Module 4

In this assignment, you will implement an online banking system. Users can sign-up with the system, log in to the system, change their password, and delete their account. They can also update their bank account balance and transfer money to another user's bank account.

You'll implement functions related to File I/O and dictionaries. Two of the functions require you to import files and create dictionaries. User information will be imported from the "users.txt" file and account information will be imported from the "bank.txt" file. Take a look at the content in the different files. The remaining functions require you to use or modify the two dictionaries created from the files.

Each function has been defined for you, but without the code. See the docstring in each function for instructions on what the function is supposed to do and how to write the code. It should be clear enough. In some cases, we have provided hints to help you get started.

In [ ]:
```
#########################################################
### EXECUTE THIS CELL BEFORE YOU TO TEST YOUR SOLUTIONS ###
#########################################################

import nose.tools as tools
```

In [ ]:
```
def import_and_create_bank(filename):
    '''
    This function is used to create a bank dictionary.  The given argument is the filename to load.
    Every line in the file should be in the following format:
        key: value
    The key is a user's name and the value is an amount to update the user's bank account with.  The value should be a
    number, however, it is possible that there is no value or that the value is an invalid number.

    What you will do:
    - Create an empty bank dictionary.
    - Read in the file.
    - Add keys and values to the dictionary from the contents of the file.
    - If the key doesn't exist in the dictionary, create a new key:value pair.
    - If the key does exist in the dictionary, increment its value with the amount.
    - You should also handle the following cases:
    -- When the value is missing or invalid.  If so, ignore that line and don't update the dictionary.
    -- When the line is completely blank.  Again, ignore that line and don't update the dictionary.
    -- When there is whitespace at the beginning or end of a line and/or between the name and value on a line.  You
    should trim any and all whitespace.
    - Return the bank dictionary from this function.

    For example, here's how your code should handle some specific lines in the file:
    The 1st line in the file has a name and valid number:
        Brandon: 5
    Your code will process this line and add the extracted information to the dictionary.  After it does,
    the dictionary will look like this:
        bank = {"Brandon": 5}

    The 2nd line in the file also has a name and valid number:
        Patrick: 18.9
    Your code will also process this line and add the extracted information to the dictionary.  After it does,
    the dictionary will look like this:
        bank = {"Brandon": 5, "Patrick": 18.9}

    The 3rd line in the file has a name but invalid number:
        Brandon: xyz
    Your code will ignore this line and add nothing to the dictionary.  It will still look like this:
        bank = {"Brandon": 5, "Patrick": 18.9}

    The 4th line in the file has a name but missing number:
        Jack:
    Your code will ignore this line and add nothing to the dictionary.  It will still look like this:
        bank = {"Brandon": 5, "Patrick": 18.9}

    The 5th line in the file is completely blank.
    Your code will ignore this line and add nothing to the dictionary.  It will still look like this:
        bank = {"Brandon": 5, "Patrick": 18.9}

    The 8th line in the file has a name and valid number, but with extra whitespace:
        Brandon:       10
    Your code will process this line and update the value associated with the existing key ('Brandon') in the dictionary.
    After it does, the value associated with the key 'Brandon' will be 10:
        bank = {"Brandon": 15, ...}

    After processing every line in the file, the dictionary will look like this:
        bank = {"Brandon": 115.5, "Patrick": 18.9, "Sarah": 827.43, "Jack": 45.0, "James": 128.87}
    Return the dictionary from this function.
    '''

    bank = {}

    # your code here

    with open(filename, "r") as file:

        for line in file:
            #print(i)
            name = line.rstrip('\n').split(":")
            #(name, amt) = line.rstrip('\n').split(":")
            zapis = True

            if (len(name)) ==2:
                name2=name[0].strip()

                amt1=name[1].strip()
                # using isdigit() + replace()
                # Check for float string
                res = amt1.replace('.', '', 1).isdigit()

                # print result
                if res :
                    #print("Is string a possible float number ? : " + str(res))
                    amt2=amt1
                else:
                    amt2=0
                    zapis = False
                #print(type(amt1))
            else:
                name2=name[0].strip()
                zapis = False
            #print(name2)
            #print(amt2)
            key=name2
            value=float(amt2)
            #print(amt)
            if zapis:
                bank[key] = bank.get(key, 0) + value

            #i += 1

    return bank
```

In [ ]:
```
##########################
### TEST YOUR SOLUTION ###
##########################
bank = import_and_create_bank("bank.txt")

tools.assert_false(len(bank) == 0)
tools.assert_almost_equal(115.5, bank.get("Brandon"))
tools.assert_almost_equal(128.87, bank.get("James"))
tools.assert_is_none(bank.get("Joel"))
tools.assert_is_none(bank.get("Luke"))
tools.assert_almost_equal(bank.get("Sarah"), 827.43)
print("Success!")
```

In [ ]:
```
def signup(user_accounts, log_in, username, password):
    '''
    This function allows users to sign up.
    If both username and password meet the requirements:
    - Updates the username and the corresponding password in the user_accounts dictionary.
    - Updates the log_in dictionary, setting the value to False.
    - Returns True.

    If the username and password fail to meet any one of the following requirements, returns False.
    - The username already exists in the user_accounts.
    - The password must be at least 8 characters.
    - The password must contain at least one lowercase character.
    - The password must contain at least one uppercase character.
    - The password must contain at least one number.
    - The username & password cannot be the same.

    For example:
    - Calling signup(user_accounts, log_in, "Brandon", "123abcABCD") will return False
    - Calling signup(user_accounts, log_in, "BrandonK", "123ABCD") will return False
    - Calling signup(user_accounts, log_in, "BrandonK","abcdABCD") will return False
    - Calling signup(user_accounts, log_in, "BrandonK", "123aABCD") will return True. Then calling
    signup(user_accounts, log_in, "BrandonK", "123aABCD") again will return False.

    Hint: Think about defining and using a separate valid(password) function that checks the validity of a given password.
    This will also come in handy when writing the change_password() function.
    '''

    # your code here

    zapis = 0
    for keys in user_accounts.keys():
        print(keys)
        print("Timto jmenem se prihlasujes ", username)
        if (keys == username):
            print("Toto jmeno uz je ulozeno")
            zapis = 1

    if (validate(username, password) and zapis == 0):
        print("Probehne zapis")
        user_accounts.update({username : password} )
        log_in.update({username : "False"})
        return True
    else:
        print("Uz zapsane jmeno, nevalidni heslo")
        return False
```

In [ ]:
```python
def import_and_create_accounts(filename):
    '''
    This function is used to create an user accounts dictionary and another login dictionary.  The given argument is the
    filename to load.
    Every line in the file should be in the following format:
      username - password
    The key is a username and the value is a password.  If the username and password fulfills the requirements,
    add the username and password into the user accounts dictionary.  To make sure that the password fulfills these
    requirements, be sure to use the signup function that you wrote above.

    For the login dictionary, the key is the username, and its value indicates whether the user is logged in, or not.
    Initially, all users are not logged in.

    What you will do:
    - Create an empty user accounts dictionary and an empty login dictionary.
    - Read in the file.
    - If the username and password fulfills the requirements, adds the username and password
    into the user accounts dictionary, and updates the login dictionary.
    - You should also handle the following cases:
    -- When the password is missing.  If so, ignore that line and don't update the dictionaries.
    -- When there is whitespace at the beginning or end of a line and/or between the name and password on a line.  You
    should trim any and all whitespace.
    - Return both the user accounts dictionary and login dictionary from this function.

    For example, here's how your code should handle some specific lines in the file:
    The 1st line in the file has a name and password:
      Brandon - brandon123ABC
    Your code will process this line, and using the signup function, will add the extracted information to the
    dictionaries.  After it does, the dictionaries will look like this:
      user_accounts = {"Brandon": "brandon123ABC"}
      log_in = {"Brandon": False}

    The 2nd line in the file has a name but missing password:
      Jack
    Your code will ignore this line.  The dictionaries will still look like this:
      user_accounts = {"Brandon": "brandon123ABC"}
      log_in = {"Brandon": False}

    The 3rd line in the file has a name and password:
      Jack - jac123
    Your code will process this line, and using the signup function, will not add the extracted information to the
    dictionaries because the password is invalid.  The dictionaries will still look like this:
      user_accounts = {"Brandon": "brandon123ABC"}
      log_in = {"Brandon": False}

    The 4th line in the file has a name and password:
      Jack - jack123POU
    Your code will process this line, and using the signup function, will add the extracted information to the
    dictionaries.  After it does, the dictionaries will look like this:
      user_accounts = {"Brandon": "brandon123ABC, "Jack": "jack123POU"}
      log_in = {"Brandon": False, "Jack": False}

    After processing every line in the file, the dictionaries will look like this:
      user_accounts = {"Brandon": "brandon123ABC, "Jack": "jack123POU", "James": "100jamesABD", "Sarah": "sd896ssfJJH"}
      log_in = {"Brandon": False, "Jack": False, "James": False, "Sarah": False}
    Return the dictionaries from this function.
    '''

    user_accounts = {}
    log_in = {}

    # your code here

    with open(filename, "r") as file:

        for line in file:
            #print(i)
            name = line.rstrip('\n').split("-")
            #(name, amt) = line.rstrip('\n').split(":")
            zapis = True

            if (len(name)) ==2:
                name2=name[0].strip()

                for keys in user_accounts:
                    if (keys == name2):
                        print("Jsem nasel stejne jmeno")
                        zapis = False



                password=name[1].strip()
                #print("name2: "+name2)
                #print("password: "+password)
                # Validation of password
                res=validate(name2,password)
                if res:
                    #print(password)
                    value=password
                else:
                    zapis = False
                #print(type(amt1))
            else:
                name2=name[0].strip()
                zapis = False
            #print(name2)
            #print(amt2)
            key=name2
            print(value)
            if zapis:
                user_accounts.update({key : value} )
                log_in.update({key : "False"})

            #i += 1
    print(user_accounts)


    return user_accounts,log_in

def validate(username, password):
        l, u, d = 0, 0, 0
        if (len(password) < 7 ):
            result = False
        for i in password:
            # counting lowercase alphabets
            if (i.islower()):
                l+=1
                # counting uppercase alphabets
            if (i.isupper()):
                u+=1
                # counting digits
            if (i.isdigit()):
                d+=1
        if (l>=1 and u>=1 and d>=1 and l+u+d==len(password)) and (username != password):
            result = True
        else:
            result = False

        return result
```

In [ ]:
```python
##########################
### TEST YOUR SOLUTION ###
##########################
user_accounts, log_in = import_and_create_accounts("user.txt")

tools.assert_false(len(user_accounts) == 0)
tools.assert_false(len(log_in) == 0)
tools.assert_equal(user_accounts.get("Brandon"),"brandon123ABC")
tools.assert_equal(user_accounts.get("Jack"),"jack123POU")
tools.assert_is_none(user_accounts.get("Jennie"))
tools.assert_false(log_in["Sarah"])
print("Success!")
```

In [ ]:
```python
##########################
### TEST YOUR SOLUTION ###
##########################
bank = import_and_create_bank("bank.txt")
user_accounts, log_in = import_and_create_accounts("user.txt")

tools.assert_false(signup(user_accounts,log_in,"Brandon","123abcABCD"))

tools.assert_false(signup(user_accounts,log_in,"BrandonK","123ABCD"))
tools.assert_false(signup(user_accounts,log_in,"BrandonK","1234ABCD"))
tools.assert_false(signup(user_accounts,log_in,"BrandonK","abcdABCD"))
tools.assert_false(signup(user_accounts,log_in,"BrandonK","1234abcd"))

tools.assert_false(signup(user_accounts,log_in,"123abcABCD","123abcABCD"))

tools.assert_true(signup(user_accounts,log_in,"BrandonK","123aABCD"))
tools.assert_false(signup(user_accounts,log_in,"BrandonK","123aABCD"))
tools.assert_true("BrandonK" in user_accounts)
tools.assert_equal("123aABCD",user_accounts["BrandonK"])
tools.assert_false(log_in["BrandonK"])
print("Success!")
```

In [ ]:
```python
def login(user_accounts, log_in, username, password):
    '''
    This function allows users to log in with their username and password.
    The user_accounts dictionary stores the username and associated password.
    The log_in dictionary stores the username and associated log-in status.

    If the username does not exist in user_accounts or the password is incorrect:
    - Returns False.
    Otherwise:
    - Updates the user's log-in status in the log_in dictionary, setting the value to True.
    - Returns True.

    For example:
    - Calling login(user_accounts, "Brandon", "123abcAB") will return False
    - Calling login(user_accounts, "Brandon", "brandon123ABC") will return True
    '''

    # your code here

    for keys in user_accounts:
        if (keys == username):
            #print("Jsem nasel jmeno")
            #print(user_accounts[keys])
            #print(password)
            if (user_accounts[keys] == password):
                #print("Jsem nasel jmeno a heslo")
                log_in[keys]='True'
                return True
            else:
                #print("Jsem nasel jmeno bez hesla")
                return False
        else:
            return False

    #print("Tamto: ", log_in)

    return True
```

In [ ]:
```python
##########################
### TEST YOUR SOLUTION ###
##########################
bank = import_and_create_bank("bank.txt")
user_accounts, log_in = import_and_create_accounts("user.txt")

tools.assert_false(login(user_accounts, log_in,"Brandon","123abcAB"))
tools.assert_true(login(user_accounts, log_in,"Brandon","brandon123ABC"))
tools.assert_false(login(user_accounts, log_in,"BrandonK","123abcABC"))
print("Success!")
```

In [ ]:
```python
def update(bank, log_in, username, amount):
    '''
    In this function, you will try to update the given user's bank account with the given amount.
    bank is a dictionary where the key is the username and the value is the user's account balance.
    log_in is a dictionary where the key is the username and the value is the user's log-in status.
    amount is the amount to update with, and can either be positive or negative.

    To update the user's account with the amount, the following requirements must be met:
    - The user exists in log_in and his/her status is True, meaning, the user is logged in.

    If the user doesn't exist in the bank, create the user.
    - The given amount can not result in a negative balance in the bank account.

    Return True if the user's account was updated.

    For example, if Brandon has 115.50 in his account:
    - Calling update(bank, log_in, "Brandon", 50) will return False, unless "Brandon" is first logged in.  Then it
    will return True.  Brandon will then have 165.50 in his account.
    - Calling update(bank, log_in, "Brandon", -200) will return False because Brandon does not have enough in his
    account.
    '''

    # your code here
```

In [ ]:
```python
##########################
### TEST YOUR SOLUTION ###
##########################
bank = import_and_create_bank("bank.txt")
user_accounts, log_in = import_and_create_accounts("user.txt")

tools.assert_false(update(bank,log_in,"Jack",100))
login(user_accounts, log_in, "Brandon", "brandon123ABC")
tools.assert_false(update(bank,log_in,"Brandon",-400))
tools.assert_true(update(bank,log_in,"Brandon",100))
tools.assert_almost_equal(bank.get("Brandon"),215.5)

signup(user_accounts, log_in, "BrandonK", "123aABCD")
tools.assert_is_none(bank.get("BrandonK"))
login(user_accounts,log_in,"BrandonK","123aABCD")
tools.assert_true(update(bank,log_in,"BrandonK",100))
tools.assert_almost_equal(bank.get("BrandonK"),100)
print("Success!")
```

In [ ]:
```python
def transfer(bank, log_in, userA, userB, amount):
    '''
    In this function, you will try to make a transfer between two user accounts.
    bank is a dictionary where the key is the username and the value is the user's account balance.
    log_in is a dictionary where the key is the username and the value is the user's log-in status.
    amount is the amount to be transferred between user accounts (userA and userB).  amount is always positive.

    What you will do:
    - Deduct the given amount from userA and add it to userB, which makes a transfer.
    - You should consider some following cases:
      - userA must be in the bank and his/her log-in status in log_in must be True.
      - userB must be in log_in, regardless of log-in status.  userB can be absent in the bank.
      - No user can have a negative amount in their account. He/she must have a positive or zero balance.

    Return True if a transfer is made.

    For example:
    - Calling transfer(bank, log_in, "BrandonK", "Jack", 100) will return False
    - Calling transfer(bank, log_in, "Brandon", "JackC", 100) will return False
    - After logging "Brandon" in, calling transfer(bank, log_in, "Brandon", "Jack", 10) will return True
    - Calling transfer(bank, log_in, "Brandon", "Jack", 200) will return False
    '''

    # your code here
```

In [ ]:
```python
##########################
### TEST YOUR SOLUTION ###
##########################
bank = import_and_create_bank("bank.txt")
user_accounts, log_in = import_and_create_accounts("user.txt")

tools.assert_false(transfer(bank,log_in,"BrandonK","Jack",100))
tools.assert_false(transfer(bank,log_in,"Brandon","JackC",100))
tools.assert_false(transfer(bank,log_in,"Brandon","Jack",100))

login(user_accounts,log_in,"Brandon","brandon123ABC")
tools.assert_false(transfer(bank,log_in,"Brandon","Jack",200))
tools.assert_true(transfer(bank,log_in,"Brandon","Jack",10))
tools.assert_almost_equal(bank.get("Brandon"),105.5)
tools.assert_almost_equal(bank.get("Jack"),55)

signup(user_accounts,log_in,"BrandonK","123aABCD")
tools.assert_is_none(bank.get("BrandonK"))
login(user_accounts,log_in,"BrandonK","123aABCD")
tools.assert_true(transfer(bank,log_in,"Brandon","BrandonK",10))
tools.assert_almost_equal(bank.get("Brandon"),95.5)
tools.assert_almost_equal(bank.get("BrandonK"),10)
print("Success!")
```

In [ ]:
```python
def change_password(user_accounts, log_in, username, old_password, new_password):
    '''
    This function allows users to change their password.

    If all of the following requirements are met, changes the password and returns True. Otherwise, returns False.
    - The username exists in the user_accounts.
    - The user is logged in (the username is associated with the value True in the log_in dictionary)
    - The old_password is the user's current password.
    - The new_password should be different from the old one.
    - The new_password fulfills the requirement in signup.

    For example:
    - Calling change_password(user_accounts, log_in, "BrandonK", "123abcABC" ,"123abcABCD") will return False
    - Calling change_password(user_accounts, log_in, "Brandon", "123abcABCD", "123abcABCDE") will return False
    - Calling change_password(user_accounts, log_in, "Brandon", "brandon123ABC", "brandon123ABC") will return False
    - Calling change_password(user_accounts, log_in, "Brandon", "brandon123ABC", c"123abcABCD") will return True

    Hint: Think about defining and using a separate valid(password) function that checks the validity of a given password.
    This will also come in handy when writing the signup() function.
    '''

    # your code here
    print("="*50)
    print("Vstup do metody change_password")
    #print(user_accounts)
    #print(username)
    if (old_password == new_password):
        return False
    else:
        for keys in user_accounts.keys():
            print(keys)
            print("Timto jmenem se prihlasujes ", username)
            if (keys == username):
                print("Toto ulozene heslo", user_accounts[keys])
                print("Toto napsane heslo", old_password)
                if (user_accounts[keys] != old_password):
                    return False
                else:
                    print("Vypisuju log_in", log_in[keys])
                    #print(((validate(username, new_password)) and log_in[keys]))
                    if (validate(username, new_password)) and log_in[keys]=='True':
                        print("2")
                        user_accounts[keys]=new_password
                        return True

                    else:
                        print("3")
                        return False

    return False

def validate(username, password):
    #print("Vstup do metody validate")
    l, u, d = 0, 0, 0
    if (len(password) < 7 ):
        result = False
    for i in password:
        # counting lowercase alphabets
        if (i.islower()):
            l+=1
        # counting uppercase alphabets
        if (i.isupper()):
            u+=1
            # counting digits
        if (i.isdigit()):
                d+=1
    if (l>=1 and u>=1 and d>=1 and l+u+d==len(password)) and (username != password):
        result = True
    else:
        result = False

    return result
```

In [ ]:
```python
##########################
### TEST YOUR SOLUTION ###
##########################
bank = import_and_create_bank("bank.txt")
user_accounts, log_in = import_and_create_accounts("user.txt")

tools.assert_false(change_password(user_accounts,log_in,"BrandonK","123abcABC","123abcABCD"))
tools.assert_false(change_password(user_accounts,log_in,"Brandon","brandon123ABC","123abcABCD"))

login(user_accounts,log_in,"Brandon","brandon123ABC")
tools.assert_false(change_password(user_accounts,log_in,"Brandon","123abcABCD","123abcABCDE"))

tools.assert_false(change_password(user_accounts,log_in,"Brandon","brandon123ABC","brandon123ABC"))

tools.assert_false(change_password(user_accounts,log_in,"Brandon","brandon123ABC","123ABCD"))

tools.assert_true(change_password(user_accounts,log_in,"Brandon","brandon123ABC","123abcABCD"))
tools.assert_equal("123abcABCD",user_accounts["Brandon"])
print("Success!")
```

In [ ]:
```python
def delete_account(user_accounts, log_in, bank, username, password):
    '''
    Completely deletes the user from the online banking system.
    If the user exists in the user_accounts dictionary and the password is correct, and the user
    is logged in (the username is associated with the value True in the log_in dictionary):
    - Deletes the user from the user_accounts dictionary, the log_in dictionary, and the bank dictionary.
    - Returns True.
    Otherwise:
    - Returns False.

    For example:
    - Calling delete_account(user_accounts, log_in, bank, "BrandonK", "123abcABC") will return False
    - Calling delete_account(user_accounts, log_in, bank, "Brandon", "123abcABDC") will return False
    - If you first log "Brandon" in, calling delete_account(user_accounts, log_in, bank, "Brandon", "brandon123ABC")
    will return True
    '''

    # your code here
```

In [ ]:
```python
##########################
### TEST YOUR SOLUTION ###
##########################

bank = import_and_create_bank("bank.txt")
user_accounts, log_in = import_and_create_accounts("user.txt")

tools.assert_false(delete_account(user_accounts,log_in,bank,"BrandonK","123abcABC"))
tools.assert_false(delete_account(user_accounts,log_in,bank,"Brandon","123abcABDC"))
tools.assert_false(delete_account(user_accounts,log_in,bank,"Brandon","brandon123ABC"))
login(user_accounts,log_in,"Brandon","brandon123ABC")

tools.assert_true(delete_account(user_accounts,log_in,bank,"Brandon","brandon123ABC"))
tools.assert_is_none(user_accounts.get("Brandon"))
tools.assert_is_none(log_in.get("Brandon"))
tools.assert_is_none(bank.get("Brandon"))
print("Success!")
```

```
In [ ]: def main():
            '''
            The main function is a skeleton for you to test if your overall programming is working.
            Note we will not test your main function. It is only for you to run and interact with your program.
            '''

            bank = import_and_create_bank("bank.txt")
            user_accounts, log_in = import_and_create_accounts("user.txt")

            while True:
                # for debugging
                print('bank:', bank)
                print('user_accounts:', user_accounts)
                print('log_in:', log_in)
                print('')
                #

                option = input("What do you want to do?  Please enter a numerical option below.\n"
                "1. login\n"
                "2. signup\n"
                "3. change password\n"
                "4. delete account\n"
                "5. update amount\n"
                "6. make a transfer\n"
                "7. exit\n")
                if option == "1":
                    username = input("Please input the username\n")
                    password = input("Please input the password\n")

                    # add code to login
                    login(user_accounts, log_in, username, password);
                elif option == "2":
                    username = input("Please input the username\n")
                    password = input("Please input the password\n")

                    # add code to signup
                    signup(user_accounts, log_in, username, password)
                elif option == "3":
                    username = input("Please input the username\n")
                    old_password = input("Please input the old password\n")
                    new_password = input("Please input the new password\n")

                    # add code to change password
                    change_password(user_accounts, log_in, username, old_password, new_password)
                elif option == "4":
                    username = input("Please input the username\n")
                    password = input("Please input the password\n")

                    # add code to delete account
                    delete_account(user_accounts, log_in, bank, username, password)
                elif option == "5":
                    username = input("Please input the username\n")
                    amount = input("Please input the amount\n")
                    try:
                        amount = float(amount)

                        # add code to update amount
                        update(bank, log_in, username, amount)
                    except:
                        print("The amount is invalid. Please reenter the option\n")

                elif option == "6":
                    userA = input("Please input the user who will be deducted\n")
                    userB = input("Please input the user who will be added\n")
                    amount = input("Please input the amount\n")
                    try:
                        amount = float(amount)

                        # add code to transfer amount
                        transfer(bank, log_in, userA, userB, amount)
                    except:
                        print("The amount is invalid. Please re-enter the option.\n")
                elif option == "7":
                    break;
                else:
                    print("The option is not valid. Please re-enter the option.\n")

        #This will automatically run the main function in your program
        #Don't change this
        if __name__ == '__main__':
            main()

In [ ]:
```