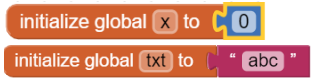
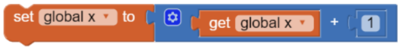
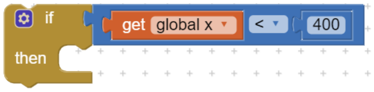
MIT App Inventor 2 Notes

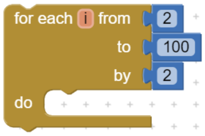
Contents

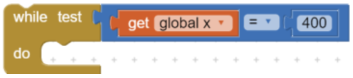
Python to App Inventor 2 Quick Reference Table

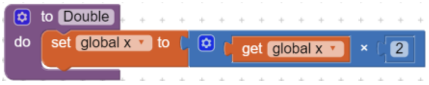
		Python	App Inventor 2 Equivalent	Where

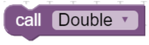
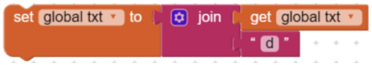
1	Declare Variables	`x=200`
		`txt="abc"`
			
			initialize global x to 0
			initialize global txt to " abc "
				Built-in: Variables
				Built-in: Math
				Built-in: Text

2	Math	`x=x+1`	 set global x to get global x + 1	Built-in: Variables Built-in: Math
4	Conditional	`if x<400:`	 if get global x < 400 then	Built-in: Control Built-in: Math Built-in: Variables

3	For Loop	`for I in range`
		`(2,102,2):`
			
			for each i from 2
			to 100
			by 2
			do
				Built-in: Control

4	While loop	`while x<400:`	 while test get global x = 400 do	Built-in: Control Built-in: Math Built-in: Variables

5	Function def	`def Double:`
		`x=x*2`
			
			to Double
			do set global x to get global x × 2
				Built-in: Procedures
				Built-in: Variables
				Built-in: Math

6	Function call	`Double`	 call Double	Built-in: Procedures
7	Concatenation	`txt=txt+"d"`	 set global txt to join get global txt " d "	Built-in: Variables Built-in: Text

1. Text to Speech[1]

a. Designer:

i. User Interface: Button

ii. User Interface: TextBox

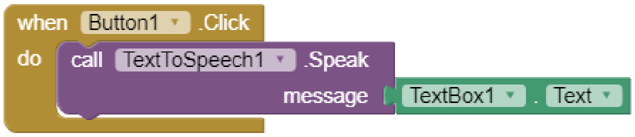
iii. Media: TextToSpeech

b. Blocks:

i. Button: When button Click

ii. Text to Speech: Call TextToSpeech:

1. Text: The words the phone will speak



```
when Button1 .Click
do  call TextToSpeech1 .Speak
                   message  TextBox1 . Text
```

2. Sensing Motion[2]

a. Designer:

[1] MIT covers this in a tutorial video. Follow this link: https://www.youtube.com/watch?v=Vdo8UdkgDD8

[2] There is an MIT tutorial video for this topic. https://www.youtube.com/watch?v=0hikoCvM3oc

i. Sensors: AccelerometerSensor

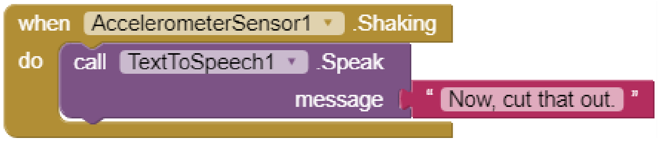
ii. Media: TextToSpeech

b. Blocks:

i. AccelerometerSensor: When AccelerometerSensor Shaking

1. TextToSpeech:

a. Text: Text: “Now cut that out.”



```
when AccelerometerSensor1 .Shaking
do  call TextToSpeech1 .Speak
              message  " Now, cut that out. "
```

3. How to Send Text Messages

a. Designer:

i. Built-In: Button <u>or</u> Screen1:Initialize

ii. Social: Texting

b. Blocks:

i. Social: Texting: Set Phone To + Text containing phone number

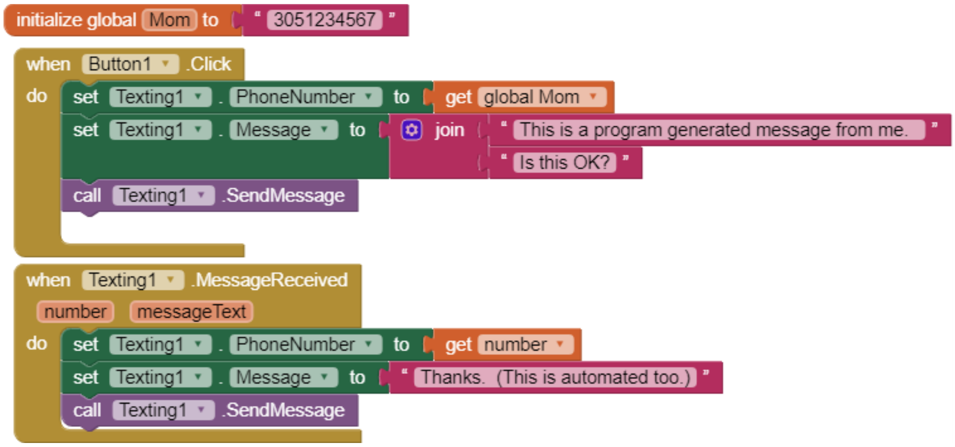
ii. Social: Texting: Set Message To + Text containing message

iii. Social: Texting: Call Send Message

c. Modifications:

i. Substitute text variables for text

ii. Add Texting: Message Received for an automated response. I



```
initialize global Mom to " 3051234567 "

when Button1 .Click
do  set Texting1 . PhoneNumber to  get global Mom
    set Texting1 . Message to  join  " This is a program generated message from me. "
                                     " Is this OK? "
    call Texting1 .SendMessage

when Texting1 .MessageReceived
 number  messageText
do  set Texting1 . PhoneNumber to  get number
    set Texting1 . Message to  " Thanks.  (This is automated too.) "
    call Texting1 .SendMessage
```

4. How to Play a Sound

a. Media: (Where to get sounds.)

i. Record your own

ii. Search http://www.findsounds.com or http://www.flashkit.com

1. Right click sound: Save link to

b. Designer:

i. User interface: Button drag to phone

ii. Media: Sound drag to phone

1. Source: click,

a. Upload file:browse

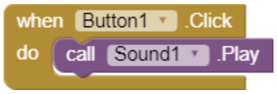
iii. Media: VideoPlayer

1. Source: Upload your video.

c. Blocks:

i. Screen 1: When Button Click

1. Screen1: Sound: Call sound1.play



```
when Button1 .Click
do   call Sound1 .Play
```

5. How to Play a Video

a. Media: (How to get video.)

i. Shoot the video, or download it. Short is better.

ii. Upload it to http://www.smallervideo.com Reduce the size of the video to less than 3MB.[3]

iii. Search YouTube and use mp4 video capture addon.

b. Designer:

i. User Interface: Button

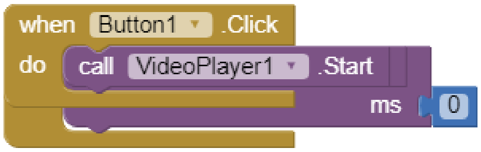
ii. Media: VideoPlayer

1. Source: Upload your video.

c. Blocks:

i. Built-In: When Button Click

ii. VideoPlayer: Call Video Player Start + Math 0 (Where to start in the video.)



```
when Button1 .Click
do   call VideoPlayer1 .Start
                         ms  0
```

6. How to Send an e-mail

a. Designer:

i. User Interface: Button

ii. Connetivity: Activity Starter

b. Blocks:

i. Button: When button Click

[3] MIT App Inventor 2 will not upload videos larger than about 3MB.

ii. Activity Starter: Set Action: (Text) “android.intent.action.VIEW” (Case sensitive.)

iii. Activity Starter: DataUri:

   a. Text: Join:

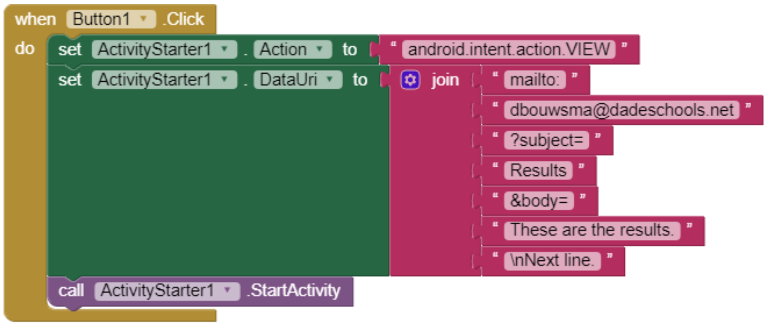
      i. Text: mailto:

      ii. Text: somebody@somewhere.com

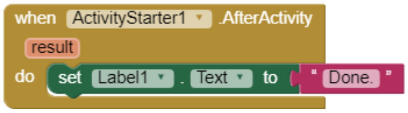
      iii. Text: ?subject=Your subject here

      iv. Text: &body=Your e-mail message here.

iv. Activity Starter: call Activity Starter Start Activity



```
when Button1 .Click
do  set ActivityStarter1 . Action to " android.intent.action.VIEW "
    set ActivityStarter1 . DataUri to join " mailto: "
                                          " dbouwsma@dadeschools.net "
                                          " ?subject= "
                                          " Results "
                                          " &body= "
                                          " These are the results. "
                                          " \nNext line. "
    call ActivityStarter1 .StartActivity
```



```
when ActivityStarter1 .AfterActivity
 result
do  set Label1 . Text to " Done. "
```

7. How to Show the Current Time

a. Designer:

i. User Interface: Label

ii. Sensors: Clock:

   1. TimerInterval: 1000 (The clock counts in thousandths of seconds, so this is one second.)

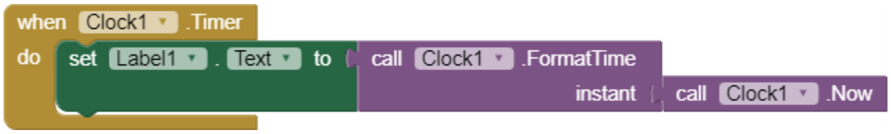
b. Blocks:

i. Clock: When Clock.Timer

ii. Set Label Text to:

   1. Call Clock FormatTime Instant:

      a. Call Clock.Now



```
when Clock1 .Timer
do  set Label1 . Text to call Clock1 .FormatTime
                                      instant call Clock1 .Now
```

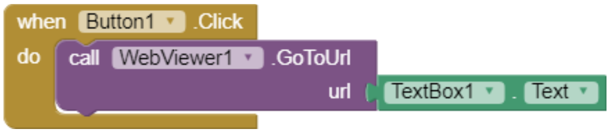
8. How to Display a Web Page[4]

a. Designer:

[4] Follow this link for a video on this subject. https://www.youtube.com/watch?v=XD_R-MCOPQ0

        i. User Interface: Button
        ii. User Interface: TextBox
        iii. User Interface: WebViewer (Draw to large size)

    b. **Blocks:**
        i. When Button Click
            1. Webviewer: Call WebViewer Goto URL:
            2. TextBox: get TextBox.Text



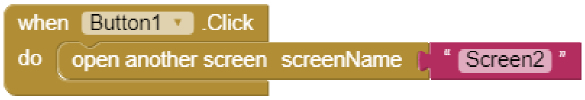
```
when Button1 .Click
do  call WebViewer1 .GoToUrl
                         url  TextBox1 . Text
```

9. How to Display a Whole New Screen

a. **App Inventor Green Bar:** Click **Add Screen**.

b. **Designer:** (Screen 1)
    i. User Interface: Button

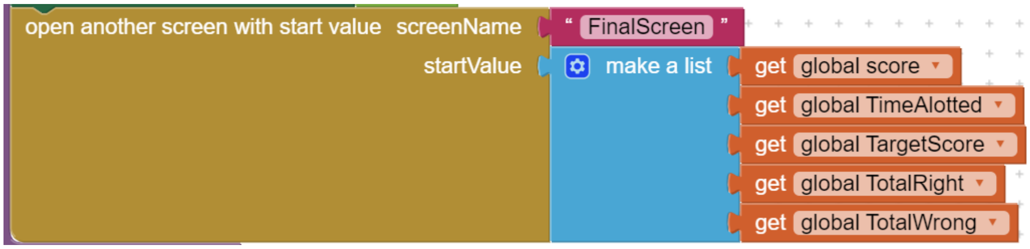
c. **Blocks:** (Screen 1)
    i. Built-in: When Button Click:
    ii. Built-in: Controls: open another screen screen Name:
        1. Text: Text:”Screen2”



```
when Button1 .Click
do  open another screen  screenName  " Screen2 "
```

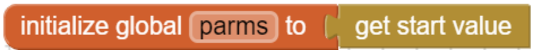
10.Passing Information to a New Screen

a. **Blocks:** (Screen 1, the sending screen)
    i. Built-in: Control: Open another screen with start value
        1. Built-in: Text: " "
        2. Built-in: Lists: make a list



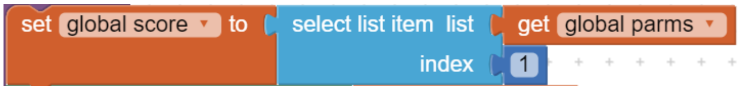
```
open another screen with start value  screenName  " FinalScreen "
                                      startValue  make a list  get global score
                                                               get global TimeAlotted
                                                               get global TargetScore
                                                               get global TotalRight
                                                               get global TotalWrong
```

b. **Blocks:** (Screen 2, the receiving screen)
    i. Built-in: Variables: Initialize global [name] to
        1. Built-in: Control: get start value



```
initialize global parms to  get start value
```

    ii. Built-in: Variables: set ▼



```
set global score to  select list item  list  get global parms
                                       index 1
```

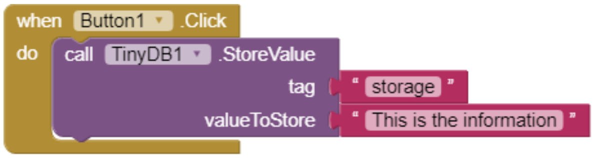
11. Persistent Storage in App Inventor 2[5]

a. Designer:

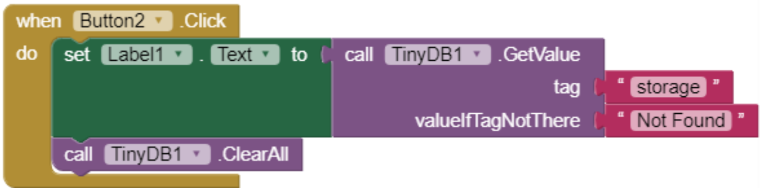
- i. User Interface: Button1 (Remember button)
- ii. User Interface: Button2 (Recall button)
- iii. User Interface: Label (Used to show the recovered data.)
- iv. Storage: **TinyDb**[6]

b. Blocks:

- i. Button1: When Button1 Click: (Remember button.)
    - 1. Call TinyDb StoreValue:
        - a. Tag:
            - i. Text: Text: “storage”   (the name the information is stored under.)
        - b. ValueToStore:
            - i. Text: Text: “The information you need to keep typed here.”
- ii. Button2: When Button2 Click: (Recall button)
    - 1. Label: Text:
        - a. Call TinyDb: GetValue:
            - i. Tag:
                - 1. Text: Text: “storage”   (the name the information is stored under.)
            - ii. ValueIfTagNotThere:
                - 1. Text: Text: “Not Found.”
- iii. Call TinyDb.ClearAll   (Erase to save space on phone.)



```
when Button1 .Click
do  call TinyDB1 .StoreValue
                        tag  " storage "
               valueToStore  " This is the information "
```



```
when Button2 .Click
do  set Label1 . Text to  call TinyDB1 .GetValue
                                            tag  " storage "
                               valueIfTagNotThere  " Not Found "
    call TinyDB1 .ClearAll
```

[5] When App Inventor 2 puts up a new screen or is turned off, it forgets everything stored in variables. To store high scores, or pass values to a new screen, use TinyDb. NOTE: TinyDb does not transfer information between programs.

[6] The DB in TinyDb stands for data base. A data base stores and looks up information.

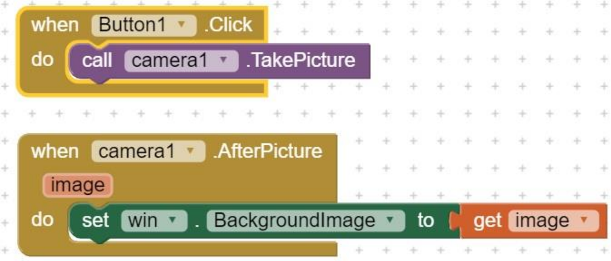
12. How to Use the Camera

a. Designer:

- i. User Interface: Button
- ii. Media: Camera

b. Blocks:

- i. Button:
  - 1. When Button Click:
    - a. Camera: call Camera.TakePicture
- ii. Camera:
  - 1. When Camera.AfterPicture:
    - a. Screen1:
      - i. Set Screen1.BackgroundImage to:
        - 1. Camera: Get Image



```
when Button1 .Click
do   call camera1 .TakePicture

when camera1 .AfterPicture
  image
do   set win . BackgroundImage to  get image
```

13. Speech Recognition[7]

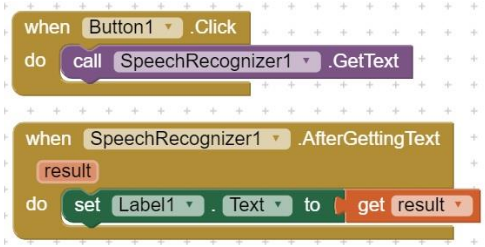
a. Designer:

- i. User Interface: Button
- ii. Media: SpeechRecognizer

b. Blocks:

- i. Built-in: When Button Click:
  - 1. SpeechRecognizer: call SpeechRecognizer:
    - a. Call SpeechRecognizer.GetText
- ii. SpeechRecognizer: When SpeechRecognizer.AfterGettingText:
    - a. Label: set Label to:
      - i. Call SpeechRecognizer: get Result (Put the mouse over result to get this.)

[7] This is a video explaining how to do this in the previous version of App Inventor. https://www.youtube.com/watch?time_continue=30&v=xGzNgzAmxCY



```
when Button1 .Click
do call SpeechRecognizer1 .GetText

when SpeechRecognizer1 .AfterGettingText
  result
do set Label1 . Text to get result
```

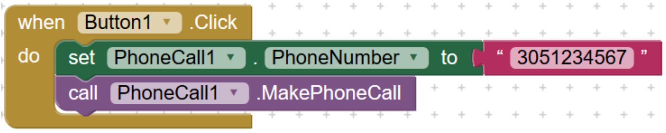
14. Auto Phone Dialing

a. Designer:

- i. User Interface: Button
- ii. Social: PhoneCall

b. Blocks:

- i. Button1: When Button Click
- ii. PhoneCall1: Set PhoneCall1 PhoneNumber to:
    - 1. Built-in: Text: Text: “3051234567”   (Enter the phone number to call.)



```
when Button1 .Click
do set PhoneCall1 . PhoneNumber to " 3051234567 "
   call PhoneCall1 .MakePhoneCall
```

15.Timing Things

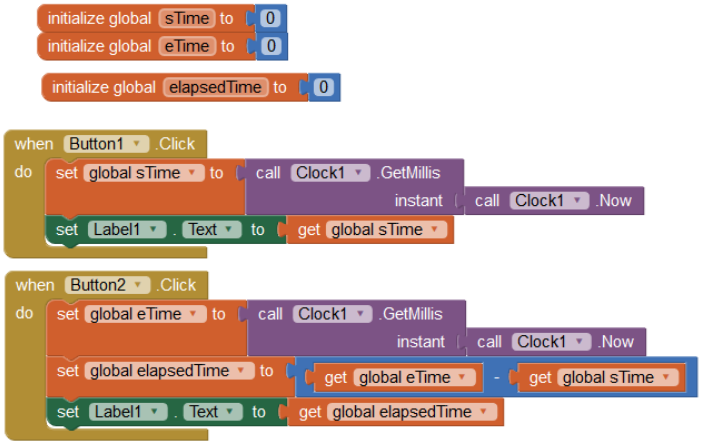
a. Designer:

- i. User Interface: Button1
- ii. User Interface: Button2
- iii. User Interface: Label
- iv. Sensors: Clock

b. Blocks:

- i. Built-in: Variables: Initialize global *name* to:
    - 1. Name = sTime
    - 2. To: Built-in: Math: 0
- ii. Built-in: Variables: Initialize global *name* to:
    - 1. *Name* = eTime
- 2. To: Built-in: Math: 0
- iii. Built-in: Variables: Initialize global *name* to:
    - 1. *Name* = elapsedTime
- 2. To: Built-in: Math: 0
- iv. Built-in: When Button1 Click:
    - 1. Button1: Set global sTime to:
        - a. Clock1: Call Clock1 Gmillis Instant:
            - i. Clock1: Call Clock1 .now
    - 2. Button2: Set global eTime to:
        - a. Clock1: Call Clock1 Gmillis Instant:

            i. Clock1: Call Clock1 .now
    3. Built-in: Variables: Set global elapsedTime to:
        a. Built-in: Math: ▯ - ▯ :
            i. Built-in: Variables: get ▼
                1. Global eTime
            ii. Built-in: Variables: get ▼ (subtracted number)
                1. Global sTime
    4. Label: Set Label1 Text:
        a. Built-in: Variables: get▼
            i. global elapsedTime



```
initialize global sTime to 0
initialize global eTime to 0

initialize global elapsedTime to 0

when Button1 .Click
do  set global sTime to  call Clock1 .GetMillis
                                  instant  call Clock1 .Now
    set Label1 . Text to  get global sTime

when Button2 .Click
do  set global eTime to  call Clock1 .GetMillis
                                  instant  call Clock1 .Now
    set global elapsedTime to  get global eTime - get global sTime
    set Label1 . Text to  get global elapsedTime
```

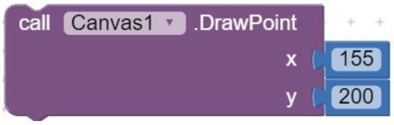
16. **Make a Graphics Area on the Screen** – often it is best to place the canvas in a horizontal or table layout.

   a. **Designer**

      i. Drawing and Animation: Canvas
         1. Canvas: property height (use fill parent or percentage)
         2. Canvas: property width (use fill parent or percentage)

   b. **Drawing a Dot** on the canvas

      i. **Blocks**
         1. Screen1: Canvas1: Call Canvas1.DrawPoint
            a. Call Canvas1.DrawPoint: X:
               i. Built-in: Math: 0 – change to distance from left
            b. Call Canvas1.DrawPoint: Y:
               i. Built-in: Math: 0 – change to distance from top

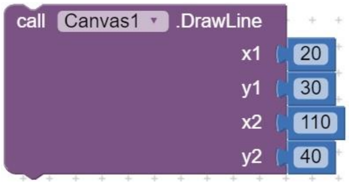


```
call Canvas1 .DrawPoint
                      x  155
                      y  200
```

   c. **Drawing a Line** on the canvas

i. Blocks

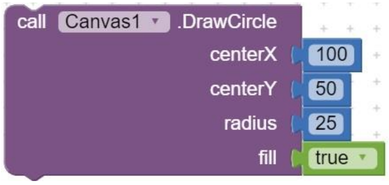
1. Screen1: Canvas1: Call Canvas1.DrawPoint
    a. Call Canvas1.DrawPoint: X1: (beginning point)
        i. Built-in: Math: 0 – change to distance from left
    b. Call Canvas1.DrawPoint: Y1: (beginning point)
        i. Built-in: Math: 0 – change to distance from top
    c. Call Canvas1.DrawPoint: X2: (ending point)
        i. Built-in: Math: 0 – change to distance from left
    d. Call Canvas1.DrawPoint: Y2: (ending point)
        i. Built-in: Math: 0 – change to distance from top



d. Drawing a Circle on the canvas

i. Blocks

1. Screen1: Canvas1: Call Canvas1.DrawPoint
    a. Call Canvas1.DrawPoint: centerX:
        i. Built-in: Math: 0 – change to distance from left
    b. Call Canvas1.DrawPoint: centerY:
        i. Built-in: Math: 0 – change to distance from top
    c. Call Canvas1.DrawPoint: radius:
        i. Built-in: Math: 0 – change to half the size of the circle



17. Make a Barcode Reader

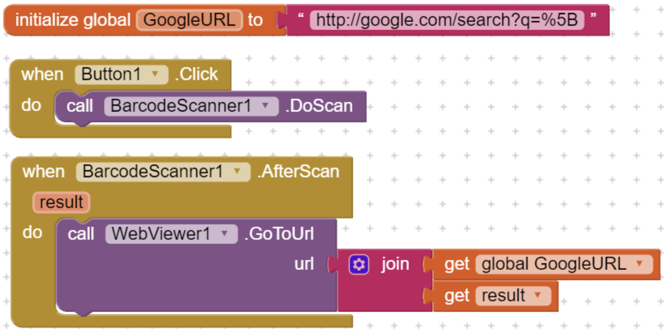
a. Designer:

i. User Interface: Button
ii. User Interface: WebViewer (Draw to large size)

b. Blocks:

i. Built-in: Variables: Initialize Global:
    1. Name: “GoogleURL”
    2. To:
        a. Built-in: Text: “ “:
            i. Type: “//http:google.com/search?q=%5B”
ii. When Button Click

1. Webviewer: Call WebViewer Goto URL:
    a. Built-in: Text: Join:
        i. Built-in: Variables: get ▼
            1. GoogleURL
        ii. Built-in: Variables: get ▼
            1. Webviewer: result



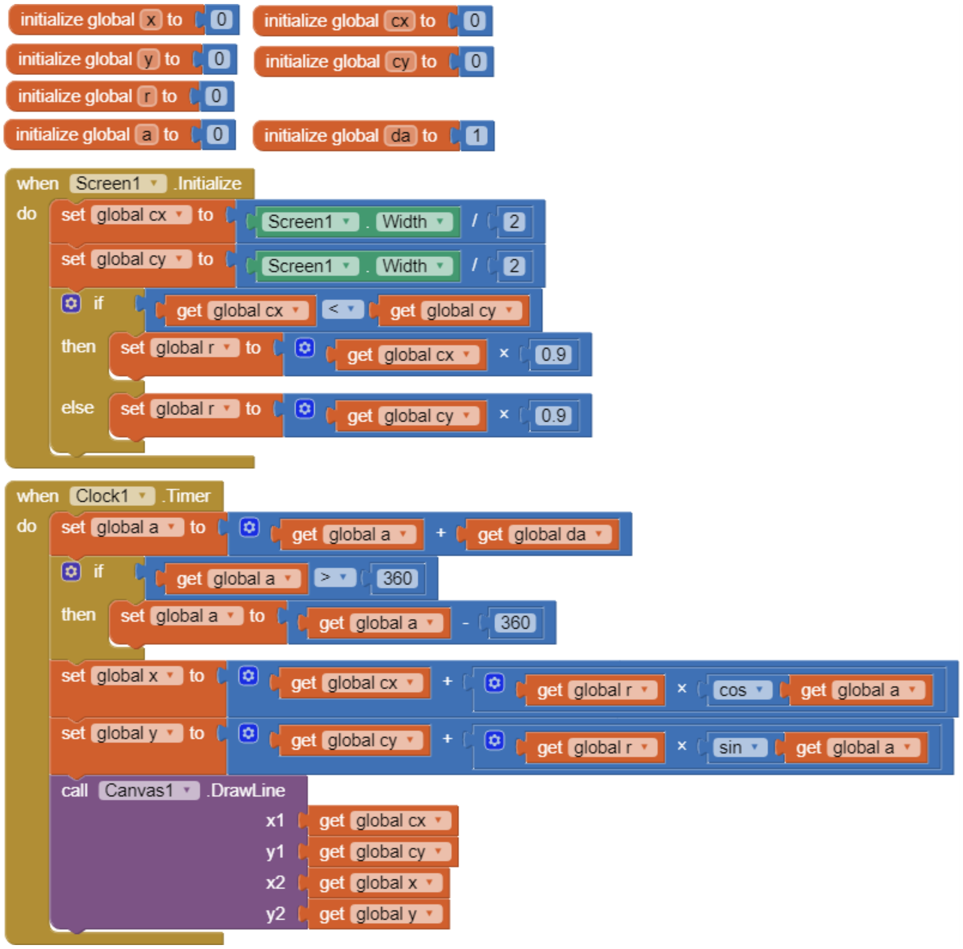
```
initialize global GoogleURL to " http://google.com/search?q=%5B "

when Button1 .Click
do  call BarcodeScanner1 .DoScan

when BarcodeScanner1 .AfterScan
  result
do  call WebViewer1 .GoToUrl
                     url   join   get global GoogleURL
                                  get result
```

18. Make a Spinning Line

a. Designer:

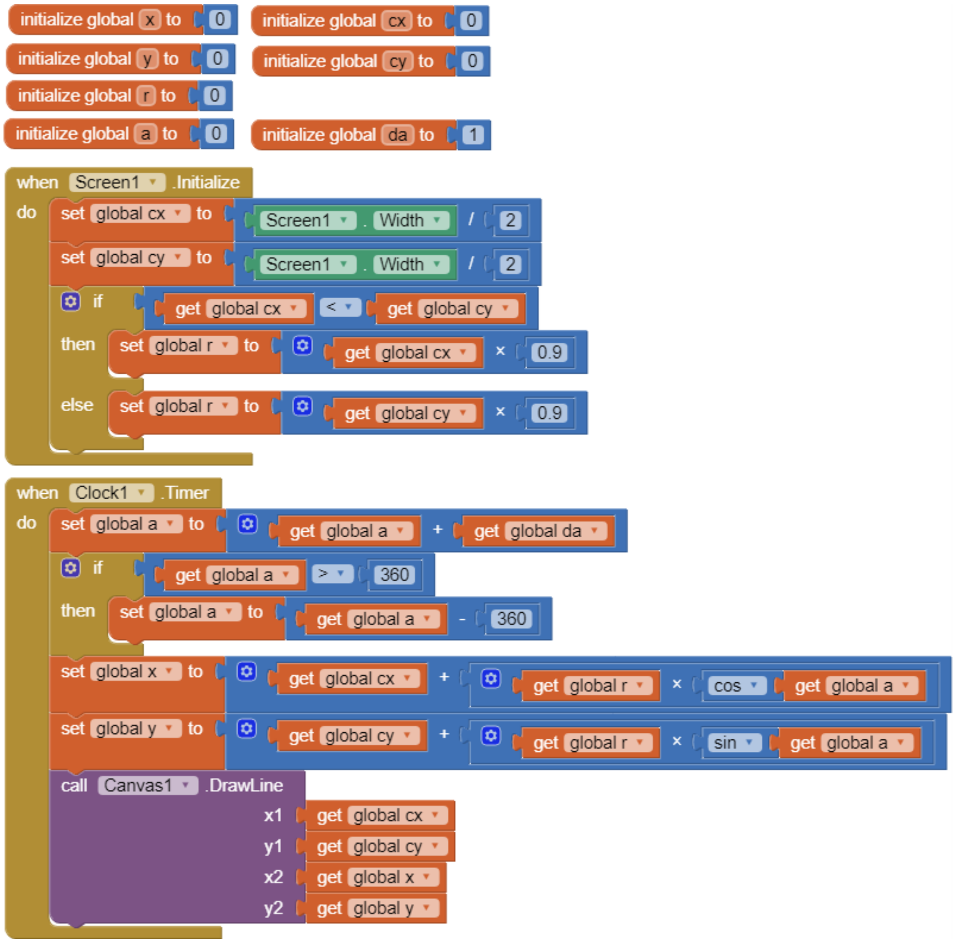
i. Drawings and animation
    1. Canvas
        a. Height to Fill Parent
        b. Width to Fill Parent

b. Blocks: Make as shown below.



```
initialize global x to 0        initialize global cx to 0
initialize global y to 0        initialize global cy to 0
initialize global r to 0
initialize global a to 0        initialize global da to 1

when Screen1 .Initialize
do   set global cx to  Screen1 . Width / 2
     set global cy to  Screen1 . Width / 2
     if    get global cx < get global cy
     then  set global r to  get global cx × 0.9
     else  set global r to  get global cy × 0.9

when Clock1 .Timer
do   set global a to  get global a + get global da
     if    get global a > 360
     then  set global a to  get global a - 360
     set global x to  get global cx + get global r × cos get global a
     set global y to  get global cy + get global r × sin get global a
     call Canvas1 .DrawLine
                      x1  get global cx
                      y1  get global cy
                      x2  get global x
                      y2  get global y
```

a. **Blocks for spin with color bounce:** Make as shown below.



```
initialize global x to 0        initialize global cx to 0
initialize global y to 0        initialize global cy to 0
initialize global r to 0
initialize global a to 0        initialize global da to 1

when Screen1 .Initialize
do  set global cx to  Screen1 . Width  / 2
    set global cy to  Screen1 . Width  / 2
    if    get global cx  <  get global cy
    then  set global r to  get global cx × 0.9
    else  set global r to  get global cy × 0.9

when Clock1 .Timer
do  set global a to  get global a + get global da
    if    get global a  >  360
    then  set global a to  get global a - 360
    set global x to  get global cx + get global r × cos get global a
    set global y to  get global cy + get global r × sin get global a
    call Canvas1 .DrawLine
                       x1  get global cx
                       y1  get global cy
                       x2  get global x
                       y2  get global y
```