HensPace

it Keeps on growing

iKog - the simple todo list
version 1.90

iKog - the simple todo list

© 2008 Steve Butler

document version 1.0.13

Printed: December 2008

Table of Contents

I

Introduction

1 Introduction

The problem with all todo lists-

it Keeps on growing

Welcome to the iKog program. This is a very simple utility designed to make the management of your to-do lists quick and easy. The emphasis is on speed and portability. The program is a text-only utility with no graphical user-interface but it does include some features specifically designed to help with techniques such as Getting Things Done (GTD). If you're looking for fancy menus and windows, then iKog is not for you. If you're comfortable with text-based applications, then perhaps you'll find it useful. Anyway its free, so you've got nothing to lose.

If you'd like to find some more free stuff, why not visit www.henspace.co.uk

Features

- Quick and easy to use – faster than a fancy gui, or so I think.
- Portable
- Powerful filters
- Encryption of your private information

This program is quite new so please be patient. If you find any problems just let me know. Visit http://www.henspace.co.uk/ikog/index.html

---

Current version 1.90

See the release history for what's been changing.

1.1 Acknowledgements

This documentation makes use of the silk icon set created by Mark James. Visit his site at http://www.famfamfam.com for more information. These icons are provided under Creative Commons Attribution 2.5 License.

1.2 Conventions in this guide

Text styles

Any text that you would enter in the program is shown like `this.`

A complete line, simulating the program screen is shown like this:

```
>>>like this
```

Special keys on the keyboard are shown between **<>** characters. For example **<F2>** or **<enter>**.

Symbols

ⓘ	This symbol shows a warning note. Typically something that you cannot do.
💬	This symbol shows a general comment.
💡	This symbol shows a tip. Typically something that will make using the program easier to use.
⚙	Refers to a new feature that has been added. This is normally followed by the version when the feature was added.
🐞	Refers to the removal of a bug.
✎	Refers to any miscellaneous change.

1.3 Licence, terms and conditions

IKog, todo list manager.
Copyright (C) 2006-2007 S. J. Butler

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

The details of the GNU General Public License are given below.

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software. Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program).Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium,provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program,and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to
this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made
generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software
Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

1.4 Why another todo list?

There are hundreds of todo lists on the web, some commercial and many free. So why do we need yet another one? The truthful answer is we probably don't, but I was looking for something a little different. What I wanted was a simple way to carry my tasks around on a USB stick. With some computers running Linux and others Windows, the program needed to be capable of running on different platforms. I didn't need fancy graphics or editing, so ikog.py was born.

II

Installing iKog

2 Installing iKog

Hopefully this should be a very short section. If you have Python (version 2.4 or above), already installed on your computer then there is nothing to install. You merely need to download the ikog.py file and launch it.

If you don't have Python, then you will need to install it on your computer. Visit http://www.python.org for details on how to get the latest version.

The next thing you need to do is to download the program file ikog.py. This is not only the program but it also contains all of your tasks - once you enter them that is. See Downloading iKog.

2.1 Downloading iKog

Check the licence details before downloading. If you need to upgrade a previous version, check the Upgrading section first.

Note you will need Python **2.4** or above. The program will not work with earlier versions. Visit http://www.python.org if you need to get the latest version.

To get hold of the program, visit http://www.henspace.co.uk/ikog/index.html

Once you have downloaded the file you will need to decompress it. On Windows you should be able to right click on the file and select Extract files.

If you are running on Windows and don't have an extraction utility you can try 7Zip at http://www.7-zip.org/. This is a free, open-source file-archiver.

On Linux systems just enter `gzip -d ikog.py`

Many modern browsers will automatically decompress a *gzip* file when it is downloaded. Unfortunately they do not rename the file. When you try to decompress the downloaded file, you might get a message along the lines of, *not in gzip format* or *ikog.py.gz is not supported* depending on the program you use to decompress the file. If you get this error and the size of your file is greater than 70 kB, then it has already been decompressed. If this is the case, just remove the *.gz* extension.

If necessary, rename your resulting file from *ikog_1.90.py* to *ikog.py*.

Once you have decompressed the file you should have a single file ikog.py.

This is a single file and is the only one you need. Typically I would put this on my USB stick but you can save anywhere you want to. For Linux users, remember to change the properties of the file to read, write and executable once you have downloaded it, e.g

```
chmod 700 ikog.py
```

Remember, once you run the program, this file will be modified and include all your tasks as well.

There's nothing to stop you renaming the file and keeping different copies for different types of list.

2.1.1 Archived versions

No reason why you should want these, but in case you do, some older versions are maintained as zip files. Always check out the release history as there may be good reasons (aka bugs) not to use them.

To get archived versions visit http://www.henspace.co.uk/ikog/index.html

2.2 Upgrading

Upgrading your copy of iKog is quite straightforward.

If you are using tasks in a separate file, see Separating your data, then you can just replace ikog.py with the new file. If you are using tasks stored in the program file, follow the instructions below. If in doubt always make a copy of ikog.py first.

Upgrading if using tasks stored in the main program

1. Use the `EXPORT` command to save your current tasks. This will create a file, *ikog.py.tasks.txt.* See Exporting and importing.
2. Download the latest version of iKog and extract the file, ikog.py. Use this to replace your old version.
3. Start the new version.
4. From the prompt, enter the `IMPORT` command followed by the file containing your exported tasks. i. e.

```
>>>IMPORT ikog.py.tasks.txt
```

5. That's all there is to it.

III

Using iKog

3 Using iKog

3.1 Getting started

Assuming you've downloaded the program and stored it on your USB stick or some other location, then you're ready to go. On Windows, you should be able to double-click on the ikog.py file. On Linux, open a terminal and type `/path to the file/ikog.py` and with any luck you should be off and running.

When the program starts you should get a screen similar to the following:

```
ikog.py v 1.25 2006-08-13
Copyright (C) 2006 S. J. Butler
Visit http://www.henspace.co.uk for more information.
This program is free software; you can redistribute it and/or modify
it under the terms of the GNU General Public Licence as published by
the Free Software Foundation; either version 2 of the License, or
(at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.  The license is available
from http://www.gnu.org/licenses/gpl.txt
Auto save is off
Review mode is off.  Enter re-displays the current task

Enter HELP for instructions.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
There are no tasks todo.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
```

Of course there are no tasks entered yet. If you enter:

```
>>>help
```

you will get a list of instructions displayed. Keep pressing **<enter>** until all of the help instructions have been displayed, or if you get bored, just enter `s` to skip.

Right, you now have two options. Choose whichever you want.

1. I can't be bothered reading all this stuff. Just let me jump straight in and work it out myself.
2. Take me to the first section about how to add tasks.

3.1.1 Jumping straight in

Jumping in at the deep end? Okay, you don't want to spend any time reading this manual and you want to go straight in and start using iKog. This section shows the typical actions you might need to do. There's not much hand-holding here; if you get stuck then you might have to actually read the rest of the documentation.

Add a task

```
>>>+ Learn how to use this program.
```

Add another task

```
>>>+ Read that book on web design.
```

Add another task but this time with a priority of 8. i.e quite important.

```
>>>+ Buy tin of red paint for the bathroom #8
```

Add another task, priority 7 and put it in a project called Bathroom

```
>>>+ Buy ceramic tiles #7 :pBathroom
```

Add a task that I want to put in a context list of jobs I can do at the computer.

```
>>>+ Search the web for a pasta recipe @Computer
```

Add another task that I can do at the computer but use the abbreviated form.

```
>>>+ By a book from Amazon @c
```

Add a task that must be done on 30th August 2006

```
>>>+ Take the cat to the vets :d2006-08-30
```

List all of my tasks.

```
>>>LIST
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] Buy tin of read paint for the bathroom #8 @Anywhere [2006-08-17]
[01] Buy ceramic tiles #7 @Anywhere Projects: Bathroom [2006-08-17]
[02] Learn how to use this program #5 @Anywhere [2006-08-17]
[03] Read that book on web design #5 @Anywhere [2006-08-17]
[04] Search the web for a pasta recipe #5 @Computer [2006-08-17]
[05] Buy a book from Amazon #5 @Computer [2006-08-17]
[06] @Date 2006-08-30 Take the cat to the vets #5 @Anywhere [2006-08-17]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

List only the tasks that I can do at the computer.

```
>>>LIST @Computer
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Filter = @Computer
[04] Search the web for a pasta recipe #5 @Computer [2006-08-17]
[05] Buy a book from Amazon #5 @Computer [2006-08-17]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Produce a list, grouped by context that I can view in my browser and print

```
>>>@>
```

Okay, I've bought that book from Amazon, lets get rid of it.

```
>>>KILL 5
```

I want to put task 4, searching the web, into an @Internet context instead of the @Computer context.

```
>>>MOD 4 @Internet
```

Go to the next task in my list.

```
>>>NEXT
```

Okay that's it. There's a lot more to learn but for that you'll have to read the manual.

3.2 Getting help

If you need help you can enter `HELP` or `H`. This will display a quick guide on how to use ikog.py.

If you are already up to speed with ikog.py and just need a quick reminder, enter `?`.

If you want to access this help, just enter `WEB`. (This is actually just a cheap plug for my other free software but there you go.)

3.3 Adding tasks

Okay you've started the program and you're ready to start adding some tasks. To add a task, simply type a `+` followed by the task you want to add. The example below shows how to add this task.

```
>>>+ Learn how to use this program
```

You must separate the command from the rest of the text with a space.
ie. Enter `+ Learn` and not `+Learn`.

You will now see that the bottom of the screen shows the task that we have just added.

```
>>>+ Learn how to use this program
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] Learn how to use this program
 Priority: 05
 Context: @Anywhere
 Created: [2006-08-10]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Note the number at the start of the task in square brackets; [00]. This is the task number 0. Many commands need the number of the task to be entered and this is where you find it.

The listing above shows a colour screen. Not all terminals support colour and the default is with colour switched off so your screen may look a little different. If you want to use colour check the Using colour section.

If you read the help, you may have noticed that many commands have different ways of achieving the same thing. The help file shows the add command as `ADD/A/+`. This means that any of the forms, `ADD`, `A` or `+` can be used. Also the commands are not case sensitive.

So all of the following entries achieve the same thing.

```
>>>+ Learn how to use this program
```

```
>>>add Learn how to use this program
```

```
>>>ADD Learn how to use this program
```

```
>>>A Learn how to use this program
```

There is another trick to add a task. Any line you enter that is not recognised as a command and is longer than 10 characters will also be added as a task. So the following will also work

```
>>>Learn how to use this program
```

You do need to be careful using this technique because if the first word is recognised as a command you might not get the result you expected. Imagine you typed the following:

```
A stitch in time saves nine
```

The actual task that is created would be `stitch in time saves nine` because the `A` would have been interpreted as a command. Until you get to grips with the program, you are probably better using the `+` format or always starting your task with a context. (Contexts are explained in Adding a context)

Tasks are normally added to the end of the list. If you want to add a task to the top, just use the `IMMEDIATE`, `I` or `++` command. This is used in exactly the same way as the normal add command but with it placed at the top.

e.g.

```
>>>++ Put this task at the top
```

It is important to remember that although you can try to add the task to the top, the tasks' priorities still take precedence.

from version 1.60
The immediate command also sets the @Date context to today. This forces the task to marked as something to do now.

3.3.1 Assigning a priority

Obviously some tasks are more important than others. We can easily assign a priority to a task simply by embedding the priority in the task. The priority is simply a number from 1 to 10 preceded by the # character. The higher the number, the more important the task. If you do not assign a priority, a value of 5 is automatically assigned. Only the first occurrence of the # symbol is used as a priority. Any subsequent occurrences are treated as normal text. So to add an unimportant task we could enter:

```
>>>+ Buy some cat food #01
```

To add and important task we might enter:

```
>>>+ Buy some #10 beer
```

Note the position of the priority doesn't matter.

3.3.2 Adding a context

Contexts are like sub-lists or sub-categories used in techniques such as GTD. Various management techniques recommend sub-dividing your tasks into context lists. You might like to check out the http://www.43Folders.com website which has lots of useful information and links about these techniques. For example, you might want to put some tasks on a list that you would work on at the computer, or at work or perhaps at home. To assign a context to a task, just embed the context name in the task, preceded by the @ character.

So continuing with our example, we might want to add a task to check out the 43folders website. We give this task the context of @Computer.

```
>>>+ Check out the 43folders website @computer
```

You can assign more than one context to a task. So you might have.

```
>>>+ Check out the 43folders website @computer @home
```

```
>>>+ Complete the test program @computer @work
```

If you do not add a context, the task is automatically given the context @Anywhere.

There are some common contexts that can be entered in a abbreviated form. A full list is given in the Context abbreviations section.

3.3.2.1 Custom abbreviations

from version 1.84
You can also create you own custom abbreviations for contexts. To create an abbreviation, simply use

the `ABBREV` or `AB` command followed by the abbreviation and finally the required expanded form. So to set @Di to expand to @Diary you would enter:

```
>>>ABBREV @Di @Diary
```

If you omit the @ character in front of the expanded form, it will added automatically. So the following entry would achieve the same result:

```
>>>AB @Di Diary
```

To list all the current abbreviations `ABBREV ?` or `AB ?`.

To remove an abbreviation just leave the expanded form blank.
So to remove shortcut @Di just enter:

```
>>>ABBREV @Di
```

3.3.3 Adding projects

Projects are very similar to contexts in that they provide another way of categorising your tasks. To assign a task to a project, just embed the project name in the task preceded by the characters `:p`.

So imagine that on top of the work we are doing to learn how to use this program, we also have a project to decorate the bathroom at home; this project is our Bathroom project. We can enter the task like this:

```
>>>+ Buy ceramic tiles @Home :pBathroom
```

Note that project names can only be single words.

3.3.3.1 Project abbreviations

from version 1.85
You can also create you own custom abbreviations for projects. To create an abbreviation, simply use the `PAB` command followed by the abbreviation and finally the required expanded form. So to set :pi to expand to :pSoftware you would enter:

```
>>>PAB :pi :pSoftware
```

If you omit the :p characters in front of the expanded form, they will added automatically. So the following entry would achieve the same result:

```
>>>PAB :pi Software
```

To list all the current abbreviations `PAB ?`.

To remove an abbreviation just leave the expanded form blank.
So to remove shortcut :pi just enter:

```
>>>PAB :pi
```

3.3.4 Setting a date

Some tasks will need to be started on a particular day. IKog provides a simple means for doing that as well. To assign a date to a task, simply embed the date in the task, preceding the date with the characters `:d`. All dates are entered in the ISO date format of YYYY-MM-DD. So to set a task to occur on a particular day we can enter.

```
>>>Take the cat to the vet :d2006-08-30
```

Once a task has been set to start on a particular date, its priority is handled slightly differently. On any day prior to the assigned date, the task is treated as having a priority of 0. After all, until the date arrives you are not going to do anything with it, and it is treated as unimportant. Once the date arrives, the task's priority is increased by 11, making it the most important of your tasks - you said it had to be done on that day so iKog is going to put it to the top of your list.

There are some short cuts for adding dates that can make adding them even quicker. If you omit the year or the month and year, they automatically default to the year or month when you entered the task. So assuming that today's date is 2006-08-05, the following entries would all achieve the same thing.

```
>>> Take the cat to the vet :d2006-08-30
```

```
>>> Take the cat to the vet :d08-30
```

```
>>> Take the cat to the vet :d30
```

from version 1.59
Another shortcut for entering dates is the `:d+X` format. This creates a date equal to today's date + X days. So `:d+1` would be tomorrow.

---

from version 1.73
If you only enter the day, eg. `:d21`, the program automatically assumes that you want this to be a day in the future. So if today's date is 2006-08-21, then entering `:d5` will result in an actual date of 2006-09-05. This may be helpful if trying to simulate the GTD technique of a tickler file. If the resulting date is invalid, e.g. 2006-04-31 the date will be adjusted to the earliest valid date, in this case 2006-04-30.

Although the ISO format uses `-` as the separator, ikog.py is a little more flexible and allows `-`, `/` or even `:` to be used. So again we could have entered the task as:

```
>>> Take the cat to the vet :d2006/08/30
```

```
>>> Take the cat to the vet :d2006:08:30
```

You can only assign one date to a task. If you assign more than one, only the last date will be used.

3.3.4.1 Adding a meeting

from version 1.82

There is a special context to handle meetings. This is the @Meeting context or @M in its abbreviated form. This is always used with a date. As you know from the section on dates, when a task becomes overdue, it's immediately assigned a high priority. If it also has an @Meeting context it is assigned an even higher priority. So the order of tasks becomes:

1. Tasks with a date equal to today or earlier and which have the @Meeting context.
2. Tasks with a date equal to today or earlier.
3. Other tasks in order of priority and which do not have a date set.
4. Task which have a date set after today.

So to add a meeting we could enter:

```
>>>@Meeting :d2006-10-12 Discuss with Bob the new function. In his office at 9:00am
```

3.3.5 Adding a note

from version 1.63

The `NOTE` or `NOTES` command can be used to add a task with a priority of 0 and a context of @Notes. These are snippets of information that you do not intend to action as a task, hence the priority of 0, but which you want to store in your list of tasks. When used with the encryption facility these can also allow you to use iKog as a store for your private information. See Encrypting your data.

```
>>>note Beer is my favorite drink
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] Beer is my favorite drink
 Priority: 00
 Context: @Notes
 Created: [2006-09-01]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

3.3.6 Formatting your text

from version 1.67

iKog does not really provide any tools for formatting your text. However, it does allow you to embed line breaks in your tasks. To do this, just embed `<BR`> or `<br>` in the task. When the line is displayed a line break will be inserted.

e.g.

```
>>>+ a formatted task<br> with more than one line. #9 @c
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[01] a formatted task
with more than one line.
 Priority: 09
 Context: @Computer
 Created: [2006-09-04]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

3.4 Encrypting your data

from version 1.62

IKog allows you to encrypt the information that you enter so that other people will not be able to read the data stored in the ikog.py file. To encrypt text in a task, just embed the `<private>` or `<secret>` tags within the task details. Alternatively you can use their abbreviated forms of `<p>` and `<s>`.

The two tags offer different strength encryption algorithms. The <private> tag is the least secure but is built-in to the program. The <secret> tag is more secure but requires an additional module to be installed. You need to decide how well encrypted your data need to be.

Any text appearing after the tag will be encrypted. You will be asked to enter a password to use for encrypting the information.

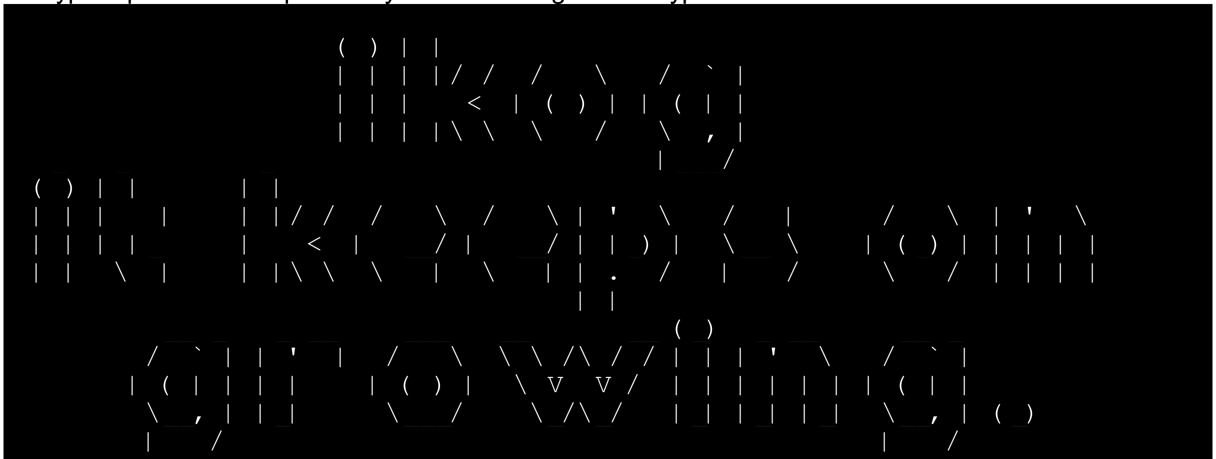
```
>>>+ This bit of text is normal @Pw <private>My secret information
Enter the master password.>>>
Re-enter the master password >>>
```

Although the password is referred to as a master password, you are not obliged to used the same password for each task; you'll need a good memory if you don't though.

Note that you cannot add special text, like contexts, dates and projects in the encrypted text. You also cannot use the filter tools on the encrypted text. If you want any text available so you can use the filter command to find a task, make sure you enter that text in the unencrypted area.

When you are prompted to enter your password, nothing will be displayed. If you are running on Linux, you are probably use to this behaviour. If you are used to running on Windows you might have expected to see `****` being displayed while you type. Don't worry, it is supposed to display nothing.

Once you have entered your task, the screen will be cleared. When your task is redisplayed the encrypted part will be replaced by text indicating the encryption.



```
Tasks saved.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] This bit of text is normal <*** private xtea ***>
 Priority: 05
 Context: @Password
 Created: [2006-09-01]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Although the screen is cleared, the information will still be available to view if you scroll back through the terminal history. As such you should quit iKog when you leave the computer. The clearing of the screen is just to protect your data from people looking over your shoulder. See also Clearing the screen.

To redisplay the encrypted part of the task, just use the `SHOW` or `SH` command followed by the number of the task you want to view. You will be asked to enter the password to decrypt the data. Obviously this needs to be the same as the password you used to encrypt the data in the first place.

```
>>>show 0
Enter your master password >>>
My secret information
Press enter to clear screen and continue.
```

3.4.1 Private encryption

The private tag uses the *Xtea* encryption algorithm. This a relatively secure encryption technique though there are some weaknesses. See http://en.wikipedia.org/wiki/XTEA for more information. Support for this algorithm is a built-in feature of iKog.

The code for the algorithm base on the public domain code by Paul Chakravarti; see http://aspn.activestate.com/ASPN/Cookbook/Python/Recipe/496737 .

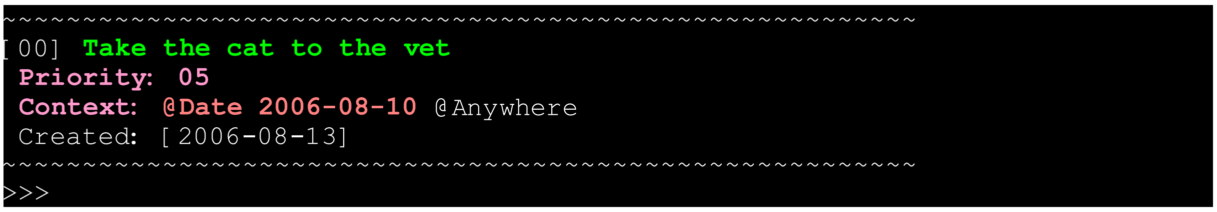
3.4.2 Secret encryption

The `<secret>` tag uses the AES - Rijndael encryption algorithm with a 256 bit Rijndael cipher. This is more secure than the Xtea algorithm used with the `<private>` tag. See http://en.wikipedia.org/wiki/Advanced_Encryption_Standard for more information.

To use this encryption. you will need to download the *pyRijndael.py* module by Jeffrey Clement, available from http://jclement.ca/software/pyrijndael/ . You can either install this into your Python installation or just save it in the same folder as ikog.py.

3.5 Navigating through the tasks

At the bottom of the screen, the current task is displayed as shown below:



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] Take the cat to the vet
 Priority: 05
 Context: @Date 2006-08-10 @Anywhere
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
```

To navigate through the list, we can use the `NEXT` or `PREV` commands or their abbreviated forms `N` and `P`.

So if we use these commands we might get:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] Take the cat to the vet
 Priority: 05
 Context: @Date 2006-08-10 @Anywhere
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>next
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[01] Learn how to use this program
 Priority: 09
 Context: @Computer
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>n
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[02] Buy ceramic tiles for the bathroom
 Priority: 05
 Context: @Anywhere
 Projects: Bathroom
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>prev
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[01] Learn how to use this program
 Priority: 09
 Context: @Computer
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
```

To jump to the top of the list again, you can use the `TOP` or `T` command.

from version 1.59
You can follow the `TOP` or `T` command with a number of tasks you want to list. So for example:

```
>>>LIST 3
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] @Date 2006-08-10 Take the cat to the vet #5 @Anywhere [2006-08-01]
[01] Learn how to use this program #9 @Computer [2006-08-13]
[02] Buy ceramic tiles for the bathroom #5 @Anywhere Projects: Bathroom,
[2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] Take the cat to the vet
 Priority: 05
 Context: @Date 2006-08-10 @Anywhere
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
```

To jump to a specific task, use the `GO` or `G` command followed by the number of the task you want to go to. e.g. to go to task 6 enter:

```
>>>GO 6
```

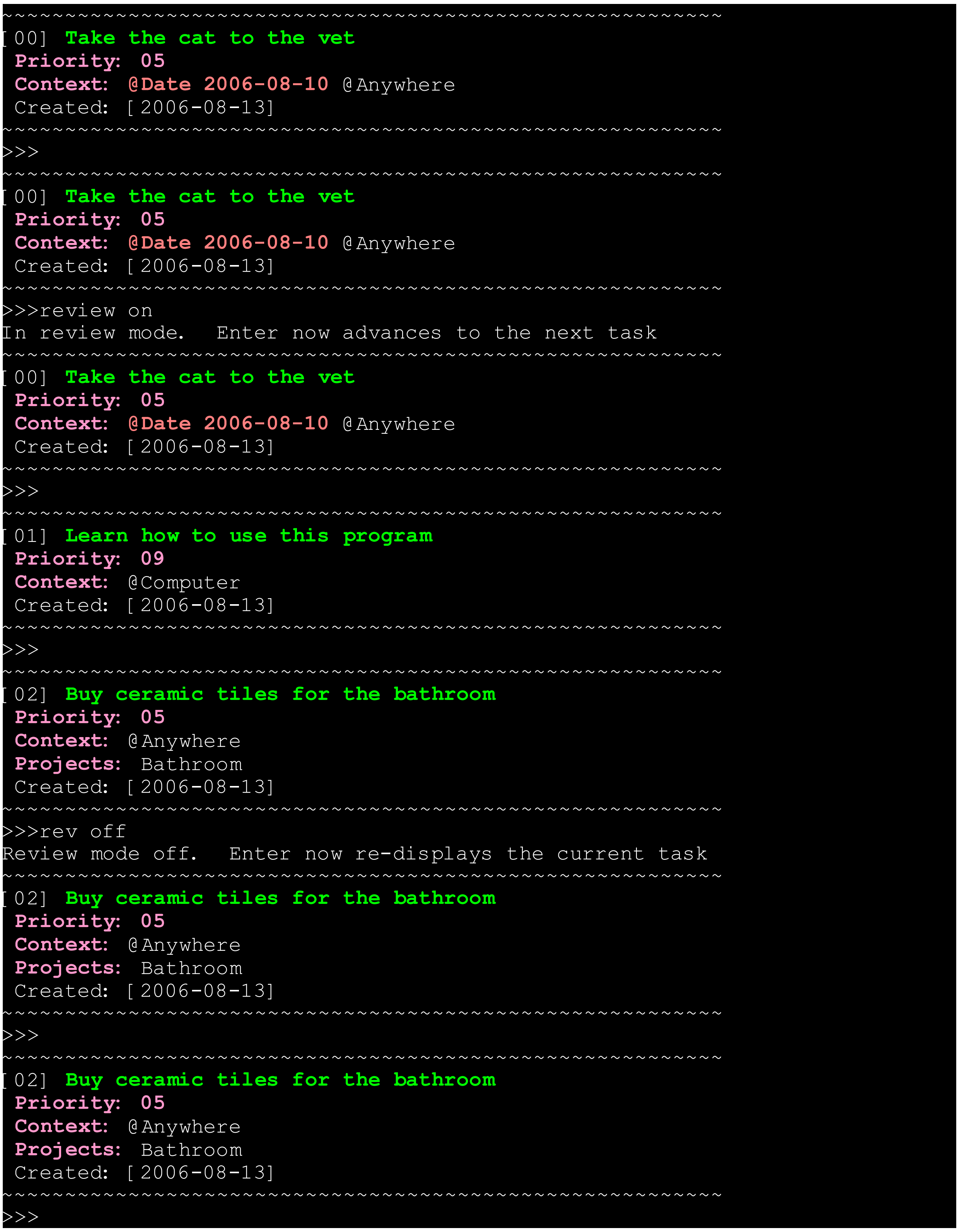
Pressing the **<enter>** key normally just re-displays the current task. If you are reviewing your list of tasks it would be more convenient to just keeping pressing **<enter>** to move through the list. You can use the review mode to do this. See Review mode.

The listing above shows a colour screen. Not all terminals support colour and the default is with colour switched off so your screen may look a little different. If you want to use colour check the Using colour section.

3.5.1 Review mode

Normally when you hit the **<enter>** key, the current task is just re-displayed. In review mode, hitting the **<enter>** key automatically advances to the next task. To enter review mode just enter the `REVIEW ON` or `REV ON` command. To revert back to the normal mode, just enter the `REVIEW OFF` or `REV OFF` command. The example below shows how this works.

The `>>>` lines are where the **<enter>** key was pressed without entering a command.



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] Take the cat to the vet
 Priority: 05
 Context: @Date 2006-08-10 @Anywhere
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] Take the cat to the vet
 Priority: 05
 Context: @Date 2006-08-10 @Anywhere
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>review on
In review mode.  Enter now advances to the next task
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] Take the cat to the vet
 Priority: 05
 Context: @Date 2006-08-10 @Anywhere
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[01] Learn how to use this program
 Priority: 09
 Context: @Computer
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[02] Buy ceramic tiles for the bathroom
 Priority: 05
 Context: @Anywhere
 Projects: Bathroom
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>rev off
Review mode off.  Enter now re-displays the current task
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[02] Buy ceramic tiles for the bathroom
 Priority: 05
 Context: @Anywhere
 Projects: Bathroom
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[02] Buy ceramic tiles for the bathroom
 Priority: 05
 Context: @Anywhere
 Projects: Bathroom
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
```

If you find yourself flipping in and out of review mode, you can also use the shortcuts of `V1` (re**V**iew on) and `V0` (re**V**iew off)

3.5.2 Viewing tasks

Okay, we've add lots of tasks to the task list, so how can we view them to see what needs to be done. IKog provides a number of means for displaying tasks. These are the list, context, date and project views. Each of the commands has an alternative form that sends the output to an HTML file ready for printing. The HTML outputs use the same commands but with the `>` character at the end, e.g `LIST` becomes `LIST>`.

The program will try to launch a browser window when the HTML report is generated. On some systems, you may need to use the command twice before the report is displayed.

from version 1.86
The html reports will try to use a style sheet if one exists. If you want to use this feature, place a style sheet called *ikog.css* in the same folder as ikog.py. An example file is provided in the downloads section.

3.5.2.1 Viewing a list

To display a simple list, just enter `LIST` or `L`. A list of all your tasks will be displayed. Those with the highest priority will be at the top and the lowest at the bottom.

```
>>>LIST
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] @Date 2006-08-10 Take the cat to the vet #5 @Anywhere [2006-08-01]
[01] Learn how to use this program #9 @Computer [2006-08-13]
[02] Buy ceramic tiles for the bathroom #5 @Anywhere Projects: Bathroom,
[2006-08-13]
[03] Visit the 43 folders website #5 @Computer [2006-08-13]
[04] Buy a new bath and taps #5 @Home Projects: Bathroom, [2006-08-13]
[05] @Date 2006-12-19 Buy birthday card for Joe #5 @Anywhere [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

To output to an HTML report, use `LIST>` of `L>`.

from version 1.83
After a list command, the current task is printed in a single line compressed format so that the list does not scroll off screen when the current task is displayed.

3.5.2.1.1 Applying a list filter

If you have a lot of tasks, you will soon realise that a simple list becomes too cumbersome. To make the tasks easier to view, you can add a filter to the list command. A filter simply restricts the list to either a date, project, context or minimum priority. To filter the list, just use the list command but add either the date, project, context or priority to the end of the command. So if we try the list command with and without filters we can see how the output changes:

```
>>>LIST
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] @Date 2006-08-10 Take the cat to the vet #5 @Anywhere [2006-08-01]
[01] Learn how to use this program #9 @Computer [2006-08-13]
[02] Buy ceramic tiles for the bathroom #5 @Anywhere Projects: Bathroom,
[2006-08-13]
[03] Visit the 43 folders website #5 @Computer [2006-08-13]
[04] Buy a new bath and taps #5 @Home Projects: Bathroom, [2006-08-13]
[05] @Date 2006-12-19 Buy birthday card for Joe #5 @Anywhere [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>LIST :pbathroom
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Filter = :pbathroom
[02] Buy ceramic tiles for the bathroom #5 @Anywhere Projects: Bathroom,
[2006-08-13]
[04] Buy a new bath and taps #5 @Home Projects: Bathroom, [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>LIST @c
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Filter = @c
[01] Learn how to use this program #9 @Computer [2006-08-13]
[03] Visit the 43 folders website #5 @Computer [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>LIST :d2006-12-19
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Filter = :D2006-12-19
[05] @Date 2006-12-19 Buy birthday card for Joe #5 @Anywhere [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>LIST #8
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] @Date 2006-08-10 Take the cat to the vet #5 @Anywhere [2006-08-01]
[01] Learn how to use this program #9 @Computer [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Note how the filter is always displayed in the output so you can see that the list is not necessarily complete.

The priority filter is slightly different to the other filters in that in it matches any task with an equal or **higher** priority. In the example above, note that the overdue task, even though its original priority was only #5, was still matched by the filter of priority #8. This is because the priority of overdue tasks is automatically increased by 11.

If you enter more than one term in your filter, then both terms must be matched. So you can enter:

```
>>>LIST @c @w
```

This will match items with a context of @Computer **and** @Work.

---

from version 1.58

You can also filter tasks that contain specific text by including them in the filter. For example:

```
>>>LIST
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] @Date 2006-08-10 Take the cat to the vet #5 @Anywhere [2006-08-01]
[01] Learn how to use this program #9 @Computer [2006-08-13]
[02] Buy ceramic tiles for the bathroom #5 @Anywhere Projects: Bathroom,
[2006-08-13]
[03] Visit the 43 folders website #5 @Computer [2006-08-13]
[04] Buy a new bath and taps #5 @Home Projects: Bathroom, [2006-08-13]
[05] @Date 2006-12-19 Buy birthday card for Joe #5 @Anywhere [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>LIST buy tiles
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Filter = buy tiles
[02] Buy ceramic tiles for the bathroom #5 @Anywhere Projects: Bathroom,
[2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Note that the filter will match parts of words and is case insensitive. So, for example,

```
>>>LIST art
```

would find *art*, *ART* or *party*.

---

from version 1.59
If you precede any filter term with the `-` character, then only tasks that **do not** match the filter will be shown.

So:

```
>>>LIST @c
```

would show all tasks with a context of @Computer, whereas:

```
>>>LIST -@c
```

would show all tasks that do not have a context of @Computer

---

from version 1.84
You can also match tasks that contain either words by including the term `OR`. See More complex filters
.

3.5.2.2 Viewing by context

To display a list grouped by context, just enter `@`.

```
>>>@
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
@Anywhere
[02] Buy ceramic tiles for the bathroom #5 @Anywhere Projects: Bathroom,
[2006-08-13]
[05] @Date 2006-12-19 Buy birthday card for Joe #5 @Anywhere [2006-08-13]

@Computer
[01] Learn how to use this program #9 @Computer [2006-08-13]
[03] Visit the 43 folders website #5 @Computer [2006-08-13]

@Home
[04] Buy a new bath and taps #5 @Home Projects: Bathroom, [2006-08-13]

@Date
[00] @Date 2006-08-10 Take the cat to the vet #5 @Anywhere [2006-08-01]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

To output to an HTML report, use `@>`.

3.5.2.3 Viewing by project

To display list grouped by project, just enter `: p`.

```
>>>: p
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Project: Bathroom
[02] Buy ceramic tiles for the bathroom #5 @Anywhere Projects: Bathroom,
[2006-08-13]
[04] Buy a new bath and taps #5 @Home Projects: Bathroom, [2006-08-13]

Project: None
[00] @Date 2006-08-10 Take the cat to the vet #5 @Anywhere [2006-08-01]
[01] Learn how to use this program #9 @Computer [2006-08-13]
[03] Visit the 43 folders website #5 @Computer [2006-08-13]
[05] @Date 2006-12-19 Buy birthday card for Joe #5 @Anywhere [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

To output to an HTML report, use `: p>`.

3.5.2.4 Viewing by dates

To display a list grouped by date, just enter `: d`. Note that only tasks with dates are displayed.

```
>>>: d
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Date: 2006-08-10
[00] @Date 2006-08-10 Take the cat to the vet #5 @Anywhere [2006-08-01]

Date: 2006-12-19
[05] @Date 2006-12-19 Buy birthday card for Joe #5 @Anywhere [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

To output to an HTML report, use `: d>`.

3.5.3 Moving the tasks

You can move tasks up and down the task list by using the `FIRST`, `UP` or `DOWN` commands or their abbreviated forms, `F`, `U` or `D`. To use these commands just enter the command followed by the number of the task you want to move. e.g to move task 4 up the list just enter:

```
>>> UP 4
```

The `FIRST` command moves the task to the top of the list whilst still honoring priorities. The `UP` command moves the task one place higher and the `DOWN` command moves it lower.

As an example, consider the following list:

```
[00] Buy me a beer #8 @Anywhere [2006-08-15]
[01] Buy the dog a kennel #5 @Anywhere [2006-08-15]
[02] Buy the cat a basket #5 @Anywhere [2006-08-15]
[03] Buy the hamster a wheel #5 @Anywhere [2006-08-15]
```

Now lets move task 2, the cat's basket, to the top. We enter:

```
>>>FIRST 2
```

If we now enter the list command, what do we get?

```
[00] Buy me a beer #8 @Anywhere [2006-08-15]
[01] Buy the cat a basket #5 @Anywhere [2006-08-15]
[02] Buy the dog a kennel #5 @Anywhere [2006-08-15]
[03] Buy the hamster a wheel #5 @Anywhere [2006-08-15]
```

Although we tried to move the task to the top, the priorities are still honored. So it went to the top of all the tasks with priority 5 but still finished behind buying me a beer. The program works perfectly.

It is important to remember that although you can try to move tasks using the `FIRST`, `UP` and `DOWN` commands, the priority takes precedence.

3.5.4 Applying a global filter

When you list your tasks you can append a filter to the list command - See Applying a list filter. Lets say you are at the computer and you only want to see the tasks with the @Computer context. Then, instead of adding the filter every time you enter a command, you can apply a global filter. To apply a filter just enter `FILTER` or `FI` followed by the filter. Like the list command, the filter can be either a date, project, context or minimum priority.

Let's list our tasks and then apply a filter

```
>>>LIST
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] @Date 2006-08-10 Take the cat to the vet #5 @Anywhere [2006-08-01]
[01] Learn how to use this program #9 @Computer [2006-08-13]
[02] Buy ceramic tiles for the bathroom #5 @Anywhere Projects: Bathroom,
[2006-08-13]
[03] Visit the 43 folders website #5 @Computer [2006-08-13]
[04] Buy a new bath and taps #5 @Home Projects: Bathroom, [2006-08-13]
[05] @Date 2006-12-19 Buy birthday card for Joe #5 @Anywhere [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>fi @c
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Filter = @c
[01] Learn how to use this program
 Priority: 09
 Context: @Computer
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>LIST
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Filter = @c
[01] Learn how to use this program #9 @Computer [2006-08-13]
[03] Visit the 43 folders website #5 @Computer [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
```

Note how the filter not only applies to the list but also to the current task. When you navigate through the tasks only those matching the filter will be displayed.

If you enter more than one term in your filter, then both terms must be matched. So you can enter:

```
>>>FILTER @c @w
```

This will match items with a context of @Computer **and** @Work.

If you apply a list filter with the `LIST` or `LIST>` commands, the filter from the `FILTER` command is applied as well. So the tasks will need to match the list filter **and** the global filter.

---

from version 1.59

If you precede any filter term with the `-` character, then only tasks that **do not** match the filter term will be shown.

3.5.4.1 More complex filters

---

from version 1.84

You can also match tasks that contain different words by including the term `OR`. So to match tasks that contain either Learn or Visit you would enter:

```
>>>FILTER Learn or Visit
```

If you enter more than 2 tasks the rule for the filter is as follows. The return task must contain all of the words that are not preceded by OR or any of the words preceded by OR. So:

```
>>>FILTER Learn or visit or ceramic
```

would return tasks containing Learn or visit or ceramic.

```
>>>FILTER Learn or visit ceramic
```

would return tasks containing (Learn and ceramic) or visit.

This can also be used with list filters. e.g.

```
>>>LIST Learn or visit ceramic
```

would return tasks containing (Learn and ceramic) or visit.

Note the character + at the start of a search is used programmatically by iKog to identify terms that must apply. You should not precede terms with + yourself.

3.5.5 Searching for tasks

from version 1.58

The global filter can also be used to help find tasks. When used in this way, it is best to run in review mode. To find the task you are looking for, set the global filter including the text you are looking for. e.g.

```
>>>FILTER text1 text2 etc
```

The first task containing **all** of the text will then be displayed. If you are in review mode, pressing **<enter>** will automatically move to the next task containing the text. If the filter cannot find any tasks to display, the filter will automatically be removed; after all there is no point in displaying nothing.

You can further restrict the returned results by using any of the normal filter elements such as projects and dates. See Applying a global filter for more information.

3.6 Changing the tasks

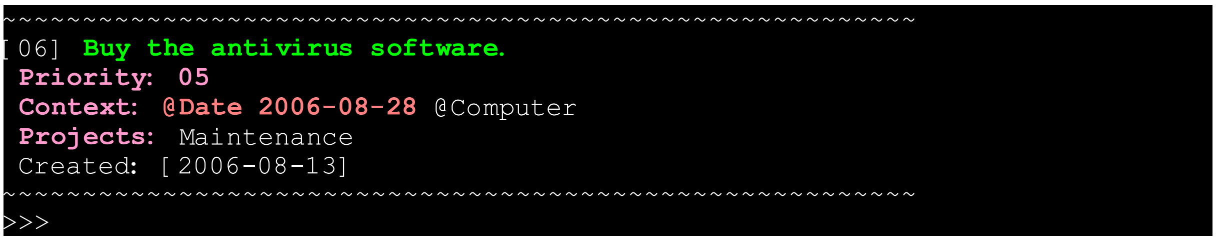
IKog provides a number of ways to modify your tasks once you've created them. There are three commands for changing your tasks: the replace command, `REP` or `R`; the modify command `MOD` or `M`; and the extend command, `EXTEND` or `E`.

Before you use the commands you should understand that every task you create has five elements: the description; the projects; the contexts; the date; and finally the priority. When you use the `REPLACE` command, all elements of the task are replaced by your new entry. When you use the `MODIFY` command, only elements you enter are changed. Finally, the `EXTEND` command just adds the new elements to the task. To use the commands you just enter the command followed by the number of the task and finally the new task entry.

See:
Replacing a task
Modifying a task
Extending a task
Substituting text

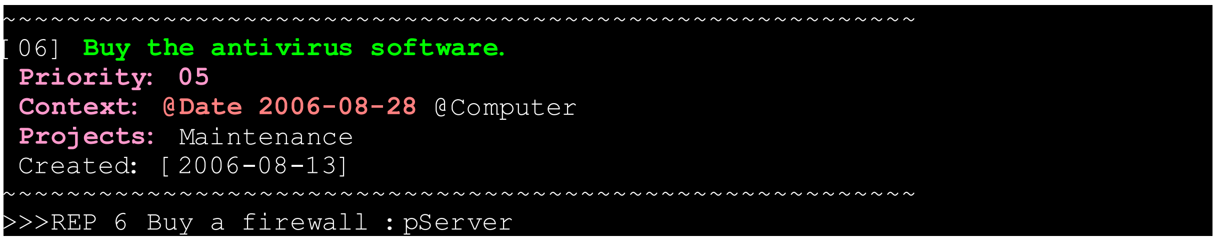
3.6.1 Replacing a task

The replace command replaces an entire task with a new entry. The format of the command is `REP N text` or `R N text` where `N` is the number of the task to change and `text` is the new task entry. To see how this works consider the following task.



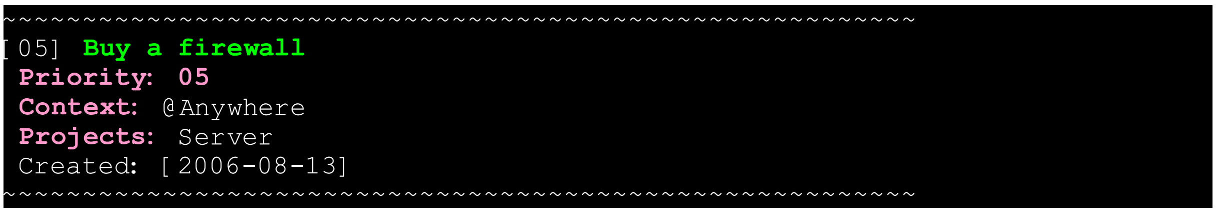
```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[06] Buy the antivirus software.
 Priority: 05
 Context: @Date 2006-08-28 @Computer
 Projects: Maintenance
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
```

The task number is shown in square brackets before the task description; in this case the number is 6. Let's see what happens when we use the command.



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[06] Buy the antivirus software.
 Priority: 05
 Context: @Date 2006-08-28 @Computer
 Projects: Maintenance
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>REP 6 Buy a firewall :pServer
```

The result is that the task is replaced by our new entry. Note that because the date has been removed, the task's position in the list may also change.



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[05] Buy a firewall
 Priority: 05
 Context: @Anywhere
 Projects: Server
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

If you don't enter the text, the program will prompt you to enter the new details.

When you change a task it may alter the order of other tasks in the list, and therefore the task numbers. If you are unsure about whether or not the task number of an item you want to edit has changed, use the `LIST` or `GO` command first to check the entry before you edit it.

3.6.2 Modifying a task

The modify command changes the elements that you enter but leaves any other elements unchanged. The format of the command is `MOD N text` or `M N text` where `N` is the number of the task to change and `text` is the new task entry. To see how this works consider the following task.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[06] Buy the antivirus software.
 Priority: 05
 Context: @Date 2006-08-28 @Computer
 Projects: Maintenance
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
```

The task number is shown in square brackets before the task description; in this case the number is 6. Let's see what happens when we use the command.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[06] Buy the antivirus software.
 Priority: 05
 Context: @Date 2006-08-28 @Computer
 Projects: Maintenance
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>MOD 6 Buy a firewall :pServer
```

The result is that the elements we enter are changed.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[06] Buy a firewall
 Priority: 05
 Context: @Date 2006-08-28 @Computer
 Projects: Server
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

In this case we changed the description and the project, but the date and context are unchanged.

If you don't enter the text, the program will prompt you to enter the new details.

A common occurrence is that you want to change the context of a task. To change the context of task 6 to @Work, you could just enter:

```
>>>M 6 @W
```

When you change a task it may alter the order of other tasks in the list, and therefore the task numbers. If you are unsure about whether or not the task number of an item you want to edit has changed, use the `LIST` or `GO` command first to check the entry before you edit it.

3.6.3 Extending a task

The extend command adds new elements to a task but leaves existing elements unchanged. The format of the command is `EXTEND N text` or `E N text` where `N` is the number of the task to change and `text` is the new task entry. To see how this works consider the following task.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[06] Buy the antivirus software.
 Priority: 05
 Context: @Date 2006-08-28 @Computer
 Projects: Maintenance
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
```

The task number is shown in square brackets before the task description; in this case the number is 6. Let's see what happens when we use the extend command.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[06] Buy the antivirus software.
 Priority: 05
 Context: @Date 2006-08-28 @Computer
 Projects: Maintenance
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>EXTEND 6 Buy a firewall :pServer
```

The result is that the task is extended by our new entries.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[05] Buy the antivirus software. ...Buy a firewall
 Priority: 05
 Context: @Date 2006-08-28 @Computer
 Projects: Maintenance, Server
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

If you don't enter the text, the program will prompt you to enter the new details.

A common occurrence is that you want to add an additional context to a task. To change the context of task 5 to @Work and @Computer, and noting that it already has the context of @Computer, you could just enter:

```
>>>E 5 @W
```

When you change a task it may alter the order of other tasks in the list, and therefore the task numbers. If you are unsure about whether or not the task number of an item you want to edit has changed, use the `LIST` or `GO` command first to check the entry before you edit it.

3.6.4 Substituting text

from version 1.77

The substitute command adds you to change text or phrases in a task. The format of the command is

`SUB N /s1/s2/` or `SU N /s1/s2/` where `N` is the number of the task to change, `s1` is the text you want to change and `s2` is the text you want to change it to. To see how this works consider the following task.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[06] Buy the antivuris software.
 Priority: 05
 Context: @Date 2006-08-28 @Computer
 Projects: Maintenance
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
```

The task number is shown in square brackets before the task description; in this case the number is 6. Let's see what happens when we use the substitute command.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[06] Buy the antivuris software.
 Priority: 05
 Context: @Date 2006-08-28 @Computer
 Projects: Maintenance
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>SUB 6 /vuris/virus/
```

The result is that the text vuris is replaced by virus.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[06] Buy the antivirus software.
 Priority: 05
 Context: @Date 2006-08-28 @Computer
 Projects: Maintenance
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

You can substitute entire phrases using this command. So you could enter:

```
>>>SUB 6 /antivirus software/internet security suite/
```

The / character is used to separate the phrases. If you need to actually use the / character as part of your phrase, precede the character with a \ character. So to replace E/mail with E-mail in task 7 you would enter:

```
>>>SUB 7 /E\/mail/E-mail/
```

Note, this command only acts on the task text.

You can also delete phrases by leaving the second phrase empty as follows:

```
>>>SUB 6 /phrase to delete//
```

3.7 Completing and removing tasks

When you complete a task, you would normally just want to remove it permanently. This assumes that you do not want a record of what you have done. See Removing a task for details.

---

from version 1.86

If you do want to keep a record of work you have done, you can use the archive or done command. See Archiving a task for details.

3.7.1 Removing a task

To remove a task, for any reason, just use the `KILL`, `K` or `X` command followed by the number of the task you want to remove. e.g. to remove task 6 just enter:

```
>>>KILL 6
```

IKog doesn't care why you are removing the task - you've finished the task or it was just a mistake, it doesn't matter.

To remove every task, use the `CLEAR` command. e.g.

```
>>>CLEAR
```

from version 1.72
If you use a task number to delete a task, you will be prompted to confirm whether or not you actually want to delete the task. This is because the task you want to delete may not be currently displayed. As with all editing commands you can also use the `^` or `THIS` option to remove the currently displayed task. See Editing commands. Because the current task will be visible, the program will not ask you to confirm the deletion.

So if the current display shows:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[06] Buy the antivirus software.
 Priority: 05
 Context: @Date 2006-08-28 @Computer
 Projects: Maintenance
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
```

Then entering:

```
>>>K 6
```

will ask you to confirm that you actually want to remove the task.

But if you want to do it a bit more quickly,

```
>>K ^
```

or

```
>>>K this
```

will remove task 6 without prompting.

3.7.2 Archiving a task

from version 1.86
To remove and archive a task, for any reason, just use the `ARCHIVE`,or `DONE` command followed by the number of the task you want to remove and any optional notes you might want to add. e.g. to archive task 6 just enter:

```
>>>ARCHIVE 6 I have finished this
```

When a task is archived, its date is changed to today, it is given a context of @Archived and then it is deleted from the tasks list and moved to a archive file. The archive file has the same name as the current script file with *archived.dat* appended. So the file name will normally be *ikog.py.archived.dat*.

As this file is a normal ikog data file, you can use the `OPEN` command to look at its contents. See Separating your data for more details about the `OPEN` command.

If you use a task number to archive a task, you will be prompted to confirm whether or not you actually want to archive the task. This is because the task you want to archive may not be currently displayed. As with all editing commands you can also use the `^` or `THIS` option to remove the currently displayed task. See Editing commands. Because the current task will be visible, the program will not ask you to confirm the archive.

So if the current display shows:

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[06] Buy the antivirus software.
 Priority: 05
 Context: @Date 2006-08-28 @Computer
 Projects: Maintenance
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
```

Then entering:

```
>>>ARCHIVE 6
```

will ask you to confirm that you actually want to remove the task.

But if you want to do it a bit more quickly,

```
>>ARCHIVE ^
```

or

```
>>>ARCHIVE this
```

will archive task 6 without prompting.

3.8 Saving your work

You can configure ikog.py to save your changes automatically as soon as you make them. To switch on this autosave option, just enter `AUTOSAVE ON` or `AS ON`.

Although the autosave options ensures that your work is always saved, it will result in a lot of disk thrashing. If you have a lot of tasks, you may prefer to manually save your work.

If you switch off the autosave option, you do run the risk of losing your changes if you close the terminal that iKog is running in. If in doubt, leave autosave on.

To switch off the autosave, enter `AUTOSAVE OFF` or `AS OFF`. Now when you make a change that needs changing, you will see the command prompt change from:

```
>>>
```

to

```
!>>
```

This makes it very easy to see whether you have work that needs saving. When you quit the program using the `QUIT` or `Q` command, the changes are automatically changed. However, you can you can save the list at any time just by entering the `SAVE` or `S` command. e.g to save the tasks, enter:

```
>>> SAVE
```

3.8.1 Exporting and importing

If you want to export your tasks to a text file, just use the `EXPORT` command. This will create a file, *ikog.py.tasks.txt* with just the task details. The file format is really only intended for use with the `IMPORT` command.

If you had renamed the program, the export file name will also change. So if you renamed the program, *MYPROG.py*, the export file name would be *MYPROG.py.tasks.txt.*

To import tasks from a file created by the `EXPORT` command just use the `IMPORT` command followed by the filename containing the tasks. e.g.

```
>>>IMPORT ikog.py.tasks.txt
```

3.9 Using colour

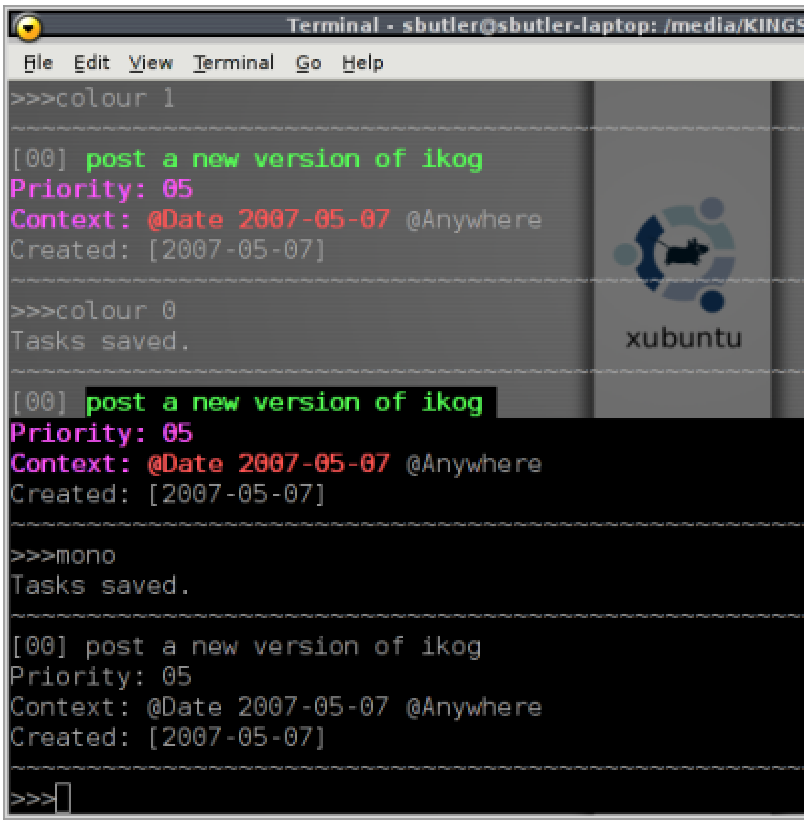
If you want a colour display, enter the command `COLOR`, `COLOUR` or `COL`. This setting will be remembered so you will not have to keep entering it.
To switch off the colour mode, enter the command `MONOCHROME` or `MONO`.

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] Take the cat to the vet
 Priority: 05
 Context: @Date 2006-08-10 @Anywhere
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>COLOUR
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] Take the cat to the vet
 Priority: 05
 Context: @Date 2006-08-10 @Anywhere
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>DEFAULT
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[01] Learn how to use this program
 Priority: 09
 Context: @Computer
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
```

If you set the colour option and then run the program on a Windows computer, the display will still appear in monochrome. This is because the Windows terminal does not support colour and iKog will not bother trying to set the colour of the text.

If you find your tasks become garbled and contain little sequences like `[0;37;40m`, this means that your Mac or Linux terminal does not support ANSI colour. If this occurs, switch off the colour using the `MONO` command.

from version 1.87
If you do use colour, you can also select optional colour sets by adding the set number after the colour command. So to choose set 1 you would enter `COLOUR 1` or `C 1`. At present there are two sets available. Set 0 is coloured text on a black background. Set 1 is the same coloured text but the background is left unchanged. This is useful if you use a terminal in a transparent mode. The screen shot below shows examples of the colour sets.


Terminal - sbutler@sbutler-laptop: /media/KINGS
File Edit View Terminal Go Help
xubuntu

```
>>>colour 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] post a new version of ikog
Priority: 05
Context: @Date 2007-05-07 @Anywhere
Created: [2007-05-07]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>colour 0
Tasks saved.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] post a new version of ikog
Priority: 05
Context: @Date 2007-05-07 @Anywhere
Created: [2007-05-07]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>mono
Tasks saved.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[00] post a new version of ikog
Priority: 05
Context: @Date 2007-05-07 @Anywhere
Created: [2007-05-07]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
```

If you set the colour set to a negative number, the colour option is switched off.

3.10 Clearing the screen

With Windows system, it is possible to clear the screen by calling a system command, *cls*. This is the best way of clearing the screen when you have sensitive data displayed such as a password. Unfortunately, some security suites such as ZoneAlarm, may pop-up a warning that Python is trying to perform a system call. As such, by default, ikog.py does not use system calls.

However, if you are comfortable with the message you can tell iKog that you would like to use system calls. To allow system calls use the command `SYS ON` and to prevent system calls use the command `SYS OFF`. If system calls are not allowed, the visible display is still cleared but the information can still be viewed by scrolling back through the terminal. Disabling system calls also prevents use of the ! CMD feature.

On some Linux terminals use of the system call, *clear*, may still allow the data to be viewed by scrolling back through the terminal history.

from version 1.64
The screen is automatically cleared after secret or private data has been displayed but you can also use the `CLS` or `CLEARSCREEN` command at any time.

3.11 Running system commands

from v1.75
If you want to run system commands you can use `! CMD` command followed by the system command

you want to run. For example, in Windows if you want to see a quick listing of the current directory, you can enter:

```
>>>! CMD dir
```

or on Linux:

```
>>! CMD ls
```

By default, iKog, does not permit system calls. To allow system calls use the command `SYS ON` and to prevent system calls use the command `SYS OFF`. If system calls are not allowed, you will not be able to use the `! CMD` feature. Disabling system calls also prevents them from being used when clearing the screen.

Use system calls carefully.

from v1.76

To provide you with some safety, ikog.py will not allow you to run *rm*, *rmdir* or *del* commands

3.12 Using an external editor

from version 1.77

It is possible to use an external editor to edit your tasks. It is normally faster to try to learn the editing commands built into ikog.py but if you want to use an editor you can.

You can only use this feature on Linux and Windows.

To use an external editor, you must have enabled system calls using the `SYS ON` command. See Running system commands.

The format of the command is `EDIT [N]` or `ED [N]` where `N` is the number of the task you want to edit. If you omit `N`, a new task is created. When you save the task from the external editor, the task is created or modified as appropriate.

The default editors for Linux are *nano*, *pico*, *vim* and finally *emacs*. IKog uses the first editor it finds starting with *nano*. For Windows, the default editor is *edit*. If you want to change the editors, use the `SETEDPOSIX editor` command for the Linux editors and `SETEDNT editor` for Windows. If you want to enter a number of editors to search for, separate them with commas.

So to set the default editor on Linux to *Vim* and only use *pico* if *Vim* isn't found, enter:

```
>>>SETEDPOSIX vim,pico
```

To set the default editor on Windows to *notepad*, enter:

```
>>>SETEDNT notepad
```

Because a temporary file is used to allow the external editor to be used, do not enter secret or private data. Although ikog.py deletes the temporary file, if your computer crashed you could finish up with your private data visible in a file on your computer.

3.13 Quick tips and tricks

Always begin your tasks with a context so you don't have to use the ADD or + command.

```
>>>@c  6 not time to do it :d+1
```

Remember this trick only works if your entry is longer than 10 characters — it probably will be.

Use the extend and the :d+X format command to postpone until tomorrow, including some notes explaining why. Here we postpone task 6.

```
>>>e 6 no time to do it at the moment :d+1
```

Take a quick look at the top five tasks you should be working on.

```
>>>t 5
```

Use the modify command to quickly change the context of a task. Here we change the context of task 5 to @Computer.

```
>>>m 5 @c
```

Learn to use global filters to help manage your tasks. Just arrived at work but the network's down. So I only want to see tasks that have a context of @Work but **do not** have a context of @Computer.

```
>>>fi @w -@c
```

I want to print a to-do list that I can work with away from the computer. I obviously don't want to see tasks with a context of @Computer so use a filter excluding the @Computer context.

```
>>>l> -@c
```

Use the :dX format to quickly move tasks into the future.

```
>>>m ^ :d29
```

Use the sub command to correct spelling mistakes.

```
>>>sub 7 /vuris/virus/
```

Add a password using encryption.

```
>>>note @pw My art forum login <p>hdf8sle0008*
```

3.13.1 Storing passwords

from version 1.62

Although not the program's main aim, it is possible to use ikog.py as a poor man's password manager.

Note I am not a cryptography expert so whether or not you trust the algorithms to store your passwords is up to you.

Add my password to the system — its added as a note because it isn't a task that needs any action.

```
>>>note @pw login password to the henspace forum <s>t6A4hjKP
```

Now find that password.

```
>>>fi @pw henspace

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Filter = @Password henspace
[ 26] login to the henspace forum <*** secret aes ***>
 Priority: 00
Context: @Notes @Password
Created: [2006-09-01]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

Now reveal the password.

```
>>>sh 26
Enter your master password >>>
t6A4hjKP
Press enter to clear screen and continue.
```

3.14 Separating your data

from version 1.66

Normally the tasks are embedded in the main program file so all you have to carry around with you is the ikog.py file. The advantage is that you only have one file to look after; the disadvantage is that every time you save a task you have to save the program as well.

If you want, you can keep your tasks in a separate file. When iKog starts it looks for a file called *ikog.py.dat*. If it finds the file, then it will read and write tasks to that file. If the file is not found, then it will read and write tasks to itself.

Follow these instructions to tell the program to use a separate data file.

1. Use the `EXPORT` command to create a file *ikog.py.tasks.txt*
2. Use the `CLEAR` command to remove the tasks from your main *ikog.py* program.
3. Use the `QUIT` command to close the program.
4. Rename the exported file from *ikog.py.tasks.txt* to *ikog.py.dat*
5. When next started, iKog will now use this file for storing your tasks.

Remember if you are carrying the program around on your USB stick you now need two files; *ikog.py* and *ikog.py.dat*

If you had renamed the program, the data name will also have to change. So if you renamed the program, *MYPROG.py*, the external data file name will be *MYPROG.py.dat*.

from version 1.69
You can also switch to another external data file at any time by using the `OPEN` or `O` command. To switch to a data file called *mydata.dat* you can just enter:

```
>>>OPEN mydata
```

Note that you do not need to add the .dat extension.

To create a new data file just use the `NEW` command followed by the name of the data file. Again you do not need to add the .dat extension. e.g.

```
>>>NEW mynewdata
```

Configuration data, such as review mode, colour and auto-saving are held within the data files, so one file could have review mode and colour enabled whereas another could have review mode disabled and run in monochrome.

from version 1.79
You can also tell iKog to use a specific external data file when it starts up. To do this, just enter the external data file as a command line argument when you start the program. So lets say you are using a file called *.mydata.dat,* you can start the program by entering:

```
ikog.py mydata.dat
```

The *.dat* extension is automatically added to file names entered from the command line.

from version 1.80
On Windows and Linux you can refer to the user' s home folder by using the ~ character. So if you had a file called *mydata.dat* in your home folder, you could start the program by entering:

```
ikog.py ~/mydata.dat
```

(change / to \ for Windows.)

or once running you could use the `OPEN` command:

```
>>>OPEN ~/mydata
```

3.15 Timing two minutes

from version 1.82

In the *Getting Things Done* approach, there is a rule that if a task can be done in 2 minutes, then you should do it now. Two minutes can be longer than you think. If you want to know how long 2 minutes is or actually time yourself, you can use the `2` command. Just enter `2` and the program will start timing. At the end of the period the screen will clear and the program will beep.

If you do not get a beep from a Linux system, your terminal may need to be configured to beep when the *Bel* character is sent; this is how the sound is created in ikog.py.

3.16 Creating shortcuts

from version 1.83

If you have a commands that you use a lot, you can assign them to a shortcut. A shortcut is simply a quick way of entering a command. There are ten shortcuts, 0 to 9. To create a shortcut simply use the `SHORTCUT` or `SC` command followed by the shortcut number and finally the required command. So to set shortcut 3 to execute the command `FILTER @Computer` you would enter:

```
>>>SHORTCUT 3 FILTER @Computer
```

To use the shortcut, just type `=` followed by the shortcut number. So to use our filter shortcut we would just enter:

```
>>>=3
```

To list all the current shortcuts just type `SHORTCUT ?` or `SC ?`.

```
>>>SHORTCUT ?
=0  unused
=1  unused
=2  unused
=3  FILTER @Computer
=4  unused
=5  unused
=6  unused
=7  unused
=8  unused
=9  unused
```

To remove a shortcut just use the shortcut command followed by the number, but with the command blank.

So to remove shortcut 3 just enter:

```
>>>SHORTCUT 3
```

3.17 Automating from the command line

from version 1.86
It is possible to automatically run the program automatically from the command line. This may be useful if you want to add tasks from another script. You can run commands when iKog starts by entering:

```
ikog.py mydata.dat commands
```

Note that you must define the external data file that you are using; in this example *mydata.dat* is being used. If you are using internal data, just use a full stop in place of the file name. e.g.

```
ikog.py . commands
```

The `commands` part of the line is just a list of valid iKog commands separated by ` / `. Note that you must have a space either side of the `/` character. So to apply a filter of @Computer and then list the tasks, we could start the program as follows:

```
ikog.py . FILTER @Computer / LIST
```

The program will run the commands when it starts and will then stay active waiting for user input. If you want to exit automatically, you must add a `QUIT` command as well.

The main use of this feature is to allow the program to be automated via a script. So to add a task from a script we could use:

```
ikog.py . + Buy some bread / QUIT
```

Because we want iKog to exit once it has finished, we added the `QUIT` command at the end.

If you are automating the program in this way, ensure that you do not use any commands that require user input as the program will wait for the input.

Deleting tasks is a little trickier as you need to be able to find the task to delete it. Using task numbers is not appropriate as these can change and also require user confirmation. To delete tasks from the command line, use the `FILTER` command to find some unique text. So to delete a task that uniquely contains the text *henspace* you could run:

```
ikog.py . FILTER henspace / KILL THIS / QUIT
```

The `FILTER` command finds the task, the `KILL THIS` command then deletes the current task and finally `QUIT` exits.

Because you need a unique reference to delete tasks automatically it is a good idea to give each task a unique reference, e.g *ref_1234* when you add it.

Do not use the caret symbol, e.g. `KILL ^`, as the symbol has a special interpretation on some systems. Always use `THIS` instead.

3.18 Customising ikog

from version 1.90
If you want to modify some of the outputs without editing the main program file you can do this by using a plugin file called *ikogPlugin.py*. This requires knowledge of Python so if you don't want to do any programming, skip this topic and don't download the file.

The ikogPlugin provides a mechanism for modifying the behaviour of iKog. If the plugin file is created it must contain every function. A zip file contain an example file is contained in the zip file below.

To download the example file visit http://www.henspace.co.uk/ikog/index.html

Extract the ikogPlugin.py and then edit the contents to manipulate the output from iKog. Documentation is contained in the file. The supplied file is not of any practical use in its own right but merely adds tags, which although disrupting the output, make it relatively easy to see where the methods are called and what their effect is. To make any practical use of the methods you will need to get out your regular expressions books and parse the strings - sorry :)

IV

Reference

4 Reference

4.1 Context abbreviations

Abbreviation	Context
@A	@Anywhere
@C	@Computer
@D	@Desk
@E	@Errand
@H	@Home
@I	@Internet
@L	@Lunch
@M	@Meeting
@N	@Next
@O	@Other
@P	@Phone
@Pw	@Password
@S	@Someday/maybe
@W4	@Waiting_for
@W	@Work

4.2 Commands

Many commands have abbreviated forms. The different forms are shown separated by the / character. So, for example, the command HELP/H means that you can enter either HELP or H.

All commands can be entered in any case, so HELP is the same as help or Help.

Some commands have alternative options. These are shown separated by the | character. So, for example, ON|OFF means that you should enter ON **or** OFF.

Some parts of the commands are optional. Optional parts are shown in square [] brackets.

4.2.1 General commands

?	Displays a quick reference card.
HELP/H	Displays more help.
VERSION/VER	Display the version of ikog.py.
COLOR/COLOUR/C	Use colour. (Linux/Unix only)
COLOR/COLOUR/C N	from version 1.87: Use colour set N. (Linux/Unix only).
MONOCHROME/ MONO	Use a monochrome display.
EXPORT	Export the tasks to the file ikog.py.tasks.txt (If you renamed ikog.py, the export filename will also change)

IMPORT filename	Import the tasks contained in the file. The file can be another copy of ikog.py or a previously exported file.
REVIEW/REV ON|OFF	Switch review mode on or off. When on, pressing enter advances to the next task. When off, pressing enter redisplays the current task.
V0	Same as REVIEW OFF.
V1	Same as REVIEW ON.
WEB	Visit the Henspace website.
SAVE/S	Save the current file.
AUTOSAVE/AS ON|OFF	Switch autosave on or off. When on, the file is automatically saved whenever a change takes place. When off, the file is saved when you use the SAVE or QUIT commands.
CLEARSCREEN/CLS	Clear the screen.
SYS ON|OFF	Allow system calls when clearing the screen and enable the !CMD command
!CMD command	Run a system command
2	Start a two minute timer
SHORTCUT/SC N cmd	Set shortcut N to command cmd
SHORTCUT ?	List shortcuts
ABBREV/AB @x @txt	Create a custom context abbreviation to expand @x to @txt
ABBREV/AB ?	List context abbreviations
PAB :px :pTitle	Create a custom project abbreviation to expand :px to :pTitle
PAB =	List project abbreviations
OPEN/O filename	Open an external data file
NEW filename	Create a new external data file

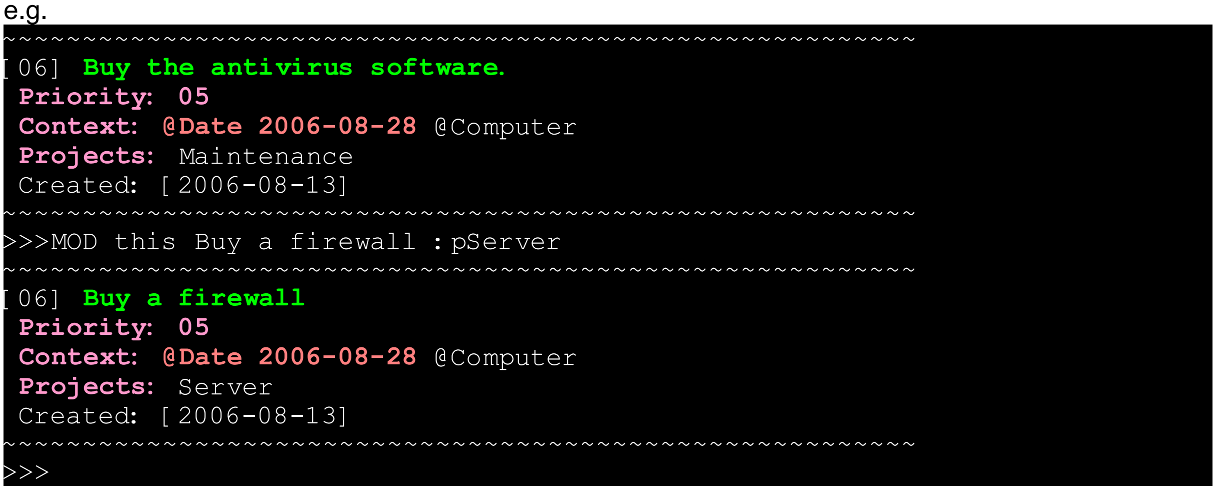
4.2.2 Editing commands

ADD/A/+ text	Add a task to the bottom of the list.
NOTE text	Add a task with a priority of #0 and a context of @Notes
IMMEDIATE/I/++ text	Add a task to the top of the list. Priorities will still be honored.
KILL/K/X/- N	Delete task N.
ARCHIVE/DONE N text	from version 1.86: Archive task N. Sets the task's date to today, gives it an @Archived context and moves it to an archive file.
CLEAR	Remove all tasks.
REP/R N [text]	Replace task N with a new entry. If you do not enter the new entry in the text, you will be prompted to enter the new task details.
MOD/M N [text]	Modify task N, changing those elements entered in the text. If you do not enter the new details in the text, you will be prompted to enter the new task details.
EXTEND/E N [text]	Extend task N, appending those elements entered in the text. If you do not enter the new details in the text, you will be prompted to enter the new task details.
SUB/SU N /s1/s2/	from version 1.77: Replace text s1 with s2 in task N.

EDIT/ED [N]	from version 1.77: Use an external editor to edit task N. If you omit N, then use an external editor to create a new task.
FIRST/F N	Make task N the first in the list. Priorities will still be honored.
DOWN/D N	Move the task down one position in the list.
UP/U N	Move the task up one position in the list.
SETEDNT editor	from version 1.77:Use the external program *editor* on Windows systems for the EDIT command.
SETEDPOSIX editor	from version 1.77:Use the external program *editor* on Linux systems for the EDIT command.

---

from version 1.57
For editing tasks that require the task number, N, to be entered, you can use `^` or `this` to refer to the current task.
e.g.



```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[06] Buy the antivirus software.
 Priority: 05
 Context: @Date 2006-08-28 @Computer
 Projects: Maintenance
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>MOD this Buy a firewall :pServer
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
[06] Buy a firewall
 Priority: 05
 Context: @Date 2006-08-28 @Computer
 Projects: Server
 Created: [2006-08-13]
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
>>>
```

---

from version 1.62
You can also encrypt tasks by embedding the `<private>` or `<secret>` tags in the task text. See Encrypting your data.

4.2.3 Navigation and view commands

FILTER/FI [filter]	Set a global filter. If the filter is not entered, then the current filter is removed.
TOP/T	Jump to the top task.
TOP/T N	from version 1.59: Jump to the top task and list N commands.
NEXT/N	Display the next task in the list.
PREV/P	Display the previous task in the list.
GO/G N	Display task N
LIST/L [filter]	List all tasks. If the filter is entered, then this is used to restrict the output.
LIST>/L> [filter]	Output all tasks to an HTML report. If the filter is entered, then this is used to restrict the output.
@	List all tasks grouped by context.

@>	Output all tasks, grouped by context, to an HTML report.
:D	List all tasks that have a date and grouped by their dates.
:D>	Output all tasks that have a date, grouped by their dates, to an HTML report.
:P	List all tasks grouped by their projects.
:P>	Output all tasks, grouped by their projects, to an HTML report.

4.3 Release history

Version	Date	Comments
1.90	2008-11-14	🐞 Fix problem with **as** now being a reserved word in Python 2.6
		⚙ Add support for simple (sort of) user modifications via a plugin file.
1.89	2007-12-01	🐞 Fix problem with invalid dates causing program to exit.
1.88	2007-11-23	🐞 Add charset UTF-8 to meta statement in HTML exports
1.87	2007-05-06	⚙ Add additional colour sets.
		⚙ Add import for readline module if available.
1.86	2007-03-25	⚙ Add CSS styles to html reports
		⚙ Add ARCHIVE/DONE commands
		⚙ Add automation via the command line
		🐞 Correct launching of html reports when the path contains spaces.
1.85	2006-12-01	⚙ Add project abbreviations
		🐞 Correct deletion of custom context abbreviations
1.84	2006-11-08	⚙ Add OR option for filters
		⚙ Add custom context abbreviations
		🐞 Correct file permissions on saved file.
1.83	2006-10-16	⚙ Add shortcuts
		✏ Print current task in compressed format after a list command.
1.82	2006-10-04	⚙ Add the @Meeting abbreviation
		⚙ Add the two minute timer
1.81	2006-10-01	✏ Add the .dat extension to filenames passed via the command line.
		✏ Set Autosave and Review mode on as the default for new files.
1.80	2006-10-01	⚙ Expand user home when entering external data filenames.
1.79	2006-09-30	⚙ Allow an external file name to be passed in from the command line.
		🐞 Add source encoding so that Unicode characters can be supported when using internal tasks.
1.78	2006-09-30	🐞 Allow the external editor path to include spaces.
1.77	2006-09-29	⚙ Add the sub command
		⚙ Add external editor option
1.76	2006-09-19	⚙ Add check on the !cmd so that deletion commands cannot be run
1.75	2006-09-19	⚙ Add the !cmd command

1.74	2006-09-17	⊖	Prevent Ctrl-C from exiting the program at input prompts.
1.73	2006-09-15	⊕	Automatically set :dN date format to the future
1.72	2006-09-14	⊕	Add confirmation for task deletion unless ^ or this are used.
1.71	2006-09-13	⊖	Add use of system calls if allowed clear to all occurrences of screen clears.
1.70	2006-09-12	⊖	Save cfg to external files
		⊕	Add the sys command
1.69	2006-09-12	⊕	Add ability to create and open files with the new and open commands
1.68	2006-09-05	✏	Use ___ instead of ... in lists to improve appearance.
		⊖	Remove extra space that appeared when lines were displayed.
1.67	2006-09-04	⊕	Allow formatting with
.
		⊖	Prevent entry of special marker characters.
1.66	2006-09-03	⊕	Allow the use of separate data file.
1.65	2006-09-01	⊖	Just correct some typos in the messages.
1.64	2006-08-31	✏	No longer use system call to clear the screen. Security suites like ZoneAlarm used to warn about the system call.
		⊕	Clear the screen after adding or modifying a task if a password was used.
		⊕	Add the clearscreen command.
1.63	2006-08-31	⊕	Add sh abbreviation for the show command
		⊕	Add the note command
		⊕	Clear the screen after displaying encrypted data.
		⊕	Add @Pw abbreviation.
1.62	2006-08-30	⊕	Add encryption.
		⊕	Add the show command.
1.61	2006-08-25	⊖	Modify the top command so that when the list option is used, the filter is also displayed.
1.60	2006-08-25	⊕	Immediate command modified so that it also sets the @Date context to today. This makes the task truly immediate.
		⊖	Correct the top command so that the correct number of lines is shown when filtered.
1.59	2006-08-24	⊕	Add the - filter.
		⊕	Add the :d+X date format.
		⊕	Add a list parameter to the top command.
		✏	Increase the number of lines displayed when printing help.
1.58	2006-08-22	⊕	Add the ability to search for text using the filter command.
		⊕	Add the version command.
		⊖	If the filter prevents any task being displayed, remove the filter — list commands excepted. Previous versions would show the current task even if the filter should have prevented it.
1.57	2006-08-21	⊕	Allow use of 'this' and '^' to refer to the current task in edit commands.
		✏	Always show the current task after the list command.

		Save, or mark as requiring a change, after an import has finished.
		Improve help to prevent skip message appearing twice.
1.55	2006-08-19	Improved detection of Python version
1.54	2006-08-17	First release

Index