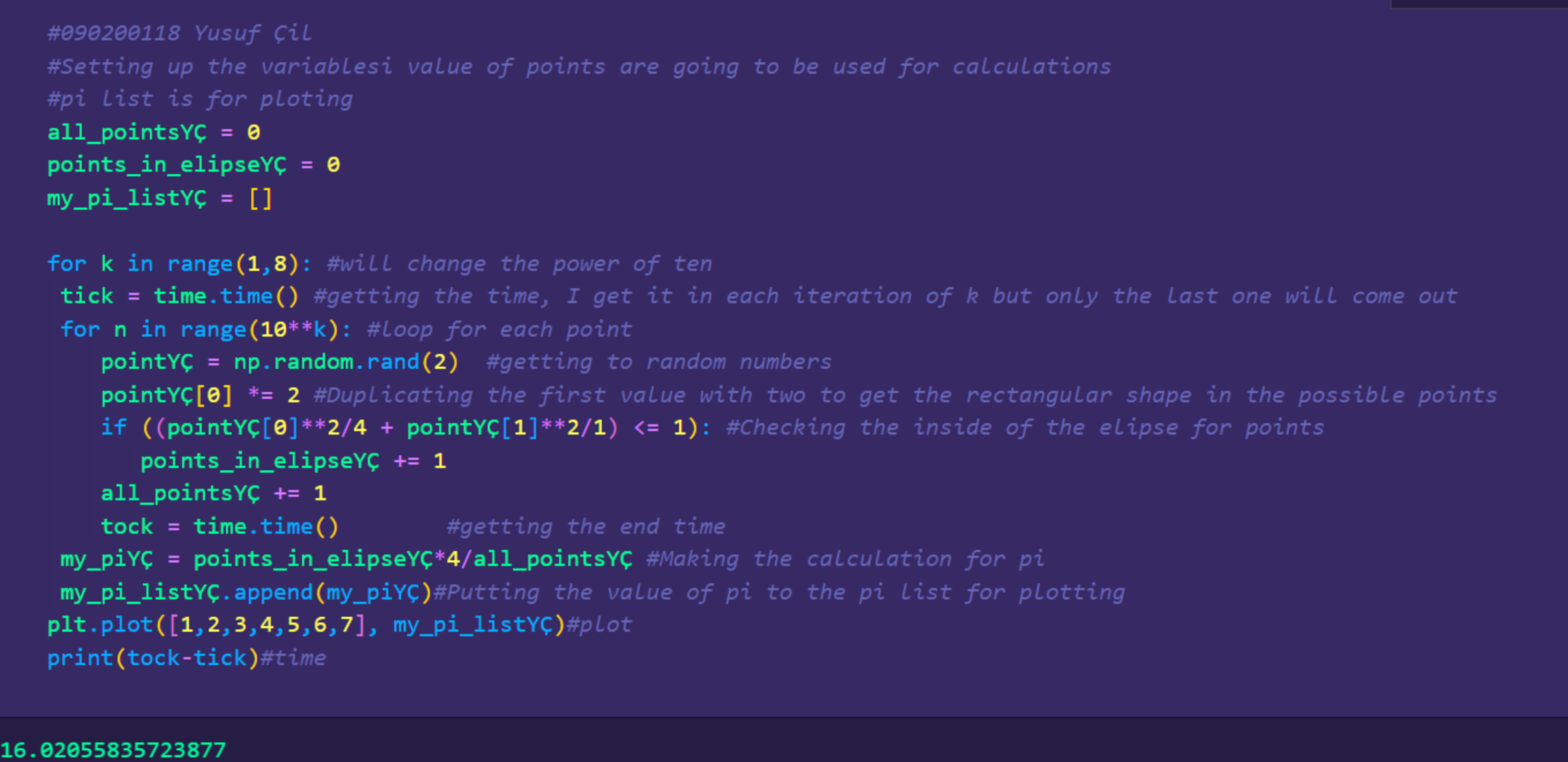
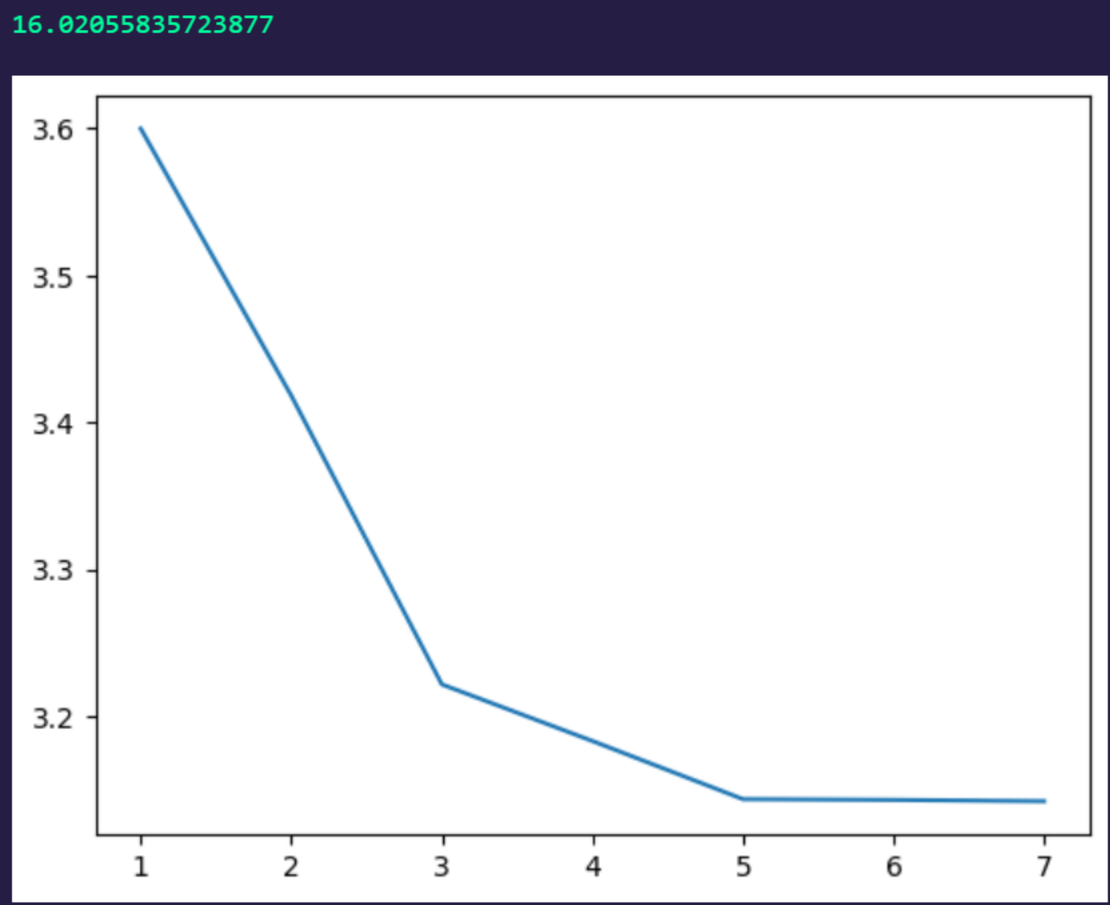
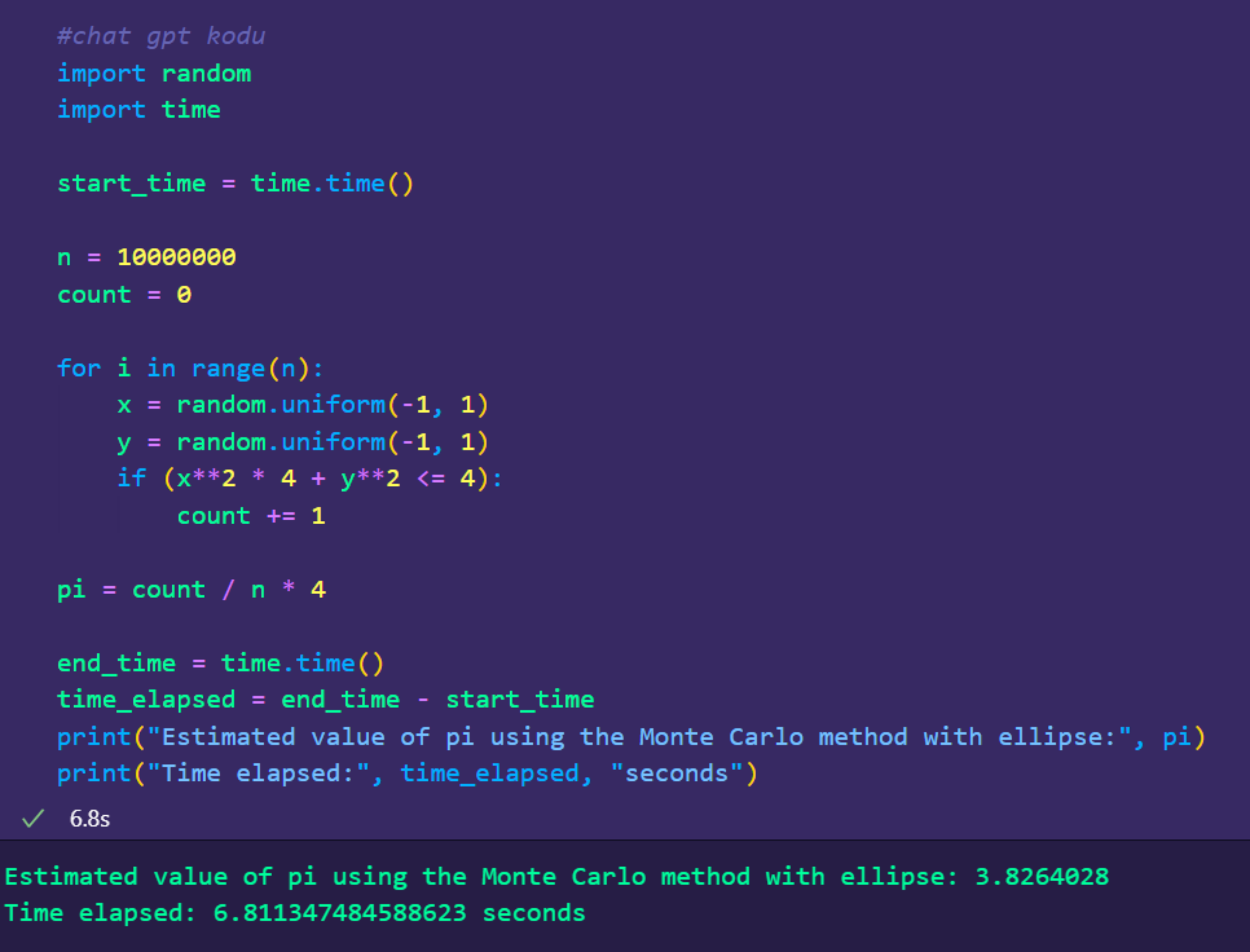


```
#090200118 Yusuf Çil
#Setting up the variablesi value of points are going to be used for calculations
#pi list is for ploting
all_pointsYÇ = 0
points_in_elipseYÇ = 0
my_pi_listYÇ = []

for k in range(1,8): #will change the power of ten
 tick = time.time() #getting the time, I get it in each iteration of k but only the last one will come out
 for n in range(10**k): #loop for each point
    pointYÇ = np.random.rand(2)  #getting to random numbers
    pointYÇ[0] *= 2 #Duplicating the first value with two to get the rectangular shape in the possible points
    if ((pointYÇ[0]**2/4 + pointYÇ[1]**2/1) <= 1): #Checking the inside of the elipse for points
       points_in_elipseYÇ += 1
    all_pointsYÇ += 1
    tock = time.time()        #getting the end time
 my_piYÇ = points_in_elipseYÇ*4/all_pointsYÇ #Making the calculation for pi
 my_pi_listYÇ.append(my_piYÇ)#Putting the value of pi to the pi list for plotting
plt.plot([1,2,3,4,5,6,7], my_pi_listYÇ)#plot
print(tock-tick)#time
```

```
16.02055835723877
```



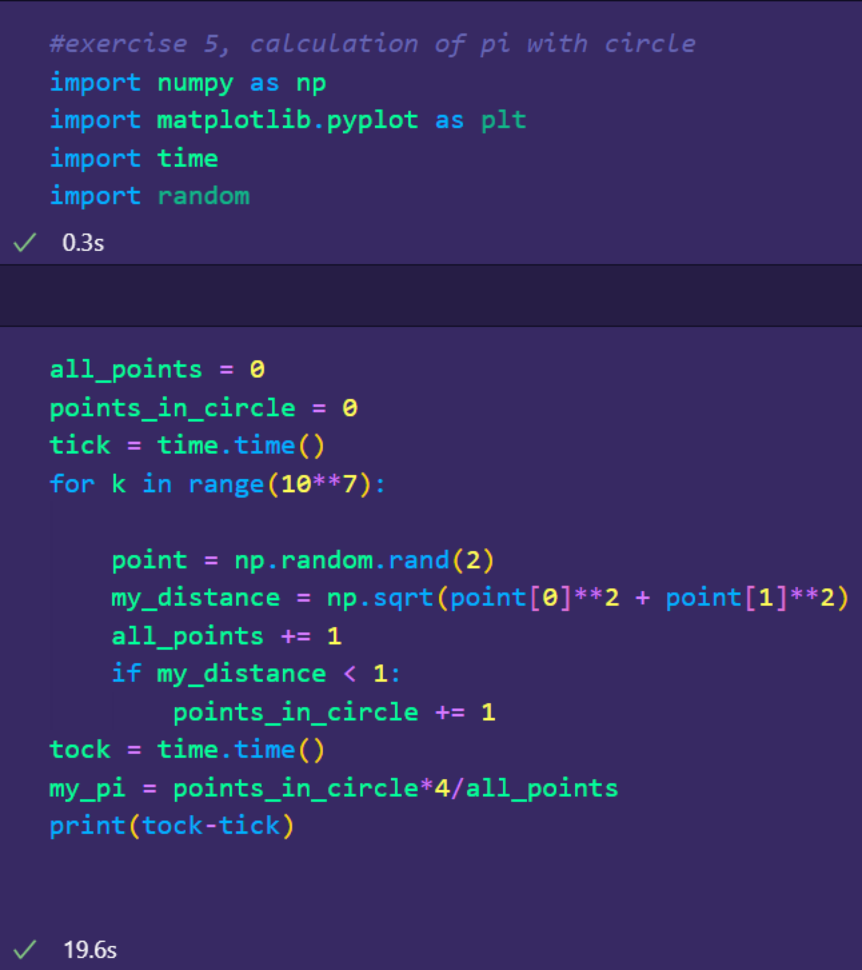
```
16.02055835723877
```



```python
#chat gpt kodu
import random
import time

start_time = time.time()

n = 10000000
count = 0

for i in range(n):
    x = random.uniform(-1, 1)
    y = random.uniform(-1, 1)
    if (x**2 * 4 + y**2 <= 4):
        count += 1

pi = count / n * 4

end_time = time.time()
time_elapsed = end_time - start_time
print("Estimated value of pi using the Monte Carlo method with ellipse:", pi)
print("Time elapsed:", time_elapsed, "seconds")
```

✓ 6.8s

```
Estimated value of pi using the Monte Carlo method with ellipse: 3.8264028
Time elapsed: 6.811347484588623 seconds
```

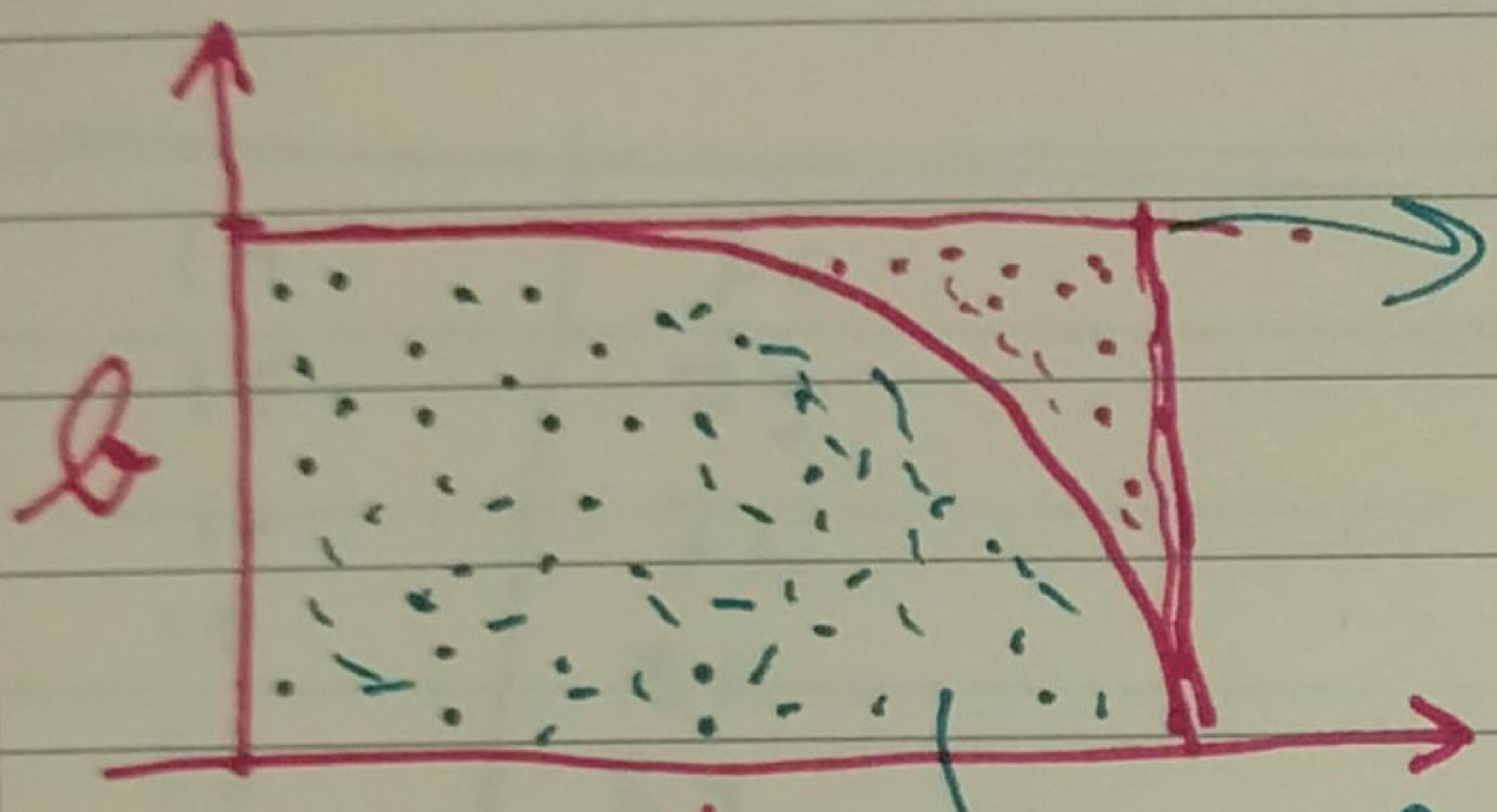


```
#exercise 5, calculation of pi with circle
import numpy as np
import matplotlib.pyplot as plt
import time
import random
```

✓ 0.3s

```
all_points = 0
points_in_circle = 0
tick = time.time()
for k in range(10**7):

    point = np.random.rand(2)
    my_distance = np.sqrt(point[0]**2 + point[1]**2)
    all_points += 1
    if my_distance < 1:
        points_in_circle += 1
tock = time.time()
my_pi = points_in_circle*4/all_points
print(tock-tick)
```

✓ 19.6s



$a\cdot b = A_{rect}$

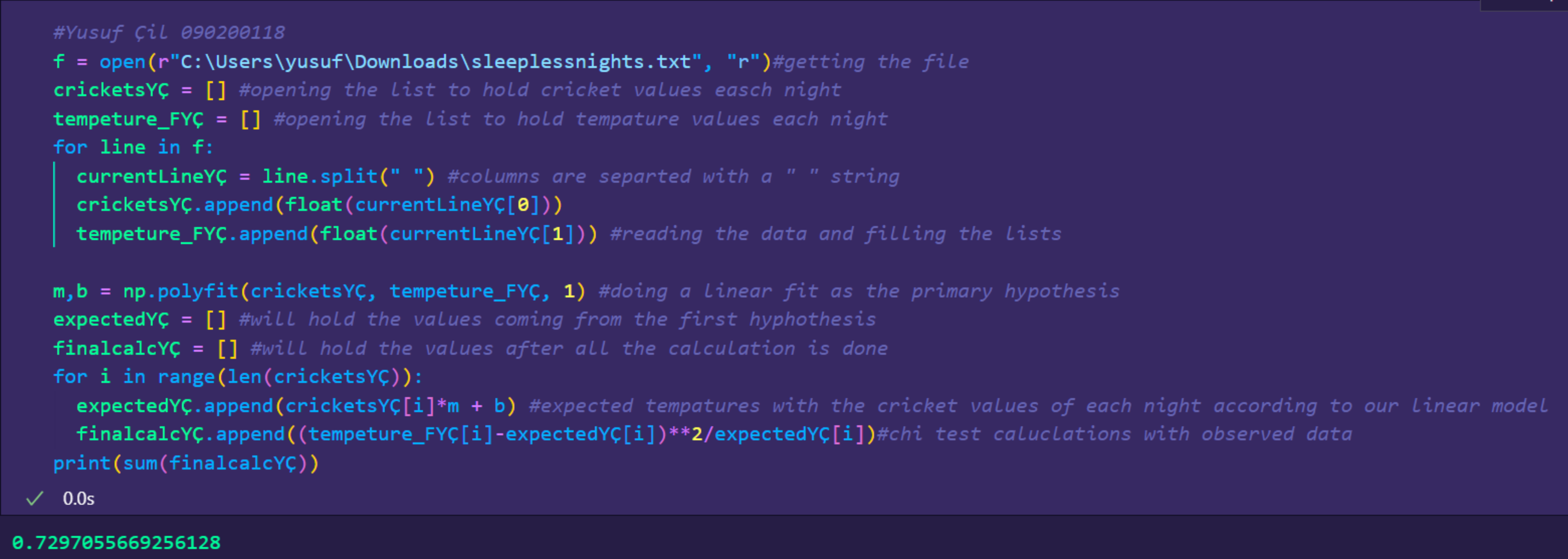
$\frac{\pi a\cdot b}{4} = A_{elipse}$

$4\cdot\left(\frac{A_{elipse}}{A_{rect}}\right) = \frac{\pi \cancel{a}\cdot \cancel{b}}{\cancel{4}\cancel{a}\cancel{b}}\cdot 4 = \underline{\underline{\pi}}$

My calculation time for 10^7 points is = 16.02 seconds ~~and~~ pi= 3.1416
Pi calculation with a circle in exercise 5 : 19.6 seconds , ~~3.1412~~ 3.1412 pi
Chat GPT code took 6.811 seconds but calculated pi as 3.82

Chat GPT did not use numpy which probably added some Performance benefit. It also used a different version of the ellipse equation.

I think using the circle was easier since it is a more intuitive shape mathematicly. Chat GPT is an excellent tool.



```python
#Yusuf Çil 090200118
f = open(r"C:\Users\yusuf\Downloads\sleeplessnights.txt", "r")#getting the file
cricketsYÇ = [] #opening the list to hold cricket values easch night
tempeture_FYÇ = [] #opening the list to hold tempature values each night
for line in f:
    currentLineYÇ = line.split(" ") #columns are separted with a " " string
    cricketsYÇ.append(float(currentLineYÇ[0]))
    tempeture_FYÇ.append(float(currentLineYÇ[1])) #reading the data and filling the lists

m,b = np.polyfit(cricketsYÇ, tempeture_FYÇ, 1) #doing a linear fit as the primary hypothesis
expectedYÇ = [] #will hold the values coming from the first hyphothesis
finalcalcYÇ = [] #will hold the values after all the calculation is done
for i in range(len(cricketsYÇ)):
    expectedYÇ.append(cricketsYÇ[i]*m + b) #expected tempatures with the cricket values of each night according to our linear model
    finalcalcYÇ.append((tempeture_FYÇ[i]-expectedYÇ[i])**2/expectedYÇ[i])#chi test caluclations with observed data
print(sum(finalcalcYÇ))
```

✓ 0.0s

```
0.7297055669256128
```

Chi-square is a statistical test to determine if there is a corelation between given data types (crickets and tempeture). It takes two hypothesis and uses the expected values from one of them in calculation with the observed ones.

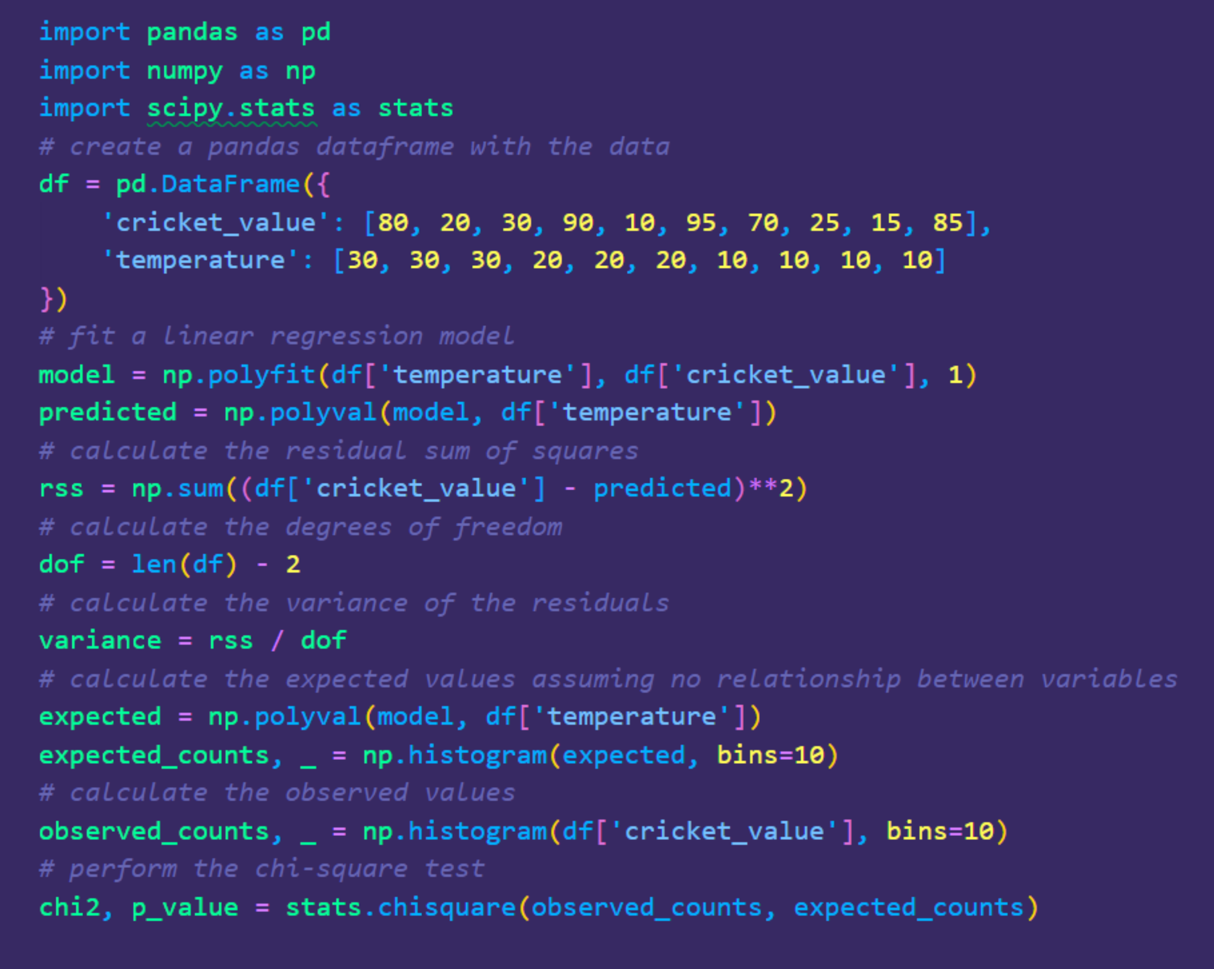
My programme gave the x^2 value of 0.729.

degree of freedom → $df = (r-1)(c-1)$ $r=2$ $c=7$

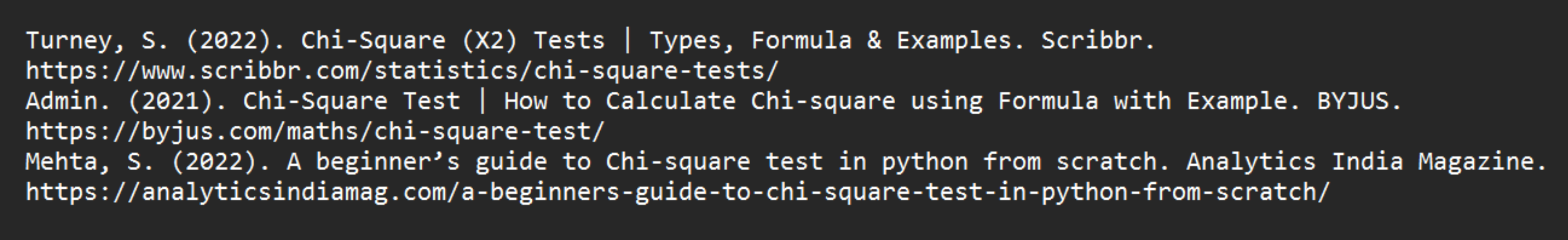
$df = 6$

with .05 significance level Critical value for chi-square is = 12.592 which is bigger then the calculated x^2 value which means there is a lineer relation and the null hypothesis is true for this data set.

Chat GPT used a pandas table to organize end control the operations It also had a better understanding of Statatistics then me so it could get some values much better then I did.

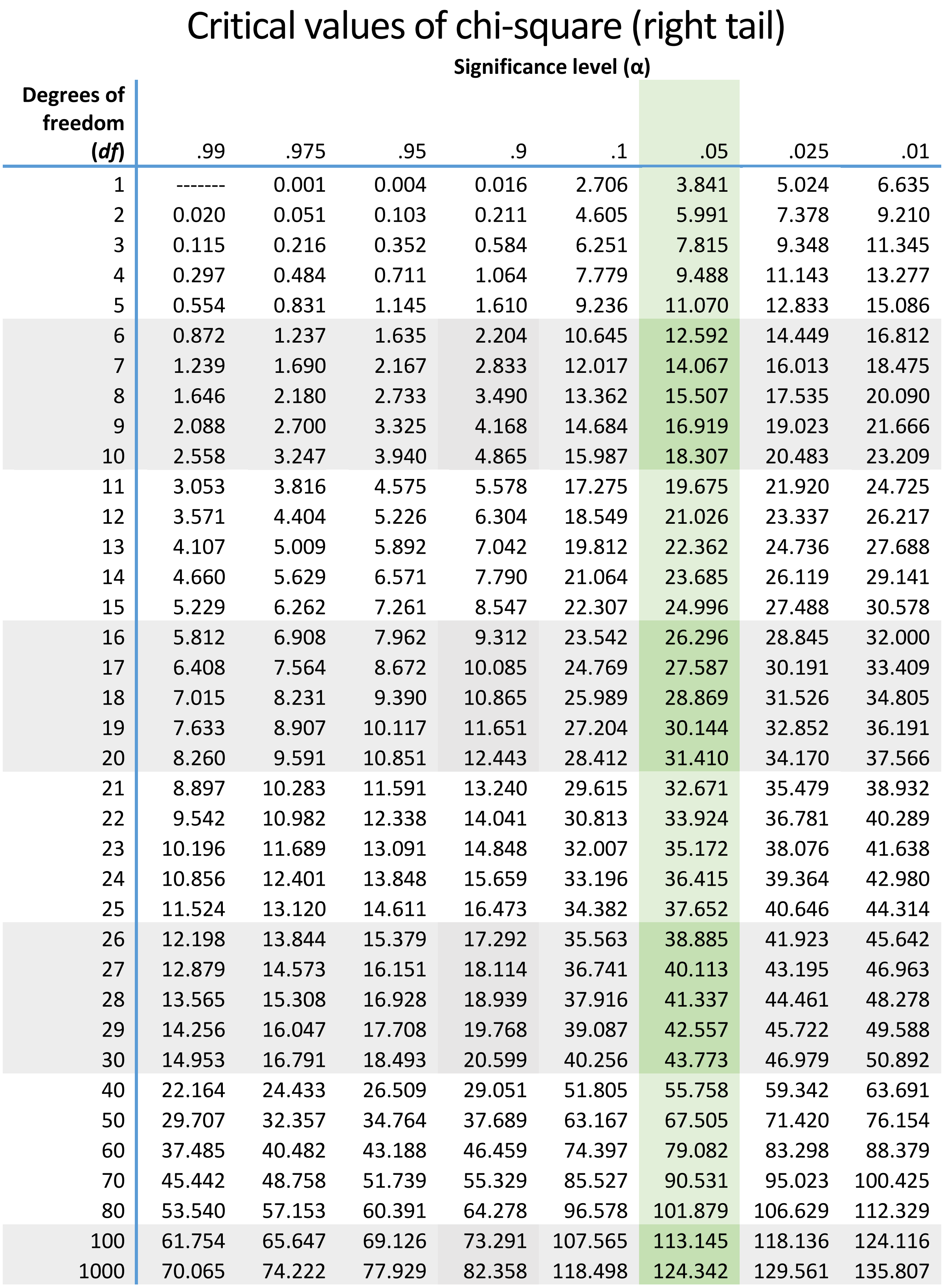


```python
import pandas as pd
import numpy as np
import scipy.stats as stats
# create a pandas dataframe with the data
df = pd.DataFrame({
    'cricket_value': [80, 20, 30, 90, 10, 95, 70, 25, 15, 85],
    'temperature': [30, 30, 30, 20, 20, 20, 10, 10, 10, 10]
})
# fit a linear regression model
model = np.polyfit(df['temperature'], df['cricket_value'], 1)
predicted = np.polyval(model, df['temperature'])
# calculate the residual sum of squares
rss = np.sum((df['cricket_value'] - predicted)**2)
# calculate the degrees of freedom
dof = len(df) - 2
# calculate the variance of the residuals
variance = rss / dof
# calculate the expected values assuming no relationship between variables
expected = np.polyval(model, df['temperature'])
expected_counts, _ = np.histogram(expected, bins=10)
# calculate the observed values
observed_counts, _ = np.histogram(df['cricket_value'], bins=10)
# perform the chi-square test
chi2, p_value = stats.chisquare(observed_counts, expected_counts)
```



Turney, S. (2022). Chi-Square (X2) Tests | Types, Formula & Examples. Scribbr.
https://www.scribbr.com/statistics/chi-square-tests/
Admin. (2021). Chi-Square Test | How to Calculate Chi-square using Formula with Example. BYJUS.
https://byjus.com/maths/chi-square-test/
Mehta, S. (2022). A beginner’s guide to Chi-square test in python from scratch. Analytics India Magazine.
https://analyticsindiamag.com/a-beginners-guide-to-chi-square-test-in-python-from-scratch/

I used this table to determine the critical value. It is taken from the first page in my referances



Critical values of chi-square (right tail)

Degrees of freedom (*df*)	Significance level (α) .99	.975	.95	.9	.1	.05	.025	.01
1	-------	0.001	0.004	0.016	2.706	3.841	5.024	6.635
2	0.020	0.051	0.103	0.211	4.605	5.991	7.378	9.210
3	0.115	0.216	0.352	0.584	6.251	7.815	9.348	11.345
4	0.297	0.484	0.711	1.064	7.779	9.488	11.143	13.277
5	0.554	0.831	1.145	1.610	9.236	11.070	12.833	15.086
6	0.872	1.237	1.635	2.204	10.645	12.592	14.449	16.812
7	1.239	1.690	2.167	2.833	12.017	14.067	16.013	18.475
8	1.646	2.180	2.733	3.490	13.362	15.507	17.535	20.090
9	2.088	2.700	3.325	4.168	14.684	16.919	19.023	21.666
10	2.558	3.247	3.940	4.865	15.987	18.307	20.483	23.209
11	3.053	3.816	4.575	5.578	17.275	19.675	21.920	24.725
12	3.571	4.404	5.226	6.304	18.549	21.026	23.337	26.217
13	4.107	5.009	5.892	7.042	19.812	22.362	24.736	27.688
14	4.660	5.629	6.571	7.790	21.064	23.685	26.119	29.141
15	5.229	6.262	7.261	8.547	22.307	24.996	27.488	30.578
16	5.812	6.908	7.962	9.312	23.542	26.296	28.845	32.000
17	6.408	7.564	8.672	10.085	24.769	27.587	30.191	33.409
18	7.015	8.231	9.390	10.865	25.989	28.869	31.526	34.805
19	7.633	8.907	10.117	11.651	27.204	30.144	32.852	36.191
20	8.260	9.591	10.851	12.443	28.412	31.410	34.170	37.566
21	8.897	10.283	11.591	13.240	29.615	32.671	35.479	38.932
22	9.542	10.982	12.338	14.041	30.813	33.924	36.781	40.289
23	10.196	11.689	13.091	14.848	32.007	35.172	38.076	41.638
24	10.856	12.401	13.848	15.659	33.196	36.415	39.364	42.980
25	11.524	13.120	14.611	16.473	34.382	37.652	40.646	44.314
26	12.198	13.844	15.379	17.292	35.563	38.885	41.923	45.642
27	12.879	14.573	16.151	18.114	36.741	40.113	43.195	46.963
28	13.565	15.308	16.928	18.939	37.916	41.337	44.461	48.278
29	14.256	16.047	17.708	19.768	39.087	42.557	45.722	49.588
30	14.953	16.791	18.493	20.599	40.256	43.773	46.979	50.892
40	22.164	24.433	26.509	29.051	51.805	55.758	59.342	63.691
50	29.707	32.357	34.764	37.689	63.167	67.505	71.420	76.154
60	37.485	40.482	43.188	46.459	74.397	79.082	83.298	88.379
70	45.442	48.758	51.739	55.329	85.527	90.531	95.023	100.425
80	53.540	57.153	60.391	64.278	96.578	101.879	106.629	112.329
100	61.754	65.647	69.126	73.291	107.565	113.145	118.136	124.116
1000	70.065	74.222	77.929	82.358	118.498	124.342	129.561	135.807