Assignment 1

due 6 Feb 2020

Complete the following tasks. Submit your solutions in a single file named `a1.py` using Canvas on or before the date listed above.

Read the PEP8 programming style guidelines. For now concentrate on the section on code layout, comments and naming. You may use the # style comments for module header and function header comments rather than the triple quote style.

Apart from the class definitions themselves, do not include any other code in `a1.py`. Place any testing code in a *seperate file* and import the assignment code. You are not required to submit the test code. The only exception to the prohibition mentioned in the first sentence of this paragraph are import statements and statement to define global constants.

1. Write a complete Python class definition for a simple credit card class, named `CreditCard`, with the following behaviour. Each instance embodies five pieces of information: (i) the name of the customer, (ii) the name of the bank, (iii) the account identifier, (iv) the limit (upper limit to how much can be charged to the card) and (v) the balance (in euros). Write a getter for each of these (e.g. `get_balance` returns the current balance). In addition the objects support two further functions.

   charge(price) Charge given price to the card, assuming sufficient credit. Return True if charge succeeded and false if charge denied.

   make_payment(amount) Process customer payment by increasing balance by 'amount'

   Include a per-class function `show_summary` that prints a list of all the cards giving the customer name, account number and balance in each case. The cards should be listed in alphabetical order by customer name.

   The term balance should be interpreted in the following way. A negative balance implies that the customer has spent (charged) more than they have paid in. Every `charge` operation reduces the balance, every `make_payment` operation increases it (i.e. makes it more positive.) Note that the balance cannot go below the limit set for that card. Any `charge`s that would take the balance below this, would

2. Write a class definition for a *priority queue* container class. A priority queue is a container that holds a collection of key-value pairs known as items. Keys need not be distinct. Priority queue objects support the following functions.

   len() Return the number of items in the container.

   is_empty() Return True if the container holds no items and False otherwise.

   add(k, v) Insert an item with the specified key, value.

   min() Return (but do not remove) a pair (k, v) representing a minimum key; where there are multiple keys sharing the same minimum value an arbitrary item with the keys is chosen; an error occurs if the priority queue is empty.

   remove_min() Remove and return a pair (k, v) representing a minimum key; where there are multiple keys sharing the same minimum value and arbitrary item with the keys is chosen; an error occurs if the priority queue is empty.

For example, if the priority queue contained the following three items

$$(3, Z) \quad (1, X) \quad (2, Y)$$

then three successive calls to `remove_min` would return $(1, X)$, $(2, Y)$ and $(3, Z)$ in that order.

Note that this class definition provides a blueprint for the priority queue container concept. The definition should allow multiple priority queue objects to be created. There should be no need for any (per-)class variables or functions.

Note the contrast with class `CreditCard` above. With that class each object is fairly simple but we employ the (per-)class variable/function machinery to provide a mechanism for holding (containing) all the cards. With `PriorityQueue` each object is itself a container (holding key-value pairs) and there is no need for (per-)class machinery to hold the individual objects.