This guide is a suggestion in how to learn Just Basic. It is intended for educators and assumes that Just Basic is already installed on the computer.

Most of this work is a summary of the tutorial to be found in the help menu. There is a lot of other good help and guidance that can be found on the web.

Most the programs can be found [here](http://justbasic.com) in a zipped file.

All good programmers plan what they are going to do before they complete a task. This ‘mission statement’ is normally written in plain English as an algorithm (or pseudo code). Because of the experiential nature of this tutorial this important step is highlighted in italics.

**Program 1 – Working out VAT**

In this first exercise we are going to create a program that:

1. Asks for an amount for goods before VAT is added;
2. Calculates a 20% VAT tax amount;
3. Displays the tax amount the total amount.

Type in the following into the main window

```
Input "Cost of Goods before VAT?"; amount
let tax = amount * 0.20
Print "Cost of Goods "; amount
Print "VAT "; tax
Print "Total Cost "; amount + tax
```

Click on [ ] will compile and run the program.

A new window should appear that allows you to enter an amount and then work out the VAT.

If you want to step through the program to see it working line by line click on [ ]. Move the windows around so you see both the output window and the debugger.

Click on [ ] to step through the program line by line.
Program 2 – Checking the input for a zero amount

We are going to complicate our program by making it repeat every time and checking to see if the amount entered is zero.

The algorithm becomes:

1. **Asks for an amount for goods before VAT is added**
2. **Check to see if amount is greater than zero**
   a. If not display help file
3. **Calculates a 20% VAT tax amount**
4. **Displays the tax amount the total amount**
5. **Returns to beginning of program**

![Image of Basic program code]

amount and tax are called **variables**. Note the way in which **label** [start] and the **sub routine** help are used.

If you enter a number such as 123.12 into this program it will give you an answer of 147.744. Whereas this is true it is not a good representation of money.

The INT() command can be used to round the number to 2 decimal places.

(int((tax*100)+0.5))/100 will round the VAT number to the correct number of decimal places.
Program 3 – A number guessing game

This is a simple guessing game. The computer will pick a number between 1 and 100. The user guesses the number in as few guesses as possible. The algorithm is:

1. Pick a number between 1 and 100
2. Print program title and give some instructions
3. Ask for the user to guess number
4. Tally the guess
5. If the guess is right go to step (9)
6. If the guess is too low tell the user to guess higher
7. If the guess is too high tell the user to guess lower
8. Go back to step (3)
9. Beep and tell the user how many guess it took to win
10. Ask the user whether to play again
11. If the user answers yes then clear the guess tally and goto step (1)
12. Give the user instructions on how to close the game window
13. End the program

Compare the program to the algorithm.

The ‘grey comments are remark statements and can be used if you come back to a program after a long while.
Program 4 – Using Arrays

Programs are often asked to process a lot of data. To do this they can use arrays. In this program we are going to store up to 20 numbers in an array and then work out their average. The algorithm is:

1. Define an array to hold up to 20 numbers
2. Ask user for a number
3. If number is 0 then step 8 to calculate average
4. Increase array number by 1
5. Add number to total
6. Check to see if 20 numbers are entered – if so step 8
7. Ask for another number – step 2
8. Display numbers entered and calculate average

Note the following

An array numbers has to be set up at the beginning of the program. The numbers in the array can be displayed by printing numbers(index)

Numbers are treated differently to other variables in BASIC.

A number variable can have a descriptor of just letter or more helpfully a name.

Word (including characters (such as !"£$%^&)) are called strings and need variables that have a $ after their name such as firstname$ or lastname$.

You can complete several task with strings – what do think this program does

input "First Name "; firstname$
input "Last Name "; lastname$

fullname$ = firstname$ + " " + lastname$

print
print fullname$
Print
print "Your first name is "; len(firstname$); " " characters long"
print "Your last name is "; len(lastname$); " " characters long"
print "Your full name is "; len(fullname$); " " characters long"

print
print "Your full name in lower case "; lower$(fullname$)
print "Your full name in upper case "; upper$(fullname$)
Teachers guide to starting to learn Just Basic

```
' AVERAGE.BAS
  ' Accept some numbers from the user, then total and average them
    dim numbers(20) ' define an array to take up to 20 numbers

    print "AVERAGE.BAS"
    print
    print "Enter up to 20 non-zero values."
    print "A zero or blank entry ends the series."

    [entryLoop] ' loop around until a zero entry or until index = 20
      print "Entry "; index + 1; ' get the user's entry
      input entry
      if entry = 0 then [endSeries] ' quit if entry is zero or blank

      index = index + 1 ' add one to index
      numbers(index) = entry ' set the specified array item to be entry
      total = total + entry ' add entry to the total

      if index = 20 then [endSeries] ' if 20 values were entered, exit loop
      goto [entryLoop] ' go back and get another entry

    [endSeries] ' entries are finished
      entryCount = index ' Set entryCount to index
      if entryCount = 0 then print "No Entries." : goto [quit]

      print "Entries completed."
      print
      print "Here are the ": entryCount; " entries:"
      print "-----------------------------"

      ' This loop displays each entered value in turn.
      ' Notice that we re-use the index variable. It
      ' can be confusing to use a new variable for each
      ' new loop.
      index = 0

    [displayLoop]
      index = index + 1
      print "Entry ": index; " is ": numbers(index)
      if index < entryCount then [displayLoop]

      ' Now display the total and average value
      print
      print "The total is ": total
      print "The average is ": total / entryCount

    [quit]

    end
```

Ready!
Program 5 – Storing the information on the computer

`AGES_STORE.BAS`
`
'Accept some names and ages from the user, then total and average them

dim numbers(20)
dim names$(20)
print "AGES.BAS"
print

'loop up to 20 times, getting numbers
print "Enter up to 20 non-zero values."
print "A zero or blank entry ends the series."

[entryLoop] 'loop around until a zero entry or until index = 20

'get the user's name and age
print "Entry "; index + 1;
input name$
if name$ = "" then [endSeries] 'quit if name$ is blank

print "Age ";
input age

index = index + 1 'add one to index
names$(index) = name$ 'set the specified array item to be name$
numbers(index) = age 'set the specified array item to be age

total = total + age 'add entry to the total

if index = 20 then [endSeries] 'if 20 values were entered, exit loop
goto [entryLoop] 'go back and get another entry

[endSeries] 'entries are finished

'Set entryCount to index

entryCount = index
if entryCount = 0 then print "No Entries." : goto [quit]

print "Entries completed."
print
print "Here are the "; entryCount; " entries:"
print "-----------------------------"

'This loop displays each entered value in turn.
'Notice that we re-use the index variable. It can be confusing to use
'a new variable for each new loop.

for index = 1 to entryCount
print "Entry "; index; "; names$(index); "; age "; numbers(index)

next index

'Write the data into ages.dat
open "C:\Documents and Settings\igover\My Documents\just basic files\ages.dat" for output as #ages
for index = 1 to entryCount
print #ages, names$(index)
print #ages, numbers(index)
next index
close #ages

'Now display the total and average value
print
print "The total age is "; total
print "The average age is "; total / entryCount

[quit] end

This program extends the previous algorithm by adding names and then storing the data on a disc.

Note the ways the names have to be stored in a variable with a dollar $ at the end – names$.

This is the part of the program that stores the data.
open "path.myfile.txt" for output as #myHandle

You can see the OUTPUT mode is specified. The last item on the line is #myHandle. It is a name (called a file handle).
The file must be closed.

Program 6 - Reading the file

The algorithm to read the file is:
1. Open the file
2. Check to see if it is the last one in the file then step 4
3. Read the entry and store in an array
4. Read next entry – step 2
5. Print entries
6. Calculate average

This program will read the ages.dat file.

The `eof(#ages)` checks to see if the end of the file has been reached. If it has it returns a -1.

The use of the #ages is the file handle.
Program 7 – Opening windows

Just Basic can be used to control windows.

This program:
1. Opens a window with defined size
2. Displays a button
3. When clicked displays message

The button statement needs a little unpacking.

You must give a handle for the button – in this case #myFirst.ok

The [okClicked] tells Basic to go to the [okClicked] routine

UL tells Basic to put the button at the Upper Left of the window. You can also have UR, LL and LR. The numbers are the position from the corner.
Program 8 – More window formatting and closing

In this development of the last program we are going to follow the following algorithm:

1. Open a window
2. Display
   a. OK Button
   b. Textbox
   c. Instructions
3. If Ok clicked then display contents
4. Check to see if close window is used

Note the use of statictext to position instruction.

The “trapclose instruction will detect if the window is being closed and move to the label [quit]
Program 9 – Drawing a Square

This program opens a graphic window as #draw and then draws a square.

‘Flush’ fixes the drawing.

Program 10 – Adding a bitmap

This program loads a bitmap into window.

There are many other things that you can do with graphics – referring to the help file and then looking at week 5 will help you explore this area.

There is a well established Just Basic forum at http://justbasic.conforums.com/index.cgi that has an amazing amount of help and information.
Teachers guide to starting to learn Just Basic

Program 11 – Bouncing Ball

This program demonstrates what can be completed with graphics.

It introduces several new commands.

See if you can follow it.
# Teachers guide to starting to learn Just Basic

## Full list of commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABS( n )</td>
<td>absolute value of n</td>
</tr>
<tr>
<td>ACS( n )</td>
<td>arc-cosine of n</td>
</tr>
<tr>
<td>&quot;addsprite&quot;</td>
<td>sprite command to add a sprite</td>
</tr>
<tr>
<td>AND</td>
<td>bitwise, boolean AND operator</td>
</tr>
<tr>
<td>APPEND</td>
<td>purpose parameter in file open statement</td>
</tr>
<tr>
<td>AS</td>
<td>used in OPEN statements</td>
</tr>
<tr>
<td>ASC( s$ )</td>
<td>ascii value of s$</td>
</tr>
<tr>
<td>ASN( n )</td>
<td>arc-sine of n</td>
</tr>
<tr>
<td>ATN( n )</td>
<td>arc-tangent of n</td>
</tr>
<tr>
<td>&quot;!autoresize&quot;</td>
<td>texteditor command to relocate control automatically</td>
</tr>
<tr>
<td>&quot;autoresize&quot;</td>
<td>graphics command to relocate control automatically</td>
</tr>
<tr>
<td>&quot;backcolor&quot;</td>
<td>graphics command to set background color</td>
</tr>
<tr>
<td>&quot;background&quot;</td>
<td>sprite command to set background image</td>
</tr>
<tr>
<td>BackgroundColor$</td>
<td>sets or returns background color for window</td>
</tr>
<tr>
<td>BEEP</td>
<td>play the default system wave file</td>
</tr>
<tr>
<td>BINARY</td>
<td>purpose parameter in file open statement</td>
</tr>
<tr>
<td>Bitwise Operations</td>
<td>modify bit patterns in an object</td>
</tr>
<tr>
<td>BMPBUTTON</td>
<td>add a bitmap button to a window</td>
</tr>
<tr>
<td>BPMSAVE</td>
<td>save a bitmap to a disk file</td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>evaluates to true or false</td>
</tr>
<tr>
<td>&quot;box&quot;</td>
<td>graphics command to draw box</td>
</tr>
<tr>
<td>&quot;boxfilled&quot;</td>
<td>graphics command to draw filled box</td>
</tr>
<tr>
<td>BUTTON</td>
<td>add a button to a window</td>
</tr>
<tr>
<td>BYREF</td>
<td>passes an argument to a subroutine or function by reference</td>
</tr>
<tr>
<td>CALL</td>
<td>call a user defined subroutine</td>
</tr>
<tr>
<td>CASE</td>
<td>specifies a value for select case statement</td>
</tr>
<tr>
<td>CHECKBOX</td>
<td>add a checkbox to a window</td>
</tr>
<tr>
<td>CHRS( n )</td>
<td>return character of ascii value n</td>
</tr>
<tr>
<td>&quot;circle&quot;</td>
<td>graphics command to draw circle</td>
</tr>
<tr>
<td>&quot;circlefilled&quot;</td>
<td>graphics command to draw filled circle</td>
</tr>
<tr>
<td>CLOSE #h</td>
<td>close a file or window with handle #h</td>
</tr>
<tr>
<td>CLS</td>
<td>clear a program's mainwindow</td>
</tr>
<tr>
<td>&quot;cls&quot;</td>
<td>graphics command to clear drawing area</td>
</tr>
<tr>
<td>&quot;tcis&quot;</td>
<td>text command to clear texteditor</td>
</tr>
<tr>
<td>&quot;color&quot;</td>
<td>graphics command to set pen color</td>
</tr>
<tr>
<td>COMBOBOX</td>
<td>add a combobox to a window</td>
</tr>
<tr>
<td>ComboboxColor$</td>
<td>sets or returns combobox color</td>
</tr>
<tr>
<td>CommandLine$</td>
<td>contains any command line switches used on startup</td>
</tr>
<tr>
<td>CONFIRM</td>
<td>opens a confirm dialog box</td>
</tr>
<tr>
<td>&quot;!contents&quot;</td>
<td>text command to replace contents of texteditor</td>
</tr>
<tr>
<td>&quot;!contents?”</td>
<td>text command returns contents of texteditor</td>
</tr>
<tr>
<td>&quot;!copy”</td>
<td>text command to copy text to clipboard</td>
</tr>
<tr>
<td>COS( n )</td>
<td>cosine of n</td>
</tr>
<tr>
<td>&quot;!cut”</td>
<td>text command to cut text and copy to clipboard</td>
</tr>
<tr>
<td>DATA</td>
<td>adds data to a program that can be read with the READ statement</td>
</tr>
<tr>
<td>DATE$( )</td>
<td>return string with today's date</td>
</tr>
<tr>
<td>DefaultDir$</td>
<td>a variable containing the default directory</td>
</tr>
<tr>
<td>&quot;delsgment”</td>
<td>graphics command to delete drawing segment</td>
</tr>
<tr>
<td>Dialog</td>
<td>window type</td>
</tr>
<tr>
<td>DIM array( )</td>
<td>set the maximum size of a data array</td>
</tr>
<tr>
<td>&quot;discard”</td>
<td>graphics command to discard unflushed drawing</td>
</tr>
<tr>
<td>DisplayWidth</td>
<td>a variable containing the width of the display</td>
</tr>
<tr>
<td>DisplayHeight</td>
<td>a variable containing the height of the display</td>
</tr>
<tr>
<td>&quot;down”</td>
<td>graphics command to lower pen</td>
</tr>
<tr>
<td>&quot;drawbmp”</td>
<td>graphics command to display a bitmap</td>
</tr>
<tr>
<td>Drives$</td>
<td>special variable, holds drive letters</td>
</tr>
<tr>
<td>DO LOOP</td>
<td>performs a looping action until/while a condition is met</td>
</tr>
<tr>
<td>DUMP</td>
<td>force the LPRINT buffer to print</td>
</tr>
<tr>
<td>&quot;ellipse”</td>
<td>graphics command to draw an ellipse</td>
</tr>
</tbody>
</table>
**Teachers guide to starting to learn Just Basic**

- "ellipsefilled" graphics command to draw a filled ellipse
- ELSE used in block conditional statements with IF/THEN
- ENABLE make a control active
- END marks end of program execution
- END FUNCTION signifies the end of a function
- END IF used in block conditional statements with IF/THEN
- END SELECT signals end of SELECT CASE construct
- END SUB signifies the end of a subroutine
- EOF( #h ) returns the end-of-file status for #h
- EXIT exits a looping structure, sub or function
- EXIT FOR terminate a for/next loop before it completes
- EXIT WHILE terminate a while/wend loop before it completes
- EXP( n ) returns e^n
- FIELD #h, list... sets random access fields for #h
- FILEDIALOG opens a file selection dialog box
- FILES returns file and subdirectory info
- "fill" graphics command to fill with color
- "font" set font as specified
- ForegroundColor$ sets or returns foreground color for window
- FOR...NEXT performs looping action
- FUNCTION define a user function
- GET #h, n get random access record n for #h
- "getbmp" graphics command to capture drawing area
- GLOBAL creates a global variable
- "go" graphics command to move pen
- GOSUB label call subroutine label
- "goto" graphics command to move pen
- GOTO label branch to label
- GRAPHICBOX add a graphics region to a window
- GROUPBOX add a groupbox to a window
- Graphics window type
- Graphics Commands a detailed summary of graphics commands in Just BASIC
- "home" graphics command to center pen
- IF THEN perform conditional action(s)
- Inkey$ contains a character or keycode from a graphics window
- INPUT get data from keyboard, file or button
- INPUT$( #h, n ) get n chars from handle #h, or from the keyboard
- INPUT purpose parameter in file open statement
- INSTR(a$,b$,n) search for b$ in a$, with optional start n
- INT( n ) integer portion of n
- JOY- global variables containing joystick information read by readjoystick command
  - Joy1x, Joy1y, Joy1z, Joy1button1, Joy1button2
  - Joy2x, Joy2y, Joy2z, Joy2button1, Joy2button2
- KILL s$ delete file named s$
- LEFT$( s$, n ) first n characters of s$
- LEN( s$ ) length of s$
- LET var = expr assign value of expr to var
- "line" graphics command to draw line
- "!line" text command to return text from specified line in texteditor control
- "!lines?" text command to return number of lines in texteditor control
- LINE INPUT get next line of text from file
- LISTBOX add a listbox to a window
- ListBoxColor$ sets or returns listbox color
- LOADBMP load a bitmap into memory
- LOC(#handle) return current binary file position
- LOCATE locate text in the mainwindow
- LOF( #h ) returns length of open file #h or bytes in serial buffer
- LOG( n ) returns the natural logarithm of n
- LOWERS$( s$ ) s$ converted to all lowercase
- LPRINT print to hard copy
Teachers guide to starting to learn Just Basic

MAINWIN
- set the width of the main window in columns and rows

MENU
- adds a pull-down menu to a window

MID$( )
- return a substring from a string

MIDIPOS( )
- return position of play in a MIDI file

MKDIR( )
- make a new subdirectory

MOD
- returns the result of integer division

"!modified?"
- text command to return modified status

NAME a$ AS b$
- rename file named a$ to b$

NEXT
- used with FOR

NOMAINWIN
- keep a program's main window from opening

"north"
- graphics command to set the current drawing direction

NOT
- logical and bitwise NOT operator

NOTICE
- open a notice dialog box

ONCOMERROR
- set an error handler for serial communications

ON ERROR
- set an error handler for general program errors

OPEN
- open a file or window

OPEN "COMn:...
- open a communications port for reading/writing

OR
- logical and bitwise OR operator

"!origin"
- text command to set origin

"!origin?"
- text command to return origin

OUTPUT
- purpose parameter in file open statement

"!paste"
- text command to paste text from clipboard

"pie"
- graphics command to draw pie section

"piefilled"
- graphics command to draw filled pie section

"place"
- graphics command to locate pen

Platform$
- special variable containing platform name

PLAYWAVE
- plays a *.wav sound file

PLAYMIDI
- plays a *.midi sound file

"posxy"
- graphics command to return pen position

"print"
- graphics command to print hard copy

PRINT
- print to a file or window

PrinterFont$
- returns or sets the font used with LPRINT

PROMPT
- open a prompter dialog box

PUT #h, n
- puts a random access record n for #h

RADIOBUTTON
- adds a radiobutton to a window

RANDOM
- purpose parameter in file open statement

READ
- reads information from DATA statements

REDIM
- redimensions an array and resets its contents

"redraw"
- graphics command to redraw segment

REM
- adds a remark to a program

RESTORE
- sets the position of the next DATA statement to read

RETURN
- return from a subroutine call

RIGHT$( s$, n )
- rightmost characters of s$

RMDIR( )
- remove a subdirectory

RND( n )
- returns a random number

"rule"
- graphics command to set drawing rule

RUN s$, mode
- run external program s$, with optional mode

SCAN
- checks for and dispatches user actions

SEEK #h, fpos
- set the position in a file opened for binary access

"segment"
- graphics command to return segment ID

SELECT CASE
- performs conditional actions

"!selectall"
- text command to highlight all text

"!selection?"
- text command to return highlighted text

"set"
- graphics command to draw a point

"setfocus"
- set input focus to control or window

SIN( n )
- sine of n

"size"
- graphics command to set pen size

SPACE$( n )
- returns a string of n spaces

Sprites
- all about using sprites in Just BASIC
Teachers guide to starting to learn Just Basic

SOR(n)  details about getting the square root of a number
STATICTEXT  add a statictext control to a window
STOP  marks end of program execution
STOPMIDI  stops a MIDI file from playing
STR$( n )  returns string equivalent of n
SUB  defines a subroutine
TAB( n )  causes tabular printing in mainwin
TAN( n )  tangent of n
Text  window type
Text Commands  a detailed summary of text window commands in Just BASIC
TEXTBOX  add a textbox (entryfield) to a window
TextboxColor$  sets or returns textbox color
TEXTEDITOR  add a texteditor widget to a window
TexteditorColor$  sets or returns texteditor color
TIME$( )  returns current time as string
TIMER  manage a Windows timer
"!trapclose"  text command to trap closing of text window
"trapclose"  trap closing of window
TRIM$( s$ )  returns s$ without leading/trailing spaces
"turn"  graphics command to reset drawing direction
TXCOUNT(#handle)  gets number of bytes in serial communications queue
UNLOADBMP  unloads a bitmap from memory
"up"  graphics command to lift pen
UPPER$( s$ )  s$ converted to all uppercase
USING()  performs numeric formatting
UpperLeftX  specifies the x part of the position where the next window will open
UpperLeftY  specifies the y part of the position where the next window will open
VAL( s$ )  returns numeric equivalent of s$
Version$  special variable containing LB version info
WAIT  stop and wait for user interaction
"when"  graphics command to trap mouse and keyboard events
WHILE...WEND  performs looping action
Window  window type
WindowWidth  specifies the width of the next window to open
WindowHeight  specifies the height of the next window to open
WORDS( s$, n )  returns nth word from s$
XOR  logical and bitwise XOR operator