```
In [1]: from authorai import blogger
```

```
In [2]: print(blogger.last_update())
```

```
add code snippets
```

```
In [3]: post = blogger.BlogPost()
        authorai_site = blogger.JekyllSite()
        authorai_site.set_site_folder('../authorai-site')
        post.set_title('AuthorAI Blogger for Jekyll Static Site Generator Automatior
```

```
In [4]: question = 'What are the well know blogs or static websites built using Jeky
        post.set_qna(question, blogger.qna(question))
```

```
In [5]: question = 'How popular is Jekyll static site generator?'
        post.set_qna(question, blogger.qna(question))
```

```
In [6]: question = 'Which companies use Jekyll static site generator?'
        post.set_qna(question, blogger.qna(question))
```

```
In [7]: post.get_qna()
```

```
Out[7]: {'What are the well know blogs or static websites built using Jekyll?': 'So
        me well-known blogs and static websites built using Jekyll include GitHub P
        ages, The Jekyll Theme Showcase, Smashing Magazine, and The Changelog.',
         'How popular is Jekyll static site generator?': 'Jekyll is a very popular
        static site generator. According to BuiltWith, Jekyll is used on over 1.2 m
        illion websites.',
         'Which companies use Jekyll static site generator?': 'Some of the companie
        s that use Jekyll static site generator include GitHub, Shopify, Airbnb, Ne
        tflix, and Microsoft.'}
```

```
In [8]: concept = 'Writing high quality blog posts is time consuming and hard work'
        topic = blogger.describe_concept(concept, 150, grade=10)
        post.set_topic(topic)
        print(post.get_topic())
```

```
Writing high quality blog posts is a difficult and time consuming task. It
requires a great deal of effort and dedication to create content that is bo
th interesting and informative. It is not enough to simply write a few sent
ences and post them online; the content must be well-researched, well-writt
en, and engaging. It also requires a good understanding of the topic, as we
ll as the ability to craft a compelling narrative. Additionally, the blog p
ost must be optimized for search engines, which requires an understanding o
f SEO principles. Finally, the post must be edited and proofread to ensure
accuracy and clarity. All of these steps take time and effort, but the end
result is worth it.
```

```
In [9]: response, response_list = blogger.tags_from_topic(post.get_title() + ' - ' +
        post.set_tags(response_list)
        print(post.get_tags())
```

```
['Topic', 'Blog post', 'Writing', 'Research', 'SEO']
```

```
In [10]: title = 'Static website generators gaining popularity'
         section = blogger.describe_concept(title, 400, grade=10)
         post.set_sections(title, section)
         print(post.get_section(title))
```

```
Static website generators are gaining popularity as a way to quickly and ea
sily create websites. These generators allow users to create websites witho
ut having to write any code. Instead, they use a template-based system to c
reate a website that looks professional and is easy to maintain.

Static website generators are becoming increasingly popular due to their ea
se of use and the fact that they are more secure than traditional websites.
With a static website, all of the content is stored on the server, meaning
that it is not vulnerable to malicious attacks. This makes it much more sec
ure than a traditional website, which can be vulnerable to hackers.

Static website generators also offer a number of advantages over traditiona
l websites. For example, they are much faster to create and maintain. This
is because they do not require any coding or programming knowledge. Instea
d, users can simply select a template and fill in the content. This makes i
t much easier for users to create a website quickly and easily.

In addition, static website generators are much more cost-effective than tr
aditional websites. This is because they do not require any hosting fees or
other costs associated with traditional websites. This makes them an attrac
tive option for those who are looking to create a website on a budget.

Finally, static website generators are becoming increasingly popular due to
their ability to create websites that are optimized for search engines. Thi
s means that they can help to improve the visibility of a website in search
engine results. This can be beneficial for businesses, as it can help to in
crease their online presence and attract more customers.

Overall, static website generators are becoming increasingly popular due to
their ease of use, security, cost-effectiveness, and ability to create webs
ites that are optimized for search engines. This makes them an attractive o
ption for those who are looking to create a website quickly and easily.
```

```
In [11]: title = 'Jekyll static website generator features'
         section = blogger.describe_concept(title, 300, grade=10)
         post.set_sections(title, section)
         print(post.get_section(title))
```

```
Jekyll is a static website generator that allows users to quickly and easil
y create websites without the need for complex coding. It is a powerful too
l that can be used to create websites with a variety of features.

Jekyll is built on the Ruby programming language and is designed to be simp
le and straightforward to use. It is a command line tool that can be used t
o generate HTML files from Markdown files. This allows users to quickly and
easily create websites without having to write any code.

Jekyll also has a number of features that make it a great choice for creati
ng websites. It has built-in support for Sass and CoffeeScript, which allow
users to quickly and easily create dynamic websites. It also has support fo
r Liquid templating, which allows users to create dynamic content without h
aving to write any code.

Jekyll also has a number of other features that make it a great choice for
creating websites. It has built-in support for RSS feeds, which allows user
s to easily syndicate their content. It also has support for pagination, wh
ich allows users to easily create multiple pages for their website.

Finally, Jekyll has a number of plugins that can be used to extend its func
tionality. These plugins allow users to add additional features to their we
bsites, such as search functionality, social media integration, and more.
```

In [12]:
```
title = 'Liquid template language for Jekyll'
section = blogger.describe_concept(title, 150, grade=10)
post.set_sections(title, section)
print(post.get_section(title))
```

```
Liquid template language for Jekyll is a powerful and versatile tool for cr
eating dynamic websites. It is a templating language that allows developers
to create custom HTML and CSS code for their websites. It is designed to be
easy to use and understand, and is used in conjunction with the popular Jek
yll static site generator. Liquid is written in Ruby and is based on the Li
quid Markup language. It allows developers to create custom HTML and CSS co
de for their websites, as well as to create dynamic content such as blog po
sts, product pages, and more. Liquid also allows developers to create custo
m tags and filters, which can be used to manipulate data and create custom
content.
```

In [13]:
```
language='HTML page with liquid template language'
usecase = 'todo list with add, delete, and complete features'
caption, snippet = blogger.generate_code(language=language, usecase=usecase,
post.set_snippets(caption, snippet)
print(post.get_snippet(caption))
```

```
<html>
<head>
  <title>Todo List with Liquid Template Language</title>
</head>
<body>
  {% assign todos = “Buy groceries, take out the trash, clean the kitchen, 
mow the lawn” | split: ‘,’ %}

  <div>
    <ul>
      {% for todo in todos %}
      <li>{{ todo }}<a onclick="completeTodo({{ forloop.index }})">Complete
</a><a onclick="deleteTodo({{ forloop.index }})">Delete</a></li>
      {% endfor %}
    </ul>

    <button onclick="addTodo()">Add Todo</button>
  </div>

  <script>
  let todos = {% for todo in todos %}"{{todo}}"{% unless forloop.last  %},
{% endunless %}{% endfor %};

  const deleteTodo = (index) => {
      todos.splice(index, 1);
      console.log(todos);
  }

  const completeTodo = (index) => {
      let todo = todos[index];
      todos[index] = `<strike>${todo}</strike>`;
      console.log(todos);
  }

  const addTodo = () => {
      let newTodo = prompt('Enter a new todo');
      todos.push(newTodo);
      console.log(todos);
  }
  </script>
</body>
</html>
```

In [14]:
```
from IPython.display import Image
blueprint_feature_description = 'Website layout design hand-drawn with multi
'translucent wall-to-wall glass with background of night city skyline'
blueprint_feature = blogger.image_from_description(blueprint_feature_descrip
```

In [15]:
```
Image(url=blueprint_feature)
```

Out[15]:

In [16]:
```
image_folder_path = authorai_site.get_site_folder() + '/' + authorai_site.ge
feature_image = blogger.save_image(blueprint_feature, 'web-design-glass-on-r
post.set_feature_image(feature_image)
print(post.get_feature_image())
```

```
web-design-glass-on-night-city 20221223215344.png
```

In [17]:

```
post_created = blogger.create_jekyll_post(post, authorai_site)
print(post_created)
```

```
../authorai-site/_posts/2022-12-23-authorai-blogger-for-jekyll-static-site-
generator-automation.markdown
```