# Базовая теория

Нормальное распределение — это распределение вероятностей, которое имеет колоколообразную форму. Значения вблизи центра распределения более вероятны, чем значения в хвостах распределения

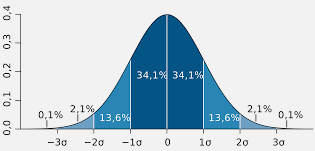


Рисунок 1 - Нормальное распределение

 Равномерное распределение - это такое распределение вероятностей, при котором каждое значение в интервале от a до b равновероятно

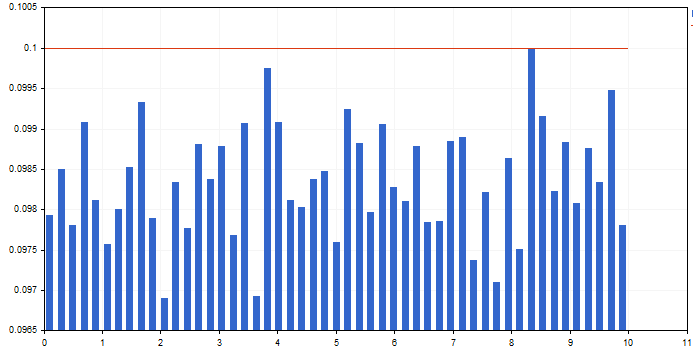


Рисунок 2 – Равномерное распределение

# Распределения с генератором псевдослучайный чисел на Python 3

## Синтаксис

Используя встроенный модуль python random (import random) можно определить массив любой размерности.

По умолчанию random.randint(min, max) возвращает псевдослучайное число типа int.

Для **Равномерного распределения** типа double random.uniform(min, max)

Для **нормального распределения** типа double random.gauss(average, sko), где average – матожидание, sko – среднее квадратичное отклонение

**Если воспользоваться циклом, получим массив из чисел, который будет подчиняться равномерному закону распределения**

    dist = [random.randint(1, 100) for x in range(10)]

    print(dist)

#dist = [74, 25, 4, 57, 11, 12, 49, 13, 36, 83]

## Визуализация

Чтобы посмотреть на гистограмме, как выглядит распределение, нужно использовать сторонние пользовательские модули, например **matplotlib**. Чтобы студенту не нужно было устанавливать дополнительные модули, была написана реализация гистограммы в терминале в виде символов ASCII. Ее листинг можно найти в приложении ниже

def ascii\_plot(dist, num\_of\_intervals=20)

-где dist – одномерный массив, num\_of\_internals – количество интервалов для разбиения (20 по умолчании для наглядности)

Функция принимает на вход одномерный массив и рисует график распределения:

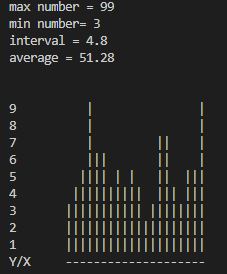


Рисунок 3 – работа ascii\_plot(), пример равномерного распределения

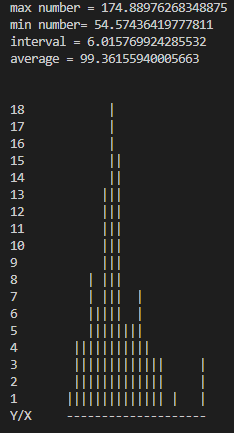


Рисунок 4 – Нормальное распределение с матожиданием = 100 и СКО = 20

# Имитация нормального распределения

**Этот способ также называется Нормированной суммой N чисел**

Суть способа в том, чтобы взять несколько чисел в диапазоне матожидания+- 3\*СКО и вернуть их среднее арифметическое. Тогда результат стремится к середине диапазона, что похоже на Нормальное распределение.

### Пример реализации

def pseudo\_norm(average = 50, sko = 3):

    """Generate a value in a normal distribution"""

    min=average-3\*sko

    max = average+3\*sko

    count= 12

    values =  sum([random.randint(min, max) for x in range(count)])

    return values/count

count – величина, которая определяет, как близко к матожиданию будут сгруппированы числа

# Задание

Создать по три массива для Нормального распределения и Равномерного используя модуль random и методы random.gauss(average, sko) и random.uniform(min, max). Воспользоваться ascii\_plot() и проанализировать резултаты. В качестве исходных данных – выбрать по варианту из таблицы (вариант = номер в списке mod 4)

|  |  |  |  |
| --- | --- | --- | --- |
| Вариант 1 | Вариант 2 | Вариант 3 | Вариант 4 |
| Норм 1: avg 10, sko 10 | Норм 1: avg 100, sko 30 | Норм 1: avg 50, sko 10 | Норм 1: avg 2, sko 30 |
| Норм 2: avg 20, sko 7 | Норм 2: avg 1000, sko 430 | Норм 2: avg 13, sko 7 | Норм 2: avg 30, sko 30 |
| Норм 3: avg 100, sko 1 | Норм 3: avg 22, sko 2 | Норм 3: avg 1, sko 1 | Норм 3: avg 150, sko 15 |
| Равном 1: min 1, max 20 | Равном 1: min 1, max 20 | Равном 1: min 1000, max 200000 | Равном 1: min 0, max 1 |
| Равном 2: min 3, max 5 | Равном 2: min 0, max 1 | Равном 2: min 30, max 51 | Равном 2: min 3, max 5 |
| Равном 3: min 0, max 1 | Равном 3: min 3, max 5 | Равном 3: min 0, max 1 | Равном 3: min 30, max 51 |

# Приложение

Листинг тестовой программы, которая выводит 10 графиков для нормального распределения. Аналогично для random.uniform(min, max)

import random

def pseudo\_norm(average = 50, sko = 3):

    """Generate a value in a normal distribution"""

    min=average-3\*sko

    max = average+3\*sko

    count= 12

    values =  sum([random.randint(min, max) for x in range(count)])

    return values/count

def ascii\_plot(dist, num\_of\_intervals=20):

    """makes plot and output base information about 2d list"""

    counted= [0] \* num\_of\_intervals

    minimal = min(dist)

    interval = (max(dist)-minimal)/num\_of\_intervals

    print(f"max number = {max(dist)}")

    print(f"min number= {min(dist)}")

    print(f"interval = {interval}")

    print(f"average = {sum(dist)/len(dist)}")

    print('\n')

    for i in dist:

        counted[round((i-minimal)/interval)-1]+=1

        #print(f"{i} goes to {round((i-minimal)/interval)} interval")

    sizeX= num\_of\_intervals

    sizeY= max(counted)

    plot =[]

    for i in range(sizeY):

        plot.append([0]\*sizeX)

    for x in range(sizeX):

        for y in range(counted[x]):

            plot[sizeY-y-1][x]=counted[x]

    #show

    labelY=sizeY

    for i in range(sizeY):

        string =f"{labelY}\t"

        labelY-=1

        for char in plot[i]:

            if char==0:

                string+=" "

            else:

                string+="|"

        print(string)

    underscore= "Y/X\t"

    for i in range(sizeX):

        underscore+=str('-')

    print(underscore)

if \_\_name\_\_=="\_\_main\_\_":

    for i in range(10):

        dist = [random.gauss(100, 20) for x in range(100)]

#dist = [random.uniform(1, 100) for x in range(100)]

    #dist = [random.pseudo\_norm(50, 3) for x in range(100)]

        ascii\_plot(dist)

        print('\n')

        print(dist)