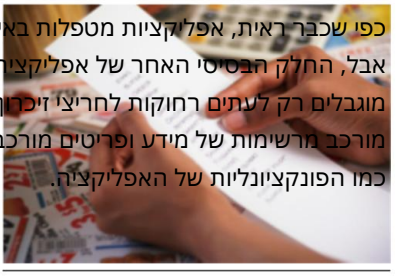
תכנות רשימות נתונים



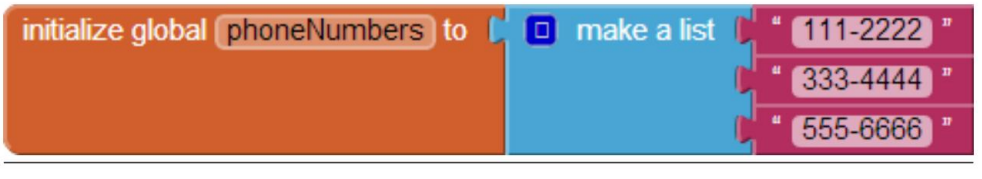
כפי שכבר ראית, אפליקציות מטפלות באירועים ומקבלות החלטות; עיבוד כזה הוא בסיסי למחשוב. אבל, החלק הבסיסי האחר של אפליקציה הוא הנתונים שלה -המידע שהיא מעבדת. נתוני אפליקציה מוגבלים רק לעתים רחוקות לחריצי זיכרון בודדים, כגון הניקוד של משחק. לעתים קרובות יותר, הוא מורכב מרשימות של מידע ופריטים מורכבים הקשורים זה בזה, שחייבים להיות מאורגנים בקפידה בדיוק כמו הפונקציונליות של האפליקציה.

איור 19-1.

בפרק זה, נבחן את הדרך שבה App Inventorמטפל בנתונים. תלמד את היסודות של תכנות הן מידע סטטי, שבו הנתונים אינם משתנים, והן מידע דינמי, שבו הנתונים מוזנים על ידי משתמש הקצה. תלמד כיצד לעבוד עם רשימות, ולאחר מכן תחקור מבנה נתונים מורכב יותר הכולל רשימות של רשימות ואפליקציית חידון רב-ברירה.

אפליקציות רבות מעבדות רשימות של נתונים. לדוגמה, פייסבוק מעבדת את הרשימה שלך של חברים ורשימות של דוחות סטטוס. אפליקציית חידון פועלת עם רשימה של שאלות ותשובות. למשחק עשוי להיות רשימה של דמויות או ציונים שיא בכל הזמנים.

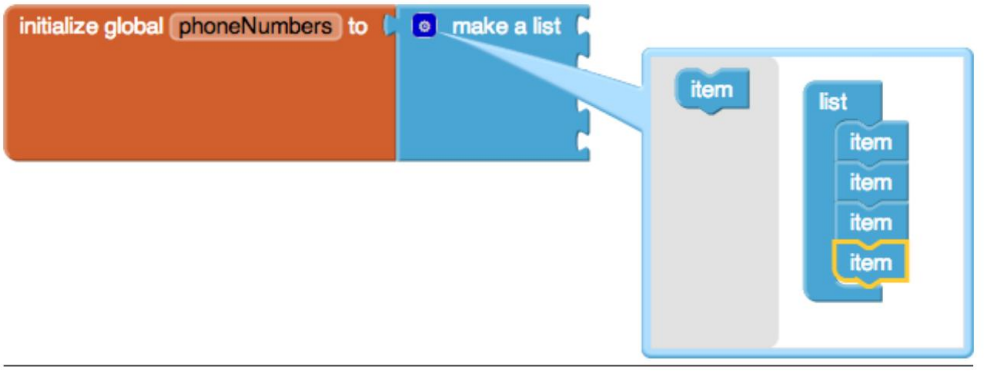
אתה מציין נתוני רשימה App Inventor-בעם משתנה, אבל במקום לתת שם ליחיד תא זיכרון עם המשתנה, אתה שם לקבוצה של תאי זיכרון קשורים. אתה מציין שמשתנה הוא מרובה פריטים על ידי שימוש ב- make a listאו יצירת בלוקי רשימה ריקים . לדוגמה, ה- phoneNumbersהמשתנים באיור 19-1מגדיר רשימה של שלושה פריטים.



איור phoneNumbers .19-2נותנת שמות לשלושה תאי זיכרון המאוחלים עם המספרים המוצגים

יצירת משתנה רשימה

אתה יוצר משתנה רשימה בעורך הבלוקים על ידי שימוש בבלוק משתנה גלובלי של אתחול ולאחר מכן חיבור לחשמל צור רשימה . אתה יכול למצוא את הבלוק יצירת רשימה במגירת הרשימות, ויש לו רק שני שקעים. אבל אתה יכול לציין את מספר השקעים שאתה רוצה ברשימה על ידי לחיצה על הסמל הכחול והוספת פריטים, כפי שמתואר באיור .19-2



איור .19-3לחץ על הסמל הכחול בצור רשימה כדי לשנות את מספר הפריטים

אתה יכול לחבר כל סוג של נתונים לשקעי "פריט" של יצירת רשימה. עבור הדוגמה של ,phoneNumbersהפריטים צריכים להיות אובייקטי טקסט, לא מספרים, מכיוון שלמספרי טלפון יש מקפים וסמלים עיצוביים אחרים שלא ניתן להכניס לאובייקט מספר, ולא תבצע חישובים כלשהם על המספרים (שבהם במקרה, תרצה אובייקטים מספר, במקום זאת).

בחירת פריט ברשימה

בזמן שהאפליקציה שלך פועלת, תצטרך לבחור פריטים מהרשימה; לדוגמה, שאלה מסוימת כאשר המשתמש עובר חידון או מספר טלפון מסוים שנבחר מתוך רשימה.
אתה ניגש לפריטים ברשימה באמצעות אינדקס; כלומר, על ידי ציון מיקום ברשימה. אם רשימה כוללת שלושה פריטים, אתה יכול לגשת לפריטים על ידי שימוש במדדים 2 ,1ו-3.
אתה יכול להשתמש בבלוק פריט הבחירה כדי לתפוס פריט מסוים, כפי שמוצג באיור .19-3

איור .19-4בחירת הפריט השני ברשימה

עם פריט בחירה, אתה מחבר את הרשימה הרצויה לשקע הראשון, ואת האינדקס הרצוי לשקע השני. עבור מדגם מספר הטלפון הזה, התוצאה של בחירת הפריט השני היא "333–4444".

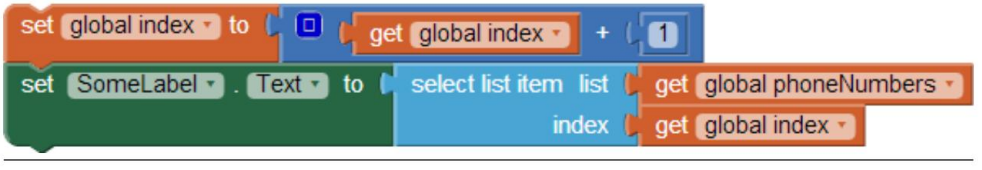
שימוש באינדקס כדי לעבור רשימה

באפליקציות רבות, תגדיר רשימה של נתונים ולאחר מכן תאפשר למשתמש לעבור (או לחצות) אותם. חידון הנשיאים בפרק 8מספק דוגמה טובה לכך: באותה אפליקציה, כאשר המשתמש לוחץ על כפתור הבא, הפריט הבא נבחר מרשימת שאלות ומוצג.

הסעיף הקודם הראה כיצד לבחור את הפריט השני ברשימה, אך כיצד לבחור את הפריט הבא? כאשר אתה חוצה רשימה, מספר הפריט שאתה בוחר משתנה בכל פעם; זה המיקום הנוכחי שלך ברשימה. לכן, עליך להגדיר משתנה כדי לייצג את המיקום הנוכחי. "אינדקס" הוא השם הנפוץ למשתנה כזה, והוא בדרך כלל מאותחל ל-1 (המיקום הראשון ברשימה), כפי שמוצג באיור 19-4.

איור 19-5. אתחול המדד המשתנה ל-1

כאשר המשתמש עושה משהו כדי לעבור לפריט הבא, אתה מגדיל את האינדקס משתנה על ידי הוספת ערך של 1אליו, ולאחר מכן בחר מתוך הרשימה באמצעות הערך המוגדל הזה. איור 19-5מציג את הבלוקים לעשות זאת.



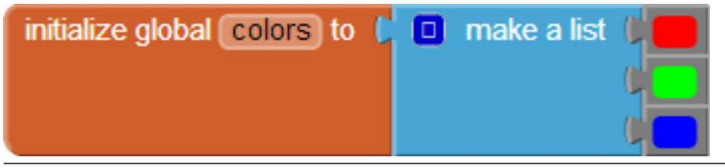
איור 19-6. הגדלת ערך האינדקס ושימוש בערך המוגדל כדי לבחור את פריט הרשימה הבא

דוגמה: מעבר ברשימת צבעי צבע

הבה נבחן אפליקציה לדוגמה שבה המשתמש יכול לעיין בכל צבע צבע פוטנציאלי עבור הבית שלו על ידי הקשה על "כפתור צבע". בכל פעם שהמשתמש מקיש, צבע הכפתור משתנה. כאשר המשתמש עובר את כל הצבעים האפשריים, האפליקציה חוזרת לראשון.

עבור דוגמה זו, נשתמש במספר צבעים בסיסיים. עם זאת, אתה יכול להתאים אישית את בלוקי הקוד כך שיעברו דרך כל קבוצה של צבעים.

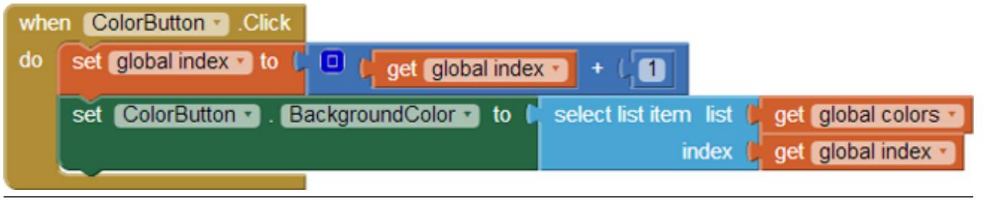
הצעד הראשון שלך הוא להגדיר משתנה רשימה עבור רשימת הצבעים ולאתחל אותו עם כמה צבעי צבע כפריטים, כפי שמתואר באיור 19-6.



איור 19-7. אתחול צבעי הרשימה עם רשימה של צבעי צבע

לאחר מכן, הגדר משתנה אינדקס שעוקב אחר המיקום הנוכחי ברשימה. זה אמור להתחיל ב-1. אתה יכול לתת למשתנה שם תיאורי כגון currentColorIndex, אבל אם אתה לא עוסק במספר אינדקסים באפליקציה שלך, אתה יכול פשוט לקרוא לו "אינדקס", כמו באיור 19-4.

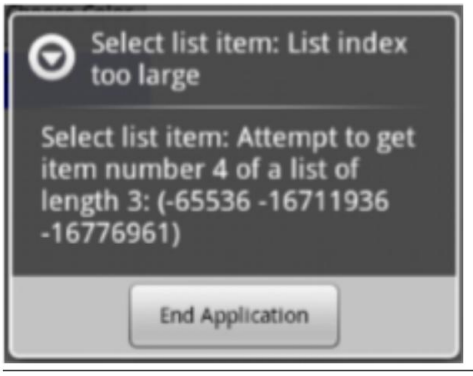
המשתמש עובר לצבע הבא ברשימה על ידי לחיצה על כפתור הצבע. בכל הקשה, יש להגדיל את האינדקס וצבע הרקע של הלחצן אמור להשתנות לפריט שנבחר כעת, כפי שמוצג באיור 19-7.



איור 19-8. כל הקשה על הכפתור משנה את צבעו

נניח שרקע הכפתור מוגדר בהתחלה לאדום. בבפעם הראשונה שהמשתמש לוחץ על הכפתור, האינדקס משתנה מערכו ההתחלתי של 1 ל-2, וצבע הרקע של הכפתור משתנה לפריט השני ברשימה, ירוק. בפעם השנייה שהמשתמש מקיש עליו, האינדקס משתנה מ-2 ל-3, וצבע הרקע עובר לכחול.

אבל מה לדעתך יקרה בפעם הבאה שהמשתמש יקיש עליו?
אם אמרת שתהיה שגיאה, אתה צודק! האינדקס יהפוך ל-4 והאפליקציה תנסה לבחור את הפריט הרביעי ברשימה, אך ברשימה יש רק שלושה פריטים. האפליקציה תאלץ לסגור, או להפסיק, והמשתמש יראה הודעת שגיאה כמו זו באיור 19-8.



איור 19-9. הודעת השגיאה המוצגת כאשר האפליקציה מנסה לבחור פריט רביעי מתוך רשימה של שלושה פריטים

ברור שההודעה הזו היא לא משהו שאתה רוצה שמשתמשי האפליקציה שלך יראו. כדי להימנע מבעיה זו, הוסף בלוק אם כדי לבדוק אם הצבע האחרון ברשימה הושג. אם כן, ניתן לשנות את האינדקס בחזרה ל-1 כך שהצבע הראשון יוצג שוב, כפי שמוצג באיור 19-9.

איור 19-10. שימוש ב-fi כדי לבדוק אם ערך האינדקס גדול מאורך הרשימה

כאשר המשתמש מקיש על הכפתור, האינדקס מוגדל ולאחר מכן נבדק כדי לראות אם הערך שלו גדול מדי. המדד מושווה לאורך הרשימה, לא ;3כך האפליקציה שלך תעבוד גם אם תוסיף פריטים לרשימה. על ידי בדיקה אם האינדקס גדול מאורך הרשימה שלך (לעומת בדיקה אם הוא גדול מהמספר הספציפי ,(3ביטלת תלות בקוד באפליקציה שלך. תלות בקוד היא מונח תכנות המתאר קוד שמוגדר באופן ספציפי מדי וחסר גמישות. לפיכך, אם אתה משנה משהו במקום אחד -בדוגמה שלנו כאן, אתה מוסיף פריטים לרשימה שלך -תצטרך לחפש כל מופע שבו אתה משתמש ברשימה הזו ולשנות אותה במפורש.

כפי שאתה יכול לדמיין, תלות מסוג זה עשויה להסתבך מהר מאוד, ו
הם בדרך כלל מובילים להרבה יותר באגים שתוכל לרדוף אחריהם גם כן. למעשה, ה

דוגמה: מעבר ברשימת צבעי צבע

עיצוב עבור אפליקציית הצבע שלנו מכיל עוד תלות בקוד כפי שהוא מתוכנת כעת. אתה יכול להבין מה זה?

אם שינית את הצבע הראשון ברשימה שלך מאדום לצבע אחר, האפליקציה הראשוניButton.BackgroundColor לא יעבוד כראוי אלא אם כן זכרתם לשנות גם את ה- שהגדרתם .Component Designer-בהדרך לבטל את התלות בקוד היא להגדיר את הצבע הראשוני ColorButton.BackgroundColorלצבע הראשון ברשימה ולא לצבע ספציפי. מכיוון ששינוי זה כרוך בהתנהגות המתרחשת כאשר האפליקציה שלך נפתחת לראשונה, אתה עושה זאת ב- Screen.Initializeמטפל באירועים המופעל בעת הפעלת אפליקציה, כפי שמוצג באיור 19-10.

איור 19-11. הגדרת צבע הרקע של הכפתור לצבע הראשון ברשימה כאשר האפליקציה מופעלת

יצירת טפסי קלט ונתונים דינמיים

אפליקציית Colorהקודמת כללה רשימה סטטית: כזו שהרכיבים שלה מוגדרים על ידי המתכנת (אתה) והפריטים שלו לא משתנים אלא אם כן אתה משנה את הבלוקים עצמם. עם זאת, לעתים קרובות יותר, אפליקציות עוסקות בנתונים דינמיים: מידע שמשתנה על סמך משתמש הקצה שמזין פריטים חדשים, או פריטים חדשים שנטענים ממסד נתונים או מקור מידע באינטרנט. בחלק זה, אנו דנים באפליקציית Note Takerלדוגמה, שבה המשתמש מזין הערות בטופס ויכול להציג את כל ההערות הקודמות שלה.

הגדרת רשימה דינמית

אפליקציות כגון Note Takerמתחילות ברשימה ריקה. כאשר אתה רוצה רשימה שמתחילה ריקה, אתה מגדיר אותה באמצעות בלוק צור רשימה ריקה , כפי שמתואר באיור 19-11.

איור 19-12. הבלוקים להגדרת רשימה דינמית אינם מכילים פריטים מוגדרים מראש

הוספת פריט

בפעם הראשונה שמישהו מפעיל את האפליקציה, רשימת ההערות ריקה. אבל כאשר המשתמש מקליד כמה נתונים בטופס ומקיש על שלח, הערות חדשות יתווספו לרשימה. הטופס עשוי להיות פשוט כמו זה שמוצג באיור 19-12.

איור 19-13. שימוש בטופס להוספת פריטים חדשים לרשימת ההערות

כאשר המשתמש מקליד הערה ומקיש על כפתור שלח, האפליקציה קוראת להוספה פריטים לרשימה פונקציה לצרף את הפריט החדש לרשימה, כפי שמוצג באיור 19-13.

איור 19-14. קורא הוסף פריטים לרשימה כדי להוסיף את ההערה החדשה כאשר המשתמש מקיש על הכפתור Submit

אתה יכול למצוא את הוספת הפריטים לבלוק הרשימה במגירת הרשימה. היזהר: יש גם בלוק הוספה לרשימה , אבל זה הוא בלוק נדיר למדי המשמש לצרף רשימה שלמה אחת לאחרת.

הצגת רשימה

התוכן של משתני רשימה, כמו כל המשתנים, אינו גלוי למשתמש. הבלוקים באיור 19-13 מוסיפים פריטים לרשימה בכל פעם שמופעל SubmitButton.Click, אך המשתמש לא יקבל משוב שהרשימה גדלה עד שתתכנת בלוקים נוספים כדי להציג בפועל את תוכן הרשימה.

הדרך הפשוטה ביותר להציג רשימה בממשק המשתמש של האפליקציה שלך היא להשתמש באותה שיטה שבה אתה משתמש להצגת מספרים וטקסט: שים את הרשימה במאפיין Text של רכיב תווית , כפי שמוצג באיור 19-14.

איור 19-15. הצגת הרשימה למשתמש על ידי הצבתה בתווית.

למרבה הצער, השיטה הפשוטה הזו להצגת רשימה אינה אלגנטית במיוחד; זה מציב את רשימה בסוגריים, כאשר כל פריט מופרד ברווח ולא בהכרח באותה שורה. לדוגמה, אם המשתמש יקליד, "האם אי פעם אסיים את הספר הזה?" בתור ההערה הראשונה, ו"אני שוכח איך הבן שלי נראה!" בתור השני, האפליקציה תציג את רשימת ההערות בדומה למה שאנו רואים באיור 19-15.

איור 19-16. ערכים אלה רשומים באמצעות עיצוב ברירת מחדל

בפרק 20, אתה יכול לראות דרך מתוחכמת יותר להציג רשימה.

הסרת פריט מרשימה

ניתן להסיר פריט מרשימה באמצעות בלוק הסר פריט רשימה , כפי שמוצג באיור 19-16.

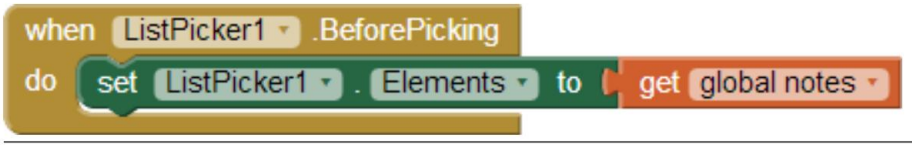
איור 19-17. הסרת פריט מרשימה

הבלוקים באיור 19-16 מסירים את הפריט השני מהרשימה הנקראת הערות.
עם זאת, בדרך כלל, לא תרצה להסיר פריט קבוע (למשל, 2), אלא תספק מנגנון למשתמש לבחור את הפריט להסרה.
אתה יכול להשתמש ברכיב ListPicker כדי לספק למשתמש דרך לבחור פריט. ListPicker מגיע עם כפתור משויך. כאשר לוחצים על הכפתור, ListPicker מציג את הפריטים של רשימה שממנה המשתמש יכול לבחור אחד. כאשר המשתמש בוחר פריט, האפליקציה יכולה להסיר אותו.
קל לתכנת את ListPicker אם אתה מבין את אירועי המפתח שלו, BeforePicking ו-AfterPicking, ומאפייני המפתח שלו, Elements, Selection ו-SelectionIndex (ראה טבלה 19-1).

טבלה 19-1. האירועים והמאפיינים העיקריים של רכיב ListPicker

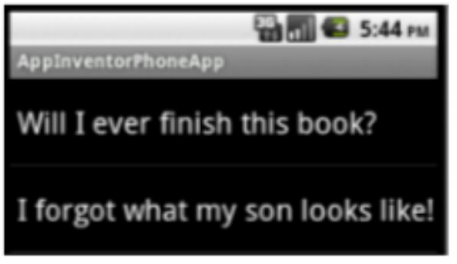
תכונה	מִקרֶה
לפני בחירה: מופעל בעת לחיצה על הכפתור. אלמנטים: רשימת האפשרויות.	
AfterPicking: מופעל כאשר המשתמש עושה בחירה. בחירה: בחירת המשתמש.	
SelectionIndex: מיקום בחירה.	

המשתמש מפעיל את אירוע ListPicker.BeforePicking על ידי הקשה על ListPicker's כפתור משויך. במטפל האירועים , ListPicker.BeforePicking תגדיר את המאפיין ListPicker.Elements למשתנה רשימה כך שהנתונים ברשימה יוצגו. עבור האפליקציה Taker, Note תגדיר את Elements למשתנה הערות שמכיל את רשימת ההערות שלך, כפי שמוצג באיור 19-17.



איור 19-18. המאפיין Elements של ListPicker1 מוגדר לרשימת ההערות

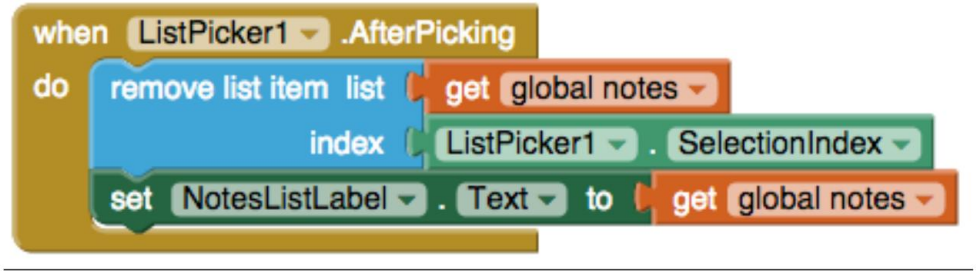
עם בלוקים אלה, הפריטים של הערות הרשימה יופיעו ב- ListPicker. אם היו שני הערות, זה היה מופיע כפי שמוצג באיור 19-18.



איור 19-19. רשימת ההערות מופיעה ב- ListPicker

כאשר המשתמש בוחר פריט ברשימה, הוא מפעיל את ListPicker.AfterPicking מִקרֶה. במטפל אירועים זה, אתה יכול לגשת לבחירת המשתמש במאפיין ListPicker.Selection .

עם זאת, המטרה שלך בדוגמה זו היא להסיר פריט מהרשימה, והסרת פריט מבלוק הרשימה מצפה לאינדקס, לא לפריט. המאפיין Selection של ListPicker הוא הנתונים בפועל (ההערה), לא האינדקס. לכן, עליך להשתמש במאפיין SelectionIndex במקום זאת מכיוון שהוא מספק לך את האינדקס של הפריט הנבחר. יש להגדיר אותו כאינדקס של בלוק הסר פריט רשימה , כפי שמוצג באיור 19-19.

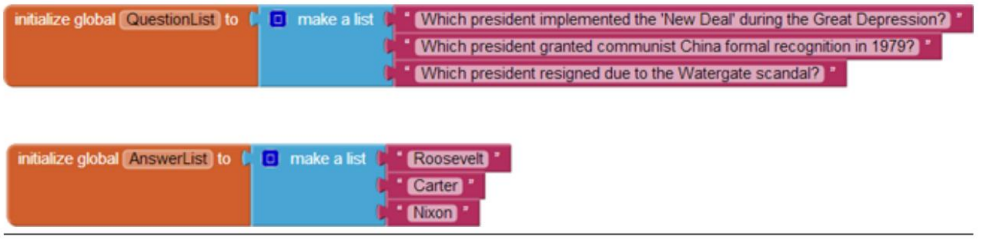


איור 19-20. הסרת פריט באמצעות ListPicker.SelectionIndex

רשימות של רשימות

הפריטים של רשימה יכולים להיות מכל סוג, כולל מספרים, טקסט, צבעים או ערכים בוליאניים (true/false). אבל, הפריטים של רשימה יכולים גם, בעצמם, להיות רשימות. מבני נתונים מורכבים כאלה נפוצים. לדוגמה, ניתן להשתמש ברשימת רשימות כדי להמיר את חידון הנשיאים (פרק 8) לחידון רב-ברירה. בואו נסתכל שוב על הבסיס

מבנה חידון הנשיאים, שהוא רשימת שאלות ורשימת תשובות, כפי שמוצג באיור .19-20



איור .19-21רשימת שאלות ורשימת תשובות

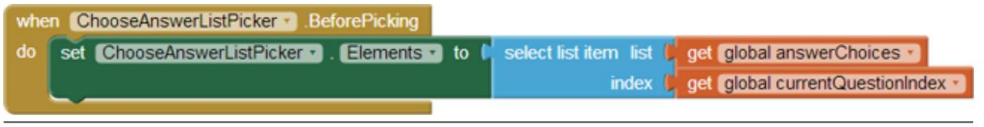
בכל פעם שהמשתמש עונה על שאלה, האפליקציה בודקת אם היא נכונה על ידי השוואת התשובה לפריט הנוכחי ברשימת התשובות.

כדי להפוך את החידון לבחירה מרובה, עליך לשמור רשימה נוספת, כזו המאחסנת את האפשרויות עבור כל תשובה לכל שאלה. אתה מציין נתונים כאלה על ידי הצבת שלושה בלוקים של יצירת רשימה בתוך בלוק יצירת רשימה פנימית , כפי שמוצג באיור .19-21

איור .19-22רשימה של רשימות נוצרת על ידי הוספת הפוך לרשימה בלוקים כפריטים בתוך בלוק הפוך לרשימה פנימית

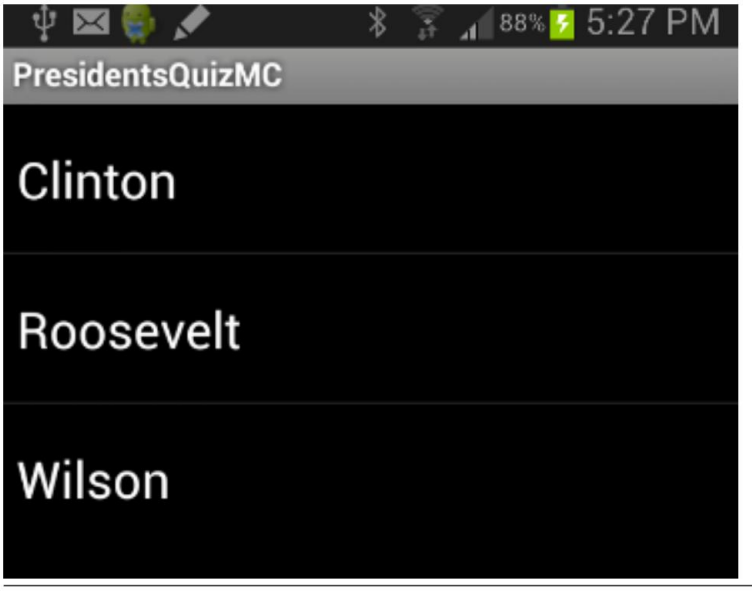
כל פריט במשתנה answerChoicesהוא בעצמו רשימה המכילה שלושה פריטים. אם תבחר פריט מתוך answerChoices, התוצאה היא רשימה. כעת, לאחר שמילאת את התשובות מרובות הברירות שלך, כיצד תציג זאת למשתמש?

כמו באפליקציית Note Taker, אתה יכול להשתמש ב- ListPickerכדי להציג בפניו את האפשרויות המשתמש. אם האינדקס היה נקרא currentQuestionIndex, האירוע ListPicker.BeforePickingיופיע כפי שמוצג באיור .19-22



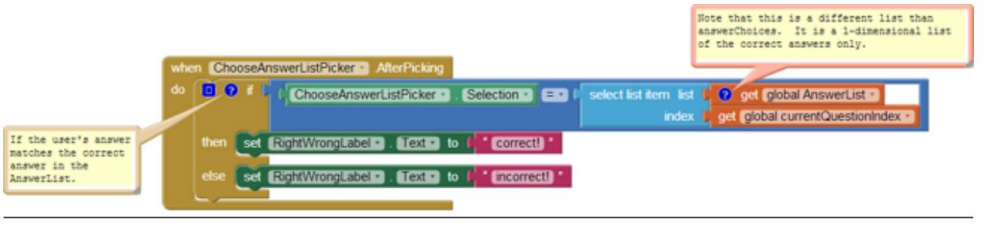
איור 19-23. שימוש בבורר הרשימות כדי להציג אחת מרשימת אפשרויות התשובות ל-
מִשׁתַמֵשׁ

בלוקים אלה יקחו את רשימת המשנה הנוכחית של אפשרויות תשובות ויאפשרו למשתמש
לבחור ממנו. אז אם currentQuestionIndex היה 1, ListPicker היה מציג רשימה כמו זו באיור 19-23.



איור 19-24. אפשרויות התשובה שהוצגו למשתמש עבור השאלה הראשונה

כאשר המשתמש בוחר, אתה בודק את התשובה עם הבלוקים המוצגים באיור 19-24.



איור 19-25. בדיקה האם המשתמש בחר בתשובה הנכונה

בלוקים אלו, הבחירה של המשתמש מתוך ListPickerמשווה לתשובה הנכונה, אשר מאוחסנת ברשימה שונה, AnswerList(מכיוון answerChoices-שמספקת רק את האפשרויות ואינה מציינת את התשובה הנכונה).

סיכום

רשימות משמשות כמעט בכל אפליקציה שאתה יכול לחשוב עליה. הבנת איך הם עובדים היא בסיסית לתכנות. בפרק זה, חקרנו את אחת מדפוסי התכנות הנפוצים ביותר: שימוש במשתנה אינדקס שמתחיל בתחילת הרשימה ומוגדל עד שכל פריט רשימה מעובד. אם אתה יכול להבין ולהתאים אישית את הדפוס הזה, אתה אכן מתכנת!

לאחר מכן כיסינו כמה מהמנגנונים האחרים למניפולציה ברשימות, כולל טפסים טיפוסיים לאפשר למשתמש להוסיף ולהסיר פריטים. תכנות כזה דורש רמה נוספת של הפשטה, מכיוון שאתה צריך לדמיין את הנתונים הדינמיים לפני שהם באמת קיימים. אחרי הכל, הרשימות שלך ריקות עד שהמשתמש מכניס בהן משהו. אם אתה יכול להבין את זה, אולי אפילו תחשוב לעזוב את העבודה היומיומית שלך.
סיימנו את הפרק בהכנסת מבנה נתונים מורכב, רשימה של רשימות.
זהו ללא ספק מושג קשה, אבל חקרנו אותו באמצעות נתונים קבועים: אפשרויות התשובות עבור חידון רב-ברירה. אם שלטת בזה ובשאר הפרק, המבחן הסופי שלך הוא זה: צור אפליקציה שמשתמשת ברשימת רשימות אבל עם נתונים דינמיים. דוגמה אחת תהיה אפליקציה שבעזרתה אנשים יכולים ליצור חידונים מרובי-ברירה משלהם, ולהרחיב עוד יותר את אפליקציית MakeQuizבפרק .10בהצלחה!

בזמן שאתה חושב על איך תתמודד עם זה, הבינו שהחקירה שלנו על רשימות לא נעשה. בפרק הבא, אנו ממשיכים בדיון ומתמקדים באיטרציה של רשימה עם טוויסט: החלת פונקציות על כל פריט ברשימה.