רשימות בפייתון - Lists in Python

1.א כללי

רשימה – list – משמשת לאחסון מספר פריטים במשתנה יחיד.

הערה : בהמשך הפרק נקרא לפעמים לפריט גם בשם אלמנט או איבר או רכיב.

רשימות הן אחד מתוך 4 סוגי נתונים מובנים ב- Python המשמשים לאחסון אוספי נתונים, 3 האחרים הם Tuple, Set ו Dictionary , שהם בעלי תכונות ושימוש שונים.

רשימות נוצרות באמצעות סוגריים מרובעים: לדוגמה :

```
myColors = ["red", "blue", "green"]
```

אם נבקש הדפסה print(myColors) נקבל : ['red', 'blue', 'green']

1.ב הפריטים ברשימה – List Items

פריטי רשימה מסודרים, ניתנים לשינוי ומאפשרים ערכים כפולים.

פריטי רשימה כלולים באינדקס, לפריט הראשון יש אינדקס [0],לפריט השני יש אינדקס [1] וכו'.

1.ג סדר הרשימה

כאשר אנו אומרים רשימות מסודרות , זה אומר כי לפריטים יש סדר מוגדר, וכי הסדר לא ישתנה.

אם נוסיף פריטים חדשים לרשימה , הפריטים החדשים ימוקמו בסוף הרשימה.

קיימות מספר מתודות שמשנות את הסדר של הרשימה אבל באופן כללי סדר הפריטים ברשימות לא ישתנה.

1.ד שינויים ברשימה

הרשימה ניתנת לשינוי, כלומר אנו יכולים לשנות, להוסיף ולהסיר פריטים ברשימה לאחר יצירתה.

1.ה כפילויות – Duplicates

מאחר שמיקום הפריטים ברשימות נמצא לפי אינדקס, רשימות יכולות לכלול פריטים בעלי ערך זהה (חברים כפולים).
לדוגמה :

```
myColors = ["red", "blue", "green", "red", "green"]
```

אם נבקש הדפסה : print(myColors) נקבל : ['red', 'blue', 'green', 'red', 'green']

1.ו אורך רשימה

כדי לקבוע כמה פריטים יש לרשימה, נשתמש בפונקציה len() . לדוגמה :

```
myColors = ["red", "blue", "green", "red", "green"]

print(len(myColors))
```

נקבל : 5

1.ז הפריטים ברשימה – טיפוסי הנתונים

פריטי הרשימה יכולים להיות מכל סוג נתונים. לדוגמה נראה 3 רשימות מטיפוס מחרוזת, שלם ובוליאני.

```
list1 = ["red", "blue", "green"]
list2 = [1, 5, 7, 9, 3]
list3 = [True, False, False]
```

אם נבקש הדפסות :

```
print(list1)
print(list2)
print(list3)
```

נקבל :

```
['red', 'blue', 'green']
[1, 5, 7, 9, 3]
[True, False, True]
```

רשימה יכולה להכיל טיפוסי נתונים שונים. לדוגמה רשימה עם טיפוסי מחרוזת, מספרים שלמים וערכים בוליאניים.

```
list4 = ["red", 29, True, -40, "male"]
```

1.ח שימוש ב type( )

מנקודת המבט של פייתון, רשימות מוגדרות כאובייקטים עם סוג הנתונים 'list' . לדוגמה :

```
list4 = ["red", 29, True, -40, "male"]
 print(type(list4))
```

נקבל : < class 'list' >

1.ט הבנאי list() - list constructor

ניתן גם להשתמש בבנאי list( ) בעת יצירת רשימה חדשה. לדוגמה :

```
list4 = list (("red", 29, True, -40, "male"))        # פעמיים סוגריים קטנים
```

אם נרצה לקבל הדפסה נרשום : print(list4)

ונקבל : ['red', 29, True, -40, 'male']

1.י אוספים (מערכים) - collections

קיימים ארבעה סוגי נתונים של אוסף בשפת פייתון והם :

*. **list** - הוא אוסף מסודר וניתן לשינוי. יש אפשרות לחברים כפולים.

*. **tuple** הוא אוסף אשר מסודר ובלתי ניתן לשינוי. יש אפשרות לחברים כפולים.

*. **set** הוא אוסף שאינו מסודר וללא אינדקס. אין חברים כפולים.

*. **dictionary** הוא אוסף אשר מסודר וניתן לשינוי. אין חברים כפולים.

במילה מסודר הכוונה שלכל פריט יש את האינדקס שלו.

החל מגרסה 3.7 של פייתון dictionaries הם מסודרים . בפייתון 3.6 וגרסאות קודמות הם לא היו מסודרים.

כאשר בוחרים טיפוס של אוסף כדאי להבין את המאפיינים של טיפוס זה. בחירת הטיפוס הנכון עבור ערכת נתונים מסוימת עשויה להיות שמירה על המשמעות של הפריטים, וזה יכול לגרום לעלייה ביעילות או באבטחה.

2. גישה לפריטים ברשימה - Access List Items

2.א גישה בעזרת אינדקס

לכל פריט ברשימה יש אינדקס וניגשים אליו על ידי הפנייה למספר האינדקס שלו. הפריט הראשון הוא באינדקס [0] .

דוגמה: הדפסת פריט כלשהו מהרשימה.

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
```

הדפסת הפריט ה 5 ברשימה : print (colors[5])

ההדפסה שנקבל היא : white .

2.ב אינדקס שלילי

אינדקס שלילי פירושו להתחיל את הספירה מהפריט האחרון . הפריט האחרון הוא באינדקס [-1] זה שלפניו באינדקס [-2] וכך הלאה . בדוגמה של הרשימה colors הצבע navy נמצא באינדקס [-1] , הצבע white באינדקס [-2] וכך הלאה עד הצבע red שנמצא ב [-7] .

דוגמה : נדפיס את האיבר במיקום 2- ברשימה colors :

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
print(colors[-2])
```

ההדפסה שנקבל היא : white .

2.ג טווח של אינדקסים

יש אפשרות לציין טווח אינדקסים על-ידי ציון היכן להתחיל והיכן לסיים את הטווח. בעת ציון הטווח, הערך המוחזר יהיה רשימה חדשה עם הפריטים שצוינו. הטווח נרשם כך : [1:5] ואז מדובר על הפריטים 1 עד 5 (אבל לא כולל את הפריט 5 !).

ניתן לרשום את התחום [:5] ללא ציון הפריט הראשון ואז הכוונה מהפריט באינדקס [0] ועד האינדקס [5] לא כולל הפריט באינדקס 5 !!

ניתן לרשום גם : [2:] ללא ציון הפריט הסופי ואז התחום הוא מהפריט במיקום [2] ועד סוף הרשימה .

דוגמאות

דוגמה 1: יש להדפיס את הפריטים מ 1 עד 5 ברשימה colors .

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
print(colors[1:5])
```

ההדפסה שנקבל היא : ['blue', 'green', 'magenta', 'black']

אם נבקש את ההדפסה הבאה :

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
print(colors[-1:-5])
```

נקבל רשימה ריקה [ ] .

לעומת זאת אם נבקש :

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
print(colors[-5:-1])
```

נקבל את ההדפסה מהפריט ב [-5] שהוא green ועד הפריט [-2] שהוא white (לא מקבלים את [-1] ! :

ההדפסה שנקבל היא : ['green', 'magenta', 'black', 'white']

דוגמה 2: הדפסת הרשימה מהפריט 0 ועד 4 לא כולל הפריט הרביעי .

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
print(colors[:4])
```

ההדפסה שנקבל היא :

['red', 'blue', 'green', 'magenta']

דוגמה 3: הדפסה מפריט במיקום 2 עד סוף הרשימה.

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
print(colors[2:])
```

ההדפסה שנקבל היא :

```
[ 'green', 'magenta', 'black', 'white', 'navy']
```

2.ד בדיקה האם פריט נמצא ברשימה

אם רוצים לבדוק האם פריט מסוים נמצא ברשימה משתמשים במילת המפתח **in** .

דוגמה 1: נבדוק האם הפריט blue נמצא ברשימה colors

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
color="blue"
if color in colors :
  print (color + " is in the list colors")
else :
  print (color + " is not in the list colors")
```

ההדפסה שנקבל : blue is in the list colors

דוגמה 2: נבדוק האם הפריט Blue נמצא ברשימה colors (שמנו B במקום b)

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
color="blue"
if color in colors :
  print (color + " is in the list colors")
else :
  print (color + " is not in the list colors")
```

ההדפסה שנקבל : blue is not in the list colors

3. שינוי ערך של פריטים

3.א שינוי ערך של פריט

כדי לשנות את הערך של פריט מסוים, מתייחסים למספר האינדקס שלו ברשימה:

דוגמה :

```
myColors = ["red", "blue", "green"]
myColors[2]="brown"
print(myColors)
```

ההדפסה שנקבל היא : ['red', 'blue', 'brown']

3.ב שינוי ערכים של מספר פריטים בתחום רצוי.

כדי לשנות את הערך של פריטים בטווח מסוים מגדירים טווח של מספרי אינדקס ורשימה עם הערכים החדשים שייכנסו לטווח מספרי האינדקסים שרוצים להוסיף בו את הערכים החדשים.

דוגמה : נשנה את האיברים 2 עד 5 (לא כולל הפריט 5) ברשימה colors .

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
colors[2:5] = ["orange", "yellow", "purple"]
print(colors)
```

ההדפסה שנקבל היא : ['red', 'blue', 'orange', 'yellow', 'purple', 'white', 'navy'] .

1.ב.3 אם נוסיף יותר פריטים ממה שביקשנו להחליף, הפריטים החדשים יתווספו למקום שציינו והפריטים שבהמשך הרשימה יעברו בהתאם. כמובן , שבמקרה כזה כמות הפריטים ברשימה תגדל.

דוגמה : נבקש לשנות את הערכים של הפריטים 2 עד 5 (לא כולל 5) – סה"כ להחליף 3 פריטים - אבל נרשום 5 צבעים במקום 3 . כל 5 הצבעים ייכנסו לרשימה, האיברים שביקשנו להחליף – יוחלפו ושאר האיברים יזוזו ויהיו אחרי הצבעים שהוספנו.

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
print(len(colors))                                  # להדפיס את כמות הפריטים ברשימה
colors[2:5] = ["orange", "yellow", "purple", "pink", "pale blue"]
print(colors)                                       # להדפיס את כמות הפריטים ברשימה החדשה
print(len(colors))
```

ההדפסות שנקבל :

```
7
['red', 'blue', 'orange', 'yellow', 'purple', 'pink', 'pale blue', 'white', 'navy']
9
```

2.ב.3 אם נוסיף פחות פריטים ממה שביקשנו להחליף , הפריטים החדשים יתווספו למקום שציינו והפריטים הנותרים יעברו בהתאם.

דוגמה : נבקש לשנות ערכים של 4 פריטים אבל נרשום רק 2 ערכים חדשים. 2 הערכים החדשים ייכנסו במיקום שביקשנו, 2 הפריטים הבאים ברשימה יימחקו מהרשימה ואורך הרשימה יקטן ב 2 .

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
```

```
print(len(colors))
colors[2:6] = ["orange", "yellow"]
print(colors)
print(len(colors))
```

ההדפסה שנקבל היא :

```
7
['red', 'blue', 'orange', 'yellow', 'navy']
5
```

4. הוספה של פריטים

4.1 הוספת פריט – שימוש במתודה ( )insert

כדי להוסיף פריט רשימה חדש מבלי להחליף אף אחד מהערכים הקיימים, באפשרותנו להשתמש במתודה **( )insert** .

המתודה **( )insert** מוסיפה פריט באינדקס שצוין.

דוגמה : הכנסת פריט נוסף לרשימת ה colors במיקום [3] (הפריט הרביעי). כמובן שאורך הרשימה גדל ב 1 .

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
colors.insert(3, "bourdeaux")                    # הוספת הצבע בורדו באינדקס 3 - פריט רביעי
print(colors)
```

ההדפסה שנקבל : ['red', 'blue', 'green', 'bourdeaux', 'magenta', 'black', 'white', 'navy']

4.ב צירוף של פריט בסוף הרשימה עם המתודה ()append

כדי להוסיף פריט לסוף הרשימה, משתמשים במתודה **( )append** :

דוגמה: נוסיף את הצבע כתום orange בסוף רשימת הצבעים colors .

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
colors.append("orange")
print(colors)
```

ההדפסה שנקבל היא : ['red', 'blue', 'green', 'magenta', 'black', 'white', 'navy', 'orange']

4.ג הרחבת רשימה עם המתודה ()extend

כדי לצרף רכיבים מרשימה אחרת לרשימה הנוכחית נשתמש במתודה **( )extend** . הרכיבים החדשים מצורפים בסוף הרשימה.

דוגמה: נגדיר 2 רשימות. בשלב ראשון נצרף את הרשימה השנייה בסוף הרשימה הראשונה. בשלב הבא נצרף את הרשימה הראשונה (המורכבת מהרשימה הראשונה ועוד הרשימה השנייה) בסוף הרשימה השנייה. התוכנית נראית כך:

```
colors1 = ["red", "blue", "green"]
colors2 = ["magenta", "black",  "white", "navy"]
colors1.extend(colors2)
print("New colors1 =  " , colors1)
colors2.extend(colors1)
print("New colors2 =  " ,colors2)
```

ההדפסות שנקבל הן :

```
New colors1 =   ['red', 'blue', 'green', 'magenta', 'black', 'white', 'navy']
New colors2 =   ['magenta', 'black', 'white', 'navy', 'red', 'blue', 'green', 'magenta', 'black', 'white', 'navy']
```

4.ד הוספה של כל משתנה/נתון/אוסף אל רשימה עם המתודה ( )extend

המתודה ( )extend לא חייבת לצרף רשימות . ניתן לצרף גם tuples, sets, dictionaries ועוד.

דוגמה : הוספת tuple לרשימה :

```
colors1 = ["red", "blue", "green"]
colors2 = ("magenta", "black",  "white", "navy")          # colors2 is a tuple and not a list
colors1.extend(colors2)
print("New colors1 =  " , colors1)
```

ההדפסה שנקבל היא : `New colors1 =   ['red', 'blue', 'green', 'magenta', 'black', 'white', 'navy']`

5. הסרת פריטים מרשימה

5.א הסרה של פריט מסוים עם המתודה ( )remove

המתודה **( )remove** מסירה פריט מסוים שנבחר.

דוגמה : נסיר פריט מסוים מתוך רשימה.

```
colors1 = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
colors1.remove("red")
print("colors1 =  " , colors1)
```

ההדפסה שנקבל : `colors1 =   ['blue', 'green', 'magenta', 'black', 'white', 'navy']`

אם נרשום עכשיו :

```
colors1.remove("magenta")
print("colors1 =  " , colors1)
```

נקבל : colors1 = ['blue', 'green', 'black', 'white', 'navy']

5.ב הסרה של אינדקס מסוים עם המתודה ( )pop

המתודה ( )pop מסירה פריט מסוים על פי האינדקס שנרשום.

דוגמה : הסרה של הפריט השני (שהאינדקס שלו 1) .

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
colors.pop(1)
print(colors)
```

נקבל : ['red', 'green', 'magenta', 'black', 'white', 'navy']

אם לא נציין מספר אינדקס אז המתודה ( )pop תסיר את הפריט האחרון .

דוגמה :

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
colors.pop()
print(colors)
```

נקבל : ['red', 'blue', 'green', 'magenta', 'black', 'white']

5.ג מחיקה של פריט ברשימה עם המילה del

גם מילת המפתח del מסירה פריט לפי האינדקס שנרשום.

דוגמה : הסרה של האיבר השלישי (האיבר באינדקס 2) :

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
del colors[2]
print(colors)
```

נקבל : ['red', 'blue', 'magenta', 'black', 'white', 'navy']

5.ד מחיקה של רשימה עם המילה ( )del

המתודה **( )del** מוחקת את הרשימה עצמה (ולא רק את הפריטים שבה).

דוגמה :

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
del colors
```

```
print(colors)
```

נקבל :

```
Traceback (most recent call last):
  File "./prog.py", line 3, in <module>
NameError: name 'colors' is not defined
```

קיבלנו הודעת שגיאה בשורה 3 שהמילה colors לא הוגדרה. ואכן , בשורה השנייה , מחקנו את הרשימה colors ובשורה השלישית מבקשים הדפסה של הרשימה שנמחקה וכבר לא קיימת ולכן קיבלנו הודעת שגיאה.

5.ה ניקוי של כל הרשימה עם המתודה clear

המתודה clear( ) מוחקת את כל הפריטים ברשימה (אבל לא את הרשימה עצמה שנשארת ריקה).

דוגמה :

```
colors = ["red", "blue", "green", "magenta", "black",  "white", "navy"]
colors.clear()
print(colors)
```

נקבל : עכשיו לא מקבלים הודעת שגיאה כמו במילה del אלא רשימה ריקה : [ ] .

6. לולאות בתוך רשימה

6.א לולאת for בתוך רשימה

אפשר לעבור בלולאה בין הפריטים של רשימה באמצעות לולאת **for** .

דוגמה :

```
colors = ["red", "blue", "green", "black",  "white"]
for x in colors :
  print(x)
```

ההדפסה שנקבל:

```
red
blue
green
black
white
```

6.ב לולאת for בתוך רשימה עם אינדקס

אפשר גם לעבור בלולאת for בין פריטי הרשימה על-ידי הפניה למספר האינדקס שלהם. לשם כך ניעזר בפונקציות **range( )** ו **len( )**

דוגמה 1: הדפסת כל הפריטים על ידי התייחסות למספר האינדקס שלהם.

```
colors = ["red", "blue", "green", "black",  "white"]
for x in range(len(colors)):
   print(colors[x])
```

ונקבל את כל הפריטם מאינדקס 0 ועד האינדקס שהוא אורך הרשימה (האורך הוא 5 פריטים) אבל ההדפסה לא כוללת 5 אלא רק את הפריטים 0,1,2,3,4 :

```
red
blue
green
black
white
```

דוגמה 2 : הדפסת אורך הרשימה – כמות הפריטים ברשימה - שימוש בפונקציה **len( )** .

```
colors = ["red", "blue", "green", "black",  "white"]
print(len(colors) )
```

נקבל את המספר : 5

על לולאות **for** נרחיב בפרק על לולאות בפייתון.

6.ג לולאת while בתוך רשימה

דוגמה 1 : הדפסה של כל הפריטים של רשימה :

אפשר לעבור בלולאה בין הפריטים של רשימה באמצעות לולאת **while** .לשם כך ניעזר בפונקציה **len( )** שבדוגמה הקודמת כדי לקבוע את אורך הרשימה . נתחיל מאינדקס 0 לעבור על כל הפריטים ברשימה בעזרת האינדקסים שלהם. יש לזכור להגדיל ב 1 את האינדקס בסיום כל איטרציה (לולאה של פקודות תוכנה) . בלולאת for ההגדלה מתבצעת אוטומטית !

```
colors = ["red", "blue", "green", "black",  "white"]
x=0
while x <  len(colors) :
  print(colors[x])
  x=x+1                    # x  את 1 לא לשכוח להגדיל ב
```

וההדפסה שנקבל :

```
red
blue
green
black
white
```

שאלה : מה קורה אם שוכחים לרשום את השורה : x=x+1 ?

תשובה : התוכנית תרוץ בלולאה אין סופית ובמסך יודפס כל הזמן red

על לולאות **while** נרחיב בפרק על לולאות בפייתון.

7. לולאות עם תחביר מקוצר - Looping Using List Comprehension

הבנה מעמיקה של רשימה מאפשרת לרשום פקודות עם תחביר קצר ביותר עבור ביצוע של לולאה בתוך רשימות.

7.א דוגמה : פקודה מקוצרת להדפסת כל הפריטים ברשימה :

```
colors = ["red", "blue", "green", "black",  "white"]
[print (x) for x in colors]
```

הפקודה השנייה כוללת את 2 הפקודות בלולאת ה for שבפסקה קודמת :

```
for x in colors :
  print(x)
```

7.ב : העתקה של פריטים מרשימה אחת שיש בהם את התו 'e' לרשימה חדשה והדפסתה

נראה 2 דרכים . האחת בדרך "הרגילה" והשנייה עם משפט מקוצר. כמובן שהרשימה המקורית איננה משתנה.

בדרך הרגילה : ללא הבנה של רשימה נכתוב משפט אם תנאי :

```
colors = ["red", "blue", "green", "black",  "white"]
newColors = [ ]
for x in colors:
 if "e" in x:
   newColors.append(x)
print(newColors)
```

וההדפסה שנקבל : ['red', 'blue', 'green', 'white']

בדרך המקוצרת :

```
colors = ["red", "blue", "green", "black",  "white"]
newColors = [x for x in colors if "e" in x]
print(newColors)
```

ונקבל : ['red', 'blue', 'green', 'white']

ניעזר בדוגמה הקודמת כדי להבין את התחביר במשפט מקוצר :

```
newColors = [x for x in colors if "e" in x]
```

באופן כללי המשפט המקוצר בנוי כך :

[**שם הרשימה החדשה** = [expression **for** item **in** iterable **if** תנאי == True]

הערך המוחזר הוא רשימה חדשה והרשימה הישנה נשארת ללא שינוי.

התנאי דומה למסנן המקבל רק את הפריטים שמעריכים ל- True. אפשר להזניח את התנאי ואז זו ההעתקה של כל הרשימה.

iterable יכול להיות כל אובייקט שניתן לעבור עליו באיטרציה - בלולאה - כמו רשימה , tuple , set וכו' .

במקום **שם הרשימה החדשה** ניתן לרשום print (…) ואז נקבל הדפסה ללא יצירה של רשימה חדשה .

Expression - ביטוי - הוא הפריט הנוכחי באיטרציה אך הוא גם התוצאה שאפשר לטפל בה לפני שהיא מסתיימת כפריט רשימה ברשימה החדשה .

ג.7 העתקה והדפסה של כל הפריטים שבהם לא מופיע התו 'e'

```
colors = ["red", "blue", "green", "black",  "white"]
newColors = [x for x in colors if "e" not in x]
print(newColors)
```

ההדפסה שנקבל : ['black']

ד.7 הדפסת הפריטים שבהם מופיעים התווים "re" :

```
colors = ["red", "blue", "green", "black",  "white"]
print ([x for x in colors if "re" in x])
```

ההדפסה שנקבל : ['red', 'green']

7.ה העתקת כל הפריטים עם הציון 100 והפריטים שבהם הערך קטן מ 55 (כמות ציונים שליליים)

```
marks = [90,70,54,55,67,89,100,94,50,48,67,100,49,100,65]
marks100 = [x for x in marks if x == 100 ]
negativeMarks = [x for x in marks if x < 55 ]
print ("marks100 =  ", marks100, "and" , len(marks100) , "students has this mark")
print("negativeMarks = ", negativeMarks , "and" , len(negativeMarks) , "students has negative marks" )
```

ההדפסות שנקבל :

```
marks100 =   [100, 100, 100] and 3 students has this mark
negativeMarks =  [54, 50, 48, 49] and 4 students has negative marks
```

7.ו שימוש בפונקציה range (טווח)

אפשר להשתמש בפונקציה range( ) כדי ליצור אובייקט שניתן לעבור עליו בלולאה.

דוגמה : נדפיס את כל הפריטים ברשימה colors בטווח מפריט 0 ועד פריט 3 (לא כולל 3).

```
colors = ["red", "blue", "green", "black",  "white"]
print ([colors[x] for x in range(3)])
```

ההדפסה שנקבל : ['red', 'blue', 'green']

7.ז הדפסת הפריטים בטווח רצוי עם תנאי

ניתן להדפיס את הפריטים בטווח רצוי עם תנאי.

דוגמה : נדפיס את כל הציונים החיוביים של 10 הסטודנטים הראשונים .

```
marks = [90,70,54,55,67,89,100,94,50,48,67,100,49,100,65]
print ([marks[x] for x in range(10) if marks[x] >= 55])
```

ההדפסה שנקבל : [90, 70, 55, 67, 89, 100, 94]

7.ח הדפסת הפריטים ברשימה עם תווים בדפוס (אותיות "גדולות")

```
colors = ["red", "blue", "green", "black",  "white"]
print ([x.upper() for x in colors])                    # המתודה upper הופכת כל תו "קטן" לתו "גדול"
```

ההדפסה שנקבל : ['RED', 'BLUE', 'GREEN', 'BLACK', 'WHITE']

7.ט שינוי הערך של כל הפריטים ברשימה ל nice

```
colors = ["red", "blue", "green", "black",  "white"]
newColors = ['nice' for x  in colors]
```

```
print(newColors)
```

ההדפסה שנקבל : ['nice', 'nice', 'nice', 'nice', 'nice']

7.י שינוי הערך של פריט רצוי ברשימה לפי תנאי

ניתן לשנות את הערך של פריט מסוים מתוך רשימה.

דוגמה : שינוי הערך "green" ברשימה ל "red"

```
colors = ["red", "blue", "green", "black",  "white"]
newColors = [x if x != "green" else "red" for x in colors1]
print(newColors)
```

בשורה השנייה רשמנו להשאיר את הפריט כמו שהוא אם הוא לא "green" ואם הוא "green" ואז יש לשנות אותו ל "red" .

ההדפסה שנקבל: ['red', 'blue', 'red', 'black', 'white']

אם יש מספר פריטים בשם "green" אז כולם ישתנו ל "red" כמו בדוגמה הבאה :

```
colors1 = ["red", "blue", "green", "black",  "white", "green", "navy"]
newColors = [x if x != "green" else "red" for x in colors1]
print(newColors)
```

נקבל : ['red', 'blue', 'red', 'black', 'white', 'red', 'navy']

8. מיון רשימות Sort lists

לאובייקטים של רשימה יש מתודה הנקראת sort( ) שתבצע מיון של הרשימה לפי קריטריון שנבקש.

8.1 מיון אלפאנומרי של רשימה בסדר עולה - ascending sort

במילה אלפאנומרי הכוונה לתווים מ a ועד z , לשאר תווי הבקרה כמו # או $ וכמו כן הספרות 0 עד 9 . לכל תו יש ערך אסקי (או Unicode משלו) כאשר התו a (ערך בינארי 0x61) והתו A (ערך בינארי 0x41) אינם זהים.

המתודה ( )sort היא case sensitive והמיון בסדר עולה מתבצע התווים הגדולים - Capital letters - לפני הקטנים – lower case.

דוגמה : מיון אלפאנומרי של רשימה בסדר עולה .

```
colors = ["red", "blue", "green", "black",  "white"]
colors.sort( )
print(colors)
```

בשורה השנייה רשמנו colors.sort( ) ובתוך הסוגריים לא רשמנו כלום. ברירת המחדל היא סדר עולה.

נקבל : ['black', 'blue', 'green', 'red', 'white']

דוגמה נוספת עם מספרים:

```
numbers = [100, 55, -20, -40,  1000, 37]
numbers.sort( )
print(numbers)
```

ונקבל : [-40, -20, 37, 55, 100, 1000]

אפשר גם למיין רשימה בעברית רק צריך לזכור שבהדפסה

```
colors = ["שחור" ,"ירוק", "כחול", "אדום",  "white"]
colors.sort( )
[print (x) for x in colors]
```

ונקבל :

white
אדום
ירוק
כחול
שחור

אם נבקש בשורה השלישית הדפסה על ידי print(colors) נקבל : ['אדום', 'ירוק', 'כחול', 'שחור', 'white'] .

המיון מתבצע נכון אבל בגלל ההדפסה בעברית נרשם קודם שחור ובסוף אדום.

8.2 מיון אלפאנומרי של רשימה בסדר יורד - descending sort

כדי לבצע מיון בסדר יורד נשתמש בארגומנט reverse = True .

דוגמה : מיון יורד של 2 רשימות : רשימה של מחרוזות ורשימה נוספת של מספרים

```
colors = ["red", "blue", "green", "black",  "white"]
numbers = [100, 55, -20, -40,  1000, 37]
colors.sort(reverse = True)
numbers.sort(reverse = True)
print (colors)
print(numbers)
```

ההדפסה שנקבל :

```
['white', 'red', 'green', 'blue', 'black']
[1000, 100, 55, 37, -20, -40]
```

8.3 מיון בהתאמה אישית customize sort function

ניתן להתאים אישית את הפונקציה שלך באמצעות הארגומנט key = function . הפונקציה תחזיר מספר שישמש למיון הרשימה (המספר הנמוך ביותר תחילה).

ניתן להשתמש בפונקציות מובנות כפונקציות מפתח בעת מיון רשימה.

דוגמה : מיון רשימה בהתייחס לכמה קרוב המספר ל 100 :

במיון הרשימה נקרא לפונקציה abs שתחזיר את הערך המוחלט של ההפרש בין מספר ששולחים לרשימה והערך 100 .
ההפרש ישמש למיון הרשימה (המספר הנמוך ביותר תחילה):

```
def func1(n):    # הגדרת פונקציה המחזירה את הערך המוחלט של ההפרש בין המספר שהיא מקבלת פחות 100
  return abs(n - 100)

numbers = [100, 55, -20, -40,  1000, 37]
numbers.sort(key = func1)
print(numbers)
```

ההדפסה שנקבל : [100, 55, 37, -20, -40, 1000]

8.4 מיון ללא התחשבות ב case sensitive (לא משנה האם התו באותיות קטנות או גדולות).

אם רוצים לבצע מיון ללא התחשבות האם התו באותיות קטנות או גדולות נשתמש ב- str.lower כפונקציית מפתח ואז אין התחשבות באם התו גדול או קטן.

דוגמה : ביצוע מיון ללא התחשבות ב case sensitive

```
newColors = ["Red", "blue", "green", "Black",  "white"]
newColors.sort()
print("newColors after normal sort : " , newColors)
newColors = ["Red", "blue", "green", "Black",  "white"]
newColors.sort(key = str.lower)
print("newColors without case sensitive sort : " , newColors)
```

ההדפסה שנקבל :

```
newColors after normal sort :  ['Black', 'Red', 'blue', 'green', 'white']
newColors without case sensitive sort :  ['Black', 'blue', 'green', 'Red', 'white']
```

דוגמה נוספת : שימשו במילת המפתח key=str.upper

```
newColors = ["Red", "blue", "green", "Black",  "white"]
newColors.sort()
print("newColors after normal sort : " , newColors)
newColors = ["Red", "blue", "green", "Black",  "white"]
newColors.sort(key = str.upper)
print("newColors without case sensitive sort : " , newColors)
```

ההדפסה שנקבל :

```
newColors after normal sort :  ['Black', 'Red', 'blue', 'green', 'white']
newColors without case sensitive sort :  ['Black', 'blue', 'green', 'Red', 'white']
```

8.5 מיון הפוך reverse order .

ניתן להפוך את הסדר של רשימה כך שהפריט הראשון יהיה האחרון והאחרון יהיה הראשון ובהתאמה שאר הפריטים ללא קשר לאלפבית נשתמש במתודה ( )reverse.

דוגמה: נהפוך את סדר הפריטים ברשימה colors

```
colors = ["red", "blue", "green", "black",  "white"]
colors.reverse()
print(colors)
```

ההדפסה שנקבל : ['white', 'black', 'green', 'blue', 'red']

9. העתקה של רשימות

לא ניתן להעתיק רשימה אחת לאחרת על ידי הפקודה : list2=list1 . הסיבה לכך היא שהכתובות בזיכרון של list1 יהיה גם אותן הכתובות של list2 ושינויים שנעשה באחת הרשימות יתבצעו גם ברשימה השנייה כי מדובר על אותן כתובות בזיכרון.

אחת הדרכים להעתיק רשימה אחת לאחרת היא בעזרת המתודה **copy( )** .

9.1 : דוגמה להעתקת רשימה עם המתודה copy( )

```
colors1 = ["red", "green", "blue"]
colors2 = colors1.copy()
print(colors2)
```

ההדפסה שנקבל : ['red', 'green', 'blue']

9.2 : דוגמה להעתקת רשימה עם המתודה ( )list

```
colors1 = ["red", "green", "blue"]
colors2 = list(colors1)
print(colors2)
```

ההדפסה שנקבל גם כאן היא : ['red', 'green', 'blue']

10. צירוף של רשימות

ישנן מספר דרכים לצרף או לשרשר שתי רשימות או יותר בפייתון. אחת הדרכים הקלות ביותר היא באמצעות האופרטור + .

10.1 צירוף רשימה בעזרת האופרטור +

```
colors1 = ["red", "green", "blue"]
colors2 = ["magenta", "white", "black", "orange"]
colors3 = colors1+colors2
print(colors3)
```

ההדפסה שנקבל : ['red', 'green', 'blue', 'magenta', 'white', 'black', 'orange']

המחרוזת colors2 שורשרה בסיום color1 .

אם נשנה את סדר השרשור ונרשום את התוכנית :

```
colors1 = ["red", "green", "blue"]
colors2 = ["magenta", "white", "black", "orange"]
colors3 = colors2+colors1
print(colors3)
```

ההדפסה שנקבל : ['magenta', 'white', 'black', 'orange', 'red', 'green', 'blue']

המחרוזת color1 שורשרה בסיום color2 .

10.2 צירוף רשימה בעזרת צירוף הפריטים מהרשימה השנייה אחד אחרי השני לרשימה הראשונה

דרך נוספת לצרף שתי רשימות היא על ידי צירוף כל הפריטים מהרשימה השנייה לרשימה הראשונה בזה אחר זה .

דוגמה :

```
colors1 = ["red", "green", "blue"]
colors2 = ["magenta", "white", "black", "orange"]
for x in colors2 :
   colors1.append(x)
```

```
print(colors1)
```

ההדפסה שנקבל : ['red', 'green', 'blue', 'magenta', 'white', 'black', 'orange']

10.3 צירוף רשימה בעזרת המתודה ( )extend

דוגמה להוספת רשימה שנייה בסיום הרשימה הראשונה עם המתודה ( )extend :

```
colors1 = ["red", "green", "blue"]
colors2 = ["magenta", "white", "black", "orange"]
colors1.extend(colors2)
print(colors1)
```

ההדפסה שנקבל : ['red', 'green', 'blue', 'magenta', 'white', 'black', 'orange']

11. מתודות של רשימה

לפייתון יש מתודות שניתן להשתמש בהן העבודה עם רשימות. הטבלה הבאה מתארת את רשימת המתודות של רשימה.

המתודה	תפקיד
append( )	מוסיפה פריט בסוף רשימה
clear( )	מנקה את הרשימה מכל הפריטים
copy( )	מחזירה עותק של הרשימה
count( )	מחזירה את מספר הפריטים עם הערך שנשלח אליה
extend( )	הוספה של פריט (או רשימה) בסוף הרשימה הנוכחית.
index( )	מחזירה את האינדקס של הפריט הראשון עם הערך שנשלח אליה
insert( )	מוסיפה פריט במיקום שצוין
pop( )	מסירה/מוחקת פריט במיקום שצוין
remove( )	מסירה פריט עם הערך שנשלח לה
reverse( )	הופכת את סדר הרשימה
sort( )	מיון הרשימה

טבלה 1 : רשימת המתודות של רשימה

11.א מתודת append( )

המתודה מוסיפה פריט בסיום רשימה.

התחביר :

list.append(elmnt)

כאשר **elmnt** הוא מחרוזת, מספר, רשימה .

דוגמה 1 : הוספת פריט בסיום רשימה

```
colors = ["red", "green", "blue"]
colors.append("white")
print(colors)
```

ההדפסה שנקבל : ['red', 'green', 'blue', 'white']

דוגמה 2 : הוספת רשימה בסיום של רשימה :

```
colors1 = ["red", "green", "blue"]
colors2 = ["white", "black"]
colors1.append(colors2)
print("colors1 =  ", colors1)
colors2.append(colors1)
print("colors2 =  ", colors2)
```

ההדפסה שנקבל :

```
colors1 =   ['red', 'green', 'blue', ['white', 'black']]
colors2 =   ['white', 'black', ['red', 'green', 'blue', [...]]]
```

11.ב מתודת ( )clear

המתודה מסירה את כל הפריטים מרשימה.

התחביר :

list.clear( )

לא שולחים פרמטר אל הפונקציה (לא רושמים בסוגריים כלום)

דוגמה :

```
colors = ["red", "green", "blue"]
colors.clear()
print(colors)
```

ההדפסה שנקבל : [ ] .

11.ג מתודת ( )copy

המתודה מחזירה עותק של הרשימה שמצוינת בפקודה.

התחביר :

list.copy( )

לא שולחים פרמטר אל הפונקציה (לא רושמים בסוגריים כלום)

דוגמה : העתקה של רשימה אחת לרשימה אחרת

```
colors1 = ["red", "green", "blue"]
colors2=colors1.copy()
print(colors2)
```

ההדפסה שנקבל : ['red', 'green', 'blue']

11.ד מתודת ( )count

המתודה מחזירה את מספר הפריטים עם הערך ששולחים לה.

התחביר:

list.count(value)

value הוא הערך ששולחים ורוצים לדעת כמה פעמים הוא מופיע ברשימה. הוא יכול להיות מסוג של מספר, מחרוזת, רשימה, tuple וכו' .

דוגמה 1: נספור כמה פעמים מופיע הפריט green הרשימה.

```
colors = [ "red", "blue", "green", "red", "green", "white","green"]
colors.count("green")
```

ההדפסה שנקבל : 3

דוגמה 2 : נספוק כמה פעמים מופיע הערך 100 ברשימה הבאה .

```
marks = [90,70,54,55,67,89,100,94,50,48,67,100,49,100,65]
print("100 appears : ", marks.count(100), "times ")
```

ההדפסה שנקבל : 100 appears : 3 times

11.ה מתודת ( )extend

המתודה מוסיפה את הפריטים ברשימה שבסוגריים (מחרוזת, רשימה, tuple , set וכו') לסוף הרשימה הנוכחית.

התחביר :

list.extend(iterable)

iterable הוא רשימה , tuple , set וכו'.

דוגמה 1 : הוספה של פריטים מהרשימה marks בסוף הרשימה colors :

```
colors = ["red", "green", "blue"]
numbers = [90,70,54,55,67,89]
colors.extend(numbers)
print(colors)
```

ההדפסה שנקבל : ['red', 'green', 'blue', 90, 70, 54, 55, 67, 89]

דוגמה 2 : הוספה של ה tuple points לרשימה colors

```
colors = ["red", "green", "blue"]
points = [90,70,54,55,67,89]
colors.extend(points)
print(colors)
```

ההדפסה שנקבל : ['red', 'green', 'blue', 90, 70, 54, 55, 67, 89]

11.ו מתודת ( )index

המתודה מחזירה את המיקום ברשימה של הפריט ששלחנו אליה. היא מחזירה רק את המיקום של הפריט הראשון של הערך .

התחביר :

list.index(elmnt)

elmnt הוא כל סוג של מחרוזת, מספר, רשימה וכו' .

דוגמה 1 : נמצא את מיקום "green" ברשימה colors .

```
colors = ["red", "green", "blue","white","black", "green", "gray"]
x= colors.index("green")
print(x)
```

ההדפסה שנקבל היא 1 כי ברשימה מופיע "red" פעמיים , גם במיקום 1 וגם במיקום 5 . ההדפסה שנקבל היא 1 כי זהו המיקום הראשון שבו הופיע .

אם נבקש פריט שאיננו ברשימה נקבל הודעה שהפריט איננו ברשימה לדוגמה נבקש "gereen" במקום green ונקבל :

```
ValueError: 'gereen' is not in list
```

11.ז מתודת ( )insert

המתודה מוסיפה את הערך שצוין במיקום שצוין.

תחביר :

list.insert(*pos, elmnt*)

pos הוא מספר המציין באיזה מיקום להוסיף את הפריט הרצוי.

elmnt הוא הפריט שרוצים להוסיף. הוא יכול להיות מכל סוג כמו מחרוזת, מספר אובייקט וכו'.

דוגמה : הכנסת צבע gray במיקום 1 ברשימה colors .

```
colors = ["red", "green", "blue","white","black"]
colors.insert(1,"gray")
print(colors)
```

ההדפסה שנקבל : ['red', 'gray', 'green', 'blue', 'white', 'black']

11.ח מתודת ( )0pop

המתודה מסירה פריט מהרשימה במיקום שציינו. המתודה מחזירה את הערך שהוצא.

התחביר :

list.pop(pos)

pos הוא המיקום ברשימה שממנו רוצים להוציא את הפריט.

דוגמה : הוצאת הפריט השני מהרשימה colors (הפריט השני נמצא במיקום 1 כי הפריט הראשון נמצא במיקום 0) .

```
colors = ["red", "green", "blue","white","black"]
colors.pop(1)
print(colors)
```

ההדפסה שנקבל : ['red', 'blue', 'white', 'black']

אם רוצים להדפיס גם את הערך שהוצאנו נרשום :

```
colors = ["red", "green", "blue","white","black"]
x=colors.pop(1)
print("colors =  ", colors, "  the value taken out : " ,x)
```

ההדפסה שנקבל : colors = ['red', 'blue', 'white', 'black'] the value taken out : green

11.ט מתודת ( )remove

המתודה מסירה את הפריט הראשון עם הערך שציינו.

התחביר :

list.remove(elmnt)

elmnt הוא כל סוג של פריט שרוצים להסיר כמו מחרוזת, מספר, רשימה וכו'.

דוגמה : הסרה של הצבע red הראשון שמופיע ברשימה colors1 :

```
colors1 = ["red", "green", "blue","white","black", "green", "gray"]
colors1.remove("green")
print(colors1)
```

ההדפסה שנקבל : ['red', 'blue', 'white', 'black', 'green', 'gray']

11.י מתודת ( )reverse

המתודה הופכת את הסדר של הפריטים ברשימה. הפריט הראשון הופך מקומות עם האחרון, הפריט השני הופך מקומות עם הפריט לפני אחרון וכך הלאה.

התחביר : 0 **list.reverse()**

דוגמה : נהפוך את סדר הפריטים ברשימה colors

```
colors = ["red", "green", "blue", "white", "black", "gray"]
colors.reverse()
print(colors)
```

ההדפסה שנקבל : ['gray', 'black', 'white', 'blue', 'green', 'red']

11.יא מתודת ( )sort

המתודה מבצעת מיון אלפא בית בסדר עולה על רשימה . ניתן גם לבצע מיון יורד.

התחביר :

list.sort(reverse=True|False, key=myFunc)

reverse הוא אופציונלי.

אם לא רושמים כלום בתוך הסוגריים של ה reverse אז המיון הוא בסדר עולה.

אם רושמים בסוגריים reverse = False אז שוב המיון בסדר עולה.

אם רושמים reverse=True אז המיון בסדר יורד וזה דומה למתודה **list.reverse()** .

במילים key=myFunc ניתן לקרוא לפונקציה (בדוגמה נקראת myFunc) שתקבע מהם הקריטריונים של המיון.

דוגמה 1 : מיון לפי סדר אלפ0אנומרי :

```
colors = ["red", "green", "blue", "white", "black", "gray"]
colors.sort()
print(colors)
```

ההדפסה שנקבל : ['black', 'blue', 'gray', 'green', 'red', 'white']

גם עבור התוכנית הבאה שדומה לקודמת חוץ מהשורה השנייה שרשמנו colors.sort(reverse=False) והיא נראית כך :

```
colors = ["red", "green", "blue", "white", "black", "gray"]
colors.sort(reverse=False)
print(colors)
```

נקבל את אותה ההדפסה : ['black', 'blue', 'gray', 'green', 'red', 'white']

דוגמה 2 : הפיכת סדר הפריטים ברשימה :

```
colors = ["red", "green", "blue", "white", "black", "gray"]
```

```
colors.sort(reverse=True)0
print(colors)
```

ההדפסה שנקבל : ['white', 'red', 'green', 'gray', 'blue', 'black']

דוגמה 3 : שימוש במילה **key =** למיון לפי קריטריון : מיון לפי אורך הפריט :

```
def func1(e):
  return len(e)

colors = ["red", "green", "blue", "white", "black", "gray"]
colors.sort(key=func1)
print(colors)
```

הסבר התוכנית : ב 2 השורות הראשונות הגדרנו פונקציה בשם func1 המקבלת פריט ומחזירה את האורך שלו (את כמות התווים שבפריט). בתוכנית עצמה הגדרנו רשימה השולחת לפונקציה func1 פריט אחרי פריט מהרשימה , מקבלת את אורך הפריט וממיינת אותם לפי האורך. פריטים עם אותו אורך היא משאירה לפי הסדר שהיה להם ברשימה. פריט שהופיע קודם – יופיע לפני הפריט ברשימה עם אותו אורך. בהדפסה נקבל את הצבעים מצבעים עם 3 תווים , אחריהם יופיעו פריטים עם 4 תווים ובסוף אלו עם החמישה התווים. בתווים השווים הסדר הוא לפי הסדר שהיה ברשימה המקורית.

ההדפסה שנקבל : ['red', 'blue', 'gray', 'green', 'white', 'black']

דוגמה 4 : מיון בסדר הפוך לפי אורך הפריט **:**

```
def func1(e):
  return len(e)

colors = ["red", "green", "blue", "white", "black", "gray"]
colors.sort(reverse=True, key=func1)
print(colors)
```

הסבר התוכנית : התוכנית דומה לקודמת להבדיל מזה שביקשנו לבצע מיון עם סידור הפוך.

ההדפסה שנקבל : ['green', 'white', 'black', 'blue', 'gray', 'red']

דוגמה 5 : מיון רשימה של dictionaries (מילונים) בשם workers התבסס על הערך של שנת הקבלה לעבודה .

```
def myFunc(e):
  return e['start']
```

```
workers = [
  {'electric': 'Moshe', 'start': 2005},
  {'mechanic': 'Dani', 'start': 2000},
  {'secretary': 'Sara', 'start': 2019},
  {'nurse': 'Kati', 'start': 2011}
]

workers.sort(key=myFunc)

print(workers)
```

ההדפסה שנקבל :

```
[{'mechanic': 'Dani', 'start': 2000}, {'electric': 'Moshe', 'start': 2005}, {'nurse': 'Kati', 'start': 2011},
{'secretary': 'Sara', 'start': 2019}]
```

אם במקום מיון בסדר עולה נבקש מיון בסדר יורד : `workers.sort(reverse=True, key=myFunc)`

נקבל :

```
[{'secretary': 'Sara', 'start': 2019}, {'nurse': 'Kati', 'start': 2011}, {'electric': 'Moshe', 'start': 2005},
{'mechanic': 'Dani', 'start': 2000}]
```

12. תרגילים

כל התרגילים יתייחסו לאחת או שתי הרשימות הבאות :

1. colors = ["red", "green", "blue", "white", "black", "gray"]
2. numbers = [90,70,54,55,67,89]

1. הדפס את האיבר השלישי בכל אחת מהרשימות.
2. שנה את הפריט השלישי בכל רשימה . ברשימה הראשונה לצבע האהוב עליך וברשימה השנייה לציון שאת/ה רוצים לקבל במבחן.
3. החלף את האיבר הראשון ברשימה colors מ "red" ל "orange" וברשימה השנייה החלף את המספר הקטן ביותר במספר 100 .
4. יש להשתמש במתודה append כדי להוסיף לרשימה הראשונה את הפריט "orange" ואת המספר 100 לרשימה numbers .
5. יש להשתמש במתודה insert כדי להכניס את הצבע "brown" כפריט השלישי ברשימה colors ואת המספר 100 כפריט הרביעי ברשימה numbers .
6. יש להשתמש במתודה remove כדי להסיר את הצבע "blue" ברשימה colors ואת המספר 70 ברשימה numbers.
7. יש להשתמש באינדקס שלילי כדי להדפיס את הפריט האחרון בכל אחת משתי הרשימות.
8. יש להשתמש בטווח של אינדקסים כדי להדפיס את הפריטים השני עד החמישי (כולל החמישי) בכל אחת מהרשימות.
9. יש להדפיס את מספר הפריטים בכל רשימה (ניתן להיעזר בפונקציה len).
10. יש לצרף את הרשימה numbers בסיום colors ולהדפיס את הרשימה החדשה שנוצרה.