

# Generative Artificial Intelligence, Python, and Gathering Grey Literature for a Systematic Literature Review with Google's Programmable Search Engine.

Cameron Frederick Atkinson

`cameron.atkinson@utas.edu.au`

University of Tasmania

---

Case Report

Keywords:

Posted Date: March 29th, 2024

DOI: <https://doi.org/10.21203/rs.3.rs-4007725/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

---

# Abstract

Systematically gathering grey literature for use in a systematic literature review (SLR) is a challenging task, given the decentralised nature of online resources. Researchers, particularly those in the social sciences, often find themselves navigating government or non-government organisation websites, manually scouring repositories for documents relevant to their SLRs. This important stage is expensive in terms of time and resources required and, in many instances, difficult to repeat. This article introduces an innovative methodology to address these challenges, providing social science researchers with a systematic approach to gather grey literature for inclusion in SLRs. Utilising the computer programming language Python, this article leverages Google's API to create a programmable search engine, facilitating a systematic search for grey literature intended for inclusion in a SLR. A case study is presented to demonstrate the efficiency of this method in locating PDF documents, within which two examples are provided. In the first example, documents from Australian government websites ending in "gov.au" related to the topic of ("energy infrastructure" AND resilience) are sought. Secondly, "un.org" is added to illustrate how multiple websites can be searched. Highlighting the effectiveness of this approach, the study successfully locates 100 documents in just 7.5 seconds, automatically saving them into an Excel CSV file for further analysis. To the authors' knowledge, this method represents an original approach in the systematic gathering of grey literature for SLRs and highlights the contribution of generative artificial intelligence systems such as ChatGPT 3.5 in assisting to script the necessary code for new SLR tools.

## 1. INTRODUCTION

The systematic literature review (SLR) is commonly regarded as the pinnacle of evidence gathering and evidence synthesis [1, 2]. SLRs seek to gather, appraise, and synthesise evidence pertaining to a phenomenon under investigation [3]. To accomplish this, SLRs entail comprehensive searches of research databases, aiming to identify all relevant work within the field under review [4, 5]. The motivations behind a SLR are multifaceted: (1) to publicise ongoing research and prevent duplications of research; (2) to inform the planning of new research initiatives; (3) to substantiate claims of originality by contrasting existing research against new research [3, 6].

This article aims to highlight an innovative approach for researchers, particularly social science researchers engaged in SLRs, to streamline the laborious and time-consuming task of collecting grey literature. Beyond addressing temporal and financial constraints, SLRs are susceptible to becoming outdated due to their prolonged duration required for completion [7–9]. Additionally, the increasing volume of reports, among other grey literatures, presents a challenge for researchers to keep pace with the growing output. Integrating new tools, such as computational methods and generative artificial intelligence (GenAI), can indeed assist in managing and analysing this vast amount of information [10]. These technologies can enhance the efficiency, transparency, and rigor of information gathering, providing valuable support for social scientists and researchers in dealing with the expanding landscape of grey literature. Consequently, techniques that enhance the speed, efficiency, and rigor of this process prove advantageous for all SLRs.

Grey literature represents one element of the broader category known as grey information. Introduced by [11], grey information serves as a classification encompassing various forms of 'grey'; including data, literature, and information. In this context, 'grey data' pertains to user-generated web content like tweets or blogs, 'grey literature' encompasses policy documents, standards, and regulations, while 'grey information' involves informal data such as meeting notes or emails [12, 13]. This is not the first instance of creating ways to systematically gather grey literature for use in a SLR. For example, [14] highlights how web-scraping can be employed to gather grey literature from online repositories.

In SLRs, grey literature consistently proves to be among the most challenging types of literature to systematically collect [15]. Bibliographic databases, a mainstay of gathering information pertinent to conducting a SLR, are generally restricted in their ability to locate relevant grey literature [16]. Therefore, the difficulties associated with the systematic gathering of grey literature can result in its haphazard inclusion or, at times, in its complete exclusion from a review, introducing a research bias into a project [17–19].

In an effort to circumvent these biases, researchers can invest months in the rigorous collection of grey literature, sometimes employing processes that lack repeatability and introduce additional biases [11]. The issues encountered with the replicability of research contributes to an ongoing methodological challenge, making it difficult, if not impossible, to replicate the findings of numerous scientific projects [20]. To address the replicability issues surrounding the gathering of grey literature, this article introduces a new methodology to systematically gather grey literature for use in a SLR. Leveraging the power provided by Google's API to create Programmable Search Engines (PSEs), this article harnesses the robust coding ability of GenAI systems such as ChatGPT 3.5 and employs the computer programming language Python to systematically gather PDF documents from Australian government websites with the (gov.au) and (un.org) URLs.

It should be noted here that the method put forward automates what would normally be extensive and manual Google searches. There are no algorithms or machine learning techniques required for this method. However, end-users of this method will need to acquaint themselves with the Pythonic script utilised. Although relatively straightforward, this article does assume some familiarity with coding on the part of the reader. To aid researchers who are unfamiliar, this article sets out a number of steps to be taken to set up the tool. It also provides links to documents that researchers may choose to utilise during this process. Additionally, this article recognises that there are other programming languages and other GenAIs that could have been utilised to create this new method for gathering grey literature. It does not seek to premise one programming language or GenAI over another, but rather seeks to utilise the work of [10] to highlight the ability of both tools in creating a new method for researchers. As set out in [10], GenAIs are excellent tools that can assist researchers in devising new ways to conduct research. However, the role of the researcher is firmly embedded in the process of determining what GenAI outputs are functional and true.

The upcoming sections of this article will unfold as follows: the ensuing theoretical background section will familiarise readers with the three elements utilised in developing the advocated technique. Firstly, it will introduce the computer programming language Python. Secondly, it will acquaint readers with the GenAI ChatGPT 3.5, which was instrumental in assisting to script and refine the code integral to the technique. Thirdly, the article will shed light on Google's PSE, serving as the platform for this novel technique. Subsequently, the methods section will present a narrow three step case study to set out the coding devised for this innovative approach, and the following results section will present the outcomes derived from the executed code alongside a guide to utilising the method. The ensuing discussion section will summarise the work conducted, alongside discussing limitations and future research potential. It will then end with concluding remarks.

## **2. THEORETICAL BACKGROUND**

### **2.1. PYTHON**

Developed in 1990 by Dutch programmer Guido van Rossum, Python is an interpreted, interactive, object-oriented programming language. Although it has a small kernel, it can be extended by importing external libraries [21]. In fact, Python maintains a strong position because it contains a vast ecosystem of libraries as well as environments [22]. Not alone in programming languages (see Ruby [23], Java [24], Rust [25], and C# [26] for other available languages), Python was chosen as the language used in this article solely due to the authors familiarity with it. The libraries associated with the proposed tool are "OS", "Requests", and "CSV".

OS or Operating System is a Python module that allows the user to gain access to operating system information [27]. Requests is one of the most downloaded Python package with approx. 30 million downloads per week. The package allows users to send HTTP (Hypertext Transfer Protocol) requests easily [28]. Lastly, the CSV or (Comma Separated Values) format is the most common import and export format for spreadsheets and databases. It provides functionalities to read and write data in CSV format, which is a common and simple way to represent tabular data [29]. The module allows programmers to work with CSV data without getting into the specifics of the underlying CSV format, making it versatile for dealing with various applications and custom CSV formats [29]. According to [10], Python is a programming language that has had great success when paired with GenAIs such as ChatGPT 3.5 for coding assistance.

### **2.2. GENERATIVE ARTIFICIAL INTELLIGENCE**

In recent times, the field of artificial intelligence (AI) has experienced a significant surge in research growth, particularly in areas related to chatbots and other conversational agents [30, 31]. Significantly, GenAIs are generating diverse arrays of content; text, images, audio, and video [32–34]. At present, GenAI systems are being applied to address common programming challenges such as code summarisation, code review, synthesis, error repairs, and debugging [10, 35, 36]. A popular GenAI, ChatGPT 3.5 by OpenAI, is recognised for its significant potential in examining source code, suggesting changes, and generating

code [10, 30, 35, 37]. This advancement has broad applications for researchers seeking to enhance their research through computational-based methods. Different Generative AI models are currently in use, each exhibiting unique strengths and weaknesses. For a detailed comparative analysis of OpenAI ChatGPT 3.5, Microsoft Bing Chat, and Google Bard, refer to Dao (38). Furthermore, for an extensive list of existing large language models and their comparisons, consult Ray (39).

As with the choice for coding with Python, the use of ChatGPT over another GenAI is purely due to the authors familiarity with the tool. As with Python, this article does not seek to place one programming language or GenAI over another. It seeks only to highlight how a new tool for systematically gathering grey literature can be created utilising both instruments. In addition, as with [10] this article seeks to highlight how GenAI, such as ChatGPT 3.5, can be used as an assistant that contributes to the process of scripting code. Within this process the researcher is still central to developing the code, especially with respect to determining what is 'functional' versus what is 'true' [10]. Indeed, [10] highlights several caveats that researchers should be aware of when utilising GenAIs such as ChatGPT 3.5 for assistance in scripting code for use in research projects. The largest issue to be aware of is the potential 'bugginess' of the code that GenAIs are trained on. A secondary issue relates to ethical considerations. As GenAIs are trained on large datasets, for example ChatGPT 3.5 was trained on the GitHub repository, there are ethical concerns when it comes to recognising the original producers of the code used to train the GenAI. [10] suggests retroactively searching for codes once they are compiled. Fundamental to the tool created for systematically gathering grey literature for a SLR is the PSE provided by Google.

## **2.3. GOOGLE'S PROGRAMMABLE SEARCH ENGINE**

Search engines – in particular Google- are the starting point for anyone wanting to locate something on the internet [40]. Aside from general searches, Google facilitates more sophisticated search features to retrieve information (this research for example uses; site: gov.au and filetype:pdf, to retrieve only PDFs from Australian government websites). The search strategy utilised alongside "site:" and "filetype:" is ("energy infrastructure" AND resilience).

A PSE is the tailored version of a generic search engine (Google, Yahoo, Bing etc), in which every feature can be altered to fit a specific purpose(s). There are two primary cases for a PSE: one designed to search the contents of a single website, and the other tailored to concentrate on a particular topic across multiple websites. The full set of instructions to create a google PSE can be found at <https://developers.google.com/custom-search/docs/overview>. For the work being conducted a Custom search JSON API is also required (to set one up, please see <https://developers.google.com/custom-search/v1/overview>). The fundamental advantage of a PSE lies in its ability to empower researchers to create a search engine tailored for searching information on specific topics chosen by the creator. In this instance, to search for Australian government PDFs pertaining to energy infrastructures and resilience. However, should the user wish to search multiple websites, they can add extra URLs to the search query, as will be demonstrated.

## 3. METHOD

### 3.1. CASE STUDY

This section will lay out the code that ChatGPT 3.5 was prompted to create. It will do this in the form of a narrow three stage case study. Stage one is the full prompt list for creating the code and is included in the supplementary material submitted alongside this article. It is included in the supplementary material due to its size. The script developed in this article is intended for use in another research project. Therefore, ChatGPT 3.5 was prompted with that research project in mind and is not relevant to the method put forward here outside of the prompts used to script the code. Stage two is reviewing and synthesising the newly scripted code. As utilising GenAIs to assist in compiling code is a new methodology in itself, the methodology for this is set out in [10]. The final stage (three) is executing the newly scripted code. Stage three will also present two examples. The first example will be of a single site search (“gov.au”) for Australian documents. The second example will illustrate how multiple websites can be added, in this instance “gov.au” will be searched alongside “un.org.” The first 100 results for both searches will be provided in the supplementary materials.

#### 3.1.1. STAGE ONE: See supplementary materials.

#### 3.1.2. STAGE TWO: Reviewing and synthesising the scripted code.

In this stage, the scripted code from the supplementary materials is put into ChatGPT 3.5. As set out in [10], this allows the GenAI to provide an overview of the newly scripted code, and also offers an opportunity to use ChatGPT 3.5 to synthesise any parts in order to tighten the coding up further. Table 1 illustrates this process.

Table 1  
Code scripted with assistance from ChatGPT 3.5.

Prompts	Responses
<p>I am going to give you some python code to review. Can you do this for me?</p>	<p>Of course! I'd be happy to help you review your Python code. Please provide the code, and let me know if there's anything specific you would like me to focus on or if you have any questions.</p>
<p><b>#1</b></p> <pre>import requests import csv import os</pre>	<p>Certainly! It looks like you've imported the <b>requests</b>, <b>csv</b>, and <b>os</b> modules. These are commonly used modules in Python. Without additional code, it's challenging to provide a comprehensive review, but if you have specific functions or tasks you'd like me to look at, feel free to provide more details or share the rest of the code. Otherwise, if you have any questions about using these modules or if you need assistance with a particular aspect of your code, feel free to ask!</p>
<p><b>#2</b></p> <pre>def search_AUST_energy_policy_documents(query, api_key, cx, num_results = 10):  base_url = "https://www.googleapis.com/customsearch/v1"  start_index = 1  all_documents = []  try:  while True:  params = {  'q': query,  'key': api_key,  'cx': cx,  'fileType': 'pdf',  'num': num_results,  'start': start_index,  }  response = requests.get(base_url, params = params)  if response.status_code == 200:</pre>	<p>This appears to be a function for searching for Australian energy policy documents using the Google Custom Search JSON API. The function takes a query, an API key, a custom search engine ID ('cx'), and an optional parameter 'num_results' which defaults to 10. The function returns a list of documents in JSON format</p>

Prompts	Responses
<pre>search_results = response.json() documents = extract_documents(search_results) all_documents.extend(documents) # Update 'start' parameter for the next page start_index += num_results else: print(f"Error: {response.status_code}") break except Exception as e: print(f"An error occurred: {e}") return all_documents</pre>	
<pre><b>#3</b> def extract_documents(search_results): items = search_results.get('items', []) documents = [] for item in items: document = { 'title': item.get('title', ""), 'link': item.get('link', ""), 'snippet': item.get('snippet', "") } documents.append(document) return documents</pre>	<p>The 'extract_documents' function looks straightforward and serves the purpose of converting search results into a list of dictionaries, where each dictionary represents a document with title, link, and snippet information.</p>



Prompts	Responses
<p><b>#4</b></p> <pre>def save_to_csv(documents, csv_filename):     with open(csv_filename, 'w', newline="",               encoding='utf-8') as csvfile:         fieldnames = ['Title', 'Link', 'Snippet']         writer = csv.DictWriter(csvfile, fieldnames =                                 fieldnames)         writer.writeheader()         for doc in documents:             writer.writerow({'Title': doc['title'], 'Link': doc['link'],                             'Snippet': doc['snippet']})</pre>	<p>Your 'save_to_csv' function looks good and fulfills the purpose of writing the extracted document information to a CSV file.</p>
<p><b>#5</b></p> <pre>def main():     query = "site:gov.au ('energy infrastructure' AND               resilience) filetype:pdf"     api_key = "YOUR_API_KEY" # Replace with your               actual API key"     cx = "YOUR_CUSTOM_SEARCH_ENGINE_ID" #         Replace with your actual custom search engine         ID     csv_filename = "YOUR FILE DESTINATION"     num_results = 10 # Adjust as needed     search_results =     search_un_energy_policy_documents(query,     api_key, cx, num_results = num_results)     if search_results:         save_to_csv(search_results, csv_filename)     print(f"{len(search_results)} results saved to           {csv_filename}")     if __name__ == "__main__":         main()</pre>	<p>Your <b>main</b> function seems to be the entry point for your script, and it orchestrates the execution of the other functions you've defined.</p>
<p><b>I have given you 5 pieces of code. Could you rewrite it to improve the script?</b></p>	<p>Certainly! I'll provide a more polished version of the script, taking into account best practices, improved readability, and additional error handling. Here's a revised version:</p>

Prompts	Responses
	<pre>import os  import requests  import csv  def search_AUST_energy_policy_documents(query, api_key, cx, num_results = 10):  base_url = "https://www.googleapis.com/customsearch/v1"  start_index = 1  all_documents = []  try:  while True:  params = {  'q': query,  'key': api_key,  'cx': cx,  'fileType': 'pdf',  'num': num_results,  'start': start_index,  }  response = requests.get(base_url, params = params)  response.raise_for_status() # Raise an HTTPError for bad responses  search_results = response.json()  documents = extract_documents(search_results)  all_documents.extend(documents)  # Update 'start' parameter for the next page  start_index += num_results  except requests.exceptions.RequestException as e:  print(f"An error occurred during the search: {e}")</pre>

Prompts	Responses
	<pre>except Exception as e:     print(f"An unexpected error occurred: {e}") return all_documents  def extract_documents(search_results):     result_items = search_results.get('items', [])     documents = []     for item in result_items:         document = {             'title': item.get('title', 'No Title'),             'link': item.get('link', 'No Link'),             'snippet': item.get('snippet', 'No Snippet'),         }         documents.append(document)     return documents  def save_to_csv(documents, csv_filename):     try:         with open(csv_filename, 'w', newline="",             encoding='utf-8') as csvfile:             fieldnames = ['Title', 'Link', 'Snippet']             writer = csv.DictWriter(csvfile, fieldnames =                 fieldnames)             writer.writeheader()             for doc in documents:                 writer.writerow({'Title': doc['title'], 'Link': doc['link'],                     'Snippet': doc['snippet']})             print(f"{len(documents)} results saved to                 {csv_filename}")     except Exception as e:         print(f"An error occurred while writing to the CSV             file: {e}")  def main():</pre>

Prompts	Responses
	<pre>query = 'site:gov.au ("energy infrastructure" AND "resilience") filetype:pdf'  api_key = "YOUR_API_KEY" # Replace with your actual API key  cx = "YOUR_CUSTOM_SEARCH_ENGINE_ID" # Replace with your actual custom search engine ID  csv_filename = os.path.join( "YOUR FILE DESTINATION" )  num_results = 10 # Adjust as needed  try:  search_results = search_AUST_energy_policy_documents(query, api_key, cx, num_results = num_results)  save_to_csv(search_results, csv_filename) if search_results else print("No results found.")  except Exception as e:  print(f"An unexpected error occurred: {e}")  if __name__ == "__main__":  main()</pre>

Prompts	Responses
<p><b>An error occurred during the search: 400 Client Error: Bad Request for url:</b>  <a data-bbox="99 241 769 277" href="https://www.googleapis.com/customsearch [...]">https://www.googleapis.com/customsearch [...]</a></p>	<p>The error message indicates a "400 Client Error: Bad Request," which means the request sent to the Google Custom Search API is malformed or incorrect. To troubleshoot and fix this issue, you can consider the following steps:</p> <ol style="list-style-type: none"> <li>1. <b>Check API Key and CX:</b> <ul style="list-style-type: none"> <li>• Ensure that your API key (<b>api_key</b>) and custom search engine ID (<b>cx</b>) are correct and valid. Double-check that there are no typos or mistakes in these values.</li> </ul> </li> <li>2. <b>Review Query Parameters:</b> <ul style="list-style-type: none"> <li>• Examine the constructed query (<b>params</b>) in your <b>search_AUST_energy_policy_documents</b> function. Ensure that the query parameters are formatted correctly and meet the requirements of the Google Custom Search API.</li> </ul> </li> <li>3. <b>URL Encoding:</b> <ul style="list-style-type: none"> <li>• Confirm that the query parameters are properly URL-encoded. The error message in your case shows that the URL includes characters like ( , ), and spaces, which should be properly encoded.</li> </ul> </li> <li>4. <b>Limitations and Quotas:</b> <ul style="list-style-type: none"> <li>• Check if you are exceeding any limitations or quotas imposed by the Google Custom Search API. For example, there might be restrictions on the number of requests you can make in a given time period.</li> </ul> </li> </ol>
<p><b>Because I am using a free version, I can only look at 10 pages. Can you change the code to reflect this?</b></p>	<p>Certainly! If you're limited to 10 pages of results, you can modify the <b>search_AUST_energy_policy_documents</b> function to restrict the number of iterations in the while loop. Here's an updated version:</p> <pre data-bbox="833 1518 1523 1969"> def search_AUST_energy_policy_documents(query, api_key, cx, num_results = 10, max_pages = 1):  base_url = "https://www.googleapis.com/customsearch/v1"  start_index = 1  all_documents = []  try:  for _ in range(max_pages): </pre>

Prompts	Responses
	<pre> params = {     'q': query,     'key': api_key,     'cx': cx,     'fileType': 'pdf',     'num': num_results,     'start': start_index, }  response = requests.get(base_url, params = params)  response.raise_for_status() # Raise an HTTPError for bad responses  search_results = response.json()  if 'error' in search_results:      raise Exception(f"Google Custom Search API error: {search_results['error']['message']}")  documents = extract_documents(search_results) all_documents.extend(documents)  # Update 'start' parameter for the next page start_index += num_results  except requests.exceptions.RequestException as e:      print(f"An error occurred during the search: {e}")  except Exception as e:      print(f"An unexpected error occurred: {e}")  return all_documents </pre>
<p>Thank you, the code is running as expected.</p>	<p>You're welcome! I'm glad to hear that the code is running as expected. If you have any more questions, need further assistance, or have other code you'd like me to review, feel free to reach out. Happy coding!</p>

Following this, and in line with the code found in [14], a User-Agent header was included. This was to ensure transparency on the agent end of the code, and to indicate that it was not a 'bot' behind the helm.

This addition is indicated in Table 2.

Table 2  
The addition of a User-Agent Header to the script.

Prompt	Response
Should I include a user-agent header in the code?	<p>Yes, including a User-Agent header in your code is a good practice. The User-Agent header provides information about the client making the request, such as the type of browser or application and its version. Some websites and APIs may use this information to identify and differentiate between various clients.</p> <p>Including a User-Agent header helps in the following ways:</p> <ol style="list-style-type: none"><li><b>1. Identification:</b> It allows the website or API server to identify your request and can be used for statistical or analytical purposes.</li><li><b>2. Avoiding Blocks:</b> Some websites implement access controls or rate limiting based on User-Agent. Providing a User-Agent that looks similar to a typical web browser can help avoid being blocked or flagged as suspicious.</li><li><b>3. Good Etiquette:</b> Including a User-Agent is considered good etiquette.</li></ol> <p>Here's how you can modify your code to include a User-Agent header using the <b>headers</b> parameter in the <b>requests.get</b> method:</p> <pre>def search_AUST_energy_policy_documents(query, api_key, cx, num_results = 10, max_pages = 1):  base_url = "https://www.googleapis.com/customsearch/v1"  start_index = 1  all_documents = []  headers = { 'User-Agent': 'Your-User-Agent-Here', }  try:  for _ in range(max_pages):  params = { 'q': query, 'key': api_key, 'cx': cx, 'fileType': 'pdf', 'num': num_results, 'start': start_index, }  response = requests.get(base_url, params = params, headers = headers)</pre>



Prompt	Response
	<pre>response.raise_for_status() # Raise an HTTPError for bad responses search_results = response.json() if 'error' in search_results:     raise Exception(f"Google Custom Search API error: {search_results['error']['message']}") documents = extract_documents(search_results) all_documents.extend(documents) # Update 'start' parameter for the next page start_index += num_results except requests.exceptions.RequestException as e:     print(f"An error occurred during the search: {e}") except Exception as e:     print(f"An unexpected error occurred: {e}") return all_documents</pre>

To assist with people utilising the method presented. Table 3 provides a full account of the code. With this new method, researchers will be able to systematically gather grey literature for their SLRs, saving significant time and money in the process. Users will have to ensure that the Python libraries that are utilised are installed on their computers, and that they input the correct information into the areas that are highlighted in all caps, bold, and italics. In the instance of this research project, the author ran the code in a Jupyter notebook in Visual Studio Code.

Table 3  
Full account of the newly scripted code.

```
import os
import requests
import csv

def search_AUST_energy_policy_documents(query, api_key, cx, num_results = 10, max_pages = 10):
    base_url = "YOUR COUSTOM SEARCH ENGINE"
    start_index = 1
    all_documents = []
    headers = {
        'User-Agent' YOUR USER-AGENT INFORMATION,
    }
    try:
        for _ in range(max_pages):
            params = {
                'q': query,
                'key': api_key,
                'cx': cx,
                'fileType': 'pdf',
                'num': num_results,
                'start': start_index,
            }
            response = requests.get(base_url, params = params, headers = headers)
            response.raise_for_status() # Raise an HTTPError for bad responses
            search_results = response.json()
            if 'error' in search_results:
                raise Exception(f"Google Custom Search API error: {search_results['error']['message']}")
            documents = extract_documents(search_results)
            all_documents.extend(documents)
            # Update 'start' parameter for the next page
            start_index += num_results
```

```

except requests.exceptions.RequestException as e:
    print(f"An error occurred during the search: {e}")
except Exception as e:
    print(f"An unexpected error occurred: {e}")
return all_documents

def extract_documents(search_results):
    result_items = search_results.get('items', [])
    documents = []
    for item in result_items:
        document = {
            'title': item.get('title', 'No Title'),
            'link': item.get('link', 'No Link'),
            'snippet': item.get('snippet', 'No Snippet'),
        }
        documents.append(document)
    return documents

def save_to_csv(documents, csv_filename):
    try:
        with open(csv_filename, 'w', newline="", encoding='utf-8') as csvfile:
            fieldnames = ['Title', 'Link', 'Snippet']
            writer = csv.DictWriter(csvfile, fieldnames = fieldnames)
            writer.writeheader()
            for doc in documents:
                writer.writerow({'Title': doc['title'], 'Link': doc['link'], 'Snippet': doc['snippet']})
            print(f"{len(documents)} results saved to {csv_filename}")
    except Exception as e:
        print(f"An error occurred while writing to the CSV file: {e}")

def main():
    query = 'site:gov.au ("energy infrastructure" AND ("resilience")) filetype:pdf'
    api_key = "YOUR API KEY"

```

```
cx = "YOUR CUSTOM SEARCH ENGINE ID"
csv_filename = os.path.join(
YOUR FILE LOCATION\\, ".csv"
)
num_results = 10 # Adjust as needed
try:
search_results = search_AUST_energy_policy_documents(query, api_key, cx, num_results =
num_results)
save_to_csv(search_results, csv_filename) if search_results else print("No results found.")
except Exception as e:
print(f"An unexpected error occurred: {e}")
if __name__ == "__main__":
main()
```

### 3.1.3. STAGE THREE: execution

The executed code provides the output "100 results saved to **File location\\ XXX.results.csv**". It accomplished this in 7.5 seconds, and automatically saved a CSV file to the specified location. The next section will provide the results followed by a guide for researchers who wish to utilise this method.

## 4. FINDINGS

The scripted code provided the following results. A CSV file was produced that contained the following information: three columns with the headings 'Title', 'Link', and 'Snippet.' Each column contained the information for 100 PDFs identified in the search. Figures 1 and 2 provides screenshot snippets of the Titles and their Links that were uncovered. Each PDF link, when copied and pasted into a web browser, is functional. The search strategy words "energy infrastructure" AND "resilience" is contained in either the title or the snippet. The full CSV file is provided in the supplementary materials. Figure 3 provides a screenshot snippet of the Blurb accompanying each result.

Within the search strategy the URL "un.org" was added. This was done to illustrate the ease in which multiple websites can be searched. A snippet of the Titles and Links from this search can be viewed in Fig. 4.

These results illustrate the speed, rigor, and transparency of the new tool and underscores its importance. Capturing every tier of Australian government, the above results provide grey literature from Victoria, New South Whales, Queensland, Torris Straight, and the Australian Federal Government. Should users wish to narrow the search to these states, they would simply change the URL to "vic.gov.au" for Victoria, or

“qld.gov.au” for Queensland. Should users wish to add other URLs such as “un.org” they need to only add the OR Boolean to the URL component of the strategy; for example, “ 'site:gov.au OR site:un.org ("energy infrastructure" AND ("resilience")) filetype:pdf ’ ”

## 4.1. GUIDE FOR RESEARCHERS.

For researcher who wish to utilise this new methodology, the following steps are recommended.

### STEP ONE

Download Python from <https://www.python.org/downloads/>.

### STEP TWO

Familiarise yourself with how libraries are installed in Python. Pip is the preferred installer program to be used in Python. You can find instructions on how to do so at <https://packaging.python.org/en/latest/tutorials/installing-packages/>. There are also a large number of YouTube videos that will not only provide in-depth instructions for this stage but will also provide a deeper introduction to Python.

### STEP THREE

Install the libraries required for the tool. You will need to install “OS”, “Requests”, and “CSV”.

### STEP FOUR

Using the google documents provided in the google portion of the theoretical background section, follow the instructions to create a google PSE and to obtain a JSON API. For the PSE component see <https://developers.google.com/custom-search/docs/overview>. For obtaining the API key see <https://developers.google.com/custom-search/v1/overview>).

### STEP FIVE

Copy and paste the full code provide in this methods paper into a Jupyter notebook (other notebooks will also support this code, in this instance the author utilised a Jupyter notebook). You will also need to ensure that you input your own User-Agent Header to the script. Also, change the search query to reflect the information being sought. Lastly, you will need to designate a location for the csv file to be saved in.

### STEP SIX

Run the code. Should any errors be encountered, [10] offers a method to help resolve them. Figure 5 provides a visual representation of six step process.

## 5. DISCUSSION AND LIMITATIONS

The above sections have presented a new tool for researchers to systematically gather grey literature for SLRs. It has illustrated that Titles, Links, and Blurbs of PDF files pertaining to “energy infrastructure” AND “resilience” can be identified, saved, and accessed in a remarkably small amount of time. This technique will not only save researchers resources (time and money), but it will also increase the rigor, transparency, and efficiency of this stage in a SLR.

As highlighted in [10, 41], AI, and in this instance GenAI, will have an impact on the world that is akin to the impact of the industrial revolution, perhaps greater. To this end, this is not the first instance of computational-based methods being applied to stages of SLRs. This article not alone in creating new methods that infuse AI and MLTs within SLRs. [42] provides a thorough account of how AI and MLTs can be applied to different stages of a SLR. Focusing primarily on how Latent Dirichlet Allocation (LDA) Topic Models can be used to abstract and synthesise data obtained during a review, [42] also introduces AI and MLTs that can be utilised in the title/abstract screening stage, as well as the data extraction stage. [14] illustrates how Python can be utilised to create a web-scraper to gather grey literature from a single online data repository. Indeed, incorporating AI and MLTs into the SLR process is the topic of contemporary works investigating how they can be inserted into current guidelines. For an account on how AI and MLTs can be incorporated within the Preferred Reporting Items for Systematic reviews and Meta-Analyses (PRISMA) guidelines, please see [43] on how this could be achieved.

The scripted code presented in this article will transform how grey literature is gathered for use in a SLR. Although cheap, extensive use of this tool is not one-hundred percent free. The results presented are limited to the free-use terms of Google’s API key. If researchers require a greater number of searches, and more results, then they will have to purchase this ability from Google. Additionally, to keep the scripted code as simple as possible for the reader, more diverse search strategies alongside data organisation and saving functions have not been included. This will be left for future research. [10] highlighted that there is a large ethical concern regarding appropriate recognition of the coding that GenAIs are trained on. Retrospective searches of coding was suggested as a possible way to attribute codes. At the time of writing, similar code has not been located. Although it is impossible to be certain where the code originated, looking for it is a step forward in the process of solving the ethical issues of utilising GenAIs to script code.

A notable limitation of this article is that researchers may not be acquainted with the tools employed. The intention is that by furnishing external resources, direction, and a guide, the technical gap necessary to employ the proposed method has been somewhat alleviated, making it more accessible for a broader audience.

## **6. FUTURE RESEARCH**

The tool that has been introduced in this article is intended to be employed to gather pertinent policy documents, standards, laws, and regulations to be utilised in a connected research project. Future research will also include scripting code to increase the sophistication of the queries that can be entered,

alongside different methods to organise and save the data generated through this new process. Applications for areas outside of grey literature are also being investigated.

## 7. CONCLUSION

This article has introduced a new and novel tool to aid in gathering grey literature for use in a SLR. Though the use of the computer programming language Python, coding assistance of ChatGPT 3.5, and the ability to create PSEs from Google, this article has introduced a technique for researchers to cheaply, quickly, rigorously, and transparently gather grey literature. No longer requiring long hours and unreliable search processes, this new tool will help every discipline to gather grey literature for their research.

## Declarations

### Funding:

Natural Hazards Research Australia.

### Conflicts of Interests/Competing Interests:

The author has neither conflicts of interests nor competing interests to declare that are relevant to the content of this article.

### Data Availability:

All data pertinent to the article has been made available in the supplementary materials.

### Code Availability:

All codes utilised in this research project have either been provided in text or are contained in the supplementary materials.

### Use of AI:

The Generative AI ChatGPT 3.5 was utilised in the methods section of this article.

### Author Contribution

C.A, as sole author, is responsible for all aspects of this manuscript.

## References

1. Thomé AMT, Scavarda LF, Scavarda AJ. Conducting systematic literature review in operations management. (2016). *Production Planning & Control*, 27(5). 408–20.  
<https://doi.org/10.1080/09537287.2015.1129464>.

2. Pati D, Lorusso LN. How to write a systematic review of the literature. (2018). *HERD: Health Environments Research & Design Journal*, 11(1). 15–30. <https://doi.org/10.1177/1937586717747384>.
3. Petticrew M, Roberts H. *Systematic reviews in the social sciences: A practical guide*. John Wiley & Sons; 2008.
4. Xiao Y, Watson M. Guidance on Conducting a Systematic Literature Review. (2019). *Journal of Planning Education and Research*, 39(1). 93–112. <https://doi.org/10.1177/0739456x17723971>.
5. Siddaway AP, Wood AM, Hedges LV. How to Do a Systematic Review: A Best Practice Guide for Conducting and Reporting Narrative Reviews, Meta-Analyses, and Meta-Syntheses. (2019). *Annual Review of Psychology*, 70(1). 747–70. <https://doi.org/10.1146/annurev-psych-010418-102803>.
6. Paul J, Lim WM, O’Cass A, Hao AW, Bresciani S. Scientific procedures and rationales for systematic literature reviews (SPAR-4-SLR). (2021). *International Journal of Consumer Studies*, 45(4). 01-016. <https://doi.org/10.1111/ijcs.12695>.
7. Sundaram G, Berleant D. Automating Systematic Literature Reviews with Natural Language Processing and Text Mining: a Systematic Literature Review. (2022). ArXiv Preprint <https://doi.org/10.48550/arXiv.2211.15397>.
8. Marshall IJ, Wallace BC. Toward systematic review automation: a practical guide to using machine learning tools in research synthesis. (2019). *Systematic reviews*, 8. 1–10. <https://doi.org/10.1186/s13643-019-1074-9>.
9. Shojania KG, Sampson M, Ansari MT, Ji J, Doucette S, Moher D. How quickly do systematic reviews go out of date? A survival analysis. (2007). *Annals of internal medicine*, 147(4). 224 – 33. <https://doi.org/10.7326/0003-4819-147-4-200708210-00179>.
10. Atkinson CF. ChatGPT and computational-based research: benefits, drawbacks, and machine learning applications. (2023). *Discover Artificial Intelligence*, 3(1). 42. <https://doi.org/10.1007/s44163-023-00091-3>.
11. Adams J, Hillier-Brown FC, Moore HJ, Lake AA, Araujo-Soares V, White M, et al. Searching and synthesising ‘grey literature’ and ‘grey information’ in public health: critical reflections on three case studies. (2016). *Systematic Reviews*, 5(1). 164. <https://doi.org/10.1186/s13643-016-0337-y>.
12. Kamei F, Wiese I, Pinto G, Ribeiro M, Soares S, editors. On the use of grey literature: A survey with the brazilian software engineering research community. *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*; 2020. <https://doi.org/10.1145/3422392.3422442>
13. Banks M. *Blog posts and tweets: the next frontier for grey literature*. *Grey literature in library and information studies*: De Gruyter; 2009.
14. Atkinson C. Cheap, Rigorous, and Transparent: How Web-scraping with Python can Improve Collecting Grey Literature for Systematic Literature Reviews. (2023). *Grey Journal (TGJ)*, 19(3).
15. Paez A. Gray literature: An important resource in systematic reviews. (2017). *Journal of Evidence-Based Medicine*, 10(3). 233 – 40. <https://doi.org/10.1111/jebm.12266>.



16. Grayson L, Gomersall A. Google – boon, bane or both? (2006). *Evid Polciy*, 2(4). 538 – 42. <https://doi.org/10.1332/174426406778881818>.
17. Haddaway NR, Bethel A, Dicks LV, Koricheva J, Macura B, Petrokofsky G, et al. Eight problems with literature reviews and how to fix them. (2020). *Nature Ecology & Evolution*, 4(12). 1582–9. <https://doi.org/10.1038/s41559-020-01295-x>.
18. Godin K, Stapleton J, Kirkpatrick SI, Hanning RM, Leatherdale ST. Applying systematic review search methods to the grey literature: a case study examining guidelines for school-based breakfast programs in Canada. (2015). *Systematic Reviews*, 4(1). 138. <https://doi.org/10.1186/s13643-015-0125-0>.
19. Winters M, Weir A. Grey matters; on the importance of publication bias in systematic reviews. (2017). 51(6). 488–9. <https://doi.org/10.1136/bjsports-2016-096679>.
20. Echtler F, Häußler M, editors. Open source, open science, and the replication crisis in HCI. Extended abstracts of the 2018 CHI conference on human factors in computing systems; 2018. <https://doi.org/10.1145/3170427.3188395>
21. Nagpal A, Gabrani G, editors. Python for data analytics, scientific and technical applications. 2019 Amity international conference on artificial intelligence (AICAI); 2019: IEEE. <https://doi.org/10.1109/AICAI.2019.8701341>
22. Kumar R. Future for scientific computing using Python. (2015). *International Journal of Engineering Technologies and Management Research*, 2(1). 30–41. <https://doi.org/10.29121/ijetmr.v2.i1.2015.28>.
23. Tilkov S, Vinoski S. Node.js: Using JavaScript to build high-performance network programs. (2010). *IEEE Internet Computing*, 14(6). 80–3. <https://doi.org/10.1109/MIC.2010.145>.
24. Arnold K, Gosling J, Holmes D. *The Java programming language*. Addison Wesley Professional; 2005.
25. Klabnik S, Nichols C. *The Rust programming language*. No Starch Press; 2023.
26. Hejlsberg A, Torgersen M, Wiltamuth S, Golde P. *The C# programming language*. Pearson Education; 2008.
27. Python Software Foundation. OS – Miscellaneous operating system interfaces Online: Python Software Foundation; (2024) [Available from: <https://docs.python.org/3/library/os.html>].
28. Python Software Foundation. Requests 2.31.0 Online: Python Software Foundation; (2024) [Available from: <https://pypi.org/project/requests/>].
29. Python Software Foundation. CSV File Reading and Writing Online: Python Software Foundation; (2024) [Available from: <https://docs.python.org/3/library/csv.html>].
30. Azaria A, Azoulay R, Reches S. ChatGPT is a Remarkable Tool–For Experts. (2023). *ArXiv Preprint* <https://doi.org/10.48550/arXiv.2306.03102>.
31. Olujimi PA, Ade-Ibijola A. NLP techniques for automating responses to customer queries: a systematic review. (2023). *Discover Artificial Intelligence*, 3(1). 20. <https://doi.org/10.1007/s44163->

023-00065-5.

32. Morris MR. Scientists' Perspectives on the Potential for Generative AI in their Fields. (2023). ArXiv Preprint <https://doi.org/10.48550/arXiv.2304.01420>.
33. Crawford K, Paglen T. Excavating AI: the politics of images in machine learning training sets. (2021). *AI & SOCIETY*, 36(4). 1105–16. <https://doi.org/10.1007/s00146-021-01162-8>.
34. Hacker P, Engel A, Mauer M, editors. Regulating ChatGPT and other large generative AI models. Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency; 2023. <https://doi.org/10.1145/3593013.3594067>
35. Tian H, Lu W, Li TO, Tang X, Cheung S-C, Klein J, et al. Is ChatGPT the Ultimate Programming Assistant—How far is it? (2023). *ArXiv Preprint* <https://doi.org/10.48550/arXiv.2304.11938>.
36. Haque MA, Li S. The Potential Use of ChatGPT for Debugging and Bug Fixing. (2023). *EAI Endorsed Transactions on AI and Robotics*, 2(1). e4-e. <https://doi.org/10.4108/airo.v2i1.3276>.
37. Liu J, Xia CS, Wang Y, Zhang L. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. (2023). ArXiv Preprint <https://doi.org/10.48550/arXiv.2305.01210>.
38. Dao X-Q. Performance comparison of large language models on vnhsge english dataset: Openai chatgpt, microsoft bing chat, and google bard. (2023). *ArXiv Preprint*. <https://doi.org/10.48550/arXiv.2307.02288>.
39. Ray PP. ChatGPT: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. (2023). *Internet of Things and Cyber-Physical Systems*, 3. 121 – 54. <https://doi.org/10.1016/j.iotcps.2023.04.003>.
40. Dhara A. A personalised discovery service using Google custom search engine. (2016). *Annals of Library and information Studies*, 63. 298–305. Available from: <https://nopr.niscpr.res.in/bitstream/123456789/39764/1/ALIS%2063%284%29%20298-305.pdf>.
41. Medaglia R, Gil-Garcia JR, Pardo TA. Artificial Intelligence in Government: Taking Stock and Moving Forward. (2021). *Social Science Computer Review*, 41(1). 123–40. <https://doi.org/10.1177/08944393211034087>.
42. Atkinson CF. Cheap, Quick, and Rigorous: Artificial Intelligence and the Systematic Literature Review. (2023). *Social Science Computer Review*. <https://doi.org/10.1177/08944393231196281>.
43. Susnjak T. Prisma-dflm: An extension of prisma for systematic literature reviews using domain-specific finetuned large language models. (2023). *ArXiv Preprint* 1–20. <https://doi.org/10.48550/arXiv.2306.14905>.

## Figures

1	Title
2	A Pathway to Infrastructure Resilience Advisory Paper 1 ...
3	A Pathway to Infrastructure Resilience
4	Building Victoria's Climate Resilience
5	NSW Critical Infrastructure Resilience Strategy Guide
6	Electricity Distribution Network Resilience Review - Final ...
7	2021 National Research Infrastructure Roadmap   Education.gov.au
8	State Infrastructure Strategy Energy
9	Torres Strait Climate Change Strategy 2014-2018: Building ...
10	South West Regional Resilience Strategy

Figure 1

Screenshot snippet of Titles from the new tool.

Link
<a href="https://www.infrastructureaustralia.gov.au/sites/default/files/2021-08/Advisory%20Paper%201%20-%20A%20pathway%20to%20Infrastructure%20Resilience%20FINAL.pdf">https://www.infrastructureaustralia.gov.au/sites/default/files/2021-08/Advisory%20Paper%201%20-%20A%20pathway%20to%20Infrastructure%20Resilience%20FINAL.pdf</a>
<a href="https://www.infrastructureaustralia.gov.au/sites/default/files/2021-08/Advisory%20Paper%202%20-%20A%20pathway%20to%20Infrastructure%20Resilience%20FINAL.pdf">https://www.infrastructureaustralia.gov.au/sites/default/files/2021-08/Advisory%20Paper%202%20-%20A%20pathway%20to%20Infrastructure%20Resilience%20FINAL.pdf</a>
<a href="https://www.climatechange.vic.gov.au/_data/assets/pdf_file/0023/558140/BuildingVictoriaClimateResilience.pdf">https://www.climatechange.vic.gov.au/_data/assets/pdf_file/0023/558140/BuildingVictoriaClimateResilience.pdf</a>
<a href="https://www.opengov.nsw.gov.au/download/19752">https://www.opengov.nsw.gov.au/download/19752</a>
<a href="https://www.energy.vic.gov.au/_data/assets/pdf_file/0030/594930/Network-Resilience-Review-Final-Recommendations-Report.pdf">https://www.energy.vic.gov.au/_data/assets/pdf_file/0030/594930/Network-Resilience-Review-Final-Recommendations-Report.pdf</a>
<a href="https://www.education.gov.au/download/13735/2021-national-research-infrastructure-roadmap/26919/2021-national-research-infrastructure-roadmap/pdf">https://www.education.gov.au/download/13735/2021-national-research-infrastructure-roadmap/26919/2021-national-research-infrastructure-roadmap/pdf</a>
<a href="https://www.statedevelopment.qld.gov.au/_data/assets/pdf_file/0023/72095/state-infrastructure-strategy-energy.pdf">https://www.statedevelopment.qld.gov.au/_data/assets/pdf_file/0023/72095/state-infrastructure-strategy-energy.pdf</a>
<a href="https://www.tsra.gov.au/_data/assets/pdf_file/0003/7419/TSRA-Climate-Change-Strategy-2014-2018-Upload4.pdf">https://www.tsra.gov.au/_data/assets/pdf_file/0003/7419/TSRA-Climate-Change-Strategy-2014-2018-Upload4.pdf</a>
<a href="https://www.qra.qld.gov.au/sites/default/files/2022-07/south_west_regional_resilience_strategy_july_2022.pdf">https://www.qra.qld.gov.au/sites/default/files/2022-07/south_west_regional_resilience_strategy_july_2022.pdf</a>

Figure 2

Screenshot snippet of the Links from the new tool.

Snippet
a clear articulation of resilience to what, resilience of what and resilience for whom ... Energy infrastructure refers to electricity generation, transmission,Â ...
The continued operation of energy infrastructure is a significant contributor to community resilience. As further outlined in Advisory Paper 1, theÂ ...
Feb 7, 2022 ... Building energy infrastructure resilience, from upgrading powerlines to reduce bushfire risks to supporting microgrids, batteries andÂ ...
energy infrastructure in Finland. Conference: Baltica X, At Helsinki, Volume ... INFRASTRUCTURE RESILIENCE GUIDE. 19. Page 20. The future of infrastructureÂ ...
May 5, 2022 ... ... resilience, operational resilience and community resilience ... â€œAs the owners and operators of energy infrastructure and supply chains, theÂ ...
the National Climate Resilience and Adaptation Strategy, which is positioning Australia to better anticipate, manage and adapt to our changing climate.
47 With our globally significant resources, we've experienced a boom in renewable energy infrastructure investment at the household and industry level. This hasÂ ...
This will be undertaken through the Torres Strait Adaptation and Resilience Project. Understanding the risks and vulnerability of communities and ecosystems toÂ ...
Jul 3, 2022 ... The South West Regional Resilience Strategy disaster resilience ... As for other townships, energy infrastructure is exposed to bushfire risk withÂ ...

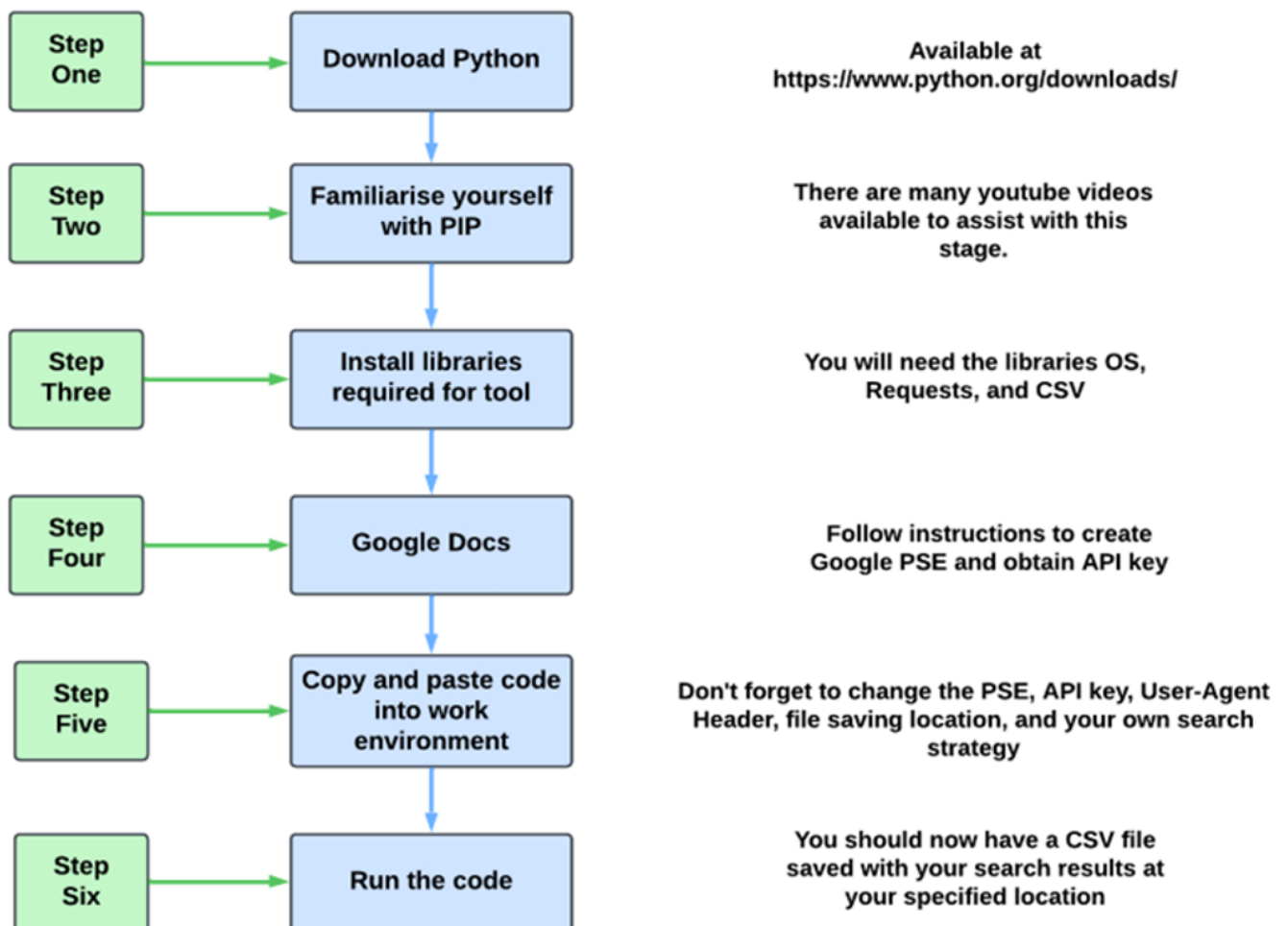
Figure 3

Screenshot snippet of the Blurbs accompanying each result from the new tool.

Title	Link
Concept Note: Island Resilience Partnership Event Title of Event ...	<a href="https://sustainabledevelopment.un.org/content/documents/2090910.UNDP_GEF_SGP.pdf">https://sustainabledevelopment.un.org/content/documents/2090910.UNDP_GEF_SGP.pdf</a>
A Pathway to Infrastructure Resilience Advisory Paper 1 ...	<a href="https://www.infrastructureaustralia.gov.au/sites/default/files/2021-08/Advisory%20Paper">https://www.infrastructureaustralia.gov.au/sites/default/files/2021-08/Advisory%20Paper</a>
A Pathway to Infrastructure Resilience	<a href="https://www.infrastructureaustralia.gov.au/sites/default/files/2021-08/Advisory%20Paper">https://www.infrastructureaustralia.gov.au/sites/default/files/2021-08/Advisory%20Paper</a>
QUESTIONNAIRE Implementation of the SAMOA Pathway and the ...	<a href="https://sdgs.un.org/sites/default/files/2023-08/Austria_Inputs_SGReport_2023.pdf">https://sdgs.un.org/sites/default/files/2023-08/Austria_Inputs_SGReport_2023.pdf</a>
Enhancing energy security in the context of the coronavirus disease ...	<a href="https://digitallibrary.un.org/record/4011285/files/ESCAP_77_17-EN.pdf">https://digitallibrary.un.org/record/4011285/files/ESCAP_77_17-EN.pdf</a>
SECTION 1: AMBITION	<a href="https://www.un.org/sites/un2.un.org/files/v3_-_energy_compact_wmo_energy_toolkit_re">https://www.un.org/sites/un2.un.org/files/v3_-_energy_compact_wmo_energy_toolkit_re</a>
Building Victoria's Climate Resilience	<a href="https://www.climatechange.vic.gov.au/__data/assets/pdf_file/0023/558140/BuildingVictori">https://www.climatechange.vic.gov.au/__data/assets/pdf_file/0023/558140/BuildingVictori</a>
NSW Critical Infrastructure Resilience Strategy Guide	<a href="https://www.opengov.nsw.gov.au/download/19752">https://www.opengov.nsw.gov.au/download/19752</a>
ENERGY TRANSITION	<a href="https://www.un.org/sites/un2.un.org/files/2021/11/2021-twg_2.pdf">https://www.un.org/sites/un2.un.org/files/2021/11/2021-twg_2.pdf</a>
Electricity Distribution Network Resilience Review - Final ...	<a href="https://www.energy.vic.gov.au/__data/assets/pdf_file/0030/594930/Network-Resilience-F">https://www.energy.vic.gov.au/__data/assets/pdf_file/0030/594930/Network-Resilience-F</a>

Figure 4

Screenshot snippet from the updated search.



## Figure 5

Stages for setting up tool.

## Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [PROMPT1.doc](#)
- [SEARCH2.csv](#)
- [SearchResultsAU.csv](#)