Introduction to programming in Python (DAT455)

26 October 2021

Examiner: Aarne Ranta
Author: Krasimir Angelov

Note: The solution must be sent to the examiner before 18:00 as a single file. The file should be the same as the Python file you have developed, but you should change its name from python_exam.py to python_exam.txt so that the email system accepts it.

All help materials are allowed, but you must work alone. Communication with other people (except email to the author of the exam) is not allowed, and cheating may lead to suspension from the university.

The total number of points is 40. 20 points is required for the grade G.

Question 1

There is a heap of red, green and blue balls which one person needs to count by using a program. The program asks for the color of the current ball and keeps track of how many red, green and blue balls have been seen so far. An example, dialog with the program looks like this:

```
What is the color of the ball? red
So far: 1 red, 0 green, 0 blue
What is the color of the ball? green
So far: 1 red, 1 green, 0 blue
What is the color of the ball? red
So far: 2 red, 1 green, 0 blue
…
What is the color of the ball? stop
```

The dialog ends when the user types in “stop”. If the user has typed in anything else besides red, green, blue or stop, an error should be printed and the program must continue the dialog.

(10 points)

Question 2

An easy (but not secure) way to encrypt a message is to shuffle the letters in the text in a certain way. Implement the function `encrypt(msg,code)` which takes the string msg and the list code and returns the encrypted text. The code itself tells us how to reorder the letters, for example `[2,1,0]` means that the letter with index 2 should become first, followed by the letters with indices 1 and 0. For example:

```
encrypt("abc", [2,1,0]) ⇒ "cba"
encrypt("abc", [1,2,0]) ⇒ "bca"
```

If the text is longer than the length of the code, then you can continue by repeating the shuffling:

```
encrypt("abcdef", [2,1,0]) ⇒ "cbafed"
encrypt("abcdef", [1,2,0]) ⇒ "bcaefd"
```

If the length of the text is not divisible by the length of the code, then you should append as many spaces as necessary to the end of the text before the encryption. Example:

```
encrypt("abcdef", [2,1,0,3]) ⇒ "cbad␣fe␣"
```

Which is equivalent to encrypting the text `"abcdef␣␣"`.

(13 points)

Question 3

Implement the class `BookContents` which models the contents of a book. The class should have the following components:

- An `__init__` method which initializes an object to represent an empty table of contents.
  (3 points)
- A method `addChapter(name, page)` which adds a new chapter with the given name and with the given start page to the contents.
  (3 points)
- A method `printContents()` which prints the chapters of the book in the order of increasing page numbers.
  (3 points)
- A method `pageToChapter(page)` which returns the name of the chapter that contains the page number. If the given page number is smaller than the page for the first chapter then the result should be `None`.
  (6 points)

Example:

```
contents = BookContents()
contents.addChapter("Computers and Programs", 1)
contents.addChapter("Objects and Graphics", 123)
contents.addChapter("Writing Simple Programs", 25)
contents.addChapter("Computing with Strings", 77)
contents.addChapter("Computing with Numbers", 51)
contents.printContents()
```

should print:

```
Computers and Programs        1
Writing Simple Programs      25
Computing with Numbers       51
Computing with Strings       77
```

```
Objects and Graphics        123

contents.pageToChapter(0) ⇒ None
contents.pageToChapter(30) ⇒ "Writing Simple Programs"
contents.pageToChapter(60) ⇒ "Computing with Numbers"
```

(17 points)