Technique

```
import os
import requests
import sys

file = f"{sys.argv[1]}"
path = os.getcwd() + file

sub_list = open(file).read()
subdoms = sub_list.splitlines()

for sub in subdoms:
sub_domains = f"http://{sub}.{sys.argv[2]}"
```

```
'''The script iterates through each subdomain in the list and

try:
requests.get(sub_domains)
except requests.ConnectionError:
pass

else:
print("Valid domain: ", sub_domains)
```

Import Statements:

- `import os`
  `import requests`
  `import sys`

  These statements import necessary modules for file handling (`os`), making HTTP requests (`requests`), and accessing command-line arguments (`sys`).
- **Command-Line Arguments:**
- `file = f"{sys.argv[1]}"`
  `path = os.getcwd() + file`

Our command will be `python3 python_file_name mainfile.txt domain` . For this to be happened

we have to use sys.argv[1] this is only for the mainfile.txt that we have selected. We are using `file = f"{sys.argv[1]}"` `path = os.getcwd() + file`

only for the commands.

  - `sys.argv` is a list in Python that contains the command-line arguments passed to the script.
  - `sys.argv[0]` is the script name itself, so `sys.argv[1]` refers to the first command-line argument provided by the user.
  - The `file` variable holds the value of the first command-line argument, which is expected to be a file name.
- **File Path Construction:**
- `path = os.getcwd() + file`
  - This line attempts to construct an absolute file path by concatenating the current working directory (`os.getcwd()`) with the provided file name.
- **Reading Subdomains from File:**
- `sub_list = open(file).read()`
`subdoms = sub_list.splitlines()`
  - The script opens the file specified by the user (or provided as a command-line argument), reads its contents, and splits them into a list of subdomains (`subdoms`) using `splitlines()`.
- **Iterating through Subdomains:**
- `for sub in subdoms: sub_domains = f"http://{sub}.{sys.argv[2]}" ```` `- The script iterates through each subdomain in the list and constructs a full URL by combining it with the`
`second command-line argument (`sys.argv[2]`), which is expected to be a domain.`
- **HTTP Request and Exception Handling:**
- `try: requests.get(sub_domains)`
`except requests.ConnectionError: pass`

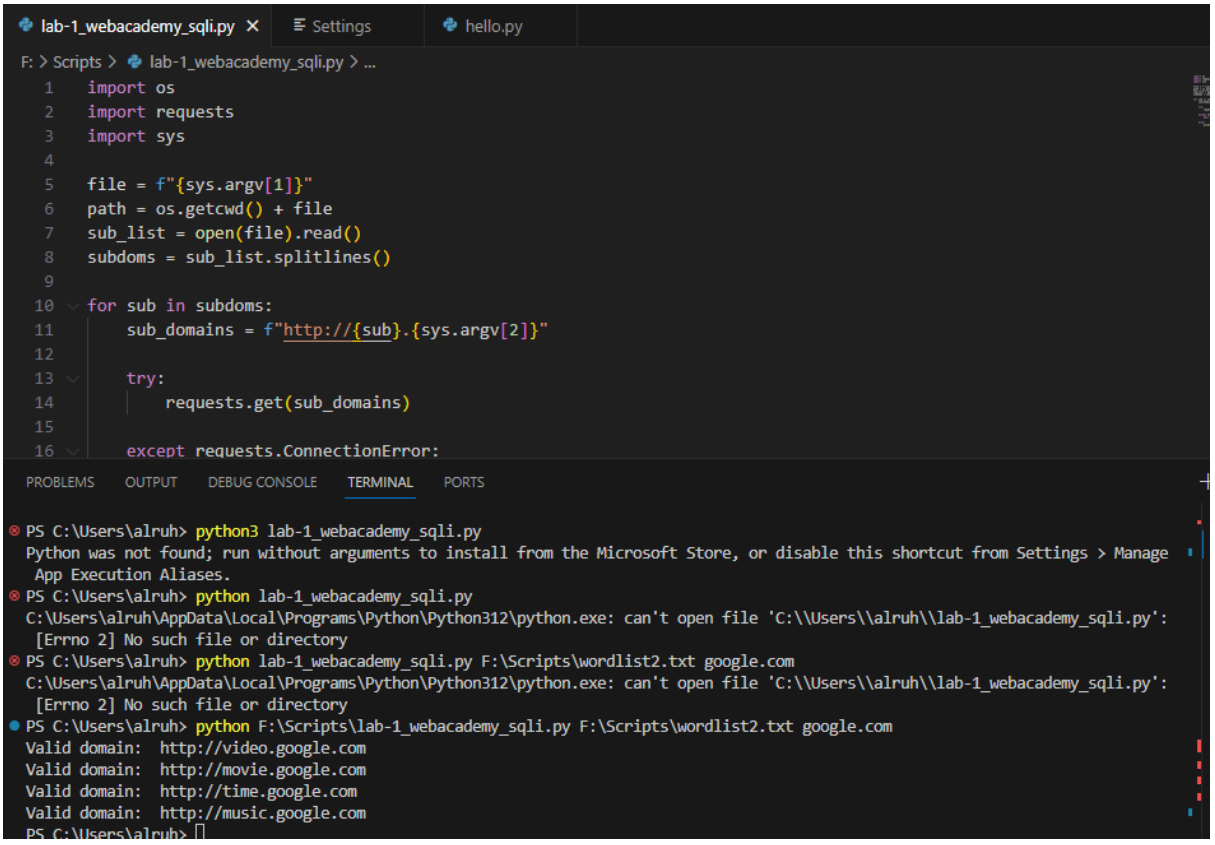
- The script attempts to make an HTTP GET request to the constructed URL using `requests.get()`.
- If a `requests.ConnectionError` occurs (indicating a failed connection), it is caught and ignored.

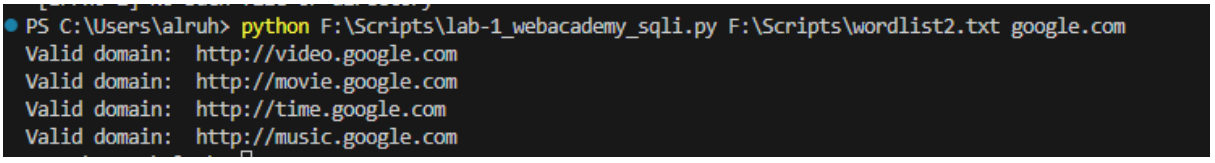
- **Printing Valid Domains:**

1. `else: print("Valid domain: ", sub_domains)`
   - If the HTTP request is successful (no exception is raised), it prints the message "Valid domain" along with the constructed URL.

In summary, this script reads a list of subdomains from a file, combines each subdomain with a provided domain, and checks if the resulting URLs are reachable by making HTTP requests. If a conn

ection is successful, it prints a message indicating a valid domain.





here look i’m not in the directory of the python file so i had to define the files directory