Pair Exercise #3

Wallace Chipidza

June 2023

Due: Write your code in a file called **pe3.py** and push it to a Github repository named **303_summer_23**. Submit a link to the Github repository by 3:59PM on Friday, 6/16/23. Link on Canvas. The exercise will be graded out of 50 according to 25 tests that it should pass for full credit.

The file **test_pe3.py** has the tests that I will evaluate your code on. To run the tests against your code, follow these instructions:

(i) Download **test_pe3.py** into the same directory that has your **pe3.py**
(ii) **pytest.ini** has some configuratioon parameters that allows you to run the tests without some annoying warnings; please download it too.
(iii) In the terminal, run the following command: **pip install pytest**.
(iv) In the terminal, run the following command: **pytest -v test_pe3.py**.

1 Functions

Caesar's cipher is a cryptography scheme attributed to the Roman Emperor Julian Caesar who used it to communicate with his associates while aiming to prevent eavesdropping. In its simplest form, the cipher uses a substitution mechanism i.e., each letter in the communicated text is replaced by another, unique letter. The substitution is specified by a shift parameter, **D**, which denotes how many positions down the alphabet the replacing letter resides. More details on how the cipher works can be found in this Wikipedia article: `https://en.wikipedia.org/wiki/Caesar_cipher`

In this part of the exercise, you will create two functions, i.e., an encoding and a decoding function.

a) Create an encoding function called **encode**, which takes 2 arguments: (i) **input_text** i.e., the text to be encrypted and (ii) **shift** i.e., the number of places down the alphabet where the replacing letter resides. The function returns a tuple of two items: (i) a list of letters of the English alphabet in lowercase and (ii) the encoded text.

Examples of function calls of function **encode** and expected outputs:

```
encode("a", 3) #should return (["a", "b", ... "z"], "d")
encode("abc", 4) #should return (["a", "b", ... "z"], "efg")
encode("xyz", 3) #should return (["a", "b", ... "z"], "abc")
encode("j!K,2?", 3) #should return (["a", "b", ... "z"], "m!n,2?")
```

b) Create a decoding function called **decode**, which takes 2 arguments: (i) **input_text** i.e., the text to be decrypted and (ii) **shift** i.e., the number of places up the alphabet where the replacing letter is. The function returns the decoded text.

Examples of function calls of function **decode** and expected outputs:

```
decode("d", 3) #should return "a"
decode("efg", 4) #should return "abc"
decode("abc", 3) #should return "xyz"
decode("m!n,2?", 3) #should return "j!K,2?"
```

2 Classes

Create a class called BankAccount.

a) The class is initialized with the owner's name (name), alphanumeric ID (ID), date of creation (creation_date), and balance. Ensure that creation_date is of type datetime.date. The date of creation can be the current date or a past date, but not a date in the future. If a creation date in the future is supplied as the creation_date value, the program must raise an exception of class **Exception**.

b) It has 3 instance methods: **deposit(amount)**, **withdraw(amount)**, and **view_balance()**.

c) There are two subclasses to BankAccount: a SavingsAccount and a CheckingAccount.
- SavingsAccount: only allows withdrawals after 6 months has passed since the creation of the account. Does not allow overdrafts. For simplicity assume that each month is 30 days long.
- CheckingAccount: allows overdraft, but adds a $30 penalty fee each time a withdrawal results in a negative balance.

d) Make sure your names are precisely consistent with the following:
- Class names - BankAccount, CheckingAccount, SavingsAccount
- Method Names - deposit(self, amount), withdraw(self, amount), view_balance(self)

e) The BankAccount class should have default values for all the arguments as follows: name: "Clocks", id: "123", creation_date: date.today(), balance: 0.
- The creation_date should be of type datetime.date