Practical-13

May 19, 2025

0.1 Practical 13 :-

Name: Sarthak Sanay
Enrollment No: 230031101611051

Suppose we have a text file named config.txt attached.

0.1.1 Problem Statement 1:-

Write a Python code to convert a dictionary containing configuration data that we want to write to a text file.

Example usage:
new_config = {
"username": "new_user",
"password": "new_password",
"database": "new_db",
"host": "localhost",
"port": "5432"}

`config.txt`
username=old_user
password=old_pass
database=old_db
host=localhost
port=8080

[4]:
```python
# read config.txt and convert it into a dictionary
config_dict = {}

with open("config.txt", "r") as file:
    for line in file:
        if "=" in line:
            key, value = line.strip().split("=", 1)
            config_dict[key] = value

print("Config dictionary from file:", config_dict)

# write a dictionary to a file in key:value format
new_config = {
    "username": "new_user",
    "password": "new_password",
    "database": "new_db",
    "host": "localhost",
    "port": "5432"
}

with open("config.txt", "w") as file:
    for key, value in new_config.items():
        file.write(f"{key}={value}\n")

print("New Config dictionary written to config.txt")
```

```
Config dictionary from file: {'username': 'new_user', 'password':
'new_password', 'database': 'new_db', 'host': 'localhost', 'port': '5432'}
New Config dictionary written to config.txt
```

0.1.2 Problem Statement 2:-

Write a Python code to update a specific configuration parameter in an existing text file. You can achieve this by reading the existing configuration, updating the desired parameter, and then writing the updated configuration back to the file.

Example usage:
update_config_parameter('config.txt', 'password', 'new_password123')

[1]:
```
def update_config_parameter(filename, key_to_update, new_value):
    config = {}

    with open(filename, "r") as file:
        for line in file:
            if "=" in line:
                key, value = line.strip().split("=", 1)
                config[key] = value

    # update the desired parameter
    if key_to_update in config:
        config[key_to_update] = new_value
    else:
        print(f"Key '{key_to_update}' not found. Adding it.")
        config[key_to_update] = new_value

    # write updated parameter
    with open(filename, "w") as file:
        for key, value in config.items():
            file.write(f"{key}={value}\n")

    print(f"Updated '{key_to_update}' to '{new_value}' in {filename}")
```

```
filename = input("Enter filename: ")
parameter = input("Enter parameter: ")
value = input("Enter value to update: ")

update_config_parameter(filename, parameter, value)
```

```
Enter filename:  config.txt
Enter parameter:  password
Enter value to update:  new_password123
```

```
Updated 'password' to 'new_password123' in config.txt
```

0.1.3 Problem Statement 3:-

Suppose you have multiple configuration files. Write a Python code to merge them into a single configuration. You can achieve this by reading all the files, merging their configurations, and then writing the merged configuration to a new file.

[2]:
```
import os

config_dir = "configs"
merged_config = {}

for filename in os.listdir(config_dir):
    if filename.endswith(".txt"):
        filepath = os.path.join(config_dir, filename)
        with open(filepath, "r") as file:
            for line in file:
                if "=" in line:
                    key, value = line.strip().split("=", 1)
                    merged_config[key] = value  # Later files will override␣
 ↪earlier ones

with open("merged_config.txt", "w") as file:
    for key, value in merged_config.items():
        file.write(f"{key}={value}\n")

print("Merged configuration written to merged_config.txt")
```

```
Merged configuration written to merged_config.txt
```

0.1.4 Problem Statement 4:-

Write a Python code to delete a specific configuration parameter from a text file. You can achieve this by reading the existing configuration, removing the desired parameter, and then writing the updated configuration back to the file.

Example usage:
delete_config_parameter('config.txt', 'port')

```
[3]: def delete_config_parameter(filename, key_to_delete):
         config = {}

         with open(filename, "r") as file:
             for line in file:
                 if "=" in line:
                     key, value = line.strip().split("=", 1)
                     config[key] = value

         if key_to_delete in config:
             del config[key_to_delete]
             print(f"Deleted '{key_to_delete}' from {filename}")
         else:
             print(f"Key '{key_to_delete}' not found in {filename}")

         with open(filename, "w") as file:
             for key, value in config.items():
                 file.write(f"{key}={value}\n")


     filename = input("Enter filename: ")
     parameter = input("Enter parameter to delete: ")

     delete_config_parameter(filename, parameter)
```

```
Enter filename:  config.txt
Enter parameter to delete:  port

Deleted 'port' from config.txt
```

```
[ ]:
```