Files

Solve the following exercises and upload your solutions to Moodle until the specified due date. Make sure to use the *exact filenames* that are specified for each individual exercise. Unless explicitly stated otherwise, you can assume correct user input and correct arguments. You are *not allowed* to use any concepts and modules that have not yet been presented in the lecture.

Exercise 1 – Submission: `a6_ex1.py` 20 Points

Write a function `file_statistics(path: str)` that prints the statistics of a given file as follows:

- `path` specifies the path to a textfile.
- Use the `utf-8` encoding when reading the file.
- The function should print the number of *lines*, *words*, *characters*, and *digits* of that file.
- If `path` does not specify a correct path, an `OSError` should be raised that notifies the user of an incorrect path.
- If `path` is a correct path but not a .txt file, a `ValueError` should be raised that notifies the user of the wrong file type.
- You can test whether you receive the expected results with the given files `ex1_1.txt`, `ex1_2.txt`, `ex1_3.txt` and `ex1_4.py`.

Example program executions:

```
file_statistics('ex1_1.txt')
--------------------
Statistics of file ex1_1.txt:
Number of lines: 79
Number of words: 3578
Number of characters: 21966
Number of digits: 0
--------------------

file_statistics('ex1_2.txt')
--------------------
Statistics of file ex1_2.txt:
Number of lines: 612
Number of words: 22308
Number of characters: 140742
Number of digits: 193
--------------------

file_statistics('ex1_3.txt')
--------------------
Statistics of file ex1_3.txt:
Number of lines: 124453
Number of words: 901325
Number of characters: 5458195
Number of digits: 3190
--------------------
```

```
file_statistics('ex1_4.py')
ValueError: Path ex1_4.py is not a text file

file_statistics('ex1_5.txt')
OSError: Path ex1_5.txt does not exist
```

Hints:

- Words can be identified by being separated with spaces.
- Everything except line endings counts as a character (including digits and spaces, excluding \r and \n).

Exercise 2 – Submission: `a6_ex2.py` 30 Points

Write a generator function `chunks(path: str, size: int, **kwargs)` that yields data chunks of size `size` from the file specified by `path`. If `path` is not a file, a `ValueError` must be raised. If `size` is smaller than 1, a `ValueError` must be raised. `**kwargs` are (optional) keyword arguments that are passed to the built-in `open` function (which your function must use internally). Note that there can be smaller chunks than `size`, e.g., if the file is too short or if the file content length is not evenly divisible by `size`. You are *not allowed* to read the entire file content at once, i.e., make sure to also implement a chunked file reading operation.

Example function call (with file `ex2_example.txt`, see exercise above):

```
for i, c in enumerate(chunks("ex2_example.txt", 25, mode="rb")):
    print(f"Chunk {i} = {c}")
```

Example output:

```
Chunk 0 = b'12 w 21  d23g780nb deed e'
Chunk 1 = b'2 21.87\r\n43 91 - . 222 mf'
Chunk 2 = b'tg 21 bx .1 3 g d e 6 de '
Chunk 3 = b'ddd32 3412\r\n0.3 0 0. 0 0 '
Chunk 4 = b'1\r\n\r\n70\r\n\r\nn 12 1    9   '
Chunk 5 = b' m1 1m 445\r\nx 100\r\n'
```

Hints:

- Use the `os` module to check if `path` is a file.

Exercise 3 – Submission: `a6_ex3.py` 25 Points

Write a function

```
merge_csv_files(
        *paths: str,
        delimiter = ';': str,
        only_shared_columns = False : bool)
```

that merges an arbitrary number of csv files into a single one.

- `*paths` is an arbitrary number of paths to csv files (you can assume it is at least one).
- `delimiter` specifies how the values inside the csv files are separated.
- `only_shared_columns` changes which columns should be merged. If `True`, the merged file should only have columns that exist in all initial files. If `False`, all columns shall be used and missing values should be filled with `NaN`.
- The merged csv file should be named `merged.csv` and saved to the same directory as all initial files. See the initial files `ex3_1.csv`, `ex3_2.csv`, `ex3_3.csv` and the resulting merged files `merged_all.csv` and `merged_shared.csv` for the expected results of two calls of this function with all files and `only_shared_columns = False` and `only_shared_columns = True` respectively.

Hints:

- Make sure to use the same delimiter for the input and output files.
- Check that the merged file includes the column names and does not have extra lines.
- The columns can be ordered arbitrarily in input and output.

Exercise 4 – Submission: `a6_ex4.py` 25 Points

Write the functions

- `create_user(name: str, data: str, password:str, log_file = 'logs.txt': str)`
- `login(name: str, password: str, log_file = 'logs.txt': str)`
- `change_password(name: str, old_password: str, new_password: str, log_file = 'logs.txt': str)`

that manage a naive user access process.

- `create_user` creates a new user for the server. The user data consists of `name`, `data`, and `password`, as well as the current time in a parameter `last_changed`, and should be saved to a pickled file that is named after the user. Afterwards, a line should be added to the log file that says that a new user was created at the current time. If an attempt is made to create a user that already exists, i.e. name.pkl already exists, it should be added to the log file that an attempt was made to create an already existing user with the given name and current time.
- In `login`, first, an attempted login for a given user and the current time should be added to the log file. Then, the given password should be compared to the one saved in the user.pkl file. If they are the same, a login successful line with the current name and time should be added to the log file, as well as returning the `data` of the user. If they are not the same, a login failed line with the current name and time should be added to the log file and None should be returned. If an attempt is made to log into a user that does not exist, i.e. there is no name.pkl, this should be logged to the log file with the given name and current date and None should be returned.
- `change_password` first logs that an attempt at changing the password of the user was made at the current time. Then, the password of the user saved in user.pkl is compared to `old_password`. If they are the same, the password is changed to `new_password` and `last_changed` is updated to the current time, as well as logging the successful password change for the current user and time. If they are not the same, a failed change is added to the log file with the current user and time but no change is made to the user.

To check for current syntax, execute the script below and check the state of your files against the given files `Franz Kafka.pkl`, `George Orwell.pkl`, `H.P. Lovecraft.pkl`, `William Golding.pkl`, and `logs.txt`.

```
create_user('Franz Kafka', 'Die Verwandlung', 'fkafka')
create_user('H. P. Lovecraft', 'The Call of Cthulhu', 'lcrft')
create_user('William Golding', 'Lord of the Flies', 'password')
create_user('George Orwell', '1984', 'orwell1948')
login('Franz Kafka', 'fkafks')
login('Franz Kafka', 'fkafka')
login('H. P. Lovecraft', 'lcrft')
login('William Golding', 'password')
change_password('William Golding', 'password', 'wigold')
login('George Orwell', 'orwell1984')
login('George Orwell', 'orwell1948')
change_password('George Orwell', 'orwell1984', 'orwell1984')
change_password('George Orwell', 'orwell1948', 'orwell1984')
login('George Orwel', 'orwell1984')
create_user('George Orwell', '1984', 'orwell1984')
```

Hints:

- You can use the `datetime` module to get the current time.