In [2]:

```
from matplotlib import pyplot as plt
import numpy as np
import pandas as pd
from random import randint, seed
import math
from scipy. stats import ttest_ind
```

1 Задача.

AI

Так как каждый таксист приезжает равновероятно, то вероятность того, что конкретный таксист приедет, равна 1/n. Вероятность того, что при 10 заказах один и тот же таксист приедет ровно один раз, равна (9/n) * ((n-1)/n)^9. Общая вероятность получить данную последовательность приездов таксистов равна (9/n) * ((n-1)/n)^9.

Таким образом, функция правдоподобия будет иметь вид L(n) = (9/n) * ((n-1)/n)^9. Ее можно записать в виде L(n) = 9 * (1 - 1/n)^9 / n. График этой функции будет иметь пик в точке n=10 (при 10 заказах конкретный таксист должен приехать ровно один раз), после чего будет монотонно убывать.

Оценку числа n методом максимального правдоподобия можно найти, продифференцировав функцию правдоподобия и приравняв ее к нулю:

dL/dn = -9 * (1 - 1/n)^8 / n^2 + 9 * (1 - 1/n)^9 / n^2 = 0

Решив это уравнение, получаем n=10. То есть оценка числа таксистов в Самарканде по данным 10 заказов равна 10.

In [56]:

```
mesto_vstrechi = 10
n = np.linspace(3, 15, dtype = int)
L = (mesto_vstrechi - 1) * (math.factorial(n) / ( math.factorial(n - (mesto_vstrechi - 1)
plt.plot(n, L)
plt.show()

n = np.linspace(1, 1000, 10000)
Z = 9 * (1 - 1/n) ** 9 / n
plt.plot(n, Z)
plt.show()
```

```
---------------------------------------------------------------------------
-
TypeError                                 Traceback (most recent call las
t)
Cell In[56], line 4
      1 mesto_vstrechi = 10
      2 n = np.linspace(3, 15, dtype = int)
----> 4 plt.plot(n, L = (mesto_vstrechi - 1) * (math.factorial(n) / ( mat
h.factorial(n - (mesto_vstrechi - 1)) * n ** (mesto_vstrechi - 1))))
      5 plt.show()
      7 n = np.linspace(1, 1000, 10000)

TypeError: only integer scalar arrays can be converted to a scalar index
```

б)

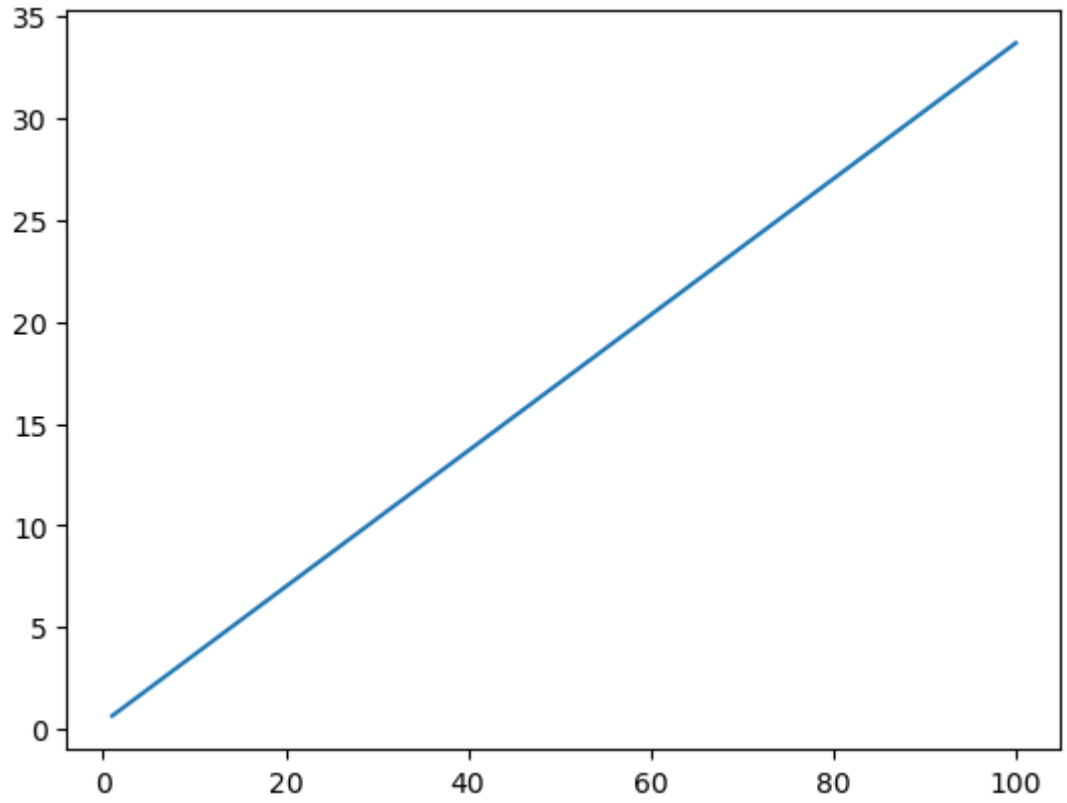
AI

Обозначим через X_i номер заказа, на котором приезжает i-ый таксист (первый, второй, третий и т.д.). Задача состоит в нахождении математического ожидания случайной величины Y, равной номеру заказа, на котором происходит первый повторный приезд. Вероятность того, что на i-м заказе приедет таксист, который уже был ранее, равна (i-1)/n. Тогда вероятность того, что первый повторный приезд произойдет на i-м заказе, равна P(Y=i) = (i-1)/n * (n-i+1)/n. Тогда математическое ожидание Y можно найти, используя формулу:

E(Y) = sum(i=1 to n) i * P(Y=i)

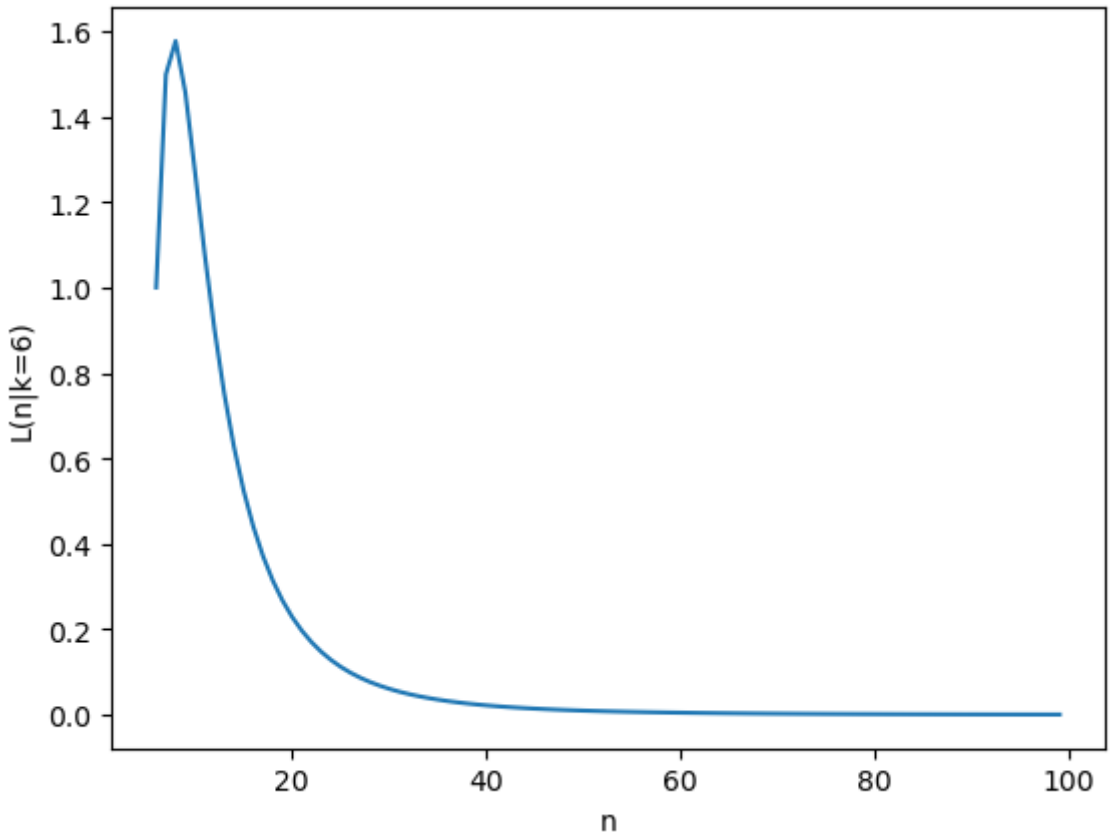
In [4]:

```
kolvo_n = np.linspace(1, 100, 10000)
ME_i = (kolvo_n+1)/3
plt.plot(kolvo_n, ME_i)
plt.show()
```



In [5]:

```
k = 6
n = np.arange(6, 100)
L = np.zeros_like(n, dtype=np.float64)
for i, value in enumerate(n):
    L[i] = np.math.comb(value, k) * (k / value) ** 10
plt.plot(n, L)
plt.xlabel('n')
plt.ylabel('L(n|k=6)')
plt.show()
```



In [6]:

```
df = pd.DataFrame()
df["nomer_vstrechi"] = ''
df
```

Out[6]:

	nomer_vstrechi

In [7]:

```
i = 0
seed(32)
while (i < 10_000):
    vstrecha = 0
    Nomera_proshedshie = []
    while (vstrecha == 0):
        nomer = randint(1, 100)
        if (nomer in Nomera_proshedshie):
            vstrecha = 1
        else:
            Nomera_proshedshie.append(nomer)
    df.loc[i] = nomer
    i += 1
df
```

Out[7]:

	nomer_vstrechi
0	26
1	5
2	74
3	11
4	90
...	...
9995	66
9996	30
9997	67
9998	6
9999	79

10000 rows × 1 columns

In [8]:

```
df["оценка n м. Моментов"] = df["nomer_vstrechi"]
```

In [ ]:

```

```

Задача 4.

In [ ]:

```

```

In [9]:

```
Grades_stat = pd.read_excel('Grades_stat.xlsx')
Grades_stat
```

Out[9]:

	Фамилия	Балл
0	Репенкова	16
1	Ролдугина	0
2	Сафина	19
3	Сидоров	26
4	Солоухин	21
...	...	...
327	Сенников	19
328	Ся	0
329	Сятова	0
330	Темиркулов	0
331	Эшмеев	16

332 rows × 2 columns

In [17]:

```
glasnie = ['А', 'Я', 'У', 'Ю', 'О', 'Е', 'Ё', 'Э', 'И', 'Ы']
Grades_stat['Пр_гласной'] = ''
i = 0
while (i < len(Grades_stat)):
    if (Grades_stat['Фамилия'][i][0] in glasnie):
        Grades_stat['Пр_гласной'][i] = "Гласная"
    else:
        Grades_stat['Пр_гласной'][i] = "Согласная"
    i += 1
Grades_stat
```

```
C:\Users\Иван\AppData\Local\Temp\ipykernel_2524\1921499877.py:8: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  Grades_stat['Пр_гласной'][i] = "Согласная"
C:\Users\Иван\AppData\Local\Temp\ipykernel_2524\1921499877.py:6: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://
pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-
view-versus-a-copy)
  Grades_stat['Пр_гласной'][i] = "Гласная"
```

Out[17]:

	Фамилия	Балл	Пр_гласной
0	Репенкова	16	Согласная
1	Ролдугина	0	Согласная
2	Сафина	19	Согласная
3	Сидоров	26	Согласная
4	Солоухин	21	Согласная
...	...	...	...
327	Сенников	19	Согласная
328	Ся	0	Согласная
329	Сятова	0	Согласная
330	Темиркулов	0	Согласная
331	Эшмеев	16	Гласная

332 rows × 3 columns

In [20]:

```
Grad_fin = Grades_stat[['Балл','Пр_гласной']]
```

In [21]:

```
agg_func_math = {'count','sum', 'mean', 'median', 'min', 'max', 'std', 'var', 'prod'}
Grad_fin.groupby(['Пр_гласной']).agg(agg_func_math).round(2)
```

Out[21]:

	Балл								
	var	prod	max	std	min	mean	median	sum	count
Пр_гласной									
Гласная	67.75	0	29	8.23	0	15.29	16.0	749	49
Согласная	62.00	0	30	7.87	0	16.36	18.0	4631	283

а) тест Уэлча

In [23]:

```
group1 = Grad_fin[Grad_fin['Пр_гласной'] == "Гласная"]
group2 = Grad_fin[Grad_fin['Пр_гласной'] == "Согласная"]
ttest_ind(group1['Балл'], group2['Балл'], equal_var= False )
```

Out[23]:

```
Ttest_indResult(statistic=-0.8519661870595602, pvalue=0.3974027153843839)
```

pvalue = 0.397 > 0.05, нет оснований отвергать гипотезу H0

б) Наивный бутстрэп

In [ ]:

```

```

In [ ]:

```

```

5 задача

In [24]:

```
Median_grade = Grades_stat['Балл'].median()
Median_grade
```

Out[24]:

```
17.5
```

In [25]:

```
Grad_fin["Пр_медианы"] = np.where(Grad_fin["Балл"] > Median_grade, 'Больше_медианы', 'Мен
Grad_fin
```

Out[25]:

	Балл	Пр_гласной	Пр_медианы
0	16	Согласная	Меньше_медианы
1	0	Согласная	Меньше_медианы
2	19	Согласная	Больше_медианы
3	26	Согласная	Больше_медианы
4	21	Согласная	Больше_медианы
...	...	...	...
327	19	Согласная	Больше_медианы
328	0	Согласная	Меньше_медианы
329	0	Согласная	Меньше_медианы
330	0	Согласная	Меньше_медианы
331	16	Гласная	Меньше_медианы

332 rows × 3 columns

In [29]:

```
Tabl_sopr = pd.pivot_table(Grad_fin, values='Пр_медианы', index='Пр_гласной', columns='Пр
Tabl_sopr
```

Out[29]:

Пр_медианы	Больше_медианы	Меньше_медианы
Пр_гласной		
Гласная	21	28
Согласная	145	138

а) Постройте 95% асимптотический интервал для отношения шансов хорошо написать экзамен

In [34]:

```
BM_Gl = Tabl_sopr['Больше_медианы']['Гласная']
MM_GL = Tabl_sopr["Меньше_медианы"]['Гласная']
BM_S = Tabl_sopr['Больше_медианы']['Согласная']
MM_S = Tabl_sopr['Меньше_медианы']['Согласная']


OR = (BM_Gl * MM_S) / (MM_GL * BM_S)
OR
```

Out[34]:

```
0.7137931034482758
```

In [49]:

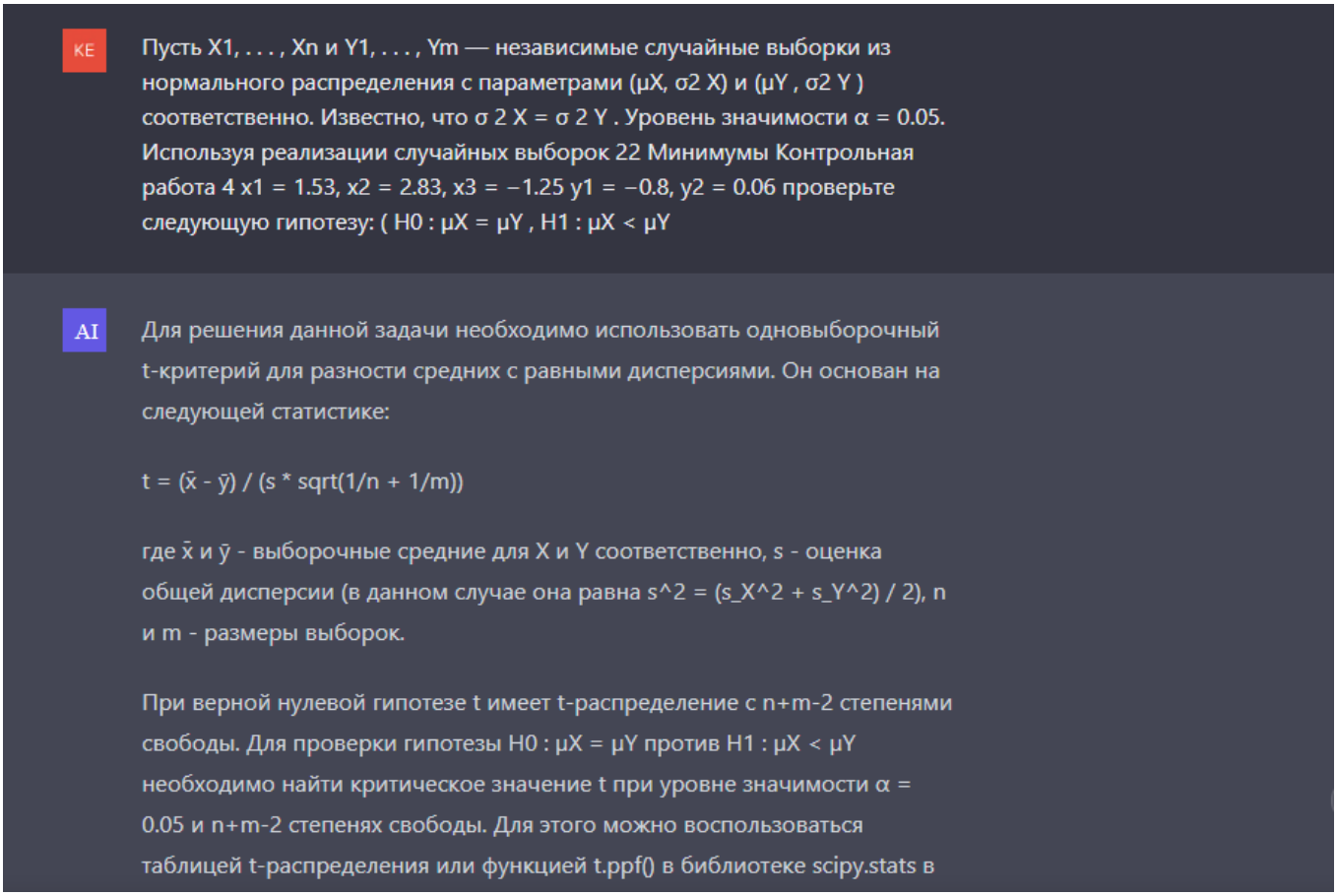
```
promezh_OR_CI = (1/BM_Gl) + (1/MM_S) + (1/MM_GL) + (1/BM_S)
OR_CI_left = OR - 1.96 * math.sqrt(promezh_OR_CI)
OR_CI_right = OR + 1.96 * math.sqrt(promezh_OR_CI)
OR_CI_left, OR_CI_right
```

Out[49]:

```
(0.10185780654491716, 1.3257284003516345)
```

7 задание

задача из минимума 4 - 15



8 задание

В подотовке к конторольным мне сильно помогают записи лекций и сименаров. Момимо этого помогает лучше понять и применить знания статистики на практике базовый курс по python. Лекции Филипа и Максима отлично дополняют курс теории вероятности. Ссылка на вики python:https://github.com/hse-econ-data-science/andan_2023/tree/main (https://github.com/hse-econ-data-science/andan_2023/tree/main)

In [ ]: