Introduction to programming – ASSIGNMENTS 10

General instructions:

- Type in and test your programs using Python Idle.
- Bring in your answers to the next assignment (i.e. demonstration) session in memory stick, or save them to a web folder accessible in class. Alternatively, you can bring written answers with you, though this is not recommended.
- Remember to comment your code – this does NOT mean, that every single line should be commented. Comment the important parts in your program.
- Prepare to present your solution to the class.

1.

Use operation *randint* from module *random* to write a program, that queries the user for a number, then proceeds to generate that many random numbers between 1 and 100 and store them into a list. Finally the list is output.

Example run:

```
Enter a number: 5
[11, 32, 4, 91, 6]
```

2.

Write a function `fractionAvg(listOfFractions)`, that gets a list of fractions as an argument, and proceeds to calculate and return the average of all fractions in that list. Use class *Fraction* from module *fractions*.

Example run (in Python Shell):

```
>>> m = [Fraction(1,2), Fraction(1,4), Fraction(1,3)]
>>> print fractionAvg(m)
13/36
```

3.

Write a function `dateAfter(d1, d2)`, which gets two strings containing dates as an argument, and returns True if d1 is later than d2. The strings are given in form "d.m.y", where d represents a day, m a month and y a year. Use *date* object from module *datetime* to compare the dates.

Example run (in Python Shell):

```
>>> first = "29.9.2001 "
>>> second = "13.3.2003"
>>> print dateAfter(first, second)
False
>>> print dateAfer(second, first)
True
```

4.

Write a function `randomString(length)`, which generates and returns a random string with given number of letters. The letters should be picked from upper and lower case letters of a…z. As an extra requirement, each letter can occur only once in generated string (note, that lower case and upper case letters are considered different characters in this case).

Use modules *string* and *random*.

Example run (in Python Shell):

```
>>> print randomString(10)
uGtVhTAfKz
```

5.

Note: in this question you need to use the file "persons.dat", which can be found in Moodle under topic "File operations".

Write a function `generateGroup(sizeOfGroup)`, which reads all names from file "persons.dat" (the program should discard age, weight and height), and generates a random group with *sizeOfGroup* members from the names read. Group is returned as a list of strings. Note, that all persons in group must of course be unique.

Example run (in Python Shell):

```
>>> group = generateGroup(4)
>>> print group
['Mary Taylor', 'Arnold White', 'Bill Thomas', 'Hannah Clark']
```

6. ** *Expert assignment (**double points)***

A deck of cards can be modeled via list containing tuples. Each tuple represents a card, and consists of two properties: a value (1…13) and a suit (a string with value of “spades”, “hearts”, “clubs” or “diamonds”).

Write a module `deck_of_cards.py`, with following operations for handling deck of cards:

function `getDeck():`

- Returns a new deck of cards containing all 52 combinations of number and suit.

procedure `shuffle(list deckOfCards):`

- Shuffles the given deck of cards into random order in place (i.e. modifies the given deck, *doesn’t return a new deck!*)

function `deal(list deckOfCards, int number):`

- Returns a new list containing the given number of cards from the top of the deck (i.e. from the beginning of the list). The cards dealt should be removed from the deck.

procedure `output(list listOfCards):`

- Outputs the given list containing card tuples in readable form. The values of *face cards* should be outputted via letters instead of numbers: 1 == A, 11 == J, 12 == Q and 13 == K.

Example usage (in Python shell):

```
>>> from deck_of_cards import *
>>> deck = getDeck()
>>> shuffle(deck)
>>> hand = deal(deck, 3)
>>> output(hand)
5 of spades, Q of hearts, A of diamonds
```