Q1. Using command-line arguments involves the sysmodule.Review the docs for this module and using the information in there write a short program that when run from the command-line reports what operating system platform is being used.

Ans:

```
import sys

def main():
    platform = sys.platform
    print(f"You are running on: {platform}")
if __name__ == "__main__":
    main()
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Python Programs\Week-05>Python program1.py
You are running on: win32
```

Q2. Write a program that, when run from the command line, reports how many arguments were provided. (Remember that the program name itself isnotan argument).

Ans:

```
import sys

def main():
    num_arguments = len(sys.argv) - 1
    print(f"Number of arguments provided: {num_arguments}")
if __name__ == "__main__":
    main()
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Python Programs\Week-05>python program2.py arg1 arg2 arg3
Number of arguments provided: 3
```

Q3. Write a program that takes a bunch of command-line arguments, and then prints out the shortest. If there is more than one of the shortest length, any will do.

Hint: Don't overthink this. A good way to find the shortest is just to sort them.'''

Ans:

```
import sys

def main():
    arguments = sys.argv[1:]
     if not arguments:
        print("No arguments provided.")
        return
    shortest_arg = sorted(arguments, key=len)[0]
    print(f"The shortest argument is: {shortest_arg}")
if __name__ == "__main__":
    main()
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Python Programs\Week-05>python program3.py cow buffelo elephent
The shortest argument is: cow
```

Q4. Write a program that takes a URL as a command-line argument and reports whether or not there is a working website at that address.

Hint: You need to get the HTTP response code.

Another Hint: StackOverflow is your friend.'''

Ans:

```
import sys
import requests
def main():
    if len(sys.argv) != 2:
        print("Usage: python check_website.py <URL>")
        return
url = sys.argv[1]
 try:
        response = requests.head(url, timeout=5)
        if 200 <= response.status_code < 400:
            print(f"The website at {url} is working (Status Code: {response.status_code}).")
        else:
            print(f"The website at {url} is not working (Status Code: {response.status_code}).")
    except requests.exceptions.RequestException as e:
        print(f"Failed to connect to {url}. Error: {e}")
if __name__ == "__main__":
    main()
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Python Programs\Week-05>python program4.py https://www.google.com
The website at https://www.google.com is working (Status Code: 200).
```

Q5. Last week you wrote a program that processed a collection of temperature readings entered by the user and displayed the maximum, minimum, and mean. Create a version of that program that takes the values from the command-line instead. Be sure to handle the case where no arguments are provided!

Ans:

```
import sys

def main():
    if len(sys.argv) < 2:
        print("No temperature readings provided. Please provide temperature values as command-line arguments.")
        return
    try:
        temperatures = [float(arg) for arg in sys.argv[1:]]
    except ValueError:
        print("Please provide valid numeric temperature readings.")
        return
    max_temp = max(temperatures)
    min_temp = min(temperatures)
    mean_temp = sum(temperatures) / len(temperatures)

    print(f"Maximum temperature: {max_temp}")
    print(f"Minimum temperature: {min_temp}")
    print(f"Mean temperature: {mean_temp:.2f}")

if __name__ == "__main__":
    main()
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Python Programs\Week-05>python program5.py 20.5 21.5 26 27
Maximum temperature: 27.0
Minimum temperature: 20.5
Mean temperature: 23.75
```

Q6. Write a program that takes the name of a file as a command-line argument, and creates a backup copy of that file. The backup should contain an exact copy of the contents of the original and should, obviously, have a different name.

Hint: By now, you should be getting the idea that there is a built-in way to do the heavy li ing here! Take a look at the "Brief Tour of the Standard Library" in the docs.

Ans:

```
import sys
import shutil
def main():
    if len(sys.argv) != 2:
        print("Usage: python backup_file.py <file_name>")
        return
    original_file = sys.argv[1]
    try:
        backup_file = f"{original_file}_backup"
        shutil.copy(original_file, backup_file)
        print(f"Backup created successfully: {backup_file}")
    except FileNotFoundError:
        print(f"Error: The file '{original_file}' does not exist.")
    except PermissionError:
```

```
        print(f"Error: Permission denied while accessing '{original_file}'.")
    except Exception as e:
        print(f"An unexpected error occurred: {e}")


if __name__ == "__main__":
    main()
```

Output:

```
C:\Users\ASUS\OneDrive\Desktop\Python Programs\Week-05>python program6.py example6.txt
Backup created successfully: example6.txt_backup
```