МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені Тараса Шевченка
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра програмних систем і технологій

Дисципліна
« Ймовірнісні основи програмної інженерії»

Лабораторна робота № 1

Виконав:	Ільницький Олександр Віталійович	**Перевірила**:	
Група	ІПЗ-24	Дата перевірки	
Форма навчання	денна	Оцінка	
Спеціальність	121		

2022

Тема – Центральні тенденції та міра дисперсії
Мета – навчитись використовувати на практиці набуті знання про центральні тенденції та міри.

Завдання

1. Побудувати таблицю частот та сукупних частот для переглянутих фільмів.

Визначити фільм, який був переглянутий частіше за інші.

2. Знайти Моду та Медіану заданої вибірки.

3. Порахувати Дисперсію та Середнє квадратичне відхилення розподілу.

4. Побудувати гістограму частот для даного розподілу.

5. Зробити висновок з вигляду гістограми, про закон розподілу.

Дано:

Вхідні данні – текстовий файл з данними, які потрібно обчислити

Що потрібно зробити описано в завданні.

Дії для виконання завдань

Для побудови таблиці знайдемо комулятивну частоту та релятивну

Код комулятивної частоти

```
c = 0
def cumulative_freq(item):
    global c
    c += item
    return str(c)
```

Код релятивної частоти

```
def relative_freq(num, Sum):
    return str(num / Sum)
```

Pseudocode:

cumulativeSum=0
Function Cumulative():

cumulative = cumulative + item

return cumulative

Function Relative(num, SumOfNumbers):

Return num/SumOfNumbers

Pseudocode End.

Для обчислення моди :

Знаходимо число яке зустрічалось найчастіше через проходження через масив данних

Код для обчислення моди де printf функція для запису в файл

```
printf(f'\nModa: {max((num.count, num.number) for num in data)}')
```

Pseudocode:

Function Count(array, number):

Int counter = 0

For each num in array:

If (num == number)

Counter+=1;

Return counter;

arr = fileData

for each number in arr:

    Print(“number” + Count(arr, number))

Pseudocode End.

Для обчислення Медіани :

Якщо маємо парну кількість чисел, то беремо два значення з середини та ділимо на 2

Якщо непарна, то беремо середину

Код для обисления медіани

```
Mediana = graph[math.floor((len(graph) / 2))]
```

Pseudocode:

File.write = WriteFile();

Array = fileData

Function Count(array):

    Counter = 0

    For each num in array:

        Counter+=1;

If Array.Length == 0:

    ArrLen = Array.length

    WriteFile(

(Array[Math.ceil(ArrLen/2)] + Array[Math.flor(ArrLen/2)]) / 2)

Else:

    WriteFile(Array[ArrLen/2])Pseudocode End.

Для обичислення Дисперсії:

Знаходиме середнє значення всіх данних у файлі , підносимо до квадрату число з файлу від якого віднімаємо середнє значення, потім додаємо всі ці числа та ділимо на кількість чисел у файлі.

Код для обчислення дисперсії

```
def dispersion(array):
    arr = []
    for item in array:
        arr.append(item)
    n = len(arr)
    mid = sum(map(int, arr)) / n
    print(mid)
    deviations = [(float(x) - mid) ** 2 for x in arr]
    variance = sum(deviations) / n
    return round(variance, 3)
```

Pseudocode:

Function Sum(array):

    Int Sum =0

    For each num in array:

        Sum += num;

    Return Sum

FindVariance():

    Arr = fileData

    Mid = Sum(array)/ Arr.Length

    Deviations = []

    For each num in Arr:

        Deviations.append((num – mid)**2)

    Variance = Sum(Deviations) / Arr.Length

    Return Variance

Pseudocode End.

Для обичислення Середнього квадратичного відхилення розподілу:

Знаходимо квадрат від дисперсії

Код для обчислення

Де disputer результат виконання функції для обчислення дисперсії

```
disputer = dispersion(graph)
mid_sqr = math.sqrt(disputer)
```

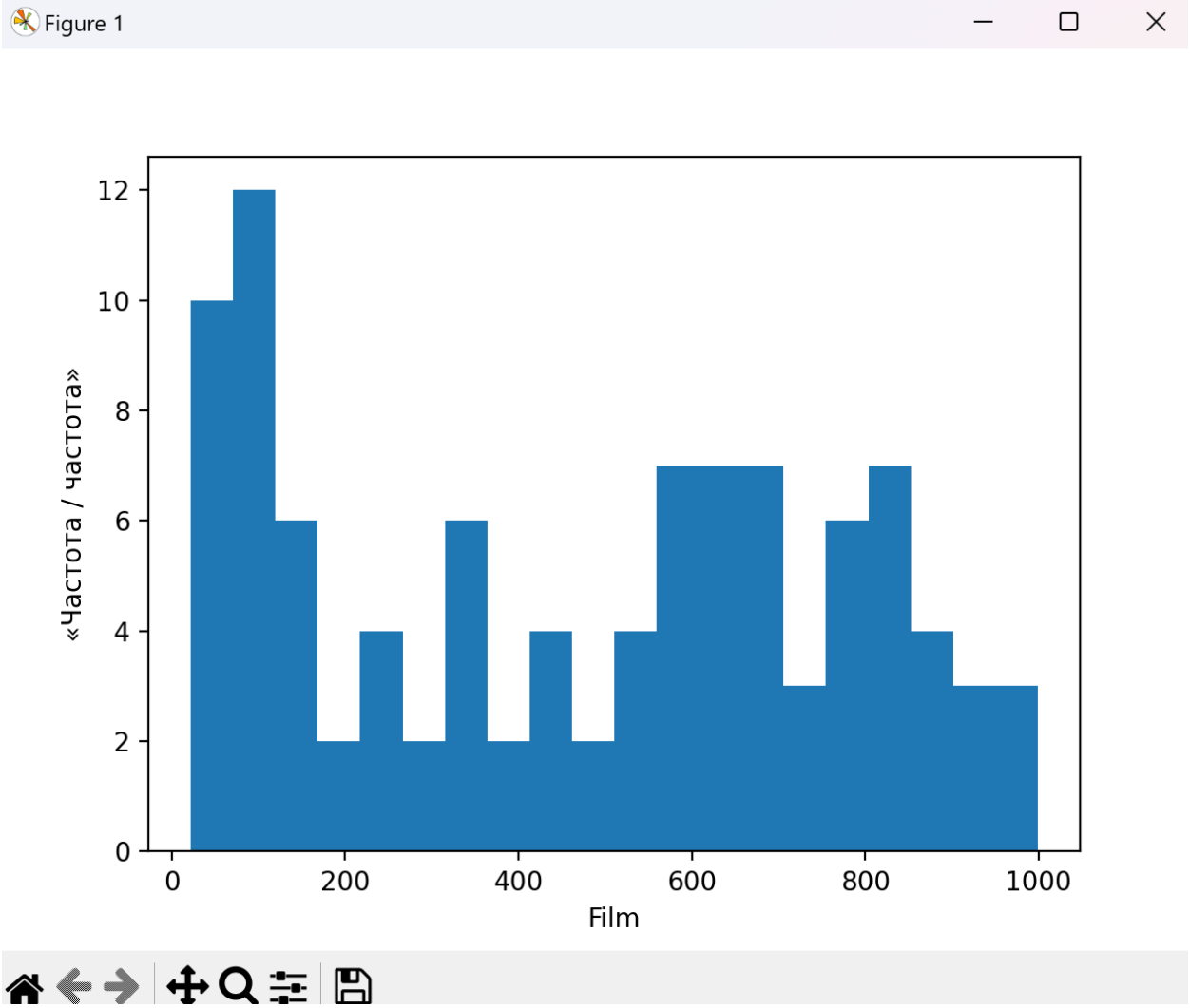
Function AvgSqrtVariance():

Return Math.Sqrt(FindVariance())

Pseudocode End.

Код програми

Побудувати Гістограму



Бачимо що програма працює добре та виводить гістограму.

Файл у якому відбувався запис частот та сукупних частот

input_10.txt | input_100.txt | Visual.py | file_writer.py | main.py | task.txt | Numbers.py

```
item      ||      fi(frequency)      ||         Rf      ||      Fi(cumulative freq.)


 22       ||            4       ||       0.00046552964577426046      ||           22
 46       ||            2       ||       0.00097338016843709     ||          68
 47       ||            1       ||       0.0009945406068813745       ||          115
 51       ||            3       ||       0.0010791823606585127       ||          166
 71       ||            1       ||       0.0015023911295442042       ||          237
 77       ||            1       ||       0.0016293537602099116       ||          314
 79       ||            2       ||       0.0016716746370984807       ||          393
 80       ||            1       ||       0.0016928350755427651       ||          473
 91       ||            1       ||       0.0019255998984298954       ||          564
 97       ||            1       ||       0.002052562529095603        ||          661
 99       ||            1       ||       0.002094883405984172        ||          760
100       ||            1       ||       0.0021160438444284566       ||          860
103       ||            1       ||       0.0021795251597613104       ||          963
119       ||            2       ||       0.0025180921748698634       ||          1082
146       ||            1       ||       0.0030894240128655466       ||          1228
147       ||            1       ||       0.003110584451309831        ||          1375
154       ||            1       ||       0.003258707520419823        ||          1529
162       ||            2       ||       0.0034279910279740996       ||          1691
168       ||            1       ||       0.003554953658639807        ||          1859
193       ||            1       ||       0.004083964619746921        ||          2052
198       ||            1       ||       0.004189766811968344        ||          2250
225       ||            1       ||       0.004761098649964027        ||          2475
250       ||            1       ||       0.005290109611071141        ||          2725
251       ||            1       ||       0.005311270049515426        ||          2976
255       ||            1       ||       0.005395911803292564        ||          3231
269       ||            1       ||       0.005692157941512548        ||          3500
288       ||            1       ||       0.006094206271953955        ||          3788
317       ||            1       ||       0.006707858986838207        ||          4105
354       ||            1       ||       0.007490795209276736        ||          4459
355       ||            1       ||       0.007511955647721021        ||          4814
359       ||            1       ||       0.0075965974014981594       ||          5173
361       ||            1       ||       0.007638918278386728        ||          5534
362       ||            1       ||       0.007660078716831013        ||          5896
382       ||            1       ||       0.008083287485716704        ||          6278
384       ||            1       ||       0.008125608362605273        ||          6662
414       ||            1       ||       0.00876042151593381     ||          7076
429       ||            1       ||       0.009077828092598078        ||          7505
447       ||            1       ||       0.0094587159845952      ||          7952
450       ||            1       ||       0.009522197299928054        ||          8402
```

Випробовування алгоритму

Для успішного виконання програми нам потрібно створити

Масив у якому є n чисел

Для знаходження суми чисел та кількості потрібно пройтись по масиву з n чисел один раз

Висновки:

Начилися робити таблиці частот та сукупних частот

Обчислювати необхідні для неї данні та використовувати для побудови діаграми.

Повний код програми

Файл main:

```
from file_writer import writeFile
from Visual import showGraph
datafile = open('./task_01_data/input_100.txt')
data = []
graph = []

for line in datafile:
    graph.append(int(line.strip()))
graph.sort()
writeFile(data,graph)
showGraph(graph)
```

Файл file_write:

```
from Numbers import replaceStr, Numbers
import math

task = open('task.txt', 'w')
task.writelines(
    '{0:>5} \t\t||\t\t {1:>5} \t\t||\t\t {2:>5} \t\t||\t\t {3:>5} \n\n'.format('item', 'fi(frequency)', 'Rf',
                                                                              'Fi(cumulative freq.)'))


def printf(data):
    task.writelines(data)


def dispersion(array):
    arr = []
    for item in array:
        arr.append(item)
    n = len(arr)
    mid = sum(map(int, arr)) / n
    print(mid)
    deviations = [(float(x) - mid) ** 2 for x in arr]
    variance = sum(deviations) / n
    return round(variance, 3)


def relative_freq(num, Sum):
    return str(num / Sum)
```

```
c = 0
def cumulative_freq(item):
    global c
    c += item
    return str(c)


def count_data_appear(data, graph):
    for i, num in enumerate(graph):
        num_of_number = graph.count(num)
        if graph[i] != graph[i - 1]:
            data.append(Numbers(num, num_of_number))


def write_data(data, arr):
    Sum = sum(map(int, arr))
    for i in range(len(data)):
        print(i)
        line = data[i].__str__()
        printf(
            f'{str(line)}\t\t || \t\t{relative_freq(data[i].number, Sum):>5}\t\t || \t\t {cumulative_freq(data[i].number):>5}')
```

```
def writeFile(data, graph):
    print(graph)
    count_data_appear(data, graph)
    write_data(data, graph)
    disputer = dispersion(graph)
    mid_sqr = math.sqrt(disputer)
    Mediana = graph[math.floor((len(graph) / 2))]
    printf(f'\n\n\nMedian:{Mediana}')
    printf(f"\nMid square:{mid_sqr}")
    printf(f'\nModa: {max((num.count, num.number) for num in data)}')
    printf(f'\nVariance: {disputer}')
```

Файл Visual:

```
import matplotlib.pyplot as plt


def showGraph(data_a, ):
    plt.hist(data_a, bins=20 )
    plt.xlabel("Film")
    plt.ylabel("«Частота / частота»")
    plt.show()
```

Файл Numbers:

```python
def replaceStr(string):
    return str(string).replace('\n', '')


class Numbers:
    number = 0
    count = 0

    def __init__(self, number, count ):
        self.number = number
        self.count = count

    def __str__(self):

        return f'\n{self.number:>5}\t\t || \t\t {self.count:>5}'
```