|  |  |  |
| --- | --- | --- |
| Теория вероятностей и математическая статистика | | |
| Лабораторная работа №5  «Первичная обработка статистических данных» | ФИО студента | Сидский Н.А. |
| Группа | ИВТ-261 |
| Дата выполнения |  |
| Дата отчета |  |
| Оценка |  |
| Подпись преподавателя |  |

Задание №1

Код:

from tkinter import filedialog  
from matplotlib import pyplot as plt  
from PIL import Image  
  
plt.rcParams["font.size"] = 16  
plt.rcParams["figure.figsize"] = (9,7)  
  
*# Ввод из файла*while(True):  
 fname = filedialog.askopenfilename(title="Выберите файл с данными")  
 if fname:  
 f = open(fname)  
 data = [float(x) for x in f.read().split()]  
 break  
*# Вывести введенные значения*print("Введенные данные:")  
[print(f"{x:<4} ", end='') for x in data]  
print()  
print("\nВыберите интересующие вас пункты:\n\  
 1. Составление вариационного ряда и вывод его на экран;\n\  
 2. Составление и вывод на экран статистического ряда частот и относительных частот, построение полигона для каждого из них;\n\  
 3. Нахождение эмпирической функции распределения F\*(x) с выводом на экран определения эмпирической функции распределения и ее конкретного вида для заданного статистического распределения, построение графика F\*(х);\n\  
 4. Вывод на экран формул для вычисления числовых характеристик выборки: ̄xᵦ, Dᵦ, σᵦ, S, их вычисление и вывод на экран полученных результатов;\n")  
choice = input("Введите необходимые номера и нажмите enter: ")  
  
*# Найти и вывести вариационный ряд*varser = data  
varser.sort()  
if("1" in choice):  
 print("\n1.")  
 print("Вариационный ряд:")  
 [print(f"{x:<4} ", end='') for x in varser]  
 input("\nНажмите enter чтобы продолжить\n")  
  
*# Частоты*varseries = list(set(varser)) *# удалить дубликаты*varseries.sort()  
ns = [data.count(x) for x in varseries] *# найти частоты*n = len(data)  
ws = [x/n for x in ns]  
if("2" in choice):  
 print("\n2.")  
 print("Статистический ряд частот:")  
 print((len(varseries) \* 6 + 5) \* '▁')  
 print("┃xᵢ ┃", end='')  
 [print(f"{x:^5}", end='┃') for x in varseries]  
 print("\n┃nᵢ ┃", end='')  
 [print(f"{x:^5}", end='┃') for x in ns]  
 print('\n' + (len(varseries) \* 6 + 5) \* '▔')  
*# Относительные частоты* print("Статистический ряд относительных частот:")  
 print((len(varseries) \* 6 + 5) \* '▁')  
 print("┃xᵢ ┃", end='')  
 [print(f"{x:^5}", end='┃') for x in varseries]  
 print("\n┃wᵢ ┃", end='')  
 [print(f"{x:^5}", end='┃') for x in ws]  
 print('\n' + (len(varseries) \* 6 + 5) \* '▔')  
 *# Найти и вывести полигоны частот* ax = plt.figure().add\_subplot(1, 1, 1)  
 ax.spines['left'].set\_position(('data', max(varseries[0]-1, min(0.0, varseries[-1]+1))))  
 ax.spines['bottom'].set\_position(('data', 0.0))  
 ax.spines['right'].set\_visible(False)  
 ax.spines['top'].set\_visible(False)  
 ax.plot(1, 0, ">k", transform=ax.get\_yaxis\_transform(), clip\_on=False, ms=8)  
 ax.plot(0, 1, "^k", transform=ax.get\_xaxis\_transform(), clip\_on=False, ms=8)  
 plt.xlim((varseries[0]-1, varseries[-1]+1))  
 plt.ylim((0, max(ns)+max(ns)/100))  
 plt.grid() *# частоты* plt.plot(varseries, ns, linewidth=3)  
 plt.title('Полигон статистического ряда частот')  
 plt.xlabel('xᵢ')  
 plt.ylabel('nᵢ')  
 plt.xticks(varseries)  
 plt.yticks(ns)  
 *# Относительных частот* ax = plt.figure().add\_subplot(1, 1, 1)  
 ax.spines['left'].set\_position(('data', max(varseries[0]-1, min(0.0, varseries[-1]+1))))  
 ax.spines['bottom'].set\_position(('data', 0.0))  
 ax.spines['right'].set\_visible(False)  
 ax.spines['top'].set\_visible(False)  
 ax.plot(1, 0, ">k", transform=ax.get\_yaxis\_transform(), clip\_on=False, ms=8)  
 ax.plot(0, 1, "^k", transform=ax.get\_xaxis\_transform(), clip\_on=False, ms=8)  
 plt.xlim((varseries[0]-1, varseries[-1]+1))  
 plt.ylim((0, max(ws)+max(ws)/100))  
 plt.grid()  
 plt.plot(varseries, ws, linewidth=3)  
 plt.title('Полигон статистического ряда относительных частот')  
 plt.xlabel('xᵢ')  
 plt.ylabel('wᵢ')  
 plt.xticks(varseries)  
 plt.yticks(ws)  
 plt.show()  
 input("Нажмите enter чтобы продолжить\n")  
  
*# Эмпирическая функция распределения*if("3" in choice):  
 print("\n3.")  
 print('Эмпирической функцией распределения называется функция вида F\*(x) действительного аргумента x, значения которой в каждой точке равно накопленной относительной часоте событий, при (X < x).\n nₓ\nФормула: F\*(x) = ———.\n n')  
 fstar = [0]  
 for i in range(1, len(ws)):  
 fstar.append(round(ws[i-1]+fstar[i-1], 5))  
 fstar.append(1)  
 print(f" ╭─\n │0 при x < {varseries[0]}")  
 for i in range(1, len(ws)):  
 if i == len(ws)//2:  
 print(f"F\*(x) = ━┥{fstar[i]:<4} при {varseries[i-1]:>2} ≤ x < {varseries[i]:<2}")  
 else:  
 print(f" │{fstar[i]:<4} при {varseries[i-1]:>2} ≤ x < {varseries[i]:<2}")  
 print(f" │1 при x > {varseries[-1]}\n ╰─")  
 *# график* ax = plt.figure().add\_subplot(1, 1, 1)  
 ax.spines['left'].set\_position(('data', max(varseries[0]-1, min(0.0, varseries[-1]+1))))  
 ax.spines['bottom'].set\_position(('data', 0.0))  
 ax.spines['right'].set\_visible(False)  
 ax.spines['top'].set\_visible(False)  
 ax.plot(1, 0, ">k", transform=ax.get\_yaxis\_transform(), clip\_on=False, ms=8)  
 ax.plot(0, 1, "^k", transform=ax.get\_xaxis\_transform(), clip\_on=False, ms=8)  
 plt.xlim((varseries[0]-1, varseries[-1]+1))  
 plt.ylim((0, 1.05))  
 plt.grid() *# относительные частоты* plt.hlines(fstar, [varseries[0]-1] + varseries, varseries + [varseries[-1]+1], linewidth=3, clip\_on=False)  
 tmp = fstar.pop(0)  
 plt.scatter(varseries, fstar, marker='<', s=100)  
 fstar.insert(0, tmp)  
 plt.suptitle('Эмпирическая функция распределения')  
 plt.xlabel('xᵢ')  
 plt.ylabel('F\*(xᵢ)')  
 plt.xticks(varseries)  
 plt.yticks([0] + fstar + [1])  
 plt.show()  
 input("Нажмите enter чтобы продолжить\n")  
  
*# Характеристики выборки*if("4" in choice):  
 print("\n4.")  
 print("Числовые характеристики выборки:")  
 xmean = sum([x\*y for x,y in zip(data, ns)])/n  
 dmean = sum([((xmean-x)\*\*2)\*y for x,y in zip(data, ns)])/n  
 sigmamean = dmean\*\*0.5  
 s = ((n\*dmean)/(n-1))\*\*0.5  
 print(f"- ̄xᵦ = {xmean}\n- Dᵦ = {dmean}\n- σᵦ = {sigmamean}\n- S = {s}")  
 *# формулы* img = Image.open("numchars.png")  
 plt.figure()  
 plt.axis("off")  
 plt.imshow(img)  
 plt.show()  
 input("Нажмите enter чтобы продолжить\n")

Используемый список:

21 21 21 21 21 21 21 21

-7 -7 -7 -7

2 2 2 2 2 2 2 2 2 2 2

14 14 14 14 14 14 14 14 14 14

-2 -2 -2 -2 -2 -2 -2 -2 -2

4 4 4 4 4 4 4

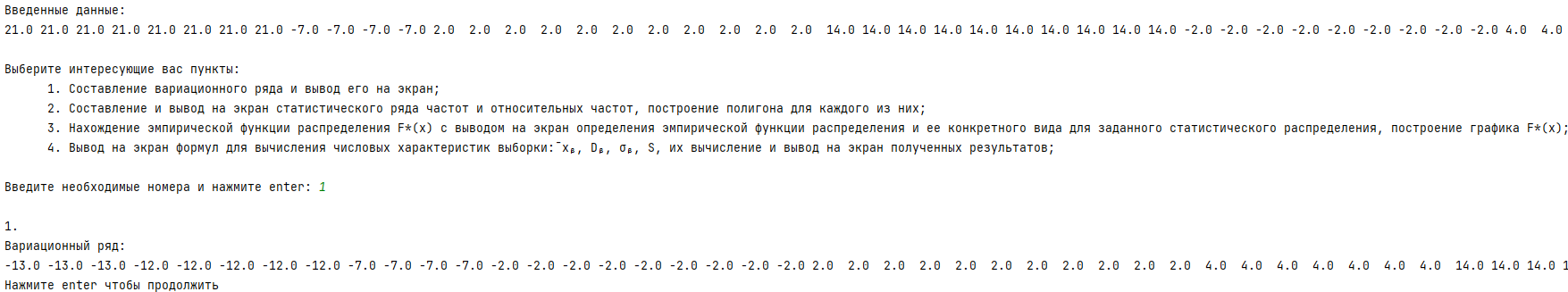
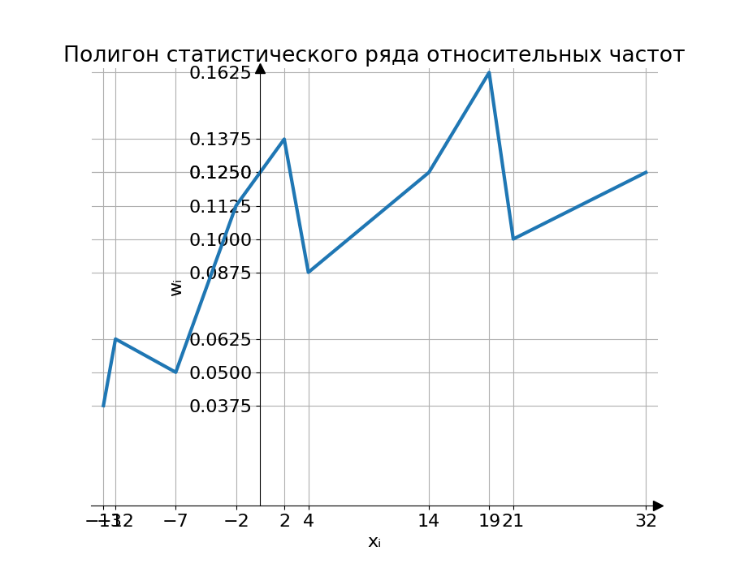
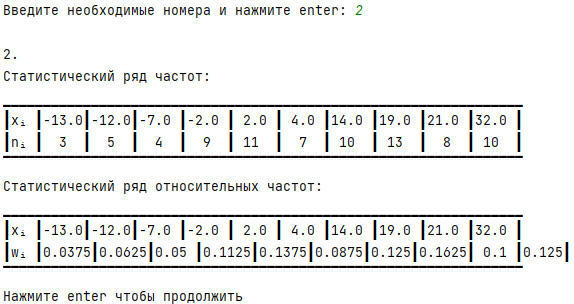
-12 -12 -12 -12 -12

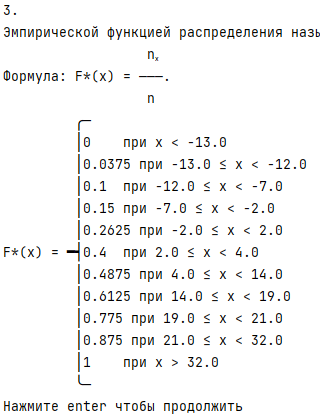
19 19 19 19 19 19 19 19 19 19 19 19 19

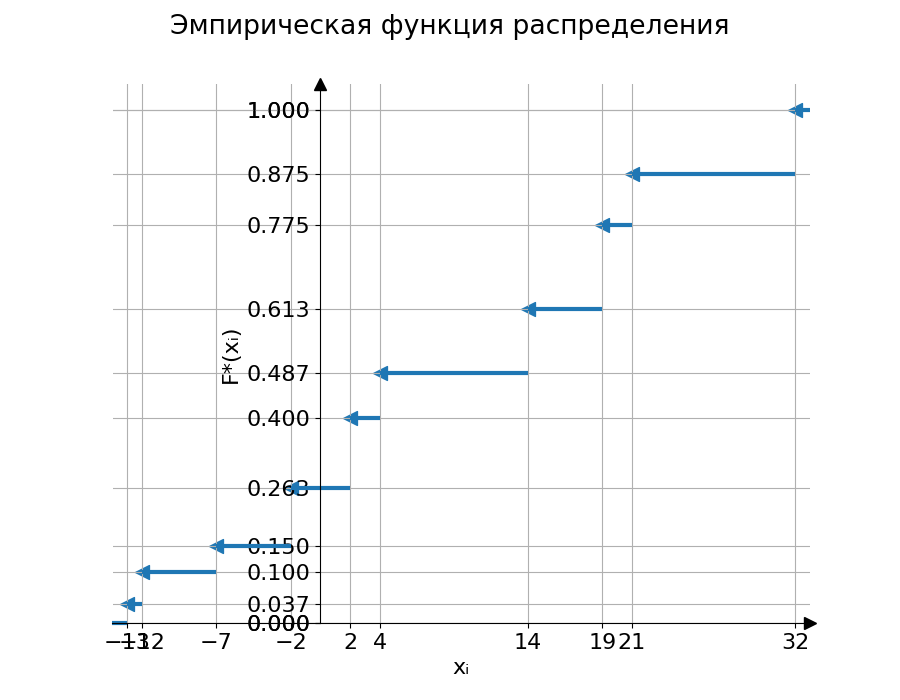
32 32 32 32 32 32 32 32 32 32

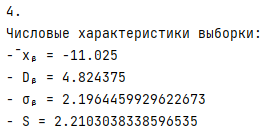
-13 -13 -13

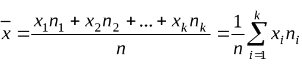
Решения:



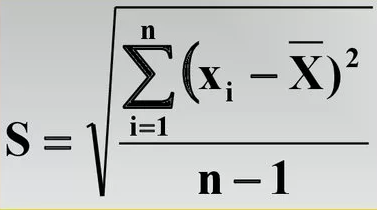










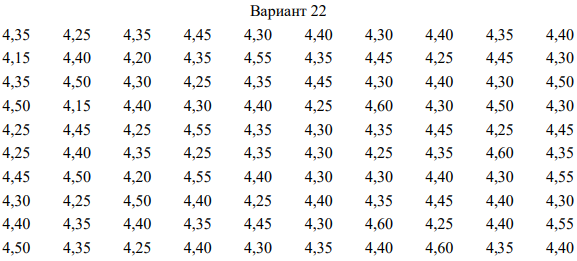


Задание 2

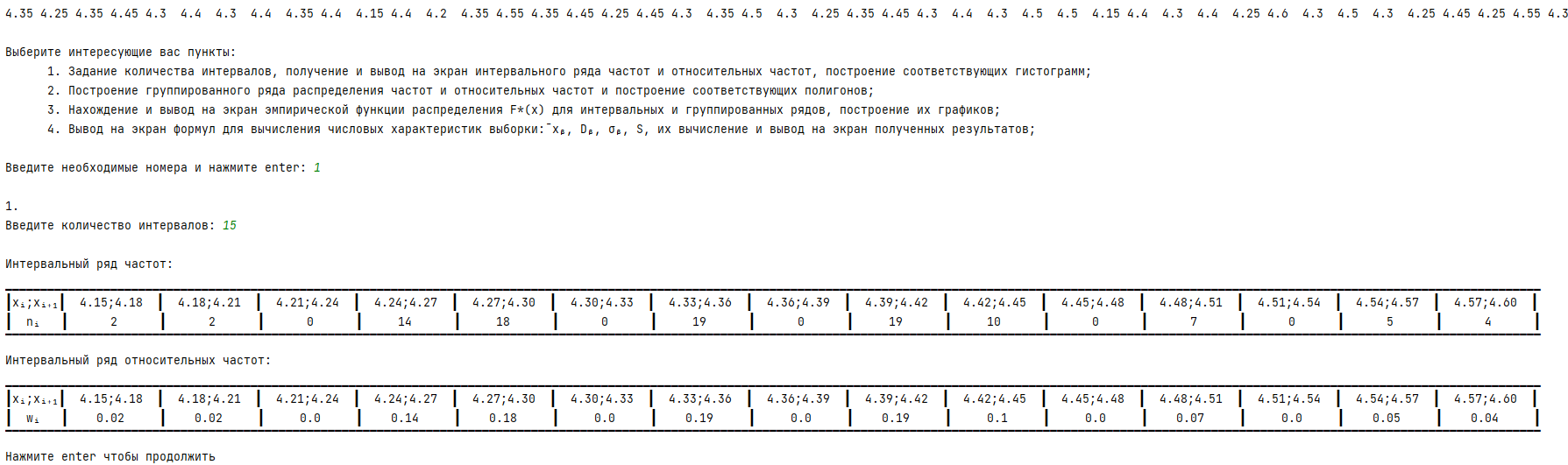
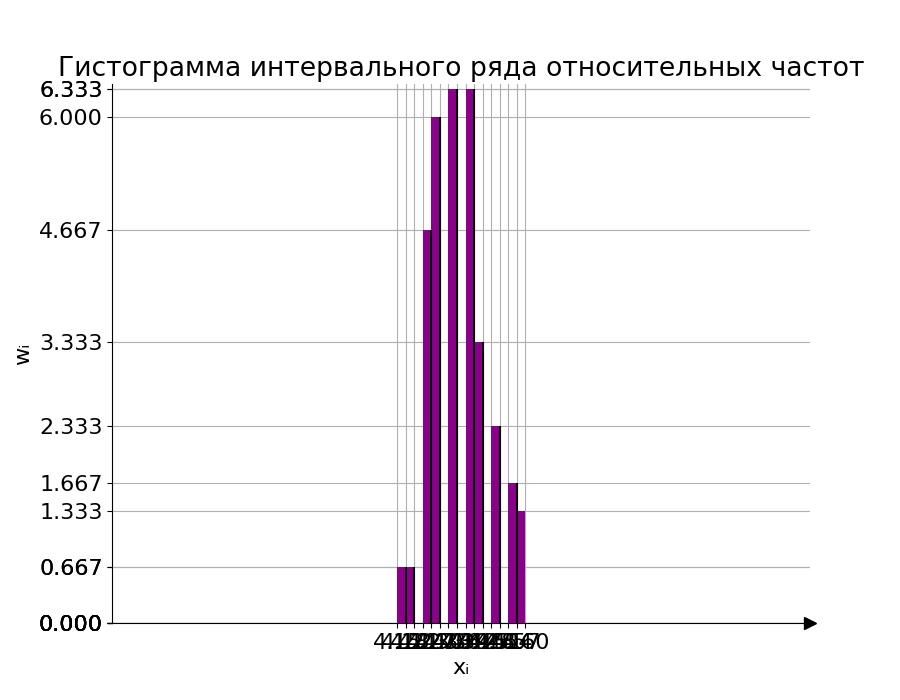
Код:

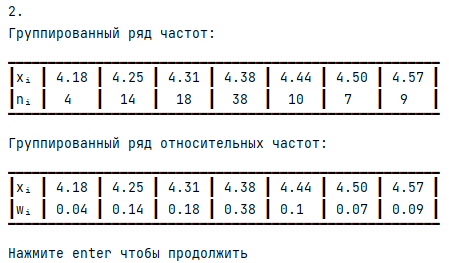
from tkinter import filedialog  
from matplotlib import pyplot as plt  
from PIL import Image  
from math import floor, log10  
  
plt.rcParams["font.size"] = 16  
plt.rcParams["figure.figsize"] = (9,7)  
  
*# Ввод из файла*while(True):  
 fname = filedialog.askopenfilename(title="Выберите файл с данными")  
 if fname:  
 f = open(fname)  
 data = [float(x) for x in f.read().split()]  
 break  
*# Вывести введенные значения*print("Введенные данные:")  
[print(f"{x:<4} ", end='') for x in data]  
print()  
print("\nВыберите интересующие вас пункты:\n\  
 1. Задание количества интервалов, получение и вывод на экран интервального ряда частот и относительных частот, построение соответствующих гистограмм;\n\  
 2. Построение группированного ряда распределения частот и относительных частот и построение соответствующих полигонов;\n\  
 3. Нахождение и вывод на экран эмпирической функции распределения F\*(х) для интервальных и группированных рядов, построение их графиков;\n\  
 4. Вывод на экран формул для вычисления числовых характеристик выборки: ̄xᵦ, Dᵦ, σᵦ, S, их вычисление и вывод на экран полученных результатов;\n")  
choice = input("Введите необходимые номера и нажмите enter: ")  
*#choice = "3"*varseries = sorted(list(set(data)))  
xmax = max(varseries)  
xmin = min(varseries)  
intervals = []  
  
*# Интервалы*if("1" in choice):  
 print("\n1.")  
 intervalAmount = int(input("Введите количество интервалов: "))  
 *#intervals = 12*else:  
 intervalAmount = floor(1 + 3.322 \* log10(len(data)))  
  
step = (xmax-xmin)/intervalAmount  
counter = xmin;  
xnmax = []  
xnmax.append(xmin)  
for i in range(intervalAmount-1):  
 intervals.append([x for x in data if counter <= x and x < counter + step])  
 counter+=step  
 xnmax.append(counter)  
intervals.append([x for x in data if counter <= x])  
xnmax.append(xmax)  
ns = [len(interval) for interval in intervals]  
mainn = len(data)  
ws = [n / mainn for n in ns]  
  
*# Частоты*if("1" in choice):  
 print("\nИнтервальный ряд частот:")  
 print((intervalAmount \* 14 + 9) \* '▁')  
 print("┃xᵢ;xᵢ₊₁┃", end='')  
 counter = xmin;  
 for i in range(intervalAmount-1):  
 print(f"{counter:>6.2f};{counter+step:<6.2f}", end='┃')  
 counter+=step  
 print(f"{counter:>6.2f};{xmax:<6.2f}", end='┃')  
 print("\n┃ nᵢ ┃", end='')  
 [print(f"{n:^13}", end='┃') for n in ns]  
 print('\n' + (intervalAmount \* 14 + 9) \* '▔')  
 *# Относительные частоты* print("Интервальный ряд относительных частот:")  
 print((intervalAmount \* 14 + 9) \* '▁')  
 print("┃xᵢ;xᵢ₊₁┃", end='')  
 counter = xmin;  
 for i in range(intervalAmount-1):  
 print(f"{counter:>6.2f};{counter+step:<6.2f}", end='┃')  
 counter+=step  
 print(f"{counter:>6.2f};{xmax:<6.2f}", end='┃')  
 print("\n┃ wᵢ ┃", end='')  
 [print(f"{w:^13}", end='┃') for w in ws]  
 print('\n' + (intervalAmount \* 14 + 9) \* '▔')  
*# Гистограммы  
 # Частоты* histns = [n/step for n in ns]  
 ax = plt.figure().add\_subplot(1, 1, 1)  
 ax.spines['left'].set\_position(('data', max(varseries[0]-1, min(0.0, varseries[-1]+1))))  
 ax.spines['bottom'].set\_position(('data', 0.0))  
 ax.spines['right'].set\_visible(False)  
 ax.spines['top'].set\_visible(False)  
 ax.plot(1, 0, ">k", transform=ax.get\_yaxis\_transform(), clip\_on=False, ms=8)  
 ax.plot(0, 1, "^k", transform=ax.get\_xaxis\_transform(), clip\_on=False, ms=8)  
 plt.xlim((varseries[0]-1, varseries[-1]+1))  
 plt.ylim((0, max(histns)+max(histns)/100))  
 plt.grid() *# частоты* plt.stairs(histns, xnmax, fill=True, facecolor="indigo", zorder=3)  
 plt.vlines(xnmax[1:-1], 0, histns[:-1], color="black", zorder=4)  
 plt.title('Гистограмма интервального ряда частот')  
 plt.xlabel('xᵢ')  
 plt.ylabel('nᵢ')  
 plt.xticks(xnmax)  
 plt.yticks(histns)  
 *# Относительные частоты* histws = [w/step for w in ws]  
 ax = plt.figure().add\_subplot(1, 1, 1)  
 ax.spines['left'].set\_position(('data', max(varseries[0]-1, min(0.0, varseries[-1]+1))))  
 ax.spines['bottom'].set\_position(('data', 0.0))  
 ax.spines['right'].set\_visible(False)  
 ax.spines['top'].set\_visible(False)  
 ax.plot(1, 0, ">k", transform=ax.get\_yaxis\_transform(), clip\_on=False, ms=8)  
 ax.plot(0, 1, "^k", transform=ax.get\_xaxis\_transform(), clip\_on=False, ms=8)  
 plt.xlim((varseries[0]-1, varseries[-1]+1))  
 plt.ylim((0, max(histws)+max(histws)/100))  
 plt.grid() *# частоты* plt.stairs(histws, xnmax, fill=True, facecolor="darkmagenta", zorder=3)  
 plt.vlines(xnmax[1:-1], 0, histws[:-1], color="black", zorder=4)  
 plt.title('Гистограмма интервального ряда относительных частот')  
 plt.xlabel('xᵢ')  
 plt.ylabel('wᵢ')  
 plt.xticks(xnmax)  
 plt.yticks(histws)  
 plt.show()  
 input("Нажмите enter чтобы продолжить\n")  
  
*# Группы*xs = [round((xnmax[i] + xnmax[i+1])/2, 2) for i in range(intervalAmount)]  
if("2" in choice):  
 print("\n2.")  
 print("Группированный ряд частот:")  
 print((len(xs) \* 7 + 5) \* '▁')  
 print("┃xᵢ ┃", end='')  
 [print(f"{x:^6.2f}", end='┃') for x in xs]  
 print("\n┃nᵢ ┃", end='')  
 [print(f"{n:^6}", end='┃') for n in ns]  
 print('\n' + (len(xs) \* 7 + 5) \* '▔')  
 *# Относительные частоты* print("Группированный ряд относительных частот:")  
 print((len(xs) \* 7 + 5) \* '▁')  
 print("┃xᵢ ┃", end='')  
 [print(f"{x:^6.2f}", end='┃') for x in xs]  
 print("\n┃wᵢ ┃", end='')  
 [print(f"{w:^6}", end='┃') for w in ws]  
 print('\n' + (len(xs) \* 7 + 5) \* '▔')  
*# Найти и вывести полигоны частот* ax = plt.figure().add\_subplot(1, 1, 1)  
 ax.spines['left'].set\_position(('data', max(varseries[0]-1, min(0.0, varseries[-1]+1))))  
 ax.spines['bottom'].set\_position(('data', 0.0))  
 ax.spines['right'].set\_visible(False)  
 ax.spines['top'].set\_visible(False)  
 ax.plot(1, 0, ">k", transform=ax.get\_yaxis\_transform(), clip\_on=False, ms=8)  
 ax.plot(0, 1, "^k", transform=ax.get\_xaxis\_transform(), clip\_on=False, ms=8)  
 plt.xlim((varseries[0]-1, varseries[-1]+1))  
 plt.ylim((0, max(ns)+max(ns)/100))  
 plt.grid() *# частоты* plt.plot(xs, ns, linewidth=3)  
 plt.title('Полигон статистического ряда частот')  
 plt.xlabel('xᵢ')  
 plt.ylabel('nᵢ')  
 plt.xticks(xs)  
 plt.yticks(ns)  
 *# Относительных частот* ax = plt.figure().add\_subplot(1, 1, 1)  
 ax.spines['left'].set\_position(('data', max(varseries[0]-1, min(0.0, varseries[-1]+1))))  
 ax.spines['bottom'].set\_position(('data', 0.0))  
 ax.spines['right'].set\_visible(False)  
 ax.spines['top'].set\_visible(False)  
 ax.plot(1, 0, ">k", transform=ax.get\_yaxis\_transform(), clip\_on=False, ms=8)  
 ax.plot(0, 1, "^k", transform=ax.get\_xaxis\_transform(), clip\_on=False, ms=8)  
 plt.xlim((varseries[0]-1, varseries[-1]+1))  
 plt.ylim((0, max(ws)+max(ws)/100))  
 plt.grid()  
 plt.plot(xs, ws, linewidth=3)  
 plt.title('Полигон статистического ряда относительных частот')  
 plt.xlabel('xᵢ')  
 plt.ylabel('wᵢ')  
 plt.xticks(xs)  
 plt.yticks(ws)  
 plt.show()  
 input("Нажмите enter чтобы продолжить\n")  
  
*# Функции распределения*if("3" in choice):  
 fstar = [0]  
 for i in range(1, len(ws)):  
 fstar.append(round(ws[i-1]+fstar[i-1], 5))  
 fstar.append(1)  
 *# Интервальный* print("F\*(x) Интервального ряда")  
 print(f" ╭─\n │0 при x < {xnmax[0]}")  
 for i in range(1, len(fstar)):  
 if i == len(fstar)//2:  
 print(f"F\*(x) = ━┥{fstar[i]:<4} при {xnmax[i-1]:>5.2f} ≤ x < {xnmax[i]:<5.2f}")  
 else:  
 print(f" │{fstar[i]:<4} при {xnmax[i-1]:>5.2f} ≤ x < {xnmax[i]:<5.2f}")  
 print(f" │1 при x > {xnmax[-1]}\n ╰─")  
 *# график* ax = plt.figure().add\_subplot(1, 1, 1)  
 ax.spines['left'].set\_position(('data', max(xnmax[0]-1, min(0.0, xnmax[-1]+1))))  
 ax.spines['bottom'].set\_position(('data', 0.0))  
 ax.spines['right'].set\_visible(False)  
 ax.spines['top'].set\_visible(False)  
 ax.plot(1, 0, ">k", transform=ax.get\_yaxis\_transform(), clip\_on=False, ms=8)  
 ax.plot(0, 1, "^k", transform=ax.get\_xaxis\_transform(), clip\_on=False, ms=8)  
 plt.xlim((xnmax[0]-1, xnmax[-1]+1))  
 plt.ylim((0, 1))  
 plt.grid() *# относительные частоты* plt.hlines(fstar, [xnmax[0]-1] + xnmax[:-1], xnmax[:-1] + [xnmax[-1]+1], linewidth=3, clip\_on=False)  
 tmp = fstar.pop(0)  
 plt.scatter(xnmax[:-1], fstar, marker='<', s=100)  
 fstar.insert(0, tmp)  
 plt.suptitle('Эмпирическая функция распределения')  
 plt.xlabel('xᵢ')  
 plt.ylabel('F\*(xᵢ)')  
 plt.xticks(xnmax)  
 plt.yticks([0] + fstar + [1])  
 *# Группированный* print("F\*(x) Групированного ряда")  
 print(f" ╭─\n │0 при x < {xs[0]}")  
 for i in range(1, len(ws)):  
 if i == len(ws)//2:  
 print(f"F\*(x) = ━┥{fstar[i]:<4} при {xs[i-1]:>2} ≤ x < {xs[i]:<2}")  
 else:  
 print(f" │{fstar[i]:<4} при {xs[i-1]:>2} ≤ x < {xs[i]:<2}")  
 print(f" │1 при x > {xs[-1]}\n ╰─")  
 *# график* ax = plt.figure().add\_subplot(1, 1, 1)  
 ax.spines['left'].set\_position(('data', max(xs[0]-1, min(0.0, xs[-1]+1))))  
 ax.spines['bottom'].set\_position(('data', 0.0))  
 ax.spines['right'].set\_visible(False)  
 ax.spines['top'].set\_visible(False)  
 ax.plot(1, 0, ">k", transform=ax.get\_yaxis\_transform(), clip\_on=False, ms=8)  
 ax.plot(0, 1, "^k", transform=ax.get\_xaxis\_transform(), clip\_on=False, ms=8)  
 plt.xlim((xs[0]-1, xs[-1]+1))  
 plt.ylim((0, 1))  
 plt.grid() *# относительные частоты* plt.hlines(fstar, [xs[0]-1] + xs, xs + [xs[-1]+1], linewidth=3, clip\_on=False)  
 tmp = fstar.pop(0)  
 plt.scatter(xs, fstar, marker='<', s=100)  
 fstar.insert(0, tmp)  
 plt.suptitle('Эмпирическая функция распределения')  
 plt.xlabel('xᵢ')  
 plt.ylabel('F\*(xᵢ)')  
 plt.xticks(xs)  
 plt.yticks([0] + fstar + [1])  
 plt.show()  
 input("Нажмите enter чтобы продолжить\n")  
  
*# Характеристики выборки*if("4" in choice):  
 print("\n4.")  
 print("Числовые характеристики выборки:")  
 xmean = sum([x\*y for x,y in zip(xs, ns)])/mainn  
 dmean = sum([((xmean-x)\*\*2)\*y for x,y in zip(xs, ns)])/mainn  
 sigmamean = dmean\*\*0.5  
 s = ((mainn\*dmean)/(mainn-1))\*\*0.5  
 print(f"- ̄xᵦ = {xmean}\n- Dᵦ = {dmean}\n- σᵦ = {sigmamean}\n- S = {s}")  
 *# формулы* img = Image.open("numchars.png")  
 plt.figure()  
 plt.axis("off")  
 plt.imshow(img)  
 plt.show()  
 input("Нажмите enter чтобы продолжить\n")

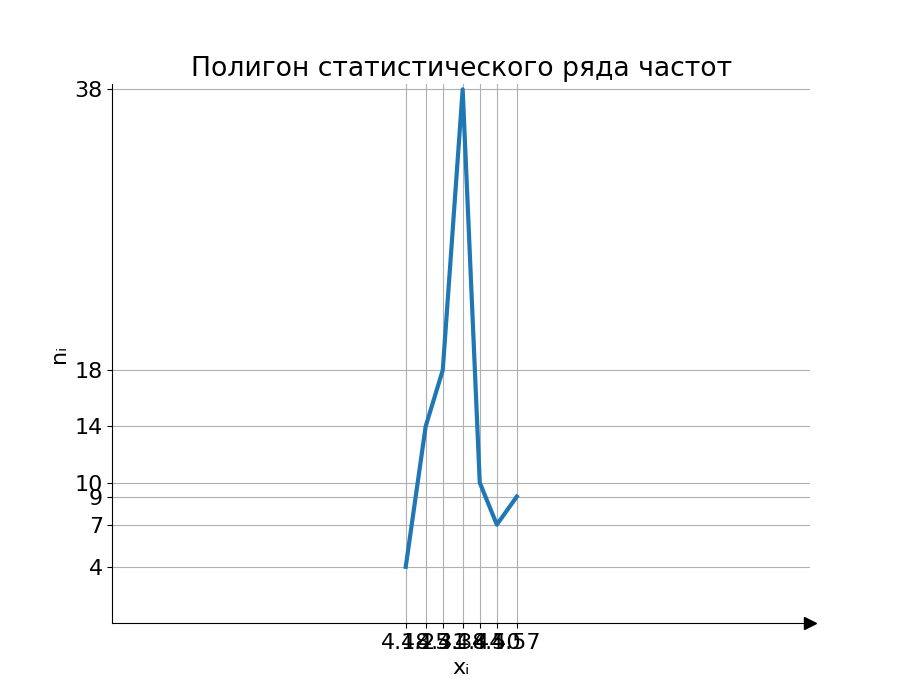
Исходная матрица:

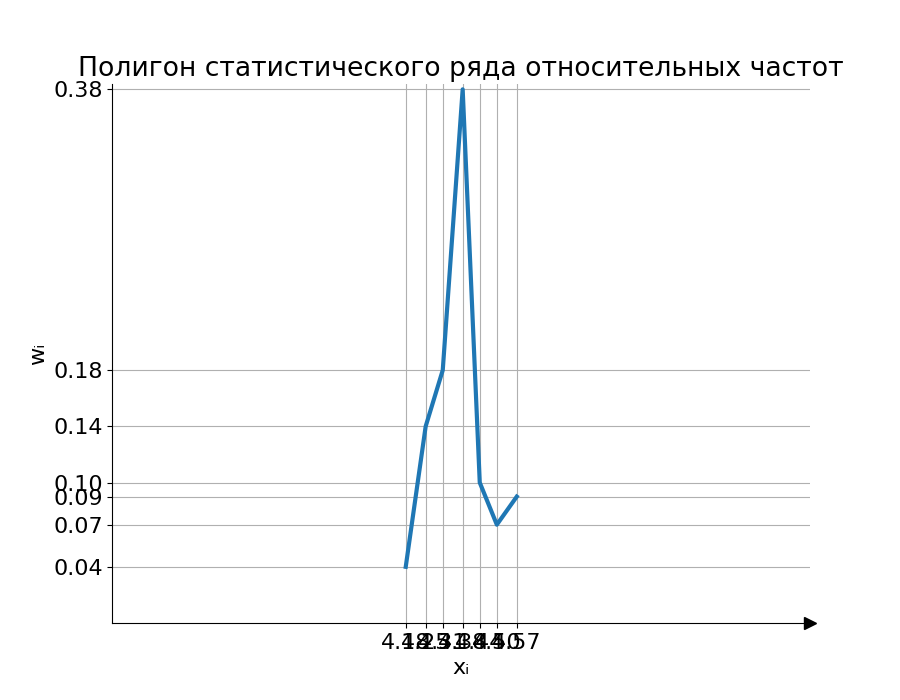


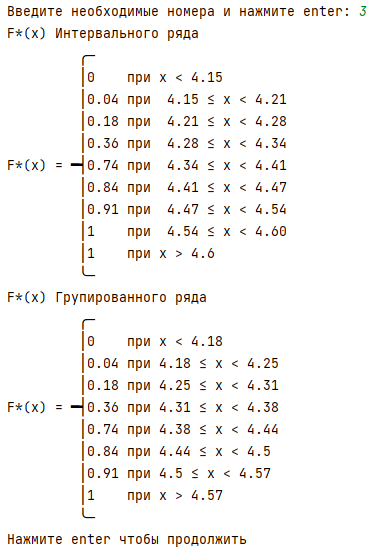
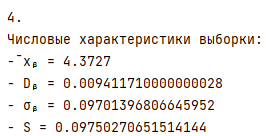
Решения:

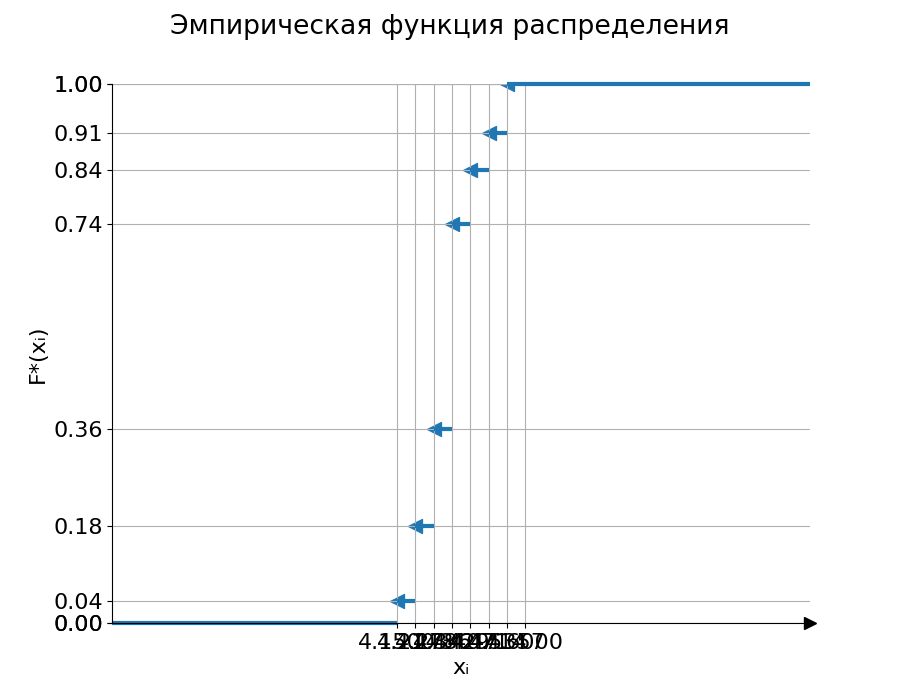








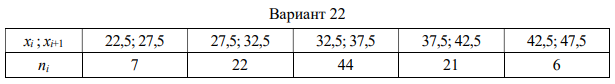




Задание №3

from matplotlib import pyplot as plt  
from PIL import Image  
  
intervalAmount = int(input("Введите количество интервалов: "))  
xnmax = set()  
ns = []  
for i in range(intervalAmount):  
 [xnmax.add(float(x)) for x in input(f"Введите {i+1}-ый интервал в формате xᵢ;xᵢ₊₁: ").split(';')]  
 ns.append(int(input(f"Введите {i+1}-ую частоту: ")))  
   
xnmax = sorted(list(xnmax))  
  
print()  
print("\nВыберите интересующие вас пункты:\n\  
 1. Получение и вывод на экран интервального ряда частот и относительных частот, построение соответствующих гистограмм;\n\  
 2. Построение группированного ряда распределения частот и относительных частот и построение соответствующих полигонов;\n\  
 3. Нахождение и вывод на экран эмпирической функции распределения F\*(х) для интервальных и группированных рядов, построение их графиков;\n\  
 4. Вывод на экран формул для вычисления числовых характеристик выборки: ̄xᵦ, Dᵦ, σᵦ, S, их вычисление и вывод на экран полученных результатов;\n")  
choice = input("Введите необходимые номера и нажмите enter: ")  
  
mainn = sum(ns)  
ws = [n / mainn for n in ns]  
  
if("1" in choice):  
 *# Частоты* print("\nИнтервальный ряд частот:")  
 print((intervalAmount \* 14 + 9) \* '▁')  
 print("┃xᵢ;xᵢ₊₁┃", end='')  
 for i in range(intervalAmount-1):  
 print(f"{xnmax[i]:>6.2f};{xnmax[i+1]:<6.2f}", end='┃')  
 print(f"{xnmax[-2]:>6.2f};{xnmax[-1]:<6.2f}", end='┃')  
 print("\n┃ nᵢ ┃", end='')  
 [print(f"{n:^13}", end='┃') for n in ns]  
 print('\n' + (intervalAmount \* 14 + 9) \* '▔')  
 *# Относительные частоты* print("Интервальный ряд относительных частот:")  
 print((intervalAmount \* 14 + 9) \* '▁')  
 print("┃xᵢ;xᵢ₊₁┃", end='')  
 for i in range(intervalAmount-1):  
 print(f"{xnmax[i]:>6.2f};{xnmax[i+1]:<6.2f}", end='┃')  
 print(f"{xnmax[-2]:>6.2f};{xnmax[-1]:<6.2f}", end='┃')  
 print("\n┃ wᵢ ┃", end='')  
 [print(f"{w:^13}", end='┃') for w in ws]  
 print('\n' + (intervalAmount \* 14 + 9) \* '▔')  
*# Гистограммы  
 # Частоты* histns = [n/(xnmax[1]-xnmax[0]) for n in ns]  
 ax = plt.figure().add\_subplot(1, 1, 1)  
 ax.spines['left'].set\_position(('data', max(xnmax[0]-1, min(0.0, xnmax[-1]+1))))  
 ax.spines['bottom'].set\_position(('data', 0.0))  
 ax.spines['right'].set\_visible(False)  
 ax.spines['top'].set\_visible(False)  
 ax.plot(1, 0, ">k", transform=ax.get\_yaxis\_transform(), clip\_on=False, ms=8)  
 ax.plot(0, 1, "^k", transform=ax.get\_xaxis\_transform(), clip\_on=False, ms=8)  
 plt.xlim((xnmax[0]-1, xnmax[-1]+1))  
 plt.ylim((0, max(histns)+max(histns)/100))  
 plt.grid() *# частоты* plt.stairs(histns, xnmax, fill=True, facecolor="indigo", zorder=3)  
 plt.vlines(xnmax[1:-1], 0, histns[:-1], color="black", zorder=4)  
 plt.title('Гистограмма интервального ряда частот')  
 plt.xlabel('xᵢ')  
 plt.ylabel('nᵢ')  
 plt.xticks(xnmax)  
 plt.yticks(histns)  
 *# Относительные частоты* histws = [w/(xnmax[1]-xnmax[0]) for w in ws]  
 ax = plt.figure().add\_subplot(1, 1, 1)  
 ax.spines['left'].set\_position(('data', max(xnmax[0]-1, min(0.0, xnmax[-1]+1))))  
 ax.spines['bottom'].set\_position(('data', 0.0))  
 ax.spines['right'].set\_visible(False)  
 ax.spines['top'].set\_visible(False)  
 ax.plot(1, 0, ">k", transform=ax.get\_yaxis\_transform(), clip\_on=False, ms=8)  
 ax.plot(0, 1, "^k", transform=ax.get\_xaxis\_transform(), clip\_on=False, ms=8)  
 plt.xlim((xnmax[0]-1, xnmax[-1]+1))  
 plt.ylim((0, max(histws)+max(histws)/100))  
 plt.grid() *# частоты* plt.stairs(histws, xnmax, fill=True, facecolor="darkmagenta", zorder=3)  
 plt.vlines(xnmax[1:-1], 0, histws[:-1], color="black", zorder=4)  
 plt.title('Гистограмма интервального ряда относительных частот')  
 plt.xlabel('xᵢ')  
 plt.ylabel('wᵢ')  
 plt.xticks(xnmax)  
 plt.yticks(histws)  
 plt.show()  
 input("Нажмите enter чтобы продолжить\n")  
  
xs = [round((xnmax[i] + xnmax[i+1])/2, 2) for i in range(intervalAmount)]  
if("2" in choice):  
 print("\n2.")  
 print("Группированный ряд частот:")  
 print((len(xs) \* 6 + 5) \* '▁')  
 print("┃xᵢ ┃", end='')  
 [print(f"{x:^5.2f}", end='┃') for x in xs]  
 print("\n┃nᵢ ┃", end='')  
 [print(f"{n:^5}", end='┃') for n in ns]  
 print('\n' + (len(xs) \* 6 + 5) \* '▔')  
 *# Относительные частоты* print("Группированный ряд относительных частот:")  
 print((len(xs) \* 6 + 5) \* '▁')  
 print("┃xᵢ ┃", end='')  
 [print(f"{x:^5.2f}", end='┃') for x in xs]  
 print("\n┃wᵢ ┃", end='')  
 [print(f"{w:^5}", end='┃') for w in ws]  
 print('\n' + (len(xs) \* 6 + 5) \* '▔')  
*# Найти и вывести полигоны частот* ax = plt.figure().add\_subplot(1, 1, 1)  
 ax.spines['left'].set\_position(('data', max(xnmax[0]-1, min(0.0, xnmax[-1]+1))))  
 ax.spines['bottom'].set\_position(('data', 0.0))  
 ax.spines['right'].set\_visible(False)  
 ax.spines['top'].set\_visible(False)  
 ax.plot(1, 0, ">k", transform=ax.get\_yaxis\_transform(), clip\_on=False, ms=8)  
 ax.plot(0, 1, "^k", transform=ax.get\_xaxis\_transform(), clip\_on=False, ms=8)  
 plt.xlim((xnmax[0]-1, xnmax[-1]+1))  
 plt.ylim((0, max(ns)+max(ns)/100))  
 plt.grid() *# частоты* plt.plot(xs, ns, linewidth=3)  
 plt.title('Полигон статистического ряда частот')  
 plt.xlabel('xᵢ')  
 plt.ylabel('nᵢ')  
 plt.xticks(xs)  
 plt.yticks(ns)  
 *# Относительных частот* ax = plt.figure().add\_subplot(1, 1, 1)  
 ax.spines['left'].set\_position(('data', max(xnmax[0]-1, min(0.0, xnmax[-1]+1))))  
 ax.spines['bottom'].set\_position(('data', 0.0))  
 ax.spines['right'].set\_visible(False)  
 ax.spines['top'].set\_visible(False)  
 ax.plot(1, 0, ">k", transform=ax.get\_yaxis\_transform(), clip\_on=False, ms=8)  
 ax.plot(0, 1, "^k", transform=ax.get\_xaxis\_transform(), clip\_on=False, ms=8)  
 plt.xlim((xnmax[0]-1, xnmax[-1]+1))  
 plt.ylim((0, max(ws)+max(ws)/100))  
 plt.grid()  
 plt.plot(xs, ws, linewidth=3)  
 plt.title('Полигон статистического ряда относительных частот')  
 plt.xlabel('xᵢ')  
 plt.ylabel('wᵢ')  
 plt.xticks(xs)  
 plt.yticks(ws)  
 plt.show()  
 input("Нажмите enter чтобы продолжить\n")  
  
*# Функции распределения*if("3" in choice):  
 fstar = [0]  
 for i in range(1, len(ws)):  
 fstar.append(round(ws[i-1]+fstar[i-1], 5))  
 fstar.append(1)  
 *# Интервальный* print("F\*(x) Интервального ряда")  
 print(f" ╭─\n │0 при x < {xnmax[0]}")  
 for i in range(1, len(fstar)):  
 if i == len(fstar)//2:  
 print(f"F\*(x) = ━┥{fstar[i]:<4} при {xnmax[i-1]:>5.2f} ≤ x < {xnmax[i]:<5.2f}")  
 else:  
 print(f" │{fstar[i]:<4} при {xnmax[i-1]:>5.2f} ≤ x < {xnmax[i]:<5.2f}")  
 print(f" │1 при x > {xnmax[-1]}\n ╰─")  
 *# график* ax = plt.figure().add\_subplot(1, 1, 1)  
 ax.spines['left'].set\_position(('data', max(xnmax[0]-1, min(0.0, xnmax[-1]+1))))  
 ax.spines['bottom'].set\_position(('data', 0.0))  
 ax.spines['right'].set\_visible(False)  
 ax.spines['top'].set\_visible(False)  
 ax.plot(1, 0, ">k", transform=ax.get\_yaxis\_transform(), clip\_on=False, ms=8)  
 ax.plot(0, 1, "^k", transform=ax.get\_xaxis\_transform(), clip\_on=False, ms=8)  
 plt.xlim((xnmax[0]-1, xnmax[-1]+1))  
 plt.ylim((0, 1))  
 plt.grid() *# относительные частоты* plt.hlines(fstar, [xnmax[0]-1] + xnmax[:-1], xnmax[:-1] + [xnmax[-1]+1], linewidth=3, clip\_on=False)  
 tmp = fstar.pop(0)  
 plt.scatter(xnmax[:-1], fstar, marker='<', s=100)  
 fstar.insert(0, tmp)  
 plt.suptitle('Эмпирическая функция распределения')  
 plt.xlabel('xᵢ')  
 plt.ylabel('F\*(xᵢ)')  
 plt.xticks(xnmax)  
 plt.yticks([0] + fstar + [1])  
 *# Группированный* print("F\*(x) Групированного ряда")  
 print(f" ╭─\n │0 при x < {xs[0]}")  
 for i in range(1, len(ws)):  
 if i == len(ws)//2:  
 print(f"F\*(x) = ━┥{fstar[i]:<4} при {xs[i-1]:>2} ≤ x < {xs[i]:<2}")  
 else:  
 print(f" │{fstar[i]:<4} при {xs[i-1]:>2} ≤ x < {xs[i]:<2}")  
 print(f" │1 при x > {xs[-1]}\n ╰─")  
 *# график* ax = plt.figure().add\_subplot(1, 1, 1)  
 ax.spines['left'].set\_position(('data', max(xs[0]-1, min(0.0, xs[-1]+1))))  
 ax.spines['bottom'].set\_position(('data', 0.0))  
 ax.spines['right'].set\_visible(False)  
 ax.spines['top'].set\_visible(False)  
 ax.plot(1, 0, ">k", transform=ax.get\_yaxis\_transform(), clip\_on=False, ms=8)  
 ax.plot(0, 1, "^k", transform=ax.get\_xaxis\_transform(), clip\_on=False, ms=8)  
 plt.xlim((xs[0]-1, xs[-1]+1))  
 plt.ylim((0, 1))  
 plt.grid() *# относительные частоты* plt.hlines(fstar, [xs[0]-1] + xs, xs + [xs[-1]+1], linewidth=3, clip\_on=False)  
 tmp = fstar.pop(0)  
 plt.scatter(xs, fstar, marker='<', s=100)  
 fstar.insert(0, tmp)  
 plt.suptitle('Эмпирическая функция распределения')  
 plt.xlabel('xᵢ')  
 plt.ylabel('F\*(xᵢ)')  
 plt.xticks(xs)  
 plt.yticks([0] + fstar + [1])  
 plt.show()  
 input("Нажмите enter чтобы продолжить\n")  
  
*# Характеристики выборки*if("4" in choice):  
 print("\n4.")  
 print("Числовые характеристики выборки:")  
 xmean = sum([x\*y for x,y in zip(xs, ns)])/mainn  
 dmean = sum([((xmean-x)\*\*2)\*y for x,y in zip(xs, ns)])/mainn  
 sigmamean = dmean\*\*0.5  
 s = ((mainn\*dmean)/(mainn-1))\*\*0.5  
 print(f"- ̄xᵦ = {xmean}\n- Dᵦ = {dmean}\n- σᵦ = {sigmamean}\n- S = {s}")  
 *# формулы* img = Image.open("numchars.png")  
 plt.figure()  
 plt.axis("off")  
 plt.imshow(img)  
 plt.show()  
 input("Нажмите enter чтобы продолжить\n")

Исходное условие:



Решения:

