NAME

penny-basics - getting started with Penny

PENNY IS DOUBLE-ENTRY ACCOUNTING

Penny is a double-entry accounting system. Other accounting systems use double-entry principles--indeed, they are double-entry accounting systems--but they do not use traditional double-entry accounting terms such as *debit* and *credit*. Or, they might use these terms, but not as they are used in a traditional accounting sense. Unlike other systems, Penny does not try to hide the details of double-entry accounting from you. Therefore, you will need to have basic knowledge of double-entry accounting to use Penny.

Ask yourself this question: what is a debit? If you said "it's what happens when my bank charges me money," read "Learning about double-entry accounting" below. If you said "left", then you're probably ready to use Penny.

LEARNING ABOUT DOUBLE-ENTRY ACCOUNTING

Since there are many great places to learn the basics of double-entry accounting, I won't try to write about it here. Instead, here are some places to look.

- The Wikipedia article on double-entry accounting (http://en.wikipedia.org/wiki/Double-entry_bookkeeping_system) is good.

- Principles of Accounting (http://www.principlesofaccounting.com/) is a great free online text.

- Or, go to your favorite bookseller and find a used textbook on accounting. These can be had for less than ten U.S. dollars, including shipping.

THE penny PROGRAM

The **penny** program is Penny's most basic tool. It gives you reports on what you have recorded in your *ledger*, which is simply a text file (or multiple files) containing your financial transactions. **penny** will never modify the data in your ledger; you maintain it yourself by hand (or by using some of the other programs in the **penny** suite, such as **penny-fit**, which will modify this data for you under some circumsances.)

Examine the *starter.pny* file, located in the *examples* directory of the *penny-bin* package. It shows the basics of how to write a Penny ledger file. This man page will use this file for examples. Please follow along by typing some of the commands yourself and experiment on your own.

HOW penny WORKS

First, some terminology. Your ledger file contains *transactions*. Each transaction consists of at least two **postings**. All the postings in a transaction must *balance*; that is, the sum of the debits for each commodity must equal the sum of the credits for that commodity. **penny** first reads in all your transactions. All the transactions must be balanced; if they are not, **penny** quits with an error message. Then **penny** splits the transactions into postings. After that, **penny** deals only with the postings, and not with the transactions that they were a part of.

penny then will discard some of the postings and keep others, depending on the *filter specification* that you give. You can also sort the postings; by default they are left in the order that they were in in the file. Then a *report* that you specify is shown. The reports have additional options that you may specify.

This is all more clear with some examples, so let's go!

SEEING THE POSTINGS IN AN ACCOUNT

Let's say you want to see the postings in your checking account.

```
penny --account Assets:Checking postings starter.pny
```

does what you would expect. If you want to limit the postings to those before or after a certain date, try one of these:

```
penny -a Assets:Checking --and -d '>=' 2012-12-06 postings starter.pny

penny -a Assets:Checking --and -d '>=' 2012=12-06 --and \
  -d '<' 2012-12-11 postings starter.pny
```

As these examples show, you can use *-a* instead of *--account*, and *-d* instead of *--date*. Also, options such as *-a* and *-d* are called *operands*. If you use more than one operand, you must join them together using *operators*. The operators available to you are *--not* ,*--and* , and *--or*, in that order of precedence. Here is a more complicated example that uses the various operators and operands:

```
penny -a Assets:Checking --and --open --payee 'Whole Foods' \
  --or --payee Comcast --close --and --date '>=' 2012-12-01 --and \
  --date '<' 2012-01-01 pos starter.pny
```

This example also shows that you can abbreviate the name of the report; that is, instead of *postings*, you can say *pos*. You can use the shortest unambiguous abbreviation, so you could even simply say *p* as there is no other report that starts with a *p*.

On the far right side of each *postings* report is a running balance, like in a checkbook register. You may have noticed that in the examples above the running total reflects only the postings you saw in the report. Sometimes that is what you want. Other times you might want to see the total running balance, even as it is affected by the postings not shown in the report. To do this, include the filtering options **after** the word *postings* rather than before. Filtering options that are included before the word *postings* affect which postings are part of the report. Only postings that are part of the report affect the running balance. Filtering options **after** the word *postings* affect which postings are **shown** in the report. Postings that are part of the report but are not shown still affect the running balance.

Here is an example:

```
penny --sort date -a Assets:Checking pos -d '>' 2012-12-10 \
  starter.pny
```

This example introduces a new option, *--sort*. You specify the field name on which you want to sort your postings. To sort them in ascending order, use all lower case letters when specifying the name of the field. To sort in descending order, capitalize the first letter. By default, **penny** does not sort your postings; it leaves them in the same order they were in in your ledger file. If you want to see the running balance of a report, it's important to make sure the postings are sorted in chronologial order, either because you told **penny** to sort them or because you always keep your postings in chronological order in your ledger file (I don't.)

Here we also see that you can use the *-d* option in place of *--date*.

SEEING THE BALANCE OF AN ACCOUNT

To see the balance of your checking account, you could run

```
penny -a Assets:Checking pos --tail 1 starter.pny
```

and look at the running balance shown. This command included all the postings in your checking account in the postings report, but it only showed the last posting. (You could have used the UNIX **tail** program, but that will not always work well because sometimes a single posting will take up more than one line on your screen.)

Another way to do it is with

```
penny -a Assets:Checking balance starter.pny
```

which shows the accounts hierarchically. The *balance* report is your friend if you want to see the balance of many accounts at once.

INCOME AND EXPENSES

How does your income compare to your expenses for a certain month? Try:

```
penny --open --account-level 0 Income --or --account-level 0 Expenses \
  --close --and -d '>=' 2012-12-01 --and -d '<' 2013-01-01 \
  balance starter.pny
```

The *--account-level* operand takes two arguments. The first one is the number of the sub-account you wish to match. For instance, the account *Expenses:Food* has two sub-accounts: the first, *Expenses*, is numbered 0, and the second, *Food*, is numbered 1. The second argument to *--account-level* is the pattern you wish to use. All matching accounts will be part of your *balance* report. The result of this command shows your total expenses and total income, and the difference between the two is shown on the top *Total* line. (You don't have to use the *--account-level* operand, but it is more precise in this instance. Even more precise would have been to specify *--exact* at the beginning of the command line. Without *--exact*, a pattern matches if the pattern you specify is found anywhere within the target text.)

BALANCING YOUR CHECKBOOK

Or, that fun task also known as "reconciling your account." First you want to make sure that the reconciled balance of your checking account is the same as what the bank says it was:

```
penny -a Assets:Checking --and --flag R bal starter.pny
```

Compare this total against the opening balance shown on your bank statment. If they match, good. If not, figure out why and fix it before proceeding. Then, find each posting shown on your bank statement in your ledger. Add a *C* flag to each posting. After you have found all the bank's postings in your ledger, run this:

```
penny -a Assets:Checking --and --open --flag R --or --flag C --close \
  bal starter.pny
```

The balance shown should match what is on your bank statement. If not, make sure you have matched up all the bank's postings with a posting in your file, and make sure the bank's amount matches your amount. You can easily see which postings you have just marked as cleared with:

```
penny -a Assets:Checking --and --flag C pos stater.pny
```

Once the balances match up, use your text editor or **penny-reconcile**(1) to change the *C* flags to *R* flags.

WHAT'S YOUR NET WORTH?

Try

```
penny -a Assets --or -a Liabilities balance starter.pny
```

COLORS

Output from **penny** is easier to read when it's colorful. **penny** can use up to 256 colors on your terminal. Just make sure that your *TERM* environment variable is set to a terminal that supports 256 colors. I use **xterm**(1), which supports 256 colors, but by default **xterm** sets the *TERM* environment variable to *xterm*, which only supports 8 colors. To make **xterm** set the *TERM* environment variable to one that supports 256 colors, I have the following text in my *~/.Xresources* file:

```
XTerm*termName: xterm-256color
```

After running

```
xrdb -merge ~/.Xresources
```

and launching a new **xterm**, the new setting should take effect. It's likely your operating system is already set up to automatically merge your *~/.Xresources* file when you launch an X session.

If you don't like colors, use *--scheme plain*. By default, **penny** does not use colors if its standard output is not a terminal, though you can override this with *--color-to-file=yes*. This can be useful if you are sending output to a pager such as **less**(1) and you want to see colors (with **less**, you will want to use the *-R* option.)

WHERE TO GO FROM HERE

This is enough to get you started with **penny**. If you want to know more, see **penny**(1), which is an exhaustive reference. Also, see **penny-suite**(7), which lists all Penny programs and documentation.