Microsoft Excel Advanced Class June 2017

Norwalk Community College — Extended Studies
Rich Malloy, Instructor

Session 2 — A Database for a Charity

Table of Contents

About the Project

A charity is basically a membership organization. Like all such organizations, it collects dues from its members. What makes charities more interesting is that they also request and accept various types of donations, from both members and nonmembers. To help these donors with their income taxes, charities must also acknowledge all donations with a thank-you letter.

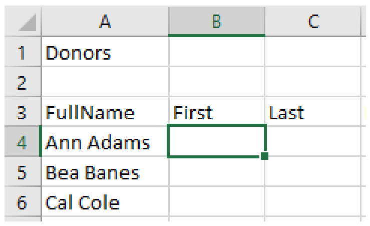
In this project, we will illustrate the use of many advanced Excel tools by setting up a small table of donors and donations. These tools will help us keep track of the donations for each member. We will also set up some macros to automate the process of sending out thank-you letters and form letters requesting additional contributions. Note that by simply changing “donors” to “members” and “donations” to “dues,” this database could be used for membership organizations as well as charities.

For this project, we’ll be using the file Charity_Database_Draft.xlsx, which can be found in the Training Files section at techhelptoday.com (direct shortlink: http://wp.me/P7Dy8G-y)

A. How to Parse Names

When they fill out a form, most people are more comfortable writing their name in the usual style: first name followed by last name. Database managers, however, like to separate the last name from the first so that the names can be sorted by last name. Excel offers several ways to do this.

Scenario: In the Donors worksheet, we have a column of full names (e.g., “Ann Ames”) followed by two blank columns, First and Last. Your job is to separate the first name from the last name and put them into the appropriate columns.



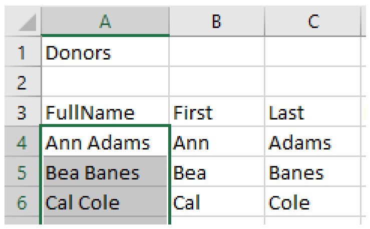
	A	B	C
1	Donors		
2			
3	FullName	First	Last
4	Ann Adams		
5	Bea Banes		
6	Cal Cole		

Text to Columns

The traditional way to parse names is to use the “Convert Text to Columns Wizard.” The procedure is fairly simple:

- Select the full names (A4:A14, excluding the label)
- Click: Data > Text to Columns
- Confirm that the option button for Delimited is chosen and click Next
- Check the box for the appropriate delimiter separating the terms (in this case, Space)
- Click: Next
- Set the destination for the separated names (in this case, **B4**)
- Click: Finish

The result looks something like the following:



	A	B	C
1	Donors		
2			
3	FullName	First	Last
4	Ann Adams	Ann	Adams
5	Bea Banes	Bea	Banes
6	Cal Cole	Cal	Cole

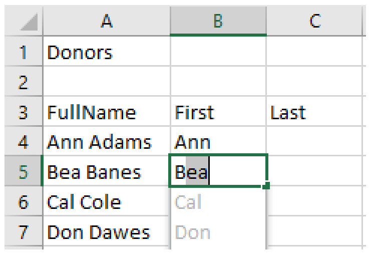
The problem with this tool is that if you do not set the destination correctly, you will overwrite the full names. But fortunately, there’s the Undo button.

Flash Fill

This is an impressive new tool that arrived with Excel 2013. This feature recognizes how you are separating or rearranging data and then repeats it for a whole column. The procedure is so natural that it may shock you the first time it happens:

- In cell B2, enter: **Ann**
- In cell B3, type: **B**

Here is what you will usually see:



	A	B	C
1	Donors		
2			
3	FullName	First	Last
4	Ann Adams	Ann	
5	Bea Banes	Bea	
6	Cal Cole	Cal	
7	Don Dawes	Don	

Excel guesses the pattern and then tentatively fills in the column. If you like what you see, just press the Enter key.

Sometimes Flash Fill does not kick in automatically. In that case you can force the issue by clicking cell B4 and clicking the Flash Fill in the Data Tools group of the Data tab.

Note that you can also use Flash Fill to reorganize and reformat data: E.g., you can convert the following:

Bea Banes	→	Banes, Bea
2035551234	→	(203) 555-1223

Parsing with Formulas

Flash Fill is impressive and easy to use, but it is not completely automatic. Each time a new batch of full names comes in, you have fire up Flash Fill again, and even the most wonderful thing can get tedious with enough repetition. Isn't there a way that Excel could parse names completely automatically? Of course.

Excel has several text functions that can be coaxed into doing the job. There are LEFT and RIGHT functions that can strip off the first and last names respectively. But to use those functions, we will need to know how many characters to skim off each side of the full name. Well, we can figure out how long the first and last names are if we knew the position of the space between them. And for that we will need to use FIND function, which will tell us the position of any string of text characters within another string. The FIND function can tell us, e.g., that the position of the space is 4, which tells us that the first name must be only 3 characters long. To determine the length of the last name, we could use the LEN function to give us the length of the whole name and then subtract the position of the space.

For this task, we'll be using two or three functions, one or two of which will be nested inside other functions. It may be a little hard to follow at first, but it is a technique that is very common in business spreadsheets.

Here are the formulas:

For cell B4 (FirstName): **=LEFT(A4, FIND(" ", A4) - 1)**

- First, Excel finds the position of the space in "Ann Adams", which is 4. (In an Excel formula, a space is represented by a space between two quotation marks.)
- The length of the first name is one less than the position of the space, so we subtract 1, giving us 3.
- Then the LEFT function strips off the leftmost 3 characters ("Ann").

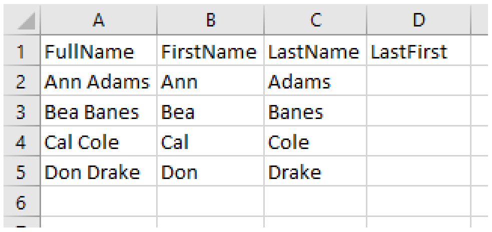
For cell C4 (LastName): **=RIGHT(A4, LEN(A4) - FIND(" ", A4))**

- First, Excel evaluates the length of "Ann Adams" as 9.
- Again, it finds the position of the space in "Ann Ames", which is 4.
- It subtracts 4 from 9 to give 5, the length of the last name.
- Then the RIGHT function strips off the rightmost 5 characters ("Adams").

A little hard follow, but you only have to set it up once.

Combining Names with the Ampersand ("&")

Once we split a full name into its first and last components, it is often helpful to recombine them in a "Last, First" format, as in "Adams, Ann". This shows us the full name, yet still enables us to sort our names by last name. Here is the scenario:



	A	B	C	D
1	FullName	FirstName	LastName	LastFirst
2	Ann Adams	Ann	Adams	
3	Bea Banes	Bea	Banes	
4	Cal Cole	Cal	Cole	
5	Don Drake	Don	Drake	
6				

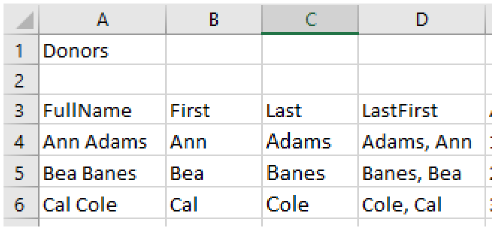
As mentioned above, we could use the Flash Fill to combine and reorganize names, but there are two ways we can use formulas to do the same thing automatically:

In cell D2, we could use either of the following approaches:

- The "&" operator: **= C4 & ", " & B4**
- The CONCATENATE function: **= CONCATENATE(C4, ", ", B4)**

Note that in the above formulas there is a comma and a space between the quotation marks. Personally, I prefer the "&" operator. It is shorter and there are fewer commas — always a good thing.

After we copy either formula down the rest of the column and widen the column, we have:



	A	B	C	D
1	Donors			
2				
3	FullName	First	Last	LastFirst
4	Ann Adams	Ann	Adams	Adams, Ann
5	Bea Banes	Bea	Banes	Banes, Bea
6	Cal Cole	Cal	Cole	Cole, Cal

B. Format the Donors Table

Let's fix up the table on the Donors worksheet. The first step is to adjust the column widths. Columns E and F need to be wider, and G could be narrower. Next, we should correctly format the Zip Codes that are used in the U.S. In this region of the country, most zip codes start with zero, but Excel thinks they are numbers (which they look like) and lops off the leading zero thinking it is unnecessary. Because Zip Codes are not really numbers (we don't add them or subtract them), they should be formatted as text, and the latest version of Excel is much more friendly to the idea of numbers formatted as text.

Formatting Zip Codes

- If necessary, insert a temporary dummy column to the right of the Zip Code column
- In the cell to the right of the first Zip Code, enter the formula: **= TEXT(I4, "00000")**
  - (Note that the formula refers to cell "eye" 4, not the number fourteen.)
  - This reformats the Zip Code as the text equivalent of the number with five mandatory digits
  - Note that the result is aligned on the left side of the cells
- Copy the formula to the rest of the cells in the column

- Copy the new Zip Codes
- Right-click the first of the old Zip Codes and Paste as Value
- Delete the now unnecessary cells containing the TEXT function (or delete the temporary column you created)

Format as Table

An Excel Table is a relatively new feature that has several great benefits. The most obvious is a great design. But it also has some other features that we will mention later. Here is the simple procedure for turning a list or range of ordinary cells into one of these amazing formatted Tables:

First: Note that the Donors list is well behaved (i.e., no blank rows or columns and only one row of column headers). If the list was not so well behaved, we would have to select the whole table instead of selecting just one cell as we do below.

- Click any cell in the Donors list.
- In the Home tab, click Format as Table
- Confirm that Excel has chosen the correct range of cells and Header Row by clicking OK
- Choose a suitable table design (my favorites are in the Medium group)
- Click any cell in the table
- Click the table tools Designs tab at the top
- At the left end of the Ribbon change the name of the table to **DonorsTable** (no spaces)
-

The formatted table now has Filter buttons in each of the column header cells. In some tables such as this one, the filters are not very useful. Here is how to hide them:

- In the Data tab, click the large button: **Filter**

C. Format the Donations Table

What is good for the Donors list will be even better for the Donations list. So now let's format the donations list as a table.

Again, note that, like the Donors list, the list of Donations is also well behaved.

- Click any cell in the Donations list.
- In the Home tab, click Format as Table
- Confirm that Excel has chosen the correct range of cells by clicking OK
- Choose a suitable table design (my favorites are in the Medium group)
- Click any cell in the table
- Click the tab: Table Tools Design
- At the left end of the ribbon change the name of the table to **DonationsTable** (note, no spaces)

As above, we can remove the filters buttons, but in this table the filter buttons can actually be very useful. So, we'll leave them in.

D. Insert Slicers to Filter Data

Slicers are a great way to filter the data in a formatted table. They are much more convenient than the filter buttons.

If you have a formatted table, it is very easy to insert slicers.

- First, insert some blank rows above the table to make room for your slicers
- Clicking any cell of the table
- Click the **Table Tools Design** tab, which appears when you click in a formatted table.
- Click the **Insert Slicers** button
- Click the check boxes for **Reason**
- Adjust the size and position of the slicers as desired

Note that when a slicer is selected, a new tab appears: The **Slicer Tools Options** tab. One thing you might do here is change the slicer into a two-column format.

Once the slices are present, you can quickly filter your tables just by clicking the appropriate button. If you want to include two or more different categories, just hold down the control key and click the second and third categories. This is a great tool. Once you use it, you will want to use it on all your tables.

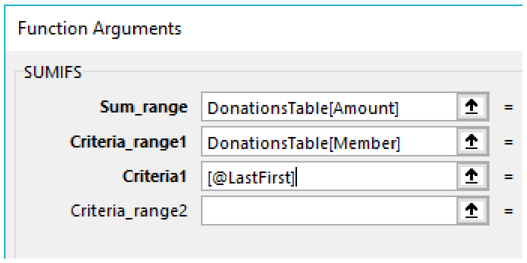
E. Summarize Donations in the Donors Table with SUMIFS

After recording the donations, we would like to summarize them along a few different lines. E.g., how much Did each of the donors donate? How much did they donate any given year? How much do they donate through the special programs that the charity has set up?

We could use a pivot table to gather this information. But a more flexible and powerful approach would be to use the summits function. Dysfunction is a little harder to use in appearance table, but you only have to set it up once, and it is Extremely powerful.

The first thing we want to do is to set up a column in the Donations table, which would list total donations for each of the donors.

- In the Donors worksheet, click cell I3, that is, the cell just to the right of the label Zip
- Enter the label **Total Donations**
- In this cell below it we will enter a formula
- Type: **= sum**
- In the list, double-click: **SUMIFS**
- Click the Insert Function button (***fx***)
- Click in the box for **Sum_range**
- Go to the Donations worksheet and select all the cells in the Amount column
- Click the box for **Criteria_range1**
- Go to the Donations table and select all the cells under the LastFirst label
- Click the box for **Criteria1**
- In the Donors table, click the first cell under the LastFirst label


Function Arguments

SUMIFS

Sum_range	DonationsTable[Amount]
Criteria_range1	DonationsTable[Member]
Criteria1	[@LastFirst]
Criteria_range2	

- That's it. Now click the OK button to finish it

Because we're using a formatted table, the formula is automatically copied to all the cells in the column

Using the SUMIFS function with two criteria

Now we want to know how much to each of the donors gave during the specific year. That means we will have to add a new criterion or condition to the SUMIFS function. But first we'll have to adding new column to the Donations table that we can use for our new criterion.

- In the Donations sheet, click the cell to the right of the Amount label
- Enter the label Year
- Click the cell below the label
- We will now create a formula that will give us the year for each of the dates.
- Type the following formula: **=YEAR(**
- Then click the first date under the Date label
- Type a closing parentheses and then press the Enter key

The completed formula should be:

=YEAR([@Date])

Now we can add up donations for each donor for the year 2016:

- In the Donors sheet, click in cell J3, or just to the right of the label Total Donations
- Enter the label: **2016 Donations**
- As above, you can use the Insert Function button to start using the SUMIFS function
- Again, we will put the **DonationsTable[Amount]** in the Sum_range box
- And in the **Criteria_range1** box we'll put **DonationsTable[LastFirst]**
- And in the **Criteria1** box we'll just click cell **D4** in the Donors table
- Click in the **Criteria_range2** box and then go to the Donations sheet
- Select all the cells under the Year column label
- In the **Criteria2** box just type the number **2016**
- Click the OK button

F. Use Conditional Formatting and the VLOOKUP Function

In any group, some people will contribute a lot more than the others. We want to be sure we know who these folks are so that we can give them proper recognition.

Use Conditional Formatting with a Rule

We will set up a rule to highlight total contributions above $75:

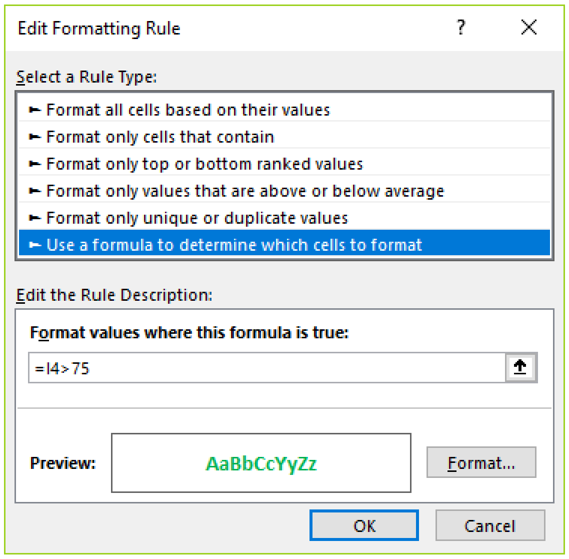
- In the Donors table, select the cells under the total Donations label
- In the Home tab, click the button **Conditional Formatting**
- Choose **Highlight Cell Rules**
- Choose **Greater than**
- In the **Greater than** dialog box type **75** in the first box
- Click the list arrow for the **Format** options
- In the list of options, choose **Custom Format**
- In the **Format Cells** dialog box choose a fine style of Bold in the color of green
- Click OK and click OK again

All total donations above $75.00 will appear as bold and green.

Use Conditional Formatting with a Formula

But now we want to highlight not just the amount of the donations you also want to highlight the name of the person who donated so much money.

- Selected all of the table cells under the label **LastFirst**
- Click the **Conditional Formatting** button
- Choose: **Highlight Cell Rules**
- Choose: **More Rules…**
- In the New Formatting Rule dialog box, choose a Rule Type of: **Use a formula to determine which cells to format**
- Click in the box labeled: Format values where this formula is true
- Type **=**
- Click the first cell under the Total Donations label in the Donors table
- Remove the dollar signs in the cell reference to make it a relative cell reference
- Type: **> 75**
- Click the **Format…** button
- Set the format has Bold and set the color as green


Edit Formatting Rule

Select a Rule Type:

- Format all cells based on their values
- Format only cells that contain
- Format only top or bottom ranked values
- Format only values that are above or below average
- Format only unique or duplicate values
- Use a formula to determine which cells to format

Edit the Rule Description:

Format values where this formula is true:

=I4>75

Preview: AaBbCcYyZz

- Click OK and click OK again

Use VLOOKUP with an Approximate Match

VLOOKUP function is a very powerful tool that is used often in business. In this project we want to motivate our donors to donate more each year by offering different levels. In other words, we want them to “level up” just as they might in a video game. On the Levels sheet we have the table of different levels and the donation amounts that are required for each level.

The first thing we need to do is add a new column to our Donors table where we can list the level attained by each of the donors:

- In the Donors table, click cell K3, or the cell just to the right of the 2016 Donations label
- In cell K3 enter: **2016 Level**
- Click the first cell underneath the label 2016 Level

Now we’re going to add the VLOOKUP function using the Insert Function button we used earlier:

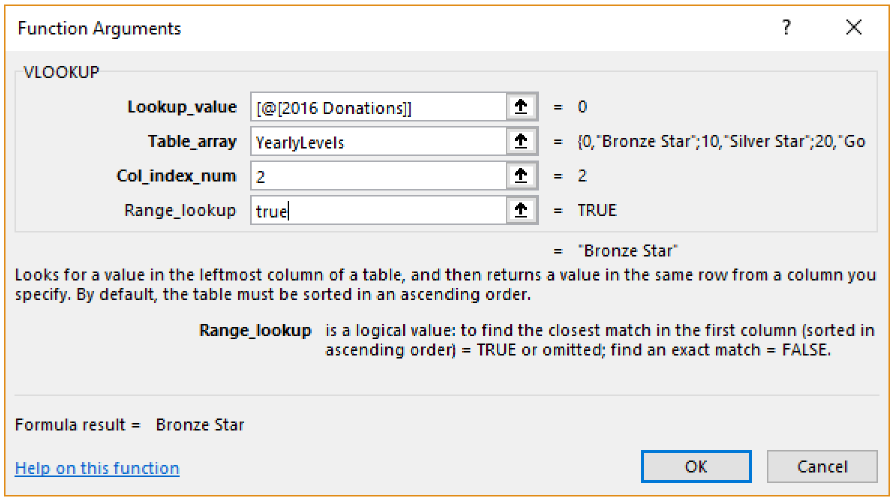
- Click the Insert Function (***fx***) button
- Type lookup and click the Go button
- In the list of functions click the VLOOKUP function and then click the OK button

The Insert Function dialog box lists four boxes for the VLOOKUP function, that is, four arguments. Only three are required, but as we shall see, the fourth one is pretty important too.

- Click in the box labeled **Lookup_value**
- In the Donors table click the first cell under the label 2016 Donations
- Click the box labeled **Table_array**
- Click the tab for the Levels worksheet
- Select all the cells in the Levels table, excluding the labels in the first row
- Note that the actual name of the table (**YearlyLevels**) appears in the box
- In the Function Arguments dialog box, click the box labeled **Col_index_num**
- Type: **2**

Note here that we simply typed the number of the column in the lookup table that has the result we want, in other words, the second column. This may seem odd. In most other places in Excel, we refer to columns by their letter, but here we must use the number. Be careful. This is a mistake that many, many people make.

- In the **Range_lookup** box type: **true**


Function Arguments

VLOOKUP

Lookup_value	[@[2016 Donations]]	= 0
Table_array	YearlyLevels	= {0,"Bronze Star";10,"Silver Star";20,"Go
Col_index_num	2	= 2
Range_lookup	true	= TRUE

= "Bronze Star"

Looks for a value in the leftmost column of a table, and then returns a value in the same row from a column you specify. By default, the table must be sorted in an ascending order.

Range_lookup is a logical value: to find the closest match in the first column (sorted in ascending order) = TRUE or omitted; find an exact match = FALSE.

Formula result = Bronze Star

By setting the **Range_lookup** value to TRUE, we're telling Excel that we will accept an approximate match. This is exactly what we want to do in this case, because it is not very likely that the donations listed in our Donors table will exactly match the first column of our lookup table. [TIP: Whenever the first column of a lookup table has numbers, you will probably use a VLOOKUP function with an approximate match.]

G. Set up a Thank-You Letter

If we want to get contributions in the future, we want to be sure to acknowledge the ones we are receiving now. A great way to do that is to send a thank-you letter.

To make it easy to send the letter, we will set up any drop-down list so that we can simply click the name of the donor to whom we want to send a letter. We would like our drop-down list of donors to expand dynamically as we had more donors to the donors table. That means that we will have to set up a named range that will list all the donors. Here is the procedure:

Create a Dynamic Named Range in a Table

- In the Donors table, select all of the cells under the LastFirst label
- Click in the **Name Box** (just above column A)
- Enter: **DonorsList** (no spaces)

Create a Data Validation List Box

- Click the tab for Thanks Letter
- Click cell C10
- Click the tab for Data
- Click the button **Data Validation**
- Click the list arrow for **Allow** and change it to **List**
- Click the **Source** box
- Type: **=**
- Press function key F3
- Choose: **DonorsList**

Now when you click on cell C10, a list arrow should appear, and when you click on the list arrow, you should see a list of the donors. As an example, let's select the first name in the donors list. And to make things more readable, let's widen the column to double its width.

Now that we've selected a donor, we need to look up the donor's name and address for the letter. Guess which function we're going to use.

Move a column in a table

When we use the VLOOKUP function, the lookup table must have the lookup value in the first column. In other words, the value that we are looking up has to be in the leftmost column, just like in a phone book, where people's last names are listed on the left while the phone numbers are in the right. The value that we are looking up in our Donors table is the LastFirst column. This means that we must move the LastFirst column so that it is the first column of the table. Because we are using a formatted table, the procedure is easier that it usually is:

- Select the LastFirst column including the label at the top
- Place your mouse pointer on the left edge of the selected cells
- Click the left mouse button and drag the column to left of the first column in the table
- A thick vertical line should appear before the first column

- Release the mouse button and the column should move over

Use VLOOKUP with an Exact Match

Now we can use the VLOOKUP function to look up the full name of the donor from our Donors table.

- Click cell C11
- Click the Insert Function button
- The VLOOKUP function should still be visible in the Recently used functions list
- In the Function Arguments dialog box, click the **Lookup_value** box
- Click cell **C10**
- Set the cell reference as an absolute reference by pressing the function key F4
- For the **Table_array**, type **DonorsTable**
- For the **Col_index_num** type 2
- For the optional **Range_lookup**, type false

We set the **Range_lookup** value as FALSE because we do not want Excel to make an approximate match. If Excel cannot find the name of the donor in C10 in the Donors table, we want to be notified about it with an error message. Otherwise Excel might pick another donor and the thank-you letter would go to the wrong person.

In cell C12, we want to see the street address for the donor. Because we used an absolute reference in the formula, we can copy the formula in C11 down to C12 and modify it so that it will show the street address instead of the full name.

- Copy C11 down to C12
- Click C12
- Click the **Insert Function** button
- Change the column index number from **2** to **5**
- Click OK

We now must add the city, state, and zip information. To make this easier, let's add a new column in the Donors table that will combine these three columns into one column.

- Click the tab for the Donors sheet
- Select all the cells in the Total Donations column, including the label at the top
- Right-click one of the selected cells
- Choose: Insert
- Choose: Table columns to the left
- Change the label of the new column to **City State Zip**
- In the first cell under the label, type: **=**
- Click the first cell under the City label
- Type: **& ", " &**
- Click the first cell under the State label
- Type: **& " " &**
- Type: **text(**
- Click the first cell under the Zip label
- Type: **, "00000")**
- Press the Enter key

Note that we had to use the TEXT function to ensure that Excel would not cut off the leading zero in the zip code.

Now let's get back to the VLOOKUP functions.

- Click the tab for the Thanks Letter
- Copy the formula from C12 to C13
- Click the Insert Function button
- Change the Column Index Number from **5** to **9**
- Click OK

All we need now is the first name for the salutation.

- Copy the formula from C13 to C15
- Click the Insert Function button
- Change the Column Index Number from **9** to **3**
- Click OK

Now that we have set up the VLOOKUP data in column C, we can import the data over to our thank-you letter.

- In cell A10 enter the formula: **= C10**
- In cell A11 enter the formula: **= C11**
- Copy this formula to the two cells below it
- Click cell A15
- Enter the formula: **= "Dear " & C15 & ":"**

Now we need to do is to enter today's date.

- Click cell A7
- Enter the formula: **= Today()**

That's it. We can now generate a thank-you letter to any of our donors simply by clicking the list arrow in C10.

But, when we print the letter, we want to be sure that the VLOOKUP data in column C is not printed. We will therefore set the print area:

- Select cells A1 through A27
- Click the tab for **Layout**
- Click Print Area and then click **Set Print Area**

To distinguish the data in column C from the actual letter, it be good idea to set the data in column C in a slightly different format:

- Select cells C10 through C15
- Set the format as blue and italic

Because cell C10 is so important, we should give it special formatting: Set it as bold and give it a light yellow fill color, and an outside border

H. Set up a Macro to Create a Thank-You Letter

Using the list box we set up, it is fairly easy to generate a thank-you letter. But Excel has a very powerful tool that will make the job even easier.

Whenever a new donation comes in, we will be adding a new row to the Donations table. Wouldn't it be nice if we could generate a thank you letter from the Donations sheet, without having to go to the Thanks Letter sheet? Then all we would need to do is click on the cell that has the name of the donor, execute a macro, and our letter would be printed just like that. Here's how to do it:

What's a Macro?

A macro is a macro instruction, a large instruction composed of many small instructions. This is a very powerful tool. If not handled carefully, it could cause a lot of mistakes. But, when you use it correctly, you can save a tremendous amount of time, and avoid hours of tedium.

To use a macro, it is helpful to display the Developer tab.:

Display the developer tab

- Right-click any of the tabs at the top of the Excel window
- Choose Customize the Ribbon
- On the right side of the Excel options dialog box, click the checkbox for Developer
- Click the OK button

The Developer tab should now be visible

We can now start creating our macro. We want to copy the macro will copy the contents of the cell we select and paste it into cell C10 on the Thanks Letter sheet. That will cause that person's name and address to appear in the letter. And then the macro will print the letter. After printing the letter, it will return to the Donations sheet so that you can start adding another row to the Donations table.

Record a Macro

- In the Developer tab, click Start Recording
- Type a name for the macro: **PrintThankYouLetter** (again, no spaces)
- Set a shortcut key as: **Ctrl + t**
  - Ordinarily the Ctrl + t shortcut will create a table, but we can redefine this shortcut key because we can create a table quickly enough using the button on the Home tab
- Choose to store the macro in: This Workbook
- Click the OK button

Note that Excel is now recording all your keystrokes and mouse clicks

- Click the tab for the Donations worksheet
- Click one of the cells in the Member column
- Copy the cell
- Click the tab for the Thanks Letter sheet
- Click C10
- Paste the name that you copied
- Click the File menu
- Choose Print

- Click the Print button
- Click the Donations tab
- In the Developer tab, click Stop Recording

Run the Macro

It is very easy to make a mistake when you are recording a macro. If you do make a mistake, no problem. You can delete the macro and re-record a new one. Below we will describe the process for deleting a macro.

If the recording process went well, we can now see if the macro actually works. In the process of recording the macro, you printed a thank-you letter. Click another name in the Member column and press on the keyboard Ctrl + t. A new letter should not be printing addressed to the newly selected member.

Delete a macro

When you make a mistake in a macro, you could edit the macro to fix it. But, it is much, much easier to simply delete the macro and re-record it. Here is how to delete a macro:

- In the Developer tab, click the button **Macros**
- In the list of macros that appears, click the macro that you want to delete
- Click the Delete button

The macro has now been deleted

Saving a workbook that contains macros

A macro is basically a small computer program. Bad guys have developed a technique for putting computer viruses in a macro. To protect you from these viruses, Microsoft prevents you from saving macros in the standard Excel workbook format. Once you have macros in a workbook, you must therefore save the workbook in a new format: this new format simply warns other users that there is a macro present in the workbook, and that it could be dangerous. Of course, you should never open a workbook that contains macros unless you are sure it came from a reliable source.

- Click the file menu
- Choose Save as
- Change the Save as Type to: **Excel Macro Enabled Workbook (.xlsm)**
- Click the Save button

I. Set up a Button to Execute a Macro

Macros are very powerful. Here’s a way to make them very convenient.

Because we have set up a shortcut key, we can execute on macro with one key stroke. To make it even easier to executor or macro, we can create a nice looking button. Here’s how:

First, we will need to space for our button.

- On the Donations worksheet, insert two blank lines above the Donations table

Add a Button

- In the Developer tab, click the button **Design Mode**
- With Design Mode on, we can now do things such as adding buttons

- Click the button insert
- A menu of controls appears.

There're many possible controls, including buttons, list boxes, checkboxes, and labels.

- In the Form Controls group, click the Button control in the upper left of the group

With your mouse, click in a blank area where you would like the upper left of the button to appear. Then drag down and to the right until a button of your desired size appears.

- A list of macros will appear
- Click the macro: **PrintThankYouLetter**
- Like OK
- Click inside the button
- Change the text caption to read: **Print Thank You Letter to Selected Member**
- Click the button **Design Mode** to turn off Design Mode

The button is now active. It will print a thank you letter to whomever you select in the member column of the Donations table.

J. Set up a Form Letter Asking for Contributions

Microsoft Word has a powerful tool called Mail Merge, which is great for generating personalized form letters. But, we can generate those same letters completely within Excel.

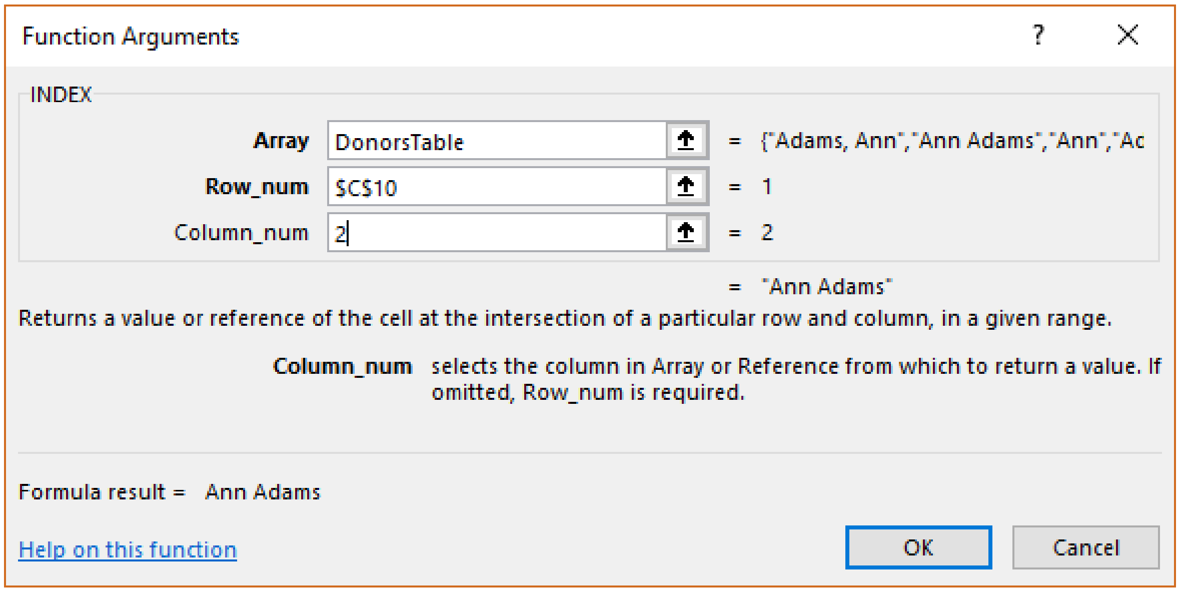
In the previous section, we created a thank you letter that we would send two individual donors. Here we have a letter requesting contributions, which we want to send out to a whole group of donors. Instead of using the VLOOKUP function, we'll use another powerful function called the INDEX function.

Retrieve Data for the Letter Using the Index Function

Rather than selecting a name for the letter, this time we will just specify the row number for the donor. The INDEX function will then retrieve the name and address for that donor. Because we only need to specify a row number rather than a name, it will be much easier to send this letter to a whole group of donors.

- Click the tab for Plea Letter
- Clicks cell C10
- Enter: **1** (the numeral one)
- With cell C11 selected, click the **Insert Function** button
- Type **index**
- Click the Go button
- In the list of functions, click **INDEX** and then click OK
- A list of possible argument lists will appear
- Be sure that the first list is selected, i.e., array, row number, column number
- Click OK
- In the Function Arguments dialog box, click the box for **Array**
- Type **donorstable** (no spaces)
- In the **Row_num** box click C10 in the Plea Letter worksheet
- Change this cell reference to be absolute by pressing the F4 key

- In the **Column_number** box, type **2**
- Click OK



Function Arguments

INDEX

Array: DonorsTable = {"Adams, Ann","Ann Adams","Ann","Ac

Row_num: C10 = 1

Column_num: 2 = 2

= "Ann Adams"

Returns a value or reference of the cell at the intersection of a particular row and column, in a given range.

Column_num selects the column in Array or Reference from which to return a value. If omitted, Row_num is required.

Formula result = Ann Adams

- Copy the formula in C11 and pasted it into cell C12 in C13
- Then paste it into cell C15
- Double click C12 and change the column number to 5
- Press Enter
- Double-click C13 and change column number to **9** and press Enter
- Double-click C15 and change the column number to **3** and press Enter

As you can see, by using the IDEX function we have been able to get the same information we obtained previously with the VLOOKUP function.

Complete the letter

- Click cell A7
- Enter the formula: **= Today()**
- Click cell A11
- Enter the formula: **= C11**
- Copy the formula in C11 to the two cells below it
- Click C15
- Enter the formula: **= "Dear " & C15 & ":"**
- Select cells A1 through A27
- In the Page Layout tab click Print Area
- Choose Set Print Area

Our plea letter is all set. All we need to do is put a row number into cell C10 and the letter will be personalized for a particular member.

K. Set up a Macro to Create Plea Letter Automatically

The power of Mail Merge is that it can produce 100 letters almost as easily as it does one. With a little help from Macros, we can do the same thing completely within Excel.

- First, we have to modify our plea letter slightly:
- Click the tab for the Plea Letter
- Click cell C5
- Type the label: Start
- In cell C6 enter the number: 1
- In cell C7 enter the label: End
- In cell C8 enter the number: 3
- Select cells C5 through C15
- Set the style as italic and the font color as blue
- Click cell C10
- Set the style as bold, with a yellow fill color and an Outside Border

Now we're going to set up a macro that will print the letter to the members in rows 1, 2, and 3 of the Donors table.

Record a Macro

- In the Developer tab, click Record Macro
- Type the macro name as: **PrintPleaLetters**
- Set the shortcut key as **Ctrl + w**
- Store the macro in: This Workbook
- Click OK
- Click cell C6
- Copy the value of C6
- Enter the formula: **= "Dear " & C15 & ":"**
- Click the **file** button
- Choose **print**
- Click the **Print** button to print the letter
- In the Developer tab, click **Stop Recording**

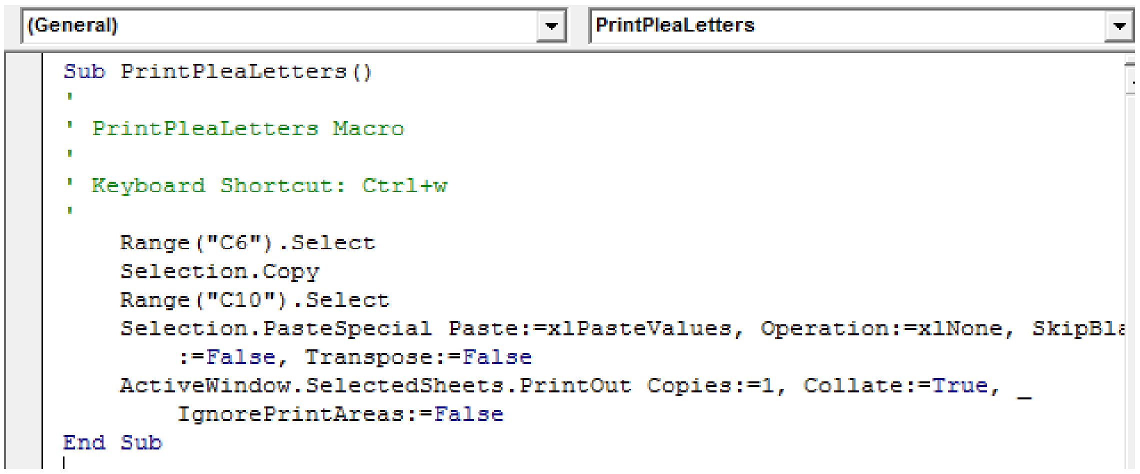
Edit the Macro Using Visual Basic

Our macro will print only one letter. Now we have to edit it so that it will print several letters.

- In the Developer tab click the button **Macros**
- Click the macro **PrintPleaLetters**
- Click the button **Edit**

A new window will pop up showing the Visual Basic editor. The macros are based on the programming language called Visual Basic for Applications. This programming language is used for all the Microsoft Office applications. Visual Basic is fairly easy to understand, but like any programming language, it is sensitive to minor mistakes. Please be sure to follow the instructions below as carefully as possible.

On the left side of the window is a list of Worksheets and Modules. The macros that we have recorded are stored in modules. On the right side of the window is a large pane for editing the code.



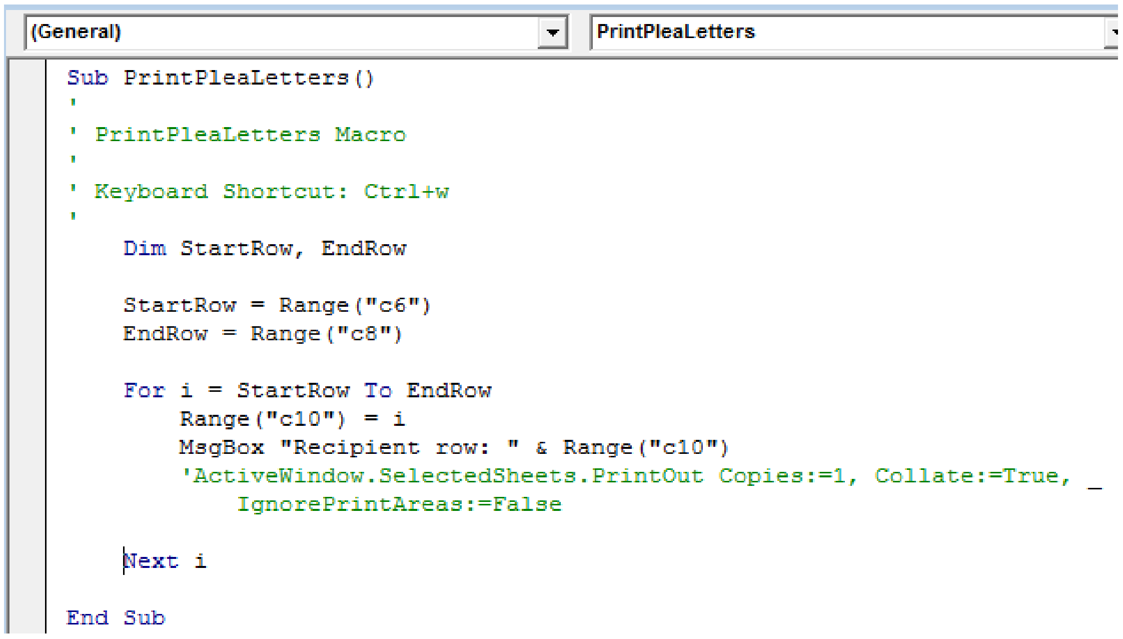
```
Sub PrintPleaLetters()
'
' PrintPleaLetters Macro
'
' Keyboard Shortcut: Ctrl+w
'
    Range("C6").Select
    Selection.Copy
    Range("C10").Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBla
        :=False, Transpose:=False
    ActiveWindow.SelectedSheets.PrintOut Copies:=1, Collate:=True, _
        IgnorePrintAreas:=False
End Sub
```

Let's go through the code line by line so that you can understand what's happening here:

- In the first line we are creating a new subroutine called **PrintPleaLetters**
- After that there were five lines that begin with an apostrophe
    - The apostrophe indicates that the text following it is a comment and is not a computer instruction
    - The Visual Basic editor formats these comments in green to distinguish them from the code
    - We can use comments to explain why we are doing various things in the program
- The next line, beginning "Range…" is actual code, and it says to select the sell C6
- Then the code says to copy the selected cell
- The code then selects cell C10
- And it pastes the data as a value
- The line beginning **Activewindow** prints the sheet
- The last line indicates that the end of the subroutine

We are going to modify the code so that the line that prints the letter will do so a number of times. We will be using a FOR loop to cause this to happen. A FOR loop will cause a segment of code to execute over and over again. Think of the proverbial instructions on a shampoo bottle: Lather, rinse, repeat.

Edit the code so that it looks like the following:



(General) PrintPleaLetters

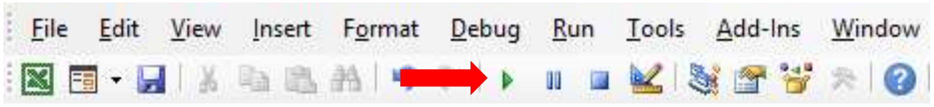
```
Sub PrintPleaLetters()
'
' PrintPleaLetters Macro
'
' Keyboard Shortcut: Ctrl+w
'
    Dim StartRow, EndRow

    StartRow = Range("c6")
    EndRow = Range("c8")

    For i = StartRow To EndRow
        Range("c10") = i
        MsgBox "Recipient row: " & Range("c10")
        'ActiveWindow.SelectedSheets.PrintOut Copies:=1, Collate:=True, _
            IgnorePrintAreas:=False

    Next i

End Sub
```

Let’s go through the new code line by line:

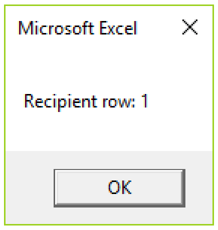
- In the line beginning DIM, we are creating two variables named **StartRow** and **EndRow**
- In the next two lines, we are setting the **StartRow** variable to be equal to whatever is in C6
- Similarly the **EndRow** variable is equal to whatever is in cell C8.
- The FOR command indicates the start of a loop and also introduces the variable **i**
- The variable **i** will be incremented by 1 each time the loop is executed
    - In this example, **i** will take three values: 1, 2, and 3
- The next line places the value of **i** into cell C10
- Then we use the message box command
    - This will create a small dialog box with the text "Recipient is " and then the current value of **i**
    - We do this so we can test the code without printing large numbers of letters, and thus wasting paper.
- In the line that begins **Activewindow**, we have inserted an apostrophe in front of the A
    - This turns to code into a comment and thus keeps the code from actually printing a letter

We can run our code by going back to the Plea Letter and pressing Ctrl + w. But it would be more convenient to run the code directly from the Visual Basic editor.

To run the code, you can simply click the green arrow in the Visual Basic editor toolbar.



When you run the code the message box will appear. In this example it should appear three times. Just click OK each time.

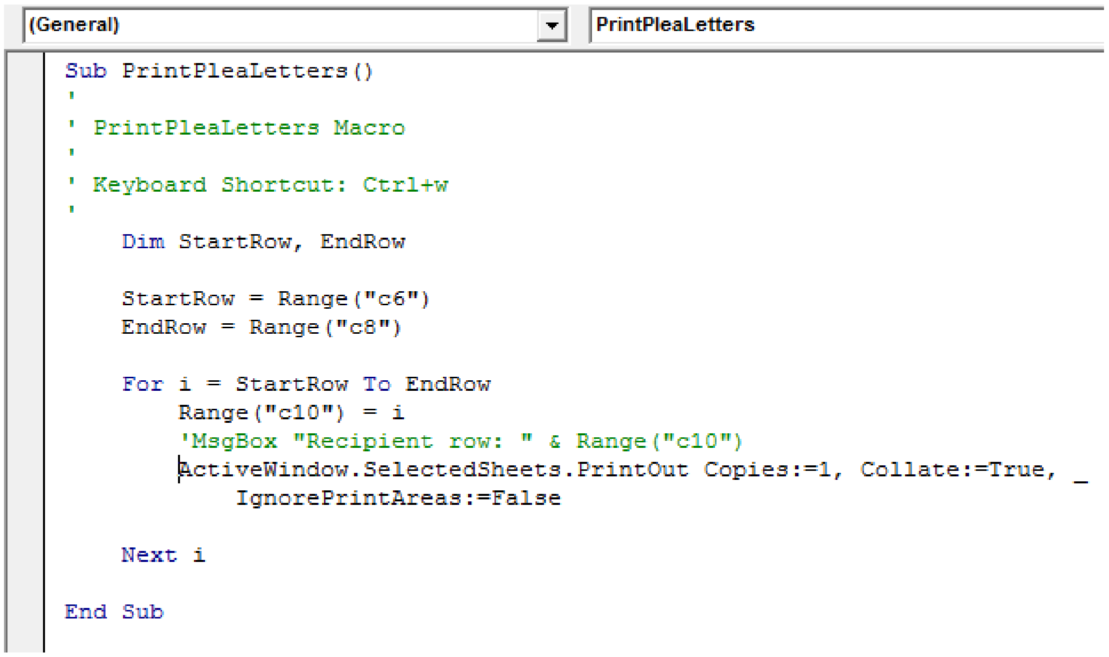

Microsoft Excel

Recipient row: 1

When you are sure that the code is working well, you can edit it so it will print letters.

- Just insert an apostrophe in front of the message box command
- And remove the apostrophe in front of the Activewindow command

The final code should look like this:



```
Sub PrintPleaLetters()
'
' PrintPleaLetters Macro
'
' Keyboard Shortcut: Ctrl+w
'
    Dim StartRow, EndRow

    StartRow = Range("c6")
    EndRow = Range("c8")

    For i = StartRow To EndRow
        Range("c10") = i
        'MsgBox "Recipient row: " & Range("c10")
        ActiveWindow.SelectedSheets.PrintOut Copies:=1, Collate:=True, _
            IgnorePrintAreas:=False

    Next i

End Sub
```

Bounds Checking

If we assume that we will always put valid numbers into cells C6 and C8, then our macro will work very well. But you know what they say about the word “assume.” There will be a time when someone puts the wrong number in cell C8, and you may end up printing dozens of useless letters, thereby wasting a lot of paper.

In programming, anytime you have a loop there is the danger of the loop executing too many times. In the worst case, it could go on for an infinite number of times. Therefore, it is a good idea to put some kind of bounds check in the code to make sure that the loop ends when it should.

Let’s examine what happens if we ask the code to create more letters than we have donors.

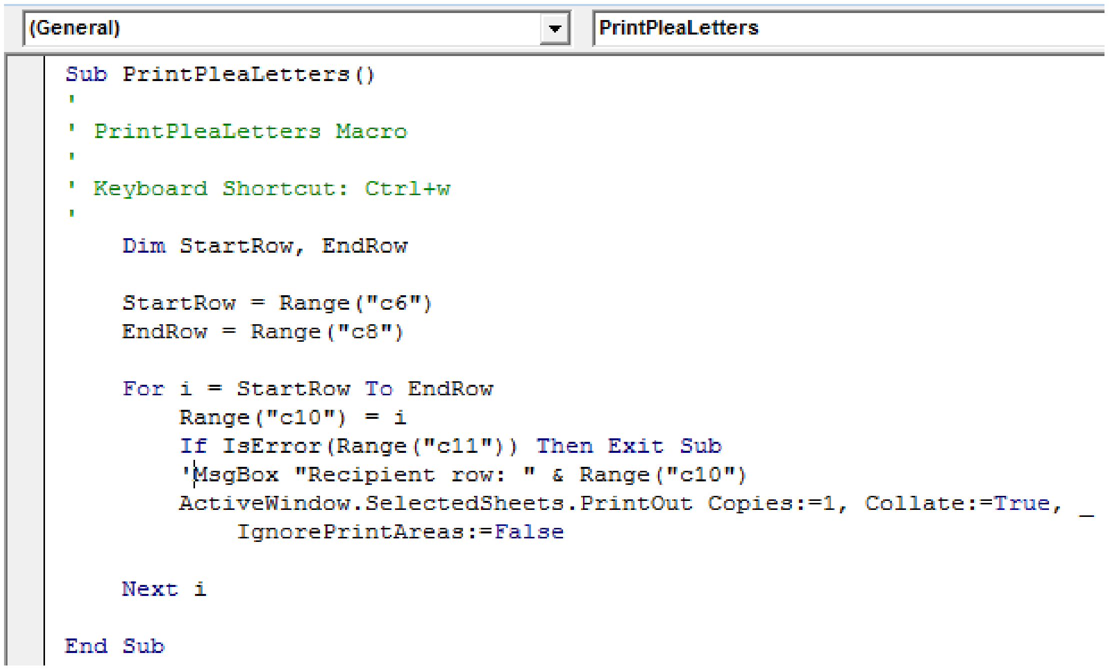
Let’s say our Donors table has 11 rows of data, and we put 15 into cell C8. The macro will print four letters addressed to an error code (“#REF!”)!

We can make sure that these erroneous letters do not print by putting in a simple IF command:

If IsError(Range("c11")) Then Exit Sub

The command says that if there is an error in cell C11, which should ordinarily display the full name of the donor, then the macro or subroutine quits.

The complete code with the error-checking or bounds checking is as follows:



```
Sub PrintPleaLetters()
'
' PrintPleaLetters Macro
'
' Keyboard Shortcut: Ctrl+w
'
    Dim StartRow, EndRow

    StartRow = Range("c6")
    EndRow = Range("c8")

    For i = StartRow To EndRow
        Range("c10") = i
        If IsError(Range("c11")) Then Exit Sub
        'MsgBox "Recipient row: " & Range("c10")
        ActiveWindow.SelectedSheets.PrintOut Copies:=1, Collate:=True, _
            IgnorePrintAreas:=False

    Next i

End Sub
```

To summarize what we did here: We started off with a simple macro, and then created a simple the VBA program. Visual Basic is a powerful programming language. Its main strength is that it integrates so closely with the Microsoft Office applications. There are many, many capabilities of Visual Basic for Applications, And there are many resources on the Internet to help you learn how to use them.

[End]