Basic Scripting Quiz

Conceptual Questions

1. **Why is error handling important in scripts, and what techniques can you use in Python to handle errors gracefully?**
    - Describe at least three error handling mechanisms in Python and when each should be used.
2. **What are some typical use cases for writing quick Bash/Python scripts in an SRE environment?**
    - Explain how automation scripts can improve reliability and reduce toil.
3. **When working with sensitive credentials in scripts, what security measures should you consider?**
    - List at least four best practices for handling sensitive information in scripts.
4. **Explain the difference between `os.system()`, `subprocess.call()`, and `subprocess.run()` for executing shell commands in Python.**
    - When would you choose one method over the others?
5. **What are context managers in Python (`with` statements), and why are they especially important when working with files?**
    - Provide an example of using a context manager with file operations.

Code Analysis

6. **Review the following code. What issues or potential bugs do you see?**

```python
def process_log_file(filename):
    log_file = open(filename, 'r')
    content = log_file.read()
    errors = []

    for line in content.split('\n'):
        if 'ERROR' in line:
            errors.append(line)

    print(f"Found {len(errors)} errors")
    return errors
```

7. **What will happen when the following script is run? Explain any issues.**

```python
import os
import sys
```

```
def set_config():
    os.environ['APP_ENV'] = 'production'
    print(f"Environment set to {os.environ['APP_ENV']}")

if __name__ == "__main__":
    config_file = sys.argv[1]
    with open(config_file, 'r') as f:
        config = f.read()
    set_config()
    os.system(f"./start_app.sh {config_file}")
```

8. **Analyze the following CSV processing code. What improvements would you make?**

```
import csv

def filter_users(csv_file, min_age):
    with open(csv_file, 'r') as f:
        reader = csv.reader(f)
        header = next(reader)

        age_index = header.index('age')
        result = []

        for row in reader:
            if int(row[age_index]) >= min_age:
                result.append(row)

    with open('filtered_users.csv', 'w') as f:
        writer = csv.writer(f)
        writer.writerow(header)
        writer.writerows(result)
```

Coding Challenges

9. **Write a Python function that takes a log file path and returns a dictionary with counts of different log levels (INFO, WARNING, ERROR, etc.).**
    - The function should handle files that don't exist and properly close file resources.
10. **Create a script that safely updates a JSON configuration file with new values.**
    - The script should preserve existing configuration, update only specified fields, and handle file locking to prevent race conditions.
11. **Implement a function that parses command-line arguments for a script that processes log files.**
    - It should accept parameters for input file, output file, log level filter, and date range.
    - Use `argparse` and include appropriate help text and error handling.
12. **Write a script to monitor a directory for new files and process them when they appear.**

- Include proper error handling and logging.
- Ensure the script doesn't process the same file twice.

Application Questions

13. **As an SRE, you need to create a script that collects system metrics from multiple servers. What approach would you take?**
    - Discuss authentication, data collection, error handling, and output formatting.
14. **You need to automate the deployment of a configuration file to multiple environments (dev, staging, prod). How would you design a script to do this safely?**
    - Address environment-specific configurations, validation, and rollback capabilities.
15. **Describe how you would create a script to automate the rotation of API keys or credentials.**
    - What security considerations and error handling would be important?
    - How would you ensure continuity of service during the rotation?