Міністерство освіти і науки України

Національний авіаційний університет

Факультет кібербезпеки, комп'ютерної та програмної інженерії

**Імітаційне моделювання**

Самостійна робота №2

«Дослідження генераторів випадкових чисел для різних законів розподілення»

Роботу виконав:

студент групи СП-325

Козлов Олексій

Роботу прийняла:

Нечипорук О.П.

Київ – 2020

**Мета роботи:** Ознайомитись із алгоритмами роботи генераторів випадкових чисел, що мають різні закони розподілення та методикою їх тестування.

1. Теоретичний опис функцій
   1. Створення файлу

При виконані цієї функції формується масив випадкових величин за заданим законом розподілу (Тип розподілу), параметри розподілу також задані у полу "номер залікової книжки", обсяг вибірки задається в полі "обсяг вибірки", після створення вибірки вона зберігається у форматі JSON у файлі що вказаний в полі "шлях до файлу", якщо файл існує, то він перезаписується, якщо ні, то створюється.

* 1. Тип величини

При виконанні цієї функції виконується перевірка типу випадкової величини, чи є вона рівномірною чи дискретною. Спочатку масив випадкових величин читається із файлу, що вказаний в полі "шлях до фалу", потім оцінюється обсяг вибірки, і кількість унікальних значень в цій вибірці, в разі якщо вони співпадають, випадкова величина є рівномірною, в іншому випадку, вона дискретна. Метод перевірки працює доки обсяг вибірки не перевищує періодичність генераторів випадкових величин.

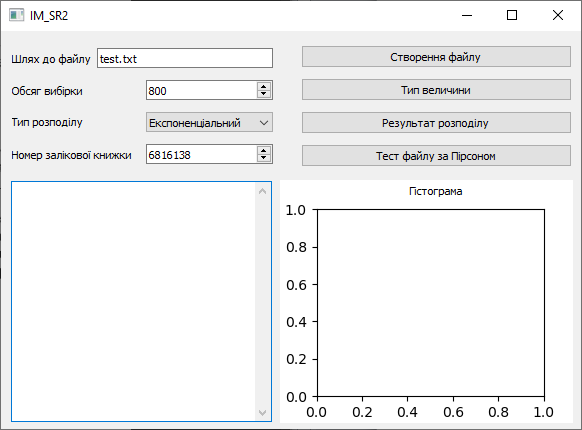
* 1. Результат розподілу

При виконанні цієї функції виводиться вся вибірка випадкових величин та гістограма цієї вибірки.

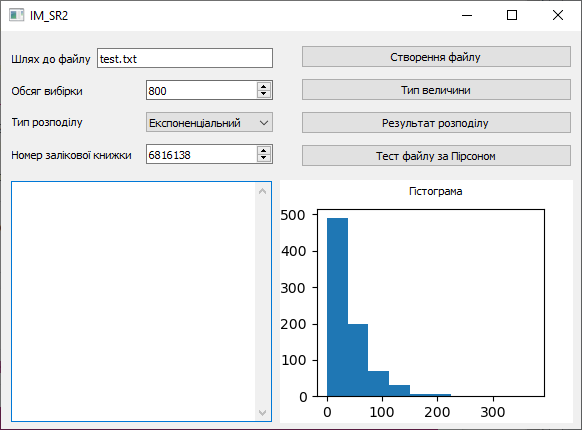
* 1. Тест за критерієм Пірсона

При виконанні цієї функції виконується перевірка статистичної гіпотези на рівномірність за критерієм Пірсона. Для всіх типів розподілу окрім рівномірного Хі-квадрат має значно перевищувати критичне значення, Для рівномірного розподілу Хі-квадрат має бути меншим за критичне значення.

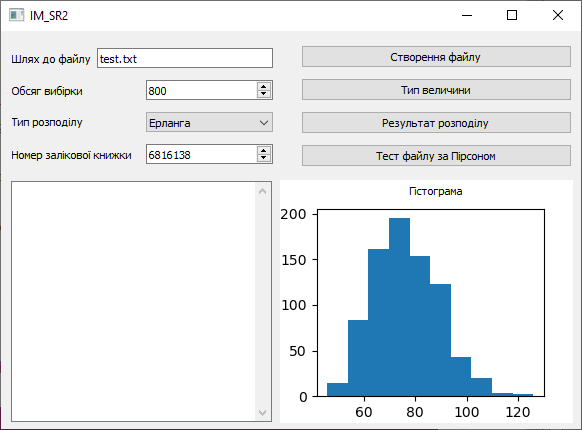
1. Інтерфейс користувача



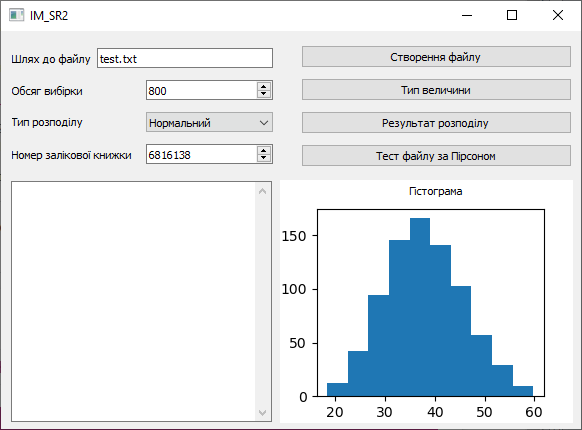
1. Генерація випадкових чисел
   1. Експоненціальний розподіл



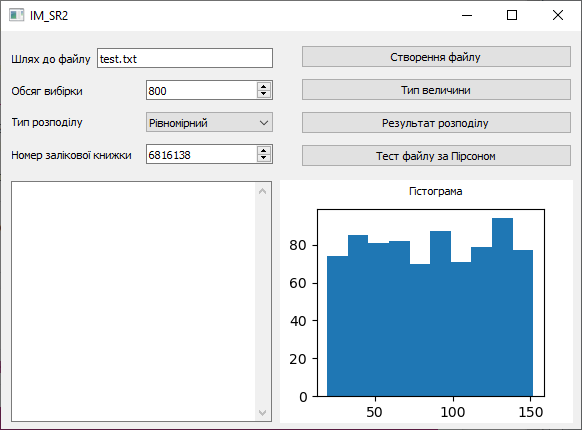
* 1. Розподіл Ерланга



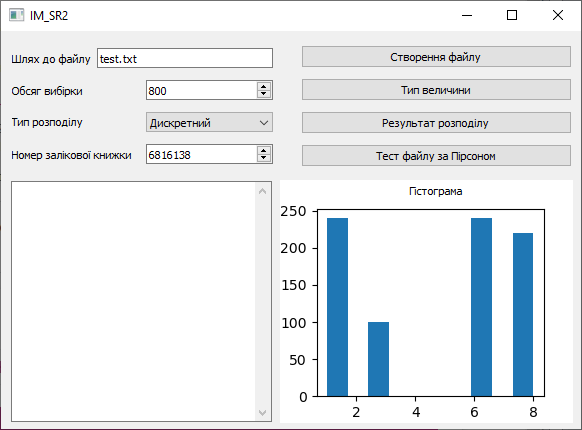
* 1. Нормальний розподіл



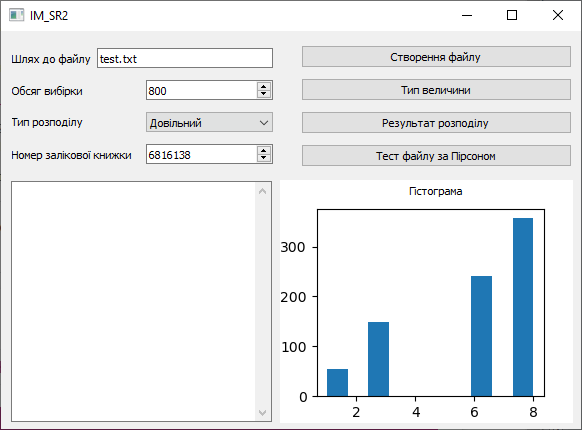
* 1. Рівномірний розподіл



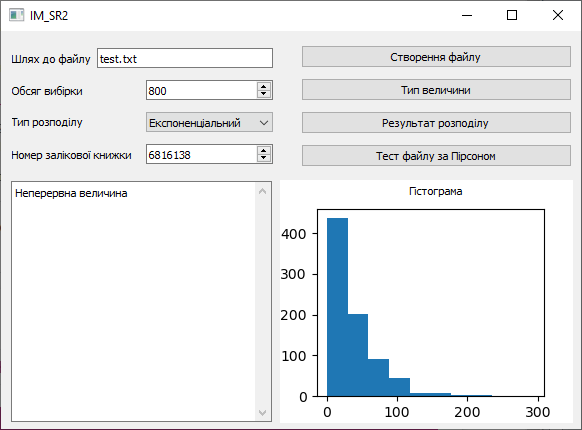
* 1. Дискретний розподіл



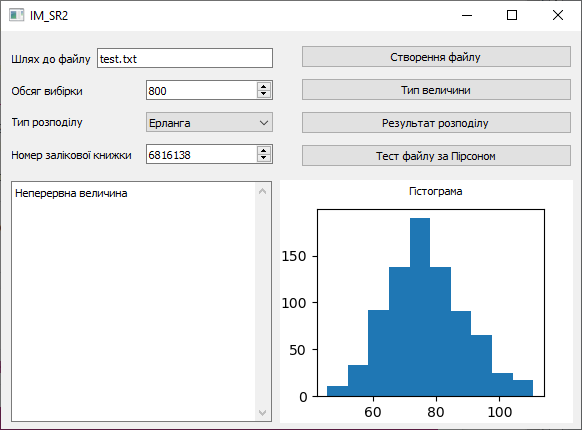
* 1. Довільний розподіл



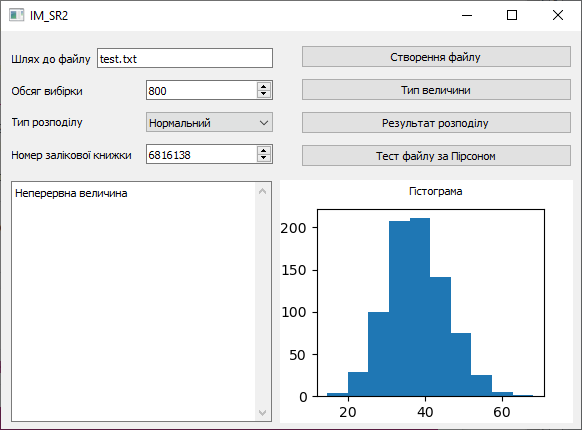
1. Перевірка типу величини
   1. Експоненціальний розподіл



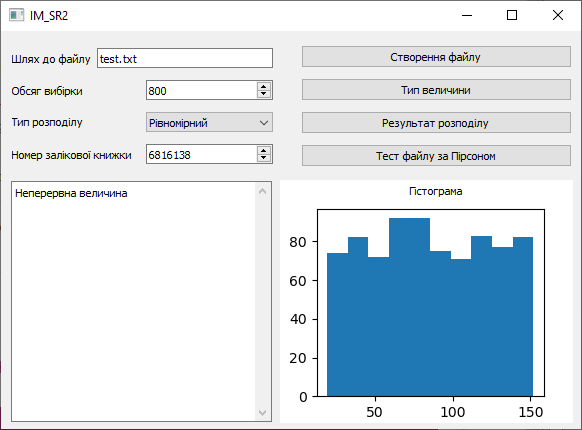
* 1. Розподіл Ерланга



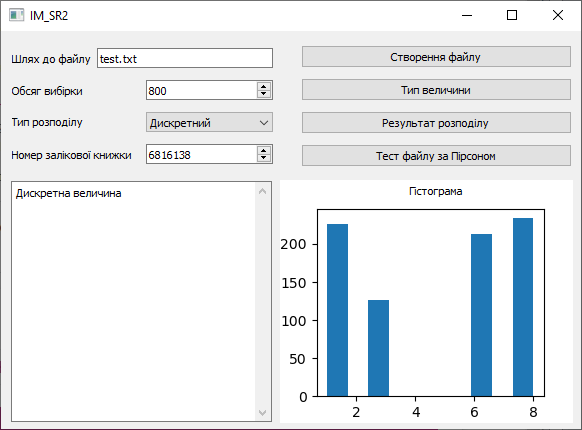
* 1. Нормальний розподіл



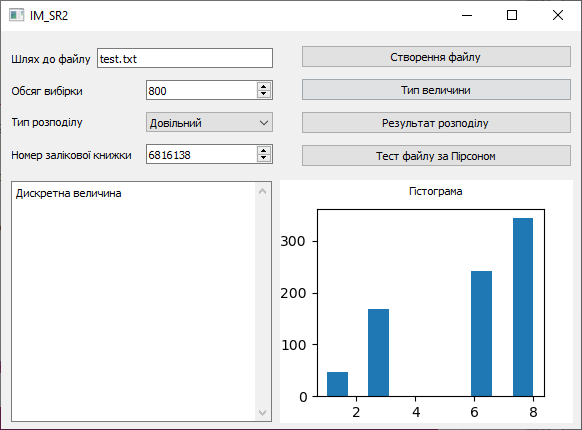
* 1. Рівномірний розподіл



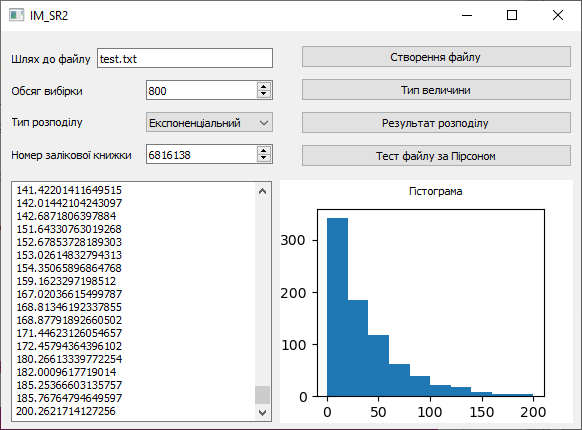
* 1. Дискретний розподіл



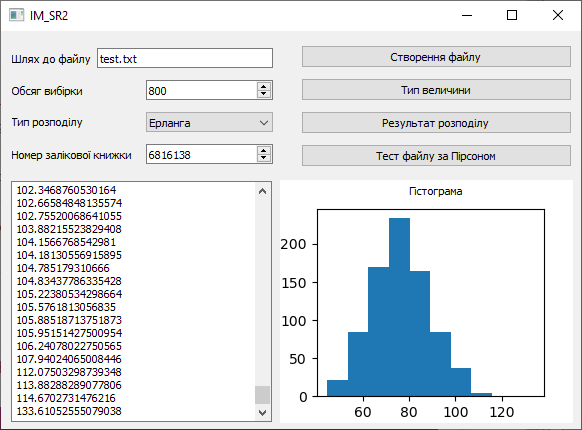
* 1. Довільний розподіл



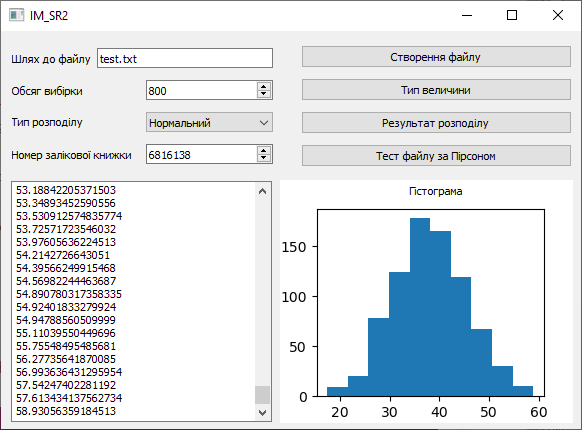
1. Результат розподілу
   1. Експоненціальний розподіл



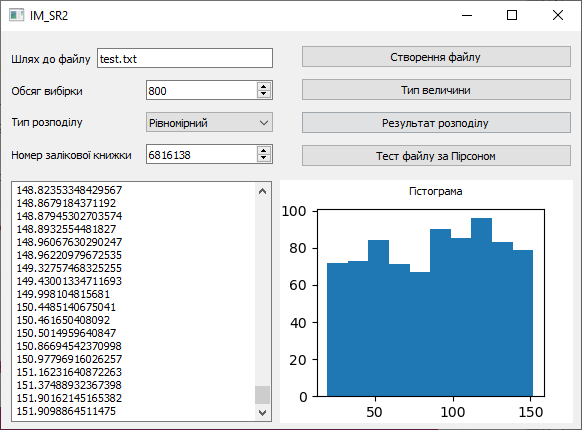
* 1. Розподіл Ерланга



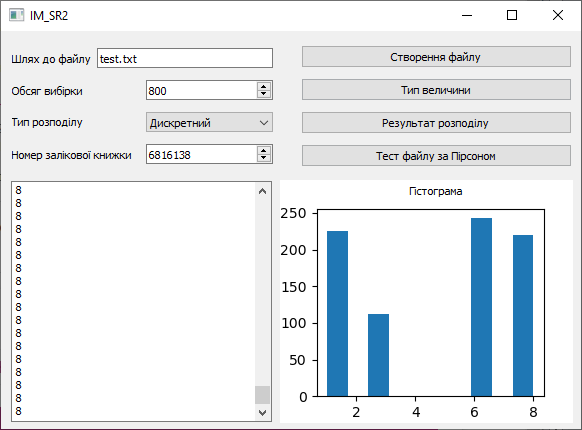
* 1. Нормальний розподіл



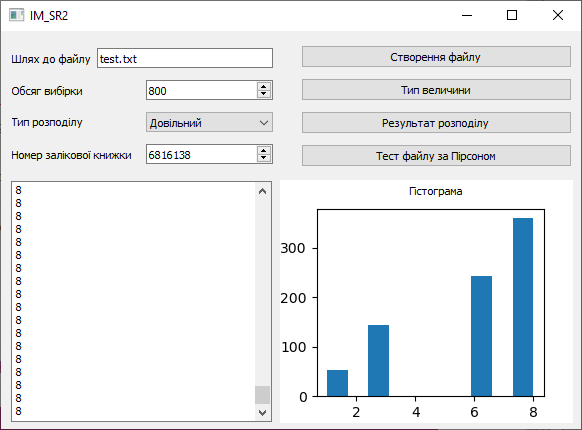
* 1. Рівномірний розподіл



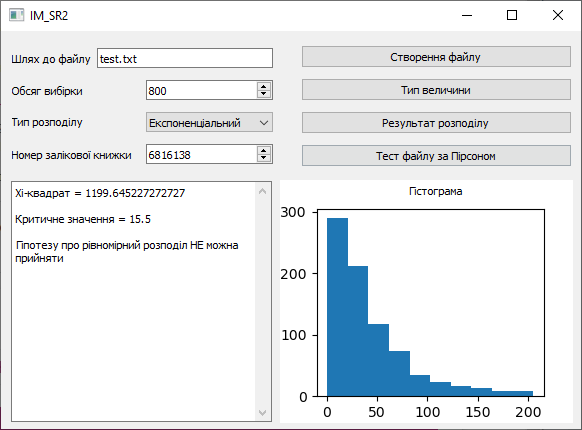
* 1. Дискретний розподіл



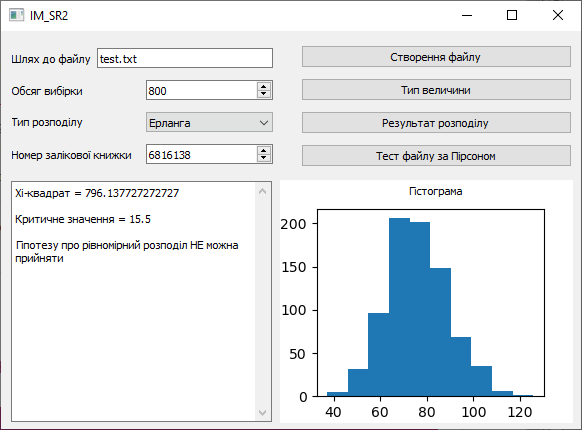
* 1. Довільний розподіл



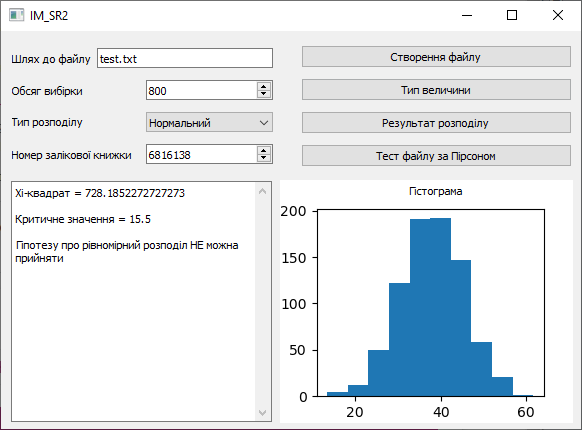
1. Тест за критерієм Пірсона
   1. Експоненціальний розподіл



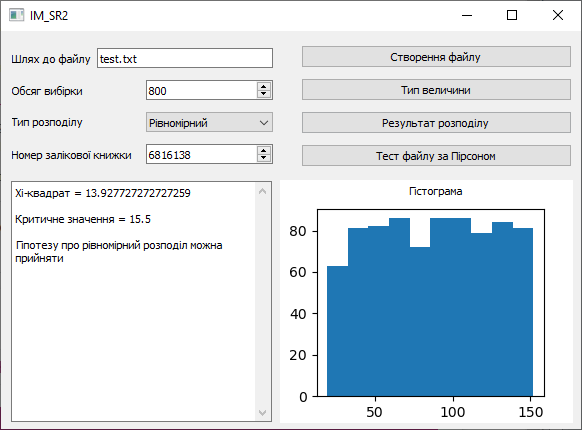
* 1. Розподіл Ерланга



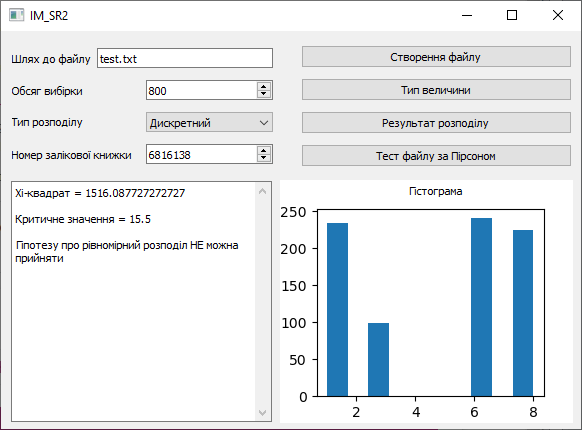
* 1. Нормальний розподіл



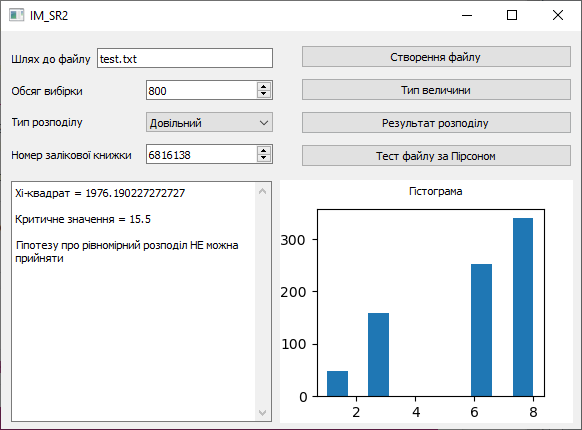
* 1. Рівномірний розподіл



* 1. Дискретний розподіл



* 1. Довільний розподіл



1. Код програми

from random import \*  
import sys  
from PyQt5 import QtCore, QtGui, QtWidgets  
from PyQt5.QtWidgets import \*  
from IM\_SR2\_ui import Ui\_MainWindow  
import numpy as np  
import json  
  
def digits\_recursive(n, digits=None):  
 if digits is None:  
 digits = []  
 return digits\_recursive(n // 10, [n % 10] + digits) if n else digits or [0]  
  
def discrete\_variate(Seed):  
 discrete\_values = digits\_recursive(Seed)  
 basic\_p = 1/len(discrete\_values)  
 p\_dict = {}  
 for i in discrete\_values:  
 if i in p\_dict:  
 p\_dict[i] += 1  
 else:  
 p\_dict[i] = 1  
 prev\_p = 0  
 for key in p\_dict:  
 prev\_p += p\_dict[key]\*basic\_p  
 p\_dict[key] = prev\_p  
 rv = random()  
 interval\_i = 0  
 for key in p\_dict:  
 if rv <= p\_dict[key]:  
 rv = key  
 break  
 else:  
 interval\_i += 1  
 return rv  
  
def arbitrary\_distribution(Seed):  
 discrete\_values = sorted(list(set(digits\_recursive(Seed))))  
 basic\_p = 1 / (2\*len(discrete\_values))  
 p\_dict = {}  
 for i in discrete\_values:  
 if i in p\_dict:  
 p\_dict[i] += 1  
 else:  
 p\_dict[i] = 1  
 prev\_p = 0  
 shift = 0  
 for key in p\_dict:  
 prev\_p += p\_dict[key] \* basic\_p + shift\*2\*basic\_p  
 p\_dict[key] = prev\_p  
 shift += 1  
 rv = uniform(0, 2)  
 interval\_i = 0  
 for key in p\_dict:  
 if rv <= p\_dict[key]:  
 rv = key  
 break  
 else:  
 interval\_i += 1  
 return rv  
  
  
class MyWindow(QMainWindow, Ui\_MainWindow):  
 def \_\_init\_\_(self):  
 QMainWindow.\_\_init\_\_(self)  
 self.setupUi(self)  
 self.setWindowTitle("IM\_SR2")  
 self.pushButton.clicked.connect(self.create\_file)  
 self.pushButton\_2.clicked.connect(self.view\_file)  
 self.pushButton\_3.clicked.connect(self.view\_params\_file)  
 self.pushButton\_4.clicked.connect(self.pirson\_test)  
  
 def create\_file(self):  
 N = self.spinBox.value()  
 Zach = self.spinBox\_2.value()  
 Last\_two = Zach % 100  
 betas = {  
 0: 2,  
 1: 3  
 }  
 beta = betas[Last\_two % 2]  
 variates = {  
 0: [expovariate(1/Last\_two) for \_ in range(N)],  
 1: [gammavariate(Last\_two, beta) for \_ in range(N)],  
 2: [normalvariate(Last\_two, Last\_two\*0.2) for \_ in range(N)],  
 3: [uniform(Last\_two/2, Last\_two\*4) for \_ in range(N)],  
 4: [discrete\_variate(Zach) for \_ in range(N)],  
 5: [arbitrary\_distribution(Zach) for \_ in range(N)],  
 }  
 X = variates[self.comboBox.currentIndex()]  
 X = sorted(X)  
 self.mplWidget.canvas.axes.clear()  
 self.mplWidget.canvas.axes.hist(X)  
 self.mplWidget.canvas.draw()  
  
 filename = self.lineEdit.text()  
 with open(filename, 'w') as fw:  
 json.dump(X, fw)  
  
 def view\_file(self):  
 filename = self.lineEdit.text()  
 with open(filename, 'r') as fr:  
 X = json.load(fr)  
 if len(X) == len(set(X)):  
 self.textBrowser.setText("Неперервна величина")  
 else:  
 self.textBrowser.setText("Дискретна величина")  
  
 def view\_params\_file(self):  
 filename = self.lineEdit.text()  
 with open(filename, 'r') as fr:  
 X = json.load(fr)  
 self.textBrowser.setText("")  
 for x in X:  
 self.textBrowser.append(str(x))  
 self.mplWidget.canvas.axes.clear()  
 self.mplWidget.canvas.axes.hist(X)  
 self.mplWidget.canvas.draw()  
  
 def pirson\_test(self):  
 filename = self.lineEdit.text()  
 with open(filename, 'r') as fr:  
 X = json.load(fr)  
  
 freq, bins = np.histogram(X)  
 theoretical\_frequency = len(X)/len(bins)  
 critical\_distribution\_points = {1: 0.05, 2: 3.8, 3: 6.0, 4: 7.8, 5: 9.5, 6: 11.1, 7: 12.6, 8: 14.1,  
 9: 15.5, 10: 16.9, 11: 19.7, 12: 21.0, 13: 22.4, 14: 23.7, 15: 25.0,  
 16: 26.3, 17: 27.6, 18: 28.9, 19: 30.1, 20: 31.4, 21: 32.7, 22: 33.9,  
 23: 35.2, 24: 36.4, 25: 37.7, 26: 38.9, 27: 40.1, 28: 41.3, 29: 42.6,  
 30: 43.8}  
 xi2 = 0  
 for i in range(len(bins) - 1):  
 xi2 += (freq[i] - theoretical\_frequency) \*\* 2 / theoretical\_frequency  
 self.textBrowser.setText("Хі-квадрат = " + str(xi2))  
 degree\_of\_freedom = (len(bins) - 1) - 1  
 critical\_point = critical\_distribution\_points.get(degree\_of\_freedom)  
 self.textBrowser.append("\nКритичне значення = " + str(critical\_point))  
  
 if xi2 < critical\_point:  
 self.textBrowser.append("\nГіпотезу про рівномірний розподіл можна прийняти")  
 else:  
 self.textBrowser.append("\nГіпотезу про рівномірний розподіл НЕ можна прийняти")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 app = QtWidgets.QApplication(sys.argv)  
 my\_app = MyWindow()  
 my\_app.show()  
 sys.exit(app.exec\_())

**Висновок:** в ході виконання самостійної роботи ми ознайомились із алгоритмами роботи генераторів випадкових чисел, що мають різні закони розподілення та методикою їх тестування. Ми виконали тест за критерієм Пірсона для випадкових величин згенерованих за різними законами розподілу і для всіх законів розподілу окрім рівномірного Хі-квадрат був значно більше критичного значення.