Assignment 1

COMP9021, Session 1, 2015

Aims: The purpose of the assignment is to:

- let you design solutions to simple problems;
- let you implement these solutions in the form of short python programs;
- practice the use of arithmetic computations, tests, repetitions, lists, random number generation, and formatted output.

Submission

Your programs will be stored in a number of files, with one file per exercise, of the appropriate name. When you have developed and tested your programs, look at the provided checklist and verify that you tick all boxes (or at least be aware that you should tick them all...). Then upload your files using WebCMS. Assignments can be submitted more than once: the last version is marked. Your assignment is due by April 5, 11:59pm.

Assessment

For each of the first two exercises, up to 2.5 marks will reward the correctness of the solution, and for the third exercise, up to 3.5 marks will reward the correctness of the solution (provided these solutions have not been hard coded...). For all exercises, up to 0.5 mark will reward good readability of the source code, good comments when needed, and reasonable complexity of the underlying logic.

For all exercises, the automarking script will let each of your programs run for 30 seconds. Still you should not take advantage of this and strive for a solution that gives an immediate output.

Late assignments will be penalised: the mark for a late submission will be the minimum of the awarded mark and 10 minus the number of full and partial days that have elapsed from the due date.

Exercise 1: Multiplication (3 marks)

Write a program named `multiplication.py` that solves the multiplication

```
          * * *
        x * * *
          -----
        * * * *
      * * * *
    * * * *
    ----------
    * * * * * *
```

in such a way that:

- each star stands for a digit, with the leftmost star on each line standing for a nonzero digit;
- all digits on a given line are distinct;
- the sum of all digits on a given line is the same for the 6 lines.

The output of your program should be of the form above, with of course the stars being replaced by appropriate digits. (Still it is irrelevant whether digits are separated by spaces, whether `x` is followed by spaces, whether there are leading spaces at the beginning of the last line, and the number of `-` on the separating lines is also irrelevant.)

Your program should not make any assumption on the actual solution, except obvious ones on the ranges of some values.

Exercise 2: Pascal triangle (3 marks)

Write a program named `pascal_triangle.py` that prompts the user for a nonnegative integer N and displays the first $N + 1$ rows of Pascal triangle. Below is a possible interaction. Make sure that the output is displayed exactly as shown.

```
$ python pascal_triangle.py
Enter a nonnegative integer: zero
Enter a nonnegative integer: 0
1
$ python pascal_triangle.py
Enter a nonnegative integer: 1
 1
1 1
$ python pascal_triangle.py
Enter a nonnegative integer: 4
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
$ python pascal_triangle.py
Enter a nonnegative integer: 8
                 1
               1   1
             1   2   1
           1   3   3   1
         1   4   6   4   1
       1   5  10  10   5   1
     1   6  15  20  15   6   1
   1   7  21  35  35  21   7   1
 1   8  28  56  70  56  28   8   1
$ python pascal_triangle.py
Enter a nonnegative integer: 10
                                1
                             1     1
                          1     2     1
                       1     3     3     1
                    1     4     6     4     1
                 1     5    10    10     5     1
              1     6    15    20    15     6     1
           1     7    21    35    35    21     7     1
        1     8    28    56    70    56    28     8     1
     1     9    36    84   126   126    84    36     9     1
  1    10    45   120   210   252   210   120    45    10     1
```

Exercise 3: Poker dice (4 marks)

Write a program named `poker_dice.py` that simulates the roll of 5 dice, at most three times, as described at

`http://en.wikipedia.org/wiki/Poker_dice`

as well as a given number of rolls of the 5 dice to evaluate the probabilities of the various hands.

The next three pages show a possible interaction.

The following observations are in order.

- Your program has to define the functions `play()` and `simulate()`, but of course it will likely include other functions.
- What is input from the keyboard is displayed in red; what is black is output by the program (in the pdf, for clarity, not at the prompt...).
- The hands are displayed in the order Ace, King, Queen, Jack, 10 and 9.
- All dice can be kept by inputting either `all`, or `All`, or the current hand in any order.
- No die is kept by inputting nothing.
- We have control over what is randomly generated by using the `seed()` function at the prompt, but do not make use of `seed()` in the program.
- To roll a die, we use `randint(0, 5)` **with 0, 1, 2, 3, 4 and 5 corresponding to Ace, King, Queen, Jack, 10 and 9, respectively**.

:

```
$ python
Python 3.4.3 ...
>>> from random import seed
>>> import poker_dice
>>> seed(0)
>>> poker_dice.play()
The roll is: Ace Queen Jack Jack 10
It is a One pair
Which dice do you want to keep for the second roll? all
Ok, done.
>>> poker_dice.play()
The roll is: Queen Queen Jack Jack Jack
It is a Full house
Which dice do you want to keep for the second roll?  All
Ok, done.
>>> poker_dice.play()
The roll is: King King Queen 10 10
It is a Two pair
Which dice do you want to keep for the second roll?  King 10 Queen King 10
Ok, done.
>>> poker_dice.play()
The roll is: Ace King Queen 10 10
It is a One pair
Which dice do you want to keep for the second roll? 10 11
That is not possible, try again!
Which dice do you want to keep for the second roll? ace
That is not possible, try again!
Which dice do you want to keep for the second roll? 10 10
The roll is: King 10 10 10 9
It is Three of a kind
Which dice do you want to keep for the third roll? all
Ok, done.
>>> poker_dice.play()
The roll is: Ace Ace Queen 9 9
It is a Two pair
Which dice do you want to keep for the second roll? Ace
The roll is: Ace Ace Queen Jack 10
It is a One pair
Which dice do you want to keep for the third roll? Ace
The roll is: Ace Queen Queen Jack 10
It is a One pair
```

```
>>> seed(2)
>>> poker_dice.play()
The roll is: Ace Ace Ace King Queen
It is Three of a kind
Which dice do you want to keep for the second roll? Ace Ace Ace
The roll is: Ace Ace Ace 9 9
It is a Full house
Which dice do you want to keep for the third roll? all
Ok, done.
>>> poker_dice.play()
The roll is: King Queen Queen 10 10
It is a Two pair
Which dice do you want to keep for the second roll?
The roll is: Ace King Jack 10 9
It is a Bust
Which dice do you want to keep for the third roll?
The roll is: Queen Jack 10 9 9
It is a One pair
>>> seed(10)
>>> poker_dice.play()
The roll is: Ace Jack Jack 10 10
It is a Two pair
Which dice do you want to keep for the second roll? Jack 10 Jack 10
The roll is: Ace Jack Jack 10 10
It is a Two pair
Which dice do you want to keep for the third roll? Jack 10 Jack 10
The roll is: King Jack Jack 10 10
It is a Two pair
>>> seed(20)
>>> poker_dice.play()
The roll is: King Queen 9 9 9
It is Three of a kind
Which dice do you want to keep for the second roll? 9 King 9 9
The roll is: King 9 9 9 9
It is a Four of a kind
Which dice do you want to keep for the third roll? 9 9 9 9
The roll is: Ace 9 9 9 9
It is a Four of a kind
```

```
>>> seed(0)
>>> poker_dice.simulate(10)
Five of a kind : 0.00%
Four of a kind : 0.00%
Full house     : 10.00%
Straight       : 0.00%
Three of a kind: 0.00%
Two pair       : 20.00%
One pair       : 60.00%
>>> poker_dice.simulate(100)
Five of a kind : 0.00%
Four of a kind : 0.00%
Full house     : 3.00%
Straight       : 4.00%
Three of a kind: 14.00%
Two pair       : 28.00%
One pair       : 45.00%
>>> poker_dice.simulate(1000)
Five of a kind : 0.10%
Four of a kind : 2.50%
Full house     : 3.80%
Straight       : 3.40%
Three of a kind: 17.20%
Two pair       : 20.60%
One pair       : 46.30%
>>> poker_dice.simulate(10000)
Five of a kind : 0.08%
Four of a kind : 1.99%
Full house     : 3.93%
Straight       : 3.33%
Three of a kind: 14.88%
Two pair       : 23.02%
One pair       : 46.58%
>>> poker_dice.simulate(100000)
Five of a kind : 0.08%
Four of a kind : 1.94%
Full house     : 3.85%
Straight       : 3.17%
Three of a kind: 15.47%
Two pair       : 23.02%
One pair       : 46.31%
```