

სასწავლო კურსის სახელწოდება:	ხელოვნური ინტელექტი და მანქანური სწავლების საწყისები
ლექტორი:	ალექსანდრე ჩახვაძე
სტუდენტის სახელი და გვარი	დათა ჭანუყვაძე

ქვიზი 2 შუალედური გამოცდა ნიმუში

სულ 20 ქულა

გამოიყენე ნებისმიერი გარემო, რაც კომპიუტერზე იქნება დაინსტალირებული. ჩასვი სათანადო ადგილას რელევანტური ინფორმაცია. შეგიძლია გამოიყენო ფაილი `codes.txt`

- (1 ქულა) მოცემული ვექტორებისთვის პითონის საშუალებით გამოინგარიშე ა) სკალარული ნამრავლი ანუ dot პროდაქტი; ბ) კოსინუს მსგავსების მნიშვნელობა.

$$a = (-5; -4; 0; 3; 5; 4); b = (1; 4; 4; 3; 1; 2)$$

კოდი:

#a

```
import numpy as np
```

```
a = np.array([-5,-4,0,3,5,4])
```

```
b = np.array([1,4,4,3,1,2])
```

```
scalar_product = np.dot(a,b)
```

```
print(scalar_product)
```

#b

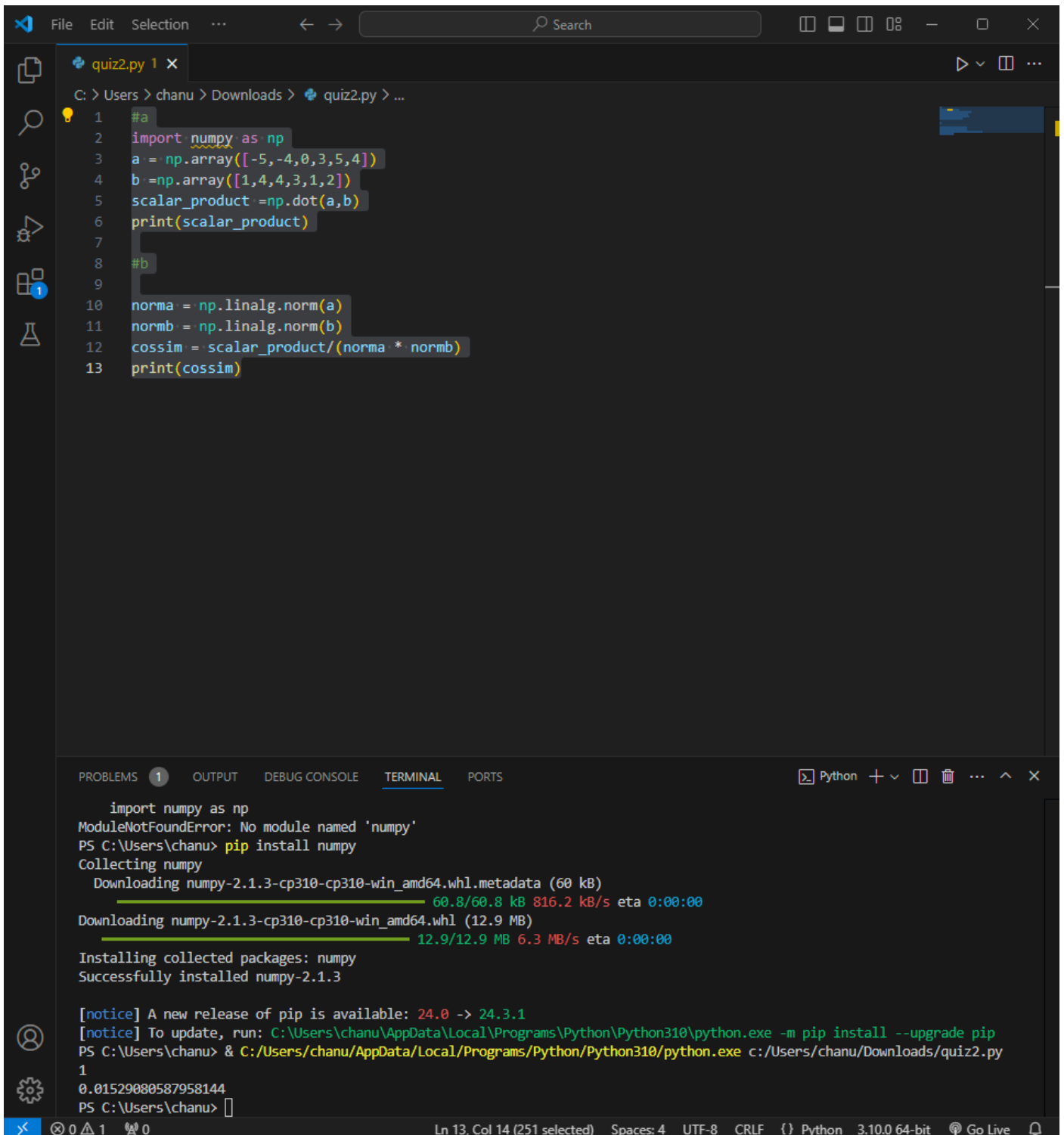
```
norma = np.linalg.norm(a)
```

```
normb = np.linalg.norm(b)
```

```
cossim = scalar_product/(norma * normb)
```

```
print(cossim)
```

სკრინი:



```

1  #a
2  import numpy as np
3  a = np.array([-5,-4,0,3,5,4])
4  b = np.array([1,4,4,3,1,2])
5  scalar_product = np.dot(a,b)
6  print(scalar_product)
7
8  #b
9
10 norma = np.linalg.norm(a)
11 normb = np.linalg.norm(b)
12 cossim = scalar_product/(norma * normb)
13 print(cossim)

```

```

import numpy as np
ModuleNotFoundError: No module named 'numpy'
PS C:\Users\chanu> pip install numpy
Collecting numpy
  Downloading numpy-2.1.3-cp310-cp310-win_amd64.whl.metadata (60 kB)
    Downloading numpy-2.1.3-cp310-cp310-win_amd64.whl (12.9 MB)
      Installing collected packages: numpy
      Successfully installed numpy-2.1.3

[notice] A new release of pip is available: 24.0 -> 24.3.1
[notice] To update, run: C:\Users\chanu\AppData\Local\Programs\Python\Python310\python.exe -m pip install --upgrade pip
PS C:\Users\chanu> & C:/Users/chanu/AppData/Local/Programs/Python/Python310/python.exe c:/Users/chanu/Downloads/quiz2.py
1
0.01529080587958144
PS C:\Users\chanu>

```

2. (3 ქულა) მოცემული მატრიცისათვის პითონის საშუალებით გამოინგარიშე ა) დეტერმინანტი; ბ) ტრანსპონირებული; გ) შებრუნებული.

$$A = \begin{pmatrix} 4 & 3 & 7 & -1 \end{pmatrix}$$

კოდი:

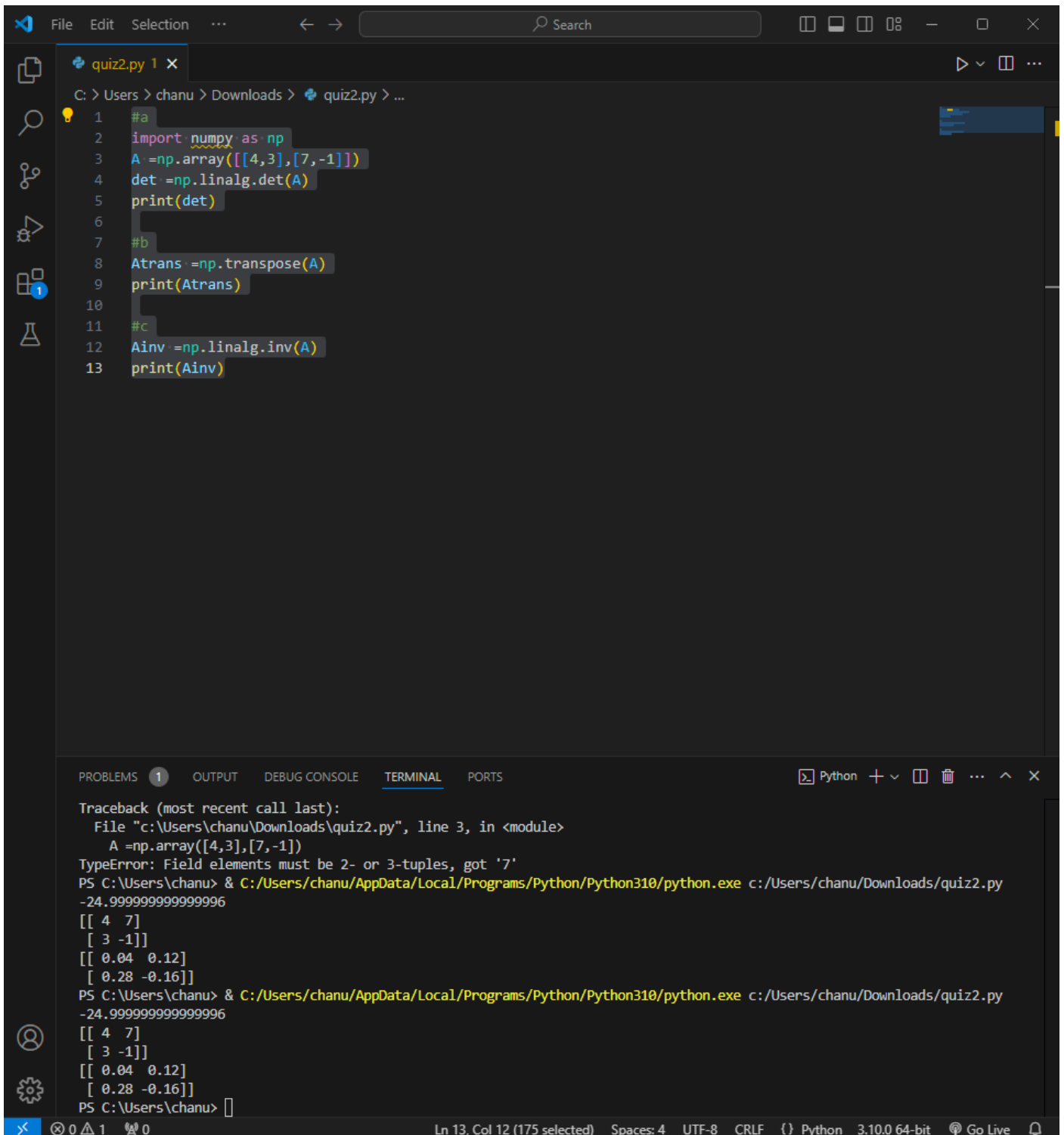
#a

```
import numpy as np
A = np.array([[4,3],[7,-1]])
det = np.linalg.det(A)
print(det)
```

```
#b
Atrans = np.transpose(A)
print(Atrans)
```

```
#c
Ainv = np.linalg.inv(A)
print(Ainv)
```

სკრინი:



```

1  #a
2  import numpy as np
3  A = np.array([[4,3],[7,-1]])
4  det = np.linalg.det(A)
5  print(det)
6
7  #b
8  Atrans = np.transpose(A)
9  print(Atrans)
10
11 #c
12 Ainv = np.linalg.inv(A)
13 print(Ainv)

```

```

Traceback (most recent call last):
  File "c:\Users\chanu\Downloads\quiz2.py", line 3, in <module>
    A = np.array([4,3],[7,-1])
TypeError: Field elements must be 2- or 3-tuples, got '7'
PS C:\Users\chanu> & C:/Users/chanu/AppData/Local/Programs/Python/Python310/python.exe c:/Users/chanu/Downloads/quiz2.py
-24.999999999999996
[[ 4  7]
 [ 3 -1]]
[[ 0.04  0.12]
 [ 0.28 -0.16]]
PS C:\Users\chanu> & C:/Users/chanu/AppData/Local/Programs/Python/Python310/python.exe c:/Users/chanu/Downloads/quiz2.py
-24.999999999999996
[[ 4  7]
 [ 3 -1]]
[[ 0.04  0.12]
 [ 0.28 -0.16]]
PS C:\Users\chanu>

```

3. (4 ქულა) ტექნიკური მოთხოვნის მიხედვით ასაგებია ერთ პანელზე 3 ფუნქცია. განსაზღვრის არე იყოს $[-4,5]$. პირველი ფუნქცია $y=-2x-3$ იყოს ყვითელი, მეორე ფუნქცია $y=-x^3$ ლურჯი, ხოლო მესამე $y=5-3x$ მეწამული. ფუნქციებს მიაწერეთ რიგითობა.

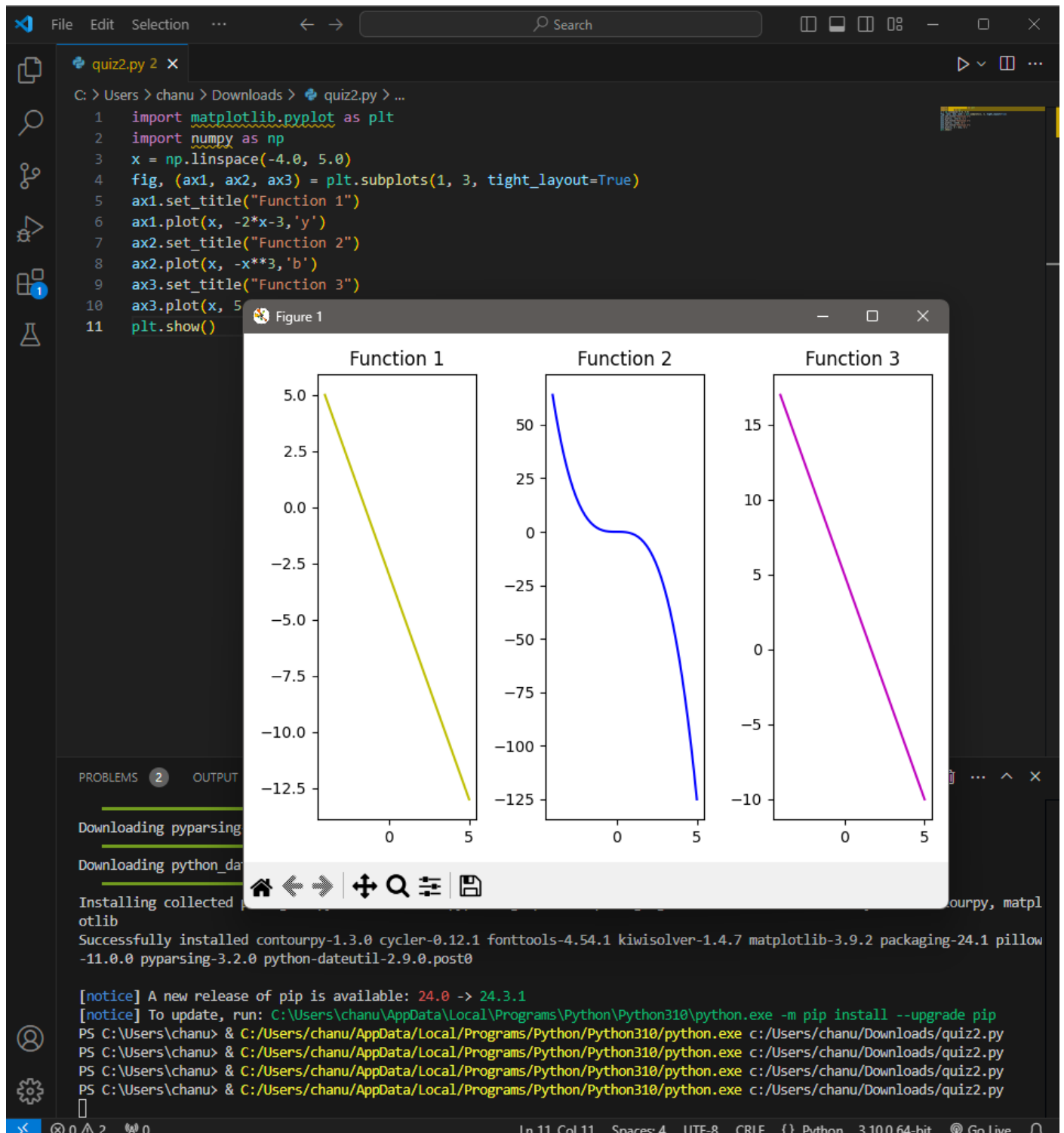
#colors:

r - red, g - green, b - blue, c - cyan, y - yellow, m - magenta, k - black, w - white

კოდი:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-4.0, 5.0)
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, tight_layout=True)
ax1.set_title("Function 1")
ax1.plot(x, -2*x-3, 'y')
ax2.set_title("Function 2")
ax2.plot(x, -x**3, 'b')
ax3.set_title("Function 3")
ax3.plot(x, 5 - 3*x, 'm')
plt.show()
```

სკრინი:



4. (4 ქულა) ვთქვათ მოცემული გაქვთ შემდეგი მასივები

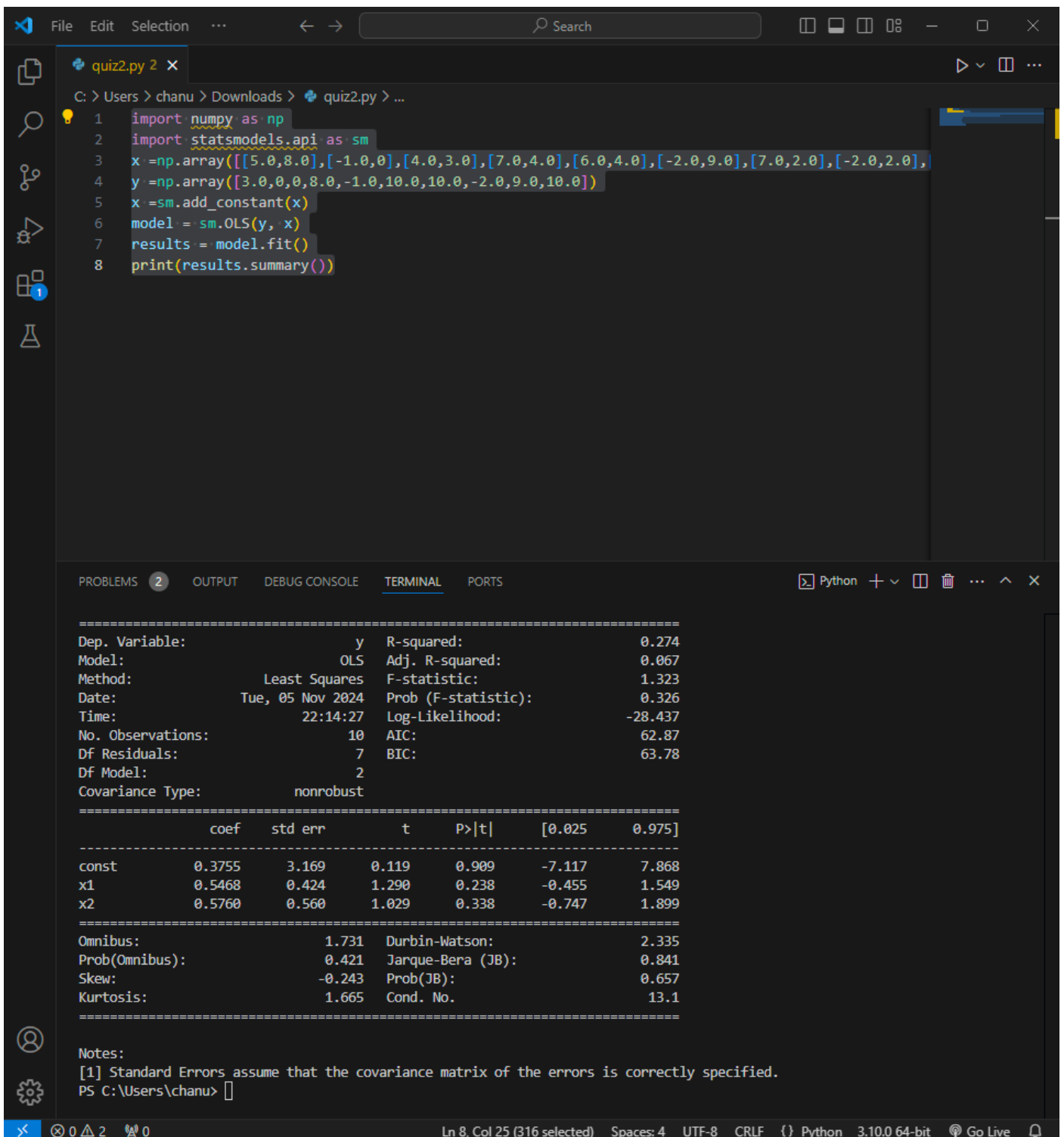
x1	x2	y
5	8	3
-1	0	0
4	3	0
7	4	8
6	4	-1
-2	9	10
7	2	10
-2	-2	-2
7	1	9
7	6	10

ააგეთ უმცირეს კვადრატთა მეთოდით წრფივი რეგრესია და გამოიტანეთ მოდელის შემფასებელი პარამეტრები და რეპორტი.

კოდი:

```
import numpy as np
import statsmodels.api as sm
x = np.array([[5.0,8.0],[-1.0,0],[4.0,3.0],[7.0,4.0],[6.0,4.0],[-2.0,9.0],[7.0,2.0],[-2.0,2.0],[7.0,1.0],[7.0,6.0]])
y = np.array([3.0,0,0,8.0,-1.0,10.0,10.0,-2.0,9.0,10.0])
x = sm.add_constant(x)
model = sm.OLS(y, x)
results = model.fit()
print(results.summary())
```

სკრინი:



```

1 import numpy as np
2 import statsmodels.api as sm
3 x = np.array([[5.0, 8.0], [-1.0, 0], [4.0, 3.0], [7.0, 4.0], [6.0, 4.0], [-2.0, 9.0], [7.0, 2.0], [-2.0, 2.0],
4 y = np.array([3.0, 0.0, 8.0, -1.0, 10.0, 10.0, -2.0, 9.0, 10.0])
5 x = sm.add_constant(x)
6 model = sm.OLS(y, x)
7 results = model.fit()
8 print(results.summary())

```

```

=====
Dep. Variable:          y      R-squared:          0.274
Model:                OLS    Adj. R-squared:       0.067
Method:             Least Squares  F-statistic:        1.323
Date:                Tue, 05 Nov 2024  Prob (F-statistic):    0.326
Time:                22:14:27  Log-Likelihood:     -28.437
No. Observations:      10    AIC:                62.87
Df Residuals:          7    BIC:                63.78
Df Model:              2
Covariance Type:      nonrobust
=====
               coef    std err          t      P>|t|      [0.025     0.975]
-----
const         0.3755    3.169      0.119    0.909    -7.117     7.868
x1            0.5468    0.424      1.290    0.238    -0.455     1.549
x2            0.5760    0.560      1.029    0.338    -0.747     1.899
=====
Omnibus:                 1.731    Durbin-Watson:       2.335
Prob(Omnibus):           0.421    Jarque-Bera (JB):     0.841
Skew:                    -0.243    Prob(JB):             0.657
Kurtosis:                 1.665    Cond. No.             13.1
=====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
PS C:\Users\chanu>

```

5. (4 ქულა) დაწერეთ კოდი რომელიც შექმნის seaborn-ის dataset tips-ზე დაყრდნობით შემდეგ მოდელებს: წრფივი რეგრესია, k უახლოესი მეზობლის რეგრესია, random forest და svm. დამოკიდებულ ცვლადად აიღეთ **total_bill**, ხოლო სხვა დანარჩენი კი დამოუკიდებელ ფაქტორებად. მოახდინეთ კატეგორიული სვეტების გადაყვანა რიცხვითში. (კატეგორიულია: 'sex', 'smoker', 'day', 'time' სვეტები).

დაყავით მოდელი 70% სატრენინგოდ დანარჩენი 30% კი სატესტოდ. მოახდინეთ მონაცემების მასშტაბირება და დაასტანდარტულეთ. მიღებული მოდელები შეაფასეთ შემდეგი მეტრიკების მეშვეობით R2, საშუალო აბსოლუტური გადახრა, საშუალო კვადრატული გადახრა და ფესვი საშუალო კვადრატული გადახრიდან.

შენიშვნა: შუალედურზე მოდელებიდან წრფივი რეგრესია და დანარჩენი სამიდან კიდეც 1 შეგზავდება.

კოდი:

```
import pandas as pd
import numpy as np
import seaborn as sns
```

```
# Load the correct dataset
```

```
tips_df = sns.load_dataset("tips")
X = tips_df.drop(['total_bill'], axis=1)
y = tips_df["total_bill"]
```

```
# Separate numerical and categorical features
```

```
numerical = X.drop(['sex', 'smoker', 'day', 'time'], axis=1)
categorical = X[['sex', 'smoker', 'day', 'time']]
```

```
# Convert categorical data to dummy variables
```

```
cat_numerical = pd.get_dummies(categorical, drop_first=True)
X = pd.concat([numerical, cat_numerical], axis=1)
```

```
# Split data into training and test sets
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=0)
```

```
# Scale the data
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Linear Regression Model

```
from sklearn.linear_model import LinearRegression
```

```
lin_reg = LinearRegression()
```

```
lin_reg.fit(X_train, y_train)
```

```
y_pred = lin_reg.predict(X_test)
```

```
from sklearn import metrics
```

```
print('LinearRegression:')
```

```
print('R^2:', metrics.r2_score(y_test, y_pred))
```

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
```

```
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
```

```
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

KNeighbors Regressor Model

```
from sklearn.neighbors import KNeighborsRegressor
```

```
knn_reg = KNeighborsRegressor(n_neighbors=5)
```

```
knn_reg.fit(X_train, y_train)
```

```
y_pred = knn_reg.predict(X_test)
```

```
print('KNeighborsRegressor:')
```

```
print('R^2:', metrics.r2_score(y_test, y_pred))
```

```
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
```

```
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
```

```
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

RandomForest Regressor Model

```
from sklearn.ensemble import RandomForestRegressor
```

```
rf_reg = RandomForestRegressor(random_state=42, n_estimators=500)
```

```
rf_reg.fit(X_train, y_train)
```

```
y_pred = rf_reg.predict(X_test)
```

```
print('RandomForestRegressor:')
print('R^2:', metrics.r2_score(y_test, y_pred))
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

# Support Vector Regressor (SVR) Model
from sklearn import svm
svm_reg = svm.SVR()
svm_reg.fit(X_train, y_train)
y_pred = svm_reg.predict(X_test)

print('SVM Regressor:')
print('R^2:', metrics.r2_score(y_test, y_pred))
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

სკრინი:

```

File Edit Selection ... Search
quiz2.py 9+ x
C: > Users > chanu > Downloads > quiz2.py > ...
60 print('R^2: ', metrics.r2_score(y_test, y_pred))
61 print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
62 print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
63 print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
64
65 # Support Vector Regressor (SVR) Model
66 from sklearn import svm
67 svm_reg = svm.SVR()
68 svm_reg.fit(X_train, y_train)
69 y_pred = svm_reg.predict(X_test)
70
71 print('SVM Regressor:')
72 print('R^2: ', metrics.r2_score(y_test, y_pred))
73 print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
74 print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
75 print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
76

PROBLEMS 12 OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + - [ ] [ ] ... ^ x

x Getting requirements to build wheel did not run successfully.
  exit code: 1
  See above for output.

note: This error originates from a subprocess, and is likely not a problem with pip.
PS C:\Users\chanu> & C:/Users/chanu/AppData/Local/Programs/Python/Python310/python.exe c:/Users/chanu/Downloads/quiz2.py
LinearRegression:
R^2: 0.5693293373056254
Mean Absolute Error: 4.2746674181541975
Mean Squared Error: 29.638496950275794
Root Mean Squared Error: 5.44412499399819
KNeighborsRegressor:
R^2: 0.3908426701922607
Mean Absolute Error: 4.666297297297296
Mean Squared Error: 41.92184243243243
Root Mean Squared Error: 6.474707903251885
RandomForestRegressor:
R^2: 0.369440163670696
Mean Absolute Error: 4.6221443441168395
Mean Squared Error: 43.39475010693965
Root Mean Squared Error: 6.587469173130121
SVM Regressor:
R^2: 0.3313376609406923
Mean Absolute Error: 4.783302689496196
Mean Squared Error: 46.016941513932466
Root Mean Squared Error: 6.783578813128986
PS C:\Users\chanu>
Ln 76, Col 1 (2913 selected) Spaces: 4 UTF-8 CRLF {} Python 3.10.0 64-bit Go Live

```