# Macros

## *Goal:*

The goal of this exercise is to practice creating and using macros.

## *Instructions:*

### Open the macros-practice.txt file

First, start a command line session on your local machine. Next, use vim to open the "macros-practice.txt" file that came in the course downloads. To do that, navigate to the location of the file. Remember this could be different for you depending on where you extracted the contents of the file. This example assumes the course download was saved into your Downloads folder and extracted from there.

```
cd Downloads
cd vimclass
vim macros-practice.txt
```

### Convert Old Python Code to New Python Code

Let's say you have some old Python code (version 2.6 or earlier) that needs to be updated to work with the new Python interpreter (version 3.0 or later). The print statement was replaced with the print() function. That means you'll need to change the following line from this:

Let's change the following line from this:

```
print "Macros are very fun!"
```

To this:

```
print("Macros are very fun!")
```

Let's do this with a macro. By the way, it's not important to know anything about Python. The goal here is to give you a practical example for you to practice your Vim macros skills with.

Here is one way to accomplish this task:

- Place your cursor on the line that reads: `print "Macros are very fun!"`

- Start recording into the **"a** register with **qa**.
- Normalize your cursor position to the beginning of the line by typing **0**.
- Position the cursor at the next space with **f<SPACE>**.
- Replace the space with an opening parenthesis with **r(**.
- Append a closing parenthesis to the line with **A)** and press **<ESCAPE>** to return to normal mode.
- Now position the cursor on the next line with **j** so the macro can be easily repeated.
- Finally type **q** to stop recording the macro.

You can examine the contents of your macro by looking at the **"a** register with **:reg a<ENTER>**. Here is what it should look like:

```
"a   0f r(A)^[j
```

Play your macro on the next line by typing **@a**. Use **@@** to repeat the macro on. Continue repeating the macro with **@@** on the remaining print statements.

NOTE: The solution provided above is not the only way to accomplish this task! One simple example is replacing **f<SPACE>** with **t"** in the macro. Both perform the same act of positioning the cursor under the space. As long as you get the desired results you're after, that's all that matters.

**Create a Shell Script From a List**

Let's say you have a list of users and you want to perform the same set of actions on each of those users. One way to accomplish this is to record a macro that performs the required command for one user and apply that macro to the rest of the list.

The goal is to turn this text:

```
jason
sophia
jack
emma
ava
```

Into this:

```
passwd -l jason && echo jason >> locked_users.txt
passwd -l sophia && echo sophia >> locked_users.txt
passwd -l jack && echo jack >> locked_users.txt
passwd -l emma && echo emma >> locked_users.txt
passwd -l ava && echo ava >> locked_users.txt
```

You could use those commands on a Linux system to lock each of the user accounts and add the account names to the locked_users.txt file. Of course, it's not important what these commands do as we're interested in Vim, but I wanted to provide you with a practical example.

Here is one way to accomplish this task:

- Place your cursor on the line that contains "jason".
- Start recording into the **"b** register with **qb**.
- Yank the user name into the default register with **yaw**.
- Start insert mode at the beginning of the line with **I**. Now type the text "passwd -l " and press **<ESCAPE>**.
- Next, start insert mode at the end of the line with **A** and type " && echo ". Press **<ESCAPE>** to return to normal mode.
- Next, paste the contents of the unnamed register after the cursor position with **p**.
- Continue appending by typing **a**, and then typing " >> locked_users.txt" followed by **<ESCAPE>**.
- Now position the cursor on the next line with **j** so the macro can be easily repeated.
- Finally type **q** to stop recording the macro.

You can examine the contents of your macro by looking at the **"b** register with **:reg b<ENTER>**. Here is what it should look like:

```
"b   yawIpasswd -l ^[A && echo ^[pa >> locked_users.txt^[j
```

Play your macro on the next line by typing **@b**. You can probably see that there are three more items in this list, so use **3@b** to repeat the macro four times.

**Normalize Phone Numbers**

Next, create a macro that changes a United States style phone number from:

```
2798265253
```

To:

```
(279) 826-5253
```

Here is one way to accomplish this task:

- Place your cursor on the line that contains "2798265253".
- Start recording into the **"p** register with **qp**.

- Start insert mode at the beginning of the line with **I**.
- Type an opening parenthesis **(** and press **<ESCAPE>** to return to normal mode.
- Position the cursor under the 9 by typing **l** 3 times.
- Type **a** to append text after the cursor and type **)<SPACE>**. Press **<ESCAPE>** to return to normal mode.
- Position the cursor under the 6 by typing **l** 3 times.
- Type **a-** to append a hyphen and press **<ESCAPE>** to return to normal mode.
- Now position the cursor on the next line with **j** so the macro can be easily repeated.
- Finally type **q** to stop recording the macro.

You can examine the contents of your macro by looking at the **"p** register with **:reg p<ENTER>**. Here is what it should look like:

```
"p   I(^[llla) ^[llla-^[j
```

Because there is a long list of phone numbers that scroll off the bottom of your screen you can't easily know how many times you need to apply the macro. Also, repeating @@ could take quite awhile. So, get the range you need to apply the macro to. Turn on line numbering with **:set nu<ENTER>**. Note that your cursor is currently on line 25. Now page down to the end of the list of phone numbers with **Ctrl-f**. You'll find the last phone number is on line 73.

Execute the macro over this range using the **normal** command. Type **:25,73 normal @p<ENTER>.** Now check that the macro was executed over that entire range by paging up with **Ctrl-b**.

**Extract Important Data from a Log File**

The next set of lines in the file contain system logs from a Linux server. Specifically these lines are logs of blocked connection attempts by the local Linux firewall. Your goal is to extract the timestamp, the source IP address of the connection attempt, and the destination port.

The source IP address follows "SRC=". Example: SRC=190.18.193.152
The destination port follows "DPT=". Example: DPT=23

This line:

```
Jan 13 09:57:01 www1 kernel: [3947771.808744] [BLOCK] IN=eth0 OUT=
MAC=e6:e9:2d:04:b6:95:3c:8a:b0:0d:6f:f0:08:00 SRC=190.18.193.152
DST=2.5.9.1 LEN=40 TOS=0x02 PREC=0x00 TTL=51 ID=25120 PROTO=TCP SPT=12502
DPT=23 WINDOW=4078 RES=0x00 SYN URGP=0
```

Will be narrowed down to just this comma separated value line:

```
Jan 13 09:57:01,190.188.193.152,23
```

Here is one way to accomplish this task:

- Place your cursor on the line that starts with "Jan 13 09:57:01".
- Start recording into the **"l** register with **ql**.
- Normalize your cursor position to the beginning of the line by typing **0**.
- Position your cursor on the space just after the time stamp by typing **tw**.
- Now delete all the text up to "SRC=". You can do this with **dtS**.
- Now delete SRC with **dw**.
- Next, replace "=" with a comma: **r,**.
- Position the cursor after the IP address with **f<SPACE>**.
- Delete the text up to "DPT=" with **d/DPT<ENTER>**.
- Now delete DPT with **dw**.
- Next, replace "=" with a comma: **r,**.
- Position the cursor after the port number with **f<SPACE>**.
- Delete the remaining text on the line with **D**.
- Now position the cursor on the next line with **j** so the macro can be easily repeated.
- Finally type **q** to stop recording the macro.

You can examine the contents of your macro by looking at the **"l** register with **:reg l<ENTER>**. Here is what it should look like:

```
"l   0twdtSdwr,f d/DPT^Mdwr,f Dj
```

As always, there are other ways to accomplish the same task. Just one simple example is using **2cw,<ESCAPE>** to change "SRC=" to "," instead of using **dwr,**. Take a moment and think of other ways to alter this macro and get the same result.

Test your macro on the next line by typing **@l**. If it looks correct, continue with **@@** until all the lines are in the desired format.

**Condense Data From Multiple Lines Into a Single Line**

Remember that macros are just a series of recorded keystrokes. Even though you've been working on macros that manipulate single lines, you can as easily manipulate multiple lines. Let's say you want to turn these three lines into one line.

Before:

```
Country China
1,380,950,000 people
```

After:

```
1,380,950,000;China
```

Here is one way to accomplish this task:

- Place your cursor on the line that reads "Country China".
- Start recording into the **"c** register with **qc**.
- Normalize your cursor position to the beginning of the line by typing **0**.
- Delete the word "Country" with **dw**.
- Move to the line below with **j**.
- Cut the number into the unnamed register with **dW**.
- Move up to the original line with **k**.
- Paste the number before your cursor position with **P**.
- Replace the space with a semicolon by typing **r;**.
- Move to the line below with **j**.
- Delete the current line and the next line with **2dd**.
- Finally type **q** to stop recording the macro.

You can examine the contents of your macro by looking at the **"c** register with **:reg c<ENTER>**. Here is what it should look like:

```
"c   0dwjdWkPr;j2dd
```

Now use a count to repeat the macro 4 more times to format the rest of the data in this set. Type **4@c**.

As always, there are multiple ways to achieve the same result in Vim.

**Extract Data from HTML**

Here are a list of links (<a> tags) on a single line of HTML.

```
<a href="#">@armyspy.com</a><a href="#">@cuvox.de</a><a
href="#">@dayrep.com</a><a href="#">@einrot.com</a><a
href="#">@fleckens.hu</a><a href="#">@gustr.com</a><a
href="#">@jourrapide.com</a><a href="#">@rhyta.com</a><a
href="#">@superrito.com</a><a href="#">@teleworm.us</a>
```

The goal is convert those links to the following:

```
armyspy.com
cuvox.de
dayrep.com
einrot.com
fleckens.hu
gustr.com
jourrapide.com
rhyta.com
superrito.com
teleworm.us
```

Here is one way to accomplish this task:

- Place your cursor at the beginning of the line that starts with "<a".
- Start recording into the **"q** register with **qq**.
- Delete the text up to and including "@" with **df@**.
- Position the cursor under the "<" with **f<**.
- Replace "</a>" with **<ENTER>** by typing **cf><ENTER><ESCAPE>**.
- Finally type **q** to stop recording the macro.

You can examine the contents of your macro by looking at the **"q** register with **:reg q<ENTER>**. Here is what it should look like:

```
"q   df@f<cf>^M^[
```

Execute the macro with **@q**. Keep repeating the macro with **@@** until you are left with the desired text.

**Exit out of vim**

If you want to abandon your changes so you can try this practice exercise again, use **:q!<ENTER>**.