TopGear Assignments

Python Programming – L1

Assignments

Estimated Effort: 3 PDs

Total Points: 150

Required VDI: Linux

Author: Jayapathi Ramamohan

jayapathi.ramamohan@wipro.com

Date: 29-March-2018

This set of case studies require application of following concepts of Python.

- File operations
- Data structures
- Strings
  Functions

To be considered as correct submission, code should meet the following guidelines:

- Source code filename : as specified for each exercise
- Input filename & output filename should not be hardcoded in program, unless it is stated as part of the problem specification
- Output should be as per the sample shown in each exercise
- Individual files have to be uploaded into gitlab, not as a single zip file

Exercise – 1: Filename: vowelcount.py

Write a program that reads contents from a file, and prints word(s) which has (have) the highest number of vowels in it, among the words of the file.

Program has to ignore any character other than alpha characters and white space (space, tab and newline) characters.

Input file is taken as command line argument.

Assume contents of "sample.txt" is as given below:

> *Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.*

Given below is a sample run and expected output.

$ **python vowelcount.py sample.txt**

5 combination

5 resolution

5 execution

$

Exercise – 2: Filename: pt.py

Write a program to print Pascal triangle as shown in the examples given below. Program should take height of the triangle as command line argument.

```
$ python pt.py 10

1  9  36  84  126  126  84  36  9  1
   1  8  28  56  70  56  28  8  1
     1  7  21  35  35  21  7  1
       1  6  15  20  15  6  1
         1  5  10  10  5  1
           1  4  6  4  1
             1  3  3  1
              1  2  1
                1  1
                 1
$

$ python pt.py 5

1  4  6  4  1
  1  3  3  1
    1  2  1
      1  1
       1
$
```

Exercise – 3: Filename: RectIntersection.py

Implement the function named **intersection()** which takes coordinates of two rectangles.

The co-ordinates of a rectangle is passed as a tuple of tuples, represented in the following way.

```
((TopLeftX, TopLeftY), (BottomRightX, (BottomRightY))
```

The function should return the coordinates of the rectangle formed by the intersection of the two rectangles passed as arguments.

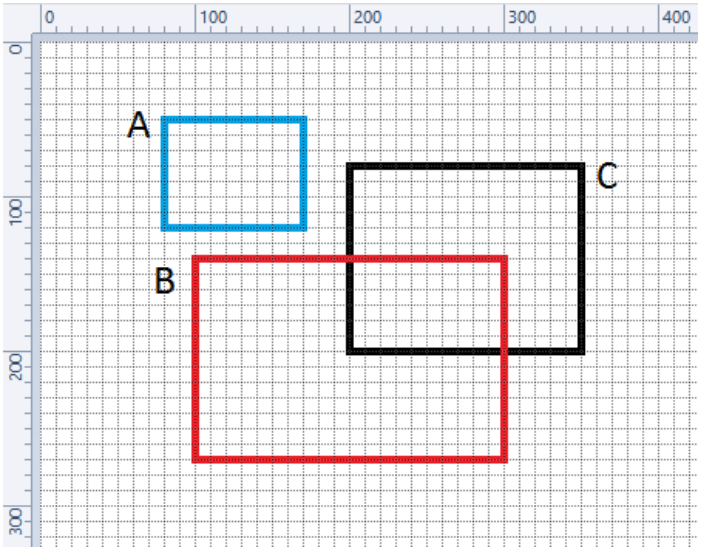
Return value should be **None**, if the rectangles do not intersect.

A = (80, 50), (170, 120)
B = ((100, 140), (300, 270))
C = ((200, 80), (350, 200))

Intersection(B, C)
should return
(200,140), (300, 200)

Intersection(A, B)
should return None

Intersection(A, C)
should return None



Given below is a sample run and expected output.

$ python RectIntersection.py 80 50 170 120 100 140 300 270

None

$

$ python RectIntersection.py 100 140 300 270 200 80 350 200

(200,140), (300, 200)

$

Exercise – 4: Filename: justify.py

Implement the function justify() as per the details given below:

justify(str, width)

where

str – string to be justified

width - column width

text should be justified such that the first non-space character is aligned to the first column and the last non-space character is aligned to the last column.

Assuming input string has N words, the function should return a list in the form

[$word_1$, $spacecount_1$, $word_2$, $spacecount_2$, … , $spacecount_{N-1}$, $Word_N$]

Implement program that takes file name and column width as command line arguments and produces the text that is justified. Program should use justify().

$ python justify.py sample.txt 20

File contents	output
sample text file python code a bc de f	sample text file python code a bc de f

Below table illustrates what kind of output is expected by justify()

Input string	Assume column width = 20	
	Expected output by justify()	Expected output
sample text file	['sample', 3, 'text', 3, 'file']	sample text file
python code	['python', 10, 'code']	python code
a bc de f	['a', 5, 'bc', 5, 'de', 4, 'f']	a bc de f

--- END ---