Do You Know? Intermediate Programming with Python

The following programming projects use the concepts taught in our **Intermediate Programming with Python** course. If you can implement both of the following programs with little difficulty, then the class **Intermediate Programming with Python** would largely serve as a review for you.

Note 1: This class uses the Python programming language. If you can solve these problems using a different object-oriented programming language, then you may be better served by learning Python on your own, as the course covers programming concepts that you would likely find review.

Note 2: The problems below are fairly routine exercises to test your understanding of the concepts covered in the course. The projects that we will do in the course will be more involved and (hopefully!) more interesting than the exercises below. Even if you can solve the exercises below, you might enjoy working on larger programming projects and may still be interested in this course, even though much of the material will likely be review for you.

1. Write a class called `BankAccount` representing a bank account. An account has an ID number and a value representing the balance. Write methods to:
   (a) Initialize the account: it should take the ID number and set the balance to 0.
   (b) Return the current balance.
   (c) Deposit money to the account: it should take the amount to add as a parameter.
   (d) Withdraw money from the account: it should take the amount to subtract as a parameter; if that amount would make the balance go negative, don't perform the withdrawal.
   (e) Define `__str__` to allow the account to be printed via `print()`: the string should include the ID number.
   (f) Transfer money from one account to another: that is, if `a1` and `a2` are `BankAccount`s, the method `a1.transfer(a2,amt)` should remove `amt` dollars from `a1` and add them to `a2` (assuming there are at least `amt` dollars in `a1`; if not, do nothing).

   Also write a subclass called `SavingsAccount` representing a savings account. A savings account has an additional attribute representing the interest rate of the account. (For example, an account paying 5% interest would have its interest rate be `0.05`.) The subclass should have an additional method that adds interest to the account, which is computed by multiplying the interest rate by the current balance.
2. Write a GUI-based program with a text box, a label, and a button labeled "Flip!". If the button is clicked, any text that appears in the text box should appear backwards in the label. For example, if I enter `python` and click "Flip!", `nohtyp` should appear in the label.

Don't look at the next page until you've attempted all the problems!

Sample solutions to the programming projects from *Do You Need Intermediate Programming with Python* are below. These are only samples and there are many different possible approaches. All of the techniques used in the programs below will be covered during the class. (Also note that these programs are written in Python 3. If you are using Python 2, there may be slight differences in your solutions.)

1.
```
class BankAccount:
    """A class representing a bank account."""
    def __init__(self, idNum):
        """Initializes the account with the specified ID number."""
        self.__id = idNum
        self.__balance = 0
    def __str__(self):
        """Returns a description of the account as a string."""
        return "Account #"+str(self.__id)+" has a balance of $"+str(self.__balance)
    def get_balance(self):
        """Returns the current balance."""
        return self.__balance
    def deposit(self, amount):
        """Deposits the specified amount into the account."""
        self.__balance += amount
    def withdraw(self, amount):
        """Withdraws the specified amount from the account. If there isn't enough
        money in the account the withdrawal won't be performed. Returns whether
        or not the withdrawal was successful (i.e. if there were enough funds)."""
        if amount <= self.__balance:
            self.__balance -= amount
            return True
        else: # not enough money
            return False
    def transfer(self, other, amount):
        """Transfers the specified amount of money from this account to other.
        If this account has insufficient funds the transfer will not be performed.
        Returns whether or not the transfer was successful."""
        if amount <= self.__balance:
            self.__balance -= amount
            other.__balance += amount
            return True
        else: # not enough money
            return False
```



```
class SavingsAccount(BankAccount):
    """A subclass of BankAccount representing a savings account"""
    def __init__(self, idNum, interestRate):
        """Initializes as a BankAccount, with additional interestRate"""
        BankAccount.__init__(self, idNum)
        self.__interestRate = interestRate
    def add_interest(self):
        """Adds interest"""
        self.deposit(self.get_balance() * self.__interestRate)
```

2. Note: it's possible to use many different GUI frameworks to solve this problem. The GUI framework used in the class, and in the solution below, is `tkinter`.

```
from tkinter import *
class FlipButtonDemo(Frame):
    """window with a text box, a label, and a button.
    Pressing the button puts the reverse of the text in the label"""
    def __init__(self):
        """Window setup"""
        Frame.__init__(self)
        self.master.title("Word Flip!")
        self.grid()
        # text box
        self.__text1 = Entry(self)
        self.__text1.grid(row=0, column=0)
        # text label
        self.__label1 = Label(self, text="")
        self.__label1.grid(row=1, column=0)
        # button
        self.__button1 = Button(self, text="Flip!",command=self._flip)
        self.__button1.grid(row=2, column=0)

    def _flip(self):
        """reverses the text in the box and puts it in the label"""
        word = self.__text1.get()
        self.__label1 = Label(self, text=word[::-1])
        self.__label1.grid(row=1, column=0)

def main():
    """initialize the window and wait for events"""
    demo = FlipButtonDemo()
    demo.mainloop()
```