Python Developer Coding Test

Instructions

1. 2 days (48 hours) from the time you receive the test
2. Tools: Python 3.x, any IDE/editor of your choice
3. Submission: Provide your code files and any additional documentation. (follow the instructions given at the end of the document)

Problem 1: Reverse a String
Write a function that takes a string as input and returns the string reversed.

Example :

```python
def reverse_string(s: str) -> str:
    pass  # Your code here

# Example
print(reverse_string("hello"))  # Output: "olleh"
```

Problem 2: List Comprehensions
Given a list of integers, write a function to return a list of squares of even numbers only.

Example :

```python
def square_evens(numbers: list) -> list:
    pass  # Your code here

# Example
print(square_evens([1, 2, 3, 4, 5]))  # Output: [4, 16]
```

Problem 3: File I/O
Write a function that reads a text file, counts the number of words in it, and returns the count.

```python
def count_words_in_file(filename: str) -> int:
    pass  # Your code here

# Example
# Assume 'sample.txt' contains "Hello world"
print(count_words_in_file("sample.txt"))  # Output: 2
```

Problem 4: Dictionary Manipulation
Write a function that takes a dictionary with string keys and integer values and returns a dictionary sorted by the values.

```python
def sort_dict_by_values(d: dict) -> dict:
    pass  # Your code here

# Example
print(sort_dict_by_values({"a": 2, "b": 1, "c": 3}))  # Output: {"b": 1, "a": 2, "c": 3}
```

Problem 5 : Bank Account Management

You need to create functions that allow a user to manage a bank account by entering values and saving these values in a text file. The functionalities should allow the user to enter the bank account number, then read the text file and proceed for deposits and withdrawals.

Instructions

1. **File:** Assume the file `accounts.txt` stores account information in the format: `account_number,balance`.
2. **Functions:** Implement the following functions:
    - **get_balance**(account_number: str) -> float : This function should take an account number as input and return the current balance.
    - **deposit**(account_number: str, amount: float) : This function should take an account number and an amount, then update the balance in the file.

- **withdraw**(account_number: str, amount: float): This function should take an account number and an amount, then update the balance in the file. Ensure the balance does not go negative.
- **update_account**(account_number: str, balance: float): This function should update the balance of an account in the file.

3. **Testing**
   To test these functions, create an accounts.txt file with some initial data, and then call the functions

Evaluation Criteria

1. **Correctness:** The functions should handle reading, writing, and updating the text file correctly.
2. **Efficiency:** The solution should efficiently handle file operations.
3. **Readability:** The code should be clear and well-documented.
4. **Robustness:** Ensure the functions handle edge cases, such as non-existent accounts and insufficient balance for withdrawals.

Instructions to submit your test

Initially you need to create a public repository in your GitHub account. Commit the separate python files to the repository for each question. Share the repository URL with us.