Домашнє завдання №27_2

Виконати домашнє завдання №27_1 повторно за допомогою мови Python використовуючи RxPY(реалізація ReactiveX для Python)

** зверніть увагу, що блоки коду в Python виділяються відступами, тому будьте уважні при редагуванні наведеного далі прикладу коду: Python 3 забороняє змішування табуляції з пробілами у відступах.*

** **коментар:** це завдання буде аналогічне завданням №28_1, №28_2 та №28_3 і є повністю тотожне до завдань №27_1 та №27_3; таким чином можна порівняти різні засоби програмування; далі наводиться приклад повністю виконаного завдання; для компіляції і запуску можна використати https://repl.it/languages/python3 .*

Вибір варіанту

Варіант завдання відповідає варіанту домашнього завдання №27_1

Приклад коду

Наведений зразок коду реалізовує завдання для 5-ти максимально допустимих спроб введення ключа ліцензії.

** **коментар:** реалізація ConsoleKeyInputDataSource передбачає створення тільки однієї підписки:*

```
inputObservable.subscribe(PrintObserver(win));
```

Максимальна кількість спроб для введення ключа ліцензії	5
Оголошення в коді	`ATTEMPTS_COUNT = 5`

Для коректного виконання коду за допомогою https://repl.it/languages/python3 віртуальну консоль з правого боку краще трохи розширити перед початком виконання коду, а **у процесі виконання розмір консолі не змінювати**.

Лістинг

```
from rx.core import Observer
from rx import from_, operators as op
import curses, time
import re

ATTEMPTS_COUNT = 5
attemptsDownCount = ATTEMPTS_COUNT;
GROUPS_DIGITS_COUNT = 5;
GROUP_DIGITS_SIZE = 5;

PRODUCT_KEY_PART1 = [
0xF, 0xF, 0xF, 0xF, 0xF,
0xD, 0xD, 0xD, 0xD, 0xD,
0x8, 0x8, 0x8, 0x8, 0x8,
0xB, 0xB, 0xB, 0xB, 0xB,
0xF, 0xF, 0xF, 0xF, 0xF
];

PRODUCT_KEY_PART2 = [
0xE, 0xE, 0xE, 0xE, 0xE,
0xF, 0xF, 0xF, 0xF, 0xF,
0xB, 0xB, 0xB, 0xB, 0xB,
```

```
0xF, 0xF, 0xF, 0xF, 0xF,
0xA, 0xA, 0xA, 0xA, 0xA
];

DIGITS_COUNT = GROUPS_DIGITS_COUNT * GROUP_DIGITS_SIZE;

outOfEdgeIndex = 0;
currIndex = 0;
data = [0] * DIGITS_COUNT;
currRowIndex = 0;

def checkProductKey(productKey):
    for index in range(0, DIGITS_COUNT):
        if(productKey[index] ^ PRODUCT_KEY_PART1[index] ^ PRODUCT_KEY_PART2[index]):
            return False;
    return True;

def toDigitPosition(win, currRowIndex, currIndex):
  positionAddon = currIndex // GROUP_DIGITS_SIZE;
  if (positionAddon and positionAddon >= GROUPS_DIGITS_COUNT):
    positionAddon -= 1;
  win.move(currRowIndex, currIndex + positionAddon);
  pass;

def printProductKey(win, productKey, outOfEdgeIndex):
    global currIndex;
    global currRowIndex;
    global DIGITS_COUNT;
    win.move(currRowIndex, 0);
    for index in range(0, DIGITS_COUNT):
        if(index >= outOfEdgeIndex):
            break;
        win.addstr("{:X}".format(productKey[index]))
    pass;

def printFormattedProductKey(win, productKey, outOfEdgeIndex):
    global currIndex;
    global currRowIndex;
    global GROUP_DIGITS_SIZE;
    global DIGITS_COUNT;
    win.move(currRowIndex, 0);
    for index in range(0, DIGITS_COUNT):
        if(index >= outOfEdgeIndex):
            break;
        win.addstr("{:X}".format(productKey[index]))
        if(not((index + 1) % GROUP_DIGITS_SIZE) and (index + 1) < DIGITS_COUNT):
            win.addstr( '-' );
    pass;

def inputHandler(win, ch, key): # ch = chr(key)
    global currIndex;
    global currRowIndex;
    global outOfEdgeIndex;
    global attemptsDownCount;
    if(not attemptsDownCount):
      return;
    if key in (10, curses.KEY_ENTER):
        if (checkProductKey(data) ):
            toDigitPosition(win, currRowIndex, outOfEdgeIndex);
            win.addstr("\nThe product key is correct\n\n");
            currRowIndex += 3;
            toDigitPosition(win, currRowIndex, outOfEdgeIndex);
            printProductKey(win, data, outOfEdgeIndex)
            win.addstr(' (COMPLETE)\n');
            win.addstr('For exit press Ctrl + C\n');
            currRowIndex += 2;
            attemptsDownCount = 0;
        else:
            toDigitPosition(win, currRowIndex, outOfEdgeIndex);
            win.addstr("\nThe product key is not correct\n");
            currRowIndex += 2;
            attemptsDownCount-=1;
            win.addstr("\nYou have " + str(attemptsDownCount) + " attempts to try");
            currRowIndex += 1;
            if(attemptsDownCount):
                win.addstr("\nPlease, enter the product key:\n");
                currRowIndex += 2;
                printFormattedProductKey(win, data, outOfEdgeIndex);
                toDigitPosition(win, currRowIndex, currIndex);
            else:
              win.addstr('\nThe product key is not entered\n');
```

```
            win.addstr('For exit press Ctrl + C\n');
            currRowIndex += 3;
            #sys.exit();
    if key in (8, curses.KEY_BACKSPACE):
        if(currIndex):
            currIndex-=1;
            toDigitPosition(win, currRowIndex, currIndex);
            data[currIndex] = 0;
            win.addstr( '0' );
            toDigitPosition(win, currRowIndex, currIndex);
    if key in (127, curses.KEY_DC):
        toDigitPosition(win, currRowIndex, currIndex);
        data[currIndex] = 0;
        win.addstr( '0' );
        toDigitPosition(win, currRowIndex, currIndex);
    elif key in (27, curses.KEY_LEFT):
        if(currIndex):
            currIndex-=1;
            toDigitPosition(win, currRowIndex, currIndex); # got to 1.5
    elif key in (26, curses.KEY_RIGHT):
        if(currIndex < outOfEdgeIndex):
            currIndex+=1;
            toDigitPosition(win, currRowIndex, currIndex);

    hexDigitRegularExpression = r'^[0-9A-Fa-f]\b'; # /[0-9A-Fa-f]/g
    re.match(hexDigitRegularExpression, ch)
    if (ch and re.match(hexDigitRegularExpression, ch) and currIndex < DIGITS_COUNT):
        data[currIndex] = int(ch, 16);# ch.upper();
        win.addstr(ch.upper()); # win.addstr(str(ch.upper()));
        if(outOfEdgeIndex <= currIndex):
            outOfEdgeIndex = currIndex + 1;
        if(currIndex + 1 < DIGITS_COUNT):
            currIndex+=1;
            if (currIndex != DIGITS_COUNT and currIndex % 5 == 0):
                win.addstr( '-' );
        if(currIndex + 1 == DIGITS_COUNT):
            toDigitPosition(win, currRowIndex, currIndex);
    pass

win = curses.initscr()
curses.noecho() # no echo and no echo '\n'
#curses.cbreak() # ... (curses.raw() ... curses.cbreak())
win.keypad(1)
class ConsoleKeyInputDataSource:
    def __init__(self, win):
        self.win = win
    def __iter__(self):
        while True:
          try:
            time.sleep(0.01)
            yield win.getch()
          except KeyboardInterrupt:
            break;

class PrintObserver(Observer):
    global currRowIndex;
    def __init__(self, win):
      self.win = win;
    def on_next(self, value):
        inputHandler(win, chr(value), value) # inputHandler(chr(key), key)
    def on_completed(self):
        win.addstr("\nDone!\n")
        #currRowIndex +=2;
    def on_error(self, error):
        win.addstr("\nError Occurred: {0}\n".format(error))
        #currRowIndex +=2;

if(attemptsDownCount):
    win.addstr('Please, enter the product key:\n');
    currRowIndex +=1;

inputObservable = from_(ConsoleKeyInputDataSource(win));
inputObservable = inputObservable.pipe(
    #filter is not used
    #op.filter(lambda key : not (key and chr(key) == 'Q')),
    op.map(lambda key :
    ord('0') if (key and (chr(key) == ' ' or chr(key) == '\t')) else key)
)
inputObservable.subscribe(PrintObserver(win));
#sys.exit(0);
```