Activity:
Exploring a simple Python shell

Task:

Review the blogs at Praka (2018) and Szabo (n.d.) and then create a CLI/ shell that implements the following:

- When you enter the command LIST it lists the contents of the current directory
- The ADD command will add the following two numbers together and provide the result
- The HELP command provides a list of commands available
- The EXIT command exits the shell

Add suitable comments to your code and add the program to your e-portfolio. Be prepared to demonstrate it in the seminar session next week.

Run the shell you have created, try a few commands and then answer the questions below. Be prepared to discuss your answers in the seminar.

- What are the two main security vulnerabilities with your shell?
- What is one recommendation you would make to increase the security of the shell?
- Add a section to your e-portfolio that provides a (pseudo)code example of changes you would make to the shell to improve its security.

Answers:

Question 1 - What are the two main security vulnerabilities with your shell?

Two main security vulnerabilities in this shell:

- Command Injection: The shell doesn't sanitize input, making it potentially vulnerable to injection attacks if expanded to execute system commands
- Directory Traversal: The LIST command doesn't restrict access to the current directory, potentially allowing navigation to sensitive directories

Question 2 - What is one recommendation you would make to increase the security of the shell?

Recommendation to increase security:

- Input Validation and Sanitization: Implement strict input validation for all commands and arguments
- Path Sanitization: Restrict file operations to a specific allowed directory

```
# 1. Add input validation
def validate_input(command, args):
    if not command.isalnum():  # Only allow alphanumeric commands
        raise ValueError("Invalid command format")
    for arg in args:
        if not re.match(r'^[a-zA-Z0-9\.\-_]+$', arg):  # Restrict argument
characters
            raise ValueError("Invalid argument format")

# 2. Add path sanitization
def safe_list_directory(path):
    # Resolve absolute path
    abs_path = os.path.abspath(path)
    # Check if path is within allowed directory
    if not abs_path.startswith(ALLOWED_BASE_DIR):
        raise ValueError("Access denied: Directory outside allowed zone")
    return os.listdir(path)

# 3. Add command rate limiting
def rate_limit_check():
    current_time = time.time()
    if current_time - last_command_time < MIN_COMMAND_INTERVAL:
        raise ValueError("Too many commands too quickly")
```

These improvements would:

1. Prevent command injection by strictly validating input
2. Prevent directory traversal by containing operations within allowed paths
3. Add rate limiting to prevent abuse
4. Add logging for security monitoring (not shown in pseudocode)