 >= 2 * 4
17. -3 == 3
18. abs (-3) == 3
19. 10 / 3 == 3
20. 17 % 6 >= 3
21. 17 / 7 * 7 == 17
22. 17 / 7 * 7 + 17 % 7 == 17
23. "ab" * 3 == "ababab"
24. "ab" + "ab" == "abab"
25. a = "python" ; a [1] == a [-5]
26. a = "python" ; a [2 : 3] == a [-4 : -3]
27. 0b10101 == 21
28. 0xff == 255
29. 0x105 + 9 >= 269
30. 0x3f / 9 == 7
31. 1.23e2 == 99 + 24
32. 2.4e2 == 0xf0
33. 0xa0 / 0o100 >= 2
34. 2 * 0b1010 == 1024
35. 0b100 ** 2 / 2 ** 2 == 0x4
36. 0b1000 ** 2 <= 4 ** 3
37. 0x8 ** 0x2 >= 2 ** 7
38. (0x13 - 0xb) ** 2 >= 60
39. a = 0 ; (a == a) != (a == a)
40. eval ("0x3f") >= eval ("70")
41. for a in range (1 , 4) :
 for b in range (1 , 4) :
 print "(" , a , "<" , b , ")" == " , a < b
 print ()

[תשובות](#)

[חזר לתוכן](#)

קלט

בפרקים הקודמים התייחסנו לסביבה האינטראקטיבית של פיתון. במקרים רבים סביבה זו אינה מספיקה ואנו נדרשים לפתח תוכנית ממש. הדבר מתבצע ע"י הקלדת פקודות התוכנית לקובץ מלל (טקסט) בעל סיומת של txt, או py. מתוך הסביבה האינטראקטיבית ניתן לטעון את הקובץ, לשנותו אם צריך ולהריצו (ע"י run או שימוש ב-F5). חלק מהתוכניות חייבות לקבל קלט מהמשתמש על מנת שתוכלנה לרוץ.

הפקודה המאפשרת קליטת נתונים (בזמן ריצה) מהמשתמש היא פקודת ה-input (למעשה כמו פקודת ההדפסה גם זו פונקציה, אולם נושא הפונקציות יורחב באחד מפרקי ההמשך).

הפונקציה הפשוטה input – נועדה לקבל קלט טקסטואלי מהמשתמש והיא מכניסה אותו אל משתנה על פי שהוגדר בתוכנית. כדי להקל על הפיתוח, פונקצית ה-input מאפשרת גם להדפיס מחרוזת לפני בקשת הקלט מהמשתמש (כדי להסביר למשתמש מה נדרש ממנו). לדוגמה:

```
>>> a = input ( "Enter an number:" )
Enter an number:123
>>> print (a)
123
```

בדוגמה, השתמשנו בפונקצית הקלט, כדי לקרוא מחרוזת טקסט שהוקשה במקלדת ולהכניסה למשתנה a. בניגוד להשמה מרובה של ערכים, פונקצית קלט עובדת בצורה שונה. חושב להדגיש שהקלט תמיד מתורגם למחרוזת, כך שאם המשתמש הקליד סדרה של ספרות וברצונו לעבד אותן כמספר, שי קודם לכן להמיר אותן למספר (למשל ע"י הפונקציה int).

דוגמה:

```
>>> a = input ( "Enter a number: ")
Enter a number: 123
>>> a
'123'
>>> print ( a * 3 )
123123123
>>> a = int ( a )
>>> a
123
>>> print ( 3 * a )
369
>>>
```

משמעות הדבר היא שגם אם הקלדנו מספרים, הם מוחזרים כמחרוזת. ואכן, כאשר הדפסנו אותם פי שלוש, קיבלנו את הערך: 123123123. כאשר רצינו להשתמש בהם כערכים מספריים המרתו אותם (ע"י int) ואז כאשר הדפסנו את הערך פי 3 קיבלנו 369. אפשר כמובן לבצע את ההמרה ישירות על תוצאת הקלט:

```
>>> a = int ( input ( "Enter a number: " ) )
Enter a number: 999
>>> a
999
>>>
```

תרגילי חזרה

רשום את התשובה שתתקבל מפיתון לאחר הקלדת כל אחת מהפקודות (בדוק את תשובתך).

1. `a = input ("Enter a number (1-10):")`
`print (a, a*2, type (a))`

בהנחה שהמשתמש הקיש 6

2. `a = input ("Enter first number:")`
`b = int (input ("Enter second numbers:"))`
`c = a * b`
`print (a, b, c, type(a), type(b), type(c))`

בהנחה שהמשתמש הריץ את התוכנית מספר פעמים ולהלן הקלט של כל ריצה:

1. 7,3
2. 999,4
3. 55.0,3
4. abc,4
5. 17-11,2

3. `sum = 0`
`a = int (input ("Enter a number:"))`
`for j in range (a) :`
 `sum = sum + j`
`print ("For first" , a , "number(s) the sum is:" , sum)`

בהנחה שהמשתמש הריץ את התוכנית מספר פעמים ולהלן הקלט של כל ריצה:

1. 5
2. 12
3. 8
4. 1
5. 0

4. `prod = 1`
`a = int (input ("Enter a number:"))`
`for j in range (2 ,a) :`
 `prod = prod * j`
`print ("The product is:", prod)`

בהנחה שהמשתמש הריץ את התוכנית מספר פעמים ולהלן הקלט של כל ריצה:

1. 1
2. 2
3. 3
4. 5
5. 7

```
5. prod = 0
a = int ( input ( "Enter a number:" ) )
for j in range ( a ) :
    prod = prod + j % 2
print ( "The reminders sum is:", prod )
```

בהנחה שהמשתמש הריץ את התוכנית מספר פעמים ולהלן הקלט של כל ריצה:

```
1 .1
2 .2
3 .3
5 .4
11 .5
```

```
6. prod = 0
a = int (input ( "Enter a number:" ) )
for j in range ( a ) :
    prod = prod + j % 3
print ( "The reminders sum is:", prod )
```

בהנחה שהמשתמש הריץ את התוכנית מספר פעמים ולהלן הקלט של כל ריצה:

```
1 .1
2 .2
3 .3
5 .4
prod = 011 .7
```

```
a = int (input ( "Enter a number (>3):" ) )
for j in range ( 3 , a , 3 ) :
    prod = prod + j - ( j - 1 ) + ( j - 2 )
print ( "The reminders sum is:", prod )
```

בהנחה שהמשתמש הריץ את התוכנית מספר פעמים ולהלן הקלט של כל ריצה:

```
6 .1
7 .2
9 .3
5 .4
11 .5
```

```
8. text = "abcdefghijklmnopqrstuvwxyz"
a = input ( "Enter first number:" )
b = input ( "Enter second number:" )
c = input ( "Enter third number:" )
print ( text [ a ] + text [ b ] + text [ c ] )
```

בהנחה שהמשתמש הריץ את התוכנית מספר פעמים ולהלן הקלט של כל ריצה:

```
3,0,-2 .1
3,4,11 .2
-5,0,-7 .3
5,14,6 .4
18,8,-7 .5
```

```
9. b = ""
   a = input ("Some text:")
   for j in range ( len ( a ) ) :
       b = b + a [ -j-1 ]
   print ( a, "<>", b )
```

בהנחה שהמשתמש הריץ את התוכנית מספר פעמים ולהלן הקלט של כל ריצה:

```
1. abcd
2. 1234567
3. ab
4. r
5. 9753
```

```
10. b=""
    a = input ( "Some text:" )
    for j in range ( 1 , len ( a ) , 2 ) :
        b = b + a [ -j ]
    print ( a, "<>", b )
```

בהנחה שהמשתמש הריץ את התוכנית מספר פעמים ולהלן הקלט של כל ריצה:

```
1. 12345678
2. 90123456
3. abcd
4. n
5. 01010101
```

```
11. b = ""
    a = input ( "Some text: " )
    for j in range ( int (len ( a ) / 2 ) ) :
        b = b + a [ j ]
        b = b + a [ -j -1 ]
    print ( a, "<V>", b )
```

בהנחה שהמשתמש הריץ את התוכנית מספר פעמים ולהלן הקלט של כל ריצה:

```
1. 12345678
2. 901238456
3. abcd
4. n
5. 01010101
```

[תשובות](#)

[חזור לתוכן](#)

תנאים (פקודות התניה)

מנגנון הלולאות שתואר בפרקים קודמים הוא סוג של פקודת תנאי. בכניסה ללולאה, שפת פיתון יודעת בדיוק כמה מחזורים יש לבצע בלולאה (על פי התנאי שבראש הלולאה). לפעמים משתמשים בפקודות תנאי שלא במסגרת לולאות.

בפיתון הפקודה אשר בודקת תנאים היא פקודת if. הפורמט שלה הוא:

```
if <condition>:
    <body>
```

ה – condition נועד לתאר את התנאי והוא מתרגם לביטוי בוליאני, אשר אם הוא נכון, שפת פיתון תבצע את הפקודות הנמצאות ב – body. במצב רגיל הפקודות הנוספות (המוגדרות בגוף התנאי) מוקלדות בשורה הבאה תוך שימוש בהזחה. אולם כמו במקרים של לולאות, אם גוף התנאי כולל פקודה אחת בלבד, ניתן להוסיפה בהמשך פקודת ה – if.

הפורמט של התנאי (condition) הוא: <expr> <relop> <expr>, כאשר: <expr> מציין ביטוי כלשהו (משתנה, פעולות על משתנים ואפילו קבועים) ו – <relop> מציין יחס מסוים (כמו היחסים הבוליאניים שהוצגו בפרקים קודמים).

למשל אם ברצוננו לפתח תוכנית דוגמת abs, אשר מחזירה את ערכו המוחלט של כל מספר אשר היא מקבלת:

```
a = int(input("Enter a number:"))
b = a
if a < 0 : b = -a
print ( a, ">>>", b )
```

דוגמת הרצת התוכנית:

```
Enter a number:55
55 >>> 55
>>> main()
Enter a number:-22
-22 >>> 22
```

לפעמים יש צורך בתנאי כפול ואכן שפת פיתון תומכת גם במבנה זה ע"י הרחבת ה – if. הפורמט במקרה זה הוא:

```
If <condition>:
    <statements1>
else:
    <statements2>
```

פירוש הדבר שרצף הפקודות המוגדרות ע"י <statements1> מבוצעות כאשר התנאי מתקיים ואם אינו מתקיים, מתבצע רצף הפקודות <statements2>. מטבע הדברים, פורמט פקודה זה מתאים כאשר אנחנו בודקים תנאי והיפוכו.

למשל את דוגמת התוכנית שהוצגה לעיל נרחיב ע"י: אם המספר שלילי, הופכים אותו לחיובי ומוסיפים לו 2, אחרת מפחיתים ממנו 2. התוכנית החדשה תהיה:

```
a = int(input("Enter number:"))
b = a
if a < 0 :
```

תכנות בסיסי

```
b = -a
b = b + 2
else:
    b = b - 2
print ( a, ">>>", b )
```

ודוגמאות הרצה:

```
Enter number:-77
-77 >>> 79
>>> targil ( )
Enter number:0
0 >>> -2
>>> targil ( )
Enter number:33
33 >>> 31
```

לפעמים גם שני התנאים אינם מספיקים ויש צורך בבדיקה של מספר תנאים. אמנם אפשר לפתור את הבעיה ע"י קינון של תנאים, אולם צריך להיזהר שלא לפגוע בקריאות התוכנית והאפשרות לתחזק אותה בעתיד. שפת פיתון תומכת גם בתנאים מרובים ע"י הוספת רכיבים לפקודת ה – if.

הפורמט המורחב הוא:

```
If <condition1>:
    <statements1>
elif <condition2>:
    <statements2>
elif <condition3>:
    <statements3>
....
else:
    <statementsn>
```

המשמעות של elif היא elseif והיא המאפשרת שילוב של תנאים (שאינם חופפים). ניתוח הפקודה בזמן הריצה כולל מעבר על כל אחד מהבלוקים הבונים אותה (בלוק הוא התנאי והפקודות המבוצעות כאשר הוא מתקיים). אם התנאי מתקיים, יתבצעו הפקודות שבבלוק שלו ומייד לאחר מכן ימשיך הביצוע בפקודה העוקבת לאחר הבלוק האחרון של ה – if. אם התנאי אינו נכון, יבדק התנאי המופיע בבלוק הבא. אם הוא נכון, יתבצעו הפקודות בבלוק שלו וכן הלאה. אם אף תנאי אינו נכון, יתבצעו הפקודות הרשומות תחת ה – else. אולם, אין חובה להגדיר את בלוק ה – else. במצב כזה, אם אף אחד מהתנאים אינו מתקיים, לא יבוצע כלום בפקודת ה – if הזאת, במהלך הריצה.

דוגמה לתוכנית המבוססת על דירוג הציונים האירופאי:

0-59 כישלון,
60-69 מתחת לממוצע,
70-79 ממוצע,
80-89 מעל לממוצע,
90-100 מצוין.

```
a = int( input ("Enter grade:"))
if a <= 59 :
    b = "Failed"
elif a <= 69 :
    b = "Below Average"
elif a <= 79 :
    b = "Average"
elif a <= 89 :
    b = "Above Average"
else :
    b = "Excellent"
print ( "Grade:" ,a, b )
```

דוגמת ההרצה:

```
Enter grade:55
Grade: 55 Failed
>>> targil ( )
Enter grade:81
Grade: 81 Above Average
>>> targil ( )
Enter grade:77
Grade: 77 Average
>>> targil ( )
Enter grade:91
Grade: 91 Excellent
>>> targil ( )
Enter grade:67
Grade: 67 Below Average
```

תרגילי חזרה

רשום במשפט אחד מה קטע הקוד מבצע ואת תוצאת הריצה. (בדוק את תשובתך).

```
1. for j in range ( 4 , 10 ) :
    for k in range ( 4 , 10 ) :
        if j == k :
            print ( "*" , end=" " )
        else:
            print ( j * k , end=" " )
    print ( )
```

כיצד משתנה תשובתך אם את פקודת ה – if מחליפים ב: if j != k

```
2. for j in range ( 5 , 10 ) :
    for k in range ( 5 , 10 ) :
        if j % k :
            print ( "*" , end=" " )
        else:
            print ( j * k , end=" " )
    print ( )
```

```
3. for j in range ( 5 , 10 ) :
    for k in range ( 1 , j ) :
        if j + k >= 11 :
            print ( "*" , end = " " )
        else :
            print ( "?" , end = " " )
    print ( )
```

```
4. a = "123456789"
sum = 0.
cnt = 0
for j in range ( len ( a ) ) :
    if j % 2 :
        sum = sum + int ( a [ j ] )
        cnt = cnt + 1
print ( sum / cnt )
```

```
5. a = "123456789"
sum = 0.
cnt = 0
for j in range ( len ( a ) ) :
    if j % 2 == j % 5 :
        sum = sum + int ( a [ j ] )
        cnt = cnt + 1
print ( sum / cnt )
```

6.

```
a = "123456789"
b = ""
for j in range( len ( a ) ) :
    b = b + a [ j ]
    if j % 3 == 0 :
        b = b + a [ j ]
print ( a , ">>>", b )
```
7.

```
a = "abcdefgh"
b = ""
for j in range( len ( a ) ) :
    b = b + a [ j ]
    if j%3 == 0 :
        b = b + a [ j ]
    elif j % 2 == 0 :
        b = b + 2 * a [ j ]
print ( a , ">>>", b )
```
8.

```
a = "54678492"
b = "62384568"
c = ""
for j in range( len ( a ) ) :
    if int ( a [ j ] ) <= int ( b [ j ] ) :
        c = c + a [ j ]
    else :
        c = c + b [ j ]
print ( a , b , c )
```
9.

```
a = "54678492"
b = "62384568"
c = ""
for j in range( len ( a ) ) :
    if int ( a [ j ] ) % 2 :
        c = c + a [ j ]
    elif int ( b [ j ] ) % 2 :
        c = c + b [ j ]
    else :
        c = c + "0"
print ( a , b , c )
```
10.

```
a = "4356"
c = 0
for j in range( len ( a ) - 1 ) :
    if int ( a [ j ] + a [ j + 1 ] ) < 50 :
        c = c + eval ( a [ j ] + a [ j + 1 ] )
    else :
        c = c + eval ( a [ j + 1 ] + a [ j ] )
print ( a , c )
```

11. a = "857368132843745632837"

ma = 0

mb = 9

xa = 0

xb = 0

for j in range(len (a)) :

if int (a [j]) > ma:

ma = int (a [j])

xa = j

if int (a [j]) < mb :

mb = int (a [j])

xb = j

print (a , ma, xa, mb , xb)

12. x = 7

if x > 10:

print ("High")

elif x < 5:

print ("Low")

else:

print ("In between")

[תשובות](#)

[חזור לתוכן](#)

רשימות

מחרוזות הן סוג מסוים של רצפים sequence, אך קיימים סוגים נוספים (שלא על כולם נעבור במסגרת מדריך זה). סוג שימושי מאוד, אשר כבר הופיע בדוגמאות בפרקים הקודמים (מבלי להגדיר את שמו) הוא רשימה.

רשימה היא סדרה של נתונים, אשר בניגוד למחרוזת, יכולים להיות מסוגים שונים. כלומר ברשימה יכולים להיות נתונים מספריים, תווים, שברים וכד'. למשל:

```
>>> list1 = ["Hi there", 2, 2.25, 0x1b, "True", "Why?"]
>>> list1
['Hi there', 2, 2.25, 27, 'True', 'Why?']
```

כל הפעולות הקיימות לטיפול במחרוזות זמינות גם עבור רשימות, אך בנוסף קיים גם הבדל משמעותי אחד. רשימות ניתנות לשינוי, בעוד מחרוזות לא מאפשרות שינוי. לדוגמה, נניח שברצוננו לשנות את האיבר הראשון ברשימה:

```
>>> list1 [ 0 ] = "Hello"
>>> list1
['Hello', 2, 2.25, 27, 'True', 'Why?']
```

את דוגמת התוכנית להמרת ציונים לפי התקן האירופאי, ניתן לכתוב שצורה שונה תוך שימוש ברשימה.

```
grades = ["Failed", "Below Average", "Average", "Above Average",
          "Excellent", "Excellent!!" ]
a = int (input ( "Enter grade:" ) )
b = int ( ( a - 50 ) / 10 )
if b < 0 : b = 0
print ( "Grade:" ,a, grades [ b ] )
```

למה לדעתך נדרשת פקודת ה – if בתוכנית?⁴

⁴ ברשימות, כמו גם במחרוזות, אינדקס בעל ערך שלילי מסמן שיש להתחיל את האיברים מסוף הרשימה. ללא פקודת ה – if, ייווצרו אינדקסים שליליים ואז למשל מי שקיבל ציון 0, יקבל הערה של Excellent!!

תרגילי חזרה

רשום במשפט אחד מה קטע הקוד מבצע ואת תוצאת הריצה. (בדוק את תשובתך).

1.

```
lst = [1, 2, 3, 4, 5, 6, 7, 8, 9]
b = [ ]
a = 3
c = 7
b = b + lst [ a : c ]
print ( b )
```
2.

```
lst = [ 1 , 2 ]
b = [ ]
a = 3
c = 2
b = b + 2 * [ "*" ] + 3 * lst
print ( b )
```
3.

```
lst = [ 1 , 2 , 3 , 4 , 5 , 6 , 7]
for j in range ( 0 , len ( lst ) , 2 ) :
    lst [ j ] = "*"
print ( lst )
```
4.

```
lst = [ 1 , 2 , 3 , 4 , 5 ]
b = [ ]
for j in range ( len ( lst ) ) :
    b = b + lst [ 0 : j ]
print ( b )
```
5.

```
lst = [ 1 , 2 , 3 , 4 ]
b = [ ]
for j in range ( len ( lst ) ) :
    for k in range ( j ):
        b = b + lst [ k : j ]
print ( b )
```
6.

```
lst = [ 1 , 4 , 6 , 7 ]
mst = [ 1 , 3 , 5 , 9 ]
new = [ ]
for j in range ( len ( lst ) ) :
    new = new + lst [ j : j + 1 ] + mst [ j : j + 1 ]
print ( new )
```


7.

```
lst = [ 1 , 4 , 6 , 7 ]
mst = [ 1 , 3 , 5 , 9 ]
new = [ ]
for j in range ( len ( lst ) ) :
    new = new + lst [ j : j + 1 ] + mst [ - j : -j - 1 ]
print ( new )
```
8.

```
lst = [ 1 , 4 , 6 , 7 ]
mst = [ 1 , 3 , 5 , 9 ]
new = [ ]
for j in range ( 0 , len ( lst ) , 2 ) :
    new = new + lst [ j : j + 1 ] + mst [ - j : -j - 1 ]
print ( new )
```
9.

```
lst = [ 3 , 5 , 2 , 8 ]
mst = [ 2 , 7 , 1 , 6 ]
new = [ ]
for j in range(len(lst),0,-1):
    new = new + lst [ j : j + 1 ] + mst [ j : j + 1 ]
print ( new )
```
10.

```
lst = [ 3 , 1 , 3 , 5 , 6 ]
mst = [ 2 , 2 , 5 , 1 , 8 ]
new = [ ]
for j in range ( len ( lst ) , 0 , -2 ) :
    new = new + lst [ j : j + 1 ] + mst [ j : j + 1 ]
print ( new )
```
11.

```
for n in [ 1, 3 ] :
    for s in [ 'a' , 'b' ] :
        print (s * n , end = " ")
print ( )
```
12.

```
for x in [ 30 , 40 ] :
    for y in [ 1 , 2 , 3 ] :
        print ( x + y , end = " ")
print ( )
```
13.

```
k = [ 19 , 11 , 5 , -5 ]
for j in k:
    print ( j - 10 )
```
14.

```
k = [ 2 , 3 , 4 ]
if k [ 2 ] == 4 :
    print ( "Four" )
else:
    print ( "Three" )
```

15. `x = [10 , 20 , 30 , 40 , 50 , 60 , 70]`

```
print ( x [ 2 : 4 ] )
print ( x [ 2 : ] )
print ( x [ : ] )
print ( x )
print ( x [ : 5 ] )
print ( x [ -2 : ] )
print ( x [ -3 : -1 ] )
```

16. `x = [10 , 20 , 30 , 40 , 50 , 60 , 70]`

```
x [ 3 : 5 ] = [ 2 , 3 , 4 ]
x [ 4 : ] = [ ]
print ( x )
```

17. `x = [88 , 11 , 99 , 66 , 77 , 44 , 55 , 33]`

```
print ( "Length of x:" , len ( x ) )
i = 1
while i <= len ( x ) :
    print ( "Item ",i," = ",x [ i-1 : i ] )
    i = i + 1
```

18. `list = [2,4,7]`

```
for a in range(1,5):
    if a in list:
        print ( a, 'is ', end = " ")
    else:
        print ( a, 'is not', end = " ")
    print ( 'in', list )
```

19. `a = [0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9]`

```
for k in range ( int ( len ( a ) / 2 ) ) :
    a [ k ] , a [ -k -1 ] = a [ -k -1 ] , a [ k ]
```

20. `a = ["A", "B", "C", "D", "E", "F", "G", "H", "K"]`

```
for k in range ( len ( a ) ) :
    if k > len ( a ) / 2 :
        print ( ( len ( a ) - k ) * a [ k ] )
    else:
        print ( ( k + 1 ) * a [ k ] )
```

21. `a = ["A", "B", "C", "D", "E", "F", "G", "H", "K"]`

```
for k in range ( len ( a ) ) :
    print ( ( k + 1 ) * a [ k ] , 2 * ( len ( a ) - k ) * "-", ( k + 1 ) * a [ k ] )
```

תכנות בסיסי

```
22. a = [[ [ 12, 13 ], [ 14, 19 ], [ 10, 11 ] ], [[ 23, 34 ], [ 27, 26 ], [ 26, 27 ] ]]  
    print ( a [ 0 ] [ 1 ] [ 1 ], a [ 0 ] [ 2 ] [ 0 ], a [ 1 ] [ 0 ] [ 1 ], a [ 0 ] [ 2 ] [ 0 ] )  
    print ( a [ 1 ] [ 0 ] [ 0 ], a [ 1 ] [ 1 ] [ 1 ], a [ 1 ] [ 2 ] [ 0 ], a [ 1 ] [ 1 ] [ 0 ] )
```

```
23. listA = [1, 2, 3]  
    for indexA in range ( len ( listA ) ) :  
        for indexB in range ( len ( listA ) ) :  
            if indexA == indexB: continue  
            for indexC in range ( len ( listA ) ) :  
                if indexC == indexA:  
                    continue  
                elif indexC == indexB:  
                    continue  
                else:  
                    print ( listA [ indexA ], listA [ indexB ], listA [ indexC ] )
```

```
24. listA = [11, 43, 52, 21, 67, 34, 58, 61, 56, 81, 28, 84, 73, 66]  
    for indexA in range ( len ( listA ) ) :  
        temp = 0  
        for indexB in range ( 1 , len ( listA ) ) :  
            temp = listA [ indexB ]  
            if listA [ indexB ] < listA [ indexB - 1 ] :  
                listA [ indexB ] = listA [ indexB - 1 ]  
                listA [ indexB - 1 ] = temp  
    print ( listA ):
```

[תשובות](#)

[חזור לתוכן](#)

לולאות while

בניגוד ללולאות ה- for שפורטו בפרקים קודמים ואשר הינן מוחלטות (בכניסה ללולאה, מספר המחזורים כבר ידוע), לולאת while הן לולאות שאינן מוחלטות, כלומר בכל מחזור נבדק מחדש התנאי אם יש לבצע את המחזור או לא. פורמט הפקודה הוא:

```
while <condition>
    <body>
```

כמו במקרה של for, התנאי הוא תנאי בוליאני וגוף הלולאה מכיל פקודה אחת או יותר. הבדל משמעותי ביותר בין שני סוגי הלולאות הוא בקידום מונה הלולאה. לולאת for מקדמת את המונה עבורנו, בעוד שבלולאת while לעיתים אין מונה ובכל מקרה, עלינו לקדמו בעצמנו. אם לא נקדם אותו, או לא נדאג לשינוי התנאי הבוליאני, הלולאה תבוצע עד אשר נפסיק אותה ידנית (למשל ע"י Cntrl C).

לולאת while יעילה, למשל כאשר עלינו לקרוא קלט מהמשתמש ואין אפשרות (קלה) להגדיר מראש את כמות הקלט (או מספר האיברים שנקבל). במצב, כזה, ניתן להגדיר סימן מיוחד שמגדיר את סוף הקלט. זה כמובן חייב להיות תו שאינו חלק מהקלט הרגיל, למשל אם מדובר בציונים, ניתן להכניס ערך שלילי, שברור שאינו מייצג ציון (שחייב להיות בין 0-100). במקרה של ציונים, ניתן גם להשתמש בתווים (בניגוד למספרים).

דוגמת תוכנית:

```
print ( )
print ( 'To find the area of a rectangle,' )
print ( 'enter the width and height below.' )
print ( )
w = int ( input ( 'Width: ' ) )
while w <= 0 :
    print ( 'Must be a positive number' )
    w = int ( input ( 'Width: ' ) )

h = int ( input ( 'Height: ' ) )
while h <= 0 :
    print ( 'Must be a positive number' )
    h = int ( input ( 'Height: ' ) )

print ( 'Width =', w, 'Height =', h, 'so Area =', w * h )
```

ובהרצתה מתקבל:

```
To find the area of a rectangle,
enter the width and height below.
```

```
Width: 12
Height: 4
Width = 12 Height = 4 so Area = 48
דוגמת הרצה נוספת (כוללת שגיאות):
To find the area of a rectangle,
enter the width and height below.
```

```
Width: -2
```

תכנות בסיסי

Must be a positive number

Width: 0

Must be a positive number

Width: 3

Height: -4

Must be a positive number

Height: -1

Must be a positive number

Height: 12

Width = 3 Height = 12 so Area = 36

בדוגמה, ניתן לראות את פקודת ה- while, אשר לא מאפשרת לתוכנית להתקדם עד אשר המשתמש יכניס את הקלט הרצוי (מספר חיובי גדול מאפס).

תרגילי חזרה

רשום את תוצאת הריצה. (בדוק את תשובתך).

1.

```
j = 5
while ( j > 0 ) :
    print ( "j = " , j )
    j = - j
```
2.

```
s = 0
i = 10
while i > 0 :
    s = s + i
    i = i - 1
print ( s )
```
3.

```
x = 1
while x < 5 :
    print ( x, end = " " )
    x = x + 2
print ( )
```
4.

```
x = 0
while x < 10 :
    x = 2 * x + 1
    print ( x, end = " " )
print ( )
```
5.

```
n = 5
j = 1
k = 1
sum = 0
while ( j < n ):
    sum = sum + k
    k = k * 2
    j = j + 1
print ( sum )
```
6.

```
m = 3
n = 5
while n < 10 :
    m = n - 1
    n = 2 * n - m
print ( n, m )
```

```
7. temp = int ( input ( "Enter temperature please: " ) )
   while temp != 0:
       if temp >= 32:
           print ( "it is hot" )
       elif temp <= 13:
           print ( "it is cold" )
       else:
           print ( "it is just right" )
       temp = int ( input ( "Enter temperature please: " ) )
   print ( "Bye!" )
```

הנח שהמשתמש מקליד 22 ואחר כך 60 אחר כך 23 אחר כך 12 ובסוף 0.

```
8. jj = 11
   cnt = 0
   while jj > 0:
       if jj % 2 == 1:
           jj = int ( jj / 2 )
       else:
           jj = jj - 1
       cnt = cnt + 1
   print ( cnt, jj )
```

כיצד משתנה תשובתך אם משנים את הפקודה ראשונה ל – $jj = 19$?

```
9. print ( "Start" )
   i = 12
   j = 40
   tmp = j-i
   while( tmp > 4 ):
       j = j – int ( tmp / 4)
       i = i + tmp / 4
       print ( j , '\t' )
       tmp = j-i

   print ( "End" )
```

```
10. x = 1
    y = 1
    while x <= 5 :
        x = x * y
        y = y + 1
        print ( x )
```

11.

```
i = 1
print ( "-" * 50 )
while i < 11:
    n = 1
    while n <= 10:
        print ( "%4d" % (i * n), end = " " )
        n += 1
    print ( "" )
    i += 1
print ( "-" * 50 )
```
12.

```
cnt = 10
row = 1
while row <= cnt :
    column = 1
    while column<=row:
        print ( "*",end = " " )
        column = column + 1
    print ( )
    row = row + 1
print ( )
```
13.

```
j = 0
while j < 5 :
    n = 0
    while n < 10 :
        print ( n, end = " " )
        n = n + 1
    print ( )
    j = j + 1
```
14.

```
n = 9
i = 1
while i <= n :
    print ( i , "x 8 =" , i * 8 )
    i = i + 1
```
15.

```
var = 10
while var > 0:
    print ( 'Current variable value is :', var )
    var = var -1
    if var == 5:
        break
```


תכנות בסיסי

```
16. var = 10
   while var > 0:
       var = var - 1
       if var == 5:
           continue
       print ( 'Current variable value is :', var )
```

[תשובות](#)

[חזור לתוכן](#)

פונקציות

פונקציות כבר הופיע במדריך זה בעבר, אם כי ללא הסבר על מהותן. למשל `type()` או `input()` הוזכרו הפונקציות. פונקציה היא קטע קוד אשר נכתב פעם אחת, אבל אפשר לקרוא לו (להפעיל אותו) במקומות שונים בתוכנית. דבר חוסך כתיבה מיותר של קוד ובעיקר נוח מבחינה תחזוקתית, כאשר יש צורך לשנות, משנים רק במקום אחד.

פונקציה היא מעין תת תוכנית בתוך התוכנית ולכן יש לה שם ייחודי (לפיו מזהים ומפעילים אותה). לכל פונקציה חייב להיות קטע בו היא מוגדרת (Function Definition). לפונקציה אפשר גם להעביר פרמטרים (בשמות משתנים בתוך הסוגריים). לכן, אם למשל נגדיר פונקציה אשר מקבלת שני פרמטרים ומחזירה את מכפלתם:

```
def fun ( x , y ) :
    return x * y
```

בכל פעם שנקרא לפונקציה `fun` בתוספת שני מספרים נקבל כתוצאה את מכפלתם. למשל בהרצה בסביבה האינטראקטיבית נקבל:

```
>>> fun ( 3 , 8 )
24
>>> fun ( 5 , 5 )
25
>>> fun ( 1 , 13 )
13
```

את הפונקציה ניתן, כמובן, גם להגדיר כחלק מתוכנית ואז יש להקצות משתנה, אשר יקבל את תוצאת הפונקציה (בהנחה היא מחזירה תוצאה אחת, משום שפונקציה יכולה גם להחזיר יותר מתוצאה אחת). למשל:

```
>>> a = fun ( 2 , 11 )
>>> print ( a )
22
>>> b = fun ( 3 , 13 )
>>> print ( b )
39
```

את תוצאת הפונקציה אפשר גם לשלוח ישירות להדפסה ע"י:

```
>>> print ( fun ( "2" , 3 ) )
```

מה לדעתך תהיה תוצאת הפקודה האחרונה?⁵

כאשר לפונקציה מספר תוצאות, למשל שתיים, יש להגדיר שני משתנים אליהם יוכנסו התוצאות. הפקודה המחזירה את התוצאות היא `return`. למשל:

```
def sumDiff ( x , y ) :
    sum = x + y
    diff = x - y
    return sum, diff
```

⁵ עקב העובדה שאחד הפרמטרים הוא מחרוזת, פקודת הכפל שבתוך הפונקציה לא תתבצע כפקודת כפל, אלא תהפוך ל"חזרה" (או שכפול). במילים אחרות המחרוזת "2" תשופל שלוש פעמים והפונקציה תדפיס: 222

והקריאה תתבצע ע"י:

```
a,b = sumDiff (num1 , num2 )
```

תרגילי חזרה

רשום את תוצאת הריצה. (בדוק את תשובתך).

```
1. def comp ( x ) :
    if x < 3:
        print ( "A", end = " " )
    elif x > 10:
        print ( "B", end = " " )
    else:
        print ( "C", end = " " )
def targil ( ) :
    a = 5
    b = 12
    c = -3
    d = 9
    a1 = comp ( a )
    b1 = comp ( b )
    c1 = comp ( c )
    d1 = comp ( d )

2. def sum ( x ) :
    total = 0
    for j in range ( 1 , x ) :
        total = total + j
    return total

def targil ( ) :
    a = 7
    b = 11
    c = -3
    d = 6
    a1 = sum ( a )
    b1 = sum ( b )
    c1 = sum ( c )
    d1 = sum ( d )
    print ( a1 , b1 , c1 , d1 )
```

3. `def square (x) :`
`return x * x`

```
def targil ( ) :
    a = 0
    b = 2
    c = -3
    d = 4
    a1 = square ( 2 * a )
    b1 = square ( b * b )
    c1 = square ( c * 4 )
    d1 = square ( d - 2 )
    print ( a1 , b1 , c1 , d1 )
```

4. `def square2 (x) :`
`x = x + 2`
`return x * x`

```
def targil ():
    a = 0
    b = 2
    c = -3
    d = 4
    a1 = square2 ( 2 * a )
    b1 = square2 ( b * b )
    c1 = square2 ( c * 4 )
    d1 = square2 ( d - 2 )
    print ( a1 , b1 , c1 , d1 )
```

5. `def powers (x) :`
`return x ** 2 , x ** 3 , x ** 4`

```
def targil ( ) :
    a , b , c = powers ( 3 )
    print ( a , b , c )
    a,b,c = powers ( -2 )
    print ( a , b , c )
```

```

6. def multiprint (x, t) :
    for j in range ( x ) :
        print ( t )

def targil ( ) :
    a = 3
    txt1 = "Hello..."
    b = multiprint ( a , txt1 )
    a = 5
    txt1 = "Hello again..."
    b = multiprint ( a , txt1 )

7. def pralines (str , num ) :
    for n in range ( 0 , num ) :
        print ( str * (n + 1) )

def targil ( ) :

    pralines ( 'BOB' , 5 )
    print ( )
    pralines( 'xox ' , 6 )
    
```

[תשובות](#)

[חזור לתוכן](#)

עוד על פונקציות

בהמשך להסבר על הפונקציות, יש להוסיף שלפונקציות בעלות פרמטרים ניתן להגדיר ברירות מחדל. פירוש הדבר שהפונקציה עובדת כפי שתואר, אך אם כאשר קוראים לה לא מגדירים את אחד הפרמטרים, הפונקציה משתמשת בערך ברירת המחדל המוגדר בה. הדבר נעשה ע"י הוספת סימן שווה וערך לכל פרמטר שאנו רוצים להקצות לו ברירת מחדל. לדוגמה נשתמש בפונקציה multisum שהוגדרה בתרגיל קודם.

```
def multisum ( x = 3, t = 4 ) :
    sum = 0
    for j in range ( t ) :
        sum = sum + j * x
        x = x + 1
    return sum

def targil():
    a = 2
    b = 3
    print ( multisum ( a + 4 , b + 3 ) )
    print ( multisum ( ( a + 3 ) / 2 ) )
    print ( multisum ( ) )
```

ולאחר ההרצה מתקבלות התוצאות:

```
145
29.0
32
```

יכולת נוספת שיש להזכיר בהקשר של פונקציות היא רקורסיה. זו היכולת של פונקציה אחת לקרוא לעצמה (עד אשר מתקיים תנאי מסוים). יכולת זו נשענת על העובדה שהמשתנים הפנימיים בתוך הפונקציה זמינים רק לה.

למשל פונקציה רקורסיבית המחשבת עצרת.

```
def fact ( n ) :
    if ( n == 1 ) :
        return 1
    else:
        return n * ( fact ( n - 1 ) )
```

תרגילי חזרה

רשום את תוצאת הריצה. (בדוק את תשובתך).

1.

```
def sum ( n ) :
    if n == [ ] :
        return 0
    else:
        return n [ 0 ] + sum ( n [ 1 : ] )

def targil ( ) :
    a = [ 6 , 4 , 7 , 11 , 5 , 8 ]
    b = sum ( a )
    print ( b )
```
2.

```
def fun ( n , m ) :
    return m - n

def targil ( ) :

    print ( ( fun ( fun ( 1 , 2 ) , 3 ) ) )
    print ( ( fun ( fun ( 1 , 2 ) , fun ( 3 , fun ( fun ( 4 , fun ( 5 , 6 ) ) , 7 ) ) ) ) ) )
```
3.

```
def alpha ( x , y ) :
    return x + beta ( y , x )

def beta ( x , y ) :
    return y - x

def targil ( ) :

    print ( alpha ( 2 , 2 ) )
```
4.

```
def fun ( x ) :
    if x ==1 :
        return 0
    else :
        a = x - 1
        print ( a )
        fun ( a )

def targil ( ) :
    a = fun ( 10 )
```



```
5. def what ( n ) :
    if n == 0 :
        return 0
    else :
        return n + what ( n - 1 )
```

```
def targil ( ) :

    print ( what ( 7 ) )
```

```
6. def fibonacci( n ):

    if n == 0 :
        return 0
    elif n == 1 :
        return 1
    else:
        return fibonacci( n - 1 ) + fibonacci( n - 2 )
```

```
def targil():

    n = 7
    print ( fibonacci ( n ) )
```

```
7. def mult ( a ) :
    if a > 1000 :
        a = div ( a )
        return a
    return a * a
```

```
def div ( b ):
    return b / 1000
```

```
def targil ( ) :

    a = 10
    print ( mult ( mult ( mult ( mult ( a ) ) ) ) )
```

```

8. def twoa ( a ) :

    return 2 * a

def threea ( b ) :
    return 3 * b

def targil ( ) :

    print ( twoa ( threea ( twoa ( 2 ) + twoa ( 3 ) ) ) ) )

9. def prod ( a , b ) :

    return a * b

def sum ( a , b ) :
    return a + b

def targil ( ) :

    print ( prod ( prod ( 3 , 2 ) / sum ( 2 , 2 ) , sum ( 12 , 18 ) / prod ( 3 , 2 ) ) ) )

10. def sentence ( start, middles = [ 'long', 'short' ] , end = '.' ) :

    for m in middles:
        print ( start + m + end )

def targil ( ) :
    sentence ( 'This road is ' )
    print ( )
    sentence ( 'This is a ', [ 'new' , 'used' , 'strange' ] , ' car.' )
    print ( )
    sentence ( 'This lesson is ', end = ' and interesting.' )
    print ( )

```

```
11. def comp ( a , b ) :
    if a > b :
        return 1
    if a == b :
        return 0
    else:
        return -1
```

```
def targil ( ) :
```

```
    print ( comp ( 3 , 3 ) , comp ( 8 , 7 ) , comp ( 7 , 8 ) )
    print ( comp ( comp ( 4 , 2 ) + comp ( 5 , 4 ) , comp ( 4 , 5 ) + comp ( 2 , 4 ) ) )
    print ( comp ( comp ( comp ( 4 , 5 ) , comp ( 5 , 7 ) ) , comp ( 1 , 3 ) ) )
```

```
12. def yeter ( a=3 , b=4 ) :
    return ( a ** 2 + b ** 2 ) ** 0.5
```

```
def targil ( ) :
```

```
    print ( yeter ( ) )
    print ( yeter( b = 32 / 8 ) )
    print ( yeter( 1 + 1 + 1 ) )
    print ( yeter ( 7 , 24 ) )
    print ( yeter ( yeter ( 9 , 12 ) / 5 , yeter ( 12 , 5 ) // 3 ) )
```

```
13. def mult ( n , end ) :
    i = 1
    while i <= end :
        print ( n * i , '\t' , end = " " )
        i = i + 1
```

```
def print_mult ( end ) :
    i = 1
    while i <= end :
        mult ( i , end )
        i = i +1
```

```
def targil ( ) :
    print_mult(10)
```

```
14. def mult ( n , end ) :  
    i = 1  
    while i <= end :  
        print ( n * i , '\t' , end = " " )  
        i = i + 1  
    print ( )  
  
def print_mult ( end ) :  
    i = 1  
    while i <= end :  
        mult ( i , i )  
        i = i + 1  
  
def targil ( ) :  
    print_mult(10)
```

[תשובות](#)

[חזור לתוכן](#)

טיפול ברשימות

בפיתון מוגדרות מספר פונקציות ספריה אשר נועדו לספק יכולות מורחבות בטיפול ברשימות. כדי לקבל גישה לפונקציות אלה, יש לכלול בתוכנית את הפקודה: `import string`.

פקודה זו מייבאת את הפונקציות המאפשרות יכולות אלה וללא הפקודה, אין אפשרות להשתמש בהן (שימוש בפונקציות ללא פקודת ה- `import` יגרור הודעת שגיאה).

כאמור כדי ליצור רשימה, יש להגדיר לה שם המלווה בסוגריים מרובעים. האיברים בתוך הסוגריים הם איברי הרשימה. ניתן גם להגדיר רשימה ריקה כמו: `a = []`. בהגדרת רשימה, ניתן גם לתת לה יותר משם אחד. למשל הפקודה: `a=b=[]` מגדירה רשימה אחת, בעלת שני שמות.

בנוסף לאפשרויות לטיפול ברשימות שתוארו בפרקים קודמים, קיימת גם אפשרות של לולאה על איברי הרשימה.

למשל:

```
>>> x = [ 1, 2, 3, 4, 5 ]
>>> for k in x:
    print ( k, end = " " )
```

```
1 2 3 4 5
```

פונקציות הרשימה מספקות אפשרות לביצוע של איטרציות על איברי הרשימה ע"י שימוש בפונקציה `iter`. למשל עבור הדוגמה הקודמת ניתן להגדיר משתנה נניח `k` אשר ישמש כמונה האיברים ברשימה. בכל פעם שנקרא לפונקציה `next(k)` נקבל את האיבר הבא ברשימה.

```
>>> x = [ 1, 2, 3, 4, 5 ]
>>> k = iter ( x )
>>> print ( next ( k ) )
1
>>> print ( next ( k ) )
2
```

בעזרת פונקציה זו ניתן להתייחס לרשימה בעזרת מספר מונים ולקדם כל מונה בהתאם לצורך, למשל:

```
>>> y = [ 1, 2, 3, 4, 5, 6 ]
>>> j = iter ( y )
>>> n = iter ( y )
>>> print ( next ( j ) , next ( j ) , next ( j ) )
1 2 3
>>> print ( next ( n ) , next ( j ) )
1 4
```

את הפונקציה `len` המחזירה את אורך המחרוזת או הרשימה, כבר אנו מכירים. קיימות פונקציות נוספות כמו למשל הפונקציה `sum`, אשר מחזירה את סכום האיברים ברשימה. למשל:

```
>>> y = [ 21, 22, 35, 67, 31, 45, 77 ]
>>> b = sum ( y )
>>> print ( b )
298
```

תכנות בסיסי

```
>>> print ( sum ( y ) / len ( y ) )
42.5714285714
```

בנוסף לפונקציה המסכמת את איברי הרשימה קיימות כמובן גם פונקציות המחזירות את האיבר הקטן ביותר והגדול ביותר ברשימה. דוגמה:

```
>>> w = [ 21, 34, 57, 89, 56, 25, 37, 47, 81, 27]
>>> print ( min ( w ) , max ( w ) )
21 89
```

כאמור, ברשימה ניתן להחליף את האיברים (אחד או יותר). דרך פשוטה לבצע זאת, היא ע"י השמה המקום הרצוי למשל אם ברצוננו להחליף את האיבר במקום השלישי (אשר האינדקס שלו הוא 2), אפשר לבצע זאת ע"י:

```
y = [ 21, 22, 35, 67, 31, 45, 77 ]
>>> print ( y )
[21, 22, 35, 67, 31, 45, 77]
>>> y [ 2 ] = 27
>>> print ( y )
[21, 22, 27, 67, 31, 45, 77]
```

נקודה חשובה שיש לשים אליה לב היא כאשר אנחנו מעתיקים רשימות. ההעתקה הרגילה לא יוצרת רשימה נוספת, אלא יוצרת רק הפנייה חדשה, כפי שניתן לראות בדוגמה הרצ"ב:

```
>>> y = [ 21, 22, 35, 67, 31, 45, 77 ]
>>> print ( y )
[21, 22, 35, 67, 31, 45, 77]
>>> y [ 2 ] = 27
>>> print ( y )
[21, 22, 27, 67, 31, 45, 77]
>>> x = y
>>> print ( x )
[21, 22, 27, 67, 31, 45, 77]
>>> x [ 3 ] = 11
>>> print ( y )
[21, 22, 27, 11, 31, 45, 77]
```

לכאורה, העתקנו את הרשימה ל- x, אך כפי שניתן לראות לאחר ששינינו את x, גם y השתנתה. האיבר שמספרו 3 (האיבר הרביעי) היה 67, אך לאחר ששינינו אותו ב- x ל- 11, גם ב- y ערכו הפך 11. במילים אחרות, x ו y מצביעים לאותה רשימה ולא לשתי רשימות נפרדות. אם ברצוננו ליצור רשימה נפרדת, יש להשתמש בפקודת השמה שונה במקצת:

```
a = b [ : ]
```

כפי שניתן לראות בדוגמה:

```
>>> y = [ 27, 23, 43, 55, 19, 63, 29 ]
>>> print ( y )
[27, 23, 43, 55, 19, 63, 29]
>>> x = y [ : ]
>>> print ( x )
[27, 23, 43, 55, 19, 63, 29]
```

תכנות בסיסי

```
>>> x [ 2 ] = 73
>>> print ( x )
[27, 23, 73, 55, 19, 63, 29]
>>> print ( y )
[27, 23, 43, 55, 19, 63, 29]
```

במקרה זה שינינו את x והרשימה y לא השתנתה.

בנוסף, קיימות גם פונקציות אחרות המאפשרות להרחיב את הרשימה: `append` מוסיפה איבר אחד בסוף הרשימה, `extend` מוסיפה איברים מרשימה (או מחרוזת) אחרת ו- `insert` מוסיפה איבר במיקום מסוים, תוך הזזת שאר האיברים ימינה. דוגמה:

```
>>> t = [ 44, 55 ]
>>> w = [ 22, 33, 11 ]
>>> s = "ABC"
>>> t.append ( 66 )
>>> t
[44, 55, 66]
>>> t.extend ( s )
>>> t
[44, 55, 66, 'A', 'B', 'C']
>>> t.extend ( w )
>>> t
[44, 55, 66, 'A', 'B', 'C', 22, 33, 11]
```

אפשר גם להשתמש בהחלפה של איברים. למשל בדוגמה הרצ"ב מוגדרת רשימה t בעלת ארבעה איברים. ורשימה r בעלת שלושה איברים. בפקודת ההחלפה אנו מחליפים את שני האיברים הראשונים ברשימה t באיברים מהרשימה r.

```
>>> t = [ 1, 2, 3, 4 ]
>>> r = [ 7, 8, 9 ]
>>> t [ 0 : 2 ] = r
>>> t
[7, 8, 9, 3, 4]
```

כדי למחוק איברים מהרשימה ניתן להשתמש בפקודה `del`. במקרה זה יש לקבוע את האיבר שברצוננו למחוק (אחד או יותר). חשוב לציין שהפקודה `del` זמינה ללא צורך ביבוא הספרייה `import string`). למשל, המחרוזת t כוללת שמונה איברים. בפקודת המחיקה הראשונה, אנחנו מוחקים את האיבר הראשון ופקודת המחיקה השנייה אנחנו מוחקים את האיברים השני והשלישי.

```
>>> t = [ 1, 2, 3, 4, 5, 6, 7, 8 ]
>>> del t [ 0 ]
>>> t
[2, 3, 4, 5, 6, 7, 8]
>>> del t [ 1:3 ]
>>> t
[2, 5, 6, 7, 8]
```

הספרייה לתמיכה ברשימות ומחרוזות כוללת פונקציות מחיקה נוספות: `pop` שנועדה לשלוף (תוך מחיקה) של איבר ממקום כלשהו ברשימה ו- `remove` אשר מוחקת איבר לא לפי מיקומו, אלא לפי ערכו. פונקציה ה- `pop` ללא פרמטרים מוחקת את האיבר האחרון ברשימה ולחילופין היא

תכנות בסיסי

יכולה לקבל פרמטר אחד המציין את מספרו של האיבר למחיקה. הפונקציה מחזירה את הערך שנמחק. פונקציה ה- `remove` מקבלת פרמטר אחד שהינו ערך של איבר ברשימה והיא מחפשת ברשימה את המופע הראשון של ערך זה ומוחקת אותו.

```
>>> t = [ 1, 2, 3, 4, 5, 6, 7, 8 ]
>>> print ( t.pop ( ) )
8
>>> t
[1, 2, 3, 4, 5, 6, 7]
>>> print ( t.pop ( 3 ) )
4
>>> t
[1, 2, 3, 5, 6, 7]
>>> t = [ 1, 2, 3, 4, 1, 2, 3, 4, 1, 1, 2]
>>> t.remove ( 2 )
>>> t
[1, 3, 4, 1, 2, 3, 4, 1, 1, 2]
>>> t.remove ( 2 )
>>> t
[1, 3, 4, 1, 3, 4, 1, 1, 2]
>>> t.remove ( 2 )
>>> t
[1, 3, 4, 1, 3, 4, 1, 1]
>>> t.remove ( 2 )
```

```
Traceback (most recent call last):
  File "<pyshell#141>", line 1, in <module>
    t.remove(2)
ValueError: list.remove(x): x not in list
```

בדוגמה לעיל, השתמשנו בפונקציה `remove` מספר פעמים ובכל פעם מחקנו את האיבר שערכו 2. הפונקציה מחפשת את המופע הראשון של 2 ומוחקת את האיבר המתאים. בקריאה השנייה לפונקציה, שוב היא מוחקת את המופע הראשון שהיא מוצאת. לאחר שלא נשארו מופעים של 2 ברשימה, מתקבלת הודעת שגיאה.

לפעמים יש צורך לסדר את הרשימה בסדר הפוך (או לחילופין להוסיף לה איברים בתחילתה – אך ללא התקורה של הוספה מעין זו). לצורך זה קיימת פונקציה נוספת `reverse`. דוגמה:

```
>>> t = [ 1, 2, 3, 4, 5, 6, 7 ]
>>> t.reverse ( )
>>> t
[7, 6, 5, 4, 3, 2, 1]
>>> t.append ( 0 )
>>> t
[7, 6, 5, 4, 3, 2, 1, 0]
>>> t.reverse ( )
>>> t
[0, 1, 2, 3, 4, 5, 6, 7]
```


תכנות בסיסי

כדי לבדוק אם ערך מסוים נמצא ברשימה, אפשר להשתמש באופרטור in למשל כחלק מפקודת תנאי:

```
>>> t = [ 0, 1, 2, 3, 4, 5, 6, 7 ]
>>> if 7 in t : print ( "OK" )
```

OK

אולם לפעמים ברצוננו לא רק לדעת שהאיבר נמצא, אלא גם לאתר את מיקומו ברשימה. לשם כך קיימת הפונקציה index. למשל בקטע הקוד הרצ"ב ברצוננו לבדוק את מיקומו של המספר 8 ברשימה:

```
>>> t = [1, 3, 0, 8, 5, 4, 7, 9]
>>> print ( t.index ( 8 ) )
3
```

ואכן הפונקציה החזירה שהוא נמצא במקום מס' 3 (המיקום הרביעי) ברשימה. פונקציה שימושית נוספת היא הפונקציה count, אשר מחזירה את מספר המופעים של ערך מסוים ברשימה. למשל:

```
t = [ 1, 0, 1, 2, 1, 3, 1, 4, 1, 5, 1 ]
>>> print ( t.count ( 1 ) )
6
```

הפונקציה האחרונה בהקשר זה היא sort, אשר ממיינת את הרשימה.

```
>>> r = [ 8, 5, 7, 3, 7, 4, 2, 9, 1, 6 ]
>>> r.sort ( )
>>> r
[1, 2, 3, 4, 5, 6, 7, 7, 8, 9]
```

תרגילי חזרה

רשום במשפט אחד מה קטע הקוד מבצע ואת תוצאת הריצה. (בדוק את תשובתך).

1.

```
a = ["A", "B", "C", "D", "E", "F", "G", "H", "K"]
cnt = 1
for k in a:
    print ( k * cnt )
    cnt = cnt+1
```
2.

```
a = [22, 33, 21, 35, 16, 11, 27, 30, 23, 25]
b = 13
a.reverse ()
a.insert ( 5 , b )
a.sort ()
a.remove ( 25 )
a.reverse ()
print ( a )
```
3.

```
a = [22, 33, 21, 35]
b = 13
for k in a:
    a.append ( b )
    a.reverse ()
    a.pop ()
a.reverse ()
print ( a )
```
4.

```
a = [22, 33, 21, 35, 43, 13, 19]
for k in 0 , len ( a ) , 2 :
    a.insert ( k , a.pop ( ) )
    a.reverse ()
print ( a )
```
5.

```
a = [22, 13, 22, 19, 13, 13, 19]
for k in range ( len ( a ) ) :
    a.insert( k , a.count ( a [ k ] ) )
    a.reverse ()
print ( a )
```
6.

```
a = [22, 13, 22, 19, 13, 13, 19]
for k in range ( len ( a ) ) :
    a.insert(k, a.index ( a [ k ] ) )
    a.reverse ()
print ( a )
```

7.

```
a = [20, 14, 20, 19, 14, 19, 19, 14]
for k in a :
    if a.count ( k ) < 3 :
        a.remove ( k )
a.sort ( )
print ( a)
```
8.

```
a = [ 11, 22, 33, 44, 55 ]
for k in range ( len ( a ) ) :
    a.reverse ( )
    a.append ( a [ k ] )
print ( a)
```
9.

```
a = [ 11, 22, 33, 44, 55 ]
for k in range ( len ( a ) - 1 ) :
    a.insert ( k + 2 , a [ k ] + a [ k + 1 ] )
print ( a)
```
10.

```
a = [ 12, 14, 15, 17, 19, 22, 26 ]
for k in range ( int ( len ( a ) / 2 ) ) :
    a.insert ( k , a.pop ( ) + a.pop ( ) )
print ( a)
```
11.

```
a = [ 12, 14, 15, 17, 19, 22, 26, 29, 31, 13, 34 ]
for k in range ( int ( len ( a ) / 3 ) ) :
    a.pop ( )
    a.reverse ( )
    a.pop ( )
print ( a)
```
12.

```
a = [ 11, 19, 26, 15, 27, 14, 22, 28 ]
for k in range ( len ( a ) ) :
    a.insert ( 0 , a.pop ( ( k + int ( len ( a ) ) / 2 ) ) )
print ( a)
```
13.

```
a = [ 9, 3, 4, 5, 2, 7, 6, 8 ]
b=0
for k in range ( len ( a ) ) :
    b=b+a.pop()
    a.insert(0,b)
print ( a)
```

```

14. a = [ 19, 13, 14, 15, 12, 17, 16, 18 ]
    b=0
    for k in range ( len ( a ) ) :
        if not k % 2 :
            b = b + a.pop ( )
        else:
            b = b - a.pop ( )
        a.insert ( 0 , b )
    print ( a )

15. a = [ 29, 23, 24, 25, 22, 27, 26, 28 ]
    for k in range ( int ( len ( a ) / 2 ) ) :
        if not k % 2 :
            a.pop ( )
        else:
            a.pop ( k )
    print ( a )

16. a = [ 39, 33, 34, 35, 32, 37, 36, 38 ]
    for k in range ( int ( len ( a ) / 2 ) ) :
        if k%2:
            a.append ( a.pop ( ) )
        else:
            a.append ( a.pop ( k ) )
    print ( a )

```

[תשובות](#)

[חזור לתוכן](#)

פעולות בוליאניות

סוג הנתונים הבוליאניים (משתנה בעל שני מצבים) הוגדרו כבר בפרקים קודמים. אמנם הם משמשים כתנאים בביצוע פקודות, אך יש להם שימושים נוספים. קיימות פעולות בין מספרים בוליאניים ובתרגול זה נתייחס לשלוש הפעולות העיקריות:

and – בהיבט של "וגם", אשר מקבל שני אופרנדים בוליאניים ומחזיר תשובה שהיא אמת (True), רק אם שניהם true.

or – מחזיר תשובת אמת אם אחד משני האופרנדים אמת. במילים אחרות, or מחזיר תשובה שקרית אם ורק אם שני האופרנדים שקריים.

xor – (Exclusive or) שדומה ל – or בהבדל אחד שהוא מחזיר תשובה שקרית אם שני האופרנדים שקריים וגם אם שניהם אמת. במילים אחרות, xor מחזיר תשובת אמת אם שני האופרנדים שונים (אחד אמת והשני שקר).

בנוסף, קיימת גם פעולה not המקבלת אופרנד אחד ומחזירה את ההפכי שלו. אם הוא אמת, היא מחזירה שקר ואם הוא שקר, היא מחזירה אמת.

בביטוי מורכב, סדר הקדימויות הוא:

not
and
or

ניתן תמיד להשתמש בסוגריים לוודא סדר שונה, או כדי להבטיח שהקורא מבין את המשמעות. למשל:

```
>>> a = True
>>> b = False
>>> a and b
False
>>> a or b
True
>>> not a
False
>>> a and not b
True
>>> a and b or b
False
>>> not a and b
False
>>> not ( a and b )
True
>>> not ( ( a and a ) and ( b and b ) )
True
>>> a and not a
False
>>> a or a
True
>>> a or not a
True
```

תכנות בסיסי

```
>>> not ( b or b )
True
```

חשוב לציין שסוג הנתונים הבוליאניים נשמר ע"י פיתון בערכים של 0 ו 1 (כאשר 1 מציין אמת). אולם, הפעולות הבוליאניות לא מוגבלות רק לשימוש בערכים 0 ו 1. כל ערך שאינו 0 מוערך ע"י פיתון שאמת ואילו 0 נשאר שקרי. במחרוזות הדבר שונה מעט מחרוזת ריקה נחשבת שקרית וכל ערך אחר נחשב כאמת. דוגמאות:

```
>>> bool ( 0 )
False
>>> bool ( 1 )
True
>>> bool ( 122 )
True
>>> bool ( "Shalom" )
True
>>> bool ( [ 1, 2 ] )
True
>>> bool ( )
False
>>> bool ( [ ] )
False
```

ניתוח הביטויים הבוליאניים נעשה בצורה שתחסוך במשאבים.

x and y – קודם מערכים את x . אם הוא שקרי (פירוש הדבר שכל הביטוי שקרי) ואז מחזירים את x כתוצאת הניתוח. אם הוא אמת, (פירוש הדבר שתוצאת הביטוי נקבעת על פי y) מחזירים את y כתוצאת הניתוח.

x or y – מערכים את x . אם הוא אמת (פירוש הדבר שהביטוי אמת) ואז מחזירים את x כתוצאת הניתוח. אם הוא שקר, פירוש הדבר שתוצאת הניתוח נקבעת על פי y ואז מחזירים את y כתוצאת הניתוח.

תרגילי חזרה

רשום את תוצאת הריצה. (בדוק את תשובתך).

1. `def comp (x , y) :`

```

    if x == y:
        print ( "A", end = " " )
    elif x < 5 and y > 2:
        print ( "B", end = " " )
    if x > 2 or y > 4:
        print ( "C", end = " " )

```

`def targil () :`

```

    a = comp ( 5 ,5 )
    print ( )
    a = comp ( 1 ,1 )
    print ( )
    a = comp ( 5 ,3 )

```

2. `def targil () :`

```

    x = 22
    if not ( x > 10 and x < 30 ) or x == 23 :
        print ( "Out of range" )
    else:
        print ( "In range" )

```

כיצד משתנה תשובתך אם את השורה הראשונה משנים ל – $x = 23$, או ל – $x = 31$?

3. `def targil () :`

```

    a1 = [ 0 , 15 , 21 , 11 , 13 , 10 , 22 , 6 ]
    for j in range ( len ( a1 ) ) :
        a = a1 [ j ]
        if a > 10 and a % 6 == 3 :
            print ( "A" )
        elif a > 10 and a < 20 :
            print ( "B" )
        else :
            print ( "C" )

```

4. `def targil () :`
 `a1 = [0 , 3 , 11 , 13 , 3 , 6 , 2 , 12]`
 `for j in range (len (a1)) :`
 `a = a1 [j]`
 `if a > 0 and a <10 :`
 `print ("Correct")`
 `else:`
 `print ("Wrong")`
5. `def targil () :`
 `a1 = [0 , 3 , 6 , 12 , 9 , 18 , 15 , 10]`
 `for j in range (len (a1)) :`
 `a = a1 [j]`
 `if a % 2 == 0 and a % 3 == 0 :`
 `print ("Correct")`
 `else:`
 `print ("Wrong")`
6. `def targil () :`
 `a1 = [0 , 3 , 6 , 12 , 9 , 18 , 15 , 10]`
 `for j in range (len (a1)) :`
 `a = a1 [j]`
 `if a % 2 == 0 or a % 3 == 0 :`
 `print ("Correct")`
 `else:`
 `print ("Wrong")`

[תשובות](#)

[חזור לתוכן](#)

פונקציות לטיפול במחרוזות

כפי שראינו, בפיתון קיימות פונקציות ספרייה מוכנות מראש שנועדו לטפל ברשימות. אולם פונקציות אלה היוו רק חלק מתוך ההיצע הקיים. פונקציות נוספות נועדו לספק יכולות לטיפול במחרוזות. לכל הפונקציות מבנה אחיד הכולל שתי מילים המופרדות ע"י נקודה ולאחריהן סוגריים עם או בלי פרמטרים. המלה הראשונה מציינת את שם המחרוזת עליה ברצוננו להפעיל את הפונקציה ואילו המלה השנייה מגדירה את הפונקציה. לפעמים הפונקציה כוללת גם פרמטרים (חובה ורשות) ואלה יופיעו בסוגריים לאחר שם הפונקציה. אם אין פרמטרים, יופיעו סוגריים ריקים.

1. capitalize - מחזירה עותק של המחרוזת, כאשר האות הראשונה (ורק היא) במחרוזת היא אות גדולה (Capital Letter).

```
>>> a = " this is an example "
>>> b = a.capitalize()
>>> a
' this is an example '
>>> b
' this is an example '
>>> a = "this is an example"
>>> c = a.capitalize()
>>> c
'This is an example'
>>>
```

בדוגמה ניתן לראות שבתחילה המחרוזת כללה רווח כתו ראשון ולכן כאשר יצרנו העתק (b), האות השנייה לא הפכה לאות גדולה (משום שאינה במקום הראשון במחרוזת). בדוגמה השנייה, a מוגדרת כך שהתו הראשון הוא אות ולכן במקרה זה הפונקציה capitalize אכן החליפה את האות הראשונה באות גדולה.

2. center – מחזירה מחרוזת שהיא העתק של המחרוזת המקורית כאשר היא ממורכזת בהתאם לפרמטרים הנוספים בפונקציה. פרמטר ראשון הינו חובה והוא מגדיר את גודל המחרוזת החדשה (אשר בתוכה תמצא המחרוזת המקורית). פרמטר שני שהינו רשות מגדיר את תו המילוי. ברירת המחדל היא רווחים. אם הפרמטר הראשון קטן מגודל המחרוזת, פיתון עדיין תחזיר את המחרוזת בשלמותה.

```
>>> a = "shalom"
>>> a.center ( 5 )
'shalom'
>>> a.center ( 10 )
' shalom '
>>> a.center ( 12, "*" )
'***shalom***'
```

3. count - מחזירה את מספר המופעים של תת המחרוזת המוגדרת כפרמטר ראשון. פרמטר שני שהינו רשות מגדיר את מספר האיבר הראשון במחרוזת המקורית ממנו מתחילים לסרוק (ברירת המחדל היא אפס – או תחילת המחרוזת) הפרמטר השלישי שהינו רשות מגדיר את האיבר האחרון לסריקה (ברירת המחדל היא האיבר האחרון במחרוזת).

```
>>> a = "abcbcabbbacabbcc"
>>> a.count ( "ab" )
3
>>> a.count ( "b" )
6
```

```
>>> a.count ( "c", 1, 6 )
2
```

4. `endswith` – פונקציה בוליאנית המחזירה אמת (True) או שקר (False), עם המחרוזת המסופקת כפרמטר ראשון לפונקציה נמצאת בסוף המחרוזת המקורית. כמו במקרה של הפונקציה `count`, גם כאן ניתן גם להשתמש בשני פרמטרי רשות המגדירים את גבולות המחרוזת הנבדקת (ברירות המחדל הן מתחילת המחרוזת ועד סופה).

```
>>> a = "abcbcabbbacabbcc"
>>> a.endswith ( "c", 0, 5 )
True
>>> a.endswith ( "a", 1, 7 )
False
>>> a.endswith ( "a", 3, 3 )
False
>>> a.endswith ( "a", 3, 6 )
True
>>> a.endswith ( "cc" )
True
```

5. `find` – מחזירה את המיקום הראשון שבו המחרוזת המועברת כפרמטר נמצאת בתוך המחרוזת הנבדקת. כמו במקרים קודמים ניתן להגדיר את טווח החיפוש ע"י שימוש בשני פרמטרי רשות אשר מגדירים את גבולות המחרוזת הנבדקת (ברירות המחדל הן מתחילת המחרוזת ועד סופה). המספר המוחזר הוא המיקום במחרוזת המקורית, גם אם החיפוש הוגבל רק לחלק מהמחרוזת. אם המחרוזת (הפרמטר) לא נמצאת במחרוזת המקורית, מוחזר הערך -1.

```
>>> a = "abcdefabcdefaabbccddaabbccdd"
>>> a.find ( "c" )
2
>>> a.find ( "cc" )
16
>>> a.find ( "dd", 10 )
18
>>> a.find ( "ee" )
-1
>>> a.find ( "e" )
4
```

6. `index` – פונקציה זהה ל- `find` בהבדל אחד. אם `index` לא מוצאת את המחרוזת במחרוזת המקורית, היא מחזירה `ValueError` (במקום ה-1 אשר `find` מחזירה).

```
>>> a = "wsdfredfredfgt"
>>> a.find ( "r" )
4
>>> a.index ( "r" )
4
>>> a.find ( "q" )
-1
>>> a.index ( "q" )
```

```
Traceback (most recent call last):
  File "<pyshell#55>", line 1, in <module>
    a.index("q")
ValueError: substring not found
```

7. `isalnum` – פונקציה בוליאנית אשר מחזירה אמת אם הערכים במחרוזת כולם תווים אלפאנומריים (אותיות או מספרים) ובתנאי שבמחרוזת לפחות תו אחד.

```
>>> a = "1234cdf"
>>> b = "acvfde"
>>> c = "12=/45"
>>> a.isalnum ( )
True
>>> b.isalnum ( )
True
>>> c.isalnum ( )
False
```

8. `isalpha` – פונקציה בוליאנית אשר מחזירה אמת אם הערכים במחרוזת כולם תווים (אותיות) ובתנאי שבמחרוזת לפחות תו אחד.

```
>>> a = "1234cdf"
>>> b = "acvfde"
>>> c = "12=/45"
>>> a.isalpha ( )
False
>>> b.isalpha ( )
True
>>> c.isalpha ( )
False
```

9. `isdigit` – פונקציה בוליאנית אשר מחזירה אמת אם הערכים במחרוזת כולם ספרות ובתנאי שבמחרוזת לפחות תו אחד.

```
>>> a = "1234cdf"
>>> b = "acvfde"
>>> c = "12=/45"
>>> d = "123456"
>>> a.isdigit ( )
False
>>> b.isdigit ( )
False
>>> c.isdigit ( )
False
>>> d.isdigit ( )
True
```

10. `islower` – פונקציה בוליאנית אשר מחזירה אמת אם הערכים במחרוזת כולם אותיות קטנות ובלבד שבמחרוזת לפחות אות אחת.

```
>>> a = "123"
>>> a.islower()
False
>>> b = "a123"
>>> c = "ABC"
>>> b.islower ( )
```

```
True
>>> c.islower ( )
False
>>> d = "abcder"
>>> d.islower ( )
True
>>> e = "Abcd"
>>> e.islower ( )
False
```

11. isupper – פונקציה בוליאנית (הפוכה ל – islower) אשר מחזירה אמת אם הערכים במחרוזת כולם אותיות גדולות ובלבד שבמחרוזת לפחות אות אחת.

```
>>> a = "Abc"
>>> b = "ABC123"
>>> c = "ccc"
>>> d = "123BBv"
>>> e = "123"
>>> a.isupper ( )
False
>>> b.isupper ( )
True
>>> c.isupper ( )
False
>>> d.isupper ( )
False
>>> e.isupper ( )
False
```

12. istitle – פונקציה בוליאנית אשר מחזירה אמת אם כל המילים המרכיבות את המחרוזת מתחילות באות גדולה וממשיכות באותיות קטנות (כמו כותרת של מסמך או מאמר) ובתנאי שהמחרוזת אינה ריקה.

```
>>> a = "the Cat ran up the tree"
>>> a.istitle ( )
False
>>> a.title ( )
'The Cat Ran Up The Tree'
>>> b = a.title ( )
>>> b.istitle ( )
True
```

13. join – פונקציה אשר מחזירה מחרוזת המורכבת מאיברי המחרוזת המועברת כפרמטר וביניהם מוכנסת מחרוזת המקור.

```
>>> a = "123"
>>> a.join ( a )
'112321233'
>>> a = "123"
>>> b = "abcdef"
>>> a.join ( b )
'a123b123c123d123e123f'
>>> a = "---"
>>> a.join("SHALOM")
'S---H---A---L---O---M'
```

14. `ljust` – פונקציה דומה לפונקציה `center`, אך בניגוד ל-`center` אשר ממרכזת את המחרוזת המקורית, `ljust` מחזירה אותה כאשר היא מוצמדת לצד שמאל. פרמטר ראשון הינו פרמטר חובה והוא מגדיר את גודל המחרוזת החדשה (אשר בתוכה תמצא המחרוזת המקורית). פרמטר שני שהינו רשות מגדיר את תו המילוי. ברירת המחדל היא רווחים. אם הפרמטר הראשון קטן מגודל המחרוזת, פיתון עדיין תחזיר את המחרוזת בשלמותה.

```
>>> a = "shalom"
>>> a.center ( 12 )
' shalom '
>>> a.ljust ( 12 )
'shalom '
>>> a.ljust ( 12 , "*" )
'shalom*****'
```

15. `lower` – פונקציה אשר מחזירה מחרוזת שהיא העתק של המחרוזת המקורית, כאשר כל האותיות הגדולות הפכו לקטנות. אם אין מה להפוך, תוחזר המחרוזת המקורית.

```
>>> a = "ABCDEF"
>>> a.lower ( )
'abcdef'
>>> b = "123"
>>> b.lower ( )
'123'
>>> c = "ddd"
>>> c.lower ( )
'ddd'
```

16. `lstrip` – פונקציה אשר מחזירה מחרוזת שהיא העתק של המחרוזת המקורית לאחר שקוצץ ממנה חלק בצידה השמאלי. הפונקציה מקבלת פרמטר רשות אחד המגדיר את התווים שיש להשמיט מהמחרוזת המקורית. ברירת המחדל היא השמטת הרווחים שנמצאים בצד שמאל

```
>>> a="abcdefgh ijklm nopq"
>>> a.lstrip ( "dcecbba" )
'fgh ijklm nopq'
>>> b = "the cat ran out"
>>> b.lstrip ( "the cat" )
'ran out'
>>> c = "    Hi everybody "
>>> c.lstrip ( )
'Hi everybody '
```

17. `replace` – פונקציה אשר מחזירה מחרוזת שהיא העתק של המחרוזת המקורית לאחר שהוחלפו בה חלק מהאיברים. לפונקציה שני פרמטרי חובה. פרמטר ראשון מגדיר את האיבר שיש להחליף ואילו הפרמטר השני מגדיר את הערך שיוכנס במקומו. כל המופעים של האיבר שהוגדר בפרמטר הראשון יוחלפו ע"י האיבר שהוגדר בפרמטר השני, אלא אם כן משתמשים בפרמטר שלישי שהינו רשות. פרמטר זה קובע את מספר האיברים שיש להחליף.

```
>>> a = "aabbccddeeffaaccff"
>>> a.replace ( "a", "q" )
'qqbbccddeeffqqccff'
>>> a.replace ( "a", "q", 2 )
'qqbbccddeeffaaccff'
```

18. `rfind` – פונקציה דומה מאוד ל-`find`, אלא שכאן החיפוש מתחיל בצד ימין והפונקציה מחזירה את המופע האחרון של תת המחרוזת. הפונקציה מחזירה את המיקום האחרון שבו המחרוזת המועברת כפרמטר נמצאת בתוך המחרוזת הנבדקת. ניתן להגדיר את טווח החיפוש ע"י שימוש בשני פרמטרי רשות אשר מגדירים את גבולות המחרוזת הנבדקת (ברירות המחדל הן מתחילת המחרוזת ועד סופה). המספר המוחזר הוא המיקום במחרוזת המקורית, גם אם החיפוש הוגבל רק לחלק מהמחרוזת. אם המחרוזת לא נמצאת במחרוזת המקורית, מוחזר הערך -1.

```
>>> a = "happy birthday"
>>> a.find ( "y" )
4
>>> a.rfind ( "y" )
13
>>> a.rfind ( "c" )
-1
```

19. `rindex` – פונקציה זהה ל-`find` בהבדל אחד. אם `index` לא מוצאת את המחרוזת במחרוזת המקורית, היא מחזירה `ValueError` (במקום ה-1 אשר `rfind` מחזירה).

```
>>> a = "happy birthday"
>>> a.find ( "a" )
1
>>> a.rindex ( "a" )
12
>>> a.rfind ( "a" )
12
>>> a.rindex ( "q" )
```

```
Traceback (most recent call last):
  File "<pyshell#160>", line 1, in <module>
    a.rindex("q")
ValueError: substring not found
```

20. `rjust` – פונקציה זהה ל-`ljust` בהבדל אחד. `ljust` מצמידה את המחרוזת לצד שמאל ואילו `rjust` מצמידה אותה לצד ימין. פרמטר ראשון הינו פרמטר חובה והוא מגדיר את גודל המחרוזת החדשה (אשר בתוכה תמצא המחרוזת המקורית). פרמטר שני שהינו רשות מגדיר את תו המילוי. ברירת המחדל היא רווחים. אם הפרמטר הראשון קטן מגודל המחרוזת, פיתון עדיין תחזיר את המחרוזת בשלמותה.

```
>>> a = "happy birthday"
>>> a.rjust ( 26 )
'          happy birthday'
>>> a.ljust ( 26 )
'happy birthday          '
>>> a.rjust ( 26, "-" )
'-----happy birthday'
>>> a.ljust ( 26, "-" )
'happy birthday-----'
>>> a.rjust ( 10 )
'happy birthday'
```

21. `split` – פונקציה אשר מחזירה רשימה של מחרוזות. הרשימה נוצרה מפיצול המחרוזת המקורית. לפונקציה שני פרמטרי רשות. פרמטר ראשון מאפשר להגדיר את תו הפיצול (ברירת מחדל היא רווח) ופרמטר שני קובע את מספר הפיצולים. אם תו הפיצול לא נמצא במחרוזת, מוחזר עותק שלה (לא מפוצל). כדי לפצל תוך שימוש בברירת המחדל (רווח), אפשר גם לרשום בפרמטר הראשון `None`.

```
'happy birthday'
>>> a.split ( )
['happy', 'birthday']
>>> a.split ( "a" )
['h', 'ppy birthd', 'y']
>>> a.split ( "a", 2 )
['h', 'ppy birthd', 'y']
>>> a.split ( "a", 1 )
['h', 'ppy birthday']
>>> a.split ( "q" )
['happy birthday']
```

22. `rsplit` – פונקציה אשר מחזירה רשימה של מחרוזות בדומה לפונקציה `split`, אלא שבניגוד ל `split`, `rsplit` מתחילה לפצל החל מצד ימין. לפונקציה שני פרמטרי רשות. פרמטר ראשון מאפשר להגדיר את תו הפיצול (ברירת מחדל היא רווח) ופרמטר שני קובע את מספר הפיצולים. אם תו הפיצול לא נמצא במחרוזת, מוחזר עותק שלה (לא מפוצל).

```
>>> a = "happy birthday lady"
>>> a.split ( "a" )
['h', 'ppy birthd', 'y l', 'dy']
>>> a.rsplit ( "a" )
['h', 'ppy birthd', 'y l', 'dy']
>>> a.split ( "a", 2 )
['h', 'ppy birthd', 'y lady']
>>> a.rsplit( "a", 2 )
['happy birthd', 'y l', 'dy']
```

23. `rstrip` – פונקציה אשר מחזירה מחרוזת שהיא העתק של המחרוזת המקורית לאחר שקוצץ ממנה חלק בצידה הימני (פונקציה הפוכה לפונקציה `lstrip` שתוארה לעיל). הפונקציה מקבלת פרמטר רשות אחד המגדיר את התווים שיש להשמיט מהמחרוזת המקורית. ברירת המחדל היא השמטת הרווחים שנמצאים בצד ימין.

```
>>> a = "the cat ran up the tree"
>>> a.rstrip ( "the" )
'the cat ran up the tr'
>>> a.rstrip ( "eht" )
'the cat ran up the tr'
>>> a.rstrip ( "e" )
'the cat ran up the tr'
>>> a.rstrip ( "c" )
'the cat ran up the tree'
>>> a.rstrip ( "erth" )
'the cat ran up the '
>>> a.rstrip ( " erth" )
'the cat ran up'
```

24. strip – פונקציה אשר מחזירה מחרוזת שהיא העתק של המחרוזת המקורית לאחר שקוצצו ממנה תווים משני צדדיה. הפונקציה מקבלת פרמטר רשות אחד המגדיר את התווים שיש להשמיט מהמחרוזת המקורית. ברירת המחדל היא השמטת הרווחים שנמצאים משני הצדדים.

```
>>> a="the cat ran up the tree"
>>> a.strip ( "tre" )
'he cat ran up the '
>>> a.strip ( "t" )
'he cat ran up the tree'
>>> b = "   hi all!   "
>>> b.strip ( )
'hi all!'
```

25. swapcase – פונקציה ללא פרמטרים אשר מחזירה מחרוזת שהיא העתק של המחרוזת המקורית לאחר שהאותיות הקטנות הפכו לגדולות והאותיות הגדולות הפכו לקטנות.

```
a = "The Cat Ran Up The Tree"
>>> a.swapcase ( )
'tHE cAT rAN uP tHE tREE'
```

26. title – פונקציה ללא פרמטרים אשר מחזירה מחרוזת שהיא העתק של המחרוזת המקורית לאחר שכל אות בתחילת מלה הפכה לאות גדולה. שאר האותיות בכל מלה (אם אינן קטנות) יהפכו לקטנות.

```
a = "the cat RAN uP The tree"
>>> a.title ( )
'The Cat Ran Up The Tree'
```

27. upper – פונקציה ללא פרמטרים אשר מחזירה מחרוזת שהיא העתק של המחרוזת המקורית לאחר שכל האותיות הופכות לאותיות גדולות (הפוך מ – lower).

```
e = "happy birthday"
>>> e.upper ( )
'HAPPY BIRTHDAY'
>>> ( e.upper ( ) ).lower ( )
'happy birthday'
```

28. zfill – פונקציה בעלת פרמטר אחד, אשר מחזירה מחרוזת מספרית שהיא העתק של המחרוזת המקורית לאחר שנוספו אפסים מובילים בתחילתה. הפרמטר מגדיר את גודלה החדש של המחרוזת (כולל האפסים המובילים). אם המחרוזת המקורית אינה מספרית, עדיין יוכנסו אפסים מובילים מצידה השמאלי. אם הגודל החדש קטן מגודלה של המחרוזת המקורית, הוא תועתק כמות שהיא.

```
e = "happy birthday"
>>> e.zfill ( 15 )
'0happy birthday'
>>> e.zfill ( 20 )
'000000happy birthday'
>>> e.zfill( 10 )
'happy birthday'
```


תרגילי חזרה

רשום את תוצאת הריצה. (בדוק את תשובתך). התרגילים קשורים האחד בשני.

1. `a = "by the rivers of babylon, there we sat down"`
`a.capitalize ()`
2. `b = a.swapcase ()`
`b`
3. `b.title ()`
4. `c = b.title ()`
`c.swapcase ()`
5. `a = "by the rivers of babylon, there we sat down"`
`a.split ("e")`
6. `b = 'BY THE RIVERS OF BABYLON, THERE WE SAT DOWN'`
`b.split ("e")`
7. `c = 'By The Rivers Of Babylon, There We Sat Down'`
`c.split (", ") [0]`
8. `c = 'By The Rivers Of Babylon, There We Sat Down'`
`(c.split (", ") [0]).split ()`
9. `c = 'By The Rivers Of Babylon, There We Sat Down'`
`(c.split (", ") [0]).split () [2]`
10. `a = "by the rivers of babylon, there we sat down"`
`a.split (None, 2)`
11. `a = "Little Drummer Boy"`
`b = a.split ()`
`b [0].upper ()`
12. `a = "Little Drummer Boy"`
`b = a.split ()`
`c = b [1]`
`d = c.lower ()`
`d.replace ("r", "--")`
13. `a = "Do you want to know a secret"`
`(a.split ("a") [0]).split () [1]`

14. a = "Do you want to know a secret"
 b = a.rsplit ("a", 1)
 c = b [1]
 d = c.center (12)
 d.replace ("e", "*")
15. a = "follow the sun"
 b = " "
 b.join (a)
16. a = "let it be"
 b = a.title ()
 b.ljust (13, "!")
17. a = "fool on the hill"
 b = a.capitalize ()
 c = b.replace ("o", "\$")
 d = c.replace ("l", "m")
 d.isalnum ()
 d.isalpha ()
18. a = "Lucy in the sky with diamonds"
 a.find ("in")
 a.find ("i")
 a.rfind ("i")
 a.find ("k")
 a.rfind ("k")
19. a = "Take me back to my boat on the river"
 if a.istitle () == 0 :
 b = a.title ()
 else:
 b = a [:]
 print (a)
 print (b.center (44, "*"))

[תשובות](#)

[חזור לתוכן](#)

עיצוב מחרוזות

במהדורות הקודמות של פיתון, עיצוב מחרוזות המודפסות נעשה ע"י הוספת הסימן % במחרוזת (במקום או מקומות בהם נדרש העיצוב). פירוש הסימן הוא שבמקום בו הוא מופיע יש להכניס ערך אחר שמופיע בהמשך השורה. למשל:

```
>>> a = 35
>>> print ( "count is: %d:" %a )
count is: 35:
```

בדוגמה הגדרנו משתנה a והכנסנו לתוכו את הערך 35. לאחר מכן השתמשנו בפקודת print הכוללת מחרוזת, אלא שבמחרוזת שילבנו את הסימן %. סימן זה מציין שברצוננו להחליפו לפני ההדפסה. מה שיחליף אותו זה ערך (או משתנה המייצג ערך) שנמצא בהמשך השורה (לאחר סוף המחרוזת) וגם הוא מוגדר ע"י הסימן %. לכן בזמן עיצוב ההדפסה, מודפסים התווים שבמחרוזת עד אשר מגיעים לסימן %. אז מדלגים לסוף המחרוזת ומחפשים את הערך הראשון שמופיע אחר הסימן % ומשלבים אותו בהדפסה (במקום ה - %). ממשיכים להדפיס את תווי המחרוזת, עד אשר היא מסתיימת או מוצאים סימן % נוסף (אשר גם אותו צריך להחליף). הערכים שיש לשלב במחרוזת, כאמור, נמצאים בסופה, מופרדים ע"י הסימן %. אם יש יותר מערך אחד, הם מוצגים כרשימה של ערכים, כפי שניתן לראות בדוגמה:

```
>>> a = 12
>>> b = 19
>>> c = 11
>>> d = -24
>>> print ( "1st= %d, 2nd= %d, 3rd= %d, 4th= %d" % ( a, b, c, d ) )
1st= 12, 2nd= 19, 3rd= 11, 4th= -24
```

בדוגמה הודפסה מחרוזת בעלת ארבעה סימני %, אשר הוחלפו בהתאמה בערכים הנמצאים בתוך המשתנים a, b, c, d.

עיצוב דומה לזה שתואר לעיל אפשר להשיג גם בפקודת הדפסה פשוטה, למשל:

```
>>> print ("1st=",a, ", 2nd=",b, ", 3rd=",c, ", 4th=",d)
1st= 12 , 2nd= 19 , 3rd= 11 , 4th= -24
```

ואכן, המטרה האמיתית של העיצוב היא לא רק להחליף ערכים בהדפסה, אלא גם לעצב אותם. בתוך המחרוזת הופיע הסימן % ואחריו האות d שעד עתה לא התייחסנו אליה. אות זו (ונוספות) מגדירה את פורמט ההדפסה. d מציינת מספר שלם ולכן השורה שהודפסה כללה מספרים שלמים. אם ברצוננו להדפיס את המספרים כשברים, יש להחליף את האות d באות f (במקום digit, יש להשתמש ב - float). כפי שניתן לראות בדוגמה:

```
>>> print ( "1st= %f, 2nd= %f, 3rd= %f, 4th= %f" % ( a, b, c, d ) )
1st= 12.000000, 2nd= 19.000000, 3rd= 11.000000, 4th= -24.000000
```

חשוב לציין שאם את דוגמת ההדפסה הקודמת אפשר היה להשיג גם בעזרת פקודת הדפסה פשוטה, אזי לצורך הדוגמה האחרונה, היינו אמורים להחליף את סוג המשתנה. אם a, b, c, d הם משנים שלמים, אזי היינו צריכים קודם להמיר אותם לשברים ורק לאחר מכן היינו יכולים להדפיס אותם בפורמט הרצוי.

אולם בדוגמה שראינו, מספר המקומות המוצגים מימין לנקודה העשרונית הוא גדול מדי. פקודות העיצוב מאפשרות לנו לקבוע את מספר הספרות. למעשה, הן גם מאפשרות לקבוע את סה"כ המיקומים שנדרשים כולל גם המיקומים שמימין לנקודה העשרונית. הפורמט המלא הוא:

תכנות בסיסי

% <Width>. <Precision><Type>

כאשר Width מציין את סה"כ התווים הנדרשים (0 או ריק מאפשר לפיתון לקבוע את הגודל הדרוש)
Precision מציין את מספר המיקומים לימין הנקודה העشرונית
Type מציין את סוג המידע (מספרי שלם, שבר, מחרוזת כד').

לכן, אם נשנה את הפקודה הקודמת (להגבלת מספר המיקומים של ימין הנקודה העשרונית) נקבל:

```
print ("1st= %.2f, 2nd= %.1f, 3rd= %.1f, 4th= %.0f" % ( a, b, c, d ))  
1st= 12.00, 2nd= 19.0, 3rd= 11.0, 4th= -24
```

עבור המשתנה הראשון הגדרנו שני מיקומים, לשני ולשלישי רק מיקום אחד ולרביעי אפס ואכן ההדפסה התקבלה בהתאם לבקשה. ניתן כמובן גם לקבוע את גודל כל משנה. למשל:

```
print ( "1st= %7.2f, 2nd= %3.1f, 3rd= %4.1f, 4th= %6.0f" % ( a, b, c, d ))  
1st= 12.00, 2nd= 19.0, 3rd= 11.0, 4th= -24
```

אם הנושא חשוב, אפשר גם להוסיף סימן לסכום. אמנם, כאשר הסכום שלילי, הסימן קיים, אך לפעמים לצורך הדגשה, נדרש גם סימן עבור המספרים החיוביים. כמו בדוגמה:

```
>>> print ( "1st= %+7.2f, 2nd= %3.1f, 3rd= %4.1f, 4th= %6.0f" % ( a, b, c, d ))  
1st= +12.00, 2nd= 19.0, 3rd= 11.0, 4th= -24
```

להדפסת המספר השני נוסף סימן חיובי.

סוגי המידע הנפוצים הם: מספרים שלמים: d,i, שברים: f, מספרים אוקטליים: o, מספרים הקסהדצימאליים: x, מספרים מדעיים: e ומחרוזת: s.

בגלל שהמרה והעיצוב מבוצעים באופן אוטומטי אפשר להשתמש בזה להמרות של מספרים: למשל:

```
>>> a = 155  
>>> b = 299  
>>> print ( " Decimal %d = Octal %o" % ( a, a ) )  
Decimal 155 = Octal 233  
>>> print ( " Decimal %d = Hexadecimal %x" % ( b, b ) )  
Decimal 299 = Hexadecimal 12b  
print ( " Decimal %d = text %s" % ( b, b ) )  
Decimal 299 = text 299  
>>> print ( " Decimal %d = Exponent %e" % ( b, b ) )  
Decimal 299 = Exponent 2.990000e+002  
>>> print ( " Decimal %d = Exponent %5.2e" % ( b, b ) )  
Decimal 299 = Exponent 2.99e+002
```

כאמור שיטות עיצוב אלה קיימות כיום כשריד של המהדורות הקודמות (Python 2), אולם החל ממהדורה 3 של פיתון הוכנסו פקודות חדשות של עריכה ויש להעדיף על פני הקודמות.

פקודת העריכה החדשה שהוגדרה מאפשרת גם החלפה של הפרמטרים המודפסים. למשל ע"י:

```
print ("Shalom {0} and welcome to the {1} class".format ("Avi", "CS"))
```

זו פקודת הדפסה רגילה, אלא שכחלק ממלל ההדפסה נוספו, במקרה זה, שני פרמטרים אשר נקראים שדות הפורמט והם נמצאים בסוגריים מסולסלים. המשמעות של שדות הפורמט היא שהם צריכים להיות מוחלפים בערכים אשר מופיעים בהמשך הפקודה (לאחר המלה פורמט ובסוגריים). הלוגיקה היא שהשדה הראשון, אשר מספרו אפס, מוחלף בערך הראשון, השדה השני, אשר מספרו אחד, מוחלף בערך השני וכן הלאה. לכן, כתוצאה של ביצוע הפקודה תתקבל ההדפסה של:

Shalom Avi and welcome to the CS class

ניתן כמובן להכניס את שדות הפורמט שלא לפי הסדר, משום שמה שקובע את הערכים הוא המספר. לכן בדוגמה:

```
print ("Shalom {1} and welcome to the {0} class".format ("MIS", "Gal"))
```

התוצאה תהיה:

Shalom Gal and welcome to the MIS class

אפשר להגדיר שמות לשדות ואז המיקום אינו משנה. למשל בדוגמה:

```
print("Shalom {name}, welcome to the {clas} class!".format (name = "Paz", clas = "MIS"
) )
```

במקרה זה, לא עושים שימוש במספר של הפרמטר, אלא ממש קובעים לו שם. למשל name מייצג את השדה של שם הסטודנט שברצוננו לברך. לעומת זאת, clas הוא השם של הקורס. השמות חייבים, כמובן להופיע בהמשך הפקודה, כולל הגדרת ערכם. לכן, בגלל שערכו של הפרמטר name הוא Paz, הערך יוכנס כחלק מההדפסה. באופן דומה יוכנס גם שם הקורס. ההדפסה שנוצרת מביצוע פקודה זו היא:

Shalom Paz, welcome to the MIS class!

אין מניעה מעירוב שתי השיטות.

בהמשך להגדרת ערכי הפרמטרים (שדות הפורמט), אפשר גם לעצבם. בצורה הפשוטה ביותר, אפשר להוסיף לשדה הפורמט נקודתיים ואחריהן מספר המייצג את גודל השדה בהדפסה. למשל הפקודה:

```
print ("Shalom {0:8}, welcome to the {1:5} class".format ( "Tal", "MIS" ) )
```

לאחר שבוצעה גורמת להדפסה:

Shalom Tal , welcome to the MIS class

המשמעות היא של החלפת הערכים, כפי שהיה במקרה הקודם, אלא שעתה התווסף גם העיצוב. עבור הערך הראשון, רשמנו שבהדפסה ברצוננו שהשדה יהיה בגודל של שמונה תווים ולכן למרות שהשם כולל שלושה תווים בלבד, הוא נרשם בשמונה תווים, כאשר שאר החמישה הם רווחים. באופן דומה רשמנו שהשדה השני צריך לכלול חמישה תווים ולכן בנוסף לשלושה התווים שבשם, נוספו שני תווי רווח.

תווי העיצוב והעריכה עובדים כמובן גם על סוגי נתונים אחרים ולא רק על מחרוזות. למשל אם ברצוננו להדפיס ערכים מספריים:

תכנות בסיסי

```
print ("One item costs {0}, three items cost {1} ".format ( a, 2*a ) )
```

בהנחה שערכו של a הוא 55, ההדפסה המתקבלת היא:

One item costs 55, three items cost 110

גם במקרה זה אפשר להוסיף פקודת עריכה, כמו למשל הגדרה של גודל השדה בהדפסה, כפי שניתן לראות בדוגמה:

```
print ("One item costs {0:5}, three items cost {1:5} ".format ( a, 2*a ) )
```

במקרה זה, גודל השדות הוא 5 תווים ואם הפרמטר האמיתי הוא של שניים או שלושה תווים, כפי שמופיע בדוגמה, יתווספו תווי רווח.

One item costs 55, three items cost 110

בהמשכו של המספר המגדיר את גודל השדה, אפשר להוסיף תו נוסף המציין את הבסיס, כאשר ברירת המחדל היא בסיס עשרוני (התו d או n), בסיס בינארי (b), אוקטלי (o) והקסהדצימלי (X). כפי שניתן לראות בדוגמאות הרצ"ב:

```
>>> print ("One item costs {0:5o}, three items cost {1:5o} ".format ( a, 2*a ) )
One item costs 67, three items cost 156
>>> print ("One item costs {0:5d}, three items cost {1:5d} ".format ( a, 2*a ) )
One item costs 55, three items cost 110
>>> print ("One item costs {0:5n}, three items cost {1:5n} ".format ( a, 2*a ) )
One item costs 55, three items cost 110
>>> print ("One item costs {0:5X}, three items cost {1:5X} ".format ( a, 2*a ) )
One item costs 37, three items cost 6E
>>> print ("One item costs {0:5b}, three items cost {1:5b} ".format ( a, 2*a ) )
One item costs 110111, three items cost 1101110
```

נקודה חשובה שיש לציין מתייחסת לגודל המינימאלי הדרוש, כפי שניתן לראות בדוגמה האחרונה. למרות שביקשנו לערוך את השדות בגודל של חמישה תווים בלבד, בגלל העובדה שהמספרים גדולים מחמישה תווים, גודל נקבע ע"י הערך בפועל ולא פקודת הפורמט. הדבר נעשה כדי למנוע מצב שבו המספר נחתך והערך המוצג אינו הערך האמיתי. במצב זה, פיתון אינו מבצע את מה שביקשנו, אך הדבר נובע מהעובדה שיש כאן שתי בקשות מנוגדות. הצגת הערך האמיתי תוך התעלמות מהעריכה שהתבקשה מזיקה פחות מהצגת ערך שגוי תוך הקפדה על הפורמט הדרוש.

כחלק מהתמיכה בעריכת מספרים, קיימת גם אפשרות לעצב את השברים. הדבר נעשה ע"י הוספה של נקודה בפורמט ואחריה מספר המייצג את משמעותיות השבר ואחריו התו f המציין נקודה צפה (הדרך לייצג מספרים ממשיים במחשב). למשל כפי שמופיע בדוגמה:

```
>>>import math
>>>print("the value of PI is {0:.7f}".format ( math.pi ) )
the value of PI is 3.1415927
>>> print("the value of PI is {0:.15f}".format ( math.pi ) )
the value of PI is 3.141592653589793
```

הפקודה הראשונה גורמת לספריה המתמטית, כול הפונקציות והקבועים שבה, להיות זמינה. מתוך הספרייה אנו רוצים להשתמש בקבוע pi. בפקודה הראשונה אנו משתמשים בו, אך מבקשים לערוך אותו כך שיודפסו רק שבע ספרות מימין לנקודה העשרונית ואילו בדוגמה השנייה, אנו משתמשים בו פעם נוספת, אלא שהפעם בחרנו להשתמש בחמש עשרה ספרות.

אפשר כמובן לשלב בין שיטות המרה ועיצוב שונות:

```
>>> print ("One item costs {0:5.3f}, three items cost {1:9b} ".format ( a, 2*a ) )
```

תכנות בסיסי

One item costs 55.000, three items cost 1101110

להרחבות נוספות לגבי יכולות העריכה והעיצוב ראה באתר.

רשום את תוצאת הריצה. (בדוק את תשובתך).

1. `print ("text = %s, value = %i" % ("123", eval ("123")))`
2. `print ("text = %s, value = %i" % ("123", int ("123")))`
3. `print ("text = %s, value = %f" % ("123", int ("123")))`
4. `print ("text = %s, value = %4.2f" % ("123", int ("123")))`
5. `print ("text = %s, value = %4.2e" % ("123", int ("123")))`
6. `print ("text = %s, value = %4.2e" % ("123", eval ("123")))`
7. `print ("value = %d, text = %s" % (7777, 7777))`
8. `print ("value = %d, text = %s" % (0x77, 0x77))`
9. `print ("value = %7.3f, text = %s" % (7777, 7777))`
10. `print ("value = %7.3f, text = %3.3e" % (7777, 7777))`
11. `print ("value = %7.3f, exponent = %.7e" % (7777, 7777))`
12. `print ("value = %7.3f, exponent = %.3e" % (77*77, 77*77))`
13. `print ("value = %7.3f, exponent = %.3e" % (0x7*77, 0x7*77))`
14. `print ("value = %7.3f, exponent = %.3e" % (0xf1, 0xf1))`
15. `print ("value = %o, integer = %i" % (0xf1, 0xf1))`
16. `print ("%o, %x, %i, %s, %e" % (199, 199, 199, 199, 199))`
17. `print ("%7o,%5x,%7i,%7s,%20e" % (199, 199, 199, 199, 199))`
18. `print ("text = %s, value = %i" % ("123"+"456", int("123"+"456")))`
19. `print ("text = %s, value = %i" % ("12"*3, int("12"*3)))`
20. `print ("text = %s, value = %i" % ("02"*3, eval("02"*3)))`
21. `def targil():`
`i = 1`
`print ("%-4s%-5s%-6s%-8s%-13s%-15s" \`
`%('i', 'i**2', 'i**3', 'i**5', 'i**10', 'i**20'))`
`while i <= 10:`
`print ("%-4d%-5d%-6d%-8d%-13d%-15d" \`
`%(i, i**2, i**3, i**5, i**10, i**20))`
`i += 1`

[תשובות](#)

[חזור לתוכן](#)

פקודות break, continue ו- else

פקודת ה- break נועדה לצאת מתוך הלולאה (או הלולאה הפנימית במקרה של לולאות מקוננות). ולכן היא יכולה להופיע רק בתוך לולאה. לדוגמה:

```
a1 = [ 0 , 3 , 6 , 12 , 9 , 18 , 15 , 10 ]
for j in range ( len ( a1 ) ) :
    a = a1 [ j ]
    if a == 9 :
        break
    else:
        print ( a )
```

תוצאת הריצה:

```
0
3
6
12
```

פקודת ה- break הופעלה לאחר שהאיבר המתאים היה 9 ואז היא גרמה להפסקת ביצוע הלולאה ולכן כל שאר האיברים שרשימה a1 לא הודפסו.

פקודת ה- continue נועדה להפסיק את ביצוע המחזור הנוכחי של הלולאה ודילוג למחזור הבא. הפקודה משמשת עבור מקרים בהם במהלך הלולאה, עבור מחזור מסוים לא צריך לבצע כלום. לדוגמה:

```
a1 = [ 0 , 3 , 6 , 12 , 9 , 18 , 15 , 10 ]
for j in range ( len ( a1 ) ) :
    a = a1 [ j ]
    if a == 9 :
        continue
    else:
        print ( a )
```

תוצאת הריצה:

```
0
3
6
12
18
15
10
```

תכנות בסיסי

לעיתים אפשר להוסיף לפקודות לולאה גם פקודת `else` שמתבצעת כאשר הלולאה הסתיימה (במצב רגיל), אך לא מתבצעת כאשר הלולאה הסתיימה עקב פקודת `break`. הדבר מאפשר להוסיף התייחסות מיוחדת למצב שהלולאה הסתיימה ולמשל תנאי מסוים לא קרה. לדוגמה, התוכנית הבודקת מספרים אם הם מתחלקים במספר כלשהו (מספיק אחד), או שהם ראשוניים:

```
for n in range (2, 10) :  
    for x in range (2, n) :  
        if n % x == 0:  
            print ( n, 'equals', x, '*', n/x )  
            break  
    else:  
        print ( n, 'is a prime number' )
```

דוגמת ההרצה:

```
2 is a prime number  
3 is a prime number  
4 equals 2 * 2  
5 is a prime number  
6 equals 2 * 3  
7 is a prime number  
8 equals 2 * 4  
9 equals 3 * 3
```

רשום את תוצאת הריצה. (בדוק את תשובתך).

1.

```
for n in range (2, 10) :
    for x in range ( 2, n ) :
        if n % x == 0 :
            print ( n, 'equals', x, '*', n/x )
            break
        else:
            print ( n, 'is a prime number' )
```
2.

```
for k in range (5, 10) :
    if k == 7 :
        print ( 'breaking out of the loop!' )
        break
    else:
        print ( k, end = " " )
```
3.

```
for k in range ( 5,10 ) :
    if k == 7 :
        print ( 'breaking out of the loop!' )
        break
    else:
        print ( k, end = " " )
```
4.

```
i=0
while i < 15:
    i = i+1
    print ( i )
    if i == 4:
        print ( 'breaking out of loop' )
        break
    else:
        print ( i )
```
5.

```
i=0
while i < 15:
    i = i+1
    print ( i )
    if i == 4 :
        print ( 'breaking out of loop' )
        break
    else:
        print ( i )
```

6.

```
a = [ 1, 2, 3, 4, 5, 6, 9, 10]
sum = 0.
for x in range ( len ( a ) ) :
    sum = sum + a [ x ]
avg = sum / len ( a )
for t in range ( len ( a ) ) :
    if avg == a [ t ] :
        print ( t )
        break
else:
    print ( "None found" )
```
7.

```
items = [ 11, 202, 37, 411, 275, 346, 509, 310, 922, 786 ]
test = [ 9, 444, 275, 787, 509, 11, 37, 666 ]
for key in test:
    for item in items:
        if item == key:
            print ( key, "found!" )
            break
    else:
        print key, ( "not found!" )
```

[תשובות](#)

[חזור לתוכן](#)

עבודה עם קבצים

לפעמים יש צורך לעבוד עם קבצים, למשל בקריאת נתונים רבים, או שמירה של תוצאות ריצה, כדי שתהיינה זמינות לריצות אחרות/נוספות. העבודה מול קבצים נעשית תוך שימוש באובייקט (נושא שיורחב בקורסי המשך), אשר מספק את כל הפונקציות הנדרשות. משמעות הדבר היא שיש לשייך את הקובץ מולו ברצוננו לעבוד למשתנה וכל הפעילויות מול הקובץ נעשות מול המשתנה שהוגדר.

כדי שנוכל לעבוד עם קובץ כלשהו, יש ראשית לפתוח אותו ע"י שימוש בפקודה open. למשל:

```
>>> my_file = open("data.txt", "w")
```

לפקודת ה- open שני פרמטרים: מחרוזת הכוללת את שם הקובץ (כפי שהוא מופיע במחשב) ופרמטר שני מחרוזת המגדירה את מהות הפעולה מול הקובץ (mode). למשל אם ברצוננו לפתוח קובץ לקריאה בלבד נשתמש בערך "r" לעומת זאת אם נרצה לכתוב לקובץ נבחר את הערך "w". כאשר פותחים קובץ לכתבייה והקובץ לא נמצא, פיתון יגדיר קובץ חדש.

באופן כללי, פיתון תמיד יחפש את הקובץ בתיקייה בה אנו נמצאים. זו גם התיקייה שבה התוכנית שאנחנו מבצעים, נשמרת. אם רוצים לחפש את הקובץ בתיקייה אחרת, אפשר לציין את הנתיב (path) של הקובץ, למשל

```
>>> my_file = open("c:/temp/data.txt", "w")
```

ברירת המחדל לפרמטר השני היא "r". בפתיחת קובץ לכתבייה, פיתון ירמוס את תוכנו (יתחיל לכתוב עליו מתחילתו). אם ברצוננו לפתוח קובץ לכתבייה אבל לכתוב רק בסופו (להוסיף לו מידע נוסף על זה הקיים כבר בקובץ), יש להשתמש בפרמטר "a" לדוגמה:

```
>>> my_file = open("data.txt", "a")
```

במקרה של פתיחת קובץ גם לקריאה וגם לכתבייה, אזי הפרמטר שיש להשתמש הוא "r+", למשל

```
>>> my_file = open("data.txt", "r+")
```

בקורס זה נעשה שימוש רק בקבצי טקסט, אולם פיתון יודע גם להשתמש בקבצים בינאריים, כמו קבצי תמונות (JPEG, TIFF, BMP) וכד'. במקרה כזה צריך להוסיף את ה- b אות b לפרמטר השני למשל

```
>>> my_file = open("data.txt", "r+b")
```

הפקודה תפתח קובץ בינארי לקריאה וכתבייה. שימוש ב- "wb" לעומת זאת יפתח קובץ בינארי לכתבייה בלבד.

הערה:

בגלל בעיות תאימות בין מערכות הפעלה שונות, תכניות שנכתבו בפיתון ואשר משתמשות בקבצים בינאריים, לא הכרח עובדות בצורה זהה על Windows ומערכות unix.

תכנות בסיסי

לאחר שנוצר הקישור בין הקובץ הפיזי (שנמצא על הדיסק) ובין המשתנה אשר מתאר אותו בפיתון, אפשר לקרוא ולכתוב לקובץ. לצורך הדוגמאות שמופיעות בהמשך פרק זה, נניח שהמשתנה שבחרנו הוא my_file (כמו בדוגמאות לעיל). בנוסף, לצורך ההסברים, נניח שיצרנו קובץ טקסט בשם example.txt אשר מכיל את התוכן:

```
This is an example
Record number 2
22/12/2012
123456789
57.89 37.5 12.5
End of example
```

לכן, לפני שנוכל לקרוא מהקובץ, יש לפתוח אותו:

```
>>> my_file = open("example.txt")
```

כזכור, ברירת המחדל של הפתיחה היא לקריאה בלבד ולכן ניתן היה להשמיט את הפרמטר השני בפקודת הקריאה.

קיימות שלוש שיטות (פקודות) לקריאה מהקובץ:

1. read – אשר קוראת מהקובץ ומחזירה מחרוזת. ניתן להשתמש בפרמטר אשר מגדיר את מספר התווים שברצוננו לקרוא מהקובץ. ברירת המחדל, או אם הפרמטר שהוכנס הוא שלילי, היא קריאה של כל תוכן הקובץ. אם הקובץ ריק, תוחזר מחרוזת ריקה (""). דוגמאות:

```
<<< a = my_file.read()
```

בעקבות הקריאה, המשתנה a יכיל מחרוזת ובה כל תוכן הקובץ. אם נדפיס את המחרוזת a ע"י שימוש בפקודת הדפסה, אשר גם עורכת את התוכן, נקבל שוב את השורות שהוכנסו לקובץ. אולם אם נרצה לראות את תוכנו של a, נראה גם את תווי הבקרה (כמו סוף שורה)

```
<<< a
'This is an example\nRecord number 2\n22/12/2012\n123456789\n12.5
37.5 57.89\nEnd of example\n'
```

ניתן לראות את תוכן הקובץ, כאשר בסוף כל שורה מופיע "\n" שמציין סוף שורה. אם צריך לעבד את תוכן הקובץ, אפשר להשתמש בפקודות הרגילות לחילוץ ועיבוד מחרוזות.

אפשר גם לקרוא חלק מהתווים בקובץ, ע"י הוספת פרמטר לפקודת הקריאה, למשל:

```
a = my_file.read(50)
<<<a
'This is an example\nRecord number 2\n22/12/2012\n1234'
```

במקרה זה נקראו 50 התווים הראשונים. אם נרצה לקרוא את המשך הקובץ נצטרך לבצע פקודת קריאה נוספת כמו:

```
a = my_file.read()
>>> a
'56789\n12.5 37.5 57.89\nEnd of example\n'
```

חשוב לציין שפיתון זוכר את המיקום בקובץ ולכן בפקודות הבאות הוא ממשיך לקרוא ממיקום זה ולא מההתחלה. לכן אם רוצים לחזור לקרוא שוב מההתחלה, יש צורך לשנות את המיקום והדבר מתבצע ע"י פקודות seek. אם ברצוננו לחזור אל ההתחלה הפקודה שיש לבצע היא

```
<<< my_file.seek(0)
0
```

באופן דומה אם ברצוננו לקרוא ממקום אחר בקובץ, ניתן להשתמש בפקודת ה – seek כדי לשנות את המיקום לתחילת הקריאה (או הכתיבה) הבאה.

2. `readline` – אשר קוראת שורה אחת מהקובץ ומחזירה מחרוזת. המחרוזת כוללת גם את תו סיום השורה ("`\n`"). אם בקובץ נמצאת שורה ריקה, `readline` יחזיר מחרוזת המכילה "`\n`" ואילו בסוף הקובץ הוא יחזיר מחרוזת ריקה. דוגמאות לשימוש ב – `readline`:

```
<<< a = my_file.readline()
>>> a
'This is an example\n'
```

כמו במקרה קודם, פיתון שומר את המיקום בקובץ ולכן בביצוע נוסף של `readline` נקבל את תוכן השורה השנייה בקובץ וכן הלאה. למשל

```
a = my_file.readline()
>>> a
'Record number 2\n'
```

אם ברצוננו לקרוא את כל תוכנו של הקובץ נצטרך להשתמש בלולאה שבכל מחזור תקרא שורה. זו יכולה להיות למשל לולאת `while` אשר תתבצע כל זמן שהמחרוזת שהתקבלה בקריאת ה – `readline` אינה ריקה. דרך אחרת לקריאת השורות מהקובץ היא ע"י שימוש בלולאת `for`:

```
>>> for a in my_file:
    print (a)
```

This is an example

Record number 2

22/12/2012

123456789

12.5 37.5 57.89

End of example

3. `readlines` – אשר קוראת את כל הקובץ ומחזירה רשימה של מחרוזות. כל מחרוזת היא שורה בקובץ. הפקודה מקבלת פרמטר אופציונלי אשר מגדיר את מספר התווים שיש לקרוא מהקובץ. אולם בניגוד לפקודה `read`, אשר קוראת בדוק את מספר התווים שהוגדרו בפרמטר, `readlines` קוראת שורות שלמות ולכן היא תשלים את התווים מעבר למוגדר בפרמטר, עד להכללת השורה השלמה האחרונה. דוגמת שימוש ב – `readlines`:

```
a = my_file.readlines()
>>> a
['This is an example\n', 'Record number 2\n', '22/12/2012\n', '123456789\n',
'12.5 37.5 57.89\n', 'End of example\n']
```

שתי הדוגמאות הבאות מדגימות את ההבדל בין `read` ו- `readlines` בהתייחס למספר התווים שיש לקרוא מהקובץ.

```
a = my_file.read(25)
>>> a
'This is an example\nRecord'
>>> my_file.seek(0)
0
>>> a = my_file.readlines(25)
>>> a
['This is an example\n', 'Record number 2\n']
```

הסיבה לשימוש בפרמטר הקובע את מספר התווים היא כדי לא לקרוא את כל תוכנו של הקובץ לזיכרון, כדי לא להעמיס יתר על המידה.

כתיבה לקובץ משתמשת בפקודה `write`, אשר כותבת את תוכן המחרוזת אל הקובץ. למשל

```
my_file = open("example1.txt", "w")
>>> some_text = "this is another example\n to write to a file\n"
>>> my_file.write(some_text)
44
>>> my_file.write("and this is the last line")
25
```

הפעם פתחתנו קובץ לכתיבה ובגלל שהוא אל היה, פיתון יצר אותו. לאחר מכן הגדרנו מחרוזת (בתוך המשתנה `some_text`) ואז השתמשנו בפקודה `write`, כדי לכתוב את המחרוזת אל הקובץ. הפקודה `write` יכולה גם לכתוב לקובץ את הטקסט שנמצא בסוגריים הצמודים אליה, כפי שניתן לראות בדוגמה. בכל מקרה, לאחר שהטקסט נכתב, הפקודה `write` מחזירה את מספר התווים שנכתבו.

לאחר שסיימנו לעבוד עם הקובץ, יש לסגור אותו ע"י פקודת `close`.

```
my_file.close()
```

הפקודה מחזירה את הקובץ למערכת ההפעלה, כך שתהליכים אחרים, או תכניות אחרות יוכלו להשתמש בו. לאחר סגירת הקובץ, אין יותר אפשרות לקרוא ממנו או לכתוב אליו תוך שימוש בשם הקישור ויש לפתוח אותו שוב לפני שניתן יהיה לגשת אליו.

[חזור לתוכן](#)

הבנת תוכניות

פרק זה מהווה מעין סיכום והוא כולל דוגמאות של תוכניות שעליך להבין ולענות על השאלה.

1. התוכנית אמורה לקרוא מספר ולהדפיס "even" אם הוא זוגי ו- "odd" אם הוא אי-זוגי. לצורך התרגיל, הנח שהספרה 0 הינה זוגית.

```
n = int (input ( "number : " ) )
if n % 2 == 0 :
    print ( "even" )
elif n % 2 == 1 :
    print ( "odd" )
```

- א. התוכנית עובדת נכון
- ב. התוכנית עובדת נכון רק למספרים זוגיים
- ג. התוכנית עובדת נכון רק לספרים אי זוגיים
- ד. התוכנית לא עובדת נכון
- ה. התוכנית עובדת נכון רק עבור 0

2. מה הערך שיתקבל מהרצת קטע קוד זה?

```
a = 0
for n in range ( 0 , 5 ) :
    a = a + 1
print ( a )
```

3. מה הערך שיתקבל מהרצת קטע קוד זה?

```
a = 0
for n in range ( 0 , 0 ) :
    a = a + 1
print ( a )
```

4. מה הערך שיתקבל מהרצת קטע קוד זה?

```
a = 0
for n in range ( 0 , 5 ) :
    for m in range ( 0 , 3 ) :
        a = a + 1
print ( a )
```

תכנות בסיסי

5. מה הערך שיתקבל מהרצת קטע קוד זה?

```
a = 0
for n in range ( 0 , 5 ) :
    for m in range ( 1 , 0 ) :
        a = a + 1
print (a)
```

6. מה הערך שיתקבל מהרצת קטע קוד זה?

```
a = 0
for n in range ( 1 , 5 ) :
    for m in range ( n , 5 ) :
        a = a + 1
print ( a )
```

7. נתונה הפונקציה הרקורסיבית הבאה:

```
def fun ( n ) :
    if n == 4 :
        return 2
    else:
        return 2 * fun ( n + 1 )
```

אם קוראים לפונקציה עם הערך 2 התוצאה המתקבלת היא:

- א. 2
- ב. 4
- ג. 8
- ד. 16
- ה. 24

8. נתונה הפונקציה הרקורסיבית הבאה:

```
def fun ( x , y ) :
    if x == y :
        return y
    else:
        return y + fun ( x - 1 , y )
```

מה הערך המתקבל אם קוראים לפונקציה ע"י `fun (4 , 3)` ?

9. נתונה הפונקציה הרקורסיבית הבאה:

```
def fun ( x , y ) :
    if x == 0 :
        return y + 1
    elif y == 0 :
        return fun ( x - 1 , 1 )
    else:
        return fun( x - 1 , fun ( x , y - 1 ) )
```

מה הערך המתקבל אם קוראים לפונקציה ע"י $\text{fun}(1, 1)$?

10. נתון קטע הקוד:

```
for n in range ( 11 ) :
    print ( " % 6d , % 6x " % ( n , n ) )
```

ענה על השאלות הבאות: (כן/לא)

- א. התוכנית מדפיסה 10 שורות
- ב. השורה הראשונה המודפסת היא: 60,60
- ג. השורה האחרונה המודפסת היא: 10,a

11. רשום מה מדפיס קטע הקוד בסיום הריצה:

```
x = 0
for a in range ( 5, 0, -1 ) :
    for b in range (0, a ) :
        x = x + 1
print ( x )
```

12. נתונה הפונקציה הרקורסיבית הבאה:

```
def sum2n ( x ) :
    if x == 1 :
        return x
    else:
        return x + sum2n ( x - 2 )
```

מה הערך המתקבל אם קוראים לפונקציה ע"י $\text{sum2n}(7)$?

13. נתונה הפונקציה הרקורסיבית הבאה:

```
def fib ( x ) :
    if x == 1 :
        return 1
    elif x == 0 :
        return 0
    else:
        return fib ( x - 1 ) + fib ( x - 2 )
```

מה הערך המתקבל אם קוראים לפונקציה ע"י $\text{fib}(6)$?

14. נתונה הפונקציה הרקורסיבית הבאה:

```
def gcd ( a , b ) :
    if b == 0 :
        return a
    return gcd ( b , a % b )
```

מה הערך המתקבל אם קוראים לפונקציה ע"י $\text{gcd}(120, 144)$?

15. נתונה הפונקציה הרקורסיבית הבאה:

```
def fun ( a , b ) :
    if b % 2 == 0 :
        return a
    return fun ( a * 2 , b // 2 )
```

מה הערך המתקבל אם קוראים לפונקציה ע"י $\text{fun}(5, 79)$?

16. רשום מה מדפיס קטע הקוד:

```
x = 3
if 2 > x :
    print ( '1', end = " " )
else :
    print ( '2' , end = " " )
    if 2 > x :
        print ( '3', end = " " )
    print ( '4' , end = " " )
print ( '5' )
```

17. רשום מה מדפיס קטע הקוד:

```
result = 0.0
num1 , num2 , num3 = 5, 12, 2
val1 , val2, val3 = 5.0, 12.0, 2.0
result = num2 / num1
print ( result )
result = val3 + num2 // num1
print ( result )
result = num3 + val2 // num1
print ( result )
result = num2 // num3 // num1
print ( result )
result = val2 * num1 + num2 / num3
print ( result )
result = int (val2 * (num1 + num2) / num3)
print ( result )
result = (val2 * (num1 + num2)) / num3
print ( result )
result = val2 * ((num1 + num2) // num3)
print ( result )
result = num1 * num3 * 4 % num2 / val3
print ( result )
result = num1 + 1 + num2
print ( result )
```

18. רשום מה מדפיס קטע הקוד:

```
array = [ 0, 2, 5, 8, 1, 5, 9, 4 ]
array [ 0 ], array [ -1 ] = array [ -1 ], array [ 0 ]
for k in range (1, len(array) -1, 2 ) :
    array [ k ] , array [ -k-1 ] = array [ -k-1 ] , array [ k ]
print ( array )
```

19. רשום מה מדפיס קטע הקוד:

```
array = [ 7, 6, 4, 9, 0, 1, 3, 5 ]
for k in range ( 0, len ( array ) , 2 ) :
    array [ k ] = array [ -k-1 ]
print ( array )
```

20. רשום מה מדפיס קטע הקוד:

```
array1 = [ -1, 9, 0, 4, 7, 3, 8 ]
array2 = [ 5, -2, 5, 6, 8, 2, 4 ]
for k in range ( len ( array1 ) ) :
    if array1 [ k ] < array2 [ k ] :
        array2 [ k ] = array1 [ k ]
print ( array2 )
```

21. רשום מה מדפיס קטע הקוד:

```
array1 = [ 1, 2, 3, 4, 5, 6, 7, 8 ]
array2 = [ 3, 4, 2, 0, 5, 7, 9, 2 ]
for k in range ( len ( array1 ) ) :
    if k % 2 :
        array2 [ k ] = array2 [ k ] + array1 [ k ]
    else:
        array2 [ k ] = array2 [ k ] - array1 [ k ]
print ( array2 )
```

22. רשום מה מדפיס קטע הקוד:

```
array = [ 3, 4, 2, 0, 5, 7, 9, 2 ]
for k in range ( 1, len(array) ) :
    array [ k ] = array [ k ] + array [ k - 1 ]
print ( array )
```

תכנות בסיסי

23. תוך התייחסות לקטע הקוד הרצ"ב, מה תהיה תוצאת ביצוע `print (newlist([1,2,3,4]))`?

```
def newlist(a):  
    newl = []  
    for k in range (len(a)):  
        a.reverse()  
        newl.append(a[0])  
        del a[0]  
    return newl
```

[תשובות](#)

[חזור לתוכן](#)

מצא את החסר

בפרק זה דוגמאות של תוכניות, כולל הסבר של מה הן עושות. חלק מהפקודות בתוכניות הוסתרו. עליך להבין את התוכנית ולהשלים את החסר (מתוך התשובות האפשריות).

1. רצ"ב שלד של תוכנית המקבלת שתי רשימות של מספרים ויוצרת רשימה שלישית הכוללת מספרים המורכבים מהמספרים שבשתי הרשימות. האיברים הנמצאים במיקומים הזוגיים מחושבים כסכום שני האיברים המתאימים ואילו האיברים במיקומים האי-זוגיים מחושבים כהפרש שבין האיבר ב- y לאיבר ב- x . למשל, $z[1]=y[1]-x[1]$ ואילו $z[4]=y[4]+x[4]$

```
x = [ 12, 34, 22, 53, 11, 2, 47, 17 ]
y = [ 22, 31, 17, 32, 44, 51, 25, 35 ]
z = [ 0, 0, 0, 0, 0, 0, 0, 0 ]
**** 1 ****
**** 2 ****
    z [ k ] = y [ k ] + x [ k ]
else:
**** 3 ****
```

את הפקודות המוגדרות ע"י **** 1 ****, **** 2 ****, **** 3 **** יש להחליף ע"י הפקודות:

א.

```
for k in range ( len ( x - 1 ) ) :
    if k !% 2 :
        z [ k ] = x [ k ] - y [ k ]
```

ב.

```
for k in range ( len ( x ) - 1 ) :
    if k % 2 :
        z [ k ] = y [ k ] - x [ k ]
```

ג.

```
for k in range ( len ( x ) ) :
    if k % 2 == 0 :
        z [ k ] = y [ k ] - x [ k ]
```

ד.

```
for k in range ( len ( x ) ) :
    if k % 2 :
        z [ k ] = y [ k ] - x [ k ]
```


תכנות בסיסי

2. רצ"ב שלד של תוכנית המקבלת רשימה של שמונה מספרים ומייצרת רשימה אחרת, כאשר כל מספר ברשימה החדשה חושב על בסיס המספר ברשימה המקורית. את המספר הראשון מכפילים פי שמונה, את השני פי שבעה וכן הלאה. את המספר הלפני אחרון מכפילים פי שניים ואת המספר האחרון משאירים כפי שהוא.

```
x = [ 11, 21, 34, 35, 37, 19, 07, 16 ]
y = [ 0, 0, 0, 0, 0, 0, 0, 0 ]
**** 1 ****
      **** 2 ****
        **** 3 ****
```

את הפקודות המוגדרות ע"י **** 1 ****, **** 2 ****, **** 3 **** יש להחליף ע"י הפקודות:

א.

```
for k in range ( len ( x ) ) :
    j = k + 1
    y [ - j ] = x [ - j ] * j
```

ב.

```
for k in range ( len ( x ) ) :
    j = k - 1
    y [ - j ] = x [ - j ] * j
```

ג.

```
for k in range ( len ( x ) ) :
    j = k + 1
    y [ j ] = x [ j ] * j
```

ד.

```
for k in range ( len ( x ) - 1 ) :
    j = k + 1
    y [ - j ] = x [ - j ] * ( j + 1 )
```

תכנות בסיסי

3. רצ"ב שלד של תוכנית המקבלת רשימת מספרים ומדפיסה את המספר הקטן ביותר, המספר הגדול ביותר והממוצע.

```
x = [ 55, 78, 88, 91, 67, 84, 93, 73, 56, 73, 74, 81, 99, 93 ]
min , max, avg = x [ 0 ], x [ 0 ], x [ 0 ]
**** 1 ****
    avg = avg + x [ k ]
**** 2 ****
    max = x [ k ]
    if min > x [ k ]:
        min = x [ k ]
**** 3 ****
print ( "min =", min, "max =", max, "average =", avg )
```

את הפקודות המוגדרות ע"י **** 1 **** , **** 2 **** , **** 3 **** יש להחליף ע"י הפקודות:

א.

```
for k in range( len ( x ) - 1 ) :
    if x [ k ] > max:
avg = avg / ( len ( x ) + 1)
```

ב.

```
for k in range(1, len ( x ) ) :
    if x [ k ] > max:
avg = avg / ( len ( x ) )
```

ג.

```
for k in range( len ( x ) ) :
    if x [ k ] < max:
avg = avg / ( len ( x ) + 1)
```

ד.

```
for k in range( len ( x ) ) :
    if x [ k ] < max:
avg = avg / ( len ( x ) )
```

4. רצ"ב שלד של תוכנית המקבלת רשימת מספרים ומדפיסה את ממוצע המספרים וכן כמה מספרים ברשימה קטנים מהממוצע וכמה מספרים גדולים ממנו.

```
x = [ 12, 15, 61, 32, 37, 45, 26, 29, 34, 36, 41, 32, 39 ]
more , less, avg = 0, 0, 0.
for k in range ( len ( x ) ) :
    **** 1 ****
    avg = avg / len ( x )
    for k in range ( len ( x ) ) :
        **** 2 ****
        more = more + 1
        **** 3 ****
        less = less + 1
    print ( "average =", avg, "above average =", more, "less =", less )
```

את הפקודות המוגדרות ע"י **** 1 **** , **** 2 **** , **** 3 **** יש להחליף ע"י הפקודות:

<pre>avg = x [k] if x [k] < avg : if x[k] > avg :</pre>	.א
<pre>avg = avg + x [k] if x [k] < avg : if x[k] > avg :</pre>	.ב
<pre>avg = avg + x [k] if x [k] > avg : if x[k] < avg :</pre>	.ג
<pre>avg = avg + x [k] if x [k] > avg : if avg < x[k] :</pre>	.ד

תכנות בסיסי

5. רצ"ב שלד של תוכנית המקבלת רשימת מספרים ומחשבת את הממוצע שלהם. לאחר מכן מחשבת את סכום השאריות המתקבלות מחלוקת איברי הרשימה בממוצע.

```
x = [ 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ]
avg, sum = 0, 0
for k in range ( len ( x ) ) :
    **** 1 ****
    **** 2 ****
    for k in range ( len ( x ) ) :
        **** 3 ****
```

את הפקודות המוגדרות ע"י **** 1 **** , **** 2 **** , **** 3 **** יש להחליף ע"י הפקודות:

א.

```
avg = avg + x [ k ]
avg = avg / len ( x )
sum = sum + x [ k ] % avg
```

ב.

```
avg = avg + x
avg = avg / len ( x )
sum = sum + x [ k ] % avg
```

ג.

```
avg = avg + x [ k ]
avg = avg / len ( x )
sum = sum + avg % x [ k ]
```

ד.

```
avg = avg + k
avg = avg / len ( x )
sum = sum + avg % x [ k ]
```

תכנות בסיסי

6. רצ"ב שלד של תוכנית המקבלת רשימת מספרים ומדפיסה את סכום המספרים הזוגיים, סכום המספרים המתחלקים בשלוש וסכום שאר המספרים.

```
x = [ 11, 23, 36, 37, 45, 39, 75, 54, 35, 78, 27, 33, 25 ]
c2, c3, co = 0, 0, 0
for k in range ( len ( x ) ) :
    **** 1 ****
        c2 = c2 + x [ k ]
    **** 2 ****
        c3 = c3 + x [ k ]
    **** 3 ****
        co = co + x [ k ]
```

את הפקודות המוגדרות ע"י **** 1 ****, **** 2 ****, **** 3 **** יש להחליף ע"י הפקודות:

א.

```
if x [ k ] % 2 == 0 :
elif x [ k ] % 3 == 0 :
else: x [ k ] % 2 != 0 and x [ k ] % 3 != 0 :
```

ב.

```
if x [ k ] % 2 != 0 :
else: x [ k ] % 3 != 0 :
if x [ k ] % 2 != 0 and x [ k ] % 3 != 0 :
```

ג.

```
if x [ k ] % 2 == 0 :
if x [ k ] % 3 == 0 :
if x [ k ] % 2 != 0 and x [ k ] % 3 != 0 :
```

ד.

```
if x [ k ] % 2 != 0 :
elif x [ k ] % 3 != 0 :
if x [ k ] % 2 == 0 and x [ k ] % 3 == 0 :
```

תכנות בסיסי

7. רצ"ב שלד של תוכנית המקבלת רשימת מספרים ומדפיסה עבור כל מספר ברשימה את המחלקים שלו.

```
x = [ 10, 42, 26, 25, 34, 64, 72, 50 ]
for k in range ( len ( x ) ) :
    **** 1 ****
        ****2****
            **** 3 ****
                print ( l, end = " " )
    print ( )
```

את הפקודות המוגדרות ע"י **** 1 **** , **** 2 **** , **** 3 **** יש להחליף ע"י הפקודות:

א.

```
print ( x [ k ], ":", end = " " )
for l in range ( 2 , int ( x [ k ] / 2 ) ) :
    if x [ k ] % l :
```

ב.

```
print ( x [ k ], ":", end = " " )
for l in range ( 2 , int ( x [ k ] / 2 ) + 1 ) :
    if not x [ k ] % l :
```

ג.

```
print ( k , ":", end = " " )
for l in range ( 2 , int ( x [ k ] / 2 ) ) :
    if k % l :
```

ד.

```
print ( x [ k ], ":", end = " " )
for l in range ( 2 , int ( x [ k ] / 2 ) + 1 ) :
    if not x [ k ] % k :
```

תכנות בסיסי

8. רצ"ב שלד של תוכנית התוכנית הבודקת 500 מספרים והמדפיסה את כל המספרים אשר סכום מחלקיהם שווה ל – 20.

```
x = 500
for k in range ( 2 , x ) :
    s = 0
    for l in range ( 2 , int ( k / 2 ) + 1 ) :
```

**** 1 ****

****2****

**** 3 ****

את הפקודות המוגדרות ע"י **** 1 **** , **** 2 **** , **** 3 **** יש להחליף ע"י הפקודות:

א.

```
if k % l :
    s = s + k
if s == 20 : print ( k )
```

ב.

```
if not k % l :
    s = s + k
if s == 20 : print ( l )
```

ג.

```
if k % l :
    s = s + l
if s != 20 : print ( l )
```

ד.

```
if not k % l :
    s = s + l
if s == 20 : print ( k )
```

תכנות בסיסי

9. שלד הקוד הרצ"ב מדפיס את המספרים החל מ-1, כאשר מספר המספרים המודפסים בכל שורה גדל באחד כל פעם. התוכנית מבקשת מהמשתמש להכניס את מספר השורות ומדפיסה בהתאם. למשל עבור הקלט 6, התוכנית תדפיס את:

```
1
2   3
4   5   6
7   8   9   10
11  12  13  14  15
16  17  18  19  20  21
```

```
def main():
    line = 1
    num = 1
    a = int(input("Enter number of lines: "))
    **** 1 ****
    **** 2 ****
    **** 3 ****
    num = num+1
    print ()
    line = line + 1
```

את הפקודות המוגדרות ע"י **** 1 ****, **** 2 ****, **** 3 **** יש להחליף ע"י הפקודות:

- א. `for k in range(0,a):`
- `for j in range(line):`
- `print (num, "\t",end="")`
- ב. `for k in range(1,a):`
- `for j in range(line,1):`
- `print (num, "\t",end="")`
- ג. `for k in range(0,a):`
- `for j in range(line):`
- `print (num, " ",end="")`
- ד. `for k in range(1,a):`
- `for j in range(0,line,1):`
- `print (num, " ",end="")`

[תשובות](#)

[חזור לתוכן](#)

כתיבת תוכנית

בפרק זה רשימה של אפיונים לתוכנית. עליך לכתוב את התוכנית כך שתבצע את שנדרש. חשוב מאוד לנסות לפתור את התרגילים לבד לפני שפונים לתשובות הכוללות הסברים מפורטים.

1. כתוב תוכנית אשר קוראת שני מספרים המייצגים צלעות של מרובע. עליך לחשב ולהדפיס את היקף המרובע ואת שטחו.
2. כתוב תוכנית אשר קוראת שבעה מספרים, מחשבת את סכומם ואת הממוצע שלהם.
3. הרחב את התוכנית הקודמת, כך שגם תדפיס את המספרים שקראה (כולם באותה שורה ומופרדים ברווח) ורק לאחר מכן תחשב את הסכום והממוצע.
4. כתוב תוכנית הקולטת שני מספרים ומחשבת את סכום המספרים שביניהם (לא כולל המספרים עצמם). אפשר להניח שהמספר הראשון הוא הקטן שבין השניים.
5. כתוב תוכנית אשר מחלקת מספרים. התוכנית קולטת שני מספרים שלמים ומחלקת את השני בראשון. היא מדפיסה את תוצאת החלוקה וכן את השארית שהתקבלה. למשל בחלוקת 9 ל-4 התוצאה תהיה 2 והשארית 1.
6. כתוב תוכנית אשר קוראת שני מספרים המייצגים את אורכי הניצבים של משולש ישר זווית ומחשבת את אורכה של הצלע השלישית (היתר).
7. כתוב תוכנית המקבלת שני קלטים. מספר ותו. התוכנית תדפיס ריבוע בגודל המספר והבנוי מהתווים. למשל עבור 3 ו- "*" התוכנית תדפיס:

8. כתוב תוכנית המקבלת מספר כקלט ומחשבת את כל המחלקים השלמים שלו. למשל עבור 15, המחלקים הם 1 ו-5.
9. פלינדרום היא מלה סימטרית (אפשר לקרוא אותה מימין ומשמאל). כתוב תוכנית אשר קוראת מחרוזת ומחזירה למשתמש אם היא פלינדרום. למשל המלה abba היא פלינדרום, או הערך (מחרוזת) 12321 גם הוא פלינדרום ואילו 123421 אינו פלינדרום.
10. כתוב תוכנית אשר קוראת מספר המכיל כמה ספרות ומדפיסה אותו בסדר הפוך. למשל אם הקלט הוא 12345, אזי התשובה צריכה להיות 54321
11. כתוב תוכנית הצפנה פשוטה הקולטת מחרוזת, באורך כלשהו, של אותיות ורווחים בלבד. התוכנית מצפינה כל תו שבמחרוזת ע"י הוספת 2 לערכו המספרי (הערך של התו ב- ASCII). כלומר התו "a" יהפוך ל- "c", התו "b" יהפוך ל- "d" וכן הלאה. שני התווים האחרונים מתנהגים קצת שונה. התו "y" הופך ל- "a" ואילו התו "z" הופך ל- "b". אפשר להניח שהטקסט שיוכנס הוא תמיד אותיות ורק קטנות כלומר abc... ולא ABC... רווחים שנמצאים במחרוזת נשארים ללא שינוי.
12. חלק מכרטיסי האשראי של ויזה ממשים אלגוריתם (ע"ש luhn) לחישוב חוקיות מספר הכרטיס. על פי אלגוריתם זה, המספר מורכב משש עשרה ספרות, כאשר סכום הביקורת מחושב ע"י חיבור כל הספרות אשר נמצאות במיקומים אי זוגיים (ספרה ראשונה, שלישית וכד') וחיבור "מיוחד" של הספרות במיקומים הזוגיים, לאחר שכל ספרה הוכפלה פי שניים. אם המספר שהתקבל לאחר ההכפלה קטן מ- 10 מחברים אותו אל סכום הביקורת. לעומת זאת, אם המספר שהתקבל לאחר ההכפלה הוא בעל שתי ספרות מחברים את שתי הספרות (אחת לשנייה) ואת הסכום מוסיפים לסכום הביקורת. אם סכום הביקורת שהתקבל, מחיבור כל במספרם, הוא כפולה של 10, אזי המספר חוקי. כתוב תוכנית הקוראת מספר בעל 16 ספרות ובודקת אם הוא מספר כרטיס אשראי חוקי. אפשר להניח שהקלט כולל ספרות בלבד ואורכו 16 ספרות. לדוגמה המספר 4580123456789000 אמור להיות חוקי. החוקיות מחושבת ע"י חיבור הספרות 4,8,1,3,5 וכן הלאה והכפלה (פי שניים) של הספרות במיקומים הזוגיים וחיבורם: $4*2$, $2*2$, $0*2$, $5*2$ וכן הלאה. כאמור במקרה של תוצאת הכפלה שהיא גדולה מ- 10 יש לחבר את הספרות.

ספרה	0	0	0	9	8	7	6	5	4	3	2	1	0	8	5	4
מכפיל	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1
תוצאה	0	0	0	9	16	7	12	5	8	3	4	1	0	8	10	4
>10	0	0	0	9	7	7	3	5	8	3	4	1	0	8	1	4

13. כתוב תוכנית הקוראת שלושה מספרים a, b, c , בעלי אותו מספר ספרות. כל מספר מייצג ווקטור של מספרים חד ספרתיים. בנה רשימה של ערכים d שהינה תוצאת $a+b-c$. אין צורך לבדוק שאין הווקטורים מכילים אותו מספר ספרות.
14. כתוב תוכנית הקוראת שתי רשימות של מספרים. בשתי הרשימות מספר זהה של מספרים. ובנוסף קוראת רשימת בקרה בעלת אותו מספר של איברים. המספרים ברשימת הבקרה כוללים רק את הערכים 0,1,2,3. התוכנית אמורה ליצור רשימה חדשה שבה כל אחד מהאיברים תלוי באיבר המתאים ברשימת הבקרה:
- אם האיבר ברשימת הבקרה הוא 0, אזי האיבר הרשימה החדשה הוא 0.
 - אם האיבר ברשימת הבקרה הוא 1, אזי האיבר ברשימה החדשה הוא האיבר המתאים מתוך הרשימה הראשונה.
 - אם האיבר המתאים ברשימת הבקרה הוא 2, אזי האיבר ברשימה החדשה הוא האיבר המתאים מתוך הרשימה השנייה.
 - ואם האיבר המתאים ברשימת הבקרה הוא 3, אזי האיבר הרשימה החדשה הוא סכום שני האיברים המתאימים בשתי הרשימות.
15. כתוב תוכנית המדפיסה את לוח הכפל. התוכנית מקבלת כקלט מספר המייצג את גודל הלוח ומדפיסה טבלה ישירה בה לכל איבר מוקצים חמישה תווים.
16. כתוב תוכנית הקולטת שני מספרים, כאשר אחד מייצג מספר כלשהו ואילו השני מייצג את הבסיס של המספר הראשון. התוכנית אמורה לחשב את ערכו העשרוני של המספר שנקלט. אין צורך לבדוק את תקינות המספרים.
17. מיון בועות (Bubble sort) הוא אלגוריתם פשוט של מיון. במסגרת האלגוריתם עוברים על כל איברי הסדרה ומשווים כל זוג של איברים, כאשר מתחילים מתחילת הסדרה וממשיכים עד לסופה. מחזורי ההשוואה חוזרים על עצמם עד אשר כל הסדרה ממוינת. האלגוריתם מממש בעזרת שתי לולאות מקוננות, כאשר הראשונה מתבצעת על כל איברי הסדרה ואילו השנייה מתחילה באיבר השני וממשיכה במחזור ראשון עד לאיבר האחרון, במחזור השני עד לאיבר הלפני אחרון וכן הלאה. מחזורי הלולאה השנייה (המקוננת) מתקצרים באיבר אחד בכל פעם משום שבשיטה זו האיבר האחרון בכל מחזור כבר נמצא במקומו. כתוב תוכנית למימוש מיון בועות. התוכנית תקרא רשימה של מספרים ותדפיס אותם לאחר שמוינו.
18. עיגול חסום בריבוע, כך שצלעות הריבוע משיקות לעיגול. כתוב תוכנית המחשבת את שטחו של הריבוע שאינו מכוסה ע"י העיגול (החלק הכחול שבתרשים). התוכנית אמורה להדפיס סדרה המייצגת את שטחי ריבועים שאינם מכוסים ע"י עיגולים עבור סדרת הריבועים שאורך הצלעות שלהם נתון ע"י: 1,2,3...9. כלומר המספר הראשון שיודפס הוא שטחו של ריבוע שאורך צלעו הוא 1, המספר השני עבור ריבוע שאורך צלעו 2 וכן הלאה. הסדרה המודפסת צריכה הינה בדיוק של שלוש ספרות מימין לנקודה העשרונית.
19. כתוב תוכנית הקוראת רשימה של מספרים שלמים המייצגים רדיוסים של עיגולים. אפשר להניח שהמספרים ממוינים ומוכנסים כאשר המספר הגדול הוא הראשון. העיגולים נמצאים האחד בתוך השני, כך שהם יוצרים סדרה של טבעות (לדוגמה הטבעת הכחולה והטבעת הצהובה. התוכנית אמורה לחשב את שטחה של כל טבעת ואת השטח הכולל של כל הטבעות גם יחד. את שטח הטבעת יש לבטא כמספר בדיוק של שלוש ספרות מימין לנקודה העשרונית ואילו את השטח הכולל של כל הטבעות יש לבטא כמספר בדיוק של שתי ספרות מימין לנקודה העשרונית. שים לב שבניגוד לתרשים, הכולל שלושה מספרים בלבד, בפועל מספר המספרים המופיעים בקלט יכול להיות שונה.
20. כתוב תוכנית הקולטת שני מספרים שלמים (a, b) . התוכנית אמורה לחשב את החזקה המרבית (n) של המספר הראשון המקיימת $a^n < b$. ההדפסה צריכה לכלול את החזקה ובסוגריים (ללא רווחים) את המספר בחזקה המתאימה והתוצאה שהתקבלה. אפשר להניח שהמספר השני גדול מהמספר הראשון.
21. כתוב תוכנית הקולטת מחרוזת (a) ומספר שלם (n) . התוכנית מדפיסה את המחרוזת המקורית ואחריה מחרוזת חדשה שבה כל איבר n -י משוכפל. למשל אם המחרוזת הראשונית היא abcdefg והמספר שהתקבל הוא 2, אזי המחרוזת החדשה תהיה: abbcddeffg.

22. מספרי BCD (Binary Coded Decimal) הם מספרים בינאריים בגודל של ארבע סיביות ואשר כל אחד מייצג ספרה עשרונית אחת. כלומר ערך כל ספרה הוא 0-9. כתוב תוכנית שקוראת רשימה של מספרי BCD ומחשבת את ערכה העשרוני של הרשימה כולה. למשל אם הרשימה שנקראה

תכנות בסיסי

- כוללת: 1001,1000,0111,1001 אזי ערכה העשרוני הוא 9879. אפשר להניח שהמספרים הבינאריים הם באמת מספרי BCD ואין צורך לבדוק זאת.
23. כתוב תוכנית אשר קוראת רשימה של מספרים ומייצרת רשימה חדשה שבה כל איבר הוא סכום האיברים שמשני צדדיו. התייחס לרשימה כאילה הייתה מעגלית, כלומר האיבר הראשון ברשימה החדשה יהיה סכום האיבר האחרון והשני ברשימה המקורית ואילו האיבר האחרון ברשימה החדשה יהיה סכום האיבר הלפני אחרון והראשון ברשימה המקורית.
24. כתוב תוכנית הקוראת רשימה של מספרים ומייצרת רשימה חדשה הכוללת רק מספרים "יחודיים" (שלא חוזרים על עצמם). במילים אחרות, איברים כפולים שהיו ברשימה המקורית נמצאים רק פעם אחת ברשימה החדשה.
25. כתוב תוכנית הקוראת רשימה של מספרים ומייצרת רשימה חדשה שבה האיברים הם סכומי ביניים. כל איבר הוא סכום האיברים שלפניו (האיבר הראשון זהה, האיבר השני הוא סכום שני האיברים הראשונים וכן הלאה)
26. כתוב תוכנית הקוראת מספר (n) בתחום שבין שלוש לתשע ומדפיסה את כל המספרים בעלי שלוש ספרות בבסיס n. אם המספר שהוקלד שגוי התוכנית תחזור ותבקש מספר חדש שעונה לדרישה.

[תשובות](#)

[חזור לתוכן](#)

כתיבת פונקציות

1. כתוב פונקציה המקבלת פרמטר אחד שהוא מחרוזת ומחזירה אותה בסדר הפוך.
2. כתוב פונקציה המקבלת פרמטר מספרי המגדיר את אורך הניצבים של משולש ישר זווית ושווה שוקיים. הפונקציה "מציירת" את המשולש תוך שימוש ב"***". למשל אם הפרמטר שהוכנס הוא 4, אזי תוצאת הקריאה לפונקציה תהיה:

```
*
**
***
****
```

3. כתוב פונקציה המקבלת שני פרמטרים (מערך ותו). הפונקציה מחזירה מערך המבוסס על המערך המקורי לאחר שכל המופעים של התו (שהועבר כפרמטר שני) בוטלו.
4. כתוב פונקציה המקבלת רשימה של איברים מספריים המייצגים ציונים. הפונקציה אמורה להחזיר שלוש תוצאות (ממוצע הציונים, מספר הציונים הגבוהים או שווים לממוצע ומספר הציונים הנמוכים מהממוצע)
5. כתוב פונקציה המקבלת מערך של תווים ומחזירה מערך אחר המבוסס על המערך המקורי לאחר שכל הרווחים שבו הוסרו. למשל אם הפונקציה תקבל מערך: "Shalom All, How Are You?", היא אמורה להחזיר: "ShalomAll,HowAreYou?"
6. מספרי ארמסטרונג (Armstrong numbers) הם מספרים מיוחדים המקיימים את התנאי שערך המספר שווה לסכום הספרות שלו בחזקת מספר הספרות. למשל 153 הוא מספר המקיים את התנאי שכן $153 = 1^3 + 5^3 + 3^3$. כתוב פונקציה המקבלת פרמטר מספרי אחד (n) ומדפיסה את כל מספרי ארמסטרונג בתחום שבין 1 ל-n
7. כתוב פונקציה המקבלת שלושה מספרים a, b, c (אשר מייצגים את אורכי צלעות המשולש) ומחשבת את שטחו. השתמש בנוסחת הרון לפיה $A = \sqrt{s(s-a)(s-b)(s-c)}$, כאשר $s = \frac{1}{2}(a+b+c)$
8. במספר מדינות נהוגה שיטה שונה של ציונים. A עבור הציונים 90-100, B עבור 80-89, C עבור 70-79, D עבור 60-69, ו-F עבור ציונים נמוכים מ-60. כתוב פונקציה המקבלת ציון "רגיל" ומחזירה את האות המתאימה לציון.
9. כתוב פונקציה המחשבת את סכום הכסף שנמצא בשקית. הפונקציה מקבלת ששה פרמטרים המייצגים:

- a. את מספר מטבעות ה-10 אגורות שבשקית,
- b. את מספר מטבעות החצי שקל שבשקית
- c. את מספר מטבעות השקל שבשקית
- d. את מספר מטבעות השני שקלים שבשקית
- e. את מספר מטבעות החמישה שקלים שבשקית
- f. את מספר מטבעות העשרה שקלים שבשקית

10. בית קולנוע גובה 40 שקלים מכל לקוח. הקרנת סרט עולה 300 שקלים ועוד תמלוגים בגובה של 5 שקלים לצופה (לקוח). כתוב פונקציה המקבלת פרמטר מספרי (מספר הלקוחות בהצגה מסוימת) ומחשבת את ההפסד/הרווח של הקולנוע.

11. כתוב פונקציה המחשבת את מס ההכנסה לשכיר בשנת 2009. הפונקציה מקבלת פרמטר אחד (השכר החודשי) ומחזירה את גובה המס לתשלום. מדרגות המס לשנת 2009 הן:

- a. עד שכר של 4,390 ₪ - 10%
- b. על סכום שבין 4,391 ועד 7,810 ₪ - 15%
- c. על סכום שבין 7,811 ועד 11,720 ₪ - 23%
- d. על סכום שבין 11,721 ועד 16,840 ₪ - 30%
- e. על סכום שבין 16,841 ועד 36,260 ₪ - 34%
- f. על סכומים הגבוהים מ-36,261 ₪ - 46%

12. כתוב פונקציה המקבלת שני פרמטרים: רשימה של מספרים ומספר. הפונקציה מחזירה רשימה הכוללת רק את האיברים ברשימה המקורית, אשר גדולים מהמספר (הפרמטר השני)
13. כתוב פונקציה המקבלת רשימה של ערכים מספריים ומחזירה את המספר הקטן ביותר ברשימה, המספר הגדול ביותר ברשימה והממוצע שלה.
14. כתוב פונקציה המקבלת שני פרמטרים: מערך בגודל כלשהו ומספר (n). הפונקציה אמורה להחזיר מערך הכולל רק את האיברים ה-n במערך המקורי. למשל אם המערך המקורי הוא: "abcdefghijklmnp" והמספר שהועבר הוא 3, אזי התוצאה תהיה: "cfilo".

15. כתוב פונקציה המקבלת ערך מספרי (n) ומחשבת את סכום ריבועי המספרים בתחום: n-1 (כולל n).
 16. כתוב פונקציה המקבלת מחרוזת של תווים מסוגים שונים ומחזירה אותה, כאשר האותיות הקטנות (Lower Case) אם קיימות הפכו לאותיות גדולות (Upper Case).
 17. כתוב פונקציה המקבלת פרמטר מספרי ומחזירה את המחלק הקטן ביותר של (ובלבד שהמחלק גדול מאחד)
 18. כתוב פונקציה המקבלת פרמטר שהוא מספר חיובי ומחזירה את הספרה השמאלית שלו
 19. כתוב פונקציה אשר מקבלת רשימה של מספרים. הפונקציה מסכמת את כל המספרים הזוגיים ואת כל המספרים האי-זוגיים ומחזירה "ODD" אם סכום המספרים האי-זוגיים גדול יותר, "EVEN" אם סכום המספרים הזוגיים גדול יותר ו- "NONE" אם שני הסכומים שווים.
 20. כתוב פונקציה המקבלת שתי מחרוזות המייצגות מספרים בינאריים. הנח שהמחרוזות באורך שווה. הפונקציה אמורה לייצר מחרוזת חדשה שהיא תוצאת הביצוע של הפעולה הבוליאנית XOR (Exclusive OR). פונקציה זו מקבלת שני פרמטרים בינאריים ומחזירה 1 אם שניהם שונים ואפס אם הם זהים. הפונקציה שיש לכתוב מבצעת XOR על כל איברי המחרוזות.
 21. כתוב פונקציה המקבלת פרמטר שהוא רשימה ומחזירה אותה לאחר שהחליפה את המיקומים של האיבר הקטן ביותר והאיבר הגדול ביותר. כלומר במקום שבו היה האיבר הקטן, ימצא האיבר הגדול ולהיפך.
- למשל עבור הקלט [3, 4, 2, 7, 11, 5, 4, 3], הפונקציה מחזירה [3, 4, 11, 7, 2, 5, 4, 3]

[תשובות](#)

[חזור לתוכן](#)

רקורסיה

רקורסיה היא דרך מקובלת לפיתרון בעיות (לא רק בפיתוח תוכנה), אשר עושה שימוש במנגנון של "הקטנת" הבעיה שיש לפתור. הרעיון הבסיסי הוא הפשטת הבעיה ממחזור למחזור עד אשר נגיע לבעיה פשוטה שאותה אנחנו יודעים לפתור בקלות. הדרך לעשות זאת היא ע"י שימוש בפונקציה עצמה, אולם רק על חלק מהבעיה הנדונה וע"י כך בכל מחזור אנחנו מתקדמים לקראת הפיתרון המלא. הדוגמה הנפוצה ביותר היא הגדרת הפונקציה עצרת. כדי לחשב למשל את $5!$, עלינו לכפול את כל האיברים בתחום שבין 1 ל- 5 . אפשר כמובן לבצע זאת ע"י לולאה, אולם רקורסיה (לאחר שהובנה) מספקת פיתרון קל ופשוט יותר. נניח שעלינו לפתח פונקציה רקורסיבית אשר אמורה לחשב את $n!$, מתוך הפרמטר n שמועבר לה. כדי לעשות זאת עלינו להבין שהכלל המאפשר לחשב את $n!$, משמעותו $n \cdot (n-1)!$... $1! = n \cdot (n-1) \cdot (n-2) \cdot \dots \cdot 1$. במילים אחרות חישוב $n!$ משמעותו הכפלה של n בביטוי $(n-1)!$. ואילו הביטוי $(n-1)!$ משמעותו $(n-2)! \cdot (n-1)$. וכן הלאה. עלינו גם להגדיר את תנאי היציאה ובמקרה זה אם $n=1$, אזי הפונקציה מחזירה את הערך 1 . בכל מצב אחר, הפונקציה מחזירה את הפרמטר שקיבלה מוכפל בערך הפונקציה המחושב על הפרמטר פחות אחד. הפונקציה הרקורסיבית אפשרית עקב העובדה שכל פונקציה מופרדת מפונקציות אחרות ואפילו מופרדת ממופנע אחר שלה עצמה. לכן כדי להגדיר פונקציה רקורסיבית, שי להגדיר:

- א. את תנאי היציאה. למשל במקרה של עצרת התנאי יהיה עבור $n=1$, הפונקציה תחזיר 1 .
- ב. את התנהגות הפונקציה במקרים האחרים. בדוגמת העצרת עבור כל n אחר (שונה מאחד), הפונקציה תחזיר את $(n-1) \cdot n!$. במילים אחרות היא תחזיר את n מוכפל בקריאה לפונקציה עם הערך $(n-1)$.

תרגילי חזרה

1. כתוב פונקציה המקבלת שני פרמטרים מספריים (a, b) , כאשר הראשון מייצג מספר מסוים ואילו השני מייצג את החזקה. הפונקציה מחשבת a^b . למרות שבפיתוח קיימת פונקציה כזאת, יש לפתח אותה ללא קשר לזו הקיימת. את הפונקציה יש לפתח פעמיים. פעם כלולאה ופעם כרקורסיה. אפשר להניח שהמספרים תקינים ואין צורך לבדקם.
2. כתוב פונקציה רקורסיבית המקבלת פרמטר מספרי ומחזירה את כל המספרים ממנו ועד אפס. למשל עבור 9 , הפונקציה תחזיר את: 9876543210 .
3. סדרת פיבונצ'י היא סדרה של איברים המקיימת: האיבר הראשון הוא 1 , האיבר השני הוא 1 וכל איבר מעבר לזה הוא סכום שני האיברים שלפניו. כתוב פונקציה רקורסיבית אשר מקבלת פרמטר מספרי (n) ומחשבת את ערכו של המספר n בסדרה.
4. כתוב פונקציה רקורסיבית אשר מקבלת פרמטר שהוא מחרוזת ומחזירה אותה בסדר הפוך
5. כתוב פונקציה רקורסיבית המקבלת פרמטר מספרי (n) ומחשבת את סכום כל הספרות מאחד ועד n .
6. נתונה סדרה המקיימת $a_1=5$ ולכל איבר אחר, $a_n = a_{n-1} + 3$. כתוב פונקציה רקורסיבית המקבלת פרמטר מספרי (n) ומחזירה את האיבר n בסדרה.
7. כתוב פונקציה רקורסיבית המקבלת פרמטר שהוא מספר ומחזירה את סכום ספרותיו
8. כתוב פונקציה רקורסיבית המקבלת מספר עשרוני וממירה אותו למספר בינארי
9. כתוב פונקציה רקורסיבית המקבלת רשימה של מספרים ומחזירה את סכומם.
10. במחשב מסוים שכחו מהנדסי החומרה לממש פקודת כפל. כתוב פונקציה רקורסיבית המקבלת שני מספרים טבעיים (חיוביים) ומחשבת את המכפלה שלהם.
11. כתוב פונקציה רקורסיבית המקבלת מספר טבעי (n) ומחזירה את סכום איברי הסדרה הנתונה ע"י: $1^2, 2^2, 3^2, \dots, n^2$.
12. כתוב פונקציה רקורסיבית המקבלת פרמטר אחד (n) שהינו מספר טבעי חיובי. ומחשבת את הערך n^2 , מבלי להשתמש בפקודות כפל, אלא בידע ש- $1^2=1$.
13. מספרי פרין (Perrin) הם מספרים המייצגים סדרה של מספרים אשר מקיימת את התנאים: $P(0)=3, P(1)=0, P(2)=2, P(n)=P(n-2)+P(n-3)$. מ מייצג מספר שאינו $0, 1, 2$. כתוב פונקציה רקורסיבית אשר מקבלת פרמטר שהוא מספר טבעי חיובי ומחזירה את מספר פרין המתאים.
14. כתוב פונקציה רקורסיבית אשר מקבלת מספר אמיתי (n) ומדפיסה משולש של כוכבים (*). כאשר מספר הכוכבים בשורה הראשונה הוא n ובכל שורה המספר קטן באחד.
15. כיצד משתנה תשובתך אם ברצוננו להדפיס משולש נוסף (הפוך) לאחר המשולש הראשון. כלומר להדפיס את המשולש הראשון ואחריו משולש נוסף, המתחיל בכוכב אחד בשורה הראשונה ובכל שורה נוספת מודפס עדו כוכב עד לשורה בעלת n כוכבים

16. כתוב פונקציה רקורסיבית המקבלת פרמטר (n) שהוא מספר טבעי שלם וחיובי ומחזירה את סכום האיברים של הסדרה הרמונית. סדרה הרמונית היא הסדרה: $1/n + 1/(n-1) + 1/(n-2) + \dots + 1/2 + 1$
17. מספרי לוקאס (Lucas) הם הכללה של מספרי פיבונצ'י וגם הם מוגדרים באופן כללי ע"י הנוסחה: $L_n = L_{n-1} + L_{n-2}$. אולם בניגוד למספרי פיבונצ'י בהם שני האיברים הראשונים מוגדרים, במספרי לוקאס ניתן להגדיר אותם ולכן מספרי פיבונצ'י הם מקרה פרטי של מספרי לוקאס. כתוב פונקציה אשר מקבלת שלושה מספרים טבעיים a, b, c . המספר הראשון מייצג את האיבר הראשון בסדרה, המספר השני מייצג את המספר השני בסדרה והמספר השלישי מייצג את האיבר שהפונקציה צריכה לחשב. שים לב שאיברי הפונקציה מתחילים מאפס.
18. כתוב פונקציה רקורסיבית המקבלת פרמטר שהוא מספר טבעי חיובי ומחזירה את מספר האפסים שבו
19. כתוב פונקציה רקורסיבית אשר מקבלת שני מספרים טבעיים (n, m) , ומחזירה את מספר הפעמים ש m מחלק את n . למשל עבור הצמד 23,3 התשובה צריכה להיות 0 ואילו עבור הצמד 27,3 התשובה צריכה להיות 3 ($27=3*3*3$).
20. כתוב פונקציה רקורסיבית אשר מקבלת מערך בגודל כלשהו ומחזירה את גודלו (בלי להשתמש בפונקציה len).

[תשובות](#)

[חזור לתוכן](#)

תרגילי תכנות ללא פתרון

1. כתוב תוכנית הקולטת מהמשתמש את אורך צלע הריבוע ומחשבת את שטחו ואת היקפו. לצורך התרגיל הנח שאורך הצלע מבוטא במספרים שלמים וחייביים

לדוגמה, בהכנסת הערך 6, תתקבל הדפסה כמו:

The area of a square with and side length of 6 is 36
and the circumference is 24

2. שנה את התרגיל הקודם, כך שהתוכנית תקבל גם אורכים שהם מספרים ממשיים (שבריים).

לדוגמה, בהכנסת הערך 6, תתקבל הדפסה כמו:

The area of a square with and side length of 7.5 is 56.25
and the circumference is 30.0

3. כתוב תוכנית הקולטת מהמשתמש שני מספרים שלמים המייצגים תחום ומחשבת את סך כל המספרים שבתחום (כולל שני מספרי הקלט). בשלב זה, אפשר להניח שהמספר הראשון תמיד יהיה הקטן שבין השניים וכן שלא מדובר בשני מספרים זהים.

דוגמת הרצה:

Please enter first number: 4

Please enter second number: 6

The sum of all numbers in the range [4 , 6] is: 15

4. שנה את התוכנית הקודמת, כך שתחשב את מכפלת כל המספרים שבתחום.

דוגמת הרצה:

Please enter first number: 3

Please enter second number: 5

The product of all numbers in the range [3 , 5] is: 60

5. כתוב תוכנית המחשבת ערך של פונקציה מתמטית: $f(x) = 3x^2 + 2x + 13$. הפונקציה תקלוט מהמשתמש את x ותחשב את ערך הפונקציה.

דוגמת הרצה:

Please enter the value of x: 2

For x = 2.0 the function value is: 29.0

6. כתוב תוכנית הקולטת מספר מהמשתמש ומחשבת את המספר בריבוע, המספר בחזקת שלוש ובחזקת ארבע.

דוגמת הרצה:

Please enter the value of x: 5

For x = 5.0, the powers are: 25.0 125.0 625.0

וגם:

Please enter the value of x: 4.5

For x = 4.5, the powers are: 20.25 91.125 410.0625

תכנות בסיסי

7. שלשה פיתגורית היא שלושה מספרים a, b, c המקיימים $a^2 + b^2 = c^2$. כתוב תוכנית הקולטת מהמשתמש שני מספרים ומדפיסה את השלשה הפיתגורית המורכבת משני מספרים אלה.

דוגמת הרצה:

Please enter first value: 3
Please enter second value: 4
The Pythagorean triple is: 3.0 4.0 5.0

8. כתוב תוכנית הקולטת מספר n , גדול מ-5 ומחשבת את סכום המספרים הזוגיים שבתחום $[0, n]$ וכן את סכום המספרים האי זוגיים שבתחום. שים לב שהתחום כולל גם את n . אין צורך לבדוק את תקינות המספר n . התוכנית תדפיס את שני הסכומים.

דוגמת הרצה:

Please enter range upper limit: 10
The even numbers sum is: 30
The odd numbers sum is: 25

9. כתוב תוכנית הקולטת מספר x , המייצג את אורך הצלע של קובייה. חשב את יחס הנפחים של הקובייה לכדור בעל רדיוס באורך x . נפח הקובייה מחושב ע"י אורך הצלע בחזקת שלוש ואילו נפח כדור מחושב ע"י הרדיוס בחזקת שלוש כפול $\frac{4}{3}\pi$ ומוכפל ב- x^3 . $V = \frac{4}{3}\pi x^3$. עבור החישוב השתמש בערך של π מתוך הספרייה המתמטית. אין צורך לבדוק את תקינות המספר x . התוכנית תדפיס את שני הנפחים ואת היחס ביניהם.

דוגמת הרצה:

Please enter size: 10
For size: 10.0
the cube volume is: 1000.0
the ball volume is: 4188.790204786391
and their ratio is: 0.238732414637843

10. בארה"ב נהוג לייצג גובה ע"י שילוב של רגלים (feet) ואינצ'ים (inch). אורכו של אינץ' הוא 2.54 ס"מ ואילו רגל היא 12 אינצ'ים. כתוב תוכנית הקולטת שני מספרים (רגל ואינץ') אשר יחד מייצגים גובה. התוכנית תמיר את הגובה מהשיטה האמריקאית לס"מ כפי שנהוג בארץ.

דוגמת הרצה:

Please enter number of feet: 5
Please enter number of inches: 8
Height: 5.0 ft , 8.0 in is: 172.72 centimeters

11. כתוב תוכנית אשר ממירה מספרים בעלי שלוש ספרות מבסיס כלשהו לבסיס עשרוני. התוכנית תקלוט שלוש ספרות x_1, x_2, x_3 , המייצגות את המספר שיש להסב. x_1 היא ספרת המאות, x_2 היא ספרת העשרות ואילו x_3 היא ספרת היחידות. ולאחר מכן תקלוט את הבסיס (b) שבו ספרות אלה רשומות. אפשר להניח שהבסיס b קטן מעשר. התוכנית תמיר את הערך למספר עשרוני.

דוגמת הרצה:

Please enter first (hundreds) number: 1

תכנות בסיסי

Please enter second (tens) number: 2
Please enter third (ones) number: 3
Please enter base: 8
123 in base 8 = 83 in decimal

12. כתוב תוכנית אשר קולטת שני מספרים ומדפיסה את סכומם, אלא אם כן שני המספרים זהים ואז היא מדפיסה את מכפלתם

דוגמת הרצה:

Please enter first number: 12
Please enter second number: 18
For the numbers: 12 , 18 the result is: 30

וגם:

Please enter first number: 12
Please enter second number: 12
For the numbers: 12 , 12 the result is: 144

13. כתוב תוכנית אשר קולטת שני מספרים ומדפיסה True אם אחד מהם הוא 10, או אם סכומם הוא 10. בכל שאר המקרים היא תדפיס False.

דוגמאות הרצה:

Please enter first number: 8
Please enter second number: 2
True
>>>
Please enter first number: 9
Please enter second number: 3
False
>>>
Please enter first number: 10
Please enter second number: 1
True

14. כתוב תוכנית אשר קולטת מספר ומדפיסה Odd אם הוא אי-זוגי, או Even אם הוא זוגי..

דוגמאות הרצה:

Please enter a number: 3
Odd
>>>
Please enter a number: 66
Even

15. כתוב תוכנית אשר קולטת מספר בן שתי ספרות. התוכנית תדפיס Bingo, אם המספר מתחלק בשבע או שהוא מכיל את הספרה 7. בכל מצב אחר, התוכנית תדפיס Better luck next time. הערה: מספר בעל שתי ספרות מכיל תא הספרה 7 אם בחלוקתו בעשר התוצאה השלמה היא 7, או השארית היא 7. אין צורך לבדוק תקינות של המספר.

דוגמאות הרצה:

Please enter a two digit number: 84

```
Bingo
>>>
Please enter a two digit number: 71
Bingo
>>>
Please enter a two digit number: 34
Better luck next time
>>>
Please enter a two digit number: 27
Bingo
```

16. שנה את התוכנית הקודמת (כתוב תוכנית אשר קולטת מספר בן שתי ספרות. התוכנית תדפיס Bingo, אם המספר מתחלק בשבע או שהוא מכיל את הספרה 7. בכל מצב אחר, התוכנית תדפיס Better luck next time), כך שבמקום מספר אחד, היא תקלוט חמישה מספרים ותבצע על כל אחד מהם את הבדיקה.

דוגמת הרצה:

```
Please enter a two digit number: 21
Bingo
Please enter a two digit number: 57
Bingo
Please enter a two digit number: 22
Better luck next time
Please enter a two digit number: 99
Better luck next time
Please enter a two digit number: 98
Bingo
```

17. כתוב תוכנית אשר קולטת שני מספרים ומדפיסה את הגדול שביניהם. אם הם שווים היא תדפיס The numbers are equal

דוגמאות הרצה:

```
Please enter first number: 123
Please enter second number: 123.1
The largest number is: 123.1
>>>
Please enter first number: 33
Please enter second number: 44
The largest number is: 44.0
>>>
Please enter first number: 99.99
Please enter second number: 99.99
The numbers are equal
```

18. כתוב תוכנית אשר קולטת שלושה מספרים ומדפיסה את הגדול שביניהם.

דוגמת הרצה:

```
Please enter first number: 11
Please enter second number: 22
Please enter second number: 33
The largest number is: 33.0
```

תכנות בסיסי

19. כתוב תוכנית אשר קולטת שני מספרים a , b ומדפיסה את כל המספרים מאחד ועד a (כולל), בתנאי שהם לא מתחלקים ב- b . אין צורך לבדוק את תקינות המספרים.

דוגמת הרצה:

```
Please enter first number: 10
Please enter second number: 3
1
2
4
5
7
8
10
```

20. כתוב תוכנית אשר קולטת שני מספרים a , b ומסכמת את כל המספרים בתחום מאחד ועד a (כולל), אשר מתחלקים ב- b . אין צורך לבדוק את תקינות המספרים.

דוגמת הרצה:

```
Please enter first number: 23
Please enter second number: 6
The sum of all numbers in the range 1 - 23 is: 36
```

21. חנות מקוונת שולחת מוצרים למדינות ברחבי העולם. עלות המשלוח משתנה בהתאם למשקל החבילה. לכל חבילה עלות של \$3 דמי טיפול. מחיר זה כולל משלוח חינם לחבילה שמשקלה לא יותר מק"ג אחד. עבור חבילה שמשקלה לא יותר מחמישה ק"ג יש להוסיף \$2.5 לכל ק"ג (מעבר לק"ג הראשון). עלות המשלוח של חבילות כבדות יותר היא \$2 לכל ק"ג מעל 5 הק"ג הראשונים. אין אפשרות למשלוח של חבילות השוקלות מעל 10 ק"ג. כתוב תוכנית הקולטת את משקל החבילה ומחשבת את עלות המשלוח.

דוגמאות הרצה:

```
Please enter package weight: 0.1
Shipping a 0.1 kg package costs $ 3
>>>
Please enter package weight: 3
Shipping a 3.0 kg package costs $ 8.0
>>>
Please enter package weight: 10.1
Sorry, too heavy!
>>>
Please enter package weight: 7.4
Shipping a 7.4 kg package costs $ 11.3
```

22. כדי לעודד את השימוש במסרונים (SMS) ברשת הטלפוניה הקווית, יצאה בזק במבצע מיוחד הזמין רק ללקוחות הרשומים לתוכנית. עלות התוכנית היא 15 ₪ בחודש ובמסגרתה עלויות משלוח המסרונים הן כמפורט בטבלה:

מספר מסרונים בחודש	מחיר
1-500	חינם (כלול בתוכנית)
501-1000	10 אג' למסרון
1001-5000	8 אג' למסרון
5001-10000	5 אג' למסרון
>10000	3 אג' למסרון

כתוב תוכנית הקולטת את מספר המסרונים ומחשבת את העלות (מעבר לעלות הקבועה של התוכנית)

דוגמאות הרצה:

```
Please enter number of SMS: 6000
the cost of 6000 SMS is: 420.0 Shekels
>>>
Please enter number of SMS: 1000
the cost of 1000 SMS is: 50.0 Shekels
>>>
Please enter number of SMS: 10000
the cost of 10000 SMS is: 620.0 Shekels
>>>
Please enter number of SMS: 12000
the cost of 12000 SMS is: 680.0 Shekels
>>>
Please enter number of SMS: 500
Sending 500 SMS is free of charge!
```

23. לאור המחסור במים, הוחלט בממשלה להעלות את תעריפי המים. התעריפים המעודכנים לשנת 2011 כוללים שני תעריפים. תעריף א' של 7.44 ₪ לקוב מים ותעריף ב' של 11.923 ₪ לקוב. כל אזרח זכאי ל – 3.5 קוב מים בתעריף א' בכל חודש, כאשר התשלום עבור הצריכה עודפת מחושב לפי תעריף ב'. מינימום הקצבה בתעריף א' היא 7 קוב (כלומר, בבית שמתגורר בו רק אדם אחד, התשלום על מים בתעריף א' יחושב כאילו בבית גרים שני אנשים). לאחר חישוב עלות המים, יש להוסיף גם מע"מ (16%). כתוב תוכנית הקולטת את מספר האנשים המתגוררים בבית ואת צריכת המים (בקוב) ומחשבת את הסכום לתשלום. ערוך את הסכומים לתשלום המופיעים בהדפסה וצמצם את מספר הספרות שמימין לנקודה העשרונית, כך שיתאימו למציאות (רק שתי ספרות).

דוגמאות הרצה:

```
Please enter number of persons: 2
Please enter number of cubic meters: 12
For 2 persons with 12.0 cubic meters of water the cost is
Water - 111.94
VAT - 17.91
Total - 129.85
>>>
Please enter number of persons: 1
Please enter number of cubic meters: 6
For 1 persons with 6.0 cubic meters of water the cost is
Water - 44.64
VAT - 7.14
Total - 51.78
>>>
Please enter number of persons: 3
Please enter number of cubic meters: 56
For 3 persons with 56.0 cubic meters of water the cost is
Water - 622.89
VAT - 99.66
Total - 722.55
```

תכנות בסיסי

24. כתוב תוכנית הקולטת שלושה מספרים שלמים.
- אם המספר הקטן ביותר זוגי, היא מדפיסה את סכום החזקות של המספרים
 - אם המספר הקטן ביותר מתחלק בשלוש היא מדפיסה את מכפלת המספרים
 - בכל מצב אחר היא מדפיסה את סכום המספרים

דוגמאות הרצה:

```
Please enter first number: 2
Please enter second number: 3
Please enter third number: 4
Sum of squares = 29
>>>
Please enter first number: 3
Please enter second number: 7
Please enter third number: 12
Product = 252
>>>
Please enter first number: 5
Please enter second number: 7
Please enter third number: 9
Sum = 21
```

25. כתוב תוכנית הקולטת מספר בעל ארבע ספרות ומדפיסה אותו בסדר הפוך. לאחר מכן מדפיסה את סכום הספרות שלו.

דוגמת הרצה:

```
Please enter a 4 digit number: 1357
1357 <==> 7531 Sum = 16
```

26. כתוב תוכנית הקולטת מספר המייצג שנה (לפי הלוח הנוצרי) ומחשבת אם היא שנה מעוברת. שנה מעוברת היא שנה שבה יש 29 ימים בפברואר. שנה מעוברת היא שנה המתחלקת ב-4, לא מחלקת ב-100, אבל כן מתחלקת ב-400.

דוגמאות הרצה:

```
Please enter the year: 1900
1900 is not a leap year
>>>
Please enter the year: 2000
2000 is a leap year
>>>
Please enter the year: 24
24 is a leap year
```

27. כתוב תוכנית המחשבת את ערכו של המספר 1111, בהנחה שהוא מבוטא בבסיסים שונים. התוכנית קולטת שני מספרים המייצגים טווח בסיסים ומחשבת את ערכו ש 1111 בכל אחד מהבסיסים שבטווח.

דוגמת הרצה:

```
Please enter first base: 2
Please enter last base: 9
Base 2 Value: 15
Base 3 Value: 40
Base 4 Value: 85
Base 5 Value: 156
Base 6 Value: 259
Base 7 Value: 400
Base 8 Value: 585
Base 9 Value: 820
```

תכנות בסיסי

28. כתוב תוכנית קולטת ערך מספרי שלם n ומחשבת את סכום הסדרה הנתונה ע"י $1/(0+1) + 1/(1+2) + 1/(2+3) + \dots + 1/(n-1+n)$ את התוצאה יש להדפיס עם שלוש ספרות מימין לנקודה העשרונית.

דוגמאות הרצה:

```
Please enter the value of n: 5
The sum is: 1.787
>>>
Please enter the value of n: 1000
The sum is: 4.435
>>>
Please enter the value of n: 10000000
The sum is: 9.04
```

29. כתוב תוכנית המחשבת את שכרו של עובד שעתי (עובד לפי שעות). התוכנית קולטת את שעת ההתחלה ושעת הסיום ואת התעריף לשעה. פורמט הקלדת השעות הוא $xx.yy$, כאשר xx מייצג את השעה ואילו yy מייצג את הדקות. כלומר השעה 7 ושש דקות מוקלדת ע"י 7.06 עבור 9 השעות הראשונות ביום התעריף הוא 100%, עבור השעתיים הבאות התעריף הוא 125% והשעות לאחר מכן, 150%. את הסכום המחושב יש לעגל לשקלים שלמים.

דוגמאות הרצה:

```
Please enter starting time: 7.06
Please enter end time: 19.06
Please enter hourly salary: 100
1300
>>>
Please enter starting time: 7.06
Please enter end time: 12.36
Please enter hourly salary: 100
550
>>>
Please enter starting time: 7.24
Please enter end time: 18.36
Please enter hourly salary: 100
1180
```

30. כתוב תוכנית הקולטת מחרוזת כלשהי מדפיסה אותה ולאחר מכן מדפיסה אותה בסדר הפוך, כאשר התו הראשון מודפס אחרון, התו השני מודפס אחר לפני האחרון וכן הלאה. התו האחרון מודפס ראשון.

דוגמת הרצה:

```
Please enter something...: abcdefg
Original string: abcdefg
Inverse string: gfedcba
```

31. כתוב תוכנית הקולטת שתי מחרוזות בעלות אותו אורך ומדפיסה איחוד מסודר שלהן. איחוד מסודר מתואר בדוגמה: אם $a_1a_2a_3a_4a_5a_6 - b_1b_2b_3b_4b_5b_6$ הן שתי המחרוזות, אזי המחרוזת החדשה תהיה $a_1b_1a_2b_2a_3b_3a_4b_4a_5b_5a_6b_6$. במילים אחרות, התוכנית תדפיס מחרוזת חדשה שבה במיקומים האי זוגיים התווים של המחרוזת הראשונה ובמיקומים הזוגיים מופיעים התווים של המחרוזת השנייה. אין צורך לבדוק שהאורכים זהים.

דוגמאות הרצה:

```
Please enter something...: abcdefg
Please enter something...: zyxwvut
```

תכנות בסיסי

```
Original strings: abcdefg zyxwvut
Combined string: azbycxdwefugt
>>>
Please enter something...: 01234
Please enter something...: 56789
Original strings: 01234 56789
Combined string: 0516273849
```

32. שנה את התוכנית של התרגיל הקודם, כך שלאחר כל תו תוסיף רווח ולאחר כל זוג תווים תוסיף רווח, לוכסן ועוד רווח

דוגמאות הרצה:

```
Please enter something...: abcdefg
Please enter something...: zyxwvut
Original strings: abcdefg zyxwvut
Combined string: a z / b y / c x / d w / e v / f u / g t /
>>>
Please enter something...: 01234
Please enter something...: 56789
Original strings: 01234 56789
Combined string: 0 5 / 1 6 / 2 7 / 3 8 / 4 9 /
```

33. כתוב תוכנית הקולטת שתי מחרוזות בעלות אותו אורך ומדפיסה שילוב שלהן. שילוב מתואר בדוגמה: אם $a_1a_2a_3a_4a_5a_6 - b_1b_2b_3b_4b_5b_6$ הן שתי המחרוזות, אזי המחרוזת החדשה תהיה $a_1b_2a_3b_4a_5b_6$. במילים אחרות, התוכנית תדפיס מחרוזת חדשה שבה במיקומים האי זוגיים מופיעים התווים המתאימים של המחרוזת הראשונה ובמיקומים הזוגיים מופיעים התווים המתאימים של המחרוזת השנייה. אין צורך לבדוק שהאורכים זהים.

דוגמאות הרצה:

```
Please enter something...: abcde
Please enter something...: fghij
Original strings: abcde fghij
Integrated string: agcie
>>>
Please enter something...: abcd
Please enter something...: wxyz
Original strings: abcd wxyz
Integrated string: axcz
```

34. בדרך כלל תרופה מאבדת אחוזים בודדים מייעילותה בכל חודש. תוקף התרופה פג כאשר יעילותה קטנה מ – 40%. כתוב תוכנית הקולטת את אחוז הקטנת היעילות החודשית ומחשבת את משך הזמן עד שתוקף התרופה פג.

דוגמאות הרצה:

```
Please enter weekly effectiveness decrease (in %)...: 5
Number of weeks = 18
>>>
Please enter monthly effectiveness decrease (in %)...: 3.5
Number of months = 26
>>>
Please enter monthly effectiveness decrease (in %)...: 1.5
Number of months = 61
```


35. כתוב תוכנית הבודקת את תקינות לחץ האוויר בצמיגי הרכב. בכל זוג צמיגים (קדמי או אחורי) לחץ האוויר צריך להיות זהה. הלחץ בצמיגים האחוריים צריך להיות גבוה מהלחץ בקדמיים. התוכנית תבקש מהמשתמש את לחץ האוויר בכל ארבעת הצמיגים ובשלב ראשון תחשב שהלחץ בכל זוג זהה. אם לא תודפס הודעת שגיאה מתאימה. אם בכל אחד מהזוגות הלחץ זהה, התוכנית תבדוק שהלחץ בצמיגים הקדמיים נמוך מזה של האחוריים

דוגמאות הרצה:

```
Please enter front left tire pressure: 34
Please enter front right tire pressure: 37
Please enter rear left tire pressure: 43
Please enter rear right tire pressure: 43
Problem -- unequal front pressure!
It is not safe to drive this way!
```

>>>

```
Please enter front left tire pressure: 38
Please enter front right tire pressure: 38
Please enter rear left tire pressure: 44
Please enter rear right tire pressure: 41
Problem -- unequal rear pressure!
It is not safe to drive this way!
```

>>>

```
Please enter front left tire pressure: 34
Please enter front right tire pressure: 37
Please enter rear left tire pressure: 46
Please enter rear right tire pressure: 42
Problem -- unequal front pressure!
Problem -- unequal rear pressure!
It is not safe to drive this way!
```

>>>

```
Please enter front left tire pressure: 44
Please enter front right tire pressure: 44
Please enter rear left tire pressure: 42
Please enter rear right tire pressure: 42
Problem -- rear pressure is not higher!
```

>>>

```
Please enter front left tire pressure: 36
Please enter front right tire pressure: 36
Please enter rear left tire pressure: 43
Please enter rear right tire pressure: 43
Good pressure – Bon Voyage
```

>>>

36. שנה את התוכנית הקודמת, כך שלחצי האוויר בשני צמיגים יחשבו שונים, רק אם הפער ביניהם גדול משלוש יחידות.

דוגמאות הרצה:

```
Please enter front left tire pressure: 34
Please enter front right tire pressure: 37
Please enter rear left tire pressure: 44
Please enter rear right tire pressure: 46
Good pressure - Bon Voyage
```

>>>

```
Please enter front left tire pressure: 34
Please enter front right tire pressure: 38
Please enter rear left tire pressure: 47
Please enter rear right tire pressure: 42
```

תכנות בסיסי

Problem -- unequal front pressure!
Problem -- unequal rear pressure!
It is not safe to drive this way!

37. כתוב תוכנית המבקשת מהמשתמש שני מספרים המייצגים תחום ומדפיסה את כל המספרים שבתחום (כולל הקצוות). לאחר מכן מדפיסה את המספרים החל מהראשון בקפיצות של שניים, בקפיצות של שלושה, ארבעה וחמישה

דוגמאות הרצה:

```
Please enter first number: 1
Please enter last number: 15
1 | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
2 | 1 3 5 7 9 11 13 15
3 | 1 4 7 10 13
4 | 1 5 9 13
5 | 1 6 11
>>>
Please enter first number: 4
Please enter last number: 28
1 | 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
2 | 4 6 8 10 12 14 16 18 20 22 24 26 28
3 | 4 7 10 13 16 19 22 25 28
4 | 4 8 12 16 20 24 28
5 | 4 9 14 19 24
```

38. כתוב תוכנית המבקשת מהמשתמש מחרוזת ומדפיסה אותה ולאחר מכן, בלולאה מדפיסה את המחרוזת, כאשר בכל פעם מורידים את התו הראשון שלה.

דוגמת הרצה:

```
Please enter string: abcdefg
abcdefg
bcdefg
cdefg
defg
efg
fg
g
```

39. שנה את התוכנית, כך שהמחרוזת תודפס ובכל מחזור התו האחרון שלה יושמט.

דוגמת הרצה:

```
Please enter first number: 12345
12345
1234
123
12
1
```

40. כתוב תוכנית המבקשת מהמשתמש שתי מחרוזות ומדפיסה אותן, כאשר ביניהן היא מדפיסה נקודות, כך שסך התווים בשורה הכוללת את שתי המחרוזות והנקודות הוא 30. אם מספר התווים בשתי המחרוזות גדול מ-30, התוכנית תדפיס הודעת שגיאה

דוגמאות הרצה:

```
Please enter first string: 1234567890
Please enter second string: 1234567890
```

תכנות בסיסי

```
1234567890.....1234567890
>>>
Please enter first string: 123456789012345
Please enter second string: 123456789012345
123456789012345123456789012345
>>>
Please enter first string: 1234567890123456
Please enter second string: 123456789012345
Sorry, strings too long
```

41. כתוב תוכנית הקולטת מספר המייצג צלע של ריבוע המורכב מכוכביות. התוכנית תדפיס את הריבוע, כאשר בין כל שתי כוכביות יש רווח אחד.

דוגמאות הרצה:

```
Please enter square size: 7
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
>>>
Please enter square size: 3
* * *
* * *
* * *
```

42. שנה את התוכנית הקודמת כך שבמקום שתדפיס ריבוע מלא היא תדפיס רק את שלד הריבוע

דוגמת הרצה:

```
Please enter square size: 8
* * * * * * * *
*               *
*               *
*               *
*               *
*               *
*               *
* * * * * * * *
* * * * * * * *
```

43. כתוב תוכנית הקולטת מספר המייצג צלע של ריבוע המורכב מכוכביות. התוכנית תדפיס את חצי הריבוע העליון (מעל האלכסון), כאשר בין כל שתי כוכביות יש רווח אחד.

דוגמאות הרצה:

```
Please enter square size: 7
* * * * *
* * * * *
* * * * *
* * * *
* * *
* *
*
>>>
Please enter square size: 4
* * * *
```

תכנות בסיסי

```
* * *  
* *  
*
```

44. שנה את התוכנית הקודמת כך שתדפיס את חצי הריבוע העליון השני.

דוגמאות הרצה:

Please enter square size: 6

```
* * * * *  
* * * * *  
  * * * *  
    * * *  
      * *  
        *  
          *
```

Please enter square size: 4

```
* * * *  
* * *  
* *  
*  
*
```

45. כתוב תוכנית הקולטת מספר n מהמשתמש ובצורה מחזורית מחשבת ממנו מספר חדש. בכל מחזור, אם n זוגי, המספר החדש מחושב ע"י $n/2$. אם n אי זוגי, המספר החדש מחושב ע"י $3n+1$. התוכנית תדפיס את מספר המחזורים הנדרש עד אשר התוצאה המתקבלת היא 1 ובנוסף את הערך המרבי ש- n הגיע אליו

דוגמאות הרצה:

Please enter number: 55

112 cycles were needed for 55. The maximum value was: 9232

>>>

Please enter number: 12

9 cycles were needed for 12. The maximum value was: 16

46. שנה את התוכנית הקודמת, כך שתקבל מהמשתמש שני מספרים המייצגים תחום ועבור כל אחד מהמספרים התחום (כולל הקצוות) תחשב את ממספר המחזורים הדרוש כדי להגיע לערך 1. התוכנית תדפיס את המספר (בתחום) עבורו מספר המחזורים הוא הגדול ביותר

דוגמאות הרצה:

Please enter first number: 1

Please enter first number: 100000

Maximum number of cycles was needed for 77031.

>>>

Please enter first number: 55

Please enter first number: 555

Maximum number of cycles was needed for 327.

>>>

Please enter first number: 111111

Please enter first number: 333333

Maximum number of cycles was needed for 230631.

תכנות בסיסי

47. כתוב תוכנית המקבלת ציון כנהוג בארץ (בתחום 0-100) ומתרגמת אותו לציון כנהוג בחו"ל. נוסחת ההמרה מוגדרת בטבלה

ציון	תנאי
A	$\text{ציון} \geq 90$
B	$90 > \text{ציון} \geq 80$
C	$80 > \text{ציון} \geq 70$
D	$70 > \text{ציון} \geq 60$
F	$\text{ציון} > 60$

דוגמאות הרצה:

```
Please enter your grade: 66
Your grade is D
>>>
Please enter your grade: 90
Your grade is A -- excellent!
>>>
Please enter your grade: 59
You failed better luck next time
>>>
Please enter your grade: 75
Your grade is C
>>>
Please enter your grade: 100
Your grade is A -- excellent!
```

48. כתוב תוכנית ה"מזיזה" מחרוזת. התוכנית תבקש מהמשתמש להקליד מחרוזת ואחריה מספר n. התוכנית תדפיס את המחרוזת, כאשר התו הראשון הוא זה הנמצא במיקום n. לאחר התו האחרון יודפס התו הראשון ועד למיקום n. אם n גדול מאורך המחרוזת, תודפס הודעת שגיאה.

דוגמאות הרצה:

```
Please enter a string: abcdefg
Please enter position: 12
Error - position greater than string length
>>>
Please enter a string: abcdefg
Please enter position: 3
cdefgab
>>>
Please enter a string: 123456789
Please enter position: 6
678912345
```

49. כתוב תוכנית הקולטת מספר בתחום 0-100 ומדפיסה "בינגו" אם הוא כולל את הספרה 7, או שהוא מתחלק ב-7.

דוגמאות הרצה:

```
Please enter a number: 57
Bingo!
>>>
Please enter a number: 84
Bingo!
```

תכנות בסיסי

```
>>>
Please enter a number: 100
>>>
Please enter a number: 13
>>>
Please enter a number: 101
Illegal number - too large
>>>
Please enter a number: 98
Bingo!
```

50. כתוב תוכנית הקולטת מחרוזת המורכבת מהערכים 1,2,x. המחרוזת מייצגת תוצאות של משחקי כדורגל, כאשר 1 מייצג ניצחון לקבוצה הביתית, 2 מייצג הפסד לקבוצה הביתית ואילו x מייצג תיקו. התוכנית תדפיס את מספר המשחקים שבהם היה ניצחון ביתי, מספר המשחקים בהם היה הפסד ביתי ומספר המשחקים בהם התוצאה הייתה תיקו. אם אחד הערכים אינו 1,2,x, התוכנית תדפיס הודעת שגיאה.

דוגמאות הרצה:

```
Please enter a string: xxxxxxxxxxxxxxxxx
Number of wins: 0
Number of loses: 0
Number of draws: 16
>>>
Please enter a string: 12x12x12x12x12xx
Number of wins: 5
Number of loses: 5
Number of draws: 6
>>>
Please enter a string: 1212121212121212
Number of wins: 8
Number of loses: 8
Number of draws: 0
>>>
Please enter a string: 12x12c12x12x1122
Illegal score at location – 6
```

51. כתוב תוכנית המדפיסה את כל המספרים בעלי שלוש ספרות השווים לסכום הספרות שלהם בחזקה שלישית

דוגמת הרצה:

```
153
370
371
407
```

52. נדבן ידוע החליט לתרום מכספו לנזקקים. כדי שלא לפגוע בחלשים, אשר ברוב המקרים אינם הראשונים בתור, החליט לתרום לכל נזקק \$100 יותר מאשר לנזקק שלפניו בתור. במילים אחרות, הראשון יקבל \$100, השני \$200, השלישי \$300 וכן הלאה. כתוב תוכנית אשר קולטת את סכום התרומה ומדפיסה כמה נזקקים ייהנו ממנה (כולל האחרון שיקבל את הסכום שנותר, אולי פחות ממה שמגיע לו לפי החישוב)

דוגמאות הרצה:

```
Please enter total sum: 500
3 people will enjoy the $500
>>>
```

תכנות בסיסי

Please enter total sum: 1000
4 people will enjoy the \$1000
>>>
Please enter total sum: 100000
45 people will enjoy the \$100000
>>>
Please enter total sum: 1000000
141 people will enjoy the \$1000000

53. בדיאטה מסוימת אדם מאבד אחוז קבוע ממשקלו בכל שבוע. כתוב תוכנית הקולטת שלושה מספרים. הראשון מייצג את משקלו של אדם, השני מייצג את המשקל הרצוי והשלישי את אחוז הירידה השבועי. התוכנית תחשב את משך הזמן הדרוש להגיע למשקל הרצוי.

דוגמאות הרצה:

Please enter original weight: 100
Please enter target weight: 70
Please enter weekly percent: 1
It will take: 36 weeks
>>>
Please enter original weight: 110
Please enter target weight: 85
Please enter weekly percent: 2
It will take: 13 weeks
>>>
Please enter original weight: 110
Please enter target weight: 109
Please enter weekly percent: 1
It will take: 1 weeks

54. כתוב תוכנית הקולטת מספר בן שלוש ספרות. אם כל שלוש הספרות זהות היא תדפיס Equal ואם לא, תדפיס Not equal. יש לוודא שהמספר שהוכנס מכיל 3 ספרות.

דוגמאות הרצה:

Please enter a 3 digit number: 99
3 digit number - please...
Please enter a 3 digit number: 9999
3 digit number - please...
Please enter a 3 digit number: 12
3 digit number - please...
Please enter a 3 digit number: 567
Not equal
>>>
Please enter a 3 digit number: 777
Equal
>>>

55. כתוב תוכנית המדפיסה את כל המספרים בעלי שלוש ספרות המקיימים את התנאי: סכום ספרת המאות וספרת האחדות שווה לספרת העשרות בריבוע. אם המספר הוא $n_1n_2n_3$, אזי הוא מודפס אם הוא מקיים: $n_1 + n_3 = n_2^2$

דוגמת הרצה:

110 123 138 222 237 321 336 420 435 534 633 732 749 831 848 930 947
>>>

תכנות בסיסי

56. כתוב תוכנית הקולטת שני מספרים (n_1, n_2) בעלי ארבע ספרות, כאשר המספר השני גדול מהראשון. התוכנית תדפיס את כל המספרים בתחום $[n_1, n_2]$ (כולל n_2) המתחלקים בסכום אבריהם.

דוגמאות הרצה:

```
Please enter starting number [4 digits]: 123
Wrong answer - 4 digits please...
Please enter starting number [4 digits]: 1234
Please enter ending number [4 digits]: 10000
Wrong answer - 4 digits please...
Please enter ending number [4 digits]: 1260
1236 1242 1251 1260
>>>
Please enter starting number [4 digits]: 0
Wrong answer - 4 digit please...
Please enter starting number [4 digits]: 1500
Please enter ending number [4 digits]: 1600
1500 1503 1512 1520 1521 1524 1526 1530 1534 1540 1545 1547 1548 1558 1560
1566 1584 1590 1596
```

57. כתוב פונקציה המקבלת שתי רשימות מספריות ומחזירה רשימה אחת מאוחדת וממוינת, כאשר איברים כפולים הושמטו ממנה.

דוגמאות הרצה:

```
list1 = [1, 3, 5, 6, 7, 8, 10, 12, 15]
list2 = [1, 2, 6, 9, 10, 19]
merged list = [1, 2, 3, 5, 6, 7, 8, 9, 10, 12, 15, 19]
>>>
list1 = [12, 3, 8, 4, 23]
list2 = [23, 12, 4, 3, 8, 17, 1, 7]
merged list = [1, 3, 4, 7, 8, 12, 17, 23]
```

58. כתוב פונקציה המקבלת פרמטר שהוא מספר שלם (n) ומדפיסה את הסכום הסדרה הנתונה ע"י $1/n, 1/2, 1/3, \dots, 1/n$ התוצאה תודפס בדיוק של שתי ספרות. שים לה המספר יכול להיות שלילי.

דוגמאות הרצה:

```
For n = 1 the sum is: 1.00
>>>
For n = 2 the sum is: 1.50
>>>
For n = -2 the sum is: -1.50
>>>
For n = -1000 the sum is: -7.49
>>>
For n = 10000 the sum is: 9.79
```

59. כתוב פונקציה המקבלת שני פרמטרים מספריים (n, m) , כאשר $m < n$. הפונקציה תבדוק שאכן $m < n$ ואם לא תדפיס הודעת שגיאה. הפונקציה תדפיס שורות של כוכביות, כאשר בשורה הראשונה m כוכביות, בשורה שאחריה $n+1$ כוכביות עד לשורה המכילה m כוכביות. לאחר מכן שורה שבה $m-1$ כוכביות, $m-2$ כוכביות וכן הלאה עד לשורה בעלת n כוכביות.

דוגמאות הרצה:

```
Min = 2 Max = 1
Min is larger than Max
```


תכנות בסיסי

```
>>>
Min = 2   Max = 5
**
***
****
*****
****
***
**

>>>
Min = 2   Max = 3
**
***
**
```

60. כתוב פונקציה המקבלת פרמטר שהוא מספר שלם בעל מספר לא ידוע של ספרות ומחשבת את סכום כל הספרות האי זוגיות שבמספר.

דוגמאות הרצה:

```
The number is: 1234567890
The Odd digits sum is: 25
>>>
The number is: 1111111111
The Odd digits sum is: 10
>>>
The number is: 24682468
The Odd digits sum is: 0
>>>
The number is: 112233445566778899
The Odd digits sum is: 50
```

61. כתוב פונקציה המקבלת פרמטר שהוא מחרוזת בעלת אותיות קטנות בלבד. ומקודדת אותה בקידוד קיסר. קידוד (או הצפנת) קיסר מחליפה כל אות במחרוזת, באות הנמצאת מספר קבוע של מקומות אחריה. הפונקציה שעליך לכתוב מותאמת להחלפה של כל אות באות שנמצאת במרחק של שלושה מקומות. לכן, a תהפוך להיות d , b תהפוך ל e וכן הלאה. שלוש האותיות האחרונות, x,y,z הופכות להיות a,b,c בהתאמה. אפשר להניח שהקלט תקין.

דוגמאות הרצה:

```
The original string is:      abcdefghijklmnopqrstuvwxyz
The Caesar coded string is:  defghijklmnopqrstuvwxyzabc
>>>
The original string is:      computer
The Caesar coded string is:  frpsxwhu
>>>
The original string is:      programming
The Caesar coded string is:  surjudpplqj
```

62. כתוב פונקציה המקבלת פרמטר מספרי (n) ומחזירה את האיבר n בסדרת טרי-בונצ'י. האיברים בסדרה הם: $T_0 = 0, T_1 = 1, T_2 = 1, T_3 = T_0 + T_1 + T_2, \dots, T_n = T_{n-1} + T_{n-2} + T_{n-3}$.

דוגמאות הרצה:

```
Number 5 in the Tribonacci numbers is 7
>>>
Number 10 in the Tribonacci numbers is 149
>>>
Number 100 in the Tribonacci numbers is 98079530178586034536500564
```

63. כתוב פונקציה המקבלת שני פרמטרים מספריים ומחשבת הערכה של השורש הריבועי של המספר הראשון. נוסחת החישוב היא: בהינתן מספר חיובי כלשהו (x) שהינו הערכה לשורש בריבועי של y , אזי המספר $(x+y/x)/2$ מהווה הערכה מדויקת יותר. יש להמשיך בחישוב עד אשר ההפרש בין שתי איטרציות של ההערכה קטן מערך קבוע (הפרמטר השני של הפונקציה).

דוגמאות הרצה:

```
The approximation for 7 SQRT is 2.75 (with accuracy of 2)
>>>
The approximation for 7 SQRT is 2.6457520483808037 (with accuracy of 0.1)
>>>
The approximation for 7 SQRT is 2.6457513110646933 (with accuracy of 1e-05)
>>>
The approximation for 9 SQRT is 3.25 (with accuracy of 2)
>>>
The approximation for 9 SQRT is 3.0000000000393214 (with accuracy of 0.001)
>>>
The approximation for 9 SQRT is 3.0 (with accuracy of 1e-10)
```

64. כתוב פונקציה המקבלת פרמטר מספרי (n) ומחשבת את כל המספרים הראשוניים בתחום $n-1$. יש להשתמש בשיטת המסננת של ארטוסטנס. יש להגדיר רשימה של כל המספרים בתחום ולמחוק ממנה את כל הכפולות של 2,3,4 וכן הלאה. המספרים שנותרו ברשימה הם מספרים ראשוניים. הערה: את הרשימה יש לממש בעזרת ערכים בוליאניים.

דוגמאות הרצה:

```
List of primes in the range 1 - 10
1      2      3      5      7
>>>
List of primes in the range 1 - 100
1      2      3      5      7      11      13      17      19      23      29
31     37     41     43     47     53     59     61     67     71     73
79     83     89     97
>>>
List of primes in the range 1 - 500
1      2      3      5      7      11      13      17      19      23      29
31     37     41     43     47     53     59     61     67     71     73
79     83     89     97     101    103    107    109    113    127    131
137    139    149    151    157    163    167    173    179    181    191
193    197    199    211    223    227    229    233    239    241    251
257    263    269    271    277    281    283    293    307    311    313
317    331    337    347    349    353    359    367    373    379    383
389    397    401    409    419    421    431    433    439    443    449
457    461    463    467    479    487    491    499
```

65. כתוב פונקציה המקבלת פרמטר מסוג מחרוזת והיא מדפיסה אותה במעין משולש. שורה ראשונה מכילה את האות הראשונה, בשורה השנייה האות השנייה מודפסת פעמיים, בשורה השלישית האות השלישית מודפסת שלוש פעמים וכן הלאה. בין כל שתי אותיות זהות יודפסו חמישה רווחים.

דוגמאות הרצה:

```
Python
=====
P
y      y
t      t      t
```

תכנות בסיסי

```

h      h      h      h
o      o      o      o      o
n      n      n      n      n      n
>>>
PROGRAMMING
=====
P
R      R
O      O      O
G      G      G      G
R      R      R      R      R
A      A      A      A      A      A
M      M      M      M      M      M      M
M      M      M      M      M      M      M      M
I      I      I      I      I      I      I      I      I
N      N      N      N      N      N      N      N      N      N      N
G      G      G      G      G      G      G      G      G      G      G

```

66. כתוב פונקציה המקבלת פרמטר שהוא רשימה של מספרים. הפונקציה תחזיר את ההפרש המינימאלי שבין כל זוג עוקב של איברים וכן את המיקום של האיבר הראשון בזוג.

דוגמאות הרצה:

```

For the list [1, 4, 2, 7, 12, 9, 13, 14]
The smallest distance is 1
And the location is 6
>>>
For the list [3, 5, 2, 8, 12, 9, 11, 14]
The smallest distance is 2
And the location is 0
>>>
For the list [-7, -3, 2, -4, -5, 3, 5, 8, 12]
The smallest distance is 1
And the location is 3

```

67. כתוב פונקציה המקבלת פרמטר שהוא מחרוזת טקסטואלית הכוללת ספרות בלבד. הפונקציה תחשב ותחזיר את סכום הספרות המרכיבות את המחרוזת.

דוגמאות הרצה:

```

For the list 123456789
The sum is 45
>>>
For the list 2020202020
The sum is 10

```

68. כתוב פונקציה המקבלת פרמטר שהוא מחרוזת טקסטואלית הכוללת ספרות בלבד. הפונקציה מחזירה רשימה שבה כל איבר הוא סכום שלושה איברים במחרוזת המקורית. את החלוקה לשלוש יש להתחיל בצד ימין.

דוגמאות הרצה:

```

For the list 2020202020
The new list is [2, 2, 4, 2]
>>>
For the list 12345678
The new list is [3, 12, 21]
>>>

```

תכנות בסיסי

```
For the list 123456789
The new list is [6, 15, 24]
>>>
For the list 1234567
The new list is [1, 9, 18]
```

69. כתוב פונקציה המקבלת שלושה פרמטרים:

- a. הראשון מספר בוליאני אשר מגדיר אם עיר מסוימת היא עיר בירה,
- b. השני מספרי, לציון מספר התושבים בעיר
- c. השלישי מספרי לציון הארנונה המשולמת עבור כל אזרח

הפונקציה תחזיר אמת אם העיר היא מטרופוליס ושקר עם לא. כדי שער תהיה מטרופוליס היא חייבת להיות עיר בירה ובעלת למעלה מ- 100,000 תושבים, או לחילופין כל עיר שבה למעלה מ- 200,000 תושבים ובלבד שההכנסות שלה גבוהות מ- 800,000,000. כתוב את הפונקציה בצורה כזאת שהיא משתמשת בתנאי בוליאני יחיד.

דוגמאות הרצה:

```
A capital city with 100,000 citizens and income of 300,000,000
is NOT a metropolis city
>>>
A capital city with 100,001 citizens and income of 300,003,000
is a metropolis city
>>>
A capital city with 100,000 citizens and income of 900,000,000
is NOT a metropolis city
>>>
A city with 100,000 citizens and income of 900,000,000
is NOT a metropolis city
>>>
A city with 200,000 citizens and income of 1,800,000,000
is NOT a metropolis city
>>>
A city with 200,001 citizens and income of 800,004,000
is a metropolis city
```

70. כתוב פונקציה המקבלת פרמטר מספרי שלם (n) ומדפיסה סדרה של מספרים שלמים. אם המספר אי-זוגי, המספר הבא מחושב ע"י $n+1$. אם המספר זוגי, המספר הבא מחושב ע"י $n/2$. הפונקציה תדפיס את כל המספרים בסדרה, עד שהיא מגיעה ל- $n=1$.

דוגמאות הרצה:

```
For 3 the numbers are:
10 5 16 8 4 2 1
>>>
For 5 the numbers are:
16 8 4 2 1
>>>
For 305 the numbers are:
916 458 229 688 344 172 86 43 130 65 196 98 49 148 74 37 112 56 28 14 7 22 11 34
17 52 26 13 40 20 10 5 16 8 4 2 1
```

71. כתוב תכנית המדמה מחשבון פשוט. התכנית קולטת מהמשתמש שלושה פרמטרים (מספר, סימן ומספר), כאשר הסימן יכול להיות אחד מארבעה סימנים: +, -, *, /. לאחר מכן התכנית תבצע את הפעולה שהתבקשה. למשל אם הוכנסו הפרמטרים 13, + ו- 12 התוצאה שתתקבל היא 25. לעומת זאת עבור $15 * 4$ התוצאה שתתקבל היא 60 אין צורך לבדוק את חוקיות הסימן.

דוגמאות הרצה:

```
Enter first number: 5
Enter operation: +
Enter second number: 12
5+12= 17
>>>
Enter first number: 34
Enter operation: /
Enter second number: 2
34/2= 17.0
>>>
Enter first number: 12
Enter operation: $
Enter second number: 6
12$6= undefined
```

72. שנה את התכנית הקדמת, כך שהמקום שתשתמש בשלוש פקודות קלט (לקריאת שלושה הפרמטרים), התכנית תשמש בפקודת קלט אחת הכוללת מחרוזת של שלושה הפרמטרים. פרט לשינוי הקלט, התכנית תתנהג כדיוק כמו התכנית הקודמת.

דוגמאות הרצה:

```
Enter string: 12,+,13
12+13= 25
>>>
Enter string: 37,-,25
37-25= 12
>>>
Enter string: 33,%,11
33%11= undefined
```

73. כתוב תכנית הקולטת שני מספרים ומחשבת את סכום המספרים שביניהם. כל אחד מהמספרים יכול להיות הקטן ביניהם

דוגמאות הרצה:

```
Enter number: 5
Enter number: 8
Sum of numbers in range 5-8= 13
>>>
Enter number: 1
Enter number: 12
Sum of numbers in range 1-12 = 65
>>>
Enter number: 8
Enter number: 3
Sum of numbers in range 3-8 = 22
>>>
Enter number: 6
Enter number: 7
Sum of numbers in range 6-7 = 0
>>>
```

תכנות בסיסי

```
Enter number: 8
Enter number: 8
Sum of numbers in range 8-8 = 0
>>>
```

74. כתוב תכנית המדפיסה (בשורה אחת) את כל המספרים בתחום 1-35 (כולל) שאינם זוגיים, או שלא כוללים את הספרה 2.

דוגמת הרצה:

```
>>>
1 3 5 7 9 11 13 15 17 19 31 33 35
>>>
```

75. כתוב תכנית הקולטת מחרוזת ומדפיסה מחרוזת חדשה שבה כל אותיות a הוחלפו באותיות b, כל האותיות b הוחלפו באותיות a וכל שאר האותיות נשארו ללא שינוי.

דוגמאות הרצה:

```
Enter string: 12aaabbbdsf

Original - 12aaabbbdsf
Modified - 12bbbbaadsf
>>>
Enter string: 1234567

Original - 1234567
Modified - 1234567
>>>
Enter string: aaabbbababab

Original - aaabbbababab
Modified - bbbbaabababa
>>>
```

76. כתוב תכנית הקוראת מספר n ומדפיסה "משולש מספרים". במשולש מספרים הכוונה שבשורה הראשונה מודפס המספר 1 פעם אחת, בשורה מתחתיו המספר 2 מודפס פעמיים וכן הלאה עד המספר n שמודפס n פעמים.

דוגמאות הרצה:

```
Enter a number: 5
1
22
333
4444
55555
>>>
Enter a number: 1
1
>>>
Enter a number: -15
>>>
```

תכנות בסיסי

```
>>>
Enter a number: 9
1
22
333
4444
55555
666666
7777777
88888888
999999999
>>>
```

77. שנה את התכנית הקודמת, כך שאת משולש המספרים תדפיס בצד ימין ולא בצד שמאל

דוגמאות הרצה:

```
Enter a number: 9
      1
     22
    333
   4444
  55555
 666666
7777777
88888888
999999999
>>>
Enter a number: 5
      1
     22
    333
   4444
  55555
>>>
```

78. שנה את התכנית פעם נוספת, כך שתדפיס משולש מספרים כפול. בשורה הראשונה יודפס המספר 1 פעמיים (באמצע), בשורה תחתיה יודפס המספר 2 ארבע פעמים וכן הלאה עד השורה האחרונה בה יודפס המספר $n(2^n)$ פעמים

דוגמאות הרצה:

```
Enter a number: 4
     11
    2222
   333333
  44444444
>>>
Enter a number: 2
     11
    2222
Enter a number: 9
```

תכנות בסיסי

```
11
2222
333333
44444444
5555555555
666666666666
77777777777777
8888888888888888
9999999999999999
>>>
Enter a number: 0
>>>
```

79. שנה את התכנית פעם אחרונה, כך שתדפיס מעוין מספרים.

דוגמאות הרצה:

```
Enter a number: 6
11
2222
333333
44444444
5555555555
666666666666
666666666666
5555555555
44444444
333333
2222
11
>>>
Enter a number: 8
11
2222
333333
44444444
5555555555
666666666666
77777777777777
8888888888888888
8888888888888888
77777777777777
666666666666
5555555555
44444444
333333
2222
11
>>>
```

80. כתוב תכנית הקולטת מספר מהמשתמש ומדפיסה ריבוי ריק של כוכביות, כאשר צלע הריבוע היא המספר שהוכנס. אפשר להניח שהקלט הוא מספרי וגדול מ-2.

דוגמאות הרצה:

```
>>>
Enter a number: 9
*****
*           *
*           *
*           *
*           *
*           *
*           *
*           *
*****

>>>
Enter a number: 3
***
* *
***

>>>
Enter a number: 14
*****
*           *
*           *
*           *
*           *
*           *
*           *
*           *
*           *
*           *
*           *
*           *
*           *
*****

>>>
```

81. כתוב תכנית הקולטת שתי מחרוזות באורך זהה. אין צורך לבדוק את האורכים ואפשר להניח שהם זהים. התכנית תאחד את המחרוזות ע"י יצירה של מחרוזת חדשה שבה איברים משתי המחרוזות לפי הסדר. האיבר הראשון במחרוזת החדשה הוא האיבר הראשון של המחרוזת הראשונה, האיבר השני של המחרוזת החדשה הוא האיבר הראשון של המחרוזת השנייה, האיבר השלישי של המחרוזת החדשה הוא האיבר השני של המחרוזת הראשונה וכך הלאה. בנוסף התוכנית תחלק את איברי המחרוזת החדשה לקבוצות של שלושה איברים ע"י הוספה של הסימן "/" ביניהם.

דוגמאות הרצה:

```
Enter string1: 11111
Enter string1: 22222
Combined string: /121/212/121/2

>>>
Enter string1: abcdefghijkl
```

תכנות בסיסי

Enter string1: mnopqrstuvwx

Combined string: /amb/nco/dpe/qfr/gsh/tiu/jvk/wlx

82. כתוב תכנית הקולטת תאריך בפורמט dd/mm (שתי ספרות ליום ושתי ספרות לחודש) ומחשבת את מספר הימים שעברו מתחילת השנה. אפשר להתעלם משנים מעוברות ואפשר להניח שהתאריך שהכנס הינו חוקי. רצ"ב טבלת חודשים ומספר הימים בכל אחד מהם

12	11	10	9	8	7	6	5	4	3	2	1	חודש
31	30	31	30	31	31	30	31	30	31	28	31	מס' ימים

דוגמאות הרצה:

```
Enter date (DD/MM): 01/01
01/01 is 0 days past January 1st
>>>
Enter date (DD/MM): 02/01
02/01 is 1 days past January 1st
>>>
Enter date (DD/MM): 15/06
15/06 is 165 days past January 1st
>>>
Enter date (DD/MM): 31/12
31/12 is 364 days past January 1st
>>>
```

83. כתוב תכנית הקולטת מחרוזת מספרית ומחשבת את מס' הספרות שבה, את סכום הספרות שבה ואת מכפלתן

דוגמאות הרצה:

```
Enter a numeric string: 1234
Number - 1234 4 digits
sum = 10 Multiplication = 24
>>>
Enter a numeric string: 123
Number - contains 123 3 digits
sum = 6 , Multiplication = 6
>>>
Enter a numeric string: 123
Number - 123 contains 3 digits
sum = 6 , Multiplication = 6
>>>
Enter a numeric string: 22222222
Number - 22222222 contains 9 digits
sum = 18 , Multiplication = 512
>>>
Enter a numeric string: 102
Number - 102 contains 3 digits
sum = 3 , Multiplication = 0
>>>
Enter a numeric string: 123456789
Number - 123456789 contains 9 digits
```

תכנות בסיסי

```
sum = 45 , Multiplication = 362880
```

```
>>>
```

84. כתוב תכנית הקולטת מחרוזת ומדפיסה מחרוזת שבה כל זוג איברים החליף מקומות- כלומר האיבר הראשון הפך שני והאיבר השני הפך ראשון, האיבר השלישי הפך רביעי והאיבר הרביעי הפך שלישי וכך הלאה

דוגמאות הרצה:

```
Enter a string: 1234
```

```
2143
```

```
>>>
```

```
Enter a string: abcdefgh
```

```
badcfegh
```

```
>>>
```

```
Enter a string: 1234567
```

```
2143657
```

```
>>>
```

85. כתוב תכנית הקולטת מהמשתמש מחרוזות, כאשר כל מחרוזת כוללת שני ערכים מספריים המופרדים בפסיק. הערך הראשון מייצג כמות מוצרים שנרכשו והערך השני מייצג את מחירו של כל מוצר. כאשר מספר המוצרים שנרכשו (הערך הראשון במחרוזת) הוא אפס, הדבר מציין את סוף הקלט. התכנית תדפיס את סך הקנייה, תוסיף את המע"מ (17%) ותחשב את הסכום לתשלום.

דוגמאות הרצה:

```
Enter item: 10,1.5
```

```
Enter item: 10,2.5
```

```
Enter item: 10,1
```

```
Enter item: 0
```

```
Total (no VAT)- 50.00
```

```
VAT - 8.50
```

```
Total Sale - 58.50
```

```
>>>
```

```
Enter item: 100,2.5
```

```
Enter item: 30,15
```

```
Enter item: 10,25
```

```
Enter item: 1,45.5
```

```
Enter item: 1,4.5
```

```
Enter item: 0
```

```
Total (no VAT)- 1000.00
```

```
VAT - 170.00
```

```
Total Sale - 1170.00
```

```
>>>
```

86. כתוב תכנית הקולטת את השכר ברוטו ומחשבת את המיסים שיש לנכות ואת השכר נטו. חוקי החישוב הם כדלקמן:

על 4000 השקלים הראשונים אחוז המס הוא אפס.

על 3500 השקלים הבאים (עד שכר של 7500 ₪) המס הוא 15%

על 2500 השקלים הבאים (עד שכר של 10000 ₪) המס הוא 25%

על 5000 השקלים הנוספים (עד שכר של 15000 ₪) המס הוא 33%

ועל כל שקל מעבר לכך המס הוא 45%.

בנוסף, לאחר שהורד המס, יש לחשב גם תשלום לביטוח לאומי בשיעור של 15%.

דוגמאות הרצה:

```
>>>
Enter salary: 5000
Brutto-      5000.00
Tax-         150.00
Bituach-     727.50
Netto-       4122.50
>>>
Enter salary: 10000
Brutto-     10000.00
Tax-        1150.00
Bituach-    1327.50
Netto-      7522.50
>>>
Enter salary: 15000
Brutto-     15000.00
Tax-        2800.00
Bituach-    1830.00
Netto-     10370.00
>>>
Enter salary: 30000
Brutto-     30000.00
Tax-        9550.00
Bituach-    3067.50
Netto-     17382.50
>>>
```

87. כתוב תכנית הקולטת מספרים ומוצאת את המספר הקטן והמספר הגדול שביניהם. התוכנית תדפיס את מס' המספרים שקראה, המספר הקטן והגדול והממוצע שלהם (כשבר עם שתי ספרות מימין לנקודה העשרונית). התכנית אמורה לקרוא את מספרים בלולאה, עד אשר המשתמש לוחץ על enter ללא מספרים

דוגמאות הרצה:

```
Enter a number (<Enter> to quit) >> 22
Enter a number (<Enter> to quit) >> 33
Enter a number (<Enter> to quit) >> 44
Enter a number (<Enter> to quit) >> 55
Enter a number (<Enter> to quit) >> 66
Enter a number (<Enter> to quit) >>
Count: 5   Min: 22   Max: 66
Avergae: 44.00
>>>
Enter a number (<Enter> to quit) >> 10
Enter a number (<Enter> to quit) >> 11
Enter a number (<Enter> to quit) >> 12
Enter a number (<Enter> to quit) >> 13
Enter a number (<Enter> to quit) >>
Count: 4   Min: 10   Max: 13
Avergae: 11.50
```

>>>

88. כתוב תכנית הקולטת מחרוזת ומדפיסה אותה לאחר שהתו הראשון והאחרון התחלפו ביניהם. כלומר התו הראשון יהיה במקום האחרון והתו האחרון יהיה במקום הראשון. יש לטפל גם במקרים של מחרוזת ריקה ומחרוזת באורך 1.

דוגמאות הרצה:

Enter a string:

>>>

Enter a string: 1

1

>>>

Enter a string: 12

21

>>>

Enter a string: 123

321

>>>

Enter a string: abcdefghijkl

lbcdefghijka

>>>

89. כתוב תכנית הקולטת מחרוזת הכוללת חלק ממשפט ומדפיסה את המספר הממוצע של אותיות במילים שבמשפט.

דוגמאות הרצה:

>>>

In the line: Jack and Jill went up the hill to fetch a pail
of water

The average number of letters per word is: 3.31

>>>

In the line: My bonnie lies over the ocean

The average number of letters per word is: 4.00

>>>

90. כתוב תכנית הקולטת מחרוזת ומספר. התכנית מזיזה את התווים במחרוזת ימינה בצורה מחזורית. פירוש הדבר שהתווים "שנפלו" בצד ימין חוזרים מצד שמאל.

דוגמאות הרצה:

>>>

Enter string: 123456789

Enter shift count: 1

912345678

>>>

Enter string: 123456789

Enter shift count: 5

567891234

>>>

Enter string: 123456789

תכנות בסיסי

```
Enter shift count: 0
123456789
>>>
Enter string: 123456789
Enter shift count: 15
456789123
>>>
```

91. כתוב תכנית הקולטת מספר ומדפיסה את כל המספרים הריבועיים הקטנים מהמספר שהוקלד. המספרים יודפסו בשורה אחת, כאשר הם מיושרים לימין. בכל ושרה יודפסו שבעה מספרים.

דוגמאות הרצה:

```
>>>
Enter a number: 20
      1      4      9
>>>
Enter a number: 50
      1      4      9      16      25
      36
>>>
Enter a number: 1000
      1      4      9      16      25
      36      49      64      81      100
      121      144      169      196      225
      256      289      324      361      400
      441      484      529      576      625
      676      729      784      841      900
>>>
Enter a number: 5000
      1      4      9      16      25
      36      49      64      81      100
      121      144      169      196      225
      256      289      324      361      400
      441      484      529      576      625
      676      729      784      841      900
      961      1024      1089      1156      1225
      1296      1369      1444      1521      1600
      1681      1764      1849      1936      2025
      2116      2209      2304      2401      2500
      2601      2704      2809      2916      3025
      3136      3249      3364      3481      3600
      3721      3844      3969      4096      4225
      4356      4489      4624      4761
>>>
```

92. כתוב תכנית הקולטת שני פרמטרים. פרמטר ראשון מייצג זמן ראשוני בפורמט HH:MM ופרמטר שני מייצג את מספר הדקות שעברו. התכנית תחשב ותדפיס את הזמן החדש ע"י הוספת הדקות לזמן המקורי

דוגמאות הרצה:

תכנות בסיסי

```
>>>
Enter time (HH:MM) - 12:57
Enter duration (in minutes) - 3
New time - 13:0
>>>
Enter time (HH:MM) - 12:57
Enter duration (in minutes) - 3
New time - 13:00
>>>
Enter time (HH:MM) - 00:01
Enter duration (in minutes) - 59
New time - 01:00
>>>
Enter time (HH:MM) - 10:55
Enter duration (in minutes) - 1440
New time - 10:55
>>>
Enter time (HH:MM) - 13:44
Enter duration (in minutes) - 9
New time - 13:53
>>>
Enter time (HH:MM) - 18:45
Enter duration (in minutes) - 303
New time - 23:48
>>>
Enter time (HH:MM) - 20:48
Enter duration (in minutes) - 302
New time - 01:50
>>>
Enter time (HH:MM) - 23:55
Enter duration (in minutes) - 0
New time - 23:55
>>>
```

93. כתוב תכנית הקולטת שלושה פרמטרים שהם מחרוזות. התוכנית תדפיס מחרוזת מאוחדת (המורכבת משרשור שלוש המחרוזות), כאשר סדר השרשור מתבצע לפי אורך המחרוזות.

דוגמאות הרצה:

```
>>>
Enter first string: aa
Enter second string: bbb
Enter third string: cccc
aabbcbccc
>>>
Enter first string: aaaa
Enter second string: bb
Enter third string: ccc
bbcccaaaa
>>>
Enter first string: aaa
```

תכנות בסיסי

```
Enter second string: bbbb
Enter third string:  cc
ccaaabbbb
>>>
```

94. כתוב תכנית הקולטת מחרוזת ומדפיסה מחרוזת חדשה שבה כל איבר מציין את מספר המופעים של האיבר המתאים המחרוזת המקורית. כלומר אם המחרוזת שהוכנסה היא "abacb" המחרוזת שתודפס היא "22212". בגלל שיש שני a במחרוזת המקורית, בכל מקום שהיה בו a תוכנס הספרה 2.

דוגמאות הרצה:

```
>>>
Enter first string:  ababab
                    333333

>>>
Enter first string:  abaccdadab
                    4242224242

>>>
Enter first string:  abcd
                    1111

>>>
Enter first string:  pqqpew
                    3132312

>>>
```

95. כתוב תכנית הקולטת מספר n ומדפיסה את כל המכפלות מ - 1*1

דוגמאות הרצה:

```
>>>
Enter number: 1
                1 * 1 = 1

>>>
Enter number: 2
                1 * 1 = 1
                1 * 2 = 2
                2 * 1 = 2
                2 * 2 = 4

>>>
Enter number: 3
                1 * 1 = 1
                1 * 2 = 2
                1 * 3 = 3
                2 * 1 = 2
                2 * 2 = 4
                2 * 3 = 6
                3 * 1 = 3
                3 * 2 = 6
                3 * 3 = 9

>>>
```


תכנות בסיסי

96. כתוב תכנית הקולטת מספר ומדפיסה את כל המחלקים שלו

דוגמאות הרצה:

```
>>>
Enter number:
1

>>>
Enter number: 15
3 5

>>>
Enter number: 99
3 9 11 33

>>>
Enter number: 100
2 4 5 10 20 25 50

>>>
```

97. כתוב תכנית הקולטת שני מספרים בסיס וגבול ומדפיסה את כל החזקות של הבסיס הקטנות מהגבול.

דוגמאות הרצה:

```
>>>
Enter base: 2
Enter limit: 1000
1 2 4 8 16 32 64 128 256 512

>>>
Enter base: 3
Enter limit: 500
1 3 9 27 81 243

>>>
Enter base: 7
Enter limit: 3000
1 7 49 343 2401

>>>
Enter base: 10
Enter limit: 1000000
1 10 100 1000 10000 100000

>>>
```

98. כתוב תכנית הקולטת שתי חרוזות ומדפיסה את כל הצירופים האפשריים של מחרוזות בעלות שני תווים, כאשר התו הראשון מהמחרוזת הראשונה והשני מהמחרוזת השנייה

דוגמאות הרצה:

```
>>>
Enter first string: 123
Enter second string: 456
14 15 16 24 25 26 34 35 36

>>>
Enter first string: 1234
```

תכנות בסיסי

```
Enter second string: 5678
15 16 17 18 25 26 27 28 35 36 37 38 45 46 47 48
>>>
Enter first string: abcd
Enter second string: wxyz
aw ax ay az bw bx by bz cw cx cy cz dw dx dy dz
>>>
```

99. כתוב תכנית הקולטת מחרוזת ומספר n המייצג "חלון" מקסימלי. התכנית תדפיס מחרוזת המהוות מעין "חלון" של תווים מתוך המחרוזת. בשלב הראשון יודפסו מחרוזות הכוללות שני תווים (תו ראשון ושני, לאחר מכן, תו שני ושלישי וכן הלאה). לאחר מכן יודפסו מחרוזות הכוללות שלושה תווים וכן הלאה עד למחרוזות הכוללות n תווים

דוגמאות הרצה:

```
>>>
Enter first string: abcdef
Enter max window: 4
ab bc cd de ef
abc bcd cde def
abcd bcde cdef
>>> Enter first string: abcdefghijklm
Enter max window: 5
ab bc cd de ef fg gh hi ij jk kl lm
abc bcd cde def efg fgh ghi hij ijk jkl klm
abcd bcde cdef defg efgh fghi ghij hijk ijkl jklm
abcde bcdef cdefg defgh efghi fghij ghijk hijkl ijklm
>>>
```

100. כתוב תכנית הקולטת מחרוזת הכוללת מספרים מופרדים בפסיקים. התכנית תחשב את ממוצע המספרים שבמחרוזת ותדפיס מחרוזת חדשה שבה במקום כל איבר מופיע ההפרש שבין האיבר המקורי לממוצע שחושב. האיברים במחרוזת במקורית ובמחרוזת החדשה שחושבה, הינם מספרים שלמים.

דוגמאות הרצה:

```
>>>
Enter string of numbers: 1,2,3,4,5,6,7,8,9
-4,-3,-2,-1,0,1,2,3,4
>>>
Enter string of numbers: 11,22,33,44,55,66,77
-33,-22,-11,0,11,22,33
>>>
Enter string of numbers: 67,92,85,94,75,82,96,88,53
-14,10,3,12,-6,0,14,6,-28
>>>
```

תשובות לתרגילים

תשובות

תשובות

שיטות מספריות

1. הפוך מבסיס דצימלי (עשרוני) לבינארי

1. 1011111101
2. 10011010010
3. 1011001101011
4. 1111101001
5. 10001111010101
6. 10000000000
7. 11111111111
8. 11110000000101
9. 1111100111
10. 1111110010

2. הפוך מבסיס בינארי לדצימלי (עשרוני)

1. 99
2. 199
3. 299
4. 35
5. 111
6. 1234
7. 88
8. 175
9. 144
10. 288

3. הפוך מדצימלי לבסיס המתאים

1. 1297
2. 11227
3. E0301
4. 3401401
5. 1311
6. 20223
7. 1053
8. 22B
9. 2160
10. 14640
11. 12422311
12. 21101
13. 113420
14. 622520
15. 40302

תכנות בסיסי

4. הפוך מבסיס לבסיס על פי הרשום

1. 141
2. 243
3. 208
4. 80
3. 441
4. 222222
5. 4436
6. 4222
7. 734
8. 3530

5. השלם את הטבלה

בינארי	בסיס 3	בסיס 5	בסיס 8	עשרוני	הקסהדצימלי
1100101	10202	401	145	101	65
11011011	22010	1334	333	219	DB
1111100	11121	444	174	124	7C
11111111	100110	2010	377	255	FF
100000101	100200	2021	405	261	105
10101010	20022	1140	252	170	AA
11001100	21120	1304	314	204	CC
11000111	21101	1244	307	199	C7
1001001	2201	243	111	73	49

6. המספרים הבאים הינם מספרים עשרוניים, בטא אותם בכל הבסיסים שלפניהם (2-9).

1. $01100100_2 = 010201_3 = 01210_4 = 0400_5 = 0244_6 = 0202_7 = 0144_8 = 0121_9$
2. $01000000_2 = 02101_3 = 01000_4 = 0244_5 = 0144_6 = 0121_7 = 0100_8 = 071_9$
3. $01001101_2 = 02212_3 = 01031_4 = 0302_5 = 0205_6 = 0140_7 = 0115_8 = 085_9$
4. $011001_2 = 0221_3 = 0121_4 = 0100_5 = 041_6 = 034_7 = 031_8 = 027_9$
5. $01010011_2 = 010002_3 = 01103_4 = 0313_5 = 0215_6 = 0146_7 = 0123_8 = 0102_9$

7. תרגם את המספרים (נקודה קבועה)

1. 14.6875
2. 9.375
3. .0625
4. 15.9375
5. 0.015625
6. 0011.1001
7. 1001.1111
8. 1011.100011001100....
9. 0.1011001100....
10. 1110.0001

[חזור לתוכן](#)

פיתון כמחשבון

1. 8
2. 25.0
3. 10
4. -34
5. -2
6. 24.0
7. 16.0
8. 4.0
9. 34
10. 504
11. 11.0
12. 364.5
13. 81.0
14. 3.0
15. 3.0
16. 36.0
17. 36.75
18. 21.25
19. 3.0
20. 11.25
21. 5
22. 0
23. 1
24. 2.0
25. 32.0
26. 8.0
27. 16.0
28. 0
29. 2
30. 1

[חזור לתוכן](#)

תשובות

סוגי נתונים מספריים

1. 58
2. 70161
3. 237
4. 36
5. 120
6. 121200.0
7. 1.313
8. 120000000.0
9. 2055.0
10. 1000.0
11. 56000.0
12. 540.0
13. 10000.0
14. 0.0123
15. 80.0
16. 80.0
17. 288.0
18. 1.6666667
19. 2.0
20. 2.0
21. 3.75
22. 3.75
23. 3.0
24. 3.0
25. 6.666666
26. 6.666666
27. 1.0
28. 1.0
29. 2.0
30. 4.0
31. 32
32. 5.0
33. 35
34. 5
35. 4.0
36. 256
37. 16
38. 1
39. 2
40. 5.0

[חזור לתוכן](#)

תשובות

דוגמת תצורות דלק

1. 82
2. 164
3. 10.25
4. 1.5
5. -4.5
6. 4.5
7. 11.25
8. 238
9. -98
10. 2
11. 49
12. 7.0
13. -2.0
14. -30.0
15. 0.5
16. 5.5
17. 0.5
18. 0.25
19. 4.0
20. 16.0
21. 18
22. 54
23. 3
24. 27
25. 2.25
26. 9.0
27. 20.0
28. 100.0
29. 3.0
30. 27.0
31. 12.0
32. 144.0
33. 9.0
34. 6.0
35. 36.0
36. 9.0
37. 3.0
38. 0.0
39. 1.0
40. 26.0
41. 8.0
42. 48.0
43. 4.0
44. 12.0
45. 36.0

[חזור לתוכן](#)

תשובות

שמות ערכים

1. 180
2. 5.0
3. 23
4. 7.5
5. 19.5
6. 50
7. 27.0
8. 0.75
9. 1.0
10. 4.0
11. 48
12. 5.0
13. 255
14. 1
15. 14.8
16. 2.6666666666666665
17. 729.0
18. 8
19. 3
20. 14

21. 49
22. 2.4285714285714284
23. 85.0
24. 2.0
25. 3.4
26. 2.0588235294117645
27. 343
28. -35831808.0
29. 0.02040816326530612
30. 19.0

חלק ב'

[חזור לתוכן](#)

תשובות

משתנים

```
1. 5
2. 6
3. 2.0
4. 0.3333333333333333
5. 0.5
6. 0.6666666666666667
7. 1.0
8. 4.0
9. 7.5
10. 1.5
11. 62.5
12. 2.2360679775
```

חלק ב'

```
1. 5.0
2. 6.0
3. 2.0
4. 0.3333333333333333
5. 0.5
6. 0.6666666666666667
7. 1.0
8. 4.0
9. 7.5
10. 1.5
11. 62.5
12. 2.2360679775
```

חלק ג'

```
1. 5.0
2. 6.0
3. 2.0
4. 0.3333333333333333
5. 0.5
6. 0.6666666666666667
7. 1.0
8. 4.0
9. 7.5
10. 1.5
11. 62.5
12. 2.2360679775
```

[חזור לתוכן](#)

תשובות

מחרוזות

1. `print ("Hello world")`
2. `print ("Hi people. \\"How are you?\\"")`
3. `print ("1. \\"abc\","cde\\" and 2. \\"fgh\","ijk\\"")`
4. `print ("\\"one\\" and \\"two\\"")`
5. `print ("\\"a\","b\","c\","d\\"")`
6. `print ("\\" \\"a\\" \\"")`
7. `print ("\\" \\"~\\"~\\"~\\"~\\" \\"")`
8. `print ("\\"(*)\\"(*)\\"(*)\\"")`
9. `print ("\\"^^\">>>>\\"<<<\\"")`
10. `print ("\\"0123456789\\"")`

[חזור לתוכן](#)

תשובות

עוד על סוגי נתונים

1. 2.0
- 2
- 3
- 300.0
- 4
2. <class 'float'>
3. <class 'long'>
4. 4
5. 4
6. 0.6666666666667
7. 150.0
8. 6.0
9. 1202.0
10. 150
11. 150.0
12. 4.0
13. 307.0
14. 1.5
15. -3600.0
16. 6.25
17. 4
18. 0
19. 0.0
20. 36.0
21. '0b100'
22. '0b1'
23. 1
24. '0b1001'
25. '0b100101100'
26. '1.0'
27. '7'
28. '1'
29. 8.0
30. '-724.0'

[חזור לתוכן](#)

תשובות

פונקציות מתמטיות נוספות

1. 7.5
2. 6.25
3. 0.8333333333333334
4. 0.5
5. 1
6. 7
7. 5
8. 0.5
9. -1.1
10. 0
11. 5
12. 0
13. 6
14. 0.4
15. 2.0
16. 5.5
17. 3
18. 3
19. 0
20. 2
21. 17.5
22. 12.25
23. 0.7
24. 1.5
25. 4
26. 9
27. 7
28. 1.5
29. -2.2142857142857144
30. 1
31. 7
32. 0
33. 8
34. 0.2857142857142857
35. 2.0
36. 22.0
37. 5
38. 5
39. 1
40. 2

[חזור לתוכן](#)

תשובות

טיפול במחרוזות

- 1.
2. 'mation '
3. 'Infor'
4. 'System'
5. 'Information Systems'
6. 'Information Systems'
7. 'Sys '
8. 'Sys Sys Sys '
9. '<Inf>'
10. 'ms'
11. 'I'
12. 'tems'
13. 'Information Systems'
14. 'Information Systems class'
15. 'Information+Systems=Information Systems'
16. 'on Systems'
17. 34
18. 'ation'
19. 'InfoInfoInfoInfo'
20. 3
21. 'or'
22. 'Information for Systems'
23. 'SysSysSys '
24. ['Syst ', 'Syst ', 'Syst ']
25. 'temSys'
26. ['SysIn', 'SysIn', 'SysIn']
27. ['ioiomamama', 'ioiomamama']
28. 'mamamia'
29. 'rmati'
30. "

[חזור לתוכן](#)

תשובות

הדפסה

1. $12 * 8 = 96$
2. This is an example
3. $3 * \text{fun} = \text{fun fun fun}$
4. Hi everybody , how are you?
5. Computer Com pu ter
6. " This is a test " is 14 character long
7. MyVar= 100
8. HelloWorld
9. One Megabytes is 2^{20} bytes, or : 1048576 bytes
10. V , V , V
11. ~~, ~~, ~~
12. a,b,c , d,e , f
13. ,,,,,, , ,,,,,,,
14. ,,
15. #,# , #, #
16. $3*5$, 33333
17. abcab
18. abcabcabcabcabc
19. print print print
20. $3*3*3*2*2*$

[חזור לתוכן](#)

תשובות

עוד על הדפסה

1. a
b
c
d
2. "a"="b?"
3. \\//\\//
4. a
ab
abc
abcd
5. a
 a
 a
6. \\\
7. \\\
 \\
 \\
 \\
8. a\\b!=b\\a
9. /\\
 /\\
 /\\
10. /\\
 ||
 \\
11. /\\
 /\\/\\
 ||||
 \\//
 \\
 /\\
 /\\/\\
 ||||
 \\//
 \\
 /\\
 /\\/\\
 ||||
 \\//
 \\
 /\\

- /\\
 ||||
 \\//
 \/
12. /
 /\\

 \\//
 \/
13. /
 /\
 /\
 /\\
 /\\
 /\\
 /\\
 \\//
 \\//
 \\//
 \/
 \/
 \/
14. /\ /\ /\ /\ /\ /
 -- -- -- -- -- --
 \/ \/ \/ \/ \/ \/
15. /
 oo
 \/
16. 1
 12
 123
 1234
17. xox oxo xox
 oxo xox oxo
 xox oxo xox
18. xox oxo xox
 oxo xox oxo
 xox oxo xox
19. \~/~\~/~\~/~\~/~\~/
20. \~/~\~/~\~/~\~/~\~/
 \~/~\~/~\~/~\~/

תכנות בסיסי

21. \~/~/~/~/~/~/~/~/
 \~/~/~/~/~/~/~/
 \~/~/~/~/~/
 \~/~/
 \~/

22. \//\//\//\//
 \//\//\//
 \//\//
 \//
 \//

[חזור לתוכן](#)

תשובות

לולאות

1. 1
5
9
2. 2
3
3
4
4
5
5
6
6
7
7
8
8
9
9
10
10
11
11
12
3. 2
0
-2
-4
-6
-8
-10
- 4.
5. 5
6. 0
4
9
16
64
9
36
16
7. 2.0
12.0
20.0
6.0
30.0
56.0

42.0
72.0

8. 1.5
1.0
2.5
2.0
3.5
3.0
4.5
4.0

9. 1.5
1.0
2.5
2.0
3.5
3.0
4.5
4.0

10. a
b
c
d
e
f
g

11.
a

b

c

d

e

f

g

12. Aa
Ba
Ca

13. 4
5
6
7
8

14. 2

4
 6
 8
 10

 15. 19
 29
 39
 49
 59

 16. 0
 1
 0
 1
 0
 1

 17. 0
 0
 0
 0
 0
 0

 18. 1
 1
 1
 1
 1
 1

 19. 2
 3
 4
 5
 6
 7

 20. 1
 1
 1
 1
 1
 1

 21. 1
 2
 3
 4
 5
 6

 22. 0
 3

תכנות בסיסי

0
5
0
7

23. 22
333
4444
55555
666666
7777777

24. 2 2
3 3
4 4
5 5
6 6
7 7

25.
*
**

26.

* *
* *
* * *
* * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *

27. * *
* ** *
* ** ** *
* ** ** ** *

28. 1

,

2

,

3

,

4

29. 1

2

3

4

30. 10

9

8

7

6

5

4

3

2

1

31. 10

9

8

7

6

5

4

3

2

1

32. Current item in list is: This

Current item in list is: is

Current item in list is: an

Current item in list is: example

33. עבור כל המספרים בתחום 1-9 התוכנית מדפיסה את המספר, את המספר בריבוע ואת המספר בשלישית

1 1 1

2 4 8

3 9 27

4 16 64

5 25 125

6 36 216

7 49 343

8 64 512

9 81 729

34. התוכנית מחשבת את ממוצע שבעה המספרים (0-6), כאשר הממוצע מחושב עבור אחד מהמספרים בנפרד.

0.0

0.5

1.0

1.5

תכנות בסיסי

2.0
2.5
3.0

35. התוכנית מחשבת את ממוצע ריבועי שבעה המספרים (0-6)

13.0

36. התוכנית מדפיסה את הערך המחושב ע"י סכום האיברים ברשימה מחולק בערך האיבר האחרון ועוד אחד.

4.6

37. התוכנית מכפילה את האיברים ברשימה, מכפילה את הכל באפס ומחשבת מודולו חמש.

0.0

38. התוכנית מחשבת את ערכי המספרים בתחום שבין מינוס 32 ועד מינוס 22 (כולל) ועוד 30.

-2
-1
0
1
2
3
4
5
6
7
8

39. התוכנית מדפיסה את המספרים 0-8, כאשר בכל שורה מתווסף מספר אחד. ולכן נוצר משולש שקודקודו ריק ובסיסו הרשימה 0-8.

0
0 1
0 1 2
0 1 2 3
0 1 2 3 4
0 1 2 3 4 5
0 1 2 3 4 5 6
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7 8

40. התוכנית מדפיסה שבע סדרות של כוכביות, כל סדרה בשורה חדשה כאשר בכל סדרה מתווספות עוד כוכביות. בסדרה הראשונה כוכבית אחת, בסדרה השנייה כוכבית אחת ואחריה (אחרי רווח) שתי כוכביות, בסדרה השלישית כוכבית אחת, אחריה שתיים ואחריהן שלוש וכן הלאה עד לסדרה האחרונה שבה רשימה של כוכבית אחת, שתיים, שלוש ועד שבע כוכביות.

*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *

41. התוכנית מסבירה את מושג הלולאות המקוננות ומדפיסה את מונה הלולאה בכל מחזור. המחזור ראשון לא מתבצע. במחזור השני j מונה הלולאה החיצונית הוא אחד ומונה הלולאה

תכנות בסיסי

הפנימית (k) הוא אפס. לאחר מכן מתקדם מונה הלולאה החיצונית ומבצעים פעמיים את הלולאה הפנימית וכן הלאה עד שמונה הלולאה החיצונית מגיע לערך 6 ועבורו מבצעים שש פעמים את הלולאה הפנימית.

outer loop = 1 inner loop = : 0

outer loop = 2 inner loop = : 0

outer loop = 2 inner loop = : 1

outer loop = 3 inner loop = : 0

outer loop = 3 inner loop = : 1

outer loop = 3 inner loop = : 2

outer loop = 4 inner loop = : 0

outer loop = 4 inner loop = : 1

outer loop = 4 inner loop = : 2

outer loop = 4 inner loop = : 3

outer loop = 5 inner loop = : 0

outer loop = 5 inner loop = : 1

outer loop = 5 inner loop = : 2

outer loop = 5 inner loop = : 3

outer loop = 5 inner loop = : 4

outer loop = 6 inner loop = : 0

outer loop = 6 inner loop = : 1

outer loop = 6 inner loop = : 2

outer loop = 6 inner loop = : 3

outer loop = 6 inner loop = : 4

outer loop = 6 inner loop = : 5

42. עבור כל אחד מהמספרים בתחום שבין 2-19 התוכנית מדפיסה את המספר מהמחלקים שלו. אם אין לא מחלקים, המספר לא מודפס.

4 2

6 2

6 3

8 2

8 4

9 3

10 2

10 5

12 2

12 3

12 4

12 6

תכנות בסיסי

14 2
14 7

15 3
15 5

16 2
16 4
16 8

18 2
18 3
18 6
18 9

43. התוכנית מחשבת את ממוצע האיברים ברשימה ולאחר מכן רושמת כמה מהמספרים קטנים מהממוצע, כמה שווים לממוצע וכמה גבוהים מהממוצע

For: [1, 2, 3, 3, 4, 2]

Above average: 3

Average: 0

Less than average: 3

44. עבור המספרים בתחום 2-6 התוכנית יוצרת את כל המכפלות בערכים 1-6.

$1 \times 2 = 2$
 $2 \times 2 = 4$
 $3 \times 2 = 6$
 $4 \times 2 = 8$
 $5 \times 2 = 10$
 $6 \times 2 = 12$

 $1 \times 3 = 3$
 $2 \times 3 = 6$
 $3 \times 3 = 9$
 $4 \times 3 = 12$
 $5 \times 3 = 15$
 $6 \times 3 = 18$

 $1 \times 4 = 4$
 $2 \times 4 = 8$
 $3 \times 4 = 12$
 $4 \times 4 = 16$
 $5 \times 4 = 20$
 $6 \times 4 = 24$

 $1 \times 5 = 5$
 $2 \times 5 = 10$
 $3 \times 5 = 15$
 $4 \times 5 = 20$
 $5 \times 5 = 25$
 $6 \times 5 = 30$

 $1 \times 6 = 6$
 $2 \times 6 = 12$

תכנות בסיסי

$$3 \times 6 = 18$$

$$4 \times 6 = 24$$

$$5 \times 6 = 30$$

$$6 \times 6 = 36$$

45. התוכנית בונה את לוח הכפל עובר המספרים 2-9

2		4	6	8	10	12	14	16	18
3		6	9	12	15	18	21	24	27
4		8	12	16	20	24	28	32	36
5		10	15	20	25	30	35	40	45
6		12	18	24	30	36	42	48	54
7		14	21	28	35	42	49	56	63
8		16	24	32	40	48	56	64	72
9		18	27	36	45	54	63	72	81

46. התוכנית מדפיסה את המספרים בתחום $1-7 \times 8$ כאשר הם בנויים בצורת מלבן בגודל של 8 שורות ושבעה ערכים בכל שורה.

```

1  2  3  4  5  6  7
8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31 32 33 34 35
36 37 38 39 40 41 42
43 44 45 46 47 48 49
50 51 52 53 54 55 56

```

47. התוכנית מדפיסה סדרה של חמישה משולשים, כאשר כל משולש מורכב מארבע שורות. בשורה הראשונה כוכבית אחת, בשנייה שתיים וכן הלאה עד אשר בשורה הרביעית ארבע כוכביות. הראשונה מוסטת תו אחד ימינה וכל סדרה מוסטת בתו אחד נוסף.

```

*
**
***
****

*
**
***
****

*
**
***
****

*
**

```

תכנות בסיסי

```
***
****

*
**
***
****
```

48. התוכנית מדגימה קינון לולאות ועבור כל לולאה חיצונית מבצעת שלוש לולאות פנימיות. התוכנית מדפיסה את ערכי מוני הלולאות (הפנימי והחיצוני).

```
k = 1 | j = 1
k = 1 | j = 2
k = 1 | j = 3
k = 2 | j = 1
k = 2 | j = 2
k = 2 | j = 3
k = 3 | j = 1
k = 3 | j = 2
k = 3 | j = 3
k = 4 | j = 1
k = 4 | j = 2
k = 4 | j = 3
Sof!
```

49. התוכנית מדגימה ארבע לולאות מקוננות, כאשר הלולאה הפנימית פשוט סופרת (מקדמת מונה באחד בכל מחזור). סה"כ מספר המחזורים המבוצע הוא 5×4 a) מתבצעת 5 פעמים ועבור כל פעם שלה, b מתבצעת חמש פעמים ועבור כל פעם שלה, c מתבצעת חמש פעמים ועבור כל פעם שלה, d מתבצעת חמש פעמים, סה"כ $5 \times 5 \times 5 \times 5$

625

50. דוגמה אחרת של קינון לולאות. הלולאה החיצונית מבוצעת 6 פעמים ועבור כל מחזור שלה מבוצעת לולאה פנימית עבור הערכים 1-מונה הלולאה החיצונית. הלולאה הפנימית מדפיסה את ערך מונה הלולאה הפנימית, ערך מונה הלולאה החיצונית והסכום שלהם.

```
1 1 1
-----
2 1 2
2 2 4
-----
3 1 3
3 2 6
3 3 9
-----
4 1 4
4 2 8
4 3 12
4 4 16
-----
5 1 5
5 2 10
5 3 15
5 4 20
5 5 25
-----
6 1 6
6 2 12
```

תכנות בסיסי

```
6 3 18
6 4 24
6 5 30
6 6 36
-----
```

51. התוכנית מדפיסה את המחרוזות ברשימה הנתונה, כאשר המחרוזות מודפסת פעם אחת כמו שהיא ופעם שנייה לאחר ששוכפלה פעמיים.

```
ab ababab c ccc
```

52. לכל אחת מהמחרוזות ברשימה outer, התוכנית מחברת כל אחת מהמחרוזות ברשימה inner.

```
ABab
ABcd
ABef
CDab
CDcd
CDef
EFab
EFcd
EFef
```

53. התוכנית מדפיסה את תווי המחרוזת, כך שהתו הראשון מודפס בשורה הראשונה (פעם אחת), התו השני מודפס פעמיים בשורה השנייה וכן הלאה.

```
A
BB
CCC
DDDD
EEEE
```

54. התוכנית מדפיסה את תווי המחרוזת בשורה אחת, כך שהתו הראשון מודפס פעם אחת, התו השני מודפס פעמיים וכן הלאה.

```
A BB CCC DDDD EEEEE
```

55. התוכנית מדפיסה את תווי המחרוזת במספר שורות. בשורה הראשונה מודפס התו הראשון (פעם אחת). בשורה השנייה מודפס התו הראשון פעם אחת והתו השני פעמיים. כך בכל שורה מתווסף התו הבא מוכפל במספר השורה.

```
A
A BB
A BB CCC
A BB CCC DDDD
A BB CCC DDDD EEEEE
```

56. התוכנית סופרת בבינארית מאפס ועד 31 (חמש סיביות)

```
0 0 0 0 0
0 0 0 0 1
0 0 0 1 0
0 0 0 1 1
0 0 1 0 0
0 0 1 0 1
0 0 1 1 0
0 0 1 1 1
0 1 0 0 0
0 1 0 0 1
```

תכנות בסיסי

```
0 1 0 1 0
0 1 0 1 1
0 1 1 0 0
0 1 1 0 1
0 1 1 1 0
0 1 1 1 1
1 0 0 0 0
1 0 0 0 1
1 0 0 1 0
1 0 0 1 1
1 0 1 0 0
1 0 1 0 1
1 0 1 1 0
1 0 1 1 1
1 1 0 0 0
1 1 0 0 1
1 1 0 1 0
1 1 0 1 1
1 1 1 0 0
1 1 1 0 1
1 1 1 1 0
1 1 1 1 1
```

57. התוכנית מדפיסה את האותיות Word, כל אות בשורה נפרדת

```
W
o
r
d
```

58. התוכנית מוסיפה אחד לאיברי הרשימה ומדפיסה אותה

[2, 3, 4, 5]

59. התוכנית מוסיפה עבור כל אחד מאיברי הסדרה הראשונה, את האיברים בסדרה השנייה ובלבד שמכפלתם גדולה מ- 25. כל זוג מספרים מודפס בשורה נפרדת.

```
2 15
2 22
3 10
3 15
3 22
4 10
4 15
4 22
```

60. התוכנית מחפשת ומדפיסה את המספר הגדול ביותר ברשימה

largest value : 44

61. התוכנית מחפשת ומדפיסה את המספר הגדול ביותר ברשימה עד האיבר שערכו 16

largest value : 31

62. התוכנית מחפשת ומדפיסה את המספר הגדול ביותר ברשימה, אבל תוך התעלמות מהמספר 44

תכנות בסיסי

largest value : 40

63. התוכנית מחברת את הערך 5 ארבע פעמים ואת התוצאה מחברת שלוש פעמים (תוכנית סינטטית שנועדה להדגישם שלוש לולאות מקוננות).

60

[חזור לתוכן](#)

תשובות

ביטויים בוליאניים

1. True
2. True
3. False
4. False
5. False
6. False
7. False
8. False
9. True
10. True
11. True
12. False
13. False
14. True
15. True
16. True
17. False
18. True
19. True
20. True
21. True
22. False
23. True
24. True
25. True
26. True
27. True
28. True
29. True
30. True
31. True
32. True
33. True
34. True
35. True
36. True
37. False
38. True
39. False
40. False
41. $(1 < 1) == \text{False}$
 $(1 < 2) == \text{True}$
 $(1 < 3) == \text{True}$
 $(2 < 1) == \text{False}$
 $(2 < 2) == \text{False}$
 $(2 < 3) == \text{True}$
 $(3 < 1) == \text{False}$
 $(3 < 2) == \text{False}$
 $(3 < 3) == \text{False}$

[חזור לתוכן](#)

תשובות

קלט

1. 6 66 <class 'str'>
2. 7 3 777 <class 'str'> <class 'int'> <class 'str'>
 999 4 9999999999999999 <class 'str'> <class 'int'> <class 'str'>
 55.0 3 55.055.055.0 <class 'str'> <class 'int'> <class 'str'>
 abc 4 abcabcabcabc <class 'str'> <class 'int'> <class 'str'>
 17-11 2 17-1117-11 <class 'str'> <class 'int'> <class 'str'>
3. For first 5 number(s) the sum is: 10
 For first 12 number(s) the sum is: 66
 For first 8 number(s) the sum is: 28
 For first 1 number(s) the sum is: 0
 For first 0 number(s) the sum is: 0
4. The product is: 1
 The product is: 1
 The product is: 2
 The product is: 24
 The product is: 720
5. 0
 1
 1
 2
 5
6. 0
 1
 3
 4
 10
7. 2
 7
 7
 2
 15
8. day
 del
 vat
 fog
 sit
9. abcd <> dcba
 1234567 <> 7654321
 ab <> ba
 r <> r
 9753 <> 3579

תכנות בסיסי

10. 12345678 <> 8642
90123456 <> 6420
abcd <> db
n <>
01010101 <> 1111
11. 12345678 <V> 18273645
901238456 <V> 96051428
abcd <V> adbc
n <V>
01010101 <V> 01100110

[חזור לתוכן](#)

תשובות

תנאים

1. התוכנית מדפיסה ריבוע כפל של 4-9, כאשר כל מכפלה של איברים שווים מוחלפת ב - * .

```
* 20 24 28 32 36
20 * 30 35 40 45
24 30 * 42 48 54
28 35 42 * 56 63
32 40 48 56 * 72
36 45 54 63 72 *
```

התוכנית מדפיסה ריבוע כפל של 4-9, כאשר כל מכפלה של איברים שונים מוחלפת ב - * .

```
16 * * * * *
* 25 * * * *
* * 36 * * *
* * * 49 * *
* * * * 64 *
* * * * * 81
```

2. התוכנית מדפיסה ריבוע כפל של 5-9, כאשר כל מכפלה של איברים שווים מוחלפת ב - * .

```
25 * * * *
* 36 * * *
* * 49 * *
* * * 64 *
* * * * 81
```

3. התוכנית מדפיסה טרפז (משולש קטום) בעל חמש שורות, כאשר השורה הראשונה בגודל של ארבעה איברים ובכל שורה מספר האיברים גדל. כל האיברים המודפסים הם כוכביות, אלא אם כן אם מספר השורה ועוד חמש ועוד מספר האיבר בתוך השורה שווה או גדול מ - 11 ואז מודפס ? במקום הכוכבית.

```
? ? ? ?
? ? ? ? *
? ? ? * * *
? ? * * * *
? ? * * * * *
? * * * * * *
```

4. התוכנית מחשבת את ממוצע המספרים ברשימה שלא מתחלקים ב - 2.

5.0

5. התוכנית מחשבת את ממוצע המספרים שהמיקום שלהם ברשימה מודולו 2 ומודולו 5 שווים.

1.5

6. התוכנית מעתיקה את המחרוזת, תוך הכפלת הספרות שנמצאות במיקומים שהם מכפלה של 3.

123456789 >>> 112344567789

7. התוכנית מעתיקה את המחרוזת תוך הכפלת התווים שהמיקום שלהם הוא מכפלה של שלוש ושילוש התווים שהמיקום שלהם הוא מכפלה של שתיים.

abcdefgh >>> aabcccddeefggh

8. התוכנית יוצרת מחרוזת חדשה על סמך הערכים בשתי מחרוזות הקלט, כאשר האיבר במחרוזת הפלט הוא האיבר הקטן מבין שני האיברים המתאימים בקלט.

54678492 62384568 52374462

9. התוכנית יוצרת מחרוזת חדשה על סמך שתי מחרוזות קלט, כאשר אם האיבר המתאים מ – a הוא אי זוגי הוא מועבר למחרוזת החדשה, אם לא נבדק האיבר המתאים במחרוזת b ואם הוא אינו זוגי הוא מועבר, אחרת מוסיפים למחרוזת את הערך 0.

54678492 62384568 50370590

10. התוכנית מחשבת ערך משתנה על סמך ערכי מחרוזת. אם סכום שני איברים סמוכים קטן מחמישים הם מתווספים למשתנה c, אם לא הם מתווספים כאשר סדר הספרות הפוך, כלומר ספרת האחדות היא המספר השני ואילו ספרת האחדות היא המספר הראשון.

4356 143

11. התוכנית מדפיסה את תוכן מחרוזת הקלט שקיבלה, את האיבר הגדול ביותר שבה (אם יש יותר מאחד, את המיקום הראשון שלו), את האיבר הקטן ביותר במחרוזת ואת המיקום שלו.

857368132843745632837 8 0 1 6

12. התוכנית מקבלת מספר (7) ובודקת אם הוא גדול מעשר (ואז מדפיסה גבוה), קטן מחמש (ואז מדפיסה נמוך), או מדפיסה באמצע.

In between

[חזור לתוכן](#)

תשובות

רשימות

1. [4, 5, 6, 7]
2. ['*', '*', 1, 2, 1, 2, 1, 2]
3. ['*', 2, '*', 4, '*', 6, '*']
4. [1, 1, 2, 1, 2, 3, 1, 2, 3, 4]
5. [1, 1, 2, 2, 1, 2, 3, 2, 3, 3]
6. [1, 1, 4, 3, 6, 5, 7, 9]
7. [1, 1, 3, 5, 4, 6, 7]
8. [1, 1, 3, 5, 6]
9. [8, 6, 2, 1, 5, 7]
10. [5, 1, 1, 2]
11. a b aaa bbb
12. 31 32 33
41 42 43
13. 9
1
-5
-15
14. Four
15. [30, 40]
[30, 40, 50, 60, 70]
[10, 20, 30, 40, 50, 60, 70]
[10, 20, 30, 40, 50, 60, 70]
[10, 20, 30, 40, 50]
[60, 70]
[50, 60]
16. [10, 20, 30, 2]
17. Length of x: 8
Item 1 = [88]
Item 2 = [11]
Item 3 = [99]
Item 4 = [66]
Item 5 = [77]
Item 6 = [44]
Item 7 = [55]
Item 8 = [33]

תכנות בסיסי

18. 1 is not in [2, 4, 7]
2 is in [2, 4, 7]
3 is not in [2, 4, 7]
4 is in [2, 4, 7]

19. הופך את סדר האיברים ברשימה.
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

20. A
BB
CCC
DDDD
EEEEEE
FFFFF
GGG
HH
K

הערה: שינוי הגופן נועד להראות את הצורה המתקבלת.

21. A ----- A
BB ----- BB
CCC ----- CCC
DDDD ----- DDDD
EEEEEE ----- EEEEE
FFFFFF ----- FFFFFFF
GGGGGGG ----- GGGGGGG
HHHHHHHH ----- HHHHHHHH
KKKKKKKKK -- KKKKKKKKK

22. 19 10 34 10
23 26 26 27

23. 1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1

24. [11, 21, 28, 34, 43, 52, 56, 58, 61, 66, 67, 73, 81, 84]

[חזור לתוכן](#)

תשובות

לולאות while

1. $j = 5$
2. 55
3. 1 3
4. 1 3 7 15
5. 15
6. 10 8
7. Enter temperature please: 22
it is just right
Enter temperature please: 60
it is hot
Enter temperature please: 23
it is just right
Enter temperature please: 12
it is cold
Enter temperature please: 0
Bye!
8. 4 0
5 0
9. Start
33
30
28
End
10. 1
2
6
11.

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40
5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100
12. *
* *

```
* * *
* * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

13. 0 1 2 3 4 5 6 7 8 9
 0 1 2 3 4 5 6 7 8 9
 0 1 2 3 4 5 6 7 8 9
 0 1 2 3 4 5 6 7 8 9
 0 1 2 3 4 5 6 7 8 9

14. 1 x 8 = 8
 2 x 8 = 16
 3 x 8 = 24
 4 x 8 = 32
 5 x 8 = 40
 6 x 8 = 48
 7 x 8 = 56
 8 x 8 = 64
 9 x 8 = 72

15. Current variable value is : 10
 Current variable value is : 9
 Current variable value is : 8
 Current variable value is : 7
 Current variable value is : 6

16. Current variable value is : 9
 Current variable value is : 8
 Current variable value is : 7
 Current variable value is : 6
 Current variable value is : 4
 Current variable value is : 3
 Current variable value is : 2
 Current variable value is : 1
 Current variable value is : 0

[חזור לתוכן](#)

תשובות

פונקציות

1. C B A C
2. 21 55 0 15
3. 0 16 144 4
4. 4 36 100 16
5. 9 27 81
4 -8 16
6. Hello...
Hello...
Hello...
Hello again...
Hello again...
Hello again...
Hello again...
Hello again...
7. BOB
BOBBOB
BOBBOBBOB
BOBBOBBOBBOB
BOBBOBBOBBOBBOB

XOX
XOX XOX
XOX XOX XOX
XOX XOX XOX XOX
XOX XOX XOX XOX XOX
XOX XOX XOX XOX XOX XOX

[חזור לתוכן](#)

תשובות

עוד על פונקציות

1. 41

2. 2
6

3. 2

4. 9
8
7
6
5
4
3
2
1

5. 28

6. 13

7. 100.0

8. 60

9. 7.5

10. This road is long.
This road is short.

This is a new car.
This is a used car.
This is a strange car.

This lesson is long and interesting.
This lesson is short and interesting.

11. 0 1 -1
1
1

12. 5.0
5.0
5.0
25.0
5.0

13. 1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	18	20
3	6	9	12	15	18	21	24	27	30
4	8	12	16	20	24	28	32	36	40

תכנות בסיסי

5	10	15	20	25	30	35	40	45	50
6	12	18	24	30	36	42	48	54	60
7	14	21	28	35	42	49	56	63	70
8	16	24	32	40	48	56	64	72	80
9	18	27	36	45	54	63	72	81	90
10	20	30	40	50	60	70	80	90	100
14.1									
2	4								
3	6	9							
4	8	12	16						
5	10	15	20	25					
6	12	18	24	30	36				
7	14	21	28	35	42	49			
8	16	24	32	40	48	56	64		
9	18	27	36	45	54	63	72	81	
10	20	30	40	50	60	70	80	90	100

[חזור לתוכן](#)

תשובות

טיפול ברשימות

1. A
BB
CCC
DDDD
EEEE
FFFFFF
GGGGGG
HHHHHHH
KKKKKKKK
2. [35, 33, 30, 27, 23, 22, 21, 16, 13, 11]
3. [35, 21, 33, 13]
4. [43, 35, 21, 33, 13, 22, 19]
5. [19, 3, 13, 3, 13, 2, 19, 2, 22, 2, 13, 3, 22, 2]
6. [19, 1, 13, 2, 13, 0, 19, 6, 22, 1, 13, 2, 22, 0]
7. [14, 14, 14, 19, 19, 19]
8. [22, 11, 55, 44, 33, 22, 11, 55, 44, 33]
9. [11, 22, 33, 55, 88, 143, 33, 44, 55]
10. [48, 36, 29, 12]
11. [29, 26, 22, 19, 17]
12. [11, 28, 19, 22, 26, 14, 15, 27]
13. [44, 35, 32, 28, 23, 21, 14, 8]
14. [2, 21, 8, 22, 7, 19, 2, 18]
15. [29, 24, 25, 27]
16. [33, 34, 32, 37, 36, 38, 39, 35]

[חזור לתוכן](#)

תשובות

פעולות בוליאניות

1. A C
A
C
2. In range
Out of range
Out of range
3. C
A
A
B
B
C
C
C
4. Wrong
Correct
Wrong
Wrong
Correct
Correct
Correct
Wrong
5. Correct
Wrong
Correct
Correct
Wrong
Correct
Wrong
Wrong
6. Correct
Correct
Correct
Correct
Correct
Correct
Correct
Correct

[חזור לתוכן](#)

תשובות

פונקציות לטיפול במחרוזות

1. 'By the rivers of babylon, there we sat down'
2. 'BY THE RIVERS OF BABYLON, THERE WE SAT DOWN'
3. 'By The Rivers Of Babylon, There We Sat Down'
4. 'bY tHE rIVERS oF bABYLON, tHERE wE sAT dOWN'
5. ['by th', ' riv', 'rs of babylon, th', 'r', ' w', ' sat down']
6. ['BY THE RIVERS OF BABYLON, THERE WE SAT DOWN']
7. 'By The Rivers Of Babylon'
8. ['By', 'The', 'Rivers', 'Of', 'Babylon']
9. 'Rivers'
10. ['by', 'the', 'rivers of babylon, there we sat down']
11. 'LITTLE'
12. 'd--umme--'
13. 'you'
14. ' s*cr*t '
15. 'follow the sun'
16. 'Let It Be!!!!'
17. False
False
18. 5
5
22
13
13
19. Take me back to my boat on the river
****Take Me Back To My Boat On The River****

[חזור לתוכן](#)

תשובות

עיצוב מחרוזות

1. text = 123, value = 123
2. text = 123, value = 123
3. text = 123, value = 123.000000
4. text = 123, value = 123.00
5. text = 123, value = 1.23e+02
6. text = 123, value = 1.23e+02
7. value = 7777, text = 7777
8. value = 119, text = 119
9. value = 7777.000, text = 7777
10. value = 7777.000, text = 7.777e+03
11. value = 7777.000, exponent = 7.7770000e+03
12. value = 5929.000, exponent = 5.929e+03
13. value = 539.000, exponent = 5.390e+020
14. value = 241.000, exponent = 2.410e+02
15. value = 361, integer = 241
16. 307, c7, 199, 199, 1.990000e+02
17. 307, c7, 199, 199, 1.990000e+02
18. text = 123456, value = 123456
19. text = 121212, value = 121212
20. text = 020202, value = 8322
21.

i	i**2	i**3	i**5	i**10	i**20
1	1	1	1	1	1
2	4	8	32	1024	1048576
3	9	27	243	59049	3486784401
4	16	64	1024	1048576	1099511627776
5	25	125	3125	9765625	95367431640625
6	36	216	7776	60466176	3656158440062976
7	49	343	16807	282475249	79792266297612001
8	64	512	32768	1073741824	1152921504606846976
9	81	729	59049	3486784401	12157665459056928801
10	100	1000	100000	10000000000	100000000000000000000

[חזור לתוכן](#)

תשובות

פקודות break ו- else

1. 3 is a prime number
4 equals 2 * 2
5 is a prime number
5 is a prime number
5 is a prime number
6 equals 2 * 3
7 is a prime number
7 is a prime number
7 is a prime number
7 is a prime number
7 is a prime number
8 equals 2 * 4
9 is a prime number
9 equals 3 * 3
2. breaking out of the loop!
3. 5 6 breaking out of the loop!
4. 1
1
2
2
3
3
4
breaking out of loop
5. 1
2
3
4
breaking out of loop
6. 4
7. 9 not found!
444 not found!
275 found!
787 not found!
509 found!
11 found!
37 found!
666 not found!

[חזור לתוכן](#)

תשובות

הבנת תוכניות

1. א'
2. 5
3. 0
4. 15
5. 0
6. 10
7. ג'
8. 6
9. 3
- 10.
- א. לא
- ב. לא
- ג. כן
11. 15
12. 16
13. 8
14. 24
15. 80
16. 2 4 5
17. 2
- 4.0
- 4.4
- 1
- 66.0
- 102
- 102.0
- 96.0
- 2.0
- 18
18. [4, 9, 5, 1, 8, 5, 2, 0]
19. [5, 6, 1, 9, 9, 1, 6, 5]
20. [-1, -2, 0, 4, 7, 2, 4]
21. [2, 6, -1, 4, 0, 13, 2, 10]
22. [3, 7, 9, 9, 14, 21, 30, 32]
23. [4, 1, 3, 2]

[חזור לתוכן](#)

תשובות

מצא את החסר

1. ג'
2. א'
3. ב'
4. ג'
5. א'
6. ג'
7. ב'
8. ד'
9. א'

חזור [לתוכן](#)

תשובות

כתיבת תוכניות

הערה כללית: הפתרונות הנתונים כאן, אינם בהכרח היעילים או הנכונים ביותר. הם נכתבו מתוך כוונה להסביר את צורת החשיבה בבואנו לפתור בעיה חישובית.

1. זאת דוגמה לתוכנית פשוטה, אשר אמורה לקבל שני מספרים מהמשתמש. המספרים מייצגים את צלעות מרובע כלשהו. אין הגדרה של מאפייני המספרים (אם מדובר במספרים שלמים או מספרים ממשיים) ולכן נניח שהם ממשיים (מסוג float). עיצוב התוכנית יכלול הדפסת כותרת התוכנית (המסבירה במשפט קצר מה מטרתה), זה אמנם לא הופיע בשאלה, אך התווסף למטרת בהירות נוספת. לאחר מכן, שתי פקודות קלט, ואחריהן הדפסה של הנתונים שהתקבלו, כולל חישוב השטח (הכפלת שני המספרים) וחישוב ההיקף שהינו פעמים סכום הצלעות.

התוכנית לדוגמה:

```
# Square.py
# A program for reading two numbers representing a square side
# It uses calculates the area and circumference
# Note: It does not check validity

def main():
    print ("This program calculates the area and circumference")
    print ()

    a = float(input("Enter the first number: "))
    b = float(input("Enter the second number: "))

    print ()
    print ("For sides:", a, b)
    print ("The area is:", a*b)
    print ("The circumference is:", 2*(a+b))
```

2. במקרה זה נתון שעלינו לקרוא שבעה מספרים ולחשב את סכומם ואת הממוצע שלהם. אנו לא נדרשים לשמור את המספרים שקראנו ולכן לאחר קריאת כל מספר רק נסכם אותו למונה צובר. עיצוב התוכנית יכול להיות הדפסת כותרת קצרה המגדירה מה התוכנית עושה. לאחר מכן נגדיר מונה צובר (sum) אליו נסכם את כל שבעת המספרים שנקלוט. המונה מאתחל ל-0.0 (אפשר היה גם לאתחל אותו ל-0 ואז המשמעות הייתה שהוא מסוג INT). לאחר מכן, התוכנית תכלול לולאה המתבצעת שבע פעמים ואשר במסגרתה אנו קוראים את שבעת המספרים. בשאלה לא הוגדר סוג הנתונים של המספרים (FLOAT או INT) ולכן, כדי לכתוב תוכנית כללית יותר נבחר במספרים ממשיים (Float). זו הסיבה שאתחול המונה הצובר יכול היה להיות כמספר שלם, או ממשי. ממילא לאחר שקלטנו את המספר הראשון (שהינו ממשי) וחיברנו אותו למונה הצובר, גם המונה הצובר היה, אפילו אם היה מספר שלם, היה הופך להיות מספר ממשי. הלולאה מתבצעת שבע פעמים ובכל פעם אנו קולטים מספר נוסף ומחברים אותו אל המונה הצובר. לאחר סיום הלולאה (ולאחר שנקלטו שבעה המספרים התוכנית תדפיס את סכומם וכן את הממוצע המתקבל מחלוקת הסכום בשבע).

התוכנית לדוגמה:

```
# Sum.py
# A program for reading seven numbers and calculates their sum
# and average
# Note: It does not check validity

def main():
    print ("This program reads seven numbers")
    print ("Calculates their sum and average")
    print ()

    sum = 0.0

    for indx in range(7):
        a = float(input("Enter a number: "))
        sum = sum + a

    print ()
    print ("the sum is:", sum)
    print ("The average is:", sum/7)
```

3. בשאלה זו עלינו להרחיב את התוכנית הקודמת ולשמור את שבעה המספרים שנקראו. כזכור, בשאלה הקודמת לא נדרשה השמירה ולכן כל מספר שנקרא רק סוכם אל המונה הצובר. כדי לשמור, עלינו להוסיף רשימה של שבעה איברים, אשר במקרה זה מאותחלים לאפס. (אפשר גם להגדיר רשימה ריקה, אשר תגדל בכל לולאה, כאשר נוסיף לה איברים, אך לצורך כך נדרשות פונקציות ספריה, אשר בשלב זה טרם עברנו עליהן). שינוי נוסף שעלינו להכניס לתוכנית כולל הוספת פקודה בתוך הלולאה הקוראת את המספרים ואשר נועדה לשמור כל מספר ברשימה שהגדרנו. מונה הלולאה משמש למפתוח המיקום לשמירה (במחזור הראשון האיבר יישמר בכתובת 0, במחזור השני בכתובת 1 וכן הלאה). לאחר שקראנו את שבעת המספרים (וגם שמרנו אותם), ולפני שנדפיס את סכומם ואת הממוצע שלהם (כפי שעשינו בתוכנית הקודמת), עלינו להדפיסם. הדבר מבוצע ע"י לולאה נוספת, אשר מתבצעת שבע פעמים ומדפיסה את איברי הרשימה. את איברי הרשימה אפשר היה גם להדפיס ללא לולאה, למשל ע"י שימוש בפקודה `print (list)`, אלא שאז הם היו מודפסים כרשימה (מוקפים בסוגריים מרובעים).

התוכנית לדוגמה:

```
# Sum.py
# A program for reading seven numbers and calculates their sum
# and average
# Note: It does not check validity

def main():
    print ("This program reads seven numbers")
    print ("Calculates their sum and average")
    print ()

    sum = 0.0
    list = [0,0,0,0,0,0,0]      # empty list for numbers to be read

    for indx in range(7):
        a = float(input("Enter a number: "))
        list[indx]=a            # populate list
        sum = sum + a

    print ("Numbers entered: ", end="")
    for ix in range(7):         # print list of numbers
        print (list[ix], " ", end="") # all on same line

    print ()
    print ("the sum is:", sum)
    print ("The average is:", sum/7)
```

4. בשאלה זו אנו נדרשים לקרוא שני מספרים, a ו- b ולפי הניסוח אפשר להבין שמדובר במספרים שלמים. לכן בשלב ראשון התוכנית תכלול (פרט להדפסת כותרת המגדירה את פעולתה) שתי פקודות קלט וכן הגדרת מונה צובר המאותחל ל-0 ואשר אליו נסכם את המספרים. בשלב שני נגדיר לולאה אשר תתבצע עבור כל המספרים בתחום שבין $a+1$ לבין b . במילים אחרות, הלולאה מתבצעת החל מהמספר הראשון ועוד אחד $(a+1)$, שכן לפי תנאי השאלה עלינו לסכם את כל המספרים שנמצאים בין a ל- b (לא כולל a ו- b עצמם). הפרמטר השני הקובע את הלולאה הוא b (מפני שבכל לולאה הפרמטר השני מגדיר את הגבול, אך בפועל הלולאה מתבצעת עד (כולל) האיבר שלפני הפרמטר). הלולאה מתקדמת מ- $a+1$ ועד ל- b משום שכתוב שאנו יכולים להניח שהמספר השני הוא הגדול שבין שני המספרים. בכל מחזור של הלולאה נחבר את המספר המתאים אל המונה הצובר, עד אשר הלולאה מסתיימת. השלב השלישי בעיצוב התוכנית הוא בהדפסת הטווח וסכום המספרים שבו.

התוכנית לדוגמה:

```
# Sum.py
# A program for reading two numbers and calculates the sum
# of all numbers between them
# Note: The program assumes the first number is the smaller one

def main():
    print ("This program reads two numbers")
    print ("and calculates the sum of all numbers")
    print ("them")
    print ()

    sum = 0
    a = int(input("Enter first number: "))      # get first number
    b = int(input("Enter second number: "))     # get second number
    for inx in range(a+1,b,1):                  # sum number between a-b
        sum = sum + inx

    print ()
    print ("the sum between", a,b, "is: ",sum)
```

5. בשאלה זו אנו נדרשים לקרוא שני מספרים שלמים, נניח a ו- b . לכן בשלב ראשון התוכנית תכלול (פרט להדפסת כותרת המגדירה את פעולתה) שתי פקודות קלט לקליטת שני מספרים שלמים. בגלל שמדובר בשאלה פשוטה, אין צורך בשלב של חישובים ואפשר לעבור לשלב האחרון של הדפסת התוצאות, כאשר החישוב יתבצע כחלק מפקודות ההדפסה. בפקודה ראשונה נדפיס את שני המספרים וכמה פעמים נכנס המספר השני בראשון (הדבר מושג ע"י שימוש בפקודה `//` - שמשמעותה חלוק של שלמים). פקודת ההדפסה השנייה מדפיסה את השארית והדבר מתבצע ע"י הפונקציה `%` (מודולו – או חישוב השארית).

התוכנית לדוגמה:

```
# Divide.py
# A program for reading two numbers and dividing them
# The program prints the result and the remainder

def main():
    print ("This program reads two numbers")
    print ("and divides them printing the rsukt and the reminder")
    print ()

    a = int(input("Enter first number: "))
    b = int(input("Enter second number: "))

    print ()
    print ("for ", b, "and", a, "b/a = ", b//a)
    print ("the reminder is: ", b%a)
```

6. בשאלה זו אנו נדרשים לקרוא שני מספרים שלמים, נניח a ו- b , אשר מייצגים את אורכי הצלעות של משולש ישר זווית. אפשר להבין מתוך השאלה שמדובר במספרים ממשיים ולכן בשלב ראשון התוכנית תכלול (פרט להדפסת כותרת המגדירה את פעולתה) שתי פקודות קלט לקליטת שני מספרים ממשיים. גם במקרה זה, בגלל שמדובר בשאלה פשוטה, אין צורך בשלב של חישובים ואפשר לעבור לשלב האחרון של הדפסת התוצאות, כאשר החישוב יתבצע כחלק מפקודת ההדפסה, אשר כוללת את אורכי שני הניצבים (הקלט שהתקבל מהמשתמש) ואת חישוב האורך של הצלע השלישית. בגלל שבכל משולש ישר זווית מתקיים משפט פיתגורס הקובע שסכום שטחי הריבועים הבנויים על הניצבים שווה לשטח הריבוע הבנוי על היתר, אזי האורך של היתר מתקבל מחישוב השורש הריבועי של סכום ריבועי הניצבים. בגלל שאנחנו משתמשים בפונקציה SQRT שהינה חלק מהספרייה המתמטית, עלינו גם לכלול פקודת `import math` בתחילת התוכנית, כדי לאפשר לתוכנית לגשת לפונקציות הכלולות בספרייה המתמטית.

התוכנית לדוגמה:

```
# Tri.py
# A program for reading two numbers that represnt two sides
# of a right triangle and calculates the third one
import math

def main():
    print ("This program reads two numbers that are two sides")
    print ("of a right triangle and calculates the third one")
    print ()

    a = float(input("Enter first number: "))
    b = float(input("Enter second number: "))

    print ()
    print ("The triagle sides are: ", a, b, math.sqrt ( a*a + b*b ) )
```

7. בשאלה זו אנו נדרשים לקבל מהמשתמש שני קלטים, נניח a ו- b . הקלט הראשון הינו מספר (ואפשר להבין מתוך השאלה שמדובר במספר שלם) ואילו הקלט השני הוא תו כלשהו. התוכנית אמורה להדפיס ריבוע אשר גודלו מיוצג ע"י הקלט הראשון ואילו התווים שמודפסים בריבוע מוגדרים ע"י הקלט השני. כרגיל, התוכנית מתחילה בהדפסה של כותרת קצרה המגדירה את פעולתה ולאחר מכן מבוצעות פקודות הקלט. המימוש יבוצע בעזרת שתי לולאות מקוננות. לולאה ראשונה מגדירה את מספר השורות שיש להדפיס ואילו הלולאה השנייה מגדירה את מספר התווים בכל שורה. בגלל שמדובר בריבוע (מספר התווים בשורה שווה למספר השורות). הלולאה הראשונה תבצע a פעמים ובתוכה תבצע הלולאה השנייה גם כן a פעמים. בתוך הלולאה הפנימית (זו אשר מדפיסה את התווים בשורה), בכל מחזור יודפס התו, כאשר אנו חייבים להבטיח שגם התו הבא יודפס באותה שורה. כזכור, כדי למנוע מפקודת ההדפסה לרדת לשורה הבאה (שזו ברירת המחדל בהדפסות), יש להוסיף את הפרמטר `end=""`. לאחר שהלולאה הפנימית התבצעה a פעמים (משמע סיימנו למלא את השורה), יש לכלול פקודת הדפסה ריקה (כדי להעביר את הסמן לשורה הבאה) ולקדם את מונה השורות (מחזור נוסף של הלולאה הראשונה). פירוש הדבר שאנחנו מתחילים למלא את השורה הבאה בריבוע שלנו. הדבר יחזור על עצמו a פעמים, עד אשר מלאנו a שורות, כאשר בכל שורה a תווים

התוכנית לדוגמה:

```
# Square.py
# A program for reading a number and a charater
# the program will print a square of length defined
# by the number. the square is made of the characters

def main():
    print ("This program reads number and a character")
    print ("and builds a square of the charters")
    print ()

    a = int(input("Enter square size: "))      # square size
    b = input("Enter character: ")            # fill character

    print ()
    for indx1 in range(a):                    # number of lines
        for indx2 in range(a):                # characters per line
            print (b, end="")
        print ()
    print ()
```


8. בשאלה זו אנו נדרשים לקבל מהמשתמש מספר אחד כקלט וברור מתוך השאלה שמדובר במספר שלם. התוכנית מתחילה בהדפסת כותרת קצרה וממשיכה בקליטת המספר. המימוש במקרה זה יתבסס על לולאה אשר תעבור על כל המספרים עד לקלט ותבדוק אם המנה המתקבלת מחלוקת המספר שהתקבל במונה הלולאה היא מספר שלם. אם כן, פירוש הדבר שמונה הלולאה הוא אחד המחלקים. למשל אם המספר שהתקבל הוא 15 ומונה הלולאה הוא 2, אזי בגלל ש $15/2$ אינו מספר שלם, ברור ש $2 -$ אינו מחלק של 15. לעומת זאת, אם מונה הלולאה הוא 3, אזי בגלל ש $15/3$ הוא מספר שלם, נובע מכך ש $3 -$ הוא חלק של 15. בקביעת גבולות הלולאה, ניתן לראות בנקל שאי אפשר להשתמש בפונקציה $\text{range}(a)$ סתם כך. אם היינו משתמשים כך בפונקציה, הערך הראשון שהיה מוכנס למונה הלולאה הוא אפס והרי אנחנו לא רוצים לבדוק על אפס. באופן דומה, אנחנו גם לא רוצים לבדוק על 1. לכן ניתן להשתמש בפונקציה range , אך לכלול את נקודת ההתחלה שלה (2). באופן דומה, אין טעם להמשיך ולבדוק את כל המספרים בתחום. אפשר לעצור את הלולאה לאחר שהיא הגיעה ל $a/2 -$, שהרי ברור שהמספרים מעבר לאמצע לא מתחלקים ללא שארית. לכן, הפרמטר השני של הפונקציה range יהיה $a/2+1$. אך, כאשר אנחנו מכניסים את $a/2+1$ לתוך הפונקציה range , מתעוררת בעיה אחרת... כל פקודת חילוק גורמת לתוצאה שהיא שבר ואילו הפונקציה range מצפה לקבל פרמטרים שהם מספרים שלמים. לכן, כדי שנוכל להשתמש בערך $a/2+1$ כחלק מהפונקציה range , עלינו להופכו למספר שלם ולכן, בתוך הפונקציה יופיע $\text{int}(a/2)+1$. גוף הלולאה כולל בדיקה לגבי השארית של החלוקה ואם השארית היא 0 (כלומר אין שארית), משמעות הדבר שהמונה הלולאה הוא מחלק של המספר שהוכנס. במקרה זה הוא מודפס בהמשך השורה, כולל הוספת הפרמטר אשר מורה לפקודת ההדפסה להישאר באותה השורה ($\text{end}=""$). רק לאחר שהלולאה הסתיימה (אנו יודעים בוודאות שלא יהיו יותר מחלקים), אפשר להדפיס שורה ריקה.

התוכנית לדוגמה:

```
# Divisors.py
# A program for reading a number and calculates its divisors

def main():
    print ("This program reads number and calculates its divisors")
    print ()

    a = int(input("Enter number: "))    # get number to be checked

    print ()
    print ("for ", a, "the divisors are:")
    for indx in range(2,int(a/2)+1):    #check all numbers from 2 to number/2
        if a%indx == 0:                # if no remainder it is a divisor
            print (indx, " ",end = "")
    print
```

9. זו שאלה קצת יותר מעניינת. אנו נדרשים לקרוא מחרוזת (נניח שנקרא לה a) ולבדוק אם היא פלינדרום (מחרוזת סימטרית הנקראת משמאל ומימין). התוכנית מתחילה בהדפסת כותרת קצרה ולאחר מכן כוללת פקודת קלט לקריאה של מחרוזת. את המימוש אפשר לבצע (כמו תמיד) בדרכים שונות. במקרה זה נשתמש בלולאה המתבצעת על איברי המחרוזת ובדוקת אם האיבר הראשון שווה לאיבר האחרון, לאחר מכן אם האיבר השני שווה לאיבר הלפני אחרון וכן הלאה עד לסוף המחרוזת. ניתן לראות שמעבר על כל המחרוזת הוא בעצם מיותר וכל שאנחנו צריכים הוא לעבור רק על חצי המחרוזת. ממילא החצי השני נבדק תוך כדי הבדיקה של החצי הראשון (שהרי כל איבר בחצי הראשון נבדק מול האיבר המתאים בחצי השני). לכן הלולאה אמנת תתחיל באיבר הראשון של המחרוזת, אבל תתקדם רק על אמצעה. את אמצע המחרוזת נחשב ע"י $(len(a)/2)$. אולם, בגלל שכל חלוקה מייצרת תוצאה שהיא מספר ממשי (float) ואילו הפונקציה range מחייבת מספרים שלמים, עלינו להמיר את המספר הממשי למספר שלם (תוך שימוש בפונקציה int). לצורך המימוש גם נגדיר משתנה sw אשר מאותחל ל-0. המשתנה מניח שהמחרוזת שקיבלנו היא פלינדרום, אלא אם כן נחליט שלא זה המצב. בתוך הלולאה אנו בודקים את האיברים הסימטריים (ראשון מול אחרון, שני מול לפני אחרון וכן הלאה). אם נמצא מקרה שבו האיברים הסימטריים אינם שווים, נשנה את ערכו של המשתנה sw. לאחר שנסיים את הלולאה, אם sw נשאר מאופס, משמע לא היה אף מקרה שבו איברים סימטריים היו שונים ולכן המחרוזת היא פלינדרום. לעומת זאת, אם בתום הלולאה sw לא מאופס, פירוש הדבר שהיה מקרה (לפחות אחד) שבו איברים סימטריים לא היו שווים ואז המחרוזת אינה פלינדרום. סיום התוכנית היא הדפסת הודעה מתאימה המבוססת על בדיקת המשתנה sw.

התוכנית לדוגמה:

```
# Palindrome.py
# A program for reading a text message and checks if it a palindrome

def main():
    print ("This program reads text message and checks if it is a palindrome")
    print ()

    a = input("Enter text: ")
    print ()

    sw=0 # set switch to zero
    for indx in range(int(len(a)/2)):
        if a[indx] != a[-1-indx]: # if not symmetric change switch
            sw=1
    if sw==0: # if switch is still zero, it is palindrome
        print (a, "is a palindrome")
    else: # switch was changed. It is not a palindrome
        print (a, "is not a palindrome")
```

חשוב לציין שאפשר לשכלל את התוכנית ולשפר את קריאותה ע"י:

- המשתנה sw בעצם מדמה משתנה בוליאני, לכן במקום להכניס לתוכו ערכים 0 ו-1 אפשר פשוט בשלה אתחול לרשום אותו כאמת (True) ובמהלך ביצוע מחזורי הלולאה, אם מתגלים איברים סימטריים שאינם שווים יש להפוך את המשתנה הבוליאני לשקרי (False). בסוף התוכנית יש לבדוק אותו ולהחליט על ההדפסה המתאימה.
- בתוך הלולאה, לאחר שמצאנו איברים סימטריים שאינם שווים, אין יותר צורך להמשיך ולבצע את הלולאה (שהרי ברור שהמחרוזת אינה פלינדרום). במקרה זה אפשר להוסיף פקודת break

התוכנית לדוגמה:

```
# Palindrome.py
# A program for reading a text message and checks if it a palindrome

def main():
    print ("This program reads text message and checks if it is a palindrome")
    print ()

    a = input("Enter text: ")
    print ()

    sw=True                                     # set switch to zero
    for indx in range(int(len(a)/2)):
        if a[indx] != a[-1-indx]:             # if not symmetric change switch
            sw=False
            break
    if sw:                                     # if switch is still zero, it is palindrome
        print (a, "is a palindrome")
    else:                                     # switch was changed. It is not a palindrome
        print (a, "is not a palindrome")
```

10. זו שאלה קצת יותר מעניינת ואפשר לפתור אותה בדרכים שונות. אמנם מדובר על קליטה של מספר ונראה הוא חייב להיות שלם, אך לאור העובדה שאנחנו נדרשים להדפיס את הספרות שבו בסדר הפוך, נראה שעדיף שנקלוט את המספר במחרוזת. בצורה זו, הגישה לכל הספרות שלו תהיה קלה. לכן, התוכנית תתחיל בהדפסה של משפט קצר המגדיר מה היא עושה ולאחר מכן קליטה של המספר. לצורך ההמרה נשתמש במחרוזת ריקה (נניח b), אשר אליה נוסיף את הספרות ובלולאה שבה מספר המחזורים נקבע ע"י אורך מחרוזת הקלט. הלולאה תתחיל מסוף המחרוזת ותנוע לתחילתה (בסדר הפוך מהרגיל) ובכל פעם תשלוף איבר אחד. ותצרף אותו אל המחרוזת החדשה (b). חשוב לציין שהגישה לפיתוח שהוצגה כאן מתאימה גם להיפוך מחרוזת כלשהי (לאו דווקא מספר) ולכן, אם השאלה מתמקדת רק במספרים, יש לבדוק שהקלט שקיבלנו הוא אכן מספרי. דרך אחת לעשות זאת, היא ע"י קליטה של ערך מספרי והפיכתו למחרוזת ע"י שימוש בפונקציה str.

התוכנית לדוגמה:

```
# Reverse.py
# A program for reading a number and printing it in a
# reverse order
# The program will reverse string messages as well

def main():
    print ("This program reads a number and prints it")
    print ("in a reverse order")
    print ()

    a = input("Enter text: ")
    b="" # a new empty string
    for indx in range(len(a)):
        b=b+a[-1-indx] # append string
    print (a, "in reverse order is: ", b)
```

11. זו שאלה מסוג שונה והיא מחייבת מעט מחשבה. נראה שיש להמיר את התווים שבמחרוזת לערכם המספרי (הערך ב – ASCII) ולכל התווים a-x (כולל) יש להוסיף 2. התווים הנותרים (y,z) יעברו המרה יחידנית. התוכנית מתחילה עם הדפסה קצרה לגבי יעודה ולאחר מכן קולטת את המחרוזת להצפנה (a). בנוסף, התוכנית מגדירה מחרוזת חדשה (b), אשר אליה יוכנסו הערכים החדשים. בשלב הבא מוגדרת לולאה אשר עוברת על כל איברי המחרוזת (מספר המחזורים נקבע ע"י האורך של מחרוזת הקלט – len(a)). גוף הלולאה כלל את הפקודות להמרה והן מורכבות מפקודת התניה (if) בעלת מספר תנאים. אם התו הנוכחי הוא z, יש להחליפו ל – b. ולא, אם התו הוא y יש להחליפו ב – a. ולא אם התו הוא רווח, יש להוסיף למחרוזת החדשה רווח (כזכור בשאלה זו איננו נדרשים להצפין את תווי הרווח). לבסוף, כל שאר התווים, ממורים לערכם המספרי (שימוש בפונקציה ord). לערך המספרי מוסיפים 2 ואת הערך החדש שקיבלנו אנו ממירים חזרה לתו (שימוש בפונקציה chr). השלב האחרון בתוכנית הוא הדפסה של המחרוזת המקורית והמחרוזת המוצפנת. חשוב לציין שהתוכנית משתמשת בידע לגבי ערכי ה – ASCII של התווים השונים ובעובדה שהערכים שלהם רציפים (אפשר לראות זאת בכל טבלת ASCII).

התוכנית לדוגמה:

```
# Encryption.py
# A program for encrypting messages:
# a->c, b->d,...y->a, z->b

def main():
    print ("This program encrypts messages")
    print ("a->c, b->d,...y->a, z->b")
    print ()

    a = input("Enter text message: ")

    b=""                # a new empty string

    for indx in range(len(a)):
        if a[indx]=="z":    # encrypt "z"
            b=b+"b"
        elif a[indx]=="y":  # encrypt "y"
            b=b+"a"
        elif a[indx]==" ":  # if blank leave as is
            b=b+" "
        else:
            b=b+chr(ord(a[indx])+2)    # encrypt all others by adding
                                         # 2 to their ASCII value

    print (a, "encrypted is: ", b)
```

12. למרות המלל הארוך, זו שאלה פשוטה יחסית. התוכנית תכלול הדפסה של שורת כותרת קצרה ולאחר מכן קריאת מספר בעל 16 ספרות. הפשוט ביותר הוא לקרוא את המספר כמחרוזת ואז ניתן לבדוק בקלות שאורכה מתאים (16 ספרות). לאחר מכן יש להגדיר משתנה לסכום הביקורת. בלולאה יש לעבור על כל ספרות המספר ואם הספרה במיקום אי זוגי (כלומר כתובתה היא זוגית $0,2,4,\dots$), אזי יש לחבר אותה לסכום הביקורת. אם כתובתה אי-זוגית (משמע מדובר בספרה במיקום זוגי) יש לכפול אותה ב-2. אם הסכום, לאחר ההכפלה קטן מ-10 יש לחברו אל סכום הביקורת. לעומת זאת, אם הסכום בעל שתי ספרות (גדול מ-9), יש להוריד ממנו 9 ולחבר לסכום הביקורת. חשוב לציין שהאלגוריתם, במקרה של מספר גדול מ-9, הוגדר בצורה שונה במקצת, אך התוצאה זהה. לפי האלגוריתם, אם המכפלה גדולה מ-9 יש לחבר את סכום הספרות. במילים אחרות, למשל אם המכפלה היא 16, אזי לסכום הביקורת יש לחבר את הסכום של 6 ו-1 (7). סכום זה מתקבל גם ע"י הפחתת 9 מהמכפלה שכן $7=16-9$. לאחר שחישבנו את סכום הביקורת, אפשר לבדוק אם הוא כפולה של 10 (ע"י שימוש בפונקציה %), אם כן, אזי המספר חוקי ואם לא המספר אינו חוקי. כדי לעקוף את הבלבול במיקום הספרות (ספרה במיקום זוגי, אך כתובת אי זוגית), אפשר לכתוב את הלולאה כאשר היא מתחילה מ-1 וממשיכה עד לאורך המחרוזת ועוד אחד. בצורה זו מיקום הספרה הוא אחד פחות ממונה הלולאה, אך המיקום הזוגי מתאים למונה זוגי ואילו מיקום אי זוגי מתאים למונה אי זוגי (הדבר לא משנה במהות התוכנית, אך אולי מקל על ההבנה). נקודה חשובה נוספת היא העובדה שיש להפוך את התווים שבמחרוזת למספרים (ע"י הפונקציה int) לפני שנוכל להשתמש בהם החישובים.

התוכנית לדוגמה:

```
# visa.py
# A program for verifying VISA numbers

def main():
    print ("This program verifies VISA numbers")
    print ()

    a = input("Enter number (16 digits): ")

    checksum=0
    for indx in range(1,len(a)+1):
        if indx%2==1:      # Odd location add number
            checksum = checksum + int(a[indx-1])
        else:              # even location multiply
            if int(a[indx-1])*2>=10:
                checksum = checksum + int(a[indx-1])*2-9
            else:
                checksum = checksum+ int(a[indx-1])*2

    if checksum%10==0:     # divides by 10
        print (a, "is a valid number")
    else:
        print (a, "is not a valid number")
```

13. זו שאלה קלה. למרות שבשאלה מוגדרים שלושה מספרים (המייצגים ווקטורים), כדאי לקרוא אותם כמחרוזות משום שאז קל מאוד להגיע לכל אחת מהספרות. התוכנית מתחילה בהדפסת יעודה וממשיכה לקרוא את שלוש המחרוזות. יש להגדיר רשימה, אשר בשלב הראשון היא עדיין ריקה. לאחר מכן בלולאה מחברים את האיבר המתאים של a לאיבר המתאים של b ומהסכום מחסרים את האיבר המתאים של c . יש לשים לב שלפני שנוכל לבצע פעילות מתמטית כלשהי (חיבור או חיסור) יש להפוך את המחרוזת למספרים (שימוש ב-`int`). את תוצאת החישוב יש להוסיף (ע"י הפונקציה `append`) לרשימה, כך שבכל מחזור של הלולאה היא גדלה באיבר אחד נוסף. בשלב האחרון יש פשוט להדפיס את הרשימה הכוללת את הווקטור החדש.
התוכנית לדוגמה:

```
# Vector.py
# A program for vector calculations

def main():
    print ("This program reads 3 vectors a,b,c")
    print ("and produces a list of a+b-c")
    print ()

    a = input("Enter first vector: ")
    b = input("Enter first vector: ")
    c = input("Enter first vector: ")

    d=[]          # a new (empty) list
    for indx in range(len(a)):
        d.append(int(a[indx])+int(b[indx])-int(c[indx]))
    print (d)
```

14. למרות השאלה הארוכה, הפיתרון שלה מובנה ולא קשה במיוחד. התוכנית מתחילה בהדפסת יעודה וממשיכה לקרוא את שלוש הרשימות (a, b, c). עקב העובדה שמדובר ברשימות (בהן האיברים מופרדים בפסיקים, אין אפשרות לקרוא את הקלט כמספרים, אלא כמחרוזת. לכן לאחר הקריאה וכדי להעלים את הפסיקים נשתמש בפונקציה split שתפרק את המחרוזת לרשימה של איברים. תו ההפרדה במקרה זה יהיה פסיק. הרשימות החדשות (atxt, btxt, ctxt) הן רשימות המכילות איברים שהם עצמם מחרוזות ולכן לפני שנוכל השתמש בהם בפעולות חשבונאיות נצטרך להפכם למספרים. יש להגדיר רשימה חדשה (d), אשר בשלב הראשון היא עדיין ריקה ולתוכה יוכנסו האיברים המתאימים על פי האלגוריתם שהוגדר. לאחר מכן בלולאה המתבצעת על איברי ctxt מתבצע האלגוריתם. יש להיזהר שלא לקבוע את מספר מחזורי הלולאה לפי אורכה של המחרוזת a שכן היא ארוכה מהדרוש (מכילה גם פסיקים) ולכן מספר מחזורי הלולאה נקבע על פי מספר האיברים ב atxt (אשר לפי תנאי השאלה זהה למספר איברי btxt וכן מספר איברי ctxt). בתוך הלולאה מתבצע האלגוריתם המבוקש שהוא:

- אם האיבר המתאים ב – c הוא "0", אזי יש להוסיף לרשימת התוצאה 0.
- אם האיבר המתאים ב – c הוא "1", יש להוסיף לרשימת התוצאה את האיבר המתאים מתוך הרשימה הראשונה.
- אם האיבר המתאים ב – c הוא "2", יש להוסיף לרשימת התוצאה את האיבר המתאים מתוך הרשימה השנייה.
- אם האיבר המתאים ב – c הוא "3", יש להוסיף לרשימת התוצאה את סכום האיברים המתאימים מ מתוך הרשימה הראשונה והשנייה.

בכל המקרים, במהלך ההשוואה יש להשתמש בתווים, משום שהרשימה ctxt מכילה תווים ולא מספרים. הרשימה d (התוצאה), לעומת זאת מכילה נתונים מספריים. ולכן כאשר מעבירים איברים מ – atxt או btxt יש להופכם קודם למספרים (בעזרת int). הדבר, נכון, כמובן גם כאשר יש לחבר את שני האיברים. בשלב הסופי, מודפסות שתי רשימות המקור, רשימת הבקרה והרשימה שהינה תוצאת הריצה. פורמט ההדפסה של רשימת התוצאה שונה (הוא מודפס כרשימה, בעוד כל היתר מודפסים כמחרוזת). אם יש צורך להדפיס את כולם בפורמט זהה, יהיה צורך לטפל ברשימת התוצאה (למשל ע"י הפיכתה למחרוזת וזאת לפני שהיא מודפסת).

התוכנית לדוגמה:

```
# Vectors.py
# A program for vector calculations

def main():
    print ("This program reads 3 vectors a,b,c")
    print ("and produces a list of combined list")
    print ("based on c - the control list as well as a and b")
    print ()

    a = input("Enter first list: ")
    b = input("Enter second list: ")
    c = input("Enter control list:")

    # convert txt message into a list of texts

    atxt = a.split(",")
    btxt = b.split(",")
    ctxt = c.split(",")

    d = []
    for i in range(len(atxt)):

    # produce new list
        if ctxt[i] == "0":
            d.append(0)
        elif ctxt[i] == "1":
```



```
        d.append(int(atxt[i]))
    elif ctxt[i] == "2":
        d.append(int(btxt[i]))
    else:
        d.append(int(atxt[i]) + int(btxt[i]))

# print results

print ("For :\n",a, "\n",b,"\n",c,"\n","the result vector is: \n",d)
```

15. בשלב ראשון התוכנית מדפיסה כותרת המסבירה בקצרה את מטרת התוכנית. לאחר מכן מבקשת מהמשתמש את גודל הלוח. למרות שלא צוין, ברור שמדובר במשתנה מספרי (שלם). הלוח עצמו מחושב ע"י שתי לולאות מקוננות. לולאה ראשונה נועדה להגדיר את מספר השורות בלוח והלולאה השנייה נועדה להגדיר את הערכים בכל שורה. בקביעת מספר המחזורים יש לשים לב שהפונקציה range בלולאה צריכה להתבצע עד $n+1$. אם נרשום range(n), תחסר לנו השורה האחרונה. בגוף הלולאה נכלול את פקודות הפורמט, אשר מאפשרת לקבוע את מספר התווים שמוקצים לכל מספר ולפי תנאי השאלה נציב שם 5. בנוסף, בפקודת ההדפסה עלינו להבטיח שכל התוצאות מודפסות באותה שורה (ע"י שימוש בתוספת end=""). רק בתום הלולאה הפנימית יש לעבור לשורה הבאה.

התוכנית לדוגמה:

```
# Multiplication.py
# A program for printing a multiplication table

def main():
    print ("This program prints a multiplication table")
    print ("The size is define by the user input")
    print ()

    n = int(input("Enter table size: "))

    for i in range(1,n+1):
        for j in range(1,n+1):
            print ("{:0:5}".format (i*j), end="")
        print ()
    print ()
```

16. בשלב ראשון התוכנית מדפיסה כותרת המסבירה בקצרה את מטרת התוכנית. לאחר מכן התוכנית קוראת את שני הקלטים. הקלט הראשון המייצג את המספר נקרא כמחרוזת, כדי לאפשר גישה קלה לאיברים השונים. הקלט השני הוא מספר שלם. התוכנית מגדירה משתנה חדש dec שאמור לכלול את הערך העשרוני המחושב. הלולאה העוברת על כל איברי המספר שהוכנס, מחושב הערך העשרוני. האיבר הימני ביותר מוכפל בבסיס בחזקת 9, המספר שלשמאלו בבסיס בחזקת 1 וכך הלאה עד לאיבר השמאלי ביותר המוכפל בבסיס בחזקת מספר האיברים פחות אחד. החזקה המתאימה לכל איבר מחושבת ע"י מספר איברי המספר, פחות אחד ופחות מונה הלולאה (בדוק!). בכל מחזור מתווסף הערך המתאים למשתנה ובסוף הלולאה מודפס הערך העשרוני.

התוכנית לדוגמה:

```
# Baseconversion.py
# A program for migrating non decimal numbers to decimal

def main():
    print ("This program accepts a number and its base")
    print ("and migrates it to decimal")
    print ()

    a = input("Enter a number: ")
    base = int(input("Enter its base: "))

    dec = 0

    for i in range(len(a)):
        dec = dec + int(a[i])*base**(len(a)-1-i)
    print (a, "in base", base, "equals", dec, "decimal")
```

17. בשלב ראשון התוכנית מדפיסה כותרת המסבירה בקצרה את מטרתה. לאחר מכן התוכנית קוראת את הקלט. הקלט הוא רשימה של ערכים מספריים שאינם ממוינים. בגלל שהם מופרדים בפסיקים, אין ברירה אלא לקרוא אותם כמחרוזת (a). את המחרוזת מפרקים לרשימה של איברים תוך שימוש בפונקציה split, כאשר איבר ההפרדה הוא פסיק. קיבלנו רשימה של איברים שכל אחד מהם הוא מחרוזת. בשלב זה אפשר לממש את אלגוריתם מיון הבועות. לולאה ראשונה מתבצעת על כל איברי הרשימה (מאפס ועד len(lst)). הלולאה שנייה (המקוננת) מתחילה מהאיבר השני (שנמצא בכתובת 1) וממשיכה עד לאורך המחרוזת פחות מונה הלולאה הראשונה. בצורה זו, במחזור הראשון, כאשר מונה הלולאה הראשונה מאופס, הלולאה השנייה ממשיכה עד סוף הרשימה. במחזור השני, כאשר מונה הלולאה הראשונה הוא אחד, הלולאה השנייה ממשיכה עד לסוף הרשימה פחות אחד. וכן הלאה, בכל מחזור של הלולאה הראשונה, הלולאה השנייה מתקצרת באיבר אחד (כאמור האיבר שכבר נמצא במקומו). גוף הלולאה כולל השוואה בין זוג איברים ואם האיבר הראשון גדול מהשני, מתבצע חילוף ביניהם (הצלבת ערכים). לאחר ששתי הלולאות סיימו, הרשימה ממוינת. החלק האחרון בתוכנית כולל לולאה נוספת אשר עוברת על כל איברי האשימה ומשנה את המחרוזות למספרים, כדי שהדפסת הרשימה הממוינת תכלול מספרים ולא מחרוזות.

התוכנית לדוגמה:

```
# Bubblesort.py
# A sort program using bubble sort

def main():
    print ("This program accepts a list of numbers")
    print ("and sorts it using a bubble sort algorithm")
    print ()

    a = input("Enter a list of numbers: ")
    lst = a.split(",") #split string into list of text items

    for i in range(len(lst)):
        for j in range(1,len(lst)-i):
            if int(lst[j-1]) > int(lst[j]):
                lst[j-1],lst[j] = lst[j],lst[j-1]

    # convert the text items into numbers

    for k in range(len(lst)):
        lst[k] = int(lst[k])

    print ("sorted:", lst)
```

18. חישוב שטח הריבוע שאינו מכוסה ע"י העיגול החסום נתון ע"י ההפרש שבין שטח הריבוע (אורך הצלע בריבוע) ושטחו של העיגול הנתון ע"י רדיוס בריבוע כפול π . לכן, התוכנית מתחילה בהדפסה של כותרת קצרה. לאחר מכן בלולאה של תשעה מחזורים יש לחשב את שטחו של הריבוע ולהפחית ממנו את שטחו של העיגול. יש לשים לב שרדיוס העיגול הוא מחצית מאורך הצלע. לצורך חישוב שטחו של העיגול נשתמש ב π – מתוך הספרייה המתמטית ולכן בתחילת התוכנית גם נצטרך להוסיף את "יבוא" היכולות של הספרייה (ע"י שימוש בפקודה `import math`). נקודה נוספת שיש לשים לב אליה, היא הדרישה שהסדרה המודפסת צריכה להיות בדיוק של שלוש ספרות מימין לנקודה העשרונית. הדבר יבוצע ע"י הוספת יכולת עריכה לפקודת ההדפסה (שימוש ב `format` – שימוש ב `format`).

התוכנית לדוגמה:

```
# roundSquare.py
# A program to calculate the square area not occupied by a circle

import math

def main():
    print ("This program calculates the square area not occupied")
    print ("by a circle. This is done for lengths: 1,2,3...9")
    print ()

    for i in range(1,10):
        print ("For side =", i, "area is: {0:.3f}".format(i*i-math.pi*(i/2)*(i/2)))
```

19. בשלב ראשון לפני שניגש לפתור את השאלה נשים לב שמספר הטבעות תלוי במספר המספרים שמופיע בקלט. n מספרים מייצרים $(n-1)$ טבעות. שטחה של כל טבעת מחושב ע"י ההפרש שבין שטחו של העיגול הגדול בטבעת לשטחו של העיגול הקטן בטבעת. עכשיו נוכל להתחיל בכתיבת התוכנית. התוכנית מתחילה בהדפסה של כותרת קצרה. לאחר מכן קוראת את רשימת המספרים שהוקלדו. בגלל שמדובר ברשימה (הכוללת גם פסיקים), עלינו לקרוא אותה כמחרוזת. לאחר מכן אפשר לפצל אותה (בעזרת הפקודה `split`), כאשר תו ההפרדה הינו פסיק). כתוצאה מהפיצול, קיבלנו רשימה. נגדיר משתנה (`sum`) המאותחל לאפס ואשר אליו נסכם את שטחי הטבעות. בלולאה שתבוצע על כל איברי הרשימה (למעט האחרון), נחשב את השטח של כל טבעת. במהלך חישוב השטח יש לקחת בחשבון שכל איבר ברשימה הינו מסוג מחרוזת ולכן לפני שנוכל לבצע עליו פעולות חשבוניות, יש להפוך אותו למספרי. לפי תנאי השאלה ברור שהמספרים מייצגים מספרים שלמים ולכן ניתן להשתמש בפונקציה `int`. השטח שחושב יודפס וכן יסוכם אל המשתנה `sum`. בתום הלולאה ולאחר שהדפסנו את שטחי הטבעות, נשאר להדפיס גם את השטח הכולל של כל הטבעות גם יחד. יש לשים לב שפקודת ההדפסה חייבת לכלול עיצוב שכן אנו נדרשים להקפיד על מספר הספרות לימין הנקודה העשרונית. גם במקרה זה, לצורך חישוב שטחו של העיגול נשתמש ב `pi` מתוך הספרייה המתמטית ולכן בתחילת התוכנית גם נצטרך להוסיף את "יבוא" היכולות של הספרייה (ע"י שימוש בפקודה `import math`).

התוכנית לדוגמה:

```
# ringsArea.py
# A program to calculate the ring areas

import math

def main():
    print ("This program calculates the ring area")
    print ("for rings defined by a list on integers")
    print ()

    a = input("enter the list of number: ")
    lst = a.split(",")

    sum = 0

    for i in range(len(lst)-1):
        tmp = math.pi*int(lst[i])**2-math.pi*int(lst[i+1])**2
        print ("For ring no.", i+1, "area is: {0:.3f}".format(tmp))
        sum = sum + tmp
    print ("The total rings area is: {0:.2f}".format(sum))
```

20. זו תוכנית פשוטה המחייבת מעט מחשבה רק לגבי ההדפסה. התוכנית מתחילה בהדפסה של כותרת קצרה. לאחר מכן קוראת את שני המספרים השלמים (a, b). בגלל שאפשר להניח שהמספר הראשון קטן מהשני, אין צורך לבדוק את התנאי הזה. נגדיר משתנה (n) המייצג את החזקה ונאתחל אותו לאחד. בלולאת while, אשר מתקיימת כל זמן שהמספר הראשון (a) בחזקת n קטן מהמספר השני (b) נקדם את החזקה (n) באחד. הלולאה מסתיימת, כאשר התנאי לא מתקיים ואז אפשר להדפיס את התוצאה. החזקה המתאימה היא אחת פחות מהחזקה המחושבת (n-1). כדי להדפיס את המספר בחזקה המתאימה ואת תוצאת החישוב ללא רווחים, יש להפוך את המספרים למחרוזות ואז ניתן יהיה להשתמש בסימן "+", כדי להצמיד את המחרוזות האחת אל השנייה.

התוכנית לדוגמה:

```
# Powewr.py
# A program to calculate the max power of a number that is less than another
number

def main():
    print ("This program reads two numbers and calculates the max power")
    print ("of the first number that is still lower than the second number")
    print ()

    a = int(input("enter first number: "))
    b = int(input("Enter second number: "))

    n = 1
    while a**n < b:
        n = n + 1
    print ("The power is:", n-1, "("+ str(a)+"**"+str(n-1)+"=" + str(a**(n-1))+")")
```

21. התוכנית מתחילה בהדפסה של כותרת קצרה. לאחר מכן קוראת את שני הקלטים (מחרוזת (a) ומספר שלם (n)). מגדירים מחרוזת חדשה (b), אשר בשלב זה היא עדיין ריקה. בלולאה המתחילה מאחד וממשיכה עד אורך המחרוזת ועוד אחד, עוברים על כל איברי המחרוזת. בכל מחזור בלולאה מעתיקים את האיבר המתאים אל המחרוזת החדשה. במחזור שהוא כפולה של המספר השלם (n), משכפלים את האיבר המתאים (ע"י כך שמעתיקים אותו פעם נוספת). הסיבה שהלולאה מתחילה מאחד וממשיכה עד לאורך המחרוזת ועוד אחד, היא כדי שנוכל לבדוק אם מס' המחזור הוא כפולה של המספר. ללא שינוי זה ובגלל שאפס הוא כפולה של כל מספר (ובפרט המספר השלם שקיבלנו) התוכנית הייתה משכפלת גם את האיבר הראשון. השלב האחרון בתוכנית כולל את הדפסת המחרוזת המקורית ואחריה הדפסת המחרוזת החדשה.

התוכנית לדוגמה:

```
# modifyString.py
# A program to modify a string based on a number

def main():
    print ("This program reads two a string and a number (n)")
    print ("it duplicates every n-th item in the string")
    print ()

    a = input("Enter first number: ")
    n = int(input("Enter second number: "))
    b = "" #new string
    for i in range(1,len(a)+1):
        b = b + a[i-1]
        if i%n==0:
            b = b + a[i-1]

    print ("The original string was:",a, "\nThe modified string is: ",b)
```

22. התוכנית מתחילה בהדפסה של כותרת קצרה. לאחר מכן קוראת את הקלט (מחרוזת a המכילה רשימה מספרי BCD). את המחרוזת מפרקים (ע"י שימוש בפונקציה split) לרשימה של מספרי BCD (אלא שכל אחד הוא מחרוזת מה שיחייב המרתו למספר לפני שניתן יהיה להשתמש בו לפעולות מתמטיות). מגדירים משתנה חדש dec שבשלב זה עדיין מאופס והוא אמור לכלול את הערך העשרוני החדש. בלולאה העוברת על כל איברי הרשימה, מפרקים כל איבר לסיביות הבונות אותו (bin1 הוא משתנה המייצג את הסיבית הימנית bin2 מייצג את הסיבית שלשמאלה וכן הלאה bin4 ו bin8 הם משתנים המייצגים את שאר הסיביות במספרי ה-BCD). בכל מחזור בלולאה יש לכפול את ערכו של המשתנה dec פי 10 שכן אנו מתקדמים אל הספרה הימנית יותר. בהמשך הלולאה מכפילים את הערכים הבינאריים במשתנים (bin1 מוכפל כפול 1, bin2 מוכפל כפול 2 וכן הלאה) ומחברים את התוצאה למשתנה dec. בסיום הלולאה מודפסת הרשימה של מספרי BCD והערך העשרוני שחושב.

התוכנית לדוגמה:

```
# BCD2Dec.py
# A program to convert a BCD list to decimal

def main():
    print ("This program reads a BCD list and converts it to decimal")
    print ()

    a = input("Enter BCD string: ")
    lst = a.split(",")

    dec = 0    # decimal value

    for i in range(len(lst)):
        dec = dec * 10
        bin1 = int(lst[i][3])
        bin2 = int(lst[i][2])
        bin4 = int(lst[i][1])
        bin8 = int(lst[i][0])
        dec = dec + bin1 + 2*bin2 + 4*bin4 + 8*bin8

    print ("The original BCD number was:",a, "\nThe decimal values is: ",dec)
```


23. התוכנית מתחילה בהדפסה של כותרת קצרה. לאחר מכן קוראת את הקלט (מחרוזת a המכילה רשימה מספרים). את המחרוזת מפרקים (ע"י שימוש בפונקציה split) לרשימה של מספרים (אלא שכל אחד הוא מחרוזת מה שיחייב המרתו למספר לפני שניתן יהיה להשתמש בו לפעולות מתמטיות). מגדירים רשימה חדשה שבשלב זה עדיין ריקה ואליה יוכנסו האיברים המחושבים. כדי שהאיברים ברשימה החדשה יהיו באותו פורמט כמו איברי הרשימה המקורית, לאחר החישוב, המספרים יומרו חזרה למחרוזת. ראשית נחשב את האיבר הראשון של הרשימה החדשה. איבר זה שווה לסכום האיבר השני והאיבר האחרון של הרשימה המקורית. בשלב הבא ותוך שימוש בלולאה נחשב את שאר איברי הרשימה החדשה (פרט לאחרון). פירוש הדבר שהלולאה תתחיל באחד (במקום אפס) ותסתיים באיבר הלפני אחרון. בגוף הלולאה יתבצע החיבור של האיבר שמספרו נתון ע"י מונה הלולאה פחות אחד לאיבר שמספרו נתון ע"י מונה הלולאה ועוד אחד. לאחר שהלולאה הסתיימה, נותר לנו רק לחשב את האיבר האחרון ברשימה החדשה. איבר זה הוא סכום האיבר הראשון והאיבר הלפני אחרון. השלב הסופי כולל הדפסת הרשימה המקורית והרשימה המעודכנת.

התוכנית לדוגמה:

```
# circularAdd.py
# A program to add neighboring cells

def main():
    print ("This program reads a list of numbers and produces")
    print ("a new list with sums of neighboring cells")
    print ()

    a = input("Enter a string: ")
    lst = a.split(",")

    b = []          # new array
    b.append(str(int(lst[1])+int(lst[-1])))

    for i in range(1,len(lst)-1):
        b.append(str(int(lst[i-1])+int(lst[i+1])))

    b.append(str(int(lst[-2])+int(lst[0])))
    print ("The original list was:",lst, "\nThe modified list is: ",b)
```

24. התוכנית מתחילה בהדפסה של כותרת קצרה. לאחר מכן קוראת את הקלט (מחרוזת a המכילה רשימה מספרים). את המחרוזת מפרקים (ע"י שימוש בפונקציה split) לרשימה של מספרים (אלא שכל אחד הוא מחרוזת). מגדירים רשימה חדשה שבשלב זה עדיין ריקה ואליה יוכנסו האיברים הייחודיים. ראשית נעתיק את האיבר הראשון שהרי ברור שהוא ייחודי. בשלב הבא ותוך שימוש בלולאה המתחילה מהאיבר השני (שהרי את הראשון כבר העברנו) נבדוק אם האיברים ייחודיים. נגדיר משתנה (sw) שמאותחל לאפס ואשר אמרו לבדוק אם המספרים ייחודיים. נגדיר לולאה נוספת אשר תרוץ מהתחלת המחרוזת ועד לאיבר שנבדק כרגע (מונה הלולאה העליונה) ולכל איבר נבדוק אם הוא שווה לאיבר הנבדק. אם יסתבר שכן, פירוש הדבר שזה איבר כפול ויש להתעלם ממנו. הלולאה תשנה את sw לציין שנמצא כפול ותבצע break שמשמעותו סיום הלולאה (שהרי כבר מצאנו שאיבר זה כפול). בסוף הלולאה מתבצעת בדיקה על sw ואם הוא נשאר אפס, משמע לא מצאנו איבר כפול ואז ניתן להעביר את האיבר הנבדק לתוך הרשימה החדשה (הוא ייחודי). לעומת זאת אם sw לא שווה אפס, האיבר כפול ויש להתעלם ממנו. בשלב הסופי מודפסת הרשימה המקורית והרשימה החדשה.

התוכנית לדוגמה:

```
# EliminateDUPs.py
# A program to eliminate duplicate entries

def main():
    print ("This program reads a list of numbers and produces")
    print ("a new list with only unique (non duplicate) numbers")
    print ()

    a = input("Enter a string: ")
    lst = a.split(",")

    b = []      # new array
    b.append(lst[0])

    for i in range(1,len(lst)):
        sw = 0
        for j in range(i):
            if lst[j]==lst[i]:
                sw = 1
                break
        if sw == 0:
            b.append(lst[i])

    print ("The original list was:",lst, "\nThe modified list is: ",b)
```

25. התוכנית מתחילה בהדפסה של כותרת קצרה. לאחר מכן קוראת את הקלט (מחרוזת a המכילה רשימה מספרים). את המחרוזת מפרקים (ע"י שימוש בפונקציה split) לרשימה של מספרים (אלא שכל אחד הוא מחרוזת). מגדירים רשימה חדשה שבשלב זה עדיין ריקה ואליה יוכנסו סכומי הביניים. ראשית נעתיק את האיבר הראשון שעבורו לא צריך לסכם דבר. בשלב הבא ותוך שימוש בלולאה המתחילה מהאיבר השני, נסכם את האיברים. הדבר מתבצע ע"י לולאה מקוננת, אשר מתחילה באיבר הראשון וממשיכה עד האיבר הנוכחי (מונה הלולאה הראשונה). בכל מחזור נגדיר משתנה מאופס ונסכם אליו את כל האיברים עד לשלב זה. לאחר סיום הלולאה הנמוכה (זו המסכמת את כל האיברים עד לאיבר הנבדק) ניתן להעביר את סכום הביניים לתוך הרשימה החדשה. בשלב הסופי מודפסת הרשימה המקורית והרשימה החדשה.

התוכנית לדוגמה:

```
# EliminateDUPs.py
# A program to eliminate duplicate entries

def main():
    print ("This program reads a list of numbers and produces")
    print ("a new list with only unique (non duplicate) numbers")
    print ()

    a = input("Enter a string: ")
    lst = a.split(",")

    b = []          # new array
    b.append(str(lst[0]))

    for i in range(1, len(lst)):
        tmp = 0
        for j in range(i+1):
            tmp = tmp + int(lst[j])
        b.append(str(tmp))

    print ("The original list was:",lst, "\nThe modified list is: ",b)
```

26. התוכנית מתחילה בהדפסה של כותרת קצרה המסבירה מה מטרתה. לאחר מכן קוראת את הקלט (מספר n). בגלל שהמספר חייב להיות בין 2 ל 9 (כולל) התוכנית כוללת לולאת while אשר מוגדרת כלולאה אינסופית (מתבצעת כל זמן שהתנאי ממתקיים, אולם התנאי שהוגדר (True) מתקיים תמיד). בתוך הלולאה מתבצעת בדיקה. אם הערך המספרי (n) שהוכנס גדול מאחד וגם קטן מעשר (במילים אחרות הוא ערך תקין), אזי הפקודה המתבצעת היא break שמשמעותה יציאה מהלולאה והמשך בפקודה שלאחריה. אם הערך אינו תקין, חוזרים לבקש את הקלט מחדש. המספר שהתקבל הוא הבסיס שעבורו יש להגדיר את כל המספרים בעלי שלוש הספרות. אם היה מדובר בבסיס עשרוני, הלולאה הייתה יכולה להיות פשוטה (לולאה אחת בלבד), אולם בגלל שמדובר בבסיס שונה, עלינו לנהל את קידום הספרות השונות. לצורך כל יש להגדיר שלול שלולאות שונה (כנגד שלוש הספרות שנמצאות במספר שעלינו להדפיס). הלולאה הראשונה (העליונה) אחראית לקדם את הספרה השמאלית ביותר. הלולאה השנייה אחראית לקידום הספרה השנייה ואילו הלולאה השלישית אחראית לקידום הספרה הימנית. אם היה מדובר בבסיס 10, אזי כל לולאה הייתה צריכה להתבצע עשרה מחזורים. במקרה שלנו הבסיס שונה ולכן כל לולאה צריכה להתבצע מספר מחזורים כפי שמצוין בבסיס. למשל אם הביב הוא 3, אזי האפשרויות לספרות בבסיס הן: 0,1,2,3. כל שלוש אפשרויות). כמו בכל מצב של לולאות מקוננות, הלולאה הפנימית מתקדמת מהר יותר מאחרות. לולאה זו מקדמת את ערכי הספרה הימנית. לאחר שעברה את n המחזורים, הלולאה שלשמאלה מתקדמת באחד ואז הלולאה הימנית מתחילה מחזור סיבוב חדש. באופן דומה, הלאחר שהלולאה האמצעית השלימה את כל האפשרויות, הלולאה השמאלית תתקדם באחד. בצורה זו מתקבלת רשימה הכוללת את כל הספרות בבסיס שנבחר. למשל עבור הבסיס שלוש, הרשימה תכלול:

000,001,002,010,011,012,020,021,022,100,101,...222

השלב האחרון בתוכנית כולל את הדפסת המספרים שחושבו. אם נדפיס אותם כמספרים, אזי יהיה רווח כלל שתי ספרות. כדי למנוע את הרווח התוכנית ממירה את המספרים למחרוזות ומדפיסה אותן כאשר הן צמודות.

התוכנית לדוגמה:

```
# countBase.py
# A program to count iv various bases

def main():
    print ("This program reads a numbe (2-9) that represents a base")
    print ("and produces all 3 digit number in that base")
    print ()

    while True:
        n = int(input("Enter number (2-9): "))
        if n>1 and n<10: break

    for d1 in range(n):
        for d2 in range(n):
            for d3 in range (n):
                print (str(d1)+str(d2)+str(d3))
```

חזור לתוכן

תשובות

כתיבת פונקציות

1. זו דוגמה פשוטה. הפונקציה אמורה לקבל מערך שהוא מסוג מחרוזת ולכן כדי להופכה נשתמש במחרוזת חדשה ריקה ובלולאה אשר תעבור על כל איברי מערך הקלט (החל מהסוף) וכל איבר יתווסף אל המערך החדש. בסיום הלולאה, הפונקציה תחזיר את המערך החדש כתשובה.

תשובה כדוגמה:

```
def reverse(a):
    b=""
    for i in range (len(a)):
        b = b + a[-1-i]
    return b
```

בפיתרון שתואר, הלולאה מתקדמת מאפס ועד לסוף המחרוזת והאיברים הנשלפים הם מהסוף. אפשר, כמובן גם לכתוב את הלולאה בצורה הפוכה, כאשר היא מתחילה מהאיבר האחרון (אורך המחרוזת פחות אחד) וחוזרת אחורנית לכיוון ההתחלה. את ההתחלה יש להגדיר ע"י מינוס אחד משום שבלולאה אמנם רשום המספר, אלא שהיא נעצרת איבר אחד לפניו. במצב כזה, מונה הלולאה מגדיר את סדר האיברים שיש לשלוף מהמחרוזת הישנה.

תשובה לדוגמה:

```
def reverse(a):
    b=""
    for i in range (len(a)-1,-1,-1):
        b = b + a[i]
    return b
```

2. מימוש הפונקציה מתבצע ע"י שתי לולאות מקוננות. הלולאה העליונה אחראית על מספר השורות שיש להדפיס ואילו הלולאה הפנימית מדפיסה את ה - * . יש לשים לב שמספר ה - * בכל שורה שונה והדבר מושג ע"י שינוי הפרמטרים של הלולאה הפנימית. בכל שורה היא מתחילה מאפס, אבל ממשיכה רק עד לערכו של מונה הלולאה העליונה (i). פקודת ההדפסה הפנימית, כמובן צריכה לדאוג שכל ה - * השייכות לאותה שורה, אכן יודפסו באותה שורה והדבר מושג ע"י הפרמטר של end="", המבטיח שפקודת ההדפסה לא תוסיף את התו שעובר לשורה הבאה.

תשובה לדוגמה:

```
def triangle(a):
    for i in range (a):
        for j in range(i):
            print ("*",end="")
        print ()
    print ()
    return
```

3. מימוש הפונקציה מתבצע ע"י לולאה שעוברת על כל איברי המערך המקורי. יש להגדיר מערך חדש, אליו יועברו התווים הרלוונטיים. בכל מחזור של הלולאה בודקים אם האיבר המתאים שווה לתו הביטול (הפרמטר השני) ואם כן, מדלגים על המשך ביצוע המחזור הנוכחי בלולאה (ע"י פקודת continue). אם מדובר בתו שונה הוא מתווסף אל המערך החדש. בסיום הלולאה, המערך החדש מוחזר כתשובת הפונקציה.

תשובה לדוגמה:

```
def deletech(a,c):
    b=""
    for i in range (len(a)):
        if a[i]==c:          # if equal to the "ignore" character
            continue        # move to next loop cycle
        b = b + a[i]
    return b
```

4. לצורך מימוש הפונקציה יש להגדיר משתנה (sum), אשר אליו יסוכם כל הציונים. הדבר מבוצע בעזרת לולאה אחת העוברת על כל האיברים. לאחר מכן, אפשר לחשב את הציון הממוצע (ע"י חלוקת ערך המשתנה במספר האיברים ברשימה. כדי לבדוק כמה ציונים נמצאים מתחת לממוצע וכמה נמצאים לעל (או שווים) לממוצע, שי להגדיר שני משתנים נוספים (מאופסים) ובלולאה לעבור שוב על האיברים, כאשר הפעם בודקים כל איבר מול הממוצע. אם הוא גבוה או שווה לממוצע מקדמים באחד את מונה הציונים הגבוהים (המשתנה more בפונקציה) לחילופין מקדמים את מונה הציונים הנמוכים (המשתנה less בפונקציה).

תשובה לדוגמה:

```
def grades(a):
    sum = 0
    for i in range (len(a)):
        sum = sum + a[i]
    avg = sum/len(a)          # calculate average
    less = 0                  # less than average grades count
    more = 0                  # more than average grades count
    for i in range(len(a)):
        if a[i]>=avg:
            more = more + 1   # grade is equal or higher than average
        else:
            less = less + 1
    return avg,more,less     #grade is less than average
```

5. פונקציה זו דומה מאוד לפונקציה שהופיעה השאלה 3. ההבדל הוא שבשאלה 3, הועבר גם תו הביטול ואילו כאן הוא מוגדר מראש

תשובה לדוגמה:

```
def removeblanks(a):
    b=""
    for i in range (len(a)):
        if a[i]==" ":        # if equal to blank (" ")
            continue        # ignore this item and move to next cycle
        b = b + a[i]
    return b
```

6. יש כמובן דרכים רבות למימוש הפונקציה. במקרה זה נעשה שימוש במערך של מחרוזות (משום שבדרך זו קל יותר להגיע אל הספרות השונות המרכיבות את המספר). הפונקציה מגדירה לולאה העוברת על כל המספרים מ -0 ועד לפרמטר. כל מספר מומר למערך. מגדירים משתנה (s) שעתיד לכלול את חישוב סכומי הספרות (לאחר שהועלו בחזקה המתאימה). בלולאה מקוננת נוספת, עוברים על ספרות המספר ומעלים כל אחת בחזקה (שהיא מספר הספרות במספר – נתון ע"י הפונקציה len). לאחר שעברנו על כל ספרות המספר וסיכמנו גם את החזקות נשאר לבדוק אם הסכום שהתקבל שווה למספר הנבדק (מונה הלולאה העליונה). אם כן, מדפיסים אותו ועוברים למספר הבא בלולאה העליונה. חשוב לציין שכל המספרים בעלי ספרה אחת בודדת מקיימים את התנאי.

תשובה לדוגמה:

```
def armstrong(a):
    for i in range(a):
        ich = str(i)          # change number to string
        s=0
        for j in range(len(ich)): #check each of the digits in the number
            s=s+int(ich[j])**len(ich)
        if s==i:
            print(i)
    return
```

7. זו כמובן דוגמה קלה. יש פשוט להציב את הפרמטרים בנוסחה (כפי שהופיעה בשאלה) ולהחזיר את התשובה שהתקבלה.

תשובה לדוגמה:

```
def triangle(a,b,c):
    s= 0.5*(a+b+c)
    area = math.sqrt(s*(s-a)*(s-b)*(s-c))
    return area
```

אפשר כמובן גם לכלול את החישוב כחלק מפקודת ה return ואז הפונקציה תכלול:

```
def triangle(a,b,c):
    s= 0.5*(a+b+c)
    return math.sqrt(s*(s-a)*(s-b)*(s-c))
```

8. גם פונקציה זו קלה למימוש. היא מורכבת מתנאי מרובה, אולם יש להיזהר שלא תהיה חפיפה בין התנאים. במקרה זה הפונקציה בודקת ראשית אם הציון גדול (או שווה) ל-90 ואם כן, אזי האות המתאימה היא A. בשלב הבא הפונקציה בודקת אם הציון גדול או שווה ל-80. בגלל שכל הציונים הגדולים מ-89 כבר נבדקו בתנאי הראשון, אזי כעת כל הציונים שגדולים (או שווים) ל-80 הם רק אלה שנשארו (בתחום שבין 80 ל-89) ולכן האות המתאימה היא B. הבדיקה ממשיכה, עד אשר כל התחומים נבדקו. חשוב לציין שכאשר הפונקציה מגיעה לפקודת ה-`return`, היא חוזרת למי שקרא לה ולא ממשיכה לבצע את הפקודה העוקבת ל-`return`.

תשובה לדוגמה:

```
def grades(a):
    if a >=90:
        return "A"
    elif a >=80:
        return "B"
    elif a >=70:
        return "C"
    elif a >=60:
        return "D"
    else:
        return "F"
```

כאשר בגוף התנאי נמצאת רק פקודה אחת, אזי אפשר לכתוב אותה בהמשך התנאי (לאחר הנקודתיים). במצב כזה, הפונקציה מתקצרת וייתכן אף שהיא קריאה יותר.

תשובה לדוגמה:

```
def grades(a):
    if a >=90: return "A"
    elif a >=80: return "B"
    elif a >=70: return "C"
    elif a >=60: return "D"
    else: return "F"
```

9. בפונקציה יש לכפול את כל אחד מששה הפרמטרים בערך המתאים ולסכם את התוצאה. חשוב כמובן לוודא שמכפילים תוך שימוש באותן יחידות. למשל חישוב בשקלים.

תשובה לדוגמה:

```
def sumMoney(a,b,c,d,e,f):
    return a*0.1+b*.5+c*1+d*2+e*5+f*10
```

10. גם במימוש פונקציה זו אין בעייתיות מיוחדת. יחד עם זאת, בגלל שהיא מחשבת רווח/הפסד על סמך קבועים (כמו מחיר ההקרנה, מחיר כרטיס ותמלוגים), רצוי שקבועים אלה יהיו מוגדרים בתחילת הפונקציה, כדי שאפשר יהיה לשנותם בעתיד, אם יהיה צורך.

תשובה לדוגמה:

```
def profit(a):
    screeningcost = 300
    royalties = 5
    ticketprice = 40

    revenue = a*ticketprice          # total revenue
    cost = screeningcost +a*royalties # total cost
    profit = revenue-cost
    return profit
```


11. פונקציה זו קצת יותר ארוכה, אם כי אין בה כל התחכמות. בגלל שהיא אחראית לחישוב מס הכנסה ומדרגות המס, כמו גם אחוזי המס נוטים להשתנות, רצוי מאוד לכלול את הנתונים במשתנים נפרדים. במימוש הפונקציה הנתונים נשמרו בנתונים, אך לפעמים עדיף דווקא לשמור אותן ברשימה (המדמה טבלה). בגלל שהמס הוא פרוגרסיבי פירוש הדבר שעבור שכר גבוה אחוז המס משתנה (כפי שנתון בשאלה). לכן אם אנו צריכים לחשב למשל את המס עבור שכר של 20,000 ₪, החישוב מורכב מסה"כ המס עבור כל המדרגות הנמוכות, או במילים אחרות, המס עבור המדרגה הראשונה ($4390 \cdot 10\%$), ועוד המס עבור המדרגה השנייה $15\% \cdot (7810 - 4390)$ ועוד המס עבור המדרגה השלישית $20\% \cdot (11720 - 7810)$ ועוד המס עבור המדרגה הרביעית $30\% \cdot (16840 - 11720)$ ועוד ההפרש שבין השכר (20,000) והמדרגה הרביעית מוכפל במס. לכן, בגלל שבמקרים רבים יש לחשב את המס לכל מדרגה, הפונקציה חישבה מראש את המס המחויב לכל מדרגה בסדרת משתנים הנקראים step1tax, step2tax, ..., step5tax. עכשיו אפשר לגשת לחישוב המס עבור השכר שהועבר כפרמטר לפונקציה. החישוב מורכב מסדרה של תנאים: אם השכר נמוך מהמדרגה הראשונה, אזי המס מחושב לפי שכר כפול אחוז המס של המדרגה הראשונה. אם לעומת זאת השכר נמוך מהמדרגה השנייה, אזי המס המחושב הוא סכום המס עבור המדרגה הראשונה (שחושב מראש ונמצא במשתנה ששמו step1tax) ועוד ההפרש שבין השכר והמדרגה הראשונה מוכפל באחוז המס של המדרגה השנייה. למשל עבור שכר של 6000 שקלים, המס המחושב הוא $15\% \cdot (6000 - 4390) + 10\% \cdot 4390$. באופן דומה מחשבים את המס גם עבור שכר גבוה יותר.

תשובה לדוגמה:

```
def tax(a):
    step1 = 4390
    step2 = 7810
    step3 = 11720
    step4 = 16840
    step5 = 36260
    tax1 = 10/100
    tax2 = 15/100
    tax3 = 23/100
    tax4 = 30/100
    tax5 = 34/100
    tax6 = 46/100

    step1tax = step1*tax1
    step2tax = step1tax + (step2-step1)*tax2
    step3tax = step2tax + (step3-step2)*tax3
    step4tax = step3tax + (step4-step3)*tax4
    step5tax = step4tax + (step5-step4)*tax5

    if a <= step1: return a*tax1
    elif a<=step2: return step1tax + (a-step1)*tax2
    elif a<=step3: return step2tax + (a-step2)*tax3
    elif a<=step4: return step3tax + (a-step3)*tax4
    elif a<=step5: return step4tax + (a-step4)*tax5
    else: return step5tax + (a-step5)*tax6
```

12. בגלל שהפרמטר הראשון לפונקציה הוא רשימה מספרית, אפשר להשוות את איברי הרשימה לפרמטר השני ואם הם גדולים מהמספר (הפרמטר השני) להעתיקם לרשימה חדשה ואם לא, פשוט להתעלם מהם.

תשובה לדוגמה:

```
def nums(a,b):
    c = []
    for i in range(len(a)):
        if a[i]>b:
            c.append(a[i])
    return c
```

13. לצורך חישוב הערך הקטן ביותר, הגדול ביותר וממוצע הרשימה, הפונקציה מגדירה שלושה משתנים. Sum אשר משמש כמונה צובר ואילו מסכמים את כל איברי הרשימה, min שימש שמשנתנה שבו יוכנס הערך הקטן ביותר ו - max שהוא משנתנה שאליו יוכנס הערך הגדול ביותר. לכל שלושה המשתנים הפונקציה מעתיקה את האיבר הראשון ברשימה. כלולאה שמתחילה מאחד (האיבר השני) ועד לאורך הרשימה, הפונקציה מוסיפה כל איבר אל המונה הצובר. לאחר מכן נבדק, אם האיבר קטן מהמשתנה min (אם כן, משמע מצאנו איבר קטן יותר ואז צריך להעתיק אותו לתוך min). באופן דומה כל איבר נבדק אם הוא גדול מהמשתנה max (אם כן, מצאנו איבר שהוא גדול יותר מהאיבר המקסימאלי ולכן יש להעתיקו לתוך max). בתום הלולאה, שני המשתנים min, max מכילים את האיבר הגדול ביותר והקטן ביותר ברשימה. יש להחזיר אותם וכן את הממוצע המחושב ע"י חלוקת sum במספר האיברים ברשימה.

תשובה לדוגמה:

```
def nums(a):
    sum = a[0]
    min = a[0]
    max = a[0]

    for i in range(1,len(a)):
        sum = sum + a[i]
        if a[i]<min: min = a[i]
        elif a[i]>max: max = a[i]
    return min, max, sum/len(a)
```

14. הפונקציה מגדירה מערך חדש אליו יוכנסו התווים הרלוונטיים. לאחר מכן, כלולאה המתחילה מאחד וממשיכה עד לאורך המערך ועוד אחד, נבדק מונה הלולאה. אם הוא מתחלק (ללא שארית - הפונקציה מודולו) בפרמטר השני של הפונקציה, משמע יש להעתיק את האיבר המתאים. יש רק לשים לב שיש להעתיק את האיבר, אשר האינדקס שלו מיוצג ע"י מונה הלולאה פחות אחד (שכן הלולאה התחילה מאחד ולא מאפס)

תשובה לדוגמה:

```
def arrayN(a,b):
    newStr = ""

    for i in range(1,len(a)+1):
        if i%b==0:
            newStr = newStr+a[i-1]

    return newStr
```

15. הפונקציה מגדירה משתנה sum אשר אליו מסכמים את ריבועי המספרים. בלולאה שהולכת מאפס ועד לפרמטר שהועבר (ועוד אחד) מחשבים את ריבועו של מונה הלולאה ומסכמים למשתנה sum. אפשר כמובן לשנות את הלולאה, כך שתתחיל מאחד ולא מאפס. הדבר לא ישנה את התוצאה.

תשובה לדוגמה:

```
def squareSum(a):
    sum=0
    for i in range(a+1):
        sum = sum + i*i

    return sum
```

16. כדי לענות על השאלה יש לערוך בדיקה קצרה. יש לחפש באינטרנט את טבלת ה-ASCII, כדי להבין מה הם המאפיינים של אותיות קטנות. הערך המספרי של אותיות אלה רציף והוא מתחיל בערך 97 עבור האות a ומסתיים ב-122 עבור z. הערך של האותיות הגדולות לעומת זאת קטן ב-32 (בהתאמה). כלומר, ערכה של A הוא 65 ואילו ערכה של Z הוא 90. מצוידים במידע זה אפשר להתחיל לתכנן את הפונקציה. מגדירים מחרוזת חדשה ובלולאה העוברת על כל איברי המחרוזת המקורית, בודקים את המספר הסידורי של התו. אם הוא בתחום שבין 97 ל-122 (כולל) יש להפחית ממנו 32 ולהפכו חזרה לתו. אחרת יש להחזירו כמות שהוא למחרוזת החדשה

תשובה לדוגמה:

```
def upperLower(a):
    newStr = ""
    for i in range(len(a)):
        if ord(a[i])>122: newStr=newStr+a[i]
        elif ord(a[i])<95: newStr=newStr+a[i]
        else: newStr=newStr+chr(ord(a[i])-32)

    return newStr
```

17. בגלל שברצוננו למצוא מחלק שהוא גדול מאחד, הלולאה שנגדיר מתחילה ב-2 ומתקדמת עד לפרמטר שקיבלנו. אם הפרמטר מתחלק ללא שארית במונה הלולאה, משמע שמצאנו מחלק. בגלל שאנחנו מתקדמים מ-2, אזי המחלק הראשון שנמצא הוא גם הקטן ביותר.

תשובה לדוגמה:

```
def divisor(a):

    for i in range(2,a):
        if a%i==0:
            return i
```

אפשר ליעל במקצת את הלולאה שכן אין טעם להמשיך ולבדוק את כל המספרים. אפשר לעצור את הבדיקה באמצע התחום. למשל אם אנחנו רוצים למצוא מחלק של 30, אפשר להסתפק בבדיקה של 15 מספרים שהרי ברור שהמספרים הגדולים מ-15 לא יכולים להיות מחלקים. במצב כזה, הפונקציה תשתנה במקצת:

```
def divisor(a):

    for i in range(2,int(a/2)):
        if a%i==0:
            return i
```

18. את הפונקציה אפשר לכתוב במספר צורות. אפשרות אחת היא לחלק את המספר שהתקבל בחזקה המתאימה של 10. למשל אם המספר בן ארבע ספרות צריך לחלק אותו בעשר בחזקת 3 ואז נקבל את הספרה השמאלית. באופן כללי, יש לחלק את המספר בעשר בחזקת מספר הספרות פחות אחת. כדי לקבל את הספרה השמאלית ללא שארית נשתמש בחלוקה של שלמים "/". כדי לחלץ מהפרמטר שקיבלנו את מספר הספרות שלו, אפשר להפכו למחרוזת ובעזרת הפונקציה len למצוא את אורכו.

תשובה לדוגמה:

```
def leading(a):
    return a//10**(len(str(a))-1)
```

לחילופין, אפשר להמיר את המספר למחרוזת (בעזרת הפונקציה str) ולאחר מכן לשלוף את תוכנו של האיבר השמאלי ולהפוך אותו חזרה למספר (שכן הפונקציה אמורה להחזיר מספר). למשל:

```
def leading(a):
    b=str(a)
    return int(b[0])
```

19. הפונקציה מגדירה שני משתנים odd שנועד לסיכום המספרים האי-זוגיים ו even לסיכום המספרים הזוגיים. בלולאה עוברים על כל איברי הרשימה. כל מספר זוגיים מתווסף (ע"י חיבור) למשתנה even ואילו כל מספר אי-זוגי מתווסף ל odd. בסוף מתבצעת בדיקה בין שני המשתנים. אם הם שווים הפונקציה מחזירה "NONE", אם odd גדול יותר הפונקציה מחזירה "ODD" ולא מחזירה "EVEN".

תשובה לדוגמה:

```
def oddEven(a):
    odd = 0
    even = 0
    for i in range(len(a)):
        if a[i]%2==0: even = even + a[i]
        else: odd=odd+a[i]
    if odd==even: return "NONE"
    elif odd>even: return "ODD"
    else: return "EVEN"
```

20. הפונקציה מגדירה מחרוזת חדשה newStr אליה יכנסו תוצאות פעולות ה XOR. בלולאה עוברים על כל איברי המחרוזות ומסכמים כל שתי סיביות (לאחר שהפכנו אותן למספרים לפני החיבור). אם תוצאת הסיכום היא 1, משמע שתי הסיביות שונות (אחת אפס והשנייה אחת). במצב זה יש להוסיף למחרוזת התוצאה את הערך 1. בכל מצב אחר (אם הסכום 0 או 2) פירוש הדבר ששתי הסיביות זהות ואז יש להוסיף למחרוזת הפלט את הספרה 0.

תשובה לדוגמה:

```
def XOR(a,b):
    newStr = ""
    for i in range(len(a)):
        if int(a[i])+int(b[i])==1: newStr=newStr+"1"
        else: newStr=newStr+"0"
    return newStr
```

21. הפונקציה מתחילה בהגדרת שני משתנים. משתנה אחד נועד לכתובת (אינדקס) של האיבר הקטן ברשימה והמשתנה השני נועד למיקום האיבר הגדול ברשימה. לאחר מכן מתבצעת לולאה שעוברת על כל איברי הרשימה. אם האיבר הנבדק קטן מהאיבר הראשון (או הקודם שנשמר), מיקומו נשמר במשתנה לשמירת מיקום המשתנה הקטן ביותר. ובמקביל אם האיבר הנבדק גדול מהאיבר הראשון (או הקודם), מיקומו נשמר במשתנה לשמירת מיקום האיבר הגדול ביותר. . כאשר מגיעים לסוף הלולאה, המשתנים מכילים את המיקום של האביר הקטן והאיבר הגדול בהתאמה. כל שנשאר הוא להחליף את האיברים בשני המיקומים.

```
def fun(lst):
    j,k = 0,0
    for i in range(len(lst)):
        if lst[i] > lst[j]:
            j = i
        elif lst[i] < lst[k]:
            k = i
    lst[k], lst[j] = lst[j], lst[k]
    return lst
```

[חזור לתוכן](#)

תשובות

רקורסיה

1. הפונקציה הראשונה (רקורסיבית) בודקת ראשית את תנאי הסיום. אם y שווה לאחד, אזי היא מחזירה את x (כלומר x בחזקת אחד שווה ל x). בכל מצב אחר, היא מחזירה את x כפול הפונקציה, כאשר הפרמטרים הם x ו $y-1$. הפונקציה השנייה (לולאה) מגדירה משתנה זמני המאותחל לאחד ובלולאה המתבצעת y פעמים מכפילה אליו את הערך x .

תשובה לדוגמה:

```
def power_r(x,y): # recursion
    if y==1:
        return x
    else:
        return x*power_r(x,y-1)

def power_l(x,y): # Loop
    value = 1
    for i in range(y):
        value = value * x
    return value
```

2. הפונקציה בודקת את תנאי הסיום (הפרמטר שהועבר) אם הוא אפס. אם כן, היא מחזירה אפס ולא מחזירה את המספר (לאחר שהומר למחרוזת) ועוד הפונקציה עם פרמטר הנמוך באחד מהפרמטר הנוכחי.

תשובה לדוגמה:

```
def printAll(x):
    if x==0:
        return "0"
    else:
        return (str(x)+printAll(x-1))
```

3. הפונקציה בודקת את תנאי הסיום (הפרמטר שהועבר) אם הוא קטן או שווה 2. אם כן, היא מחזירה 1 ולא מחזירה את הסכום המורכב מהפונקציה עם הפרמטר פחות ועוד הפונקציה והפרמטר פחות שניים.

תשובה לדוגמה:

```
def fibo(n):
    if n<=2:
        return 1
    else:
        return fibo(n-1)+fibo(n-2)
```

4. הפונקציה בודקת את תנאי הסיום. אם המחרוזת ריקה, היא מוחזרת. אם אינה ריקה היא מוחזרת לאחר שהאיבר הראשון הורד ממנה והצטרף בסופה (ע"י שימוש באופרטור "+")

תשובה לדוגמה:

```
def reverse(str):
    if str=="":
        return str
    else:
        return reverse(str[1:])+str[0]
```

5. ראשית הפונקציה בודקת את תנאי הסיום. אם הפרמטר הוא אחד, היא מחזירה אחד ולא היא מחזירה את סכום הפרמטר שהועבר ועוד הפונקציה עם הפרמטר פחות אחד.

תשובה לדוגמה:

```
def rSum(n):
    if n==1:
        return 1
    else:
        return n + rSum(n-1)
```

6. ראשית הפונקציה בודקת את תנאי הסיום. אם הפרמטר הוא אחד, היא מחזירה את הערך 5. אם לא היא מחזירה את סכום של 3 ועוד ערך הפונקציה עבור האיבר הקודם (הערך שמתקבל מקריאה לפונקציה עם פרמטר הקטן באחד מהפרמטר הנוכחי).

תשובה לדוגמה:

```
def itemN(n):
    if n==1:
        return 5
    else:
        return itemN(n-1)+3
```

7. כמו בכל פונקציה רקורסיבית ראשית היא בודקת את תנאי הסיום. אם הפרמטר הוא אפס, היא מחזירה אותו כתשובה. אם לא היא מחזירה את הסכום של המספר (הפרמטר) מודולו 10 (או במילים אחרות היא מחזירה את ערכה של ספרת האחדות) ועוד תשובת הפונקציה עבור פרמטר שהוא המספר מחולק ב-10 (ללא שארית). כלומר המחובר השני הוא סכום הספרות במספר המקורי לאחר שהורדה ממנו הספרה הימנית ביותר.

תשובה לדוגמה:

```
def digitSum(n):
    if n==0:
        return n
    else:
        return n%10+digitSum(n//10)
```

8. כמו בכל פונקציה רקורסיבית ראשית היא בודקת את תנאי הסיום. ובמקרה זה אם הפרמטר הוא אפס, היא מחזירה אותו כתשובה. אם לא, היא מחזירה את הסכום של המספר (הפרמטר) מודולו 2 (או במילים אחרות היא מחזירה את השארית מחלוקת המספר ב-2) ועוד תשובת הפונקציה עבור פרמטר שהוא המספר מחולק ב-2 (ללא שארית), כאשר מספר זה מוכפל פי 10. הסיבה שאנחנו מכפילים את תוצאת הפונקציה פי 10 היא מפני שאנחנו רוצים ליצור את המספר הבינארי. הפונקציה מתקדמת מימין לשמאל, כאשר בכל קריאה אנחנו מתקדמים לסיבית הימנית יותר.

תשובה לדוגמה:

```
def Dec2Bin(n):
    if n==0:
        return n
    else:
        return n%2+10*Dec2Bin(n//2)
```

9. גם כאן, הפונקציה ראשית בודקת את תנאי הסיום. ובמקרה זה אם הפרמטר הוא רשימה ריקה, היא מחזירה אפס כתשובה. אם לא, היא מחזירה את הסכום של המספר הנמצא באיבר הראשון ברשימה ועוד תשובת הפונקציה עבור פרמטר שהוא הרשימה המקוצרת (ללא האיבר הראשון).

תשובה לדוגמה:

```
def listSum(lst):
    if lst==[]:
        return 0
    else:
        return lst[0]+listSum(lst[1:])
```

10. תנאי הסיום במקרה זה יהיה בדיקה על הפרמטר השני אם הוא אפס. אם כן, הפונקציה מחזירה אפס. אם אינו מאופס הפונקציה מחזירה את הסכום של הפרמטר הראשון ועוד תוצאת הקריאה לפונקציה עם שני הפרמטרים, אלא שהפרמטר השני קטן באחד. המשמעות של הפונקציה היא שהכפל $a*b$ מחושב ע"י $a+a*(b-1)$, כל זמן ש b עדיין לא אפס.

תשובה לדוגמה:

```
def mult(a,b):
    if b==0:
        return b
    else:
        return a+mult(a,b-1)
```

11. תנאי הסיום במקרה זה יהיה בדיקה של הפרמטר אם הוא אפס. אם כן, הפונקציה מחזירה אפס. אם אינו מאופס הפונקציה מחזירה את הסכום של הפרמטר בריבוע ועוד תוצאת הקריאה לפונקציה עם פרמטר שהינו קטן באחד מהפרמטר המקורי.

תשובה לדוגמה:

```
def sumSquare(a):
    if a==0:
        return 0
    else:
        return a*a+sumSquare(a-1)
```

12. שאלה זו מתחכמת מעט. עקב העובדה שאיננו יכולים להשתמש בפקודות כפל, יש לחשב את הערכים הריבועיים על סמך הנוסחאות המתמטיות הרגילות. למשל $n^2 - 2n + 1 = (n-1)^2$. אנחנו רוצים לבטא את n^2 תוך שימוש ב $(n-1)^2$. לכן בהעברת ביטויים בין שני אגפי המשוואה, נקבל: $n^2 = (n-1)^2 + 2n - 1$. עכשיו אפשר להגדיר את הפונקציה הרקורסיבית. תנאי הסיום במקרה זה יהיה בדיקה של הפרמטר אם הוא אחד. אם כן, הפונקציה מחזירה אחד. אם אינו מאופס הפונקציה מחזירה את הסכום של הפרמטר מוכפל פי 2 פחות אחד ועוד ערך הפונקציה המתקבל תוך שימוש בפרמטר שהינו קטן באחד מהפרמטר המקורי.

תשובה לדוגמה:

```
def Square(a):
    if a==1:
        return 1
    else:
        return Square(a-1)+2*a-1
```


13. זו דוגמה פשוטה, אלא שתנאי היציאה שלה מורכב משלושה תנאים שונים. אם ערכו של הפרמטר הוא 0, אזי הפונקציה מחזירה 3, אם ערכו 1 אזי הפונקציה מחזירה 0 ואם ערכו 2, אזי הפונקציה מחזירה 2. לכל ערך אחר הפונקציה מחזירה את הסכום של שתי קריאות לפונקציה. אחת עם פרמטר קטן $h - 2$ מהפרמטר הנוכחי והשנייה עם פרמטר שהוא קטן ב-3 מהפרמטר הנוכחי.

תשובה לדוגמה:

```
def Perrin(a):
    if a==0: return 3
    elif a==1: return 0
    elif a==2: return 2
    else:
        return Perrin(a-2)+Perrin(a-3)
```

14. כמו במקרים קודמים, תנאי היציאה הוא אם הפרמטר שהועבר מאופס. אם לא הפונקציה מדפיסה מחרוזת בעלת n כוכבים וקוראת לפונקציה עם פרמטר הקטן באחד מהמספר הנוכחי.

תשובה לדוגמה:

```
def stars(a):
    if a==0: return
    else:
        print ("*" * a)
        stars(a-1)
```

15. כדי לקבל משולש נוסף (הפוך) יש להוסיף פקודות הדפסה זהה לאחר הקריאה לפונקציה. משמעות הדבר שלאחר שהפונקציה הרקורסיבית תתחיל לחזור, בכל מופע היא תדפיס את השורה פעם נוספת.

תשובה לדוגמה:

```
def stars(a):
    if a==0: return
    else:
        print ("*" * a)
        stars(a-1)
        print ("*" * a)
```

16. גם פונקציה זו תתקדם אחורנית (מהסוף להתחלה). תנאי העצירה הוא אם הפרמטר שהועבר הוא אחד ואז היא מחזירה אחד. בכל מצב אחר היא מחזירה את $1/n$ ועוד ערך הפונקציה כאשר הפרמטר שהועבר לה קטן באחד מהפרמטר הנוכחי.

תשובה לדוגמה:

```
def harmonicSum(a):
    if a==1: return 1
    else:
        return harmonicSum(a-1)+1/a
```

17. כאן קיימים שני תנאי עצירה. אם הפרמטר השלישי (מס' האיבר) שווה לאפס, אזי הפונקציה מחזירה את הפרמטר הראשון (האיבר הראשון בסדרה). אם הפרמטר השלישי שווה לאחד, הפונקציה מחזירה את הפרמטר השני (האיבר השני בסדרה) ואילו אם הוא גדול מאחד, אזי הפונקציה מחזירה את הסכום של שתי קריאות לפונקציה. קריאה אחת כוללת שאת שני הפרמטרים המקוריים ופרמטר שלישי שהוא קטן באחד מהפרמטר הנוכחי וקריאה שנייה שכוללת את שני הפרמטרים הראשונים ופרמטר שלישי שהוא קטן בשתיים מהפרמטר הנוכחי.

תשובה לדוגמה:

```
def Lucas(a,b,n):
    if n==0: return a
    elif n==1: return b
    else:
        return Lucas(a,b,n-1)+Lucas(a,b,n-2)
```

18. לפני שכותבים את הפונקציה, צריך לחשוב מעט על האלגוריתם. אם הפרמטר שהועבר הוא אפס, אזי הפונקציה צריכה להחזיר 1 (מספר האפסים במספר). אם לעומת זאת המספר אינו אפס, אך הוא קטן מעשר, הפונקציה תחזיר אפס (זה מספר חד ספרתי שאינו כולל אפס). אם המספר מודולו 10 שווה אפס (במילים אחרות הוא מתחלק בעשר), הפונקציה תחזיר אחד ועוד תוצאת הפונקציה על פרמטר שהוא המספר הנוכחי מחולק בעשר (חלוקת שלמים). אם לעומת זאת, המספר הנוכחי לא מתחלק בעשר, הפונקציה תחזיר אפס ועוד תוצאת הפונקציה על פרמטר שהוא המספר הנוכחי מחולק בעשר. לדוגמה, אם הפרמטר הוא 20020, אזי הוא מתחלק בעשר ולכן הפונקציה תחזיר 1 ועוד בדיקה של 2002 (שהוא המספר הנוכחי מחולק בעשר).

תשובה לדוגמה:

```
def zeros(n):
    if n==0: return 1
    if n<10: return 0
    if n%10==0: return 1 + zeros(n//10)
    else: return zeros(n//10)
```

19. הבדיקה הראשונה שעלינו לבצע היא האם המספר מתחלק בערך שהועבר כפרמטר השני (או במילים אחרות האם הפרמטר השני הוא אכן מחלק של הראשון). אם לא, אזי הפונקציה תחזיר אפס. אם כן, הפונקציה תחזיר אחד ועוד תוצאת הפונקציה, כאשר הפרמטר הראשון במקרה זה יהיה המספר המקורי מחולק בפרמטר השני.

תשובה לדוגמה:

```
def factor(n,m):
    if n%m !=0 : return 0
    return 1 + factor (n/m,m)
```

20. הבדיקה הראשונה שעלינו לבצע היא האם המערך ריק. אם כן, הפונקציה תחזיר אפס. אם אינו ריק, הפונקציה תחזיר אחד ועוד הערך שיתקבל מקריאה נוספת לפונקציה, אלא שעכשיו המערך קטן באחד. המערך החדש מכיל את איברי המערך הנוכחי, כאשר האיבר הראשון הושמט.

תשובה לדוגמה:

```
def length(str):
    if str=="": return 0
    return 1 + length(str[1:])
```

[חזור לתוכן](#)