Code

txstc55

May 20, 2024

```
import os
import sys
import json
import random
from datetime import datetime

# Decorator for logging function execution time
def log_execution_time(func):
    def wrapper(*args, **kwargs):
        start_time = datetime.now()
        result = func(*args, **kwargs)
        end_time = datetime.now()
        execution_time = end_time - start_time
        print(f"Function {func.__name__} executed in {execution_time}")
        return result
    return wrapper

# A class representing a simple bank account
class BankAccount:
    def __init__(self, account_holder, balance=0):
        self.account_holder = account_holder
        self.balance = balance
        self.transaction_history = []

    def deposit(self, amount):
        self.balance += amount
        self.transaction_history.append((datetime.now(), f"Deposited {amount}"))
        return self.balance

    def withdraw(self, amount):
        if amount > self.balance:
            raise ValueError("Insufficient funds")
        self.balance -= amount
        self.transaction_history.append((datetime.now(), f"Withdrew {amount}"))
        return self.balance

    def get_balance(self):
        return self.balance

    def get_transaction_history(self):
        return self.transaction_history

    def __str__(self):
        return f"Account holder: {self.account_holder}, Balance: {self.balance}"

# A function demonstrating various Python features
@log_execution_time
def demonstrate_python_features():
    print("Demonstrating various Python features:")

    # Lists and list comprehensions
    numbers = [random.randint(1, 100) for _ in range(10)]
    print(f"Random numbers: {numbers}")

    even_numbers = [num for num in numbers if num % 2 == 0]
    print(f"Even numbers: {even_numbers}")

    # Dictionary comprehension
    number_square_dict = {num: num ** 2 for num in numbers}
    print(f"Number squares: {number_square_dict}")

    # String formatting
    formatted_string = f"Formatted string with random number: {random.choice(numbers)}"
    print(formatted_string)

    # Exception handling
    try:
        result = 10 / random.choice([0, 1, 2])
        print(f"Division result: {result}")
    except ZeroDivisionError as e:
        print(f"Error occurred: {e}")

    # File I/O
    file_path = "example.txt"
    with open(file_path, 'w') as file:
        file.write("This is an example file.\n")
        file.write(f"Random numbers: {numbers}\n")
    with open(file_path, 'r') as file:
        file_content = file.read()
        print(f"File content:\n{file_content}")

    # JSON handling
    json_data = json.dumps(number_square_dict, indent=4)
    print(f"JSON data: {json_data}")

    # Creating a bank account object
    account = BankAccount("John Doe", 1000)
    print(account)
```