

תכנות בסיום חברת להסביר ותרגול פיתון

מסמך זה נוצר במקור ע"י ד"ר אהרון ידין ועובדן ע"י ד"ר רמי רשקוביץ

מהדורה 3.8

הערה מבוא

החברת נוסחה בלשון זכר מטumi נוחיות בלבד – עמק הסליחה!
כל ההתייחסויות לשפט הפתיחה פיתון נעשו בלשון נקבה.

חברת זו נכתבה כדי להקל על לימוד הקורס **תכנות בסיסי** (בעיקר עבור הסטודנטים אשר נחשפים לחומר זה בפעם הראשונה). החברה כוללת מגוון רחב של תרגילים בנושאי תכנות בשפט פיתון. במקומות רבים, בפירוט פקדות (קוד ופרמטרים) התווסף רוחחים לצורך שיפור הקריאות.

פיתון – מדריך ללימוד עצמי

1. מהו מחשב
2. שיטות מספריות
3. הסביבה האינטראקטיבית – IDLE
4. פיתון כמחשב
5. סוגים ותوابים מספריים
6. דוגמת תרוכת דלק
7. שמות ערכיהם
8. משתנים
9. מחזוזות
10. עד על סוגים נתונים
11. פונקציות מתמטיות נוספת
12. טיפול במחזוזות
13. הדפסה
14. עד על הדפסה
15. לולאות
16. ביטויים בוליאניים
17. גלוט
18. תנאים
19. רשימות
20. לולאות while
21. פונקציות
22. עד על פונקציות
23. טיפול בראשיות
24. פעולות בוליאניות
25. פונקציות לטיפול במחזוזות
26. עיצוב מחזוזות
27. פקודות else - break, continue
28. טיפול בקבצים
29. הבנת תוכניות
30. מצא את החסר
31. כתיבת תוכניות
32. כתיבת פונקציות
33. רקורסיה
34. תרגילי תכנות (לא פתרו)

תשובות לתרגילים

1. מהו מחשב
2. שיטות מספריות
3. הסביבה האינטראקטיבית – IDE
4. פיתון כמחשבנו
5. סוגים ותوابים מספריים
6. דוגמת תרצהך דלק
7. שמות ערכים
8. משתנים
9. מחוזות
10. עד על סוגים נתונים
11. פונקציות מתמטיות נוספת
12. טיפול במחוזות
13. הדפסה
14. עד על הדפסה
15. לולאות
16. ביטויים בוליאניים
17. גלוט
18. תנאים
19. רשימות
20. לולאות while
21. פונקציות
22. עד על פונקציות
23. טיפול בראשיות
24. פעולות בוליאניות
25. פונקציות לטיפול במחוזות
26. עיצוב מחוזות
27. פקודות else - i break, continue
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 מתוכניתת הקולטת, מעבדת ופולטת נתונים"², וכן
- "יחידה או מערכת אשר מסוגלת לבצע רצף של פקודות באופן שהוגדר מראש. הפעולות הן לעתים חישובים מסוימים, או עיבודי נתונים, אך הן גם יכולות אפשריות לביצוע קלט ופלט"³.

ללא כל קשר להגדרה בשימוש, מחשב הוא מכונה אשר נועדה לבצע רצף של פעולות, מעין "מתכוון" מסוים, על אוסף נתונים שגם הוא מוגדר ב"מתכוון". המתכוון הוא לא יותר מאשר האלגוריתם, או התוכנית שנכתבה לפתרון הבעיה. את התוכנית כתובים תוך שימוש בפקודות אשר ייחד מבצעות את מה שצריך לבצע (האלגוריתם). המعالגים האלקטרוניים במחשב נבנו כך שייכרו את אוסף הפקודות וייהו מסוגלים לבצען. פקודות המכונה הבסיסיות (פקודות סף), הינן בדרך כלל פשוטות מאוד ומשמשות לביצוע פעולות פשוטות (אוטומיות) כמו למשל:

- השוואה בין שני מספרים,
- בדיקה אם מספר כלשהו קטן מאוד,
- חיבור שני מספרים,
- הכפלת שני מספרים, ועוד'

הפקודות פשוטות שתוארו לעיל, הין חלק משפט תכנות בסיסית, שפט המכונה. שפה זו, שהינה ייחודית לכל מעבד, מורכבת מוסף של סיביות (ספרות בינהיות) המגדירות את הפעולה שיש לבצע. כאמור, לכל מעבד שפת מכונה מיוחד, ותוכנית שנכתבה תוך שימוש בשפת המכונה למעבד אחד, לא תוכל לרוץ על מעבד אחר (אלא אם כן, המעבדים שייכים לאותה משפחה – דוגמת מעבדי משפחת ה – PC).. השימוש בפקודות פשוטות נבע מהריצון ומהוצרך לתכנן ופותח של מחשבים שהם פשוטים ובעיקר זולים מאוד. פשוטות המחשב (המעבד) והוזלת עלות אפשרות להניע את ה"מחזור הכלכלי", אשר אחראי במידה רבה לפיתוח המואץ בתחום המחשב השוני ואשר בעטיו יכולים רוב המכשירים המוכרים לאדם עושים שימוש, לעתים נרחב מאוד, במחשבים. "מחזור כלכלי" זה, מאופיין ע"י מספר המספרים לאדם עשויים שימוש, לדוגמה כירכום או מעגלים. תוך שימוש במחשבים/מעבדים חדשים אלה, ניתן לפתח מוצריים חדשים, או ישונים חדשים. מוצריים ויישומים חדשים אלה מספקים פתרון טוב יותר לצרכים בשוקים קיימים וכן מאפשרים לפולח שוק חדשים. הפניה לקהל חדש, יצירת הזדמנויות שיווקיות וגורמת לפיתיחה של חברות חדשות, הקמות כדי להתרחות בשוקים החדשניים, ועל קהל החדש, אך תוך יישום רעיונות חדשים. רעיונות חדשים לעתים דוחפים לייצור טכנולוגיה חדשה והמחזור הכלכלי חוזר על עצמו פעמי נספתח. הכוח המניע את כל המחזורי הוא כמובן כסף ולכן הוא מכונה מחזור כלכלי. דוגמה להמחשת האמור לעיל, ניתן למצוא בעולם התקשורות הסלולרית. כל הטלפונים מבוססי מעבדים, אולם הפיתוח המאושם, אשר לווה גם בשיווק.Aggressive, החיל רק לאחר שהטכנולוגיה הבשילה וונתן היה ליצור מעבדים שהם זולים, אמינים, בעל הספק נmor וקטנים מספיק כדי שיוכלו להיכلل במכשיר הטלפון הנני. ברגע שהמחזור החל, אכן עדים למורץ מהיר מאוד של שכליים, מכשירים חדשים בעלי יכולות חדשות, חברות חדשות וחוזר חלילה.

כאמור, עקב התפתחות הטכנולוגיה, כיום, רוב המכשירים הינם מבוססי מחשב, אפילו מכשירים ביתיים "פשוטים" כמו טלוייזיה, תנור, מקרר, מכונת כביסה, מערכת אזעקה וכו'. לעתים, אף מדובר במכשירים זולים במיוחד, כמו למשל שעונים מבוססי מחשב שנמכרים במחירים של דולרים בודדים. השימוש הנרחב במערכות כוח המניע את המכשירים הרבים, מתאפשר רק בגלל העבודה מהם זמינים ובעלויות שהין 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/beamsactivity/6thgrade/vocabulary/

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

תכנות בסיסי

שפט המכונה, הכוללת רצף סיביות בינהרי, קשה מאוד לשימוש ע"י בני אדם ولكن פותחה שפט הסוף שהינה תרגום ישר (1:1) לשפט המכונה. במקומם להשתמש בערכים מספריים לייצוג פקודות, שפט הסוף מגדרה פקודות בעלות שםמשמעותי.

לדוגמה:

הערך הבינארי
1011 000 0110 0100

הוא פקודה בשפט מכונה במחשב אישי רגיל (մבוסס מעבד אינטל ממפחota ה - 86X).

פקודה זו כאשר מתורגמת לשפט סוף היא: MOV AL, 100

הפקודה מכניסה את הערך 100 לטור אגר AL שהינו חלק מהמעבד (הדוגמה תהיה ברורה הרבה יותר בהמשך, אך היא נמצאת כאן רק כדי להציג את העובדה ששפט מכונה הינה בינהרית, קשה לכנתיבתה והבנה וכן פותחה שפט הסוף שהינה הרבה יותר ידידותית למפתח).

ഫישוטות המוממשת בשפט הסוף, מצד אחד, אמנם תורמת את חלקה בהורדת עליונות וקידום המחזור הכלכלי, אך מצד שני שימוש בשפט סוף, עבור מפתחי תוכנה, שהינו הרבה יותר קל בהשוואה לשפט מכונה, הינו עדין קשה ועמיף. כדי להתגבר על בעיה זו, עולם המחשב ממסר רמות של הפשטה. כל רמה גנית וمتבססת על הרמה מתחתיה, ומוסיפה יכולות / או הפשטה של תהליך פיתוח התוכנה. בקרה זו ניתן לגשר בין פשטוות תכנון ופיתוח החומרה (כדי להוריד עליונות), לבין הצורך הגובר בכלל פיתוח מתוחכמים, שיעזרו למפתחי התוכנה לצמצם את הפער ההולך וגדל, בין צרכי המשתמשים ובין יכולות הפיתוח.

שיטת מספריות

כבר מחרה ההיסטוריה, האדם נזקק למערכת שתאפשר לעיר כמוויות. וכן עם הזמן פותחו שיטות שונות שמנעו לחת מענה לבעה של מדידת והערכת כמוויות. חלק מושיטות אלה היו פשוטות ואחרות מורכבות, חלקן התייחסו רק לכמוויות מוחשיות (למשל הספרה אפס לא הייתה אצל המצרים הקדמונים), וחלק אחר התייחס למושג כמות, גם כאשר היא לא קיימת (הספרה אפס הייתה מוגדרת חלק מהסימנים שבשימוש בני המאה בדרום אמריקה). ההבנה לגבי מאפיין שיטת המספר התפתחה עם הזמן.

כל שיטת מספר חיבת להשתמש בסימנים מסוימים ומוגדרים לתיאור הגודל (ביתי' למספרים השונים). רצוי, כמובן, שכמה הסימנים תהיה קטנה ככל האפשר, כדי להקל על בני האנוש להשתמש בהם בקלות (אך לא קטנה מדי – כפי שנראה בפרק הדן במספרים בינאריים). אוסף המספרים, כמובן, חייב לכוסות אל כל אפשרות המספרים.

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

שיטת העשרונית

שיטת 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 סיכון 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:

d – מגדר את מיקום הספרה

b – מגדר את בסיס השיטה המספרית (במקרה זה 10)

$d = \{0,1,2,3,4,5,6,7,8,9\}$

שיטת העשרונית היא הטבעית ביותר לאדם, אך קיימות שיטות נוספות מבוססות על שימוש בבסיסים אחרים. חלק מהשיטות הין תיאוריות בלבד, ואחרות פותחו מתוך כוונה לעזר ולהקל בעיקר בעבודה מול מחשבים.

כאמור, קיימות שיטות נוספות בסיס הנוסחה הכללית שתוארה בסעיף הקודם. יש כמובן, להחליפ את הבסיס ואז מתיקל מספר אחר.

לדוגמה:

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

תכנות בסיס.

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

במקרה זה 573 בסיס אוקטלי ערכו 379 עשרוני.

ובאופן כללי:

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

כאשר:

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

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

באופן דומה ניתן להגיד מספרים לפי כל בסיס נתון.

שיטת הבינארית

שיטת הבינארית (על בסיס 2) היא השיטה הפשטota ביוטר מבין כל השיטות מבוססות המיקום (כפי שהוגדר בסעיף 2, שיטות בהן הערך נקבע לא רק לפ' הספרה, אלא גם עפ' מיקומה). שיטה זו גם מעניינת בכל הקשור למחשבים. אמנם המחשבים הראשונים היו מבוססים במספרים עשרוניים, אך העובדה שמדובר בمعالגים אלקטרוניים הנמצאים בשני מצבים - דילק/כבי (On/Off), גרמה לשינויו וכיום כל המחשבים המודרניים מבוססי מספרים בינריים. השיטה הבינארית, למעשה אינה שונה מאשר השיטות. גם כאן הנוסחה זהה, אלא שאם בשיטה העשרונית הבסיס הוא 10, כאן, הבסיס הוא 2 והספרות האפשריות הן רק 0 ו- 1. ספרות בינריות נקראות סיביות (סיבית – ספרה בינארית) אוanganilit (Bit - Binary Digit).

לדוגמה:

הערך של 11011_2 מחושב ע"י:

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

במקרה זה 11011_2 בינארי ערכו 27 עשרוני.

ובאופן כללי:

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

כאשר:

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

תכנות בסיסי

כאמור השיטה הבינארית היא השיטה הנהוגה בעולם המחשב, אך הינה 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 לחישוטין.

לדוגמה:

$$3,212,580_{10} = 1100010000010100100100_2$$

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

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

הדרך לעקוף את הבעיה היא ע"י שימוש בשיטה האוקטלי (בבסיס 8), או השיטה הקסודותצימלית (בבסיס 16). השיטה הקסודותצימלית 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 ברויתן מבוססות בסיס שהינו חזקה של שתיים, מאפשרות המרת קלה ע"י קיבוץ ספרות (ראה תרשימים 1-2). מספר הספרות שיש לקבץ הוא החזקה המתאימה של שתיים. (כל שלוש ספרות בינאריות הופכות לספירה אוקטלית אחת, וכל ארבע ספרות בינאריות הופכות לספירה הקסודותצימלית אחת). הקיבוץ מתבצע תמיד מצד ימין!

לכן:

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

ובאופן דומה:

$$3,212,580_{10} = 1100010000010100100100_2$$

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

תהליך ההמרה בין מספרים אוקטליים והקסודותצימליים לבינאריים הוא כMOVN DO סטריא. כדי להפוך מספרים בינאריים יש לקבץ מספר ספרות וכדי להפוך מספרים אוקטליים או הקסודותצימליים יש להפוך כל ספרה לערך הבינארי המתאים.

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 1-2: טבלת המרת מספרים בינהירים

שברים

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

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

לדוגמה:

הערך של 675.438 מחושב ע"י

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

המרת מספרים שלמים

כאמור, המרות מבסיס בינהרי לבסיס אחר שהוא חזקה של שתיים (כמו בסיס אוקטלי או הקסהדצימלי) מתבצעות בקלות ע"י קיבוץ ספרות בינהירות (סיביות או ביטים). מספר הספרות שיש לקבץ הוא החזקה של הבסיס החדש. המירה הפוכה מתבצעת ע"י תרגום הספרה האוקטלית או הhexadicimalית לספרות הבינהירות המתאימות (ראה תרשימים 1-2). ספרה אוקטלית נרשמת כשלוש סיביות ואילו ספרה הhexadicimalית נרשמת ארבע סיביות (החזקות המתאימות של שתיים).

לדוגמה:

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

IX

$$5BA6_{16} = 0101\ 1011\ 1010\ 0110_2 \\ = 101101110100110_2$$

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

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

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 המתיקבלת היא אפס.

למשל: המرة של 37¹⁰ למספר בינהי 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 122₁₀ לבסיס 7

$$\begin{array}{rcl} 122/7 & = & 17 \text{ remainder: } 3 \\ 17/7 & = & 2 \text{ remainder: } 3 \\ 2/7 & = & 0 \text{ remainder: } 2 \end{array}$$

ולכן,

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

תכנות בסיסי

המרה מבסיס כלשהו n (כאשר $n > 2$ – אין בסיסים עשרוניים, בינאריים או חזקה של שניים), מתבצעת ע"י המרה כפולה. בשלב ראשון, המרה לבסיס עשרוני ולאחר מכן המרה נוספת מבסיס עשרוני לבסיס n . (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 מבוצעות ע"י הכפלת השבר בבסיס, איסוף השלים, הכפלה נוספת (רק של השבר) וחזר חילתה. כל זאת עד אשר התוצאה המתתקבלת היא 1.0 (שלם ללא שבר).

למשל:

הפית 0.375_{10} למספר בינארי:

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

לכן

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

חשוב להזכיר שיש מקרים בהם שבר עשרוני אינו ניתן להמרה פשוטה למספר בינארי ולכן, התוצאה תהיה שבר מ חוזר אינסופי.

לדוגמה:

הפית 0.3_{10} למספר בינארי:

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

לכן

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

תכנות בסיס.

שאלות חוזרת
1. הופר מבסיס דצימלי (עשרוני) לבינארי

1. $765_{10} = \underline{\hspace{2cm}}_2$
2. $1234_{10} = \underline{\hspace{2cm}}_2$
3. $5739_{10} = \underline{\hspace{2cm}}_2$
4. $1001_{10} = \underline{\hspace{2cm}}_2$
5. $9173_{10} = \underline{\hspace{2cm}}_2$
6. $1024_{10} = \underline{\hspace{2cm}}_2$
7. $2047_{10} = \underline{\hspace{2cm}}_2$
8. $15365_{10} = \underline{\hspace{2cm}}_2$
9. $999_{10} = \underline{\hspace{2cm}}_2$
10. $1010_{10} = \underline{\hspace{2cm}}_2$

2. הופר מבסיס בינארי לדצימלי (עשרוני)

1. $1100011_2 = \underline{\hspace{2cm}}_{10}$
2. $11000111_2 = \underline{\hspace{2cm}}_{10}$
3. $100101011_2 = \underline{\hspace{2cm}}_{10}$
4. $100011_2 = \underline{\hspace{2cm}}_{10}$
5. $1101111_2 = \underline{\hspace{2cm}}_{10}$
6. $10011010010_2 = \underline{\hspace{2cm}}_{10}$
7. $1011000_2 = \underline{\hspace{2cm}}_{10}$
8. $10101111_2 = \underline{\hspace{2cm}}_{10}$
9. $10010000_2 = \underline{\hspace{2cm}}_{10}$
10. $100100000_2 = \underline{\hspace{2cm}}_{10}$

3. הופר מדצימלי לבסיס המתאים

1. $4759_{10} = \underline{\hspace{2cm}}_{16}$
2. $4759_{10} = \underline{\hspace{2cm}}_8$
3. $918273_{10} = \underline{\hspace{2cm}}_{16}$
4. $918273_{10} = \underline{\hspace{2cm}}_8$
5. $117_{10} = \underline{\hspace{2cm}}_4$
6. $555_{10} = \underline{\hspace{2cm}}_4$
7. $555_{10} = \underline{\hspace{2cm}}_8$
8. $555_{10} = \underline{\hspace{2cm}}_{16}$
9. $777_{10} = \underline{\hspace{2cm}}_7$
10. $999_{10} = \underline{\hspace{2cm}}_9$
11. $123456_{10} = \underline{\hspace{2cm}}_5$
12. $199_{10} = \underline{\hspace{2cm}}_3$
13. $9876_{10} = \underline{\hspace{2cm}}_6$
14. $998877_{10} = \underline{\hspace{2cm}}_{11}$

תכנות בסיס.

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

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 הtablלה

בינארי	בסיס 3	בסיס 5	בסיס 8	עשרוני	הקסדצימלי
					65
				219	
			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 בס.ו.י.

3. 0110.0001_2 = _____₁₀
4. 1111.1111_2 = _____₁₀
5. 0.000001_2 = _____₁₀
6. 3.5625_{10} = _____₂
7. 9.9375_{10} = _____₂
8. 11.55_{10} = _____₂
9. 0.7_{10} = _____₂
10. 14.0625_{10} = _____₂

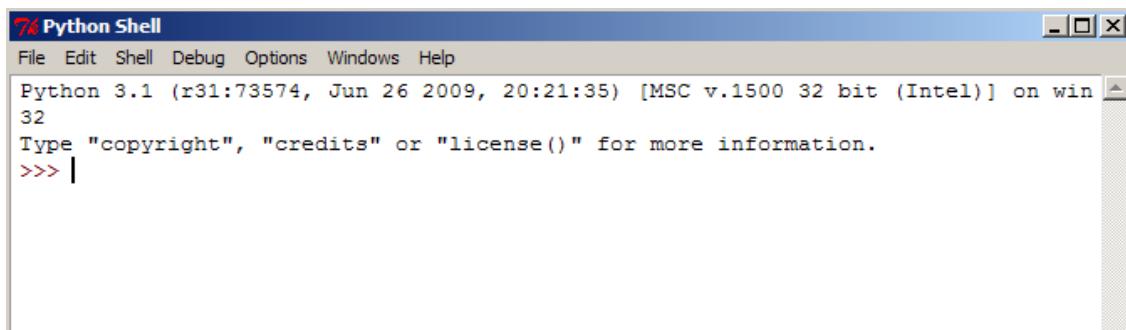
[תשובות](#)

[חזרה לתוך](#)

הסביבה האינטראקטיבית – IDLE

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

לכניתה לסייע, הפעל את פיתון. יוצר מסך דמוי:



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

הסימן "<<>" מציין שפיתון מוכן לקבל את הפקודה הבאה שלנו.

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

בהכנסת פקודות, רוח הואתו בלתי נראה, אך שפת פיתון מתעלמת ממנו.

[חזר לתוך](#)

פיטון כמחשבון

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

>>> 2 + 7

9

>>> 3 * 5

15

>>> 16 / 4

4

>>> 12 - 9

3

>>>

בודאי שמת לב לעובדה שתרגילים המתיחסים לפקודות חיבור חישור וכפל, התשובה נרשמת כמספר שלם ואילו כאשר אנו מחלקים התשובה נרשמת כמספר. זו תcona חדשה שהוכנסה במהדורה 3.1 של פיטון והיא נועדה בעיקר למנוע איבוד מידע, כאשר אנו מחלקים. אנו נתיחס לנקודה זו ברחבה גם בהמשך. בכלל פעולה חשבונאית בין שבר למספר שלם, התוצאה תהיה שבר (כדי למנוע איבוד מידע).

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

>>> 5 ** 2

25

>>> 2 ** 10

1024

>>>

סדר ביצוע הפעולות החשבונאיות הוא הסדר הרגיל: קודם חזקות, אחר כך כפל או חילוק ובסיום חיבור ו/או חישור. ניתן כמובן להשתמש בסוגרים 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 בסוגרים, אשר למעשה מבצע פעולה חשבונאית שונה. בנווסף, בגלל שהפעולה החשבונאית כוללת בתוכה חילוק וכפ' שהוסבר לעיל, חילוק גוררת תשובה שפורמת של שבר, כל התשובות שהתקבלו בתרגיל זה הין שברים גם כן.

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

>>> 16 // 5

תכנות בPython

3

>>>

>>> 16 / 5

3.2

>>>

אם ברצוננו ליצג מספר בפורמט של שבר, כל שיש לעשות זה להוסיף לו נקודה עשרונית. لكن, המספר 3 מבטא מספרשלם ואילו המספר 3.0 מבטא שבר. בפעולות בין מספרים שלמים התוצאה תהיה מספרשלם, כאמור כמעט במדויק 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, במיל'ם אחריות אם התשובה היא מספרשלם 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. $2^{**} (2^{**} 2 // 256 // 6)$

[תשיבות](#)

[חזרה לתוכן](#)

סוגי נתונים

בפרק הקודם התייחסנו לשני סוגים נתונים מסוימים, מספרים שלמים וশברים. שפת פיתון תומכת ב有很多种 של סוג נתונים (כולל פקודות לטיפול נתונים אלה). בפרק זה נדון אך ורק נתונים מסוימים, אולם סוג נתונים נוספים יפורטו בהמשך. הטבלה הרצ"ב מפרטת את סוגי הנתונים המספריים הקיימים:

פירוט	סוג הנתון (Data Type)
מספר שלם (עם או בלי סימן)	מספרים שלמים (Integer)
שבר (חיובי/שלילי) בגודל 64 סיביות. מיוצג ע"י שבר עשרוני דוגמת: 2.57, או ייצוג המדס' 1.23e-15 (אשר משמעותו הינה 1.23×10^{-15})	শברים (Float)
מספר שלם המוגדר ע"י אפס מוביל ואחריו האות האנגלית <code>s</code> . למשל <code>0b1001001</code>	שלמים בבסיס בינארי
מספר שלם המוגדר ע"י אפס מוביל ואחריו האות האנגלית <code>o</code> . למשל <code>0o56746</code>	שלמים בבסיס אוקטלי
מספר שלם המוגדר ע"י הקידומת <code>0x</code> , למשל <code>0xACDF</code>	שלמים בבסיס הקסדצימלי
מספרים מרוכבים החלק ממשי והחלק המדומה נכתבים ע"י: <code>j</code> (2+3j, או <code>2+3j</code>)	מספרים מרוכבים

כאמור, כדי לעבוד עם שברים, גם כאשר המספרים הינם שלמים, יש לרשום את המספר בתוספת נקודה עשרונית, או שבר עשרוני (אפילו מאפס). המספר 7 מיוצג מספר שלם ואילו המספר 7.0 מייצג שבר. כל הפקודות החשבוןאיות, עובדות כMOV 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)
```

```
>>> 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
1.888888888888888
>>> 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 להגיד את התשובה, כולל סוג הנתונים שלו (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 המבצעות בין מספרים שלמים גוררות תוצאה שהיא מספר שלם (פרט למספר של חילוק שבו חילוק אפילו בין שני שלמים יגרור תוצאה שהוא שבר) ואילו פעולות בין שברים (או שברים ומספרים שלמים) גוררת תוצאה שהוא 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
```

אבל,

```
>>> 15 * 3
5.0          שבר
```

בגלל שבჭיתון לעיתים ערכים מסוימים חייבים להיות שלמים, יש לשימוש לב לפורמט התוצאה. אם אנו מחלקים, אפילו אם הפרמטרים הם מספרים שלמים, התוצאה תהיה שבר ולפניהם שוכן להשתמש בה

תכנות בסיסי

בפקודות המשמשות במספרים שלמים בלבד, נדרש להמיר את השבר ולהפכו למספרשלם. עקב בחשיבותו, הנושא יורח במסמך החוברת.

נזהה לעירק 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

[תשיבות](#)

[חזר לתוך](#)

דוגמת תזרוכת דלק

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

>>> 669 / 49.

13.653061224489797

עכשו אם ברצונו לגלות כמה עולה לו כל ק"מ (הכוונה כאן להוצאות יישירות בלבד), עליו לחלק את מחירו של ליטר דלק בתזואה שהתקבלה, פירוש הדבר להעתק את התזואה שהתקבלה. פלוני הינו דיין מאוד ולכן עליו להעתיק את כל 15 הספרות שמיין לנקודה העשrownית. כדי לחסוך את העתקה, סיבת הפיתוח של פיתון שומרת את תזואת הפעולה האחרונות והוא נגישה ע"י הסימן "_" (קו תחתית). לכן אם נניח שמחיר ליטר דלק הינו 4.80 ₪, אז הפעולה היא:

>>> 4.8 / _

0.35156950672645737

וממנה עולה שמחיר הדלק עבור כל קילומטר הינו 35 אגורות. אם פלוני רוצה לחשב כמה תעלה הנסעה מעופלה לאלית (מרחק של 470 ק"מ), עליו לכפול את התזואה الأخيرة ב – 470:

>>> _ * 470

165.23766816143495

ואז יתברר שהדלק לנסעה עולה 165 ₪.

תרגילי חזרה

התרגילים קשורים (תוצאת תרגיל אחד מזנת לתרגום העוקב). נסה להעריך את התשובה (כולל סוג הנתונים שלה) לפני שתנסה את התרגום 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 בPython.

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

[תשובות](#)

[חזרה לתוכן](#)

שמות ערכיים

הקו התיכון בפיתוח מאפשר שמיירה של הערך האחרון שחוושב. אולם לעיתים עלינו לשמר יותר מערך אחד (על מנת שלא נדרש להקלידו כל פעם מחדש). למשל אם מחיר ליטר דלק משתנה, נדרש להעתיק את מספר הקילומטרים שהמכונית גומעת בliter, כדי להשתמש בו בחישוב החדש. פתרון טוב היה יכול להיות קו תחתית לכל ערך שיחסנו....

סבירות הפיתוח פיתון אכן מאפשר מנגנון זהה, ע"י נתינת שם לכל ערך שחוושב (literal) או סימן בעברית. זו מעין הרחבה של המחשבון שתואר קודם לכן. קיימת רשיימה של מילים שמורות (ראה באתר [python.org](http://www.python.org)), אשר לא ניתן להשתמש בהן כשמות (למשל: and, del, is, for...) נניח שנרצה לערך שחוושב (kilometer per liter) kpl.kilometer per liter שם, סימן "=" (שווה) והערך אותו השם מייצג (ערך זה יכול גם להיות תוצאה חישוב כלשהו):

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

עכשו אם נרצה להשתמש בערך שחוושב, יוכל פשוט להשתמש בשם kpl. אם נרצה לראות את ערכו, פשוט נקליד kpl.

```
>>> kpl
```

```
13.653061224489797
```

השם kpl מייצג את הערך שחוושב והוא קיים כל הזמן שאנו נמצאים בסביבת העבודה. חשוב לציין שבפיתוח אותיות קטנות וגדלותן הן בעלות משמעות שונה. לכן, KPL למשל אינו kpl, אלא שם שונה (אשר במקרה שלנו לא מוגדר). באופן דומה, אנחנו יכולים להגיד שם עבור המרחק בין עפולה לאלית, למשל distance ואז נדרש לש"ר לשם גם ערך ע"י:

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

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

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

את כל החישובים שעשינו קודם, אפשר לעשות עכשו, אלא שבמקרה השימוש בערכים, יוכל להשתמש בשמות שלהם. (כאמור שמות שרירותיים שאנו קבענו). מחיר הדלק לנסעה של קילומטר אחד יחוושב ע"י:

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

```
0.35156950672645737
```

באופן דומה, מחיר הנסעה לאלית תחוושב ע"י:

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

```
165.23766816143498
```

תכנות ב-SO.

תרגילי חזרה

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

עבור עשרה התרגילים הראשונים הנה 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 הנה 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'

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

[תשובות](#)

[חזר לתוכן](#)

משתנים

ניתן להקצות שם לכל דבר, ולא רק לערכים. יתרה מזאת, ניתן לשנות גם הערכים המוצגים ע"י השמות ולכך שמות אלה נקראים משתנים. הם מייצגים "משהו" שיכול להיות מספר, מחזורת טקסט, רשימה ועוד. משתנים הם עקרון בסיסי וחשוב בפיתוח תוכנה. בדוגמה הקודרנו שמות לערכים, אך למעשה אלה היו משתנים שכן במקרה אחר, הערכים המוצגים ע"י משתנים אלה יכולים להשנות. למשל אם מחיר הדלק משתנה, או מספר הקילומטרים שהמכונית נסעה ליותר דלק וכו'. אין מוגבלת על מספר המשתנים שברצוננו להגדיר, או מספר הפעמים שאנו מושנים את ערכו של כל משתנה. בכל פעם שימושים את ערכו של משתנה, הערך הקודם נעלם ורק הערך החדש נשמר.

אם המשתנה הוגדר ואנו מקלדים אותו, סיבית הפיתוח תדפיס את ערכו. לעומת זאת, הקלדה של שם שלא הוגדר קודם לכן, תגורר הודעה 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, המשתנה חדש `a32` והכנוו לתוכו ערך שבוינו תוצאה החישוב. הקלדת שם המשתנה, גורמת לפיתון להדפיס את ערכו. לעומת זאת, כאשר ניסינו לקבל את ערכו של המשתנה `a33`, התקבלה הודעה שגיאה (`a33` לא מוגדר).

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

תכנות בסיו.

תרגילי חזרה

חשב את ערכם של המשתנים: e, f, g, h, l, 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. $l = b / d$
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: $a = 1.0$, $b = 2$, $c = 3.0$, $d = 4$
 $a = 1$, $b = 2.0$, $c = 3$, $d = 4.0$ הנח ש:

חזור לתוכן

מחוזות

בנוסף לסוגי הנטוונים המספריים, שפת פיתון גם תומכת בסוגים נוספים המיועדים למלל. מחוזת (רצף של תוים) מוגדרת בפיוון ע"י הכללה בין גרשים (גרשיים כפולים, משולשים או בודדים). פרט לעובדה שלא מדובר על ערכיהם מספריים, כל שנאמר לעיל מתקיים גם עבור מחוזות. מطبع הדברים, פעולות חשבונאיות לא מתבצעות על מחוזות, אם כי חלק מהפעולות מתבצעות בצורה שונה (כפי שiorchib בהמשך). אולם המשתנים שהוגדרו לעיל, יכולים לשמש כמשתנים מספריים וגם כמשתנים המכילים מחוזות:

```
>>> "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 הגרשים היא כדי לאפשר הכללת הגרשים כחלק מהמחוזת. למשל:

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

אולם פיתון מאפשר גם הכללת הגرش או הגרשים כחלק מהמחוזת ע"י הכללת הלוכן ההפוך לפני הגרשים:

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

הלוכן ההפוך מאפשר המשך המחזוזת בשורה הבאה (כאשר המחזוזת 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, אז בזכות האינטראקטיבית אפשר פשוט לכתב את שמו של המשתנה וazz 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. בmailto אחרות, את כל מה שאנו מבקש להדפיס, יש להכניס לסוגרים.

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 הפוקודות הבונות את המחרוזות: (בדוק את תשובה 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"

[תשובה](#)

[חזר לתוכן](#)

עוד על סוג נטוניים

כאמור, בנוסף לשוג' הנטוניים המספריים, גם מחרוזות הן סוג נטוניים (וקי' מים סוגים נוספים שיפורטו בהמשך). פיתון מספקת אפשרות להמרת סוג נטוניים וכן אפשרות לבדוק את סוג המשתנה. את בדיקת המשתנה מבצעים ע"י פקודת `type` אשר גם היא מומשה כפונקציה (פירוש הדבר את הפעמטר לפקודה יש לספק בתוך סוגרים). 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 מסווגי הנטוניים, פיתון שומרת את הסוג (למשל עבור המשתנים `e`, `a`, `b`, `c`). המשתנים אחרים (`c`, `b`) הוגדרו כמשתנים בבסיסים אחרים, אלא שלאחר שהמספר הומר לבסיס עשרוני, המשתנה הפך עשרוני גם כן.

כשם שפיתון מאפשרת לגלוות את סוג המשתנה, היא כוללת גם פונקציות ספריה אשר מאפשרות להמיר המשתנה מסוג אחד לשוג' 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
```

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

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

פונקציות ההמרה המתוארכות הן:

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

אם ברצוננו להמיר מספרים לערכים הבינרי, 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 (בנחיה, כМОון שהערכים השמורים במחרוזת מייצגים מספרים). בנוסף, גם קיימת הפונקציה ההיפוכה הממירת ערכים מספריים למחרוזות str.

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

וההמרה ההיפוכה:

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

תרגילי חזרה

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

כתב את התשובה שתתקבל מפייתן לאחר הקלדת כל אחת מהפקודות:

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

[תשובות](#)

[חזרה לתוכן](#)

פונקציות מתמטיות נוספות

בנוסף לפעולות החשבוןאיות הרגילים פיתון מאפשר פעולות נוספות:
 (x) `abs` – אשר מוחזרה ערך מוחלט של המשתנה
 y % x – שארית חילוק במספר אחר (שהוא בסיס המודול או מודולו). למשל $5 \% 2 = 1$
 משמעוito 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 החשבוןאיות ה"רגילים", פיתון כמו שפות פיתוח רבות כוללת גם ספרייה מתמטית המרחיבה את יכולות השפה. הספרייה מכילה פונקציות מתמטיות רבות, אשר זמינים למפתח התוכנה.
 כדי שאפשר יהיה להשתמש בפונקציות אלה, יש לכלול בתחלת התוכנית פקודה:

```
Import math
המשמעות היא שאנו מבקשים ל"יבא" לתוכנית שלנו את היכולות המוגדרות בספריה המתמטית.  

כדי לפונקציה מסוימת המוגדרת בספריה علينا להוציא את הקידומת math. למשל אם ברצוננו
לקרא 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 לpermeter שמועבר לה.
 למשל:

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

2. `copysign` – מקבלת שני מספרים כפרמטרים ומוחזרת תוצאה אחת שהיא הpermeter הראשון
 (בפורמת של שבר), כאשר סימנו הוא סימן המספר שהועבר כפרמטר השני:

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

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

.3 – מקבלת פרמטר אחד שהוא מספרשלם וחובי ומחזירה את העצרת שלו:

```
>>> math.factorial ( 5 )
120
>>> math.factorial ( 5.5 ) <<<<<<<
Dagemat Shevra <<<<<<<
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 ) <<<<<<<
Dagemat Mafsaf Sheliy <<<<<<<
Traceback (most recent call last):
  File "<pyshell#14>", line 1, in <module>
    math.factorial(-3)
ValueError: factorial() not defined for negative values
```

.4 – מחזירה את המספר השלם הגדל ביותר שהוא עדין 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 – 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 – מקבלת פרמטר אחד (גודל מאפס וחובי) 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
```

```
>>> 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 הfrmטט הראשן בחזקת frmטט 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 frmטט 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) המקובלות frmטט ברדיאנים כמו כן שתי פונקציות שנועדו להמרה בין מעלות לרדיאנים (radians – i degrees)

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

תכנות בPython.

10. קבועים – e ו- pi:

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

תרגילי חזרה

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

כתבו את התשובה שתתקבל מפיטון 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. $d \% c \% b$

PLITK_B'

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

[תשובות](#)

[חזר לתוכן](#)

טיפול במחוזות

כאמור, מחוזת היא רצף תווים (טקסט) והיא מוגדרת בין גרשים (בודדים, כפולים או משולשים). פיתון מספקת פונקציות מיוחדות המאפשרות לטפל במחוזות, לשולף מהן נתונים ו גם לאחד (לחבר) מחוזות.

בניגוד לערכים מספריים, השמורים במחשב בצורה בינהית, המוחזק שמוורה כמלול וכן ע"י שימוש במפתח (אינדקס) הנמצא בסוגרים מרובעים וצמוד למחוזת, ניתן להתייחס לכל אחד מהתווים שבמחוזות. מיקום התווים מתחילה מצד שמאל עם המספר 0 וממשך ימינה עד -1, כאשר ח מייצג את אורך המציגת. למשל, נגידר מחוזת a המכילה את המלול Shalom. במחוזת מייצגת בזיכרון המחשב רצף תווים כתוואר בטבלה הרץ"ב:

S	H	a	l	o	m	תו
0	1	2	3	4	5	מיקום

בדוגמה, נשלוף מהמחוזת 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 מחוזות בגודל משתנה, פיתון מאפשר גישה לתווים שנמצאים מצד ימין. הדבר מתבצע ע"י שימוש במפתח שלילי (1- הוא התו הימני ביותר, 2- התו שלושמו ון הלהה), 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'
```

המפתח שתוואר לעיל אפשר לשולוףתו אחד בלבד. לעיתים יש צורך לשולוף מספר תווים (תת-מחרוזת). פיתון מאפשרת את השליפה ע"י הרחבות מגנון המפתחות ע"י שימוש בשני פרמטרים – מיקום התו הראשון לשילפה ומיקום התו האחרון (פלואו אחד) המופרדים ע"י נקודות. 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, המפתחות (כתובות התחילה והסיום) יכולים להיות ביטויים הכללים נוסחאות מתמטיות שונות (ובclud שתוואר הנוסחה תהיה מספר שלם. אין גם חובה לציין את שני הביטויים. אם הם חסרים, פיתון תשלים את כתובות הקצה. לעומתם אם הביטוי הראשוני חסר, הוא יוחלף באופן ואילו אם הביטוי השני חסר, משמעו הדבר עד התו האחרון במחרוזת. כפי שניתן לראות בדוגמה:

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

בפיתון קיימות עוד מספר פונקציות שנעדו לעזור בטיפול במחרוזות:
+ (concatenation) שמשמשות להצמד שתיים (או יותר) מחרוזות כדי ליצור מחרוזת
גדולה.
* שכפול המחרוזת מספר פעמים
(x) len – מחזירה את אורךה 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"
```

תכנות בPython

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

תרגילי חזרה

רשום את התשובה שתתקבל מפתרון 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]

הדפסה

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

בשביבת האינטראקטיבית, הקלדת שם משתנה גורמת להדפסת ערכו (הציגו על גבי המסך), אולם כאשר כתבים תוכנית, יש צורך בפקודה מיוחדת להדפסה. פקודה זו היא `print` –

בצורתה הבסיסית הפקודה מדפיסה את הפרמטרים שקיבלה (אף או יותר). אם לא מספקים פרמטרים היא תדפיס שורה ריקה (למענה תקדם את הסמן לשורה הבאה). אם הפרמטרים הם משתנים, פיתון תדפיס אותם בהתאם לסוג. לדוגמה תוויסף הנΚודה העשוי. אולם אם הביטוי בפקודה הוא לא משתנה, אלא ערך, פיתון תדפיס אותו, כולל ביצוע הפעולות הנדרשות. למשל במספרים, בעלי פעולה אריתמטית, פיתון תdag לבעיה ותדפיס את התשובה.

נקודה חשובה שיש לזכור היא העבודה שפקודת הדפסה בפיתון מומשה כfonction ולכן את הפרמטרים שלה יש להכניס לסוגרים. יתרה מזאת, גם אם רוצים "להדפיס" שורה ריקה יש להשתמש בפקודת הדפסה, כאשר היא מלאה בסוגרים ריקים: () `print()`

דוגמאות לשימוש בפקודת הדפסה:

```
>>> 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
```

קדימות הפעולות המתמטיות הן כפי שהסביר בפרק הדן במחשבון.

תרגילי חזרה

רשום את התשובה שתתקבל מפתרון לאחר הקלדת כל אחת מהפניות:

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](#)

[חזרה לתוכן](#)

עורך על הדפסה

במה שפרק זה מגדיר מחרוזות ולפרק זה הדרישה, עליה לעתים הצורך לשלב אתתו הסימן (גראש או גרשאים) בתוך המחרוזת. לצורך כך הוגדרתו מיוחדת Escape sequence שמאפשר יכלולת נוספת:

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

לאחר שהוגדרתו, הוא מאפשר יכלולת נוספת, כמו הוספת מעבר שורה באמצעות המחרוזת (\n), הוספת TAB (\t) והגדלת יכולת להוציא כל תו (אפילו אלה 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
```

תרגילי חזקה

רשום את התשובה שתתקבל מפתרון לאחר הקלדה כל אחת מהפקודות:

תשובה

חזר לתוכן

לולאות

לפעמים בתוכנית علينا לבצע רצף פקודות בצורה מחרוזית. 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 מחרזר ערכו משתנה, עד אשר הלולאה מסתיימת.
- `<sequence>` מייצג רצף כלשהו, אשר מגדר את מספר מחרזר הביצוע ואת ערכי מונה הלולאה.
- `<body>` מייצג את קצף הפקודות שיש לבצע בכל מחרזר של הלולאה.

את פקודות ה – `for` מסיים בנקודותים ושרה חדשה, ואחריהם סבית הפיתוח מבצעת הזחה זהה המוקם להקליד את גוף הלולאה (רצף הפקודות לביצוע). במקרה שגוף הלולאה כולל פקודה אחת בלבד, ניתן להקלידה לאחר הנקודותים באותה שורה.

חשוב לציין שבעת ביצוע לולאת `for`, פיתון יודעת את מספר המחרזרים שעלייה 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 בסוגרים (נתונים מסווג רשיימה, שתוסבר בהמשך)קובעים את הערכים של מונה הלולאה. גופ הלולאה במקרה זה כולל רק את פקודת ההדפסה של מונה הלולאה. ניתן לראות את ערכו של מונה הלולאה בכל מחרזר. ערכים אלה חופפים את הערכים אשר נמצאים בראשימה.

כדי לתמוך בפעולות שונות ובין היתר גם בלולאות, הוגדרה בפייתון פונקציה עזר `range`. הפונקציה מקבלת פרמטר מסווני אחד (`n`) ומחייבת את רצף המספרים `[0,1,2,...n-1]`. אך 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
```

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 המרוחה 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 לולאה מבוססת `range`:

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

```
3
10
17
24
```

במקרה זה, הרץף החל במספר 3 ובכל מחזור התקדם בשבע. لكن 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 ה被执行.

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

תרגילי חזרה

רשום את התשובה שתתקבל מפתרון לאחר הקלדת כל אחת מהפקודות (בדוק את תשובהך). בגלל שהלוואות רשות בשורה אחת, יש צורך בהקשת Enter נוספת, כדי שפיטון תritz את הקוד. חלק מהתרגילים, בעיקר האחרונים שבהם כוללים התייחסות לפוקודות שופיעו בפרקם הבאים בחומרה.

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 ("*\t*" * 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 וקינון 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)

```

```
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](#)

[חזר לתוכן](#)

## ביטויים בוליאניים

מספרים בוליאניים הם סוג נתונים נוסף (מעבר לאלה שתוארו בפרק 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 בפייתון משתמשים בסימנים מוכרים (להלן) על פי הפירוט:

| משמעות       | תנאי מתמטי | תנאי בפייתון |
|--------------|------------|--------------|
| קטן          | <          | <            |
| קטן שווה     | $\leq$     | $\leq$       |
| שווה         | =          | ==           |
| גדול או שווה | $\geq$     | $\geq$       |
| גדול         | >          | >            |
| שונה         | $\neq$     | $\neq$       |

## 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)
```

## תכונות בO/O.

```
True
>>> a / /b == b // a
False
```

## תרגילי חזרה

רשום את התשובה שתתקבל מפתרון לאחר הקלדה כל אחת מהפניות (בדוק את תשובהך).

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 ( )

|                        |
|------------------------|
| <a href="#">תשובות</a> |
|------------------------|

|                            |
|----------------------------|
| <a href="#">חזרה לתוכן</a> |
|----------------------------|

בפרקים הקודמים התייחסנו לסביבה האינטראקטיבית של פיתון. במקרים רבים סביבה זו אינה מספקת לנו גדרים לפתח תוכנית ממש. הדבר מתבצע ע"י הקלהת פקודות התוכנית לקובץ מלל (טקסט) בעל סימול של txt, או עק. מתוך הסביבה האינטראקטיבית ניתן לטען את הקובץ, לשנותו אם צירך ולהריצו (ע"י חנוך או שימוש ב – F5). חלק מהתוכניות חייבות לקבל קלט מהמשתמש על מנת שתוכנה תזרז.

הפקודה המאפשרת קליטת נתונים (זמן ריצה) מהמשתמש היא פקודת ה – `input` (למעשה כמו פקודת הדפסה גם זו פונקציה, אולם נשא הפונקציות יורחב באחד מפרקיה המשך).

הfonktsia הפשוטה – `input` נועדה לקבל קלט טקסטואלי מהמשתמש והיא מכינה אותו אל משתנה על פי שהוגדר בתוכנית. כדי להקל על הפיתוח, פונקציית ה – `input` מאפשרת גם להדפיס מחרוזת לפניו בקשת הקלט מהמשתמש (כדי להסביר למשתמש מה נדרש ממנו). לדוגמה:

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

בדוגמה, השתמשנו בפונקציית הקלט, כדי לקרוא מחרוזת טקסט שהוקשה במקלדת ולהכינה למשתנה `a`. בנויגוד להשמה מרובה של ערכים, פונקציית קלט עובדת בצורה שונה. חשוב להציג 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 שאם אם הקלדנו מספרים, הם מוחזרים כמחרוזות. וכן, כאשר הדפסנו אותם פי שלוש, קיבלנו את הערך: 123123123. כאשר רצינו להשתמש בהם כערכים מספריים המרטטו אותם (ע"י `int`) ועוד כאשר הדפסנו את הערך פי 3 קיבלנו 369. אפשר כמובן לבצע את המרה ישירות על תוצאה הקלט:

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

## תרגילי חזרה

רשום את התשובה שתתקבל מפייתן לאחר הקלדת כל אחת מהפניות (בדוק את תשובה).

1. 

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

בנהנча 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))
```

בנהנча שהמשתמש הרץ את התוכנית מספר פעמים ולהן הקלט של כל ריצה:

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

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)
```

בנהנча שהמשתמש הרץ את התוכנית מספר פעמים ולהן הקלט של כל ריצה:

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

4. 

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

בנהנча שהמשתמש הרץ את התוכנית מספר פעמים ולהן הקלט של כל ריצה:

|   |    |
|---|----|
| 1 | .1 |
| 2 | .2 |
| 3 | .3 |
| 5 | .4 |
| 7 | .5 |

## תכנות בPython

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

בנהנה שהמשתמש הרץ את התוכנית מספר פעמים ולהן 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)
```

בנהנה שהמשתמש הרץ את התוכנית מספר פעמים ולהן 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)
```

בנהנה שהמשתמש הרץ את התוכנית מספר פעמים ולהן 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])
```

בנהנה שהמשתמש הרץ את התוכנית מספר פעמים ולהן 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)
```

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

|         |    |
|---------|----|
| abcd    | .1 |
| 1234567 | .2 |
| ab      | .3 |
| r       | .4 |
| 9753    | .5 |

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

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

|          |    |
|----------|----|
| 12345678 | .1 |
| 90123456 | .2 |
| abcd     | .3 |
| n        | .4 |
| 01010101 | .5 |

```
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, "</>" , b)
```

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

|           |    |
|-----------|----|
| 12345678  | .1 |
| 901238456 | .2 |
| abcd      | .3 |
| n         | .4 |
| 01010101  | .5 |

[תשובות](#)

[חזרה לתופס](#)

## תנאים (פקודות התניה)

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

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

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

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

הפורט של התנאי (`condition`) הוא: `>expr< <expr> <relop> <expr>`, כאשר: `>expr<` מציין ביטוי כללי (משתנה, פעולות על משתנים ואילו קבועים) ו- `<relop>` מציין יחס מסוים (כמו היחסים הבוליאניים שהוצגו בפרקים קודמים).

למשל אם ברצוננו לפתוח 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 וכן שפט פיתון תומכת גם במבנה זה ע"י הרחבות ה – `if`. הפורט במקרה זה הוא:

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

פירוש הדבר שרצף הפקודות המוגדרות ע"י `<statements1>` מבוצעות כאשר התנאי מתקיים ואם אינו מתקיים, ממבצע רצף הפקודות `<statements2>`. מטיבם הדברים, פורט פקודת זה מתאים כאשר אנחנו בודקים תנאי והיפכו.

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

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

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

דוגמאות 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. אמן אפשר לפתור את הבעיה ע"י קינון של תנאים, אולם צריך להיזהר שלא לפגוע בקריאות התוכנית והאפשרות לתזקק אותה בעtid. שפת פיתון תומכת גם בתנאים מרובים ע"י הוספה רכיבים לפקודת ה – if.

הפורמט המורחב הוא:

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

else:
 <statementsn>
```

המשמעות של if היא elseif או שהיא אפשרות 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, יבוצעו הפקודות שבבלוק שלו ומיד לאחר מכן ימשיר הביצוע בפקודה העוקבת לאחר הבלוק האחרון של ה – if. אם התנאי אינו נכון, יבדק התנאי המופיע בבלוק הבא. אם הוא נכון, יבוצעו הפקודות בבלוק שלו וכן הלאה. אם אף תנאי אינו נכון, יבוצעו הפקודות הרשותות תחת ה – else. אולם, אין חובה להגדיר את בלוק ה – else. במקרה זה, אם אף אחד מהתנאים אינו מתקיים, לא יבוצע כלום בפקודת ה – if הזאת, במהלך הריצה.

דוגמה לתוכנית המבוססת על דירוג הציונים האירופאי:  
 0-59 כישלי,  
 60-69 מתחת למוצע,  
 70-79 ממוצע,  
 80-89 מעל למוצע,  
 90-100 מצין.

```
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 ה蟋זה:

```
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 את תוצאה הרצויה. (בדוק את תשובה!)

```
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 תשובה אם את פקודת ה – if מחליפים ב:  $k \neq j$

```
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.
cn t= 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 = "abcdefghijklm"  
 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](#)

[חזרה לתוכן](#)

## רשימות

מחוץ להן סוג מסוים של רצפים sequence, אך קיימים סוגים נוספים (שלא על כלם נבעור במסגרת מדריך זה). סוג שימושי מאוד, אשר כבר הופיע בדוגמאות בפרקם הקודמים (ambil להגדיר את שמו) הוא רשימה.

רשימה היא סדרה של נתונים, אשר בניגוד למחוץ, יכולים להיות מסוגים שונים. כמו למשל ברשימה יכולים להיות נתונים מספריים, תווים, שברים וכו'. למשל:

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

כל הפעולות הקיימות לטיפול במחוץ זמינים גם עבור רשימות, אך בנוסף קיימם הבדל משמעוני אחד. רשימות ניתנות לשינוי, בעוד מחוץ לא אפשרות 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 בצורה 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 לערך נדרש פקודת ה – if בתוכנית?<sup>4</sup>

---

<sup>4</sup> ברשימות, כמו גם במחוץ, אינדקס בעל ערך שלילי מסמן שיש להתחילה את האיברים מסוף הרשימה. ללא פקודת ה – if, יווצרו אינדקסים שליליים ואז למשל מי שקיבל ציון 0, קיבל הערכה של !!Excellent!!

## תרגילי חזרה

רשום במשפט אחד מה קטע הקוד מבצע את תוצאה הרצויה. (בדוק את תשובהך).

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')
 else:
 print (a, 'is not')
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 בPython

```
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](#)

[חזר לתוכן](#)

## לולאות while

בניגוד לולאות ה – `for` שפורטו בפרקם קודמים ואשר הין מוחלטות (בכונסה לולאה, מספר המחרוזים כבר ידוע), לולאת `while` הן לולאות שאין מוחלטות, כלומר בכל מחזור נבדק מחדש התנאי אם יש לבצע את המחזור או לא. פורמת הפקודה הוא:

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

כמו במקרה של `for`, התנאי הוא תנאי בוליани וגוף הלולאה מכיל פקודה אחת או יותר. הבדל משמעותי בין שני סוגי הלולאות הוא בכךם מונה הלולאה. לולאת `for` מקדמת את המונה עבורנו, בעוד שבולולאת `while` לעיטיות אין מונה ובכל מקרה, علينا לקדם בעצמו. אם לא נקדם אותו, או לא נdag לשינוי התנאי הבוליани, הלולאה תבצע עד אשר נפסיק אותה ידנית (למשל ע"י `Cntrl C`).

לולאת `while` עילה, למשל כאשר علينا לקרוא קלט מהמשתמש ואין אפשרות (קללה) להגדיר מראש את כמות הקלט (או מספר האיברים שנתקבל). במצב, זה, ניתן להגדיר סימן מיוחד שmagdir את סוף הקלט. זה כמו ח"ב להיותתו שאין 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)
```

וביצעה מתקובל:

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

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

Width: -2

## תכונות ב-SQL

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 יכנס את הקלט הרצוי (מספר חיובי גדול מאפס).

**תרגילי' חזרה**

רשום את תוצאות הריצה. (בדוק את תשובה).

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 ולאחר כר 60 אחר כר 23 אחר כר 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 תשובה אם משנים את הפוקודה 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

```

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

[תשובות](#)

[חזר לתוכן](#)

## פונקציות

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

פונקציה היא מעין תחנת תוכנית בתוך התוכנית ולכן יש לה שם ייחודי (לפי מזהים ומפעלים אותה). לכל פונקציה חייב להיות קטע בו היא מוגדרת (`Function Definition`). לפונקציה אפשר גם להעיבר פרמטרים (בשמות משתנים בתוך הסוגרים). לכן, אם למשל נגדיר פונקציה אשר מקבלת שני פרמטרים ומחזירה את מכפלתם:

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

בכל פעם שנקרוא 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 להגיד חלק מתוכנית ואז יש להקצתו משתנה, אשר יקבל את תוצאה 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 תוצאה הפונקציה אפשר גם לשלוח ישירות להדפסה ע"י:

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

מה לדעת תהיה תוצאה הפקודה الأخيرة?<sup>5</sup>

כאשר לפונקציה מספר תוצאות, למשל שתיים, יש להגיד שני משתנים אליום יוכנסו התוצאות. הפקודה המחזיר את התוצאות היא `return`. למשל:

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

<sup>5</sup> עקב העבודה שאחד הפרמטרים הוא מחזורת, פקודת הכפל שבתוך הפונקציה לא תבוצע כפודת כפלי, אלא תהפוך ל"חזרה" (או שכפול). במלים אחרות המחרוזת "2" תשוכפל שלוש פעמים והפונקציה תדפיס: 222

## תכנות בס.ו.ו.

והקריאה תתבצע ע"י:

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

## תרגילי חזרה

רשום את תוצאות הריצה. (בדוק את תשובה).

```

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 multiprnt (x, t) :
 for j in range (x) :
 print (t)
```

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

[תשובה](#)

[חזרה לתוכן](#)

## עוד על פונקציות

במה שקדם להסביר על הפונקציות, יש להוסיף שלפונקציות בעלות פרמטרים ניתן להגיד ברירות מחדל. פירוש הדבר שהפונקציה עובדת כפי שתואר, אך אם כאשר קוראים לה לא מגדירים את אחד הפרמטרים, הפונקציה משתמשת בערך בירית המוחדר בה. הדבר נעשה ע"י הוספה סימן שווה וערך לכל פרמטר שאנו רוצים להקצתו לו בירית מוחדר. לדוגמה משתמש 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 יכולת של פונקציה אחת לקרוא עצמה (עד אשר מתקיים תנאי מסוים). יכולה זו נשענת על העבודה 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 תוצאה הרצויה. (בדוק את תשובה).

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](#)

[חזרה לתוך](#)

## טיפול בראשימות

בפיטון מוגדרות מספר פונקציות ספירה אשר נועד לספק יכולות מורחבות בטיפול בראשימות. כדי לקבל גישה לפונקציות אלה, יש לכלול בתוכנית את הפקודה: `import string`.

פקודה זו מיבאת את הפונקציות המאפשרות יכולות אלה ולא הפקודה, אין אפשרות להשתמש בהן (שימוש בפונקציות ללא פקודה היגור הוודעת שגיאה).

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

בנוסף לאפשרויות לטיפול בראשימות שתוארו בפרק 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 לבצע על איברי הרשימה ע"י שימוש בפונקציה `iter`. למשל עבור הדוגמה הקודמת ניתן להגיד משתנה נניח k אשר ישמש כמונה האיברים בראשימה. בכל פעם שנקרא לפונקציה `(k)next` מקבל את האיבר הבא בראשימה.

```
>>> 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` המחזיר את אורך המחרוזת או הרשימה, כבר אנו מכירים. קיימות פונקציות נוספות כמו למשל הפונקציה `sum`, אשר מחזירה את סכום האיברים בראשימה. למשל:

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

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

בנוסף לפונקציה המסכמת את איברי הרשימה קיימות כמובן גם פונקציות המוחזירות את האיבר הקטן ביותר והגדול ביותר בהרשימה. 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. העתקה הרגילה לא יוצרת רשימה נוספת, אלא יוצרת רק הפניה 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]
```

לכוארה, העתקנו את הרשימה ל – x, אך כפי שניתן לראות לאחר ששינו את x, גם y השתנתה. האיבר שמספרו 3 (האיבר הרביעי) היה 67, אך לאחר ששינו אותו ב – x ל – 11, גם ב – y ערכו הפר 11. במקרה אחרות, x ו y מצביעים אותה 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]
```

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

במקרה זה שינוינו את x והרשימה y לא השנתנה.

בנוסף, קיימות גם פונקציות אחרות המאפשרות להרחיב את הרשימה: `append` מוסיף איבר אחד בסוף הרשימה, `extend` מוסיף איברים מרשימה (או מחוץ לה) אחרית - `insert` מוסיף איבר במקומות מסוימים, תוך הuzzת 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 `z` בעלת שלושה איברים. בפקודת ההחלפה אנו מחליףים את שני האיברים הראשונים ברשימה `t` באיברים מהרשימה `z`.

```
>>> 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` זמיןela צורך ביבוא הספרייה לטיפול ברשימות ומחרוזות (`import string`). למשל, המחרוזות `t` כוללת שמותם של שלושה איברים. בפקודת המחקה הראשונה, אנחנו מוחקים את האיבר הראשון ובפקודת המחקה 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 מחקה נוספות לשולף (תוקן מחקה) של איבר מסוים כלשהו ברשימה – `remove` אשר מוחקת איבר לא לפי מיקומו, אלא לפי ערכו. פונקציית `remove` – סוק ללא פרמטרים מוחקת את האיבר האחרון ברשימה ולחילופין היא

יכולת לקבל פרמטר אחד המציין את מספרו של האיבר למחיקה. הפונקציה מחזירה את הערך שנקרא remove – מתקבל פרמטר אחד שהינו ערך של איבר בראשימה והוא מוחפש בראשימה את המופיע הראשון של ערך זה ומוחקתו 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]
```

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 מוחפשת את המופיע הראשון של 2 ומוחקת את האיבר המתאים. בקריאה השנייה לפונקציה, שוב היא מוחקת את המופיע הראשון שהוא מוצאת. לאחר שלא נשארו מופיעים של 2 בראשימה, מקבלת הודעה שגיאה.

לפעמים יש צורך לסדר את הרשימה בסדר הפוך (או לחילופין להוסיף לה איברים בתחליתה – 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 בPython

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 נמצוא במקומות מס' 3 (המקום הרביעי) ברשימה. פונקציה שימושית נוספת היא הפונקציה `count`, אשר מחזירה את מספר המופיעים 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 את תוצאה הריצה. (בדוק את תשובה).

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](#)

[חזרה לתוכן](#)

## פעולות בוליאניות

סוג הנתונים הבוליאניים (משתנה בעל שני מצבים) הוגדרו כבר בפרקים קודמים. אולם הם משמשים כנתאים בביטויים פΚודות, אך יש להם שימושים נוספים. קיימות פעולות בין מספרים בוליאניים ובתרגול זה נתיכון לשלוש הפעולות העיקריות:

**and** – בהיבט של "גם", אשר מקבל שני אופרנדים בוליאניים ומחזיר תשובה שהיאאמת (True), רק אם שניהם true.

**or** – מחזיר תשובה אמת אם אחד משני האופרנדים אמת. במקרה אחר, or מחזיר תשובה שקרית אם ורק אם שני האופרנדים שקרים.

**xor** – (Exclusive or) שדומה ל – or בהבדל אחד שהוא מחזיר תשובה שקרית אם שני האופרנדים שקרים וגם אם שניהם אמת. במקרה אחר, xor מחזיר תשובה אמת אם שני האופרנדים שונים (אחד אמת והשני שקר).

בנוסף, קיימת גם פעולה **not** המקבילה לאחד ומחזירה את ההפכי שלו. אם הוא אמת, היא מחזירה שקר ואם הוא שקר, היא מחזירה אמת.

בביטוי מורכב, סדר הקידמיות הוא:

not  
and  
or

ניתן תמיד להשתמש בסוגרים 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
```

```
>>> 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 וילו 0 נשאר שקר. במחוזות הדבר שונה מעט מהרזה 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 שת חוסף | במשאים.

ב `y and x` – קודם מערכים את x. אם הוא שקר (פירוש הדבר שכל הביטוי שקר) אז מחזירים את x כתוצאה הניתוח. אם הוא אמת, (פירוש הדבר שתוצאה הביטוי נקבעת על פי y) מחזירים את y כתוצאה הניתוח.

ב `y or x` – מערכים את x. אם הוא אמת (פירוש הדבר שהביטוי אמת) ואז מחזירים את x כתוצאה הניתוח. אם הוא שקר, פירוש הדבר שתוצאה הניתוח נקבעת על פי y ואז מחזירים את y כתוצאה הניתוח.

## תרגילי חזרה

רשום את תוצאה הרצויה. (בדוק את תשובה).

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 תשובה אם את השורה הראשונה משנה ל – ?x = 31 או ל – x = 23

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](#)

[חזרה לתוך](#)

## פונקציות לטיפול במחרוזות

כפי שראינו, בפיתוח קיימות פונקציות ספרייה מוכנות מראש שנעדו לטפל ברשימות. אולם פונקציות אלה היו רק חלק מתוך ה欽ע הק"ם. פונקציות נוספות נועדו לספק יכולות לטיפול במחרוזות. לכל הפונקציות מבנה אחד הכלול בשתי מיללים המופרדות ע"י נקודה ולאחריה סוגרים עם או בלי פרמטרים. המלה הראשונה מצינית את שם המחרוזת אליה ברצונו להפעיל את הפונקציה ואילו המלה השנייה מגדרה את הפונקציה. לפעמים הפונקציה כוללת גם פרמטרים (חובב ורשאות) ואלה יופיעו בסוגרים לאחר שם הפונקציה. אם אין פרמטרים, יופיעו סוגרים ריקים.

1. מחרידה 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 שבתחילת המחרוזת כלל רווח כתו ראשון וכן כאשר ייצרנו העתק (b), האות השנייה לא הפכה לאות גדולה (משמעותה שנייה במקום הראשון במחרוזת). בדוגמה השנייה, במודרך כך שהתו הראשון הוא אות וכן במקרה זה הפונקציהcapitalizes את החליפה את האות הראשונה באות גדולה.

2. center – מחרירה מחרוזת שהיא העתק של המחרוזת המקורי כאשר היא ממורכצת בהתאם לפרמטרים הנוספים בפונקציה. פרמטר ראשון הינו חובה והוא מגדיר את גודל המחרוזת החדשה (אשר בתוכה תמצא המחרוזת המקורי). פרמטר שני שהוא רשوت מגדיר את תו המילוי. ברירת המחדל היא רווחים. אם הפרמטר הראשון קטן מגודל המחרוזת, פיתון עדין תחזיר את המחרוזת בשלמותה.

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

3. count – מחרירה את מספר המופיעים של תת המחרוזת המוגדרת כפרמטר ראשון. פרמטר שני שהוא רשوت מגדיר את מספר האיבר הראשוני בחרוזת המקורי ממנו מתחילה לסרוק (ברירת המחדל היא אפס – או תחילת המחרוזת) הפרמטר השלישי שהוא רשوت מגדיר את האיבר האחרון לסריקה (ברירת המחדל היא האיבר האחרון במחרוזת).

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

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

.4 – פונקציה בוליאנית המחזיר את (True) או שקר (False), עם המחרוזת המסופקת כפרמטר ראשון לפונקציה נמצאת בסוף המחרוזת המקורי. כמו במקרה של הפונקציה count, גם כאן ניתן גם להשתמש בשני פרמטרי רשות המגדירים את גבולות המחרוזת הנבדקה (ברירות המהידל הן מתחילה המחרוזת ונגמרה (ברירות המחרוזת ועד סופה).

```
>>> a = "abcbcabbacabbcc"
>>> 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 – מחזירה את המיקום הראשון שבו המחרוזת המועברת כפרמטר נמצא בתוך המחרוזת הנבדקה. כמו במקרים קודמים ניתן להגדיר את טווח החיפוש ע"י שימוש בשני פרמטרי רשות אשר מגדירים את גבולות המחרוזת הנבדקה (ברירות המהידל הן מתחילה המחרוזת ועד סופה). המספר המוחזר הוא המיקום במחרוזת המקורי, גם אם החיפוש הוגבל רק לחלק מהמחרוזת. אם המחרוזת (הפרמטר) לא נמצאת במחרוזת המקורי, 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 – פונקציה זהה ל – find בהבדל אחד. אם index לא מוצאת את המחרוזת במחרוזת המקורי, היא מוחזירה ValueError (במקום ה -1 - אשר find מוחזירה).

```
>>> 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 במחוזת כלם תווים אלפאנומריים (אותיות או מספרים) ובתנאי שבמחוזת 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 במחוזת כלם תווים (אותיות) ובתנאי שבמחוזת 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 במחוזת כלם ספרות ובתנאי שבמחוזת 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 במחוזת כלם אותיות קטנות בלבד שבמחוזת 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. `upper()` – פונקציה בוליאנית (הפוכה ל-`islower()`) אשר מחזירה אמת אם הערכים במחרוזת כולם אותיות גדולות בלבד שבמחרוזת לפחות 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 מתחילה באות גדולה ומשיכות באותיות קטנות (כמו סימן או מארט) 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 מאיובי 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 בנייגוד ל-`center` אשר ממרכזת אשר המחרוזת המקורית, `ljust`, מחזירה אותה כאשר היא מוצמדת לצד שמאל. פרמטר ראשון הינו פרמטר חובה והוא מגדר את גודל המחרוזת החדשה (אשר בתוכה תמצא המחרוזת המקורית). פרמטר שני שהוא רשות מגדר את תו המילוי. ברירת המחדל היא רווחים. אם הפרמטר הראשון קטן מגודל המחרוזת, פיתון עדין 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, תחזיר 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 שקוץץ ממנו חלק בצד השמאלי. הפונקציה מקבלת פרמטר רשות אחד המגדיר את התווים שיש להשמיט מהחרוזת המקורית. ברירת המחדל היא השטמת הרוחחים שנמצאים מצד שמאל

```
>>> a = "abcdefghijklm 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 מגדר את הערך שיוכנס במקומו. כל המופעים של האיבר שהוגדר בפרמטר הראשון יוחלפו ע"י האיבר שהוגדר בפרמטר השני, אלא אם כן משתמש בפרמטר שלישי שהוא רשות. פרמטר זה קובע את מספר האיברים שיש להחליף.

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

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

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

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

```
>>> 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. `rjust` מוצמidea את המחרוזת לצד שמאל ואילו `ljust` מוצמidea אותה לצד ימין. פרמטר ראשון הינו פרמטר חובה והוא מגדיר את גודל המחרוזת החדשה (אשר בתוכה תמצא המחרוזת המקורי). פרמטר שני שהוא רשות מגדיר את תו המילוי. ברירת המחדל היא רווחים. אם הפרמטר הראשון קטן מגודל המחרוזת, פיתון עדין 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 המקורי. לפונקציה שני פרמטרי רשות. פרמטר ראשון מאפשר להגדיר את تو הפיצול (ברירת מחדל היא רווח) ופרמטר שני קובע את מספר הפיצולים. אם تو הפיצול לא נמצא בחרוזת, מוחזר עותק שלה (לא מפוץ). 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 לו split, rsplit מתחילה לפצל החל מצד ימין. לפונקציה שני פרמטרי רשות. פרמטר ראשון מאפשר להגדיר את تو הפיצול (ברירת מחדל היא רווח) ופרמטר שני קובע את מספר הפיצולים. אם تو הפיצול לא נמצא בחרוזת, מוחזר עותק שלה (לא מפוץ).

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

23. rstrip – פונקציה אשר מחזירה מחרוזת שהיא העתק של המחרוזת המקורי לאחר שקוצץ ממנה חלק בצדיה הימני (פונקציה הפוכה לפונקציה strip) שתוארה לעיל). הפונקציה מקבלת פרמטר רשות אחד המגדיר את התווים שיש להשמיט מהמחרוזת המקורי. ברירת המחדל היא השטת הרווחים שנמצאים מצד 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 שկוצצו ממנה תוים מסוימים צדדיים. הפונקציה מקבלת פרמטר רשות אחד המגדיר את התווים שיש להשמיט מהמחרוזת המקורית. בירית המandal היא השמטת הרוחמים 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的大גדולות הפכו לקטנות.

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

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

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

27. upper – פונקציה לא פרמטרים אשר מחזירה מחרוזת שהיא העתק של המחרוזת המקורית לאחר שכל האותיות הופכות לאותיות גדולות (הפור 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, עדין יוכנסו אפסים מובילים מצדיה השמאלי. אם הגודל החדש קטן מגודלה של המחרוזת המקורית, הוא תועטג 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 תשובהך). התרגילים קשורים האחד לשני.

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](#)

[חזרה לתוכן](#)

## עיצוב מחרוזות

במהדורות הקודמות של פיתון, עיצוב מחרוזות המודפסות נעשה ע"י הוספת הסימן % במחזורת (במקום או מקומות בהם נדרש העיצוב). פירוש הסימן הוא שבמוקום בו הוא מופיע יש להכניס ערך אחר שמוופיע 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 מחזורת, אלא שבמחזורת שילבנו את הסימן %. סימן זה מציין שברצוננו להחליפו לפני הדפסה. מה שיחליף אותו זה ערך (או משתנה המציג ערך) שנמצא בהמשך השורה (לאחר סוף המחרוזת) וגם הוא מוגדר ע"י הסימן %. لكن בזמן עיצוב הדפסה, מודפסים התווים שבמחזורת עד אשר מגיעים למספר %. אז מدلגים לסוף המחרוזת ומוחפשים את הערך הראשון שבמחזורת עד אשר הסימן % ומשלבים אותו בהדפסה (במקום ה - %). ממשיכים להדפיס את תוכן המחרוזת, עד אשר היא מסתיימת או מוצאים סימן % נוספת (אשר גם אותו צריך להחליף). הערכים שיש לשלב במחזרזת, 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 בהתאם בערכים הנמצאים בתוך המשתנים d, a, b, c, d.

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

```
>>> print ("1st=%d, 2nd=%d, 3rd=%d, 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 הופיעו הסימן % ואחריו האות d שעד עתה לא התיחסנו אליה. אותן (וונוספות) מדירותה את פורמט הדפסה. d מציין מספרשלם ולן השורה שהודפסה כללה מספרים שלמים. אם ברצוננו להדפיס את המספרים כמספרים, 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 הדפסה הקודמת אפשר היה להשיג גם באמצעות פקודת הדפסה פשוטה, כדי לצורך הדוגמה האחרונה, היינו אמורים להחליף את סוג המשתנה. אם d, a, b, c, הם שלמים, אז היינו צריכים קודם להמיר אותם לשברים ורק לאחר מכן יוכל 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 שאינם מנוקדת העשרונית. הפורט המלא הוא:

% <Width>. <Precision><Type>  
 כאשר Width מציין את סה"כ התווים הנדרשים (0 או ריק) אפשר לפיתון לקבוע את הגודל הדרוש  
 Precision מציין את מספר המיקומים לימין הנקודה העשרונית  
 Type מציין את סוג המידע (מספר שלים, שבר, מחרוזת וכו').

לכן, אם נשנה את הפוקודה הקודמת (להגבילת מספר המיקומים שלימין 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 ולבביעי אף ואכן ההדפסה התקבלה בהתאם לבקשה. ניתן כמובן גם לקבוע את גודל כל משתנה. 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 להוציא סימן לסכום. אולם, כאשר הסכום שלילי, הסימן קיים, אך לעיתים לצורך הדגשה, נדרש גם סימן עבור המספרים החשובים. 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 חיבור.

סוגי המידע הנפוצים הם: מספרים שלמים: `i`, `d`, שברים: `f`, מספרים אוקטליים: `o`, מספרים הקסהדצימליים: `x`, מספרים מדעיים: `e` ומחרוזות: `s`.

בגלל שהמרה והעיצוב מבוצעים באופן אוטומטי אפשר להשתמש בזה להמרת 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 (2, Python), אולם החל מהדורה 3 של פיתון הוכנסו פוקודות חדשות של עריכה ויש להעדיפן 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 שחלק ממלל ההדפסה נוספו, במקורה זה, שני פרמטרים אשר נקראים שדות הפורט וهم נמצאים בסוגרים מסולסים. המשמעות של שדות הפורט היא שהם צריכים להיות מוחלפים בערכים אשר מופיעים בהמשך הפוקודה (לאחר המלה פורט ובסוגרים). הלוגיקה היא שהשדה הראשון, אשר מספרו אפס, מוחלף בערך הראשון, השדה השני, אשר מספרו אחד, מוחלף בערך השני וכן הלאה. לכן, כתוצאה של ביצוע הפוקודה תתקבל ההדפסה של:

Shalom Avi and welcome to the CS class

ניתן כמובן להכניס את שדות הפורט שלא לפי הסדר, משום שהוא 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 להגיד שמות לשדות ואז המיקום אינם משנה. למשל בדוגמה:

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

במקורה זה, לא עושים שימוש במספר של הפורט, אלא ממש קובעים לו שם. למשל שם Paz מייצג את השדה של שם הסטודנט שברצוננו לברך. לעומת זאת, clas הוא השם של הקורס. השמות חיברים, כמובן להופיע בהמשך הפוקודה, כולל שארכו של הפורט name והוא Paz. ההדפסה שנוצרת מביצוע פקודה זו היא:

Shalom Paz, welcome to the MIS class!

אין מניעה מעירוב שתי השיטות.

בהמשך להגדלת ערכי הפורטים (שדות הפורט), אפשר גם לעצבם. בצורה פשוטה ביותר, אפשר להוסיף לשדה הפורט נקודותים ולאחריהם מספר המיציג את גודל השדה בהדפסה. למשל הפוקודה:

```
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 שהיא במקורה המקורי, אלא שעתה התווסף גם העיצוב. עבור השער הראשון, רשםנו שבהדפסה ברצוננו שהשדה יהיה בגודל של שמונה תוויים וכן למורთ שהשם כולל שלושה תוויים בלבד, הוא נכתב בשמונה תוויים, כאשר שאר החמשה הם רווחים. באופן דומה, רשםנו שהשדה השני צריך לכלול חמישה תוויים וכן בנוסח שלושה התווים בשם, בנוסף שני תווי רווח.

תווי העיצוב והעריכה עובדים כמובן גם על סוגים נוספים אחרים ולא רק על מחוזות. למשל אם ברצוננו להדפיס ערכים מסוימים:

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

בנהנזה שערך של a הוא 55, ההדפסה המתבקשת היא:

One item costs 55, three items cost 110

גם במקרה זה אפשר להוסיף פקודות עירכה, כמו למשל האדרה 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 האמייתי הוא של שניים או שלושה תווים, כפי שמופיע בדוגמה, יתווסף תוו' רווח.

One item costs 55, three items cost 110

בالمangelo של המספר המגדיר את גודל השדה, אפשר להוסיף תוו' נוסף המציין את הבסיס, כאשר ברירת המחדל היא בסיס עשרוני (התו d או h), בסיס בינארי (b), אוקטלי (o) והקסהדצימלי (X). כפי שניתן לראות בדוגמאות הרצל'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 לעורר את השדות בגודל של חמישה תווים בלבד, בגלל העובדה שהמספרים גדולים מחמשה תווים, גודל נקבע ע"י הערך בפועל ולא פקודת הפורמט. הדבר נעשה כדי למונע מצב שבו המספר נחתך והערך המוצג אינו הערך האמייתי. במצב זה, פיתון אינו מבצע את מה שביקשנו, אך הדבר נובע מהעובדה שיש כאן שתי בקשות מנוגדות. הצגת הערך האמייתי תוך התעלמות מהערכה שהຕבקשה מזיקה פחות מהציג ערך שגוי תוך הקפדה על הפורמט הדרוש.

חלק מהתמייקה בעריכת מספרים, קיימת גם אפשרות לעצב את השברים. הדבר נעשה ע"י הוספה של נקודה בפורמט ואחריה מספר המיצג את משלימות השבר ואחריו התו f המציין נקודה צפה (הדרך ליצג 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 למספריה המתמטית, כל הפונקציות והקבועים שבה, להיות זמינים. מתוך הספרייה אנו רוצים להשתמש בקבוע נק. בפקודה הראשונה אנו משתמשים בו, אך מבקשים לעורר אותו כך שיודפסו רק שבע ספרות מימין לנקודה העשוריית ואילו בדוגמה השנייה, אנו משתמשים בו פעם נוספת, אלא שהפעם בחרנו להשתמש בחמש עשרה ספרות.

אפשר כמובן לשלב בין שיטות המירה ועיצוב שונות:  
>>> print ("One item costs {0:5.3f}, three items cost {1:9b} ".format ( a, 2\*a ) )

## **תכנות בסיום.**

One item costs 55.000, three items cost 1101110

להרחבות נוספות לגבי יכולות העריכה והעיצוב ראה באתר.

## תרגילי חזרה

רשום את תוצאה הריצה. (בדוק את תשובה).

```

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

```

**פקודות else - ו break, continue**

פקודת ה – break נועדה לצאת מטווח הלולאה (או הלולאה הפנימית במקרה של לוולאות 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 הופעלת לאחר שהאיבר המתאים היה 9 ואז היא גורמת להפסקת ביצוע הלולאה  
ולכן כל שאר האיברים שרשימה a לא הודפסו.

פקודת ה – continue נועדה להפסיק את ביצוע המחזור הנוכחי של הלולאה ודילוג למחזור הבא.  
הפקודה משמשת עבור מקרים בהם במהלך הלולאה, עברו מחזורי מסוימים 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
```

לעתים אפשר להוציא לפוקודת לוולא גם פקודת `else` שמתבצעת כאשר הלולאה הסתיימה (במצב רגיל), אך לא מתבצעת כאשר הלולאה הסתיימה עקב פקודת `break`. הדבר מאפשר להוציא התוצאות מיוחדת למצב שהלולאה הסתיימה ולמשל תנאי מסוים לא קרה. לדוגמה, התוכנית הבודקת מספרים אם הם מחלקיים במספר כלשהו (מספר 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 הרצה:

```
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
```

## תרגילי חזרה

רשום את תוצאות הריצה. (בדוק את תשובה).

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!")
```

[תשובה](#)

[חזרה לתוכן](#)

## עבודה עם קבצים

לפעמים יש צורך לעבוד עם קבצים, למשל בקריאה נתונים רבים, או שמירה של תוצאות ריצה, ועוד. שתהינה זמינות לRICTOT אחודה/נוספות. העבודה מול קבצים נעשית תוך שימוש באובייקט (נושא שIORCHET בקורסי המשך), אשר מספק את כל הפקנציונליות הנדרשת. משמעו הדבר היא שיש לשירות את הקובץ מולו ברצונו לעבוד למשתנה וכל הפעולות 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 הכללת את שם הקובץ (כפי שהוא מופיע במחשב) ופרמטר שני מחרוזת המגדירה את מהות הפעולה מול הקובץ (`mode`). למשל אם ברצונו לפתח קובץ לקרוא בלבד נשתמש בערך "z" לעומת זאת אם רצча לכתוב לקובץ נבחר את הערך "w". כאשר פותחים קובץ לכתיבה והקובץ לא נמצא, פיתון יGENERATE קובץ חדש.

באופן כללי, פיתון תמיד יחפש את הקובץ בתיקייה בה אנו נמצאים. זו גם התיקייה שבה התוכנית שאנו חנו מבצעים, נשמרת. אם רוחים 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 "z". בפתיחת קובץ לכתיבה, פיתון ירמואס את תוכנו (יתחיל לכתוב עליו מתחילה). אם ברצונו לפתח קובץ לכתיבה 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 לקרואיה וגם לכתיבה, אז 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 BINARIM, כמו קבצי תמונות (JPEG, TIFF, BMP) ועוד. במקרה זה נדרש להוסיף את האות `b` לפרמטר השני למשל

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

הפקודה תפתח קובץ BINARI לקריאה וכתיבה. שימוש ב – "wb" לעומת זאת יפתח קובץ BINARI לכתיבה בלבד.

הערה:

בגל בעיות תאימות בין מערכות הפעלה שונות, תכניות שנכתבו בפייטון ואשר משתמשות בקבצים BINARIM, לא הכרח עובדות בצורה זהה על Windows ומערכות אחרות.

לאחר שנוצר הקישור בין הקובץ הפיזי (שנמצא על הדיסק) ובין המשתנה אשר מתאר אותו בפייתון, אפשר לקרוא ולכתוב לקובץ. לצורך הדוגמאות שמויפות בהמשך פרק זה, נניח שהמשתנה שבחרנו הוא `my_file` (כמו בדוגמאות לעיל). בנוסף, לצורך ההסבירים, 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 להשמיט את הfrmטר השני בפקודת הקריאה.

קיימות שלוש שיטות (פקודות) לקריאה מהקובץ:

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

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

בעקבות הקריאה, המשתנה `a` יוכל מחזורת 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'
```

ניתן לראות את תוכן הקובץ, כאשר בסוף כל שורה מופיע "ח" שמצוין סוף שורה. אם צריך לעבד את תוכן הקובץ, אפשר להשתמש בפקודות הרגילות לחילוץ ועיבוד מחזורות.

אפשר גם לקרוא חלק מהתווים בקובץ, ע"י הוספה frmטר 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 נדרש לבצע פקודת קריאה נוספת כמו:

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

חשוב לציין שפיטון זוכר את המיקום בקובץ ולכן בפקודות הבאות הוא ממשיר לקרוא ממיקום זה ולא מההתחלת. אך אם רוצים לחזור לקרוא שוב מההתחלת, יש צורך לשנות את המיקום והדבר מתבצע ע"י פקודות `seek`. אם ברצוננו לחזור אל התחלת הפקודה שיש לבצע היא

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

באופן דומה אם ברצוננו לקרוא ממיקום אחר בקובץ, ניתן להשתמש בפקודת `seek` כדי לשנות את המיקום לתחילת הקיראה (או הכתיבה) הבאה.

2 – אשר קוראת שורה אחת מהקובץ ומחזירה מחוזצת. המחרוזת כוללת גם את תחילת השורה ("ח"). אם בקובץ נמצאת שורה ריקה, `readline` יחזיר מחוזצת המכילה "ח" ואילו בסוף הקובץ הוא יחזיר מחוזצת ריקה. דוגמאות לשימוש ב-`readline`:

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

כמו במקורה קודם, ניתן לשמור את המיקום בקובץ ולכון ביצוע 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 תוכן של הקובץ נצטרך להשתמש בלולאה שבכל מחזור תקרא שורה. זו יכולה להיות למשל לולאת `while` אשר תבצע כל זמן שהמחוזצת שהתקבלה בקריאה ה-`readline` אינה ריקה. דרך אחרת לקרוא 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 – אשר קוראת את כל הקובץ ומחזירה רשימה של מחוזצות. כל מחוזצת היא שורה בקובץ. הפקודה מקבלת פרמטר אופציונילי אשר מגדיר את מספר התווים שיש לקרוא מהקובץ. אולם במקרה לפקודה `read`, אשר קוראת לבדוק את מספר התווים שהוגדר בפרמטר, `readlines` קוראת שורות שלמות וכך היא תשלים את התווים מעבר למוגדר בפרמטר, עד להכללת השורה השלמה الأخيرة. 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 הקובץ את מספר התווים היא כדי לא לקרוא את כל תוכנו של הקובץ לזכרו, כדי לא להעמס יתר על המידה.

כתביה לקובץ משתמש בפקודה `write`, אשר כתבתה 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 היל, פיתון יצר אותו. לאחר מכן הגדרנו מחרוזת (בתוכה המשתנה `some_text`) ואז השתמשנו בפקודה `write`, כדי לכתוב את המחרוזת אל הקובץ. הפקודה `write` יכולה גם לכתוב לקובץ את הטקסט שנמצא בסוגרים הצמודים אליה, כפי שניתן לראות בדוגמה. בכל מקרה, לאחר שהtekst נכתב, הפקודה `write` מוחזרה את מספר התווים שנכתבו.

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

```
my_file.close()
```

הפקודה מחדירה את הקובץ למערכת הפעלה, כך שתהליכיים אחרים, או תוכניות אחרות יכולות להשתמש בו. לאחר סגירת הקובץ, אין יותר אפשרות לקרוא ממנו או לכתוב אליו תוך שימוש בשם הקישור ויש לפתח אותו שוב לפני שניתן יהיה לגשת אליו.

[חזר לתוכן](#)

## הבנת תוכניות

פרק זה מראה מעין סיכום והוא כולל דוגמאות של תוכניות שעילר להבין ולענות על השאלה.

1. התוכנית אומרת לקרוא מספר ולהדפיס "even" אם הוא זוגי – "odd" אם הוא אי-זוגי. לצורך התרגיל, הנה שהמספר 0 הינה זוגית.

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

- א. התוכנית עבדת נכון
- ב. התוכנית עבדת נכון רק למספרים זוגיים
- ג. התוכנית עבדת נכון רק למספרים אי-זוגיים
- ד. התוכנית לא עבדת נכון
- ה. התוכנית עבדת 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 בPython

5. מה הערך שיתקבל מחרצת 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 מחרצת 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 המתבקשת 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 המתkeletal אם קוראים לפונקציה ע"י ( 3 , 4 ) ?

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 המתקיים 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)
- התוכנית מדפסה 10 שורות
  - השורה הראשונה המודפסת היא: 60,60
  - השורה האخונה המודפסת היא: a,10

11. רשום מה מדפס קטע הקוד בסימן 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 המתקיים אם קוראים לפונקציה ע"י  $? \text{sum2n} ( 7 )$

## תכנות בסיסטיות

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 ? 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 ? 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 ? fun ( 5 , 79 )

16. רשום מה מדפס 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 מדפסו 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 מדפסו 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 מדפסו 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 מדפסו 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 מדפסו 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 מדפסו 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 בPython

23. תוך התichosot לkatu haKod haRatz'v, ma tahia tzoat biTzou ([4]).

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

[Tshuvot](#)

[Chzur liTocn](#)

## 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 שلد של תוכנית המקבלת שתי רשימות של מספרים ויצרת רשימה שלישית הכוללת מספרים המורכבים מהמספרים שבשתי הרשימות. האיברים הנמצאים במקומות הזוגיים מחושבים סכום שני האיברים המתאימים ואילו האיברים במיקומים הא-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 \*\*\*\* 3 \*\*\*\*\*, \*\*\*\* 2 \*\*\*\*\*, \*\*\*\* 1 \*\*\*\* 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 בPython

רוצ"ב שלד של תוכנית המקבלת רשימה של שמונה מספרים ומיצרת רשימה אחרת, כאשר כל מספר ברשימה החדשה חושב על בסיס המספר ברשימה המקורי. את המספר הראשון מכפילים פי שמונה, את השני פי שבעה וכן הלאה. את המספר הלפניו אחרון מכפילים פי שניים ואת המספר האחרון משאים כפי שהוא.

```
x = [11, 21, 34, 35, 37, 19, 07, 16]
y = [0, 0, 0, 0, 0, 0, 0, 0]
***** 1 *****
***** 2 *****
***** 3 *****
```

את הפעולות המוגדרות ע"י \*\*\*\* 3 \*\*\*\* , \*\*\*\* 2 \*\*\*\* , \*\*\*\* 1 \*\*\*\* יש להחליף ע"י הפעולות:  
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 בPython

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 \*\*\*\* 3 \*\*\*\*, \*\*\*\* 2 \*\*\*\*, \*\*\*\* 1 \*\*\*\* 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 רשימה מספרים ומדפיס את ממוצע המספרים וכן כמה מספרים בראשימה 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 \*\*\*\* 3 \*\*\*\*, \*\*\*\* 2 \*\*\*\*, \*\*\*\* 1 \*\*\*\* יש להחליף ע"י הפקודות:

.א.  
avg = x [ k ]  
if x [ k ] < avg :  
if x[ k ] > avg :

.ב.  
avg = avg + x [ k ]  
if x [ k ] < avg :  
if x[ k ] > avg :

.ג.  
avg = avg + x [ k ]  
if x [ k ] > avg :  
if x[ k ] < avg :

.ד.  
avg = avg + x [ k ]  
if x [ k ] > avg :  
if avg < x[ k ] :

## תכנות בPython

5. רצ"ב שלד של תוכנית המקבלת רשימה מספרים ומחשבת את הממוצע שלהם. לאחר מכן מחשבת את סכום השאריות המתפלות 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 \*\*\*\* 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 בPython

6. רצ"ב שלד של תוכנית המקבלת רשימה 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 \*\*\*\* 3 \*\*\*\*, \*\*\*\* 2 \*\*\*\*, \*\*\*\* 1 \*\*\*\* 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 בPython

7. רצ"ב שلد של תוכנית המקבלת רשימת מספרים ומדפיסה עבורה 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 \*\*\*\* 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 :
```

8. רצ"ב שילד 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 \*\*\*\* 3 \*\*\*\*, \*\*\*\* 2 \*\*\*\*, \*\*\*\* 1 \*\*\*\* 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 גדול באחד כל פעם. התוכנית מבקשת 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 \*\*\*\* 3 \*\*\*\* ,\*\*\*\* 2 \*\*\*\* ,\*\*\*\* 1 \*\*\*\* יש להחליפ 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="")
.

[תשובה](#)

[חזר לתוכן](#)

כטיבת תוכנית

בפרק זה רשימה של אפיונים לתוכנית. עליך לכתוב את התוכנית כך שתבצע את שנדרש. חשוב מאוד  
לנסות לפתח את התרגילים בלבד לפני שפונים לתשובה הקוללות הסבירים מפורטים.

- כתוב תוכנית אשר קוראת שני מספרים המיצגים צלעות של מרובע. עליך לחשב ולהדפיס את היקף המרובע ואת שטחו.

כתוב תוכנית אשר קוראת שבעה מספרים, מחשבת את סכום ואת הממוצע שלהם.

הרחיב את התוכנית הקודמת, כך שגם תדפיס את המספרים שקרה (כולם באותה שורה ומופרדים ברוחן) וכן לאחר מכן תחשב את הסכום והממוצע.

כתוב תוכנית הקולעת שני מספרים ומחשבת את סכום המספרים שביניהם (לא כולל המספרים עצמם). אפשר להניח שהמספר הראשון הוא הקטן שבין השניים.

כתוב תוכנית אשר מחלקת מספרים. התוכנית קולעת שני מספרים שלמים ומחלקת את השני הראשון. היא מדפיסה את תוצאה החלוקה וכן את השארית שהתקבלה. למשל בחלוקת 9 ל – 4 התוצאה תהיה 2 והשארית 1.

כתוב תוכנית אשר קוראת שני מספרים המיצגים אורך הנקודות של משולש ישר זווית ומחשבת את אורךה של הצלע השלישית (היתר).

כתוב תוכנית המקבלת שני קלטים. מספר ותו. התוכנית תדפיס ריבוע בגודל המספר והבני מהנתונים. למשל עבור 3 ו – \*\*"\*\* התוכנית תדפיס:

\*\*\*

\* \* \*

\*\*\*

8. כתוב תוכנית המקבלת מספר קלט ומחשבת את כל המחלקים השלמים שלו. למשל עבור 15 המחלקים הם 3 – 5.

9. פלינדרום היא מלה סימטרית (אפשר לקרוא אותה מימין ומשמאלי). כתוב תוכנית אשר קוראת מחזרות ומחריזה למשתמש אם היא פלינדרום. למשל המלה abba היא פלינדרום, או הערך (מחזרות) 12321 גם הוא פלינדרום ואילו 123421, אינו פלינדרום.

10. כתוב תוכנית אשר קוראת מספר המכיל כמה ספרות ומדפיסה אותן בסדר הפוך. למשל אם הקלט הוא 12345, אז התשובה צריכה להיות 54321

11. כתוב תוכנית האזינה לשורת הקולטות מחזרות, באורך כלשהו, של אותיות ורווחים בלבד. התוכנית מצפינה כל תו שבמחזרות ע"י הוספה 2 לערכו המספרי (הערך של התו ב – ASCII). כמו כן התו "a" יהפוך ל – "c", התו "b" יהפוך ל – "d" וכן הלאה. שני התווים האחרונים מתנהגים קצת אחרת. התו "u" יהפוך ל – "a" ואילו התו "z" יהפוך ל – "b". אפשר להניח שהטקסט שיעוכנס הוא תמיד אותיות ורק קטעות כלומר abc... ולא ...ABC. רווחים שנמצאים במחזרות נשארים ללא שינוי.

12. חלק מכרטיסי האשראי של ויזה ממשים אלגוריתם (ע"ש Cheshon) לחישוב חוקיות מספר הcrc32. על פי אלגוריתם זה, המספר מורכב משערה ספרות, כאשר סכום הביקורת מחושב ע"י חיבור כל הספרות אשר נמצאות במיקומים אי זוגיים (ספרה ראשונה, שלישית וכו') וחיבור "מיוחד" של הספרות במיקומים הזוגיים, לאחר שכל ספרה הוכפלה פי שניים. אם המספר שהתקבל לאחר הכפלת הזוגיים מ – 10 מחברים אותו אל סכום הביקורת. לעומת זאת, אם המספר שהתקבל לאחר הכפלת הזוגיים מ – 10 נחלק ב-16 נקבל ספרות אחת לשניה (אחת לשנייה) ואת הסכום מוסיפים לסכום הביקורת. אם סכום הביקורת שהתקבל, מחלוקת כל במספרם, הוא כפולה של 10, אז המספר חוקן. כתוב תוכנית הקוראת מספר בעל 16 ספרות ובודקת אם הוא מספר CRC32 אשראי חוקן. אפשר להניח שהקלט כולל ספרות בלבד ואורכו 16 ספרות. לדוגמה המספר 4580123456789000 אמרו להיות חוקן. החוקיות מחושבת ע"י חיבור הספרות 4,8,1,3,5 ועוד הלאה והכפלה (פי שניים) של הספרות במיקומים הזוגיים וחיבורם:  $4 \cdot 2^0 + 8 \cdot 2^1 + 1 \cdot 2^2 + 3 \cdot 2^3 + 5 \cdot 2^4$  וכן הלאה. כאמור במקורה של הכפלה שהיא גודלה מ – 10 יש לחבר את הספרות.

|   |    |   |   |   |   |   |   |   |    |   |    |   |   |   |   |       |
|---|----|---|---|---|---|---|---|---|----|---|----|---|---|---|---|-------|
| 4 | 5  | 8 | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 8  | 9 | 0 | 0 | 0 | ספירה |
| 1 | 2  | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2  | 1 | 2  | 1 | 2 | 1 | 2 | מכפיל |
| 4 | 10 | 8 | 0 | 1 | 4 | 3 | 8 | 5 | 12 | 7 | 16 | 9 | 0 | 0 | 0 | תוצאה |
| 4 | 1  | 8 | 0 | 1 | 4 | 3 | 8 | 5 | 3  | 7 | 7  | 9 | 0 | 0 | 0 | >10   |

13. כתוב תוכנית הקוראת שלושה מספרים c, b, a בעלי אותו מספר ספרות. כל מספר מייצג ווקטור של מספרים חד ספרתיים. בנה רשימה של ערכי d שהינה תוצאה c-b-a. אין צורך לבדוק שכאן הוקטורים מכילים אותו מספר ספרות.

14. כתוב תוכנית הקוראת שתי רשימות של מספרים. בשתי הרשימות מספר זהה של מספרים. ובנוסף הערך 0,1,2,3. התוכנית אמורה ליצור רשימה חדשה שבה כל אחד מהאיברים תלוי באיבר המתאים בראשימת הבקרה:

- אם האיבר בראשימת הבקרה הוא 0, אז האיבר הרשימה החדשה הוא 0.
- אם האיבר בראשימת הבקרה הוא 1, אז האיבר בראשימת החדשה הוא האיבר המתאים מתוך הרשימה הראשונה.
- אם האיבר המתאים בראשימת הבקרה הוא 2, אז האיבר בראשימת החדשה הוא האיבר המתאים מתוך הרשימה השנייה.
- ואם האיבר המתאים בראשימת הבקרה הוא 3, אז האיבר הרשימה החדשה הוא סכום שני האיברים המתאים בשתי הרשימות.

15. כתוב תוכנית המדפיסה את לוח הכלfel. התוכנית מקבלת כקלט מספר המייצג את גודל הלוח ומדפיסה טבלה ישרה בה לכל איבר מוקצים חמישה תוויות.

16. כתוב תוכנית הקולטת שני מספרים, כאשר אחד מייצג מספר כלשהו ואילו השני מייצג את הבסיס של המספר הראשון. התוכנית אמורה לחשב את ערכו העשוני של המספר שנקלט. אין צורך לבדוק את תקינות המספרים.

17. מין בועות (Bubble Sort) הוא אלגוריתם פשוט של מיון. במסגרת האלגוריתם עוברים על כל איברי הסדרה ומשווים כל זוג של איברים, כאשר מתחילה מתחילה הסדרה וממשיכים עד לסופה. מחרורי ההשוואה חוזרים על עצם עד אשר כל הסדרה ממינית. האלגוריתם ממומש בעזרת שתי לולאות מקוונות, כאשר הרשונה מתבצעת על כל איברי הסדרה ואילו השניה מתחילה באיבר השני וממשיכה במחזור ראשון עד לאיבר האחרון, במחזור שני עד לאיבר לפניו אחרון וכן הלאה. מחרורי הלולאה השנייה (המקוונת) מתקזרים באיבר אחד בכל פעם משום שבשיטה זו האיבר האחרון בכל מחזור כבר נמצא במקומו. כתוב תוכנית למימוש מין בועות. התוכנית תקרא רשימה של מספרים ותדפיס אותם לאחר שימושו.

18. עיגל חסום ביריבוע, כך שצלעות הריבוע משיקות לעיגול. כתוב תוכנית המחשבת את שטחו של הריבוע שאינו מכוסה ע"י העיגול (החלק הכהול שבתרשים). התוכנית אמורה להדפיס סדרה המייצגת את שטח ריבועים שאינם מכוסים ע"י עיגולים עבורי סדרת הריבועים שאורך הצלעות שלהם נתון ע"י: 9...1,2,3,4. קלומר המספר הראשון שיודפס הוא שטחו של ריבוע שאורך צלעו הוא 1, המספר השני עבורי ריבוע שאורך צלעו 2 וכן הלאה. הסדרה המודפסת צריכה הינה בדיקות של שלוש ספרות מימין לנקודה העשונית.

19. כתוב תוכנית הקוראת רשימה של מספרים שלמים המייצגים רדיוסים של עיגולים. אפשר להנify שהמספרים ממוינים ומוכנסים כאשר המספר הגדול הוא הראשון. העיגולים נמצאים האחד בתוך השני, כך שהם יוצרים סדרה של טבעות (לדוגמה הטבעת הכהולה והטבעת הצהובה. התוכנית אמורה לחשב את שטחה של כל טבעת ואת השטח הכלול של כל הטבעות גם יחד. את שטח הטבעת יש לבטא כמספר בדיקות של שלוש ספרות מימין לנקודה העשונית ואילו את השטח הכלול של כל הטבעות יש לבטא כמספר בדיקות של שתי ספרות מימין לנקודה העשונית. שם לבי שבニアג'וד לתרשים, הכלול שלושה מספרים בלבד, בפועל מספר המספרים המופיעים בקלט יכול להיות שונה.

20. כתוב תוכנית הקולטת שני מספרים שלמים (b, a). התוכנית אמורה לחשב את החזקה המרבית (a) של המספר הראשון המקיימת  $b^a$ . ההדפסה צריכה לכלול את החזקה ובסוגרים (לא רווחים) את המספר בחזקה המתאימה והຕואאה שהתקבלה. אפשר להנify 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 ח-י משוכפל. למשל אם המחרוזת הראשונית היא abcdefg והמספר שהתקבל הוא 2, אז המחרוזת החדשה תהיה : abbcdddeffg.

22. מספרי (Binary Coded Decimal) הם מספרים בינאריים בגודל של ארבע סיביות ואשר כל אחד מייצג ספרה עשונית אחת. קלומר ערך כל ספרה הוא 0-9. כתוב תוכנית שקוראת רשימה של מספרי BCD ומחשבת את ערכה העשוני של הרשימה כולה. למשל אם הרשימה שנקרה

## תכונות בסיסי

- כוללת: 1001,1000,0111,1001 אדי ערכה העשורי הוא 9879. אפשר להניח שהמספרים הבינריים הם באמת מספרי BCD ואין צורך לבדוק זאת.
23. כתוב תוכנית אשר קוראת רשימה של מספרים ומיצרת רשימה חדשה שבה כל איבר הוא סכום האיברים שמשני צדדי. התיכון לרשימה כאלה הייתה מעגלית, כלומר האיבר הראשון ברשימה החדש יהיה סכום האיבר האחרון והראשון ברשימה המקורית.
24. כתוב תוכנית הקוראת רשימה של מספרים ומיצרת רשימה חדשה הכוללת רק מספרים ייחודיים (שלא חוזרים על עצמם). במקרים אחדות, איברים כפולים שהיו ברשימה המקורית נמצאים רק פעם אחת ברשימה החדשה.
25. כתוב תוכנית הקוראת רשימה של מספרים ומיצרת רשימה חדשה שבה האיברים הם סכומי ביניים. כל איבר הוא סכום האיברים שלפנויו (האיבר הראשון זהה, האיבר השני הוא סכום שני האיברים הראשונים וכן הלאה)
26. כתוב תוכנית הקוראת מספר (ח) בתחום שבין שלוש לתשע ומדפיסת את כל המספרים בעלי שלוש ספרות בסיסי ח. אם המספר שהוקלד שגוי התוכנית תחזיר ותבקש מספר חדש שעונה לדרישה.

[תשובות](#)

[חזרה לתוך](#)

## 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 ושווא שוקיים. הפונקציה "מצירף" את המשולש תוך שימושו ב\*\*". למשל אם הפרמטר שהוכנס הוא 4, אז תוצאה הקריאה לפונקציה תהיה:

\*  
\*\*  
\*\*\*  
\*\*\*\*

3. כתוב פונקציה המקבלת שני פרמטרים (מערך ותו). הפונקציה מוחזירה מערך המבוסס על המערך המקורי לאחר של המופעים של התו (שהועבר כפרמטר שני) בוטלו.
4. כתוב פונקציה המקבלת רשימה של איברים מספריים המייצגים ציוניים. הפונקציה אמורה להחזיר שלוש תוצאות (ממוצע הציוניים, מספר הציוניים הגבוהים או שווים לממוצע ומספר הציוניים הנמוכים מהממוצע)
5. כתוב פונקציה המקבלת מערך של תווים ומחזירה מערך אחר המבוסס על המערך המקורי לאחר שכל הרוחחים שבו הוסרו. למשל אם הפונקציה מקבל מערך: "Shalom All, How Are You??" הפונקציה יחזיר "ShalomAll,HowAreYou??"
6. מספרי ארמסטרונג (Armstrong numbers) הם מספרים מייחדים המקיימים את התנאי שערך המספר שווה לסכום הספרות שלו בחזקת מספר הספרות. למשל 153 הוא מספר המקיים את התנאי ש $1^3 + 5^3 + 3^3 = 153$ . כתוב פונקציה המקבלת פרמטר מספרי אחד (ח) ומדפיסה את כל מספרי ארמסטרונג בתחום שבין 1 – ח
7. כתוב פונקציה המקבלת שלושה מספרים c, b, a (אשר מייצגים את אורכי צלעות המשולש) ומחשבת את שטחו. השתמש בנוסחת הרט ליפה  $s = \sqrt{a+b+c}$ , כאשר  $s = \frac{1}{2}(a+b+c)$
8. במספר מדיניות נהוגה שיטה שונה של ציוניים. (A עבור הציוניים 100-90, B עבור 89-80, C עבור 79-70, D עבור 69-60, F עבור ציוניים נמוכים מ – 60). כתוב פונקציה המקבלת ציון "רגיל" ומוחזירה את האות המתאימה לציון.
9. כתוב פונקציה המחשבת את סכום הכספי שנמצא בשקייה. הפונקציה מקבלת ששה פרמטרים המיצגים:  
 a. את מספר מטבעות ה – 10 אגורות שבשקית,  
 b. את מספר מטבעות החצי שקל שבשקית  
 c. את מספר מטבעות השני שקלים שבשקית  
 d. את מספר מטבעות השלישי שקלים שבשקית  
 e. את מספר מטבעות החמישי שקלים שבשקית  
 f. את מספר מטבעות העשירי שקליםים שבשקית
10. בית קולנוע גובה 40 שקלים מכל לקוח. הקרנת סרט עולה 300 שקלים ועוד תלוגים בגובה של 5 שקלים לצופה (לקוח). כתוב פונקציה המקבלת פרמטר מספרי (מספר הלהקות בהציג מסויימת) ומחשבת את ההפסד/רווח של הקולנוע.
11. כתוב פונקציה המחשבת את מס ההכנסה לשכיר בשנת 2009. הפונקציה מקבלת פרמטר אחד (השכר החודשי) ומוחזירה 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 מוחזירה רשימה הכוללת רק את האיברים ברשימה המקורי, אשר גדולים מהמספר (הפרמטר השני)
13. כתוב פונקציה המקבלת רשימה של ערכים מספריים ומוחזירה את המספר הקטן ביותר ברשימה, המספר הגדל ביותר ברשימה והממוצע שלה.
14. כתוב פונקציה המקבלת שני פרמטרים: מערך בגודל כלשהו ומספר (ח). הפונקציה אמורה להחזיר מערך הכלול רק את האיברים ה – ח ים במערך המקורי. למשל אם המערך המקורי הוא: "cfilegijklmnop" והמספר שהועבר הוא 3, אז התוצאה תהיה: "cfilo"

## תכונות בסיסי

15. כתוב פונקציה המקבלת ערך מספרי (ח) ומחשבת את סכום ריבועי המספרים בתחום: ח-1 (כולל ח).
16. כתוב פונקציה המקבלת מחרוזת של תווים מסוימים 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 שלו (ובלבד שהמחלק גדול אחד)
18. כתוב פונקציה המקבלת פרמטר שהוא מספר חיובי ומחזירה את הספירה השמאלית שלו.
19. כתוב פונקציה אשר מקבלת רשימה של מספרים. הפונקציה מסכמת את כל המספרים הזוגיים ואת כל המספרים הא-זוגיים ומחזירה "ODD" אם סכום המספרים הא-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. הנה שהמחרוזות באורך שווה. הפונקציה אמורה ליצר מחרוזת חדשה שהיא תוצאה הביצוע של הפעולה הבוליאנית XOR (Exclusive OR). פונקציה זו מקבלת שני פרמטרים ביןaries 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 הגדל ביותר. ככלمر במקומות שבו היה האיבר הקטן, ימצא האיבר הגדל ולהיפך.  
למשל עבור הקלט [3,4,11,7,2,5,4,3], הפונקציה מחזירה [3,4,2,7,11,5,4,3]

[תשובה](#)

[חזרה לתוכן](#)

## רקעוטיה

רקעוטיה היא דרך מקובלת לפיתרון בעיות (לא רק בפתרונות תוכנה), אשר עושה שימוש במנגנון של "הקטנת" הבעיה שיש לפטור. הרעיון הבסיסי הוא הפשטה הבעיה ממחזר עד אשר נגיעה לבעה 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 מחזר אנחנו מתקדים לkrarat הפיתרון המלא. הדוגמה הנפוצה ביותר היא הגדרת הפונקציה עצרת. כדי להסביר למשל את !<sup>5</sup>, לעינו לכפול את כל האיברים בתחום שבין 1 ל – 5. אפשר כמובן לבצע זאת ע"י לולאה, אולם רקווטיה (לאחר שהובנה) מספקת פיתרון קל ופשוט יותר. נניח שעליינו לפתח פונקציה רקווטית אשר אמורה לחשב את !<sup>ch</sup>, מtower הפרמטר ch שמעובר להה. כדי לעשות זאת עלינו להבין שהכל המאפשר לחשב את !<sup>ch</sup>, משמעו – ch)!...!(ch)(-ch)=1. במקרים אחרים חישוב ch משמעו – ch (לאחר שהובנה) מספקת פונקציה מופרדת מפונקציות אחרות ואילו פונקציה ממופע אחר שלא עצמה. לכן כדי להגדיר פונקציה רקווטית, שי להגדיר:

- א. את תנאי היציאה. למשל במקורה של עצרת התנאי יהיה עבור 1=ch, הפונקציה תחזיר 1
- ב. את התנהגות הפונקציה במקירים האחרים. בדוגמא העוצרת עבר כל ch אחר (שונה מאחד), הפונקציה תחזיר את !(1-ch). במקרים אחרים את !(1-ch) מוכפל בקריאה לפונקציה עם הערך (1-ch).

## תרגילי חזרה

1. כתוב פונקציה המקבלת שני פרמטרים מספריים (b, a), כאשר הראשון מייצג מספר מסוים ואילו השני מייצג את החזקה. הפונקציה מחשבת <sup>b</sup>a. למרות שבפתרון קיימת פונקציה כזו, יש לפתח אותה ללא קשר ליזה הקיימת. את הפונקציה יש לפתח פערם. פעם ככלאה ופעם כרקווטיה. אפשר להניח שהמספרים תקיים ואין צורך לבדוקם.
2. כתוב פונקציה רקווטיבית 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 וכל איבר מעבר לזה הוא סכום שני האיברים שלפני. כתוב פונקציה רקווטיבית אשר מקבלת פרמטר מספרי (ch) ומחשבת את ערכו של המספר ה – ch – בסדרה.
4. כתוב פונקציה רקווטיבית אשר מקבלת פרמטר שהוא מחרוזת ומחזירה אותה בסדר הפו.
5. כתוב פונקציה רקווטיבית המקבלת פרמטר מספרי (ch) ומחשבת את סכום כל הספרות אחד ועד ch.
6. נתונה סדרה המקיימת  $5 = 1 + a_{n-1} + a_n$ . כתוב פונקציה רקווטיבית המקבלת פרמטר מספרי (ch) ומחזירה את האיבר ה – ch – בסדרה.
7. כתוב פונקציה רקווטיבית המקבלת פרמטר שהוא מספר שערוני וממירה אותו למספר בינארי.
8. כתוב פונקציה רקווטיבית המקבלת רשימה של מספרים ומחזירה את סכומם.
9. כתוב פונקציה רקווטיבית המקבלת פקודות כפל. כתוב פונקציה רקווטיבית המקבלת שני מספרים טבעיות (חיוביים) ומחשבת את המכפלת שלהם.
10. במחשב מסוים שכחו מהנדסי החומרה למשוך פקודות כפל. כתוב פונקציה רקווטיבית המקבלת שני מספרים טבעיות (חיוביים) ומחשבת את המכפלת שלהם.
11. כתוב פונקציה רקווטיבית המקבלת מספר טبعי (ch) ומחזירה את סכום איברי הסדרה הנתונה ע"י:  $1^2, 2^2, 3^2, \dots, n^2$ .
12. כתוב פונקציה רקווטיבית המקבלת פרמטר אחד (ch) שהינו מספר טעני חיובי. ומחשבת את הערך  $ch^2$ , מבלי להשתמש בפוקודות כפל, אלא בידע ש  $-1^2 = 1$ .
13. מספרי פרין (Perrin) הם מספרים המיצגים סדרה של מספרים אשר מקיימת את התנאים:  $(-ch+P)(-P+2) = P(1-P) = 0$ ,  $P(2-P) = 2$ ,  $P(3-P) = 3$ . ממייצג מספר שאיןו 0, 1, 2. כתוב פונקציה רקווטיבית אשר מקבלת פרמטר מספר פרין המתאים.
14. כתוב פונקציה רקווטיבית אשר מקבלת מספר אמיתי (ch) ומדפיסה שלושת של כוכבים (\*). כאשר מספר הכוכבים בשורה הראשונה הוא ch ובכל שורה המספר קטן באחד.
15. כיצד משתנה תשובה אם ברצוננו להדפיס משולש נוסף (הפור) לאחר המשולש הראשון. כלומר להדפיס את המשולש הראשון ואחריו משולש נוסף, המתחל בכוכב אחד בשורה הראשונה ובכל שורה נוספת כוכב עד לשורה בעלת ch כוכבים

16. כתוב פונקציה רקורסיבית המקבלת פרמטר (ח) שהוא מספר טבעי שלם וחובי ומחזירה את סכום החאיברים של הסדרה הרמוניית. סדרה הרמוניית היא הסדרה:  $\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}$
17. מספרי לוקאוס (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 לוקאוס ניתן להגדיר אותם ولكن במספרי פיבונצ'י הם מקרה פרטי של מספרי לוקאוס. כתוב פונקציה אשר מקבלת שלושה מספרים טבעיות  $a, b, c$ . המספר הראשון מיצג את האיבר הראשון בסדרה, המספר השני מיצג את המספר השני בסדרה והמספר השלישי מיצג את האיבר שהפונקציה צריכה לחשב. שם לב שאיברי הפונקציה מתחלים מאפס.
18. כתוב פונקציה רקורסיבית המקבלת פרמטר שהוא מספר טבעי חיובי ומוחזירה את מספר האפסים שבו
19. כתוב פונקציה רקורסיבית אשר מקבלת שני מספרים טבעיות ( $m, n$ ), ומוחזירה 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 ( $27=3^3$ ).
20. כתוב פונקציה רקורסיבית אשר מקבלת מערך בגודל כלשהו ומוחזירה את גודלו (בלי לשימוש בפונקציה `len`).

[תשובות](#)

[חזרה לתוכן](#)

### תרגילי תוכנות ללא פתרון

1. כתוב תוכנית הקולטת מהמשתמש את אורך צלע היריבוע ומחשבת את שטחו ואת היקפו. לצורך התרגיל הנה שארוך הצלע מボוטא במספרים שלמים וחיברים

לדוגמה, בהכנסת הערך 6, תתקבל הדפסה כמו:

The area of a square with side length of 6 is 36  
and the circumference is 24

2. שנה את התרגיל הקודם, כך שהתוכנית תקבל גם אורךים שהם מספרים ממשיים (শברים).

לדוגמה, בהכנסת הערך 6, תתקבל הדפסה כמו:

The area of a square with side length of 7.5 is 56.25  
and the circumference is 30.0

3. כתוב תוכנית הקולטת שני מספרים שלמים המייצגים תחום ומחשבת את סך כל המספרים שבתחום (כולל שני מספרי הקולט). בשלב זה, אפשר להניח שהמספר הראשון תמיד יהיה הקטן שבין השניים וכן שלא מדובר בשני מספרים זרים.

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  
ורם:

Please enter the value of x: 4.5  
For x = 4.5, the powers are: 20.25 91.125 410.0625

7. שלשה פיתגורית היא שלושה מספרים  $a, b, c$  המקיימים  $c^2 = a^2 + b^2$ . כתוב תוכנית הקולעת 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 הקולעת מספר  $\chi$ , גדול מ – 5 ומחשבת את סכום המספרים הזוגיים שבתוחם  $[0, \chi]$ . וכן את סכום המספרים האי זוגיים שבתוחם. שים לב שהתחום כולל גם את  $\chi$ . אין צורך לבדוק את תקינות המספר  $\chi$ . 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 הקולעת מספר  $x$ , המיצג 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 זוק ומוכפל ב – ארבעה שלישים  $3^3 = 4\pi r^3$ . עבר החישוב השתמש בערך של  $\pi$  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 ל"יצג גובה ע"י שילוב של רגליים (feet) ואינץ' (inch). אורך של אינץ' הוא 2.54 ס"מ ואילו רגל היא 12 אינץ'. כתוב תוכנית הקולעת שני מספרים (רגל ואינץ') אשר יחד מייצגים גובה. התוכנית תמיר את הגובה מהсистемה 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 ממירת מספרים בעלי שלוש ספרות מבסיס Celsius לבסיס עשרוני. התוכנית תקלוט שלוש ספרות  $x_1, x_2, x_3$ , המיצגות 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 בPython

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  
שבע 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 השלה 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 (כתב תוכנית אשר קולטת מספר בן שני ספרות. התוכנית תדפיס Bingo, אם המספר מתחילה数字 7 או שהוא מכיל את הספרה 7. בכל מצב אחר, התוכנית תדפיס Better luck next time), כך שבמקרה 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 שוויים 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 בPython

19. כתוב תוכנית אשר קולטת שני מספרים a, b ומדפיסה את כל המספרים מאחד ועד a (כולל), בתנאי שהם לא מחלקים b - a. אין צורך לבדוק את תקינות המספרים.

דוגמת הרצה:

```
Please enter first number: 10
Please enter second number: 3
1
2
3
4
5
6
7
8
9
10
```

20. כתוב תוכנית אשר קולטת שני מספרים a, b ומסכמת את כל המספרים בתחום אחד ועד a (כולל), אשר מחלקים b - a. 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 למיניות ברחבי העולם. עלות המשלוח משתנה בהתאם למשקל החבילה. לכל חבילה עלות של \$3 דמי טיפול. מחיר זה כולל משלה חינם לחבילה שימושה לא יותר מק"ג אחד. עבור חבילה שימושה ק"ג יש להוציא 2.5\$ לפחות 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) בראש הטלפונייה הקווית, יצאא בזק במבצע מיוחד האזמין רק ללקוחות הרשומים לתוכנית. עלות התוכנית היא 15 ש' בחודש ובמסגרתה עלויות משולוח המסרונים הן כמפורט בטבלה:

| מחיר                | מספר מסרוונים בחודש |
|---------------------|---------------------|
| חינם (כלול בתוכנית) | 1-500               |
| 10 אג' למסרון       | 501-1000            |
| 8 אג' למסרון        | 1001-5000           |
| 5 אג' למסרון        | 5001-10000          |
| 3 אג' למסרון        | >10000              |

כתב תוכנית הקולעת את מספר המספרונים 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 המחשבור במים, הוחלט במשלה להעלות את תערify 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 חדש, כאשר התשלום עבור הцrica 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 הקולעת את מספר האנשים המתגוררים בבית ואת צרכת המים (בקוב) ומחשבת את הסכום לתשלום. עורך את הסכומים לתשלום המופיעים בהדפסה וצמצם את מספר הספרות שמנון למקודם העשונית, כך שייתאימו 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 מחלק 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 הנוצר) ומחשבת אם היא שנה מעוברת. שנה מעוברת היא שנה שבה יש 29 ימים בפברואר. שנה מעוברת היא שנה המחלקת ב – 4, לא מחלקת ב – 100, אבל כן מחלקת 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
>>>
```

27. כתוב תוכנית המחשבת את ערכו של המספר 1111, בהנחה שהוא מבוטא במסיים שונים. התוכנית קולטת שני מספרים המייצגים טווח במסיים ומחשבת את ערכו של 1111 בכל אחד מהבסיסים שבתווח.

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 בPython

28. כתוב תוכנית קולטת ערך מספרי שלם  $n$  ומחשבת את סכום הסדרה הנתונה ע"י  $\frac{1}{1+1} + \frac{1}{1+2} + \dots + \frac{1}{1+2n}$ . את התוצאה יש להדפיס עם שלוש ספרות מימין לנקודה העשונית.

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. פורמת הקלדת השעות הוא  $yy.xx$ , כאשר  $xx$  מייצג את השעה ואילו  $yy$  מייצג את הדקות. קלונר השעה 7 ושש דקות מוקלדת ע"י 7.06 עבור 9 שעות הראשונות ביום התעריף הוא 100%, עבור השעות הבאות התעריף הוא 125% והשעות לאחר מכן 150%. את הסכום המחשב יש לעגל ל שקלים 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 מדפסה אותה ולאחר מכן מדפסה 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 ומדפסה אותן איזוד מסודר מຕואר בדוגמה: אם  $a_1a_2a_3a_4a_5a_6$  –  $b_1b_2b_3b_4b_5b_6$  הן שתי המחרוזות, אז המחרוזת החדשה תהיה  $a_1b_1a_2b_2a_3b_3a_4b_4a_5b_5a_6b_6$ . במלילם אחרות, התוכנית תדפיס מחרוזת חדשה שבה במקומות הא' זוגיים מופיעים התווים של המחרוזת הראשונה ובמקומות הזוגיים מופיעים התווים של המחרוזת השנייה. אין צורך לבדוק שהארכיהם זהים.

דוגמאות הרצה:

```
Please enter something...: abcdefg
Please enter something...: zyxwvut
```

```
Original strings: abcdefg zyxvwut
Combined string: azbycxdwevfugt
>>>
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 TWO תוסיף רווח ולאחר כל זוג TWOים תוסיף רווח, LOCEN ועוד רווח

דוגמאות הרצה:

```
Please enter something...: abcdefg
Please enter something...: zyxvwut
Original strings: abcdefg zyxvwut
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, אז המחרוזת החדשה תהיה מופיעים התווים המתאימים של המחרוזת הראשונה ובמקומות הזוגיים מופיעים התווים המתאים 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 בזדדים מייעלהה בכל חודש. תוקף התרופה פג כאשר ייעלהה קטנה מ – 40%. כתוב תוכנית הקולטת את אחוז הקטנת הייעלהה 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 וbashlev ראשון תחשב שהלחץ בכל זוג זהה. אם לא תודפס הודעה שגיאה מתאימה. אם בכל אחד מ הזוגות הלחץ זהה, התוכנית תבודק שהלחץ בצמיגים הקדמיים נמור מזה של האחוריים

דוגמאות ריצה:

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 גודל משלוש יחידות.

דוגמאות ריצה:

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

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 וחמשה

דוגמאות ריצה:

```
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 מדפיסה את המחרוזת, כאשר בכל פעם מורידים את התו הראשון שלה.

דוגמת ריצה:

```
Please enter string: abcdefg
abcdefg
bcdefg
cdefg
defg
efg
fg
g
```

39. שנה את התוכנית, כך שהמחרוזת תודפס ובכל מחזור התו האחרון שלה יושטט.

דוגמת ריצה:

```
Please enter first number: 12345
12345
1234
123
12
1
```

40. כתוב תוכנית המבקשת מהמשתמש שתי מחרוזות ומדפיסה אותן, כאשר ביןיהן היא מדפיסה נקודות, וכך התווים בשורה הכוילת את שתי המחרוזות והנקודות הוא 30. אם מספר התווים בשתי המחרוזות גדול מ – 30, התוכנית תדפיס הודעה שגיאה

דוגמאות ריצה:

```
Please enter first string: 1234567890
Please enter second string: 1234567890
```

```
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 המיצג צלע של ריבוע המורכב מכוכבות. התוכנית תדפיס את הריבוע, כאשר בין כל שתי כוכבות יש רווח אחד.

דוגמאות הרצה:

```
Please enter square size: 7
```

```
* * * * * * *
* * * * * * *
* * * * * * *
* * * * * * *
* * * * * * *
* * * * * * *
* * * * * * *
```

```
>>>
Please enter square size: 3
```

```
* * *
* * *
* * *
```

42. שנה את התוכנית הקודמת כך שבמקרה שתדפיס ריבוע מלא היא תדפיס רק את שלד הריבוע

דוגמאות הרצה:

```
Please enter square size: 8
```

```
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
* * * * * * * *
```

43. כתוב תוכנית הקולטת מספר המיצג צלע של ריבוע המורכב מכוכבות. התוכנית תדפיס את חצי הריבוע העליון (מעל האלכסון), כאשר בין כל שתי כוכבות יש רווח אחד.

דוגמאות הרצה:

```
Please enter square size: 7
```

```
* * * * * * *
* * * * * * *
* * * * * * *
* * * * * * *
* * * * * * *
* * * * * * *
* * * * * * *
```

```
>>>
Please enter square size: 4
```

```
* * * *
```

## תכנות בPython

```
* * *\n* *\n*
```

44. שנה את התוכנית הקודמת כך שתדפיס את חצי הריבוע העליון השני.

דוגמאות הרצה:

Please enter square size: 6

```
***** *\n**** *\n*** *\n** *\n* *
```

Please enter square size: 4

```
*** *\n** *\n* *
```

45. כתוב תוכנית הקולטת מספר  $n$  מהמשתמש ובצורה מחזוריית מחשבת ממנו מספר חדש. בכל מחזור, אם  $n$  זוגי, המספר החדש מחושב  $u_i/2$ . אם  $n$  אי-זוגי, המספר החדש מחושב  $3u_i+1$ . התוכנית תדפיס את מספר המחזורים הנדרש עד אשר התוצאה המתבקשת 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 שתתקבל מהמשתמש שני מספרים המייצגים תחום ועbor כל אחד מהמספרים התחום (כולל הקצוות) תחשב את מספר המחזורים הדרוש כדי להגיע לערך 1. התוכנית תדפיס את המספר (בתוחם) עבורו מספר המחזורים הוא הגדל 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 בPython

47. כתוב תוכנית המקבלת ציון כנהוג בארץ (בתוחם 0-100) ומתרגם אותו לציון כנהוג בחו"ל. נוסחת ההמרה מוגדרת בטבלה

| תנאי | ציון              |
|------|-------------------|
| A    | = ציון <= 90      |
| B    | = ציון > 90 >= 80 |
| C    | = ציון > 80 >= 70 |
| D    | = ציון > 70 >= 60 |
| F    | > ציון 60         |

דוגמאות ריצה:

```
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"מיזה" מחרוזת. התוכנית תבקש מהמשתמש להקליד מחרוזת ואחריה מספר ח. התוכנית תדפיס את המחרוזת, כאשר התו הראשון הרשום הוא זה הנמצא במקומות ח. לאחר התו האחרון יודפס התו הראשון ועד למיקום ח. אם ח גדול מאורך המחרוזת, תודפס הودעת שגיאה.

דוגמאות ריצה:

```
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 ומדפסה "bingo" אם הוא כולל את הספרה 7, או שהוא מחלק ב – 7.

דוגמאות ריצה:

```
Please enter a number: 57
Bingo!
>>>
Please enter a number: 84
Bingo!
```

```
>>>
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 מערכיים 1,2. המחרוזת מייצגת תוצאות של משחק כדורגל, כאשר 1 מייצג ניצחון לקבעה הביתית, 2 מייצג הפסד לקבעה הביתית ואילו X מייצג תיקו. התוכנית מדפיס 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 מדפיס הודעה 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 בחזקת שלישית

דוגמת הרצה:

```
153
370
371
407
```

52. נדע ידוע החיליט לתרום מכספו לנזקקים. כדי שלא לפגוע בחילשים, אשר ברוב המקרים אינם הראשונים בתור, החיליט לתרום לכל נזקק \$100 יותר מאשר לנזקק שלפניו בתור. במיללים אחרות, הראשון יקבל \$100, השני \$200, השלישי \$300 וכן הלאה. כתוב תוכנית אשר קולטת את סכום התרומה ומדפיסה כמה נזקקים יהנו ממנו (כולל האחרון שיקבל את הסכום שנותר, אולי פחות ממה שmagiu לו לפיה חישוב)

דוגמאות הרצה:

```
Please enter total sum: 500
3 people will enjoy the $500
>>>
```

## תכנות בPython

```
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 מ Abed אחז קבע משקלו בכל שבוע. כתוב תוכנית הקולטת שלושה מספרים. הראשון מייצג את משקלו של אדם, השני מייצג את המשקל הרצוי והשליש את אחז הירידה השבועי. התוכנית תחשב את משך הזמן הדרוש להגיע למשקל הרצוי.

דוגמאות ריצה:

```
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 ריצה:

```
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 צחרור, אז הוא מודפס אם הוא מקיים:  $n_1 + n_3 = n_2^2$

דוגמת ריצה:

```
110 123 138 222 237 321 336 420 435 534 633 732 749 831 848 930 947
>>>
```

## תכנות בסיסי

55. 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  $n_1$  ו- $n_2$ ) המתחלקים בסכוםabrahem.

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
```

56. כתוב פונקציה המקבלת שתי רשימות מספריות ומחזירה רשימה אחת מאוחדת וממוינת, כאשר איברים כפולים הושמו 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]
```

57. כתוב פונקציה המקבלת פרמטר שהוא מספר שלם ( $n$ ) ומדפיסה את הסכום הסדרה הנתונה ע"י  $1/1, 1/2, 1/3, \dots, 1/n$  התוצאה תודפס בדיק 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
```

58. כתוב פונקציה המקבלת שני פרמטרים מספריים ( $m, n$ ), אשר מוחק. הפונקציה תבודוק שאכן  $m < n$  ואם לא תדפיס הודעה שגיאה. הפונקציה תדפיס שורות של כוכביות, אשר בשורה הראשונה מכוכビות, בשורה שאחריה  $n-m$  כוכביות עד לשורה המכילת  $m-n$  כוכביות. לאחר מכן שורה שבה  $n-m$  כוכביות, וכן השורה עד לשורה בעלת  $n-m$  כוכביות.

דוגמאות הרצה:

```
Min = 2 Max = 1
Min is larger than Max
```

```
>>>
Min = 2 Max = 5
```

```
**
```

```

```

```

```

```

```

```

```

```

```

```
**
```

```
>>>
```

```
Min = 2 Max = 3
```

```
**
```

```

```

```
**
```

60. כתוב פונקציה המתקבלת 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 המתקבלת פרמטר שהוא מחרוזת בעלת אותיות קטנות בלבד. ומקודדת אותה בקידוד קיסר. קידוד (או הצפנה) קיסר מחליף כל אות במחרוזת, באמצעות הנמצאת מספר קבוע של מקומות אחרת. הפונקציה שעלייר לכתוב מותאמת להחלפה של כל אות באות שנמצאת למרחק של שלושה מקומות. לכן, a תփוך להיות d, b תփוך ל - e וכן הלאה. שלוש האותיות האחרונות, z,y,x הופכות להיות c,b,a בהתאם. 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 המתקבלת פרמטר מספרי (n) ומחזירה את האיבר ה - n בסדרת טרי-בונצ'י.  
 $T_0 = 0, T_1 = 1, T_2 = 1, T_3 = T_0 + T_1 + T_2, \dots, T_n = T_{n-1} + T_{n-2} + \dots + T_0$

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
```

## תכנות בPython

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  $y$ , אז  $y = \frac{x}{2} + \frac{1}{x}$  מהוות הערכה מדויקת יותר. יש להמשיך בחישוב עד אשר הפרש בין שתי איטרציות של הערכה 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 3.000000000393214 (with accuracy of 0.001)

>>>

The approximation for 9 SQRT is 3.0 (with accuracy of 1e-10)

64. כתוב פונקציה המקבלת פרמטר מסווג (n) ומחשבת את כל המספרים הראשוניים בתחום ח' - 1. יש להשתמש בשיטת המסננת של ארטוטנסו. ש להציג רשימה של כל המספרים הראשוניים בתחום ולמחק ממנו את כל הכפולות של 2,3,4 וכן הלאה. המספרים שנפטרו ברשימה הם מספרים ראשוניים.

הערה: את הרשימה יש למשב עזרת ערכיהם 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 מסווג מחוץ זה והוא מדפסה אותה במעין משולש. שורה ראשונה מכילה את האות הראשונה, בשורה השנייה האות השנייה מודפסת פעמיים, בשורה השלישית האות השלישי מודפסת שלוש פעמים וכן הלאה. בין כל שתי אותיות זהות ידפסו חמייה רווחים.

דוגמאות הרצה:

Python

=====

P

y       y

t       t       t

## תכנות בPython

```
h h h h
o o o o o
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
I I I I I I I I
N N N N N N N N N
G G G G G G G G G G
```

66. כתוב פונקציה המקבלת פרמטר שהוא רשימה של מספרים. הפונקציה תחזיר את ההפרש המינימלי 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 ספורות 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 ספורות בלבד. הפונקציה מחזירה רשימה שבה כל איבר הוא סכום שלושה איברים במחרוזת המקורית. את החלוקה לשולשות יש להתחיל מצד 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]
>>>
```

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 מספרי בוליאני אשר מגדיר אם עיר מסוימת היא עיר בירה,  
 b. השני מספרי, לציין מספר התושבים בעיר  
 c. השלישי מספרי לציין הארנוונה המשולמת עבור כל אזור  
 הפונקציה תחזיר אם העיר היא מטרופוליס וشكער עם לא. כדי שעיר תהיה מטרופוליס  
 היא חייבת להיות עיר בירה ובעלת למעלה מ – 100,000 תושבים, או לחילופין כל עיר שבנה  
 למעלה מ – 200,000 תושבים ובלבד שההכנסות שלה גבוהות מ – 800,000,000. כתוב  
 את הפונקציה בצוותה כדי שהיא משתמשת בתנאי בוליאני ייחד.

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 המתקבל פרמטר מספרי שלם (n) ומדפיסה סדרה של מספרים שלמים. אם  
 המספר אי-זוגי, המספר הבא מחושב ע"י  $n+1-3^*$ . אם המספר זוגי, המספר הבא מחושב ע"י  
 $n/2$ . הפונקציה תדפיס את כל המספרים בסדרה, עד שהוא מגיעה ל – 1=ה.

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 חישוב פשוט. התכנית קולטת מהמשתמש שלושה פרמטרים (מספר,  
 סימן ומספר), כאשר הסימן יכול להיות אחד מרבעה סימנים: +, -, \*, / לאחר מכן התכנית  
 תבצע את הפעולה שהຕבקה. למשל אם הוכנסו הפרמטרים 13, + 12 - התוצאה שתתקבל  
 היא 25. לעומת זאת עבור  $4 * 15$  התוצאה שתתקבל היא 60 אין צורך לבדוק את חוקיות  
 הסימן.

## תכנות בPython

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 הקודמת, כך שהמקומ שתשמש בשלוש פקודות קלט (לקראת שלושה הפקודים), התכנית תשמש בפקודת קלט אחת הכול מחרוזת של שלושה הפקודים. פרט לשינוי הקלט, התכנית תתנהג 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 תוכנית הקולעת שני מספרים ומחשבת את סכום המספרים שביניהם. כל אחד מהמספרים יכול להיות הקטן ביןיהם

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 בPython

```
Enter number: 8
Enter number: 8
Sum of numbers in range 8-8 = 0
>>>
```

74. כתוב תכנית המדפסה (בשורה אחת) את כל המספרים בתחום 1-35 (כולל) שאין 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 ומדפסה 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: aaabbbbababab
```

```
Original - aaabbbbababab
Modified - bbbbaaabababa
>>>
```

76. כתוב תכנית הקוראת מספר n ומדפסה "משולש מספרים". במשולש מספרים הכוונה  
שבשורה הראשונה מודפס המספר 1 פעמי' אחד, בשורה מתחתיו המספר 2 מודפס פעמי' שני'  
וכן הלאה עד המספר n שמודפס n פעמי'

דוגמאות הרצה:

```
Enter a number: 5
1
22
333
4444
55555
>>>
Enter a number: 1
1
>>>
Enter a number: -15
>>>
```

## תכנות בPython

```
>>>
Enter a number: 9
1
22
333
4444
55555
666666
7777777
88888888
999999999
>>>
```

77. שנה את התכנית הקודמת, כך שתאת מושלש המספרים תדפיס הצד ימין ולא הצד 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 שתתדייס מושלש מספרים כפול. בשורה הראשונה יודפס המספר 1 פעמיים (באמצע), בשורה תחתיה יודפס המספר 2 ארבע פעמים וכן הלאה עד השורה الأخيرة בה יודפס המספר ח ( $h^*$ ) פעמיים

דוגמאות הרצה:

```
Enter a number: 4
11
2222
333333
44444444
>>>
Enter a number: 2
11
2222
Enter a number: 9
```

```

11
2222
333333
44444444
5555555555
666666666666
77777777777777
888888888888888
99999999999999999
>>>
Enter a number: 0
>>>

```

79. שנה את התכנית פעם אחרונה, כך שתדפיס מעין מספרים.

דוגמאות הריצה:

```

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
888888888888888
888888888888888
77777777777777
666666666666
5555555555
44444444
333333
2222
11
>>>

```

80. כתוב תכנית הקולטת מספר מהמשתמש ומדפיסה ריבוי ריק של כוכיות, כאשר כל ערך הריבוע היא המספר שהוכנס. אפשר להניח שהקלט הוא מספרי וגודל מ – 2.

דוגמאות הרצה:

```
>>>
Enter a number: 9

* *
* *
* *
* *
* *
* *

>>>
Enter a number: 3

* *

>>>
Enter a number: 14

* *
* *
* *
* *
* *
* *
* *
* *
* *
* *
* *
* *
* *

>>
```

81. כתוב תכנית הקולעת שתி מחרוזות באורך זהה. אין צורך לבדוק את האורכים ואפשר להניח שהם זרים. התכנית תאחד את המחרוזות ע"י יצירתה של מחרוזת חדשה שבה איברים משתי המחרוזות לפי הסדר. האיבר הראשון במחוזת 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, ועוד. בנוסף התוכנית תחלק את איברי המחרוזת החדשה לקוביות 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: mnopqrstuvwxyzx
Combined string: /amb/nco/dpe/qfr/gsh/tiu/jvk/wlx
```

82. כתוב תכנית הקולעת תאריך בפורמט dd/mm (שתי ספרות ליום ושתי ספרות לחודש) ומחשבת את מספר הימים ש עברו מתחילת השנה. אפשר להתעלם משנים מעוברות ואפשר להניח שהתאריך שהכנס הינו חוקי. רצ"ב טבלת חודשים ומספר הימים בכל אחד מהם

| חודש                       | ימים                                |
|----------------------------|-------------------------------------|
| 12 11 10 9 8 7 6 5 4 3 2 1 | 31 30 31 30 31 31 30 31 30 31 28 31 |

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 הקולעת 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: 222222222
Number - 222222222 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
```

```
sum = 45 , Multiplication = 362880
>>
```

84. כתוב תכנית הקולטת מחרוזת ומדפיסה מחרוזת שבה כל זוג איברים החליף מקומות- קלומר האיבר הראשון הפרק שני והאיבר השני הפרק ראשון, האיבר השלישי הפרק רביעי והאיבר הרביעי הפרק שלישי וכן הלאה

דוגמאות הרצה:

```
Enter a string: 1234
2143
>>
Enter a string: abcdefgh
badcfehg
>>
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 מחירן של כל מוצר. כאשר מספר המוצרים שנרכשו (הערך הראשון במחרוזת) הוא אפס, הדבר מציין את סוף הקלט. התכנית\_TDPIIS את סך הקניה,תווסף את המ"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 לנכונות את השכר נטו. חוקי

הчисוב הם כדלקמן:

על 4000 שקלים הרשונים אחוז המס הוא אפס.

על 3500 שקלים הבאים (עד שכר של 7500 ש"ח) המס הוא 15%

על 2500 שקלים הבאים (עד שכר של 10000 ש"ח) המס הוא 25%

על 5000 שקלים הנוספים (עד שכר של 15000 ש"ח) 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 הגדל שביניהם. התוכנית  
תדפיס את מס' המספרים שקרה, המספר הקטן והഗדול והממוצע שלהם (כsharp עם שתי'  
ספרות מימין לנקודה העשורה). 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
```

## תכנות בסיסי

>>>

88. כתוב תכנית הקולטת מחרוזת ומדפיסה אותה לאחר שהתו הראשון והאחרון הוחלפו ביניהם. למשל התו הראשון יהיה במקומות האחרן והטו האחרון יהיה במקומות 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
lbcdefghijk
>>>
```

89. כתוב תכנית הקולטת מחרוזת הכוללת חלק משפט 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" מצד 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 בPython

```
Enter shift count: 0
123456789
>>>
Enter string: 123456789
Enter shift count: 15
456789123
>>>
```

91. כתוב תכנית הקולטת מספר ומדפיסה את כל המספרים הריבועים הקטנים מהמספר שהוקלד. המספרים יודפסו בשורה אחת, כאשר הם מיושרים לימין. בכל וירה 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 ראשון בפורמט HH:MM ופרמטר שני מייצג את מספר הדקות שעברו. התכנית תחשב ומדפיס את הזמן החדש ע"י הוספת הדקות לזמן המקורי

דוגמאות הרצה:

מהדורה 3.8

```

>>>
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 מהם מחזרות. התוכנית תדפיס מהזרות מאוחדת (המורכבת מרששור שלוש המזרות), כאשר סדר השרשור מתבצע לפי אורך המזרות.

דוגמאות הריצה:

```

>>>
Enter first string: aa
Enter second string: bbb
Enter third string: cccc
aabbbcccc
>>>
Enter first string: aaaa
Enter second string: bb
Enter third string: ccc
bbcccaaaa
>>>
Enter first string: aaa

```

## תכנות בPython

```
Enter second string: bbbb
Enter third string: cc
ccaaabbbb
>>>
```

94. כתוב תכנית הקולטת מחרוזת ומדפיסה מחרוזת חדשה שבה כל איבר מצין את מספר המופעים של האיבר המתאים)mחרוזת המקורית. כולם אם המחרוזת שהוכנסה היא "abacb" המחרוזת שתודפס היא "22212". בגלל שיש שני א' במחרוזת המקורית, בכל מקום שהוא בו א' תוכנס הספרה 2.

דוגמאות הרצה:

```
>>>
Enter first string: ababab
333333
>>>
Enter first string: abaccdadab
4242224242
>>>
Enter first string: abcd
1111
>>>
Enter first string: pqwpew
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 בPython

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 מהחרוזות בעלות שני תוים, כאשר התו הראשון מהחרוזת הראשונה והשני מהחרוזת 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 בPython

```
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 ח המיצג "חלון" מקסימלי. התכנית תדפיס מחרוזות המהוות מעין "חלון" של תווים מטרח המחרוזת. בשלב הראשון יודפסו מחרוזות הצלולות שנינוים (תו ראשון ושני, לאחר מכן,תו שני ושלישי וכן הלאה). לאחר מכן יודפסו מחרוזות הצלולות שלושה תווים וכן הלאה עד למחרוזות הצלולות ח 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 efgf fghi ghij hijk ijk l jklm
abcde bcdef cdefg defgh efgfhi fghij ghi jk hijkl ijk lm
>>>
```

100. כתוב תכנית הקולטת מחרוזת הצלולת מספרים מופרדים בפסיקים. התכנית תחשב את ממוצע המספרים שבמחרוזת ותדפיס מחרוזת חדשה שבאה במקום כל איבר מופיע ההפרש שבין האיבר המקורי לממוצע שחושב. האיברים בחרוזת המקורי ובחרוזת 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. הפור מבסיס דצימלי (עשרוני) לבינארי:

1. 1011111101
2. 10011010010
3. 1011001101011
4. 1111101001
5. 10001111010101
6. 100000000000
7. 1111111111
8. 11110000000101
9. 1111100111
10. 1111110010

2. הפור מבסיסBINARI (בינארי) לדצימלי (עשרוני):

1. 99
2. 199
3. 299
4. 35
5. 111
6. 1234
7. 88
8. 175
9. 144
10. 288

3. הפור 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

4. הפרק 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 | הקסדצימלי |
|-----------|--------|--------|--------|--------|-----------|
| 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

חזר לתוכן

**תשובות**

**פתרון מחשבוני**

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

[חזרה לפרק](#)

**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

[חזר לתוכן](#)

**תשובות**

**דוגמת תצורת 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

[חזר 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

חלק ב'

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

[חזר לתוכן](#)

**תשובות**

**משתנים**

1. 5
2. 6
3. 2.0
4. 0. 333333333333
5. 0.5
6. 0. 666666666667
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. 0.333333333333
5. 0.5
6. 0.666666666667
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. 0.333333333333
5. 0.5
6. 0.666666666667
7. 1.0
8. 4.0
9. 7.5
10. 1.5
11. 62.5
12. 2.2360679775

[חזר 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\\")

[חזרה לתוכן](#)

**תשובות**

**עוד על סוגים 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'

חזר [לtopic](#)

**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

[חזרה לתוכן](#)

**תשובות**

**טיפול במחuzeות**

- 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. "

**חזרה לתוכן**

תשנות

הדף



חזר לתוכן

## תשובות

עוד על הדפסה

**תכנות בO/O.**

/ / \ \  
| | | |  
\ \ //  
  \ /

12.       / \/  
/ / \ \/  
-----  
\ \ //  
  \ /

13.       / \/  
/ \/  
/ \/  
/ / \ \/  
/ / \ \/  
/ / \ \/  
| | | |  
| | | |  
| | | |  
| | | |  
\ \ //  
\ \ //  
\ \ //  
  \ /  
  \ /  
  \ /

14.       / \     / \     / \     / \     / \     / \/  
--     --     --     --     --     --  
  \ /     \ /     \ /     \ /     \ /     \ /

15.       / \/  
oo  
\ /

16.           1  
12  
123  
1234

17.       xox o xo xox  
o xo xox o xo  
xox o xo xox

18.       xox    o xo    xox  
o xo    xox    o xo  
xox    o xo    xox

19.       \ ~ / ~ \ ~ / ~ \ ~ / ~ \ ~ /

20.       \ ~ / ~ \ ~ / ~ \ ~ / ~ \ ~ /  
  \ ~ / ~ \ ~ / ~ \ ~ /

תכנות בסיום

21. \~/\~/\~/\~\~/\~/\~/  
      \~/\~\~/\~/\~/\~/  
      \~/\~\~/\~/  
      \~/

$$22. \quad \begin{array}{c} \backslash \diagup \diagup \diagup \diagup \\ \backslash \diagup \diagup \diagup \\ \backslash \diagup \diagup \\ \backslash \diagup \end{array}$$

[חזר 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

**תכנות בס.ו.**

|        |  |
|--------|--|
| 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 בO/O.**

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 מדפסה 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

## תוכנות בסיסי

2.0  
2.5  
3.0

35. התוכנית מחשבת את ממוצע ריבועי שבעה המספרים (6-0)  
13.0

36. התוכנית מדפיסה את הערך המוחושב ע"י סכום האיברים בראשימה מחלוקת בערך האיבר האחרון ועוד אחד.

4.6

37. התוכנית מכפילה את האיברים בראשימה, מכפילה את הכל באפס ומחשבת מודולו חמיש.

0.0

38. התוכנית מחשבת את ערכי המספרים בתחום שבין מינוס 32 ועד מינוס 22 (כולל) ועד 30.

-2  
-1  
0  
1  
2  
3  
4  
5  
6  
7  
8

39. התוכנית מדפיסה את המספרים 8-0, כאשר בכל שורה מתווסף מספר אחד. וכך נוצר משולש שקודקודה ריק ובבסיסו הרשימה 8-0.

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 כוכビות, כל סדרה בשורה חדשה כאשר בכל סדרה מתווספות עוד כוכビות. בסדרה הראשונה כוכבית אחת, בסדרה השניה כוכבית אחת נוספת (אחרי רווח) שתי כוכビות, בסדרה השלישית כוכבית אחת, אחרת שתיים ואחריהן שלוש וכן הלאה עד לסדרה האخונה שבה רשימה של כוכבית אחת, שתים, שלוש ועד שבע כוכビות.

\*  
\* \*\*  
\* \*\*\*  
\* \*\*\* \*\*\*  
\* \*\*\* \*\*\* \*\*\*\*  
\* \*\*\* \*\*\* \*\*\*\*\*  
\* \*\*\* \*\*\* \*\*\*\*\* \*\*\*\*\*  
\* \*\*\* \*\*\* \*\*\*\*\* \*\*\*\*\* \*\*\*\*\*

41. התוכנית מסבירה את מושג הלולאות המקנון ומדפיסה את מונח הלולאה בכל מבחן. המבחן ראשון לא מתבצע. במחזור השני מונח הלולאה החיצונית הוא אחד ומונח הלולאה

## תכנות בסיסי

הפנימית (k) הוא אף. לאחר מכן מתקדם מונה הלולאה החיצונית ומבצעים פעמיים את הלולאה הפנימית וכן הלאה עד שמונה הלולאה החיצונית מגיע לערך 6 ועבורו מבצעים פעמיים 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

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

$$\begin{array}{l} 1 \times 2 = 2 \\ 2 \times 2 = 4 \\ 3 \times 2 = 6 \\ 4 \times 2 = 8 \\ 5 \times 2 = 10 \\ 6 \times 2 = 12 \end{array}$$


---

$$\begin{array}{l} 1 \times 3 = 3 \\ 2 \times 3 = 6 \\ 3 \times 3 = 9 \\ 4 \times 3 = 12 \\ 5 \times 3 = 15 \\ 6 \times 3 = 18 \end{array}$$


---

$$\begin{array}{l} 1 \times 4 = 4 \\ 2 \times 4 = 8 \\ 3 \times 4 = 12 \\ 4 \times 4 = 16 \\ 5 \times 4 = 20 \\ 6 \times 4 = 24 \end{array}$$


---

$$\begin{array}{l} 1 \times 5 = 5 \\ 2 \times 5 = 10 \\ 3 \times 5 = 15 \\ 4 \times 5 = 20 \\ 5 \times 5 = 25 \\ 6 \times 5 = 30 \end{array}$$


---

$$\begin{array}{l} 1 \times 6 = 6 \\ 2 \times 6 = 12 \end{array}$$

## תכנות בסיו.

$$\begin{aligned}3 \times 6 &= 18 \\4 \times 6 &= 24 \\5 \times 6 &= 30 \\6 \times 6 &= 36\end{aligned}$$

---

45. התוכנית בונה את לוח הכפל עבור 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 מדפסה את המספרים בתחום 8\*7-1 כאשר הם בנויים בצורה 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 מדפסה סדרה של חמישה משולשים, כאשר כל משולש מורכב מארבע שורות. בשורה הראשונה כוכבית אחת, בשניה שתיים וכן הלאה עד אשר בשורה הרביעית ארבע כוכビות. הראשונה מוסetta two אחד ימינה וכל סדרה מוסetta בתו אחד נוספת.

\*  
\*\*  
\*\*\*  
\*\*\*\*

\*  
\*\*  
\*\*\*  
\*\*\*\*

\*  
\*\*  
\*\*\*  
\*\*\*\*

\*  
\*\*

\*\*\*  
\*\*\*\*  
  
\*  
\*\*  
\*\*\*  
\*\*\*\*

48. התוכנית מדגימה קינון לולאות ועבור כל לולאה חיצונית מבצעת שלוש לולאות פנימיות.  
התוכנית מדפיסה את ערכי מונה הlolאות (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 הlolאה 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 חמיש פעמים ועבור כל פעם שלה, c מתבצעת חמיש פעמים ועבור כל פעם שלה, d מתבצעת חמיש פעמים, סה"כ  $5^{*}5^{*}5^{*}5$ )

625

50. דוגמה אחרת של קינון לולאות. הlolאה החיצונית מבצעת 6 פעמים ועבור כל מחזור שלה מבוצעת לולאה פנימית עבור הערכים 1-מונה הlolאה החיצונית. הlolאה הפנימית מדפיסה את ערך מונה הlolאה הפנימית, ערך מונה הlolאה 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

## תוכנות בסיסי

6 3 18  
6 4 24  
6 5 30  
6 6 36  
-----

51. התוכנית מדפיסה את המחרוזות בראשימה הנתונה, כאשר המחרוזת מודפסת פעם אחת כמו שהיא ופעם שנייה לאחר ששוכפלת פעמיים.

ab ababab c ccc

52. לכל אחת מהמחרוזות בראשימה outer, התוכנית מחברת כל אחת מהמחרוזות בראשימה .inner

ABab  
ABcd  
ABef  
CDab  
CDcd  
CDef  
EFab  
EFcd  
EFef

53. התוכנית מדפיסה את תווים המחרוזת, כך שהתו הראשון מודפס בשורה הראשונה (פעם אחת), התו השני מודפס פעמיים בשורה השנייה וכן הלאה.

A  
BB  
CCC  
DDDD  
EEEEE

54. התוכנית מדפיסה את תווים המחרוזת בשורה אחת, כך שהתו הראשון מודפס פעם אחת, התו השני מודפס פעמיים וכן הלאה.

A BB CCC DDDD EEEEEE

55. התוכנית מדפיסה את תווים המחרוזת במספר שורות. בשורה הראשונה מודפס התו הראשון (פעם אחת). בשורה השנייה המודפס התו הראשון פעם אחת והתו השני פעמיים. וכך בכל שורה מתווסף התו הבא מוכפל במספר השורה.

A  
A BB  
A BB CCC  
A BB CCC DDDD  
A BB CCC DDDD EEEEEE

56. התוכנית סופרת בביניarity מופיע 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 בC#.

```
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 מדפסה את האותיות Word, כל אות בשורה נפרדת

```
W
o
r
d
```

58. התוכנית מוסיף אחד לאיובי הרשימה ומדפסה אותה

[2, 3, 4, 5]

59. התוכנית מוסיף 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 ומדפסה את המספר הגדל ביותר ברשימה

largest value : 44

61. התוכנית מחפשת ומדפסה את המספר הגדל ביותר ברשימה עד האיבר שערכו 16

largest value : 31

62. התוכנית מחפשת ומדפסה את המספר הגדל ביותר ברשימה , אבל תוך הタルמות מהמספר

44

largest value : 40

63. התוכנית מחברת את הערך 5 ארבע פעמים ואת התוצאה מחברת שלוש פעמים (תוכנית סינטטית שנועדה להציגם שלוש לולאות מקוננות).

60

[חזר 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. ( $1 < 1$ ) == False  
      ( $1 < 2$ ) == True  
      ( $1 < 3$ ) == True  
      ( $2 < 1$ ) == False  
      ( $2 < 2$ ) == False  
      ( $2 < 3$ ) == True  
      ( $3 < 1$ ) == False  
      ( $3 < 2$ ) == False  
      ( $3 < 3$ ) == False

**חזר לתוכה**

**תשובות**

**קלט**

1. 6 66 <class 'str'>
2. 7 3 777 <class 'str'> <class 'int'> <class 'str'>  
999 4 999999999999 <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

10. 12345678 <> 8642  
90123456 <> 6420  
abcd <> db  
n <>  
01010101 <> 1111

11. 12345678 </> 18273645  
901238456 </> 96051428  
abcd </> adbc  
n </>  
01010101 </> 01100110

[חזרה לתוכן](#)

**תשובות**

**תנאים**

1. התוכנית מדפיסה ריבוע כפל של 9-4, 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 9-4, כאשר כל מכפלה של איברים שווים מוחלפת ב - \* .

16 \* \* \* \* \*  
 \* 25 \* \* \* \*  
 \* \* 36 \* \* \*  
 \* \* \* 49 \* \*  
 \* \* \* \* 64 \*  
 \* \* \* \* \* 81

2. התוכנית מדפיסה ריבוע כפל של 9-5, 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 מסוף האיברים גדול. כל האיברים המודפסים הם כוכבות, אלא אם כן אם מסוף השורה ועוד חמישה ועוד מסוף האיבר בתוך השורה שווה או גדול מ – 11 ואז מודפס ? במקום הכוכבית.

? ? ? ?  
 ? ? ? ? \*  
 ? ? ? \* \* \*  
 ? ? \* \* \* \*  
 ? \* \* \* \* \* \*

4. התוכנית מחשבת את ממוצע המספרים בראשימה שלא מתחלקים ב – 2.

5.0

5. התוכנית מחשבת את ממוצע המספרים שהמייקום שלהם בראשימה מודולו 2 ומודולו 5 שווים.

1.5

6. התוכנית מעתקה את המחרוזת, תוך הכפלת הספרות שנמצאות במיקומים שהם מכפלה של .3

123456789 >> 112344567789

7. התוכנית מעתקה את המחרוזת תוך הכפלת התווים שהמייקום שלהם הוא מכפלה של שלוש ושליש התווים שהמייקום שלהם הוא מכפלה של שתיים.

abcdefg >>> aabcccddeeffghh

8. התוכנית יוצרת מחרוזת חדשה על סמך הערכים בשתי מחרוזות הקלט, כאשר האיבר במחוזת 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 מעובר למחרוזת החדשה, אם לא נבדק האיבר המתאים במחוזת b ואמם אינו זוגי הוא מעובר, אחרת מוסיף למחרוזת את הערך 0.

54678492 62384568 50370590

10. התוכנית מחשבת ערך משתנה על סמך ערכי מחרוזת. אם סכום שני איברים סמוכים קטן מחייבים 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 אחד, את המיקום הראשון שלו), את האיבר הקטן ביותר במחוזת ואת המיקום שלו.

857368132843745632837 8 0 1 6

12. התוכנית מקבלת מספר (7) ובודקת אם הוא גדול מעשר (ואז מדפיסה גובה), קטן מבחןש (ואז מדפיסה נמוך), או מדפיסה באמצעות.

In between

חזר [לתוכן](#)

**תשובות**

**RESHIMOT**

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 בסיו'.**

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 בראשימה  
[9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

20. A  
BB  
CCC  
DDDD  
EEEEEE  
FFFFF  
GGG  
HH  
K

הערה: שינוי הגוף נועד להראות את הצורה המתבקשת.

21. A ----- A  
BB ----- BB  
CCC ----- CCC  
DDDD ----- DDDD  
EEEEEE ----- EEEEE  
FFFFFF ----- FFFFFF  
GGGGGGGG ----- GGGGGGGG  
HHHHHHHHH ----- HHHHHHHHH  
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]

**חזרה לתוכן**

## תשובות

## פתרונות while

1. j = 5
2. 55
3. 13
4. 13715
5. 15
6. 108
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. 40  
50
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.  $1 \times 8 = 8$   
 $2 \times 8 = 16$   
 $3 \times 8 = 24$   
 $4 \times 8 = 32$   
 $5 \times 8 = 40$   
 $6 \times 8 = 48$   
 $7 \times 8 = 56$   
 $8 \times 8 = 64$   
 $9 \times 8 = 72$

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

[חזר 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

[חזר 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 בסיוו.**

|    |    |    |    |    |    |    |    |    |     |
|----|----|----|----|----|----|----|----|----|-----|
| 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 |

**חזר לתוכן**

**תשובות**

**טיפול בראשימות**

1. A  
BB  
CCC  
DDDD  
EEEEE  
FFFFFF  
GGGGGGG  
HHHHHHHH  
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]

[חזרה 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

[חזרה לתוך](#)

**תשובות**

**פונקציות לטיפול במחזרות**

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. 'f o l l o w   t h e   s u n'
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\*\*\*\*

[חזר לתוכן](#)

תשנות

עיאוב 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 100000000000 1000000000000000000000000

## חזרה לתוך

## תשובות

**פניות else - i break**

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  
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  
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!

חזר לתוכן

## תשובות

## הבנת תוכניות

|                              |      |
|------------------------------|------|
| 'א                           | .1   |
| 5                            | .2   |
| 0                            | .3   |
| 15                           | .4   |
| 0                            | .5   |
| 10                           | .6   |
| 'ג                           | .7   |
| 6                            | .8   |
| 3                            | .9   |
|                              | .10  |
| 'א                           | לא   |
| 'ב                           | לא   |
| 'ג                           | C    |
| 15                           | .11  |
| 16                           | .12  |
| 8                            | .13  |
| 24                           | .14  |
| 80                           | .15  |
| 245                          | .16  |
|                              | 2.17 |
| 4.0                          |      |
| 4.4                          |      |
| 1                            |      |
| 66.0                         |      |
| 102                          |      |
| 102.0                        |      |
| 96.0                         |      |
| 2.0                          |      |
| 18                           |      |
| [4, 9, 5, 1, 8, 5, 2, 0]     | .18  |
| [5, 6, 1, 9, 9, 1, 6, 5]     | .19  |
| [-1, -2, 0, 4, 7, 2, 4]      | .20  |
| [2, 6, -1, 4, 0, 13, 2, 10]  | .21  |
| [3, 7, 9, 9, 14, 21, 30, 32] | .22  |
| [4, 1, 3, 2]                 | .23  |

חזרה לתוך

**תשובות**

**מצא את החסר**

- .1 ג'
- .2 א'
- .3 ב'
- .4 ג'
- .5 א'
- .6 ג'
- .7 ב'
- .8 ד'
- .9 א'

[חזר לתוכן](#)

## תשובות

## כתיבת תוכניות

**הערה כללית:** הפתרונות הנתונים כאן, אינם בהכרח היעילים או הנכונים ביותר. הם נכתבו מתוך כוונה להסביר את צורת החשיבה בבואהן לפטור 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 (סוג float). עיצוב התוכנית יכול הדפסת כתרתת התוכנית (המסבירה במשפט קצר מה מטרתה), זה אמן לא הופיע בשאלת, אך התווסף למטרת בהירות נוספת. לאחר מכן, שתי פקודות קלט, ואחריהן הדפסה של הנתונים שהתקבלו, כולל חישוב השטח (הכפלת שני המספרים) וחישוב ההיקף שהינו פעמיים 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))
```

במקרה זה נתון שעלינו לקרוא שבעה מספרים ולחשב את סכומם ואת הממוצע שלהם. אנו לא נדרשים לשמר את המספרים שקראנו וכן לאחר קראת כל מספר רק סכום אותו למונה צובר. עיצוב התוכנית יכול הדפסת כוורת קצירה המגדירה מה התוכנית עשויה. לאחר מכן נגידיר מונה צובר (sum) אליו נסכם את כל שבעת המספרים שנקלוט. המונה מאוחחל ל-0.0 (אפשר היה גם לאותו ל-0 ואז המשמעות היא שהיא שהוא INT). לאחר מכן, התוכנית תוכל לולאה המתבצעת שבע פעמים ואשר במסגרתה אנו קוראים את שבעת המספרים. בשאלת לא הוגדר סוג הנתונים של המספרים (INT או FLOAT) וכן, כדי לכתוב תוכנית כללית יותר נבחר במספרים ממשיים (Float). זו הסיבה שאוחחל המונה הצובר יכול 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 המתkeletal 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. בשאלת הקודמת לא נדרשה השמירה ולשםו את שבעה המספרים שנקלרו. כזכור, בשאלת הקודמת לא נדרשה השמירה ולכן כל מספר שנקלר רק סוכם אל המונה הצבור. כדי לשמר, علينا להוסיף רשיימה של שבעה איברים, אשר במקורה זה מאותחלים לאפס. (אפשר גם להגיד רשיימה ריקה, אשר תגדל בכל LOLAH, כאשר נוסיף לה איברים, אך לצורך כך נדרשות פונקציות ספריה, אשר בשלב זה טרם עברנו עליהן). שנית נוסף שעליינו להכין לתוכנית כולל הוספה פקדת בטור הולאה הקוראת את המספרים ואשר נעודה לשמור כל מספר בראשימה שהגדכנו. מונה הולאה משמש למפתח המיקום לשמיירה (במחזור הראשון האיבר יישמר בכתבota 0, במחזור השני בכתבota 1 וכן הלאה). לאחר שקבענו את שבעת המספרים (וגם שמרנו אותם), ולפני שנדפיס את סכומם ואת הממוצע שלהם (כפי שעשינו בתוכנית הקודמת), علينا להדפיסם. הדבר מבוצע ע"י LOLAH נוספת נספtha, אשר מתבצעת שבע פעמים ומדפסה את איברי הרשימה. את איברי הרשימה אשר היה גם להדפיס ללא LOLAH, למשל ע"י שימוש בפקודה `print(list)`, אלא שאז הם יהיו מודפסים כרשימה (מוקפים בסוגרים 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. בשאלת זו אנו נדרשים לקרוא שני מספרים, נניח  $a = b$  וולפי הניסוח אפשר להבין שמדובר במספרים שלמים. لكن בשלב ראשון התוכנית תכלול (פרט להדפסת כוורתה המגדירה את פעולתה) שתי פקודות קלט וכן הגדרת מונה צובר המאותחל ל-0 ואשר אליו נסכם את המספרים. בשלב שני נגדר לו לא אשר תבצע עבור כל המספרים בתחום שבין  $a+1$  ל- $b$  בין  $b$ . בambilים אחרות, הלולאה מתבצעת החל מהמספר הראשון ועוד אחד ( $a+1$ ), שכן לפי תנאי השאלה עליינו לסכם את כל המספרים שנמצאים בין  $a$  ל- $b$  (לא כולל  $a$  ו- $b$  עצםם). הפעמטר השני הקובע את הלולאה הוא  $b$  (מןוי שככל לו לא הפעמטר השני מגדר את הגבול, אך בפועל הלולאה מתבצעת עד (וכולל) האיבר שלפני הפעמטר). הלולאה מתקדמת מ- $a+1$  ועד  $b$  משום שכתוב לנו יכולם להניח שהמספר השני הוא גדול שבין שני המספרים. בכל מחזור של הלולאה נחבר את המספר המתאים אל המונה הצובר, עד אשר הלולאה מסתיימת. השלב השלישי ביצוע התוכנית הוא בהדפסת הטוויה 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. בשאלת זו אנו נדרשים לקרוא שני מספרים שלמים, נניח  $a$  ו- $b$ . אך בשלב ראשון התוכנית תוכל (פרט להדפסת כוורתה המגדירה את פעולה) שתי פקודות קלט לקליטת שני מספרים שלמים. בגלל שמדובר בשאלה פשוטה, אין צורך בשלב של חישובים ואפשר לעורר לשלב האחרון של הדפסת התוצאות, כאשר החישוב יבוצע חלק מפקודות הדפסה. בפקודה ראשונה נדפיס את שני המספרים וכמה פעמים נכנס המספר השני בראשון (הדבר מושג ע"י שימוש בפקודה `//` - שימושו הוא חלק של שלמים). פקודת הדפסה השנייה מדפסה 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 result and the remainder")
 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 remainder is: ", b%a)
```

6. בשאלת זו אנו נדרשים לקרוא שני מספרים שלמים, נניח  $a = b$ , אשר מייצגים את אורך הצלעות של משולש ישר זווית. אפשר להבין מתוך השאלה שמדובר במספרים ממשיים ולכן בשלב ראשון התוכנית תכלול (פרט להדפסת כוורתה המגדירה את פעולה) שתי פקודות קלט לקליטת שני מספרים ממשיים. גם במקרה זה, בגלל שמדובר בשאלת פשוטה, אין צורך בשלב של חישובים ואפשר לעבור לשלב האחרון של הדפסת התוצאות, כאשר החישוב יבוצע חלק מפקודת הדפסה, אשר כוללת את אורך שני הניצבים (הקלט שהתקבל מהמשתמש) ואת חישוב האורך של הצלע השלישי. בגלל שבכל משולש ישר זווית מתקיים משפט פיתגורס הקובע שסכום שטחי הربועים הבנויים על הניצבים שווה לשטח הربוע הבנוי על היתר, אז האורך של היתר מתקבל מחישוב השורש הריבועי של סכום ריבועי הניצבים. בגלל שאנו משתמשים בפונקציה `SQRT` שהינה חלק מהספרייה המתמטית,علינו גם לכלול פקודה `import math` בתחלת התוכנית, כדי לאפשר לתוכנית לגשת לפונקציות הצלולות בספרייה המתמטית.

התוכנית לדוגמה:

```
Tri.py
A program for reading two numbers that represent 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 triangle sides are: ", a, b, math.sqrt (a*a + b*b))
```

7. בשאלת זו אנו נדרשים לקבל מהמשתמש שני קלטים, נניח א – ב. הקלט הראשון הינו מספר (אפשר להבין מトוך השאלה שמדובר במספר שלם) ואילו הקלט השני הוא תוו קלשו. התוכנית אמורה להדפיס ריבוע אשר גודלו מוצג ע"י הקלט הראשון ואילו התווים שמודפסים בריבוע מוגדרים ע"י הקלט השני. כרגע, התוכנית מתחילה בהדפסה של כותרת קצרה המגדירה את פעולה ולאחר מכן מביצעת פקודות הקלט. המימוש יבוצע באמצעות שתי לולאות מקוננות. לולאה ראשונה מגדירה את מספר השורות שיש להדפיס ואילו הלולאה השנייה מגדירה את מספר התווים בכל שורה. בכלל שמדובר בריבוע (מספר התווים בשורה שווה למספר השורות). הלולאה הראשונה תבצע א פעמים ובתוכה תבצע הלולאה השנייה גם כן א פעמים. בתוך הלולאה הפנימית (זו אשר מדפיסה את התווים בשורה), בכל מחזור יודפס התו, כאשר אנו חיברים להבטיח שגם התו הבא יודפס באותה שורה. כאמור, כדי למנוע מפקודות הדפסה לרדת לשורה הבאה ("`z`ו" ברירת המחדל בהדפסות), יש להוסיף את הפרמטר `"=end"`. לאחר שהlolאה הפנימית ה�ביעה א פעמים (משמעותו למלא את השורה), יש לכלול פקודות הדפסה ריקה (כדי להעביר את הסמן לשורה הבאה) ולאחר מכן השורות (מחזור נוסף של הלולאה הראשונה). פירוש הדבר שאנו מתחילה למלא את השורה הבאה בריבוע שלנו. הדבר יחול על עצמו א פעמים, עד אשר מלאנו א שורות, כאשר בכל שורה א תווים

התוכנית לדוגמה:

```
Square.py
A program for reading a number and a character
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. בשאלת זו אנו נדרשים לקבל מהמשתמש מספר אחד כקלט ובורר מתחה השאלת שמדובר במספר שלם. התוכנית מתחילה בהדפסת כותרת קבוצה וממשיכה בקלטת המספר. המימוש במקורה זה יתבסס על לולה אשר תעבור על כל המספרים עד קלט ותבדוק אם המנה המתקבלת מחלוקת המספר שהתקבל במוניה הלולאה היא מספר שלם. אם כן, פירוש הדבר שМОונה הלולאה הוא אחד מהמחלקים. למשל אם המספר שהתקבל הוא 15 ומוניה הלולאה הוא 2, אז בגלל ש – 15/2 אינו מספר שלם, ברור ש – 2 אינו מחלק של 15. לעומת זאת, אם מוניה הלולאה הוא 3, אז בgalל ש – 15/3 הוא מספר שלם, נבע מכך ש – 3 הוא חלק של 15. בקביעת גבולות הלולאה, ניתן לראות בכך שאי אפשר להשתמש בפונקציה range(ב). סתם כך. אם היינו משתמשים כך בפונקציה, הערך הראשון שהוא מוכנס למוניה הלולאה הוא אפס והרי אנחנו לא רוצים לבדוק על אפס. באופן דומה, אנחנו גם לא רוצים לבדוק על 1. אך ניתן להשתמש בפונקציה range, אך לכלול את נקודת ההתחלה שלה (2). באופן דומה, אין טעם למשיך ולבדוק את כל המספרים בתחום. אפשר לעצור את הלולאה לאחר שהיא הגיעה ל – 2/a, שהרי ברור שהמספרים מעבר לאמצע לא מחלקים ללא שארית. לכן, הפעמטר השני של הפונקציה range יהיה 2+1/a. אך, כאשר אנחנו מכנים את 1/a לTOR ההפונקציה range, מתעוררת בעיה אחרת... כל פקודה חילוק גורמת לתוצאה שהיא שבר ואילו הפונקציה range מזכה לקבל פרמטרים שהם מספרים שלמים. לכן, כדי שנוכל להשתמש בערך a/2+1 כחלק מההפונקציה range, علينا להופכו למספר שלם ולכן, בתוך הפונקציה יופיע int(a/2)+1. גוף הלולאה כולל בדיקה לגבי השארית של החלוקה ואם השארית היא 0 (כלומר אין שארית), משמעות הדבר שהМОונה הלולאה הוא מחלק של המספר שהוכנס. במקרה זה הוא מודפס בהמשך השורה, כולל הוספת הפעמטר אשר מורה לפיקודת ההדפסה להישאר באותה השורה ("end"). רק לאחר שהЛОאלה הסתיימה (אם יודעים בזדאות 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 שאלת קצת יותר מעניינת. אנו נדרשים לקרוא מחרוזת (נניח שנקרה לה a) ולבדוק אם היא פלינדרום (מחרוזת סימטרית הנקראת משמאלי ומימין). התוכנית מתחלפת בהדפסת כוורתה קצרה ולאחר מכן כוללת פקודת קלט לкриאה של מחרוזת. את המימוש אפשר לבצע (כמו תמיד) בדרכים שונות. במקרה זה נשתמש בלולאה המתבצעת על איבר המחרוזת ובודקת אם האיבר הראשון שווה לאיבר הלפני אחרון וכן הלאה עד לסוף המחרוזת. ניתן לראות שמעבר על כל המחרוזות הוא בעצם מיותר וכל שאנו ח纠正 צריכים הוא לעבור רק על חצי המחרוזת. מילא החצי השני נבדק תוך כדי הבדיקה של החצי הראשון (שהרי כל איבר בחצי הראשון נבדק מול האיבר המתאים בחצי השני). لكن הלולאה אמונה תתחל באיבר הראשון של המחרוזת, אבל תתקדם רק על אמצעה. את אמצע המחרוזת נחשב ע"י (len(a)/2). אולם, בכלל ככל חלוקה מייצרת תוכאה שהיא מספר ממשי (float) ואילו הפונקציה range מחייבת מספרים שלמים, علينا להמיר את המספר המשמי למספר שלם (תור שימוש בפונקציה int). לצורך המימוש גם נגיד' משטנה sw אשר מאותחל ל-0. המשטנה מניח שהמחרוזת שקיבלנו היא פלינדרום, אלא אם כן נחליט שלא זה המצב. בתוך הלולאה אנו בודקים את האיברים הסימטריים (ראשון מול אחרון, שני מול לפניו אחרון וכן הלאה). אם נמצא מקרה שבו האיברים הסימטריים אינם שווים, נשמע לא היה אף מקרה שבו איברים סימטריים היו שונים ולכן המחרוזת היא פלינדרום. לעומת זאת, אם בתום הלולאה ws לא מאופס, פירוש הדבר שהוא מקרה (לפחות אחד) שבו איברים סימטריים לא היו שווים וזה המחרוזת אינה פלינדרום. סיום התוכנית היא הדפסת הודעה מתאימה המבוססת על בדיקת המשטנה 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 קראותה ע"י:
  - 1) המשטנה ws בעצם מדמה משתנה בוליאני, لكن במקומם להכניס לתוכו ערכים (0 ו-1) אפשר פשוט בשללה אתחול לרשום אותו כאמת (True) ובמהלך ביצוע מחרוזי הלולאה, אם מתגlimים איברים סימטריים שאינם שווים יש להפוך את המשטנה 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 פקودת 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 שאלת קצת יותר מעניינת ואפשר לפטור אותה בדרכים שונות. אמן מדבר על קליטה של מספר ונראה הוא חיב להיות שלם, אך לאור העובדה שנחננו נדרשים להציג את הספרות שבו בסדר הפוך, נראה שעדיף שנקלוט את המספר במחוזות. בצורה זו,agisha לכל הספרות שליה תהיה קלה. لكن, התוכנית תתחיל בהדפסה של משפט קצר המגדיר מה היא עשוה ולאחר מכן קליטה של המספר. לצורך ההמרה השתמש במחוזות ריקה (גניך 6), אשר אליה נוסיף את הספרות ובלולאה שבה מספר המחזוריים נקבע ע"י אורך מחוזות הקלט. הלולאה תתחיל מסוף המחזוריות ותונוע לתחילתה (בסדר הפוך מהרגיל) ובכל פעם תשלוף איבר אחד. ותצרף אותו אל המחזוריות החדשה (b). חשוב לציין שהגשה לפיתוחו שהוצגה כאן מתאימה גם להיפוך מחוזות כלשהו (לאו דווקא מספר) ולכן, אם השאלה מתמקדת רק במספרים, יש לבדוק שהקלט שקיבלנו הוא אכן מספר. דרך אחת לעשות זאת, היא ע"י קליטה של ערך מספרי והפיכתו למחוזות 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 מסווג שונה והיא מחייבת מעט מחשבה. נראה שיש להמיר את התווים שבמחרוזת לערוצים המספריים (הערך ב – ASCII) ולכל התווים א-ב (כולל) יש להוסיף 2. התווים הנוראים (ב, ג, י) יעברו המרה ייחדנית. התוכנית מתחללה עם הדפסה קצרה לגבי יעדות ואחר מכן קולעת את המחרוזת להצפנה (a). בנוסף, התוכנית מגדרה מחרוזת חדשה (b), אשר אליה יוכנסו הערכים החדשניים. בשלב הבא מוגדרת לולאה אשר עוברת על כל איברי המחרוזת (מספר המחרוזים נקבע ע"י האורך של מחרוזת הקלט – len(a)). גוף הולאה כלל את הפקודות להמרת והן מורכבות מפקודת התנינה (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 בשאלת זו איןנו נדרשים להזין את תווי הרוח). לבסוף, כל שאר התווים, ממורים לערכם המספריים (שימוש בפונקציה ord). לערך המספרי מוסיפים 2 ואת הערך החדש שקיבלנו אנו ממיררים חזירה לתו (שימוש בפונקציה chr). השלב האחרון בתוכנית הוא הדפסה של המחרוזת המקורית והמחרוזת המוצפנת. חשוב לציין שהתוכנית משתמשת במידע לגבי ערכי ה – ASCII של התווים השונים ובעובדה שהעריכים 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 המיל הארוך, זו שאלת פשוטה יחסית. התוכנית תכפול הדפסה של שורת כותרת קצרה ולאחר מכן קריית מספר בעל 16 ספרות. הפשט 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 ואמם הספרה במיקום אי זוגי (כלומר כתובתה היא זוגית ...0,2,4,0), אז 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 (גודל מ-9), יש לחברו אל סכום הביקורת. חשוב לציין שהאלגוריתם, במקרה של מספר גדול מ-9, הוגדר בצורה שונה במקצת, אך התוצאה זהה. לפי האלגוריתם, אם המכפלה גודלה מ-9 יש לחבר את סכום הספרות. במקרים אחרים, למשל אם המכפלה היא 16, אז לסכום הביקורת יש לחבר את הסכום של 6 – 1 (7). סכום זה מתקיים גם ע"י הפעחתת 9 מהמכפלה שkn-9=7. לאחר שהיחסנו את סכום הביקורת, אפשר לבדוק אם הוא כפולה של 10 (ע"י שימוש בפונקציה %), אם כן, אז המספר חוקי ואם לא המספר אינו חוקי. כדי לעקוף את הבלבול במיקום הספרות (ספרה במיקום הזוגי, אך כתובת אי-זוגית), אפשר לכתוב את הלולאה כאשר היא מתחילה מ-1 ומשיכה עד לאורך המכילות ועוד אחד. בaczroה זו מיקום הספרה הוא אחד פחות מאשר המונה הלולאה, אך המיקום הזוגי מתאים למונה זוגי ואילו מיקום אי-זוגי מתאים למונה אי-זוגי (הדבר לא משנה במקרים התוכנית, אך אולי מקל על ההבנה). נזקודה חשבנה 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 כמחזורי משום שאז קל לגייע לכל אחת מהמספרות. התוכנית מתחליה בהדפסת יעדיה וממשיכה לקרוא את שלוש המחרוזות. יש להגדיר רשימה, אשר בשלב הראשון היא עדין ריקה. לאחר מכן בולולה מחברים את האיבר המתאים של a לאיבר המתאים של b ומהסכים מחסרים את האיבר המתאים של c. יש לשים לב שלפנינו שוכל לבצע פעילות מתמטית כלשהי (חיבור או חיסור) יש להפוך את המחרוזת למספרים (שימוש ב – int). את תוצאה החישוב יש להוציא (ע"י הפונקציה append) לרשימה, כך שבכלழור של הלולאה היא גדלה באיבר אחד נוספת. 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 הארכאה, הפתרון שלה מובנה ולא קשה במיוחד בהדפסת יעדודה וממשיכה לקרוא את שלוש הרשימות (c, b, a). עקב העבודה שמודבר בראשיות (בהתאם לאיברים 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 ה הפרדה במרקחה זה יהיה פסיק. הרשימות החדשות (atxt, btxt, ctxt) הן רשימות המכילות איברים שהם עצם מחרוזות ולכן לפני שנוכל להשתמש בהם בפעולות חשבונאיות נצטרך להפכם למספרים. יש להגדיר רשימה חדשה (d), אשר בשלב הראשון היא עדין ריקה ולתוכה יוכנסו האיברים המתאים על פי האלגוריתם שהוגדר. לאחר מכן בולאה המבצעת על איברי ctxt מtbody האלגוריתם. יש להיזהר שלא לקובע את מספר מחרורי הלולאה לפי אורכה של המחרוזת a שכן היא ארוכה מהדרוש (מכילה גם פסיקים) ולכן מספר מחרורי הלולאה נקבע על פי מספר האיברים ב ctxt (אשר לפי תנאי השאלה זהה למספר איברי atxt וכנumeric מספר איברי ctxt). בתוך הלולאה מtbody האלגוריתם המבוקש שהוא:

- אם האיבר המתאים ב - c הוא "0", אז 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 המתאים מ מתוך הרשימה הראשונה והשנייה.

בכל המקרים, במהלך ההשוויה יש להשתמש בתווים, משומש שהרשימה ctxt מכילה תווים ולא מספרים. הרשימה d (התוצאה), לעומת זאת מכילה נתונים נתונים מספרים. ולכן כאשר מעבירים איברים מ - atxt או btxt יש להופכם קודם למספרים (בעזרת int). הדבר, כמובן, כמובן גם כאשר יש לחבר את שני האיברים. בשלב הסופי, מודפסות שתי רשימות המקור, רשימת הקריאה והרשימה שהינה תוצאה הריצה. פורמת ההדפסה של רשימת התוצאה שונה (הוא מודפס כרשימה, בעוד כל היתר מודפסים כמחרוזות). אם יש צורך להדפיס את כלם בפורמט זהה, יהיה צורך לטפל ברשימה 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 lists")
 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 הלולאה. למרות שלא צוין, ברור שמדובר במשתנה מספרי (שלם). הלולאה עצמה מחושב ע"י שתי לולאות מקוננות. לולאה ראשונה נועדה להציג את מספר השורות בלוח והלולאה השנייה נועדה להציג את הערכים בכל שורה. בקביעת מסוף המזהירים יש לשימוש לב שהפונקציה range בלולאה צריכה להתבצע עד +1. אם נרשום (a), תחסר לנו השורה الأخيرة. בגין הלולאה נכלול את פקודות הפורמט, אשר מאפשרת לקבוע את מספר התווים שמוקצים לכל מספר ולפי תנאי השאללה נציב שם 5. בנוסף, בפקודת ההדפסה עלינו להבטיח שככל התוצאות מודפסות באותה שורה (ע"י שימוש בתוספת "end="). רק בתום הלולאה הפנימית יש לעبور 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 ("{}{:>5}{:<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 המיצג את המספר נקרא כמחרוזת, כדי לאפשר גישה קלה לאיברים השונים. הקלט השני הוא מספר שלם. התוכנית מגדרה משתנה חדש dec שאמור לכלול את הערך העשורי המוחושב. הלולאה העוברת על כל איברי המספר שהוכנס, מחושב הערך העשורי. האיבר הימני ביותר מוכפל בסיס בחזקת 9, המספר שלשماלו בסיס בחזקת 1 וכן הלאה עד לאיבר השמאלי ביותר המוכפל בסיס 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 ובסיום הלולאה מודפס הערך העשורי.

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 מדפסה כותרת המסבירה בקצרה את מטרתה. לאחר מכן המכונית קוראת את הקלט. הקלט הוא רשימה של ערכים מספריים שאינם מזויינים. בגלל שהם מופרדים בפסיקים, אין ברירה אלא לקרו 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 למש את אלגוריתם מיון הבועות. לו לא רצונה מתחילה מהאיבר השני (שנמצא בכתובת 1) ומשיכה עד לאריך המחרוזת פחות מונה הלולאה הראשונה. בקרה זו, במחזור הראשון, כאשר מונה הלולאה הראשונה הוא אחד, הלולאה השנייה ממשיכה עד סוף הרשימה. במחזור השני, כאשר מונה הלולאה הראשונה הוא אחד, הלולאה השנייה ממשיכת באלבר אחד (כאמור האיבר שכבר נמצא באותו מקום). גוף הלולאה כולל השוואת בין זוג איברים ואם האיבר הראשון גדול מהשני, מתחINU חילוף ביניהם (הצלבת ערכים). לאחר ששתי הלולאות סיימו, הרשימה ממוקמת. החלק האחרון בתוכנית כולל לו לא נוספת אשר עוברת על כל איברי האשימה ומשנה את המחרוזות למספרים, כדי שהדפסת הרשימה הממונית 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)
```

## תכנות בPython

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 נז. לכן, התוכנית מתחילה בהדפסה של כוורת קצורה. לאחר מכן בולולהה של תשעה מחזוריים יש ליחס את שטחו של הריבוע ולהפחית ממנו את שטחו של העיגול. יש לשים לב שרדיוס העיגול הוא מחזית מאורך הצלע. לצורך חישוב שטחו של העיגול משתמש ב – נז מתוך הספרייה המתמטית וכן בתחלת התוכנית גם נוצרך להוסיף את "יבוא" היכולות של הספרייה (ע"י שימוש בפקודה `import math`). נקודה נוספת שיש לשים לב אליה, היא הדרישה שהסדרה המודפסת צריכה להיות בדיק ששלוש ספרות מימין לנקודה העשרונית. הדבר יבוצע ע"י הוספת יכולת עירכה לפיקודת הדפסה (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 לפתרור את השאלה נשים לב שמספר הבעיות תלוי במספר המספרים שמופיע בקלט. ח' מספרים מייצרים (1-ח) טבעות. שטחה של כל טבעת מחושב ע"י ההפרש שבין שטחו של העיגול הגדל בטבעת לשטחו של העיגול הקטן בטבעת. oczywiście יוכל להתחילה בכתיבת התוכנית. התוכנית מתחילה בהדפסה של כוורת קצרה. לאחר מכן קוראת את רשימת המספרים שהוקלדו. בגין שמדובר בראשימה (הכוללת גם פסיקים), עלינו לקרו אותה כמחרוזת. לאחר מכן אפשר לפצל אותה (בעזרת הפקודה `split`, כאשר תז' ההפרצה מחרוזת). כתוצאה מהפיצול, קיבלנו רשימה. נגיד' משטנה (`sums`) המאותחל לאפס ואשר אליו נסכם את שטחי הבעיות. בלולאה שתבוצע על כל איבר הראשימה (למעט האחרון), נחשב את השטח של כל טבעת. במהלך חישוב השטח יש לחתה בחשבוןiscal איבר בראשימה הינו מסווג מחרוזת וכן לפניו שnochol לבצע עלייו פעולות חשבונאיות, יש להפוך אותו למספר. לפ"י תנאי השאלה ברור שהמספרים מייצגים מספרים שלמים וכן ניתן להשתמש בפונקציה `int`. השטח שחושב יודפס וכן יוסכם אל המשטנה `sums`. בתום הלולאה ולאחר שהדפסנו את שטחי הבעיות, נשאר להדפס גם את השטח הכללי של כל הבעיות גם יחד. יש לשים לב שפקודת הדפסה חייבת לכלול עיצור שכן אנו נדרשים להקפיד על מספר הספרות לימין הנקודה העשרונית. גם במקרה זה, לצורך חישוב שטחו של העיגול משתמש ב – נק' מתוך הספרייה המתמטית וכן בתחלת התוכנית גם נדרש להוציא 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 המחשבה רק לגבי הדפסה. התוכנית מתחילה בהדפסה של כוורת קצרה. לאחר מכן קוראת את שני המספרים השלמים (b, a). בגלל שאפשר להניח שהמספר הראשון קטן מהשני, אין צורך לבדוק את התנאי זהה. נגידır משתנה (ch) המציג את החזקה ונתחל אותו לאחד. בולולאת while, אשר מתקיים כל זמן שהמספר הראשון (a) בחזקת ch קטן מהמספר השני (b) נקדם את החזקה (ch) באחד. הלולאה מסתיימת, כאשר התנאי לא מתקיים ואז אפשר להדפיס את התוצאה. החזקה המתאימה היא אחת פחות מהחזקה המוחשבת (ch-1). כדי להדפיס את המספר בחזקה המתאימה ואת תוצאת החישוב ללא רווחים, יש להפוך את המספרים למחרוזות ואז ניתן יהיה להשתמש בסימן "+" כדי להציג את המחרוזות האחת אל השניה.

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 כוורת קצורה. לאחר מכן קוראת את שני הקלטים (מחרוזת (a) ומספר שלם (n)). מגדרים מחרוזת חדשה (b), אשר בשלב זה היא עדין ריקה. בלולאה המתחילה מאחד וממשיכה עד לאחר המחרוזת ועוד אחד, עוברים על כל איבר המחרוזת. בכל מחזור בלולאה מעתיקים את האיבר המתאים אל המחרוזת החדשה. במחזור שהוא כפולה של המספר השלם (n), משכפלים את האיבר המתאים ("ע"י כר שמעתקים אותו פעמיinus). הסיבה שהלולאה מתחילה מאחד וממשיכה עד לאחר המחרוזת ועוד אחד, היא כדי שנוכל לבדוק אם מס' המחרוזת 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 המקורי 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 כוורת קצרה. לאחר מכן קוראת את הקלט (מחוזת a המכילה רשימה מספרי BCD). את המחווזת מפרקם ("ע"י שימוש בפונקציה split) לרשימה של מספרי BCD (אלא שכל אחד הוא מהחווזת מה שיחיב המרתתו למספר לפני שניתן יהיה להשתמש בו לפעולות מתמטיות). מגדירים משתנה חדשה dec שבשלב זה עדין מאפס והוא אמור לכלול את הערך העשורי החדש. בלולה העוברת על כל איברי הרשימה, מפרקם כל איבר לשיביות הבונות אותו (bin1 הוא משתנה המציג את הסיביות bin2 bin מיצג את הסיבית השמאלית וכן הלאה bin4 bin1 – 8 bin הם משתנים המייצגים את שאר הסיביות במספרי ה – BCD). בכל מחזור בלולה יש לכפול את ערכו של המשתנה dec פי 10 שכן אנו מתקדמיים אל הספרה הימנית יותר. בהמשך הלולה מכפילים את הערכים הבינאריים במשתנים (bin1 bin2 bin4 bin8) מוכפל כפול 1, 2 מוכפל כפול 2 וכן הלאה) ומוחברים את התוצאה למשתנה dec. בסיום הלולאה מודפסת הרשימה של מספרי BCD והערך העשורי שחוושב.

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 כוורת קצраה. לאחר מכן קוראת את הקלט (מחרוזת *a* המכילה רשימה מספרים). את המחרוזת מפרקם ("ע"י שימוש בפונקציה *split*) לרשימה של מספרים (אליא שכל אחד הוא מחרוזת מה שיחייב המרתתו למספר לפני שנייה יהיה להשתמש בו לפעולות מתמטיות). מגדרים רשימה חדשה שבשלב זה עדין ריקה ואליה יוכנסו האיברים המוחשבים. כדי שהאיברים ברשימה החדשיה יהיו באותו פורמט כמו איברי הרשימה המקורי, לאחר החישוב, המספרים יומרו חזרה למחרוזת. ראשית נחשב את האיבר הראשון של הרשימה החדשיה. איבר זה שווה לסכום האיבר השני והאיבר האחרון של הרשימה המקורי. בשלב הבא ותיקן שימוש בלולאה נחשב את שאר איברי הרשימה החדשיה (פרט לאחרון). פירוש הדבר שהלולאה תתחיל באחד (במקום אפס) ותשתיים באיבר הלפני אחרון. בגין הלולאה יבוצע 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 החדשיה. 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 כוורת קצרה. לאחר מכן קוראת את הקלט (מחרוזת a המכילה רשימה מספרים). את המחרוזת מפרקם ("ע' שימוש בפונקציה split) לרשימה של מספרים (אליא שכל אחד הוא מחרוזת). מגדרים רשימה חדשה שבשלב זה עדין ריקה ואליה יוכנסו האיברים הייחודיים. ראשית נעתיק את האיבר הראשון ברור שהוא ייחודי. בשלב הבא ותיק שימוש בלולאה המתחילה מהאיבר השני (שהרי את הראשון כבר העברנו) נבדוק אם האיברים ייחודיים. נגדיר משתנה (sw) שמאתחל לאפס ואשר אמור לבדוק אם המספריים ייחודיים. נגדיר לולה נוספת אשר תרוץ מהתחלת המחרוזת ועד לאיבר שנבדק (מוני הלולאה העילונה) ולכל איבר נבדוק אם הוא שווה לאיבר הנבדק. אם יסתבר שכן, פירוש הדבר שהוא איבר כפול ויש להתעלם ממנו. הלולאה תנסה את sw לצין שנמצא כפוף 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 המקורי 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 כוורת קצרה. לאחר מכן קוראת את הקלט (מחרוזת a המכילה רשימה מספרים). את המחרוזת מפרקם ("ע"י שימוש בפונקציה split) לרשימה של מספרים (אליא שכל אחד הוא מחרוזת). מגדרים רשימה חדשה שבשלב זה עדין ריקה ואליה יוכנסו סכומי הביניים. ראשית נעתיק את האיבר הראשון שעבורו לא צריך לסכם דבר. בשלב הבא ותיק שימוש בלולאה המתחילה מהאיבר השני, נסכם את האיברים. הדבר מתבצע "ע"י לולאה מקוונת, 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 המסתכמה את כל האיברים עד לאיבר הנבדק) ניתן להעביר את סכום הביניים לתוך הרשימה החדשה. בשלב הסופי מודפסת הרשימה המקורי 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)
```

### התוכנית לדוגמה:

```
countBase.py
A program to count in various bases

def main():
 print ("This program reads a number (2-9) that represents a base")
 print ("and produces all 3 digit numbers 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))
```

חזר לתוכ

## תשובות

## כתיבת פונקציות

1. זו דוגמה פשוטה. הפונקציה אמורה לקבל מערך שהוא מסוג מחוץ ולקמן כדי להופכה נשתמש במחוץ חדש ריקה ובולולה אשר תעבור על כל איבר מערך הקלט (החל מהסוף) וכל איבר יתווסף אל המערך החדש. בסיום הלולאה, הפונקציה תחזיר את המערך החדש כתשובה.

תשובה לדוגמה:

```
def reverse(a):
 b=""
 for i in range (len(a)):
 b = b + a[-1-i]
 return b
```

בפתרון שתואר, הלולאה מתקדמת מאפס ועד לסוף המחרוזת והאיברים הנשלפים הם מהסוף. אפשר, כמובן גם לכתב את הלולאה בצורה הפוכה, כאשר היא מתחילה מהאיבר האחרון (או ר' המחרוזת פחות אחד) וחוזרת אחוריות לכיוון ההתחלתה. את התחלתה יש להגיד ע"י מינוס אחד משומם שבলולאה אמנים רשות המספר, אלא שהיא נעצרת איבר אחד לפני. במצב זה, מונה הלולאה מגדר את סדר האיברים שיש לשולף מהמחרוזת הישנה.

תשובה לדוגמה:

```
def reverse(a):
 b=""
 for i in range (len(a)-1,-1,-1):
 b = b + a[i]
 return b
```

2. שימוש הפונקציה מתבצע ע"י שתי לולאות מקוונות. 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 - \* השיעיות לאותה שורה, אך יודפסו באותה שורה והדבר מושג ע"י הפרמטר של ""=end, המבטיח שפקודת ההדפסה לא תוסיף את התו שעובד לשורה הבאה.

תשובה לדוגמה:

```
def triangle(a):
 for i in range (a):
 for j in range(i):
 print ("*",end="")
 print ()
 print ()
 return
```

## תכנות בסיסי

3. מימוש הפונקציה מתבצע ע"י לולאה שועברת על כל איברי המערך המקורי. יש להגדיר מערך חדש, אליו יועברו התווים הרלוונטיים. בכל מחזור של הלולאה בודקים אם האיבר המתאים שלו לתו הביטול (הפרמטר השני) ואם כן, מדגים על המשך ביצוע המחזור הנוכחי בלולאה (ע"י פקודת continue). אם מדובר בתו שונה הוא מתווסף אל המערך החדש. בסיום הלולאה, המערך החדש מוחזר כתשובה לפונקציה.

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 יসוכמו כל הציוןים. הדבר מבוצע באמצעות לולאה אחת העוברת על כל האיברים. לאחר מכן, אפשר לחשב את הציון הממוצע (ע"י חילוקת ערך המשטנה במספר האיברים בראשימה. כדי לבדוק כמה ציונים נמצאים מתחת לממוצע וכמה נמצאים לעל (או שוים) לממוצע, שי להגדיר שני משתנים נוספים (maoposim) ובולולאה לעבור שוב על האיברים, כאשר הפעם בודקים כל איבר מול הממוצע. אם הוא גבוה או שווה לממוצע מקדים באחד את מונה הציוןים הגבוהים (המשטנה more בפונקציה) לחילופין מקדים את מונה הציוןים הנמוכים (המשטנה 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]==" ":
 continue # if equal to blank (" ")
 b = b + a[i]
 return b
```

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 מגדרה לולאה העוברת על כל המספרים מ – 0 ועד לפרמטר. כל מספר מומר למערך. מגדרים משתנה (s) שעתיד לכלול את חישוב סכומי הספרות (לאחר שהועלו בחזקה המתאימה). בלולה מקוננת נוספת, עוברים על ספרות המספר ומעלהים 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 העילונה). אם כן, מדפיסים אותו ועוברים למספר הבא בלולאה העילונה. חשוב לציין שככל 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
```

זו כמובן דוגמה קלה. יש פשוט להציב את הפרמטרים בנוסחה (כפי שהופיעה בשאלת) ולהציג את התשובה שהתקבל.

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 מפוקודת ה – `return` ואז הפונקציה תcalcul:

```
def triangle(a,b,c):
 s= 0.5*(a+b+c)
 return math.sqrt(s*(s-a)*(s-b)*(s-c))
```

## תכנות בסיסי

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, אזiert האות המתאימה היא A. בשלב הבא הפונקציה בודקת אם הציון גדול או שווה ל – 80. בגלל שכלי הציונים הגדולים מ – 89 כבר נבדקו בתנאי הראשון, אזןicut כל הציונים שגדולים (ו/או שווים) ל – 80 הם רק אלה שנשארו (בתחום שבין 80 ל – 89) וכךן התנאים המתאימים היא B. הבדיקה ממשיכה, עד אשר כל התחומים נבדקו. חשוב לציין שכארה הפונקציה מגיעה לפקודת ה – return, היא חוזרת למי שקרה 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 פקודת אחת, אז ניתן לכתוב אותה בהמשך התנאי (לאחר הנקודות). במצב זה, הפונקציה מתקצרת ויתכן אף שהיא קרייה 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 ולסכום את התוצאה. חשוב כזכור לוודא שמכפילים תור 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 בעיות מיוחדות. 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 במקרה, 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 התחכחות. בגלל שהיא אחראית לחישוב מס הכנסה ומדרגות המס, כמו גם אחוזי המס נוטים להשתנות, רצוי מאד לצלול את הנתונים במשתנים נפרדים. בימוש הפונקציה הנתונים נשמרו בנזינים, אך לעיתים עדיף דוקא לשמור אותן בראשימה (הmdma טבלה). בgal שהמס הוא פורגרטיבי פירוש הדבר שבעור שכר גובה אחוז המס משתנה (כפי שנition בשאלתך). لكن 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\*10%), ועוד המס עבור המדרגה השנייה 15%\*(7810-4390) ועוד המס עבור המדרגה השלישית 20%\*(11720-7810) ועוד המס עבור המדרגה הרביעית 30%\*(16840-11720) ועוד ההפרש שבין השכר (20,000) והמדרגה הרביעית מוכפל במס. לכן, בgal שבמקרים רבים יש לחשב את המס לכל מדרגה, הפונקציה חישבה מראש את המס המוחיב לכל מדרגה בסדרת משתנים הנקראים step1tax, step2tax,...step5tax. על כן אפשר לגשת לחישוב המס עבור השכר שהועבר כפרמטר לפונקציה. החישוב מורכב מסדרה של תנאים: אם השכר נמוך מהמדרגה הראשונה, אז המס מחושב לפי שכר כפול אחוז המס של המדרגה הראשונה. אם לעומת זאת השכר נמוך מהמדרגה השנייה, אז המס המוחשב הוא סכום המס עבור המדרגה הראשונה (שוחשב מראש ונמצא במשתנה שלו tax1) ועוד ההפרש שבין השכר והמדרגה הראשונה מוכפל באחוז המס של המדרגה השנייה. למשל עבור שכר של 6000 שקלים, המס המוחשב הוא 4390\*10% + (6000-4390)\*15%.

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 להעתלם 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, הגדל ביותר וממוצע הרשימה, הפונקציה מגדרה שלושה משתנים. `Sum` אשר משמש כמונה צובר ואילו `sum` מסכמים את כל איברי הרשימה, `min` שיישמש שמשתנה שבו יוכנס הערך הקטן ביותר – `max` שהוא משתנה שלו יוכנס הערך הגדל ביותר. לכל שלושה המשתנים הפונקציה מעתקה את האיבר הראשון לרשימה. בלולה שמתחליה מאחד (האיבר השני) ועד לאורך הרשימה, הפונקציה מוסיפה כל איבר אל המונה הצובר. לאחר מכן נבדק, אם האיבר קטן מהמשתנה `min` (אם כן, משמענו מצאו 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 `min`). יתר מהאיבר המקורי ולכן יש להעתיקו לתוך `max`). בתום הלולאה, שני המשתנים `min`, `max` מכילים את האיבר הגדל ביותר והקטן ביותר ברשימה. יש להחזיר אותם וכן את הממוצע המחשב 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 מגדרה מערך חדש אליו יוכנסו התווים הרלוונטיים. לאחר מכן, בלולה מתחליה מאחד וממשיכה עד לאורך המערך ועוד אחד, נבדק מונה הלולאה. אם הוא מתחילה – `אם הוא שארית – הפונקציה מודולו` (בפרמטר השני של הפונקציה, משמע יש להעתיק את האיבר המתאים. יש רק לשימר לב שיש להעתיק את האיבר, אשר האינדקס שלו מיוצג ע"י מונה הלולאה פחות אחד (שכן הלולאה התחליה 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
```

## תכנות בסיסי

15. הפונקציה מגדרה משתנה sum אשר אליו מסכימים את ריבועי המספרים. בוללה שהולכת מאפס ועד לפרמטר שהועבר (ועוד אחד) מחשבים את ריבועו של מונה הלולה ומסכימים למשתנה sum. אפשר כמובן לשנות את הלולה, אך שתתחל 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 מתחילה בערך 97 עבור האות א ומסתיים ב – 122 עבור ז. הערך של האותיות הגודלות לעמודת זאת קטן ב – 32 (בהתאםה). לעומת זאת, ערכה של A הוא 65 ואילו ערכה של Z הוא 90. מצדדים במיידע זה אפשר להתחיל לתכנן את הפונקציה. מגדרים מחרוזת חדשה ובלולה העוברת על כל איברי המחרוזת המקורי, בודקים את המספר הסידורי של התו. אם הוא בתחום שבין 97 ל – 122 (כולל) יש להפחית ממנו 32 ולהפכו חוזרת לו. אחרת יש להחזירו כmoות 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, הלולה שנגדיր מתחילה ב – 2 ומתקדמת עד לפרמטר שקיבלנו. אם הפרמטר מחלק ללא שארית במונה הלולה, משמע שמצאנו מחלק. בغالל שאנו 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 הלולה שכן אין טעם להמשיך ולבדוק את כל המספרים. אפשר לעצור את הבדיקה באמצע התחום. למשל אם אנחנו רוצים למצוא מחלק של 30, 30, אפשר להסתפק בבדיקה של 15 מספרים שחיי בורר שהמספרים הגודלים מ – 15 לא יכולים להיות מחלקים. במצב זה, 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 לקבל את הספרה השמאלית. באופן כללי, יש לחלק את המספר בעשר בחזקת מספר ספרות פחות אחת. כדי לקבל את הספרה השמאלית ללא שארית נשתמש בחלוקת של שלמים "///". כדי לחלק מהפונט שקיבלנו את מספר הספרות שלו, אפשר להפכו למחרוזת ובעזרת הפונקציה `len` למצוא את אורךו.

תשובה לדוגמה:

```
def leading(a):
 return a//10**(len(str(a))-1)
```

לחילופין, אפשר להמיר את המספר למחרוזת (בעזרת הפונקציה `str`) ולאחר מכן לשולף את תוכנו של האיבר השמאלי ולהפוך אותו חזרה למספר (שכן הפונקציה אמרה להחזיר מספר). למשל:

```
def leading(a):
 b=str(a)
 return int(b[0])
```

19. הפונקציה מגדרה שני משתנים `odd` שנועד לשיכום המספרים אי-זוגיים ו – `even` לשיכום המספרים הזוגיים. בollowאה עוברים על כל איברי הרשימה. כל מספר זוגים מתווסף (ע"י חיבור) למשתנה `even` ואילו כל מספר אי-זוגי מתווסף ל – `odd`. בסוף מתבצעת בדיקה בין שני המשתנים. אם הם שוויים הפונקציה מחזירה "NONE", אם `odd` גדול יותר הפונקציה מחזירה "ODD" ולא "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 מגדרה מחרוזת חדשה `newStr` אליה יכנסו תוצאות פעולות ה – XOR. בollowאה עוברים על כל איברי המחרוזות ומסכימים כל שתי סיביות (לאחר שהפכנו אותן למספרים לפני החיבור). אם תוצאה הסיכון היא 1, משמע שתי הסיביות שונות (אחד 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 ועוד 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. הפעונקציה מתחילה בהגדרת שני משתנים. משתנה אחד נועד לכטובות (אינדקס) של האיבר הקטן ברשימה והמשתנה השני נועד למיקום האיבר הגדול ברשימה. לאחר מכן מטבחצת לולאה שעובדת על כל איברי הרשימה. אם האיבר הנבדק קטן מהאיבר הראשון (או הקודם שנשמר), מיקומו נשמר במשתנה לשמירת מיקום המשטנה הקטן ביותר. ובמקרה 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 האיבר הקטן והאיבר הגדול בהתאם. כל ששאר 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
```

[חזר לתוכן](#)

## תשובות

## רקעosiya

1. הפונקציה הראשונה (רקעosiya) בודקת ראשית את תנאי הסיום. אם  $y$  שווה לאחד, אז היא מחזירה את  $x$  (כלומר  $x$  בחזקת אחד שווה ל- $x$ ). בכלל מצב אחר, היא מחזירה את  $x$  כפול הפונקציה, כאשר הפרמטרים הם  $x$  ו- $y-1$ .  
 הפונקציה השנייה (לולאה) מגדרה משתנה  $z$  מני 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 והצטרכ בסיופה ("") 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 הפעמטר הוא אחד, היא מחזירה אחד ולא היא מחזירה את סכום הפעמטר שהועבר ועוד הפונקציה עם הפעמטר 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 הפעמטר הוא אחד, היא מחזירה את הערך 5. אם לא היא מחזירה את סכום של 3 ועוד ערך הפונקציה עבור האיבר הקודם (הערך שמתקיים מקריאה לפונקציה עם פרמטר הקטן באחד מהפעמטר 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 הפעמטר הוא אפס, היא מחזירה אותו כתשובה. אם לא היא מחזירה את הסכום של המספר (הפעמטר) מודולו 10 (או במילים אחרות היא מחזירה את ערכה של ספרת האחדות) ועוד תשובה הפונקציה עבור פרמטר שהוא המספר מחולק ב – 10 (לא שארית). כולם המחבר השני הוא סכום הספרות במספר המקורי לאחר שהורדה ממנו ספרה 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 הפעמטר הוא אפס, היא מחזירה אותו כתשובה. אם לא, היא מחזירה את הסכום של המספר (הפעמטר) מודולו 2 (או במילים אחרות היא מחזירה את השארית מחולקת המספר ב – 2) ועוד תשובה הפונקציה עבור פרמטר שהוא המספר מחולק ב – 2 (לא שארית), כאשר מספר זה מוכפל פי 10. הסיבה שאנו מכפילים את תוצאה הפונקציה פי 10 היא מפני שאנו רצים ליצור את המספר הבינארי. הפונקציה מתקדמת מימין לשמאל, כאשר בכל קראיה אנחנו מתקדמים לסיבית הימנית יותר.

תשובה לדוגמה:

```
def Dec2Bin(n):
 if n==0:
 return n
 else:
 return n%2+10*Dec2Bin(n//2)
```

## תכנות בPython

9. גם כאן, הפונקציה ראשית בודקת את תנאי הסיום. ובמקרה זה אם הפעמטר הוא רשיימה ריקה, היא מחזירה אף כתשובה. אם לא, היא מחזירה את הסכום של המספר הנמצא באיבר הראשון ברשיימה ועוד תשובה הפונקציה עבור פערמטר שהוא הרשיימה המקורית (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 הפעמטר השני אם הוא אף. אם כן, הפונקציה מחזירה אף. אם איננו מואפס הפונקציה מחזירה את הסכום של הפעמטר הראשון ועוד תוצאה הקרייה לפונקציה עם שני הפעמטרים, אלא שהפעמטר השני קטן באחד. המשמעות של הפונקציה היא שהכפל  $a^*a$  מחושב ע"י  $(a-1)^*a + a$ , כל זמן ש  $a$  עדין לא אף.

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 הפעמטר אם הוא אף. אם כן, הפונקציה מחזירה אף. אם איננו מואפס הפונקציה מחזירה את הסכום של הפעמטר בריבוע ועוד תוצאה הקרייה לפונקציה עם הפעמטר שבוינו קטן באחד מהפעמטר המקורי.

תשובה לדוגמה:

```
def sumSquare(a):
 if a==0:
 return 0
 else:
 return a*a+sumSquare(a-1)
```

12. שאלת זו מתחכמת מעט. עקב העובדה שאיננו יכולים להשתמש בפקודות כפל, יש לחשב את העריכים הריבועיים על סמך הנוסחאות המתמטיות הרגילים. למשל  $1 + 2n - n^2 = (1-n)^2$ . אנחנו רוצים לבטא את  $n^2$  תוך שימוש ב- $(1-n)^2$ . לכן בהעברת ביטויים בין שני אגפי המשוואה, נקבל:  $1 + 2n - (1-n)^2 = n^2$ . עכשו אפשר להגיד את הפונקציה הרקורסיבית. תנאי הסיום במקרה זה יהיה בדיקה של הפעמטר אם הוא אחד. אם כן, הפונקציה מחזירה אחד. אם איננו מואפס הפונקציה מחזירה את הסכום של הפעמטר מוכפל פי 2 פחות אחד ועוד ערך הפונקציה המתתקבל תוך שימוש בפעמטר שבוינו קטן באחד מהפעמטר המקורי.

תשובה לדוגמה:

```
def Square(a):
 if a==1:
 return 1
 else:
 return Square(a-1)+2*a-1
```

## תכנות בסיסי

13. זו דוגמה פשוטה, אלא שתנאי היציאה שלה מורכב משלושה תנאים שונים. אם ערכו של הפעמטר הוא 0, אז הפעקציה מחזירה 3, אם ערכו 1 אז הפעקציה מחזירה 0 ואם ערכו 2, אז הפעקציה מחזירה 2. לכל ערך אחר הפעקציה מחזירה את הסכום של שתי קריאות לפונקציה. אחת עם פרמטר קטן ה – 2 מהפעמטר הנוכחי והשנייה עם פרמטר שהוא קטן ב – 3 מהפעמטר 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 הפעמטר שהועבר מאפס. אם לא הפעקציה מדפיסה מחרוזת בעלת חוכמים וקוראת לפונקציה עם פרמטר הקטן באחד מהמספר הנוכחיים.

תשובה לדוגמה:

```
def stars(a):
 if a==0: return
 else:
 print ("***"a)
 stars(a-1)
```

15. כדי לקבל משולש נסוף (הפוך) יש להוציא פקודות הדפסה זהה לאחר הקריאה לפונקציה. משמעות הדבר שלאחר שהפעקציה 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 (מהוסף להתחלתה). תנאי העצירה הוא אם הפעמטר שהועבר הוא אחד ואז היא מחזירה אחד. בכל מצב אחר היאמחזירה את ח/1 ועוד ערך הפעקציה כאשר הפעמטר שהועבר לה קטן באחד מהפעמטר 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 הפעמטר השלישי (מש' האיבר) שווה לאפס, אז הפונקציה מחזירה את הפעמטר הראשון (האיבר הראשון בסדרה). אם הפעמטר השלישי שווה לאחד, הפונקציה מחזירה את הפעמטר השני (האיבר השני בסדרה) ואילו אם הוא גדול מאחד, אז הפונקציה מחזירה את הסכום של שתי קריאות לפונקציה. קריאה אחת כוללת שתי שני הפעמטרים המקוריים ופעמטר שלישי שהוא קטן באחד מהפעמטר הנוכחי וקריאה שנייה שכוללת את שני הפעמטרים הראשונים ופעמטר שלישי שהוא קטן בשתיים מהפעמטר 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 שכתבים את הפונקציה, צריך לחשב מעט על האלגוריתם. אם הפעמטר שהועבר הוא אפס, אז 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 יכול להיות אפס). אם המספר מודולו 10 שווה אפס (במילים אחרות הוא מחלק בעשר), הפונקציה תחזיר אחד ועוד תוצאה הפונקציה על פורטר שהוא המספר הנוכחי מחלוקת (חלוקת שלמים). אם לעומת זאת, המספר הנוכחי לא מחלק בעשר, הפונקציה תחזיר אפס ועוד תוצאה הפונקציה על פורטר שהוא המספר הנוכחי מחלוקת בעשר. לדוגמה, אם הפעמטר הוא 20020, אז הוא מחלק בעשר ולכן הפונקציה תחזיר 1 ועוד בדיקה של 2002 (שהוא המספר הנוכחי מחלוקת 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 שלנו לבעץ היא האם המספר מחלק בערך שהועבר כפעמטר השני (או במילים אחרות האם הפעמטר השני הוא אכן מחלק של הראשון). אם לא, אז הפונקציה תחזיר אפס. אם כן, הפונקציה תחזיר אחד ועוד תוצאה הפונקציה, כאשר הפעמטר הראשון במקרה זה יהיה המספר המקורי מחלוקת בפעמטר השני.

תשובה לדוגמה:

```
def factor(n,m):
 if n%m !=0 : return 0
 return 1 + factor (n/m,m)
```

20. הבדיקה הראשונה שלנו לבעץ היא האם המערך ריק. אם כן, הפונקציה תחזיר אפס. אם אין ריק, הפונקציה תחזיר אחד ועוד הערך שיתקבל מקריאה נוספת לפונקציה, אלא שעכשו 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:])
```

חזר לתוכן