

# **Sitemask User's Guide**



**by Jason Sando**

**Version 07 July 2002**

## Table of Contents

<b>1</b>	<b>INSTALLATION .....</b>	<b>1</b>
<b>2</b>	<b>USING SITEMASK.....</b>	<b>2</b>
2.1	QUICK START .....	2
<b>3</b>	<b>REFERENCE .....</b>	<b>5</b>
3.1	PROGRAM OPTIONS .....	5
3.2	THEME ELEMENTS .....	6
3.2.1	<i>Site Layout File</i> .....	6
3.2.2	<i>Theme folder</i> .....	7
3.2.3	<i>Theme function</i> .....	7
3.3	OBJECT MODEL .....	8
3.3.1	Class " <i>Page</i> " .....	8
3.3.2	Class " <i>Node</i> ".....	9
3.3.3	Class " <i>Site</i> ".....	10

# 1 Installation

To install Sitemask you must do the following:

## **Step 1: Download and Unzip**

The Sitemask project page should have a link to download the current Sitemask distribution . Follow that link and download the ZIP or TGZ file to a temporary directory. Use WinZip or some similar tool to extract the archive. You'll need to add the "sitemask.py" folder to your PATH, so you may want to extract it to either "c:\\" or "c:\program files", wherever you tend to keep things.

## **Step 2: Update your PATH Environment Variable**

Once the archive is extracted, you should have a new folder named **sitemask** where you extracted it. Add this to your PATH environment variable. In Windows this means you go to Properties on My Computer, select "Advanced", then "Environment Variables". Select the PATH (under either System or User) variable and append ";" followed by the new sitemask folder. For example, if you extracted the archive to "c:\", you should add ";c:\sitemask" to the path.

## **Step 3: Close and Re-Open any Command Prompt windows**

If you have any open Command Prompt windows you'll need to close and re-open them for this change to take effect.

## **Step 4: Install Python (if necessary)**

If you have never installed Python before you must do this. Download and install the latest Python release (2.2.1 as of this writing) from:

<http://www.python.org/download/>

## **You're Ready!**

You can now run Sitemask from a command prompt by typing "sitemask.py". Try "sitemask.py -?" to get help, and be sure to read the following section before using Sitemask for the very first time.

## 2 Using Sitemask

To run the Sitemask program you must open a Command Prompt window. If unsure how to do this, on Windows you can try "Start -> Programs -> Accessories -> Command Prompt". The icon usually is black, with the text "C:\\" on it in white. If you cannot find the program, use "Start -> Run ...", then type "cmd" if you use Windows NT/2000/XP, or "command" if you use Windows 95/98/ME.

Once you have a Command Prompt window open, try running the program by typing:

```
sitemask.py -?
```

You should see a several screens full of help information from the Sitemask program.

If instead you see an error message, like

```
'sitemask.py' is not recognized ...
```

then your PATH is not set properly. Review the installation instructions above to correct it.

### 2.1 Quick Start

You must add some folders to your website before you can start adding themes to it. To start, just copy the "sitemask\themes" folder into your website. If your website is on your local hard drive under "c:\samples\mysite", then you should end up with "c:\samples\mysite\themes" (see **Image 1**).

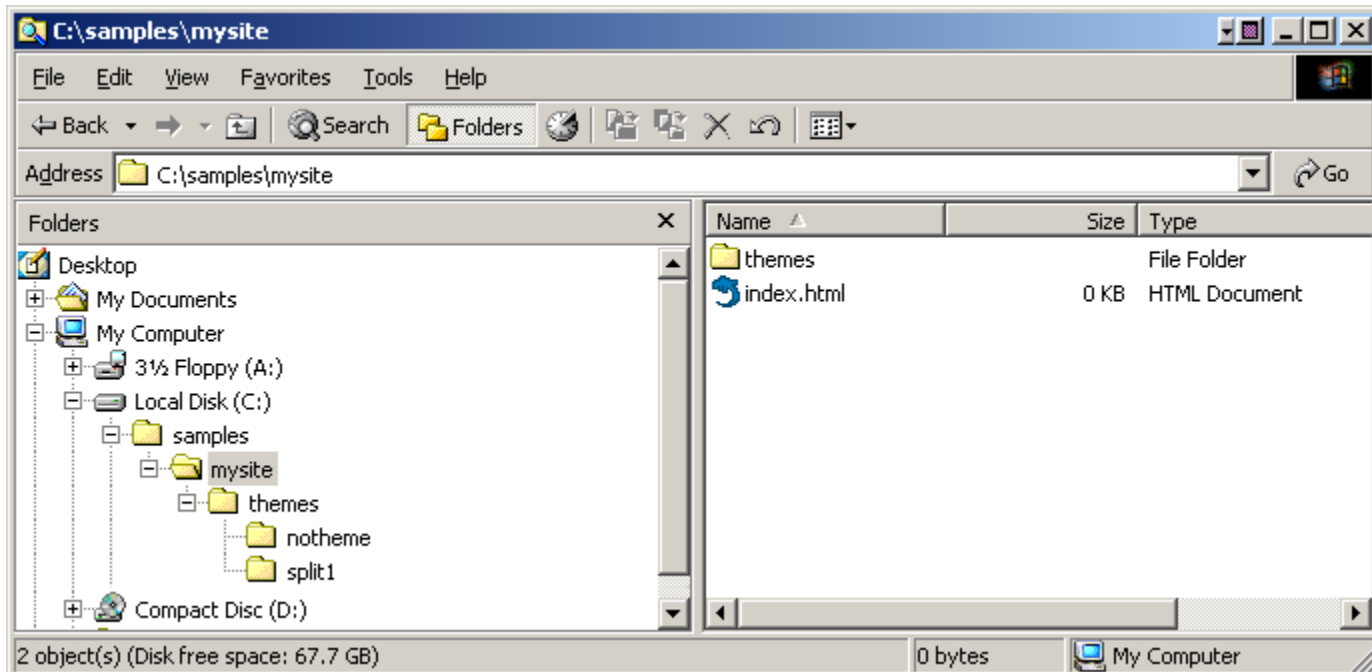
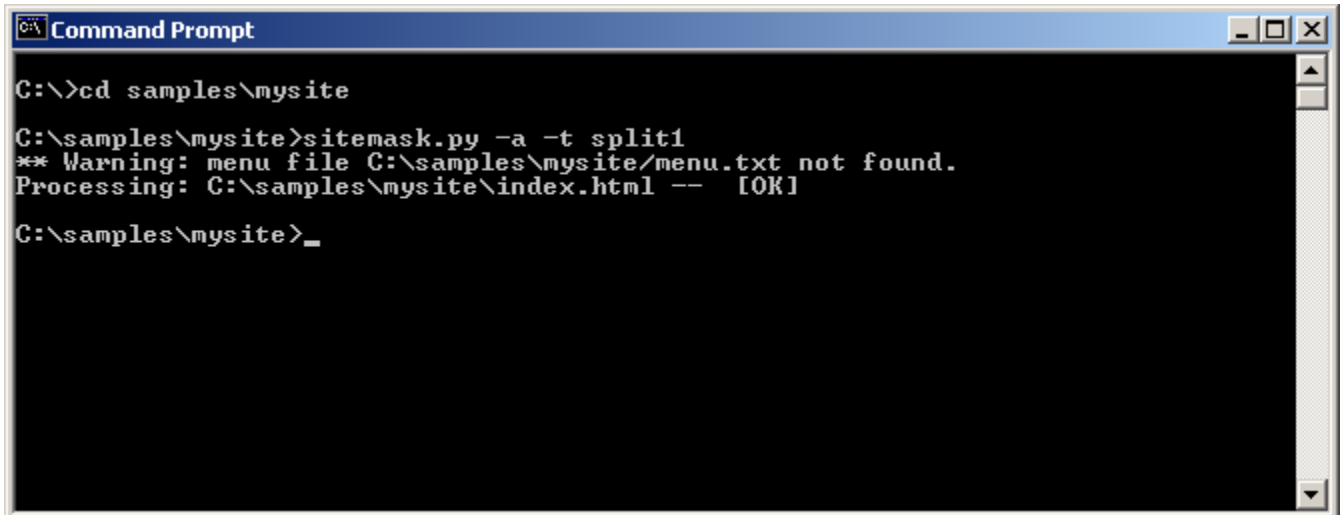


Image 1: A sample website, showing the themes folder.

Now you can run Sitemask against your website, although you will want to create a customized theme. We'll get back to that in a moment. Let's try this out first!

To run Sitemask, open a command prompt (as described above in "Using Sitemask"), and change directory to your web site root folder. In our example this is "c:\samples\mysite".

Suppose we want to apply the Sitemask test theme "split1" to our website, just to see how it looks. We would type the following (see **Image 2**):



```
Command Prompt
C:\>cd samples\mysite
C:\samples\mysite>sitemask.py -a -t split1
** Warning: menu file C:\samples\mysite/menu.txt not found.
Processing: C:\samples\mysite\index.html -- [OK]
C:\samples\mysite>_
```

Image 2: Running Sitemask, using a test theme.

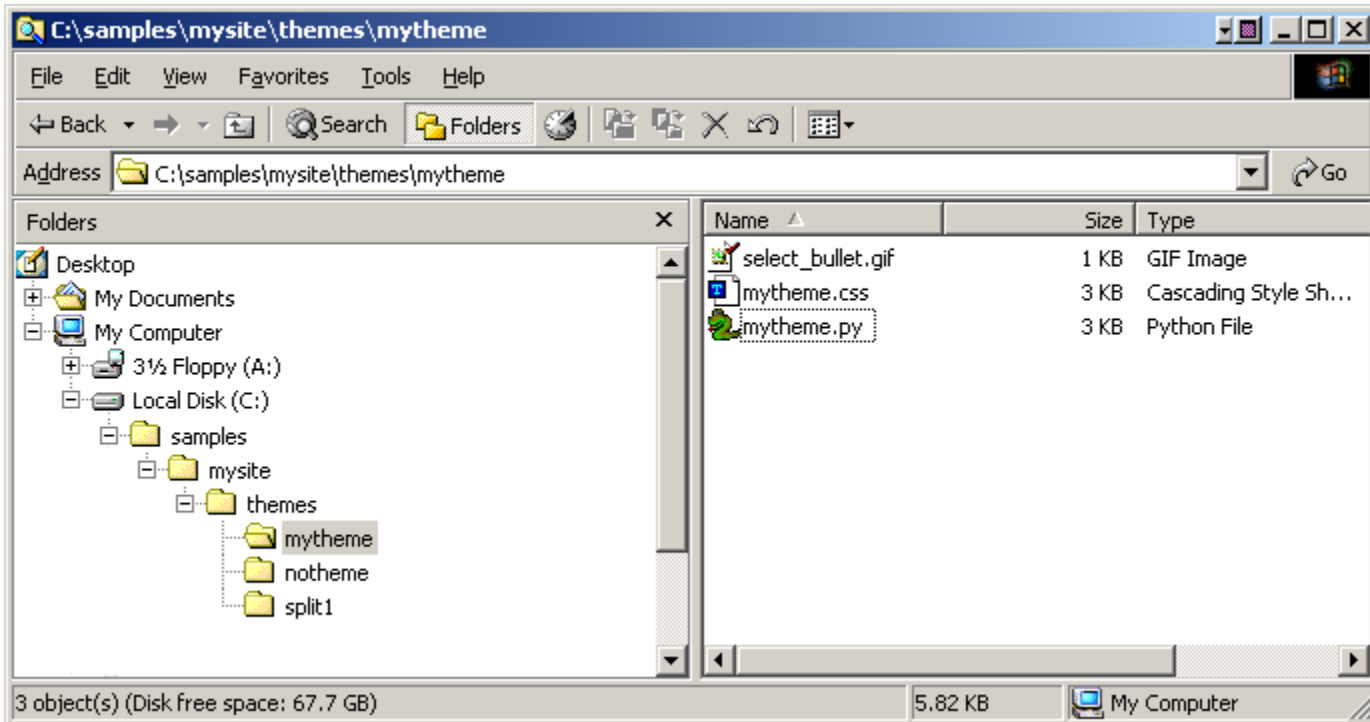
The image shows that we first changed directory, then ran Sitemask. We told it to *add the theme* (the "-a" option) named *split1* (the "-t split1" option). And because we didn't tell it which files to use, or what directory to look in, it just found all the HTML files in every directory beneath the current directory (there was only 1 - named *index.html*).

If you look at your website now, you'll find it looks good, but its not exactly what you want. For one, the header and footer on every page probably says "Sitemask Test". What you need to do is create your own customized theme. You can try all the test themes to see if one is close to what you want. If you find one where the layout is roughly what you want, go ahead and copy it. The colors and fonts are generally in a single stylesheet, so those are quite easy to modify.

To create your own theme you must create the theme folder, under "themes". Start by copying one of the test themes — the entire folder — and give it a new name. We'll copy "split1" to "mytheme", by first creating the new folder "themes\mytheme", and then copying all the files from "split1" into it.

The main file in each theme is the Python script that implements it. This file must be named the same as the theme, and it must end with ".py". So now, to follow in our example, the file "mytheme/split1.py" must be renamed to "mytheme/mytheme.py".

**Image 3** shows the new theme, with all files having been renamed.



**Image 3: Copy an existing theme's files to a new theme folder to make your own.**

If there are other support files with the same name as the theme, go ahead and rename them all. For example, if there is a "mytheme/split1.css", rename it also to "mytheme/mytheme.css". It is important to keep your custom themes separate from Sitemask's test themes since those will be updated with every new release of Sitemask — if you customize them, your changes will be overwritten. So always create your own custom themes.

After all the files are renamed, the final step is to customize the new theme to generate the proper header, images, and footer for your site. This is done by editing the Python script for the theme. If you don't understand Python, there is no need to be intimidated. Just make sure each line begins with a Tab character, and modify what you find in the existing theme. It should be easy to spot where the header text, images, and so forth are being generated, and a simple matter to modify the file. Once you modify it you can immediately run Sitemask again — there is no compile step.

There may be a ".pyc" file — this is the *compiled* version of the theme Python script, and is created automatically when the theme is applied. You can delete this file.

For further information on customizing a theme see the reference section.

## 3 Reference

### 3.1 Program Options

The program options are detailed in the help message (run *sitemask.py -?*) as well as shown here.

```
SiteMask -- Website Theme and Navbar Automation
(c)2002 Jason Sando

Options:

[-d basepath] The base folder of the website (defaults to ".")
[-t name]     Theme name - must be folder beneath basepath/themepath. If
              not specified, it uses the theme referred to by each file.
[-a]         Add theme to processed files.
[-r themepath] Theme root path (defaults to "basepath/themes/").
[-m menupath] Menu definition file (defaults to "basepath/menu.txt").
[files]     One or more files to process. If not specified it
              processes all HTML files recursively (the entire site).
[-?]       Print this message.

Defaults:

Note that all the settings above have defaults. The program assumes that
themes are installed, one subdirectory for each theme, beneath the root of
your website in a folder called "themes".

Each theme folder contains the resources needed for that theme, such as
icons, script files, and the Python template function that generates
the pages.

For example, assume you use two themes "chrome" and "jellybean" in
your website. Under the site root folder you will have the
following directories:

    ./themes/chrome/
    ./themes/jellybean/

If you use the theme menuing features, you will also have the menu
configuration file in the site root folder:

    ./menu.txt

Examples:

UPDATE ENTIRE SITE
-----
To update all web pages in the current site, and assuming your command
prompt is in the base directory of the website, just type:
    sitemask.py
and the entire website will be updated.

ADD THEME TO PLAIN FILE
-----
To add the theme "java" to the file "projects/index.html", type:
    sitemask.py -t java -a projects/index.html

REPLACE THEME ON ENTIRE SITE
-----
To change the theme for every page within the web site to "java",
type:
    sitemask.py -t java -a
```

## 3.2 Theme Elements

This section describes the different parts of Sitemask in detail.

### 3.2.1 Site Layout File

The *Site Layout File* is commonly named “menu.txt” and found in the site's root directory. This optional file describes the menu structure for the site. The menus can be hierarchical, i.e. each menu item can have other “submenus” beneath it.

The name and location of this file can be overridden using the “-m menupath” command option.

The file format is very simple – just indented lines containing a *caption*, an optional *link URL*, and an optional *icon URL*. Consider the following example (please note: leading whitespace is tab characters):

```
#
# SiteMask test web site.
#
Home: /
Products: /products/
  Widget 1: ./products/widget1.gif
    Features: /products/features1
    Pricing: /products/pricing1
    Download: /downloads/
  Widget 2
    Features: /products/features2
    Pricing: /products/pricing2
    Download: /downloads/
Downloads: /downloads/
```

#### 3.2.1.1 Comments

Any line whose first non-whitespace character is a pound sign “#” is considered a comment and the entire line is ignored.

Note that *only* most script languages, comments are *not* allowed at the end of a valid line, *only* on a line by themselves.

#### 3.2.1.2 Empty lines

Any empty lines or lines consisting entirely of whitespace will be ignored.

#### 3.2.1.3 Tabs

The hierarchy of the menus is determined by the number of leading tabs before a non-empty line. In the above example, “Products” has no leading tabs, “Widget 1” has one leading tab, and “Widget 1/Features” has two leading tabs.

A tab on an empty line is ignored.

#### 3.2.1.4 Line format

Each non-empty, non-comment line has from one to three data elements.

1. Menu caption (required) – the user-readable caption that will be displayed.

2. Link URL (optional) – the link to follow when the user selects the menu.
3. Icon URL (optional) – the link to an icon to be displayed in the submenu.

If items 2 or 3 are omitted, the intervening “:” (colon) characters may also be omitted. So for example an item that has no link or icon needs no colons, and an item with no icon URL needs no trailing colon after its link URL.

### 3.2.1.5 How Pages are Matched to Menu Items

The method used by Sitemask to process the files in a site is to search the site's root folder and all child folders, except the “themes” folder, for files that end with “.html”. As each file is processed, Sitemask attempts to identify it objectively as a member of the menu structure. It does so by converting the path to an absolute path, and also converting the menu link URL to an absolute path. The absolute paths are in terms of the local filesystem, so on Windows it would look like “c:\projects\mysite\samples\index.html”.

Menu links can be written without the trailing “.html”, as Sitemask will check the existence of the link URL when it reads the layout file. If no file is found it attempts to add “.html” and checks again. Directory links can also be used in this way – if the link is to a directory, Sitemask looks for “index.html” in that directory, and uses that as the match criteria when processing pages (ie, the page folder/index.html will match the link URL “/folder/”).

Although Sitemask can accommodate these variations, the user must ensure that their target web server behaves similarly.

## 3.2.2 Theme folder

Themes are defined by a Python module and resource files, each residing with a *theme folder*. The theme folders are commonly located in a folder called “themes”, residing beneath the web site's root folder. The name and location can be changed using the “-r themepath” command option. The “themes” folder contains only other folders, each representing a single theme.

Each theme folder must contain a Python module of the same name. E.g., a theme named “chrometabs” is defined by `siteroot/themes/chrometabs/chrometabs.py`. Both the folder and Python module *must have the same name*.

Each theme folder can contain any number of support files in addition to the Python module.

## 3.2.3 Theme function

Each theme is implemented by a Python module of the same name. The module defines a function that is invoked once for each page in the site, thus given the function a chance to modify each page if desired.

The Python module for each theme must define a function as follows:

```
def applyTheme (page, file):  
    file.write (...)
```

The module can of course invoke other modules, and can declare other functions and variables, so long as it properly defines *applyTheme*.

The arguments to *applyTheme* are the *Page* object, and a file to write to. The file is the html file being written (the original file). The *Page* class is detailed below under **Object Model**.

### 3.3 Object Model

The primary classes available to theme developers are *Page*, *Node*, and *Site*. The *Page* object representing the current HTML page is what is passed to the *applyTheme* function. This in turn has a reference to the *Site* object representing the site to which the page belongs. And, optionally, the *Page* might reference a menu *Node* object if Sitemask matched the page's URL against something in the site layout file.

#### 3.3.1 Class "Page"

The *Page* class is used to represent each page in the site as it is processed.

string	<b>bodyHtml</b> The original HTML, minus any old theme code (so really the original HTML), that appeared <i>between</i> the <body> and </body> tags. The theme function typically writes this into a section of the newly themed file, such as in a centered table cell or div.
string	<b>bodyTag</b> The original <body ...> tag, including any attributes. It is important to write out this tag instead of just writing "<body>", since it may have script events, style, color, etc.
string	<b>bottomHtml</b> The HTML appearing after the </body> tag, up to the end of the document. While a theme function could simply write the end tags, there may be original comments in them so use this string instead.
string	<b>headHtml</b> Any HTML appearing in the <head> </head> section.
Node	<b>node</b> The menu <i>Node</i> this page was matched to, i.e. this page represents a menu selection as defined by the referenced <i>Node</i> . Otherwise <i>None</i> .
string	<b>originalHtml</b> The original HTML as read from the file, prior to any old theme code being removed.
string	<b>pageAbsPath</b> The absolute path (in the local filesystem) to the page.
string	<b>pageAbsUrl</b> The absolute URL, i.e. relative to the site root, to the page.
Site	<b>site</b> The <i>Site</i> object to which this page belongs.
function	<b>themeFunction</b> If the page had a prior theme, this is the function. Otherwise <i>None</i> .

string	<b>themeName</b> If the page had a prior theme, this is the name. Otherwise <i>None</i> .
string	<b>title</b> The text <i>between</i> the <title> and </title> tags.
string	<b>topHtml</b> Any HTML preceeding the <head> tag, usually a DOCTYPE and the <html> tag but possibly including comments.

### 3.3.2 Class "Node"

Each menu item is represented as a *Node*, with the entire menu tree belonging to a *root node*. This root node contains children representing the top level menu items.

If a site has no site layout file then it has no menus, and no root node.

Node[]	<b>children</b> The list of child nodes ("submenus") beneath this node, possibly empty but not <i>None</i> .
void	<b>dump (string prefix)</b> Write the node hierarchy beneath this node to stdout using <i>prefix</i> for indentation.
string	<b>getIconPath ()</b> Return the nearest non- <i>None</i> iconPath, from this Node or its nearest parent having a non- <i>None</i> iconPath. If no nodes in the hierarchy above this node has an iconPath then <i>None</i> is returned.
string	<b>iconPath</b> <i>None</i> , or a URL for an icon to display for this Node.
boolean	<b>isParentOf (Node child)</b> Return true (non-zero) if this node is the direct or indirect parent of the given child node.
int	<b>level</b> Return the level of this node, where zero is the top level. Useful when splitting navbars, e.g. a sidebar is used for secondary navigation.
string	<b>link</b> <i>None</i> , or a URL to use as a hyperlink when this menu item is activated.
Node[]	<b>nodesAtLevel (int level)</b> Return the list of nodes at the given level from this node. E.g., if this node is at level 1, and the list of nodes at level 2 is desired, this returns the list of child nodes beneath this node. If at level 2 and level 1 is desired, then the level 1's above this node in the hierarchy are returned. Returns <i>None</i> if the level is inapplicable.
Node	<b>parent</b> Returns the parent node or <i>None</i> if this is the root node.
string	<b>path</b> Returns the absolute path in the local filesystem for the target link.
string	<b>title</b> Returns the title text (aka "caption") for displaying to the user.

### 3.3.3 Class "Site"

The *Site* class holds fields and methods relating to the site as a whole. The same Site object is references by all pages of the site.

boolean	<b>addTheme</b> True (non-zero) if user indicated desire to add theme to un-themed files, else false (zero).
string	<b>basePath</b> The base path of the site, possibly relative (e.g., it could be ".").
function	<b>defaultTheme</b> The default theme function or <i>None</i> .
string	<b>defaultThemeName</b> The name of the theme to use if adding a theme to an unthemed file, or if overriding the existing theme. Possibly <i>None</i> if user did not specify a theme on the command line.
Node	<b>getPageNode (String pageAbsPath)</b> Returns the menu <i>Node</i> corresponding to the given absolute path. Note that the path is filesystem absolute, not site absolute (i.e., "/usr/sites/mysite/index.html" instead of "/index.html"). Returns <i>None</i> if the page cannot be mapped to a menu node.
function	<b>getTheme (string themeName)</b> Return the function for the named theme. If not found and exception is raised (from failing to import the module).
Node	<b>menuRoot</b> The root node of the menus, i.e. the node whose children represent the top level menus. <i>None</i> if there are no menus (no site layout file found).
string	<b>themePath</b> The folder in which to find themes, possibly relative.

This document was created with Win2PDF available at <http://www.daneprairie.com>.  
The unregistered version of Win2PDF is for evaluation or non-commercial use only.