Introduction to programming in Python

DAT 455 (also re-exam DAT425, DAT445, TIN214, EEN110) Chalmers University of Technology
Exam 2021-08-17 kl 8:30-12:30 at distans via Canvas
Examiner Aarne Ranta, email aarne@chalmers.se

The exam has 3 questions. The grading scale is G/U. For a G, 50% of points are required, i.e. at least 15 points out of 30.

Answers are to be given in a file with the name `python_exam.py` which can be imported in Python3. The file must include the functions and classes that are specified in the questions. The file can also include comments that Python3 ignores but the teachers can read. The file may not import any libraries.

You can use all material that is available on the web or in books. But you are not allowed to communicate with other people during the exam. The only exception is email to aarne@chalmers.se in case you have any questions.

Our best advice is that you actually test your file in Python3 to make sure it works before you submit the file.

Enjoy!

Before you start, write the following on top of your python_exam.py file, either in Swedish,

```
# Jag bekräftar härmed att jag inte kommunicerar med andra personer än kursens lärare under tentans gång.

# Jag är medveten om att fusk i tentan kan leda till disciplinåtgärder.
```

or in English:

```
# I hereby confirm that I do not communicate with other people than the teacher of the course during the exam.

# I am aware that cheating in the exam may lead to disciplinary measures.
```

Question 1: Pandemic Rules (8p)

Norway has the following rules for persons entering the country from Sweden:

- you are welcome to Norway if you come from a "green" region, you can enter freely
- you are also welcome if you are fully vaccinated or have had Covid-19 during the last 6 months
- otherwise, you are still welcome but must get tested and stay in quarantine

(These rules are a simplified version from the official rules in https://www.udi.no/om-koronasituasjonen/innreise-og-opphold-jeg-er-utlandet/bosatt-i-eu-eos-schengen-eller-storbritannia/#link-24945 ; you don't need to read these :-)

Your task is to write a function norway_pandemic(), which implements these rules in a dialogue. It asks a series of three questions (in Norwegian of course!): region, vaccine, and earlier Covid. It expects answers as follows:

- the green regions are ones that start with a 'V' (Västra Götaland, Värmland, etc - another simplification)
- the yes/no questions interpret the answer 'ja' as positive, all others as negative
- **important: the program terminates as soon as it can be sure about the outcome, and does not pose any other questions then**

The screenshot on the right shows four different ways the dialogue can proceed in Python3. **Copy the system's questions and answers from the dialogue.** The user's answers are underlined.

```
>>>  norway_pandemic()
Hvor er du bosatt? Värmland
Velkommen til Norge!

>>> norway_pandemic()
Hvor er du bosatt? Stockholm
Er du fullvaksinert? ja
Velkommen til Norge!

>>> norway_pandemic()
Hvor er du bosatt? Stockholm
Er du fullvaksinert? nej
Har du gjennomgått koronasykdom de siste seks månedene? ja
Velkommen til Norge!

>>> norway_pandemic()
Hvor er du bosatt? Skåne
Er du fullvaksinert? nej
Har du gjennomgått koronasykdom de siste seks månedene? nej
Velkommen til Norge, men du må teste deg och sitte i karantene.
```

Question 2. Scrambled text

Accroidng to a rseaecrehr at Cmarbdige Uinevsrtiy , it deons't mtaetr in waht odrer the lteetrs in a wrod are , the olny ipmroatnt tihng is taht the frist and lsat lteetr be at the rgiht palce .
(adapted from https://www.dictionary.com/e/typoglycemia/)

Your task is to implement a word scrambling algorithm that works as follows: for every word in a text, keep the first and the last letter as they are, but **alter** the order of the letters in between. The screen dump on the right shows how this works with several examples.

Altering here means that you swap the first letter with the second, the third with the fourth, and so on. If there are an odd number of letters, the last letter remains in its original place.

Your solution should consist of three functions:

- **alter(s),** which swaps the letters of string s as described
- **scramble(s),** which keeps the first and last letters of string s and applies alter() to the letters between them
- **scrambles(s)**, which splits the string s into words and applies scramble() to each of them (you can assume that punctuation marks are separated by spaces, as in our example)

Bonus question: how do you unscramble a scrambled text, i.e. bring it back to the original? Do you need a new function for this?

```
>>> alter("ab")
'ba'
>>> alter("abcdef")
'badcfe'
>>> alter("abcde")
'badce'
>>> alter("a")
'a'
>>> scramble("Raby")
'Rbay'
>>> scramble("Rab")
'Rab'
>>> scramble("Rabely")
'Rbaley'
>>> scrambles("According to a researcher at Cambridge
University , it doesn't matter in what order the
letters in a word are , the only important thing is
that the first and last letter be at the right place
.")
"Accroidng to a rseaecrehr at Cmarbdige Uinevsrtiy ,
it deons't mtaetr in waht odrer the lteetrs in a wrod
are , the olny ipmroatnt tihng is taht the frist and
lsat lteetr be at the rgiht palce ."
```

Question 3. Bank accounts (12p)

Your task is to implement a simple class for bank accounts. The class keeps track of two things:

- the account number, which is set when the account is created, and never changes
- the account balance, which starts with an initial amount, and changes when money is transferred between accounts

Write the `Account` class definition, initialization, and the two methods by filling in what is missing from the `Account` code on the right above.

Also write a function `transfer()` for making transfers between accounts. The function must check that there is enough money in the account from which money is taken.

Your class and function should behave in the way shown in the Python3 session on the right below.

```
class Account:
    # an account has a number and a balance

    def get_balance(self):
        # return the actual balance

    def set_balance(self,b):
        # change the balance to b

def transfer(account1,account2,amount):
     # transfer amount from account1 to account2
     # report "OK" if successful,
     # "not enough money" otherwise
```

```
>>> acc1 = Account('0010', 2000)
>>> acc2 = Account('0201', 2000)
>>> transfer(acc1,acc2,600)
OK
>>> acc1.get_balance()
1400
>>> acc2.get_balance()
2600

>>> transfer(acc1,acc2,1600)
not enough money
>>> acc1.get_balance(), acc2.get_balance()
(1400, 2600)
```