Arguments - Flags

Some arguments don't need values. They are just indications to the script to do things one way or another. The reverse argument in the for statement is an example: it doesn't take a value, it just indicates whether or not to iterate backwards. These types of arguments are flag arguments. Flag arguments are very similar to keyword arguments and all of what you have learned about keyword arguments applies.

We will rewrite the discount script of cash-register to accept a flag argument, double. If the double argument is present, then the script will compute twice the normal discount.

```
$ on amount, double.

$ percent.

double ?= false.

percent := 0.

if (amount >= 10.00) then [ percent := 0.025 ].

if (amount >= 25.00) then [ percent := 0.05 ].

if (amount >= 100.00) then [ percent := 0.075 ].

if double then [ percent := percent * 2 ].

return amount * percent.
```

The declaration line now additionally declares an argument double. Notice that we've used the short cut where the name of the variable is the same as the keyword. This is common for flag arguments, although not required.

The third line defaults the value of the flag to false. When a message includes a flag argument, such as in the expression:

```
cash-register discount on 23.99 double
```

the system sets the flag argument in the script to true. If the message omits the flag argument, as in:

```
cash-register discount on 23.99
```

then like all omitted arguments, the system doesn't give the argument in the script any value at all. In this case, the default assignment expression in the script:

```
double ?= false.
```

will give the argument the value of false. After the default assignment expression has executed, you can be sure that the argument is either true or false.

Conditionally using Flags

Sometimes you don't know when you use a message that takes a flag if you want the flag or not: it may depend on values of variables when the script is run. For example, suppose after we sold five hundred dollars in a day, we want to start giving out double discounts. We could rewrite the script for grand-total to read:

```
$ amt-off, tax, be-generous.

be-generous := daily-total >= 500.

amt-off := self discount on sub-total double be-generous.

tax := sub-total * 0.0725.

total := sub-total - amt-off + tax.

daily-total := daily-total + total.
```

```
sub-total := 0.
```

The local variable be-generous will be true if we've sold over five hundred dollars today, otherwise it will be false. Then, when the discount message uses the double flag argument, it specifies a value, the value of be-generous. In the script for discount the flag argument double will be set to either true or false and will apply the double discount if directed.

!!
If you give a flag argument a value, be sure to give it only true or false. The script that understands the flag variable will only expect these two values and will probably generate an error otherwise.