Q1

$$\psi = \frac{4n_{He}}{n_p+n_n} = \frac{2n_n\frac{1}{n_p}}{n_p+n_n\frac{1}{n_p}} = \frac{2f}{1+f}$$

Q2

ראינו בתרגול שעבור זמן מחצית חיים קצר יותר
פי 6 יהיה פחות הליום (בערך שישית מהערך
המקורי). אז אם זמן מחצית החיים היה קטן בסדרי גודל
לא היה הליום ביקום.

Q3

1) $a = \left(\frac{t}{t_0}\right)^{1/2}$ יקום של) קרינה ולכן

$\rho \propto a^{-4} = \left(\frac{t}{t_0}\right)^{-2}$

$$\Rightarrow \quad t = t_0\left(\frac{\rho_{sun}}{\rho_{crit}}\right)^{-1/2} = 14\,Gyr \cdot \left(\frac{\frac{a}{c^2}(10^7 K)^4}{\rho_{crit}}\right)^{-1/2} = 4.6\cdot 10^6\,s$$

$$\rho_{sun}\,c^2 = a\,T_{sun}^4$$

2) $2.7 k_B T = 1\,TeV$ אנר' ממוצעת לחלקיק (קרינה)

$T = 4.3\cdot 10^{15}\,K$

$k_B T = MeV\left(\frac{t}{1s}\right)^{-1/2}$ הפעם נשתמש בקירוב

$\Rightarrow \quad t = 7.3\cdot 10^{-12}\,s$

$$\frac{1}{n-1}\frac{1}{x^{n-1}} \xrightarrow{\partial_x} \frac{1}{x^n}$$

Q4

$$\Gamma \propto T^n \propto a^{-n} \propto t^{-nm}$$

מספר
התנגשויות
ממוצע

$$= \int_{t_0}^{\infty} \Gamma(t)\,dt = \int_{t_0}^{\infty} \Gamma_0\left(\frac{t}{t_0}\right)^{-nm} dt = \Gamma_0 t_0^{nm}\int_{t_0}^{\infty} t^{-nm}\,dt$$

$$= \frac{\Gamma_0 t_0^{nm}}{nm-1}\,t_0^{1-nm} = \frac{1}{nm-1}\,\Gamma_0 t_0 = \frac{m}{nm-1}\,\frac{\Gamma_0}{H(t_0)}$$

$$H = \frac{\dot a}{a} = \frac{m t^{m-1}}{t^m} = \frac{m}{t}$$

$N = \frac{m}{nm-1}$ אז $\frac{\Gamma_0}{H(t_0)} = 1$ אם

אם $m > \frac{1}{2}$ ו- $n > 3$ אז לבדוק מקרים אלו:

$$N = \frac{\frac{1}{2}}{\frac{3}{2}-1} = 1\,!$$

ועבור $m = \frac{3}{2}$. $n = 4$:
$$N = \frac{\frac{3}{2}}{6-1} = \frac{3}{10} < 1$$

בקיצור הכוון שבוודאי יש מכפלה אינסופית ~ ערכים גדולים

מאלה $N < 1$.

Q5

1) $U = Lt = 8.8 \cdot 10^{43} \frac{erg}{s} \cdot 10^{10} \cdot 3 \cdot 10^{7} = 26.5 \cdot 10^{60}\ erg$

2) $N \left[\frac{\varepsilon}{\frac{\varepsilon}{N}}\right] = \frac{U}{28.4\,MeV} = 5.8 \cdot 10^{65}$

3) $M = 2 \cdot 10^{44} g = 2 \cdot 10^{41}\ kg$

מסת ההליום הנוצרה: $4 \cdot m_p \cdot 5.8 \cdot 10^{65} = 3.8 \cdot 10^{39}\ kg$

$$Y' = \frac{3.8 \cdot 10^{39} + 0.24 \cdot 2 \cdot 10^{41}}{2 \cdot 10^{41}} = \frac{3.8 \cdot 10^{-2} + 0.48}{2} = 0.26$$

$$\Rightarrow \quad \Delta Y = 0.02$$

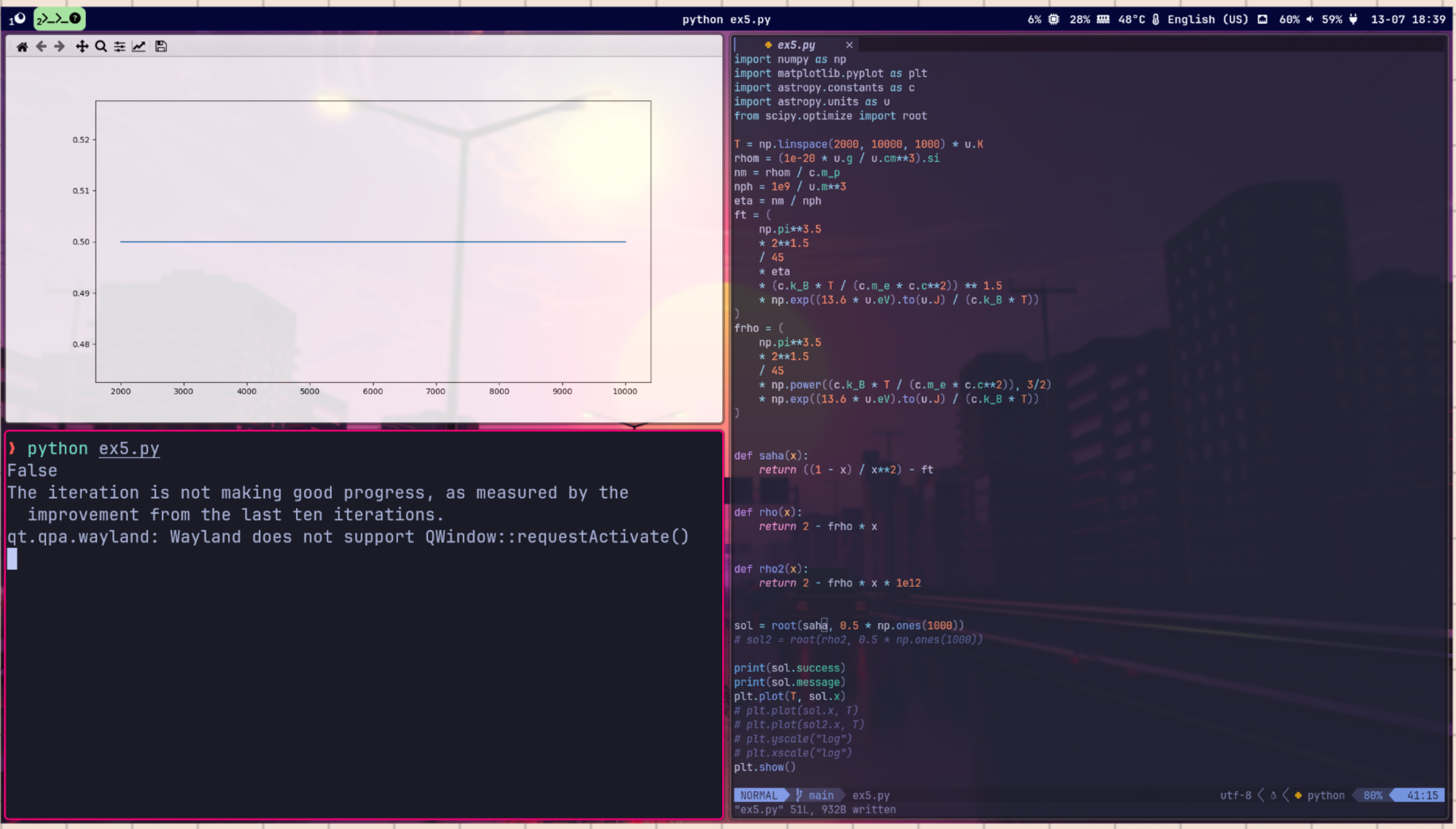
שינוי של 2% גם אם ההנחה שגה ל המקור להליום

חדש. כלומר ששמש לא מייצרת הרבה הליום ביחס לכמות

ההליום המקורית הקדום.

Q5

לצערי ניסיתי ממש הרבה ולא
הצלחתי משו. לא יודע מה הבעיה שלי.



```
❯ python ex5.py
False
The iteration is not making good progress, as measured by the
  improvement from the last ten iterations.
qt.qpa.wayland: Wayland does not support QWindow::requestActivate()
```

```python
import numpy as np
import matplotlib.pyplot as plt
import astropy.constants as c
import astropy.units as u
from scipy.optimize import root

T = np.linspace(2000, 10000, 1000) * u.K
rhom = (1e-20 * u.g / u.cm**3).si
nm = rhom / c.m_p
nph = 1e9 / u.m**3
eta = nm / nph
ft = (
    np.pi**3.5
    * 2**1.5
    / 45
    * eta
    * (c.k_B * T / (c.m_e * c.c**2)) ** 1.5
    * np.exp((13.6 * u.eV).to(u.J) / (c.k_B * T))
)
frho = (
    np.pi**3.5
    * 2**1.5
    / 45
    * np.power((c.k_B * T / (c.m_e * c.c**2)), 3/2)
    * np.exp((13.6 * u.eV).to(u.J) / (c.k_B * T))
)


def saha(x):
    return ((1 - x) / x**2) - ft


def rho(x):
    return 2 - frho * x


def rho2(x):
    return 2 - frho * x * 1e12


sol = root(saha, 0.5 * np.ones(1000))
# sol2 = root(rho2, 0.5 * np.ones(1000))

print(sol.success)
print(sol.message)
plt.plot(T, sol.x)
# plt.plot(sol.x, T)
# plt.plot(sol2.x, T)
# plt.yscale("log")
# plt.xscale("log")
plt.show()
```

```
"ex5.py" 51L, 932B written
```

```
❯ python ex5.py
True
The solution converged.
qt.qpa.wayland: Wayland does n
ot support QWindow::requestAct
ivate()
```

```python
import numpy as np
import matplotlib.pyplot as plt
import astropy.constants as c
import astropy.units as u
from scipy.optimize import root

T = np.linspace(2000, 10000, 1000) * u.K
rhom = (1e-20 * u.g / u.cm**3).si
nm = rhom / c.m_p
nph = 1e9 / u.m**3
eta = nm / nph
ft = (
    np.pi**3.5
    * 2**1.5
    / 45
    * eta
    * (c.k_B * T / (c.m_e * c.c**2)) ** 1.5
    * np.exp((13.6 * u.eV).to(u.J) / (c.k_B * T))
)
frho = (
    np.pi**3.5
    * 2**1.5
    / 45
    * np.power((c.k_B * T / (c.m_e * c.c**2)), 3/2)
    * np.exp((13.6 * u.eV).to(u.J) / (c.k_B * T))
)


def saha(x):
    return ((1 - x) / x**2) - ft


def rho(x):
    return 2 - frho * x


def rho2(x):
    return 2 - frho * x * 1e12


sol = root(rho, 0.5 * np.ones(1000))
sol2 = root(rho2, 0.5 * np.ones(1000))

print(sol.success)
print(sol.message)
# plt.plot(T, sol.x)
plt.plot(sol.x, T)
plt.plot(sol2.x, T)
plt.yscale("log")
plt.xscale("log")
plt.show()
```

```
"ex5.py" 51L, 923B written
```