**Код:**

import numpy as np  
n=5  
  
#1. За допомогою функції с() сформувати вектор, що містить:  
# а) всі цілі числа від 15n до n у спадному порядку;  
# б) вектор, елементами якого є літери Вашого прізвища.  
print('-'\*20)  
print('#1')  
  
def ca(n):  
 return [\*range(15\*n, n, -1)]  
print(ca(n))  
  
def cb(str):  
 return list(str)  
print(cb('Погодаєв'))  
  
  
#2. За допомогою функції seq() створити:  
#а) вектор, що містить 40 - n чисел, що є елементами арифметичної прогресії з першим елементом 3n - 1 та останнім елементом n;  
#б) вектор, що містить взяті з кроком 0,2 числа від 2n до 6n.  
print('-'\*20)  
print('#2')  
  
def seqa(n):  
 return [3\*n - 1 + i\*(1 - 2\*n)/(39 - n) for i in range(40-n)]  
print(seqa(n))  
  
def seqb(n):  
 m = []  
 value = 2\*n  
 while value <= 6\*n:  
 m.append(value)  
 value = round(value+0.2, 1)  
 return m  
print(seqb(n))  
  
#3. За допомогою функції matrix() сформувати;  
#а) з елементів першого вектора попереднього завдання матрицю, що містить 4 рядки та 3 стовпчики;  
#б) матрицю, що містить 4n рядків та 3 стовпчики з векторів, що є взятими у висхідному порядку послідовностями всіх цілих чисел від 1 до n + 7 та від n - 8 до 2n, відповідно.  
print('-'\*20)  
print('#3')  
  
def matrixa(n):  
 m = []  
 q = seqa(n)  
 for i in range(4):  
 m.append([])  
 for j in range(3):  
 m[i].append(q[i\*4+j])  
 return m  
  
print(matrixa(n))  
  
  
def matrixb(n):  
 vector1 = np.arange(1, n + 9)  
 vector2 = np.arange(n - 8, 2 \* n + 1)  
 matrix = np.zeros((4 \* n, 3))  
 combined\_vector = np.concatenate((vector1, vector2))  
 while True:  
 for i in range(4 \* n):  
 for j in range(3):  
 matrix[i][j] = combined\_vector[(j\*4\*n+i)%len(combined\_vector)]  
 break  
 return matrix  
  
print(matrixb(n))  
  
  
#4. Знайти:  
#а) суму другого елемента першого стовпчика та першого елемента другого стовпчика отриманої в останньому завданні матриці;  
#б) вектор, елементи якого є сумами відповідних елементів другого та третього стовпчиків цієї матриці.  
print('-'\*20)  
print('#4')  
  
matr = matrixb(n)  
print(matr[1][0]+matr[0][1])  
matrix4b = []  
for i in range(4\*n):  
 matrix4b.append(matr[i][0] + matr[i][1] + matr[i][2])  
print(matrix4b)  
  
#5.  
#а) Створити матрицю розмірності 5\*4, елементами якої є цілі числа від n + 5 до n + 24;  
#б) вивести елементи першого рядка цієї матриці;  
#в) присвоїти цим елементам значення n + 3 та вивести результат перетворення;  
#г) прирівняти до n/2 всі елементи, які є більшими, ніж 2n та вивести результат перетворення.  
print('-'\*20)  
print('#5a')  
matrix5a = np.arange(n + 5, n + 25).reshape(5, 4)  
print(matrix5a)  
print('#5b')  
print(matrix5a[0])  
print('#5c')  
matrix5c = matrix5a  
matrix5c[0, :] = n + 3  
print(matrix5c)  
print('#5d')  
matrix5c[matrix5c > 2 \* n] = n / 2  
print(matrix5c)  
  
#6. Побудувати масив у вигляді таких трьох матриць розмірності n\*3:  
#перша матриця містить послідовність цілих чисел від 1 до 3n, взятих у спадному порядку;  
#друга матриця – послідовність цілих чисел від 3n + 1 до 6n;  
#третя матриця – послідовність цілих чисел від n - 10 до 4n - 11;  
print('-'\*20)  
print('#6')  
  
vector = np.concatenate((np.arange(1, 3\*n+1), np.arange(3\*n+1, 6\*n+1), np.arange(n-10, 4\*n-10)))  
print(vector)  
  
#7. Побудувати транспоновану матрицю для матриці із завдання 5а. Зберегти отриманий масив у вигляді файлу  
# формату \*.csv, зчитати його в R та визначити суму і добуток всіх елементів отриманої матриці.  
print('-'\*20)  
print('#7')  
  
matrix7 = np.transpose(matrix5a)  
file\_path = "output\_matrix.csv"  
# Збереження матриці у файл  
np.savetxt(file\_path, matrix7, delimiter=",")  
# Зчитуємо матрицю з файлу  
read\_matrix = np.loadtxt(file\_path, delimiter=",")  
# Визначаємо суму та добуток  
sum\_of\_elements = np.sum(read\_matrix)  
product\_of\_elements = np.prod(read\_matrix)  
print("Сума всіх елементів:", sum\_of\_elements)  
print("Добуток всіх елементів:", product\_of\_elements)  
  
#8. Створити список, що містить такі компоненти:  
#v1 – вектор назв видів промислової продукції (взяти назви видів з n + 5-го до n + 8 –го із таблиці за посиланням (при визначенні номерів враховувати тільки рядки з даними):  
#http://ukrstat.gov.ua/operativ/operativ2006/pr/prm\_ric/xls/vov20XX\_u.xls;  
#v2 – вектор одиниць вимірювання обсягів виробництва для тих самих видів продукції (одиниці, що повторюються, дублювати не треба);  
#v3 – обсяги виробництва для відповідних видів продукції у 2018 р.  
#Вивести отриманий список на консоль та визначити його структуру  
print('-'\*20)  
print('#8')  
  
import xlrd  
  
n = 5  
start\_row = n + 5 + 4  
end\_row = n + 9 + 4  
file\_name = 'vov20XX\_u.xls'  
# Відкриваємо файл та зчитуємо дані  
try:  
 workbook = xlrd.open\_workbook(file\_name)  
 sheet = workbook.sheet\_by\_index(0)  
  
 # Ініціалізація векторів  
 v1 = []  
 v2 = []  
 v3 = []  
  
 # Зчитуємо назви, одиниці вимірювання та обсяги виробництва  
 for row in range(start\_row, end\_row + 1):  
 industrial\_product = sheet.cell\_value(row, 0)  
 unit\_of\_measurement = sheet.cell\_value(row, 2)  
 production\_volume\_2018 = sheet.cell\_value(row, 10)  
  
 # Додаємо дані до відповідних векторів  
 v1.append(industrial\_product)  
 v2.append(unit\_of\_measurement)  
 v3.append(production\_volume\_2018)  
  
except Exception as e:  
 print(f"Помилка при читанні файлу: {e}")  
  
# Виведення результату  
print("v1 (Назви видів промислової продукції):", v1)  
print("v2 (Одиниці вимірювання обсягів виробництва):", v2)  
print("v3 (Обсяги виробництва у 2018 р.):", v3)  
  
  
#9. Зберегти дані зазначеної у попередньому завданні таблиці у форматі \*.csv зчитати п’ять рядків таблиці  
# даних, починаючи з n-го. Для обраних видів продукції визначити:  
#обсяг виробництва у 2018 р. у відсотках до відповідного показника 2011 р.;  
#абсолютний приріст обсягу виробництва у 2018 р. по відношенню до відповідного показника 2011 р.  
print('-'\*20)  
print('#9')  
  
import csv  
import pandas as pd  
  
n = 4  
start\_row = n  
end\_row = n + 10  
  
# Ім'я файлу  
input\_file\_name = 'vov20XX\_u.xls'  
output\_csv\_file\_name = 'output\_data.csv'  
  
# Зчитуємо дані з Excel і зберігаємо їх у файл CSV  
try:  
 workbook = xlrd.open\_workbook(input\_file\_name)  
 sheet = workbook.sheet\_by\_index(0)  
  
 data = []  
 for row in range(start\_row, end\_row + 1):  
 data.append(sheet.row\_values(row))  
  
 # Збереження у файл CSV  
 with open(output\_csv\_file\_name, 'w', newline='', encoding='utf-8') as csv\_file:  
 csv\_writer = csv.writer(csv\_file)  
 csv\_writer.writerows(data)  
except Exception as e:  
 print(f"Помилка при читанні файлу Excel: {e}")  
  
  
# Читаємо п'ять рядків таблиці з CSV, починаючи з n-го  
df = pd.read\_csv('output\_data.csv', skiprows=5, nrows=5, header=None)  
  
# Виведення перших п'яти рядків  
print("Перші п'ять рядків:")  
print(df)  
  
# Додаємо індекс до DataFrame  
df.reset\_index(inplace=True)  
  
# Конвертація стовпців до числового формату (всі невірні значення замінюються на NaN)  
df.iloc[:, 4] = pd.to\_numeric(df.iloc[:, 4], errors='coerce')  
df.iloc[:, 10] = pd.to\_numeric(df.iloc[:, 10], errors='coerce')  
  
# Обробка NaN значень та обчислення відсотків та абсолютного приросту  
df['Відсоток 2018 від 2011'] = (  
 (df.iloc[:, 10] - df.iloc[:, 4]) / df.iloc[:, 4] \* 100  
)  
  
df['Абсолютний приріст'] = (  
 df.iloc[:, 10] - df.iloc[:, 4]  
)  
  
# Виведення результатів  
print("\nРезультати:")  
print(df[['index', 4, 10, 'Відсоток 2018 від 2011', 'Абсолютний приріст']])  
  
  
#10. Створити вектори: b1, який містить n + 50 випадкових чисел, що підпорядковуються нормальному розподілу  
# із середнім значенням 2n та стандартним відхиленням 0,5n; b2, який містить 60 - n випадкових чисел, що  
# підпорядковуються рівномірному розподілу на відрізку від n до 3n. Написати цикл, що створює вектор b3 за таким  
# правилом: якщо b1i < b2i , то b3i = 2, у протилежному випадку b3i = 0.  
print('-'\*20)  
print('#10')  
  
n = 5  
  
# Створюємо вектор b1 за нормальним розподілом  
b1 = np.random.normal(2 \* n, 0.5 \* n, n + 50)  
  
# Створюємо вектор b2 за рівномірним розподілом  
b2 = np.random.uniform(n, 3 \* n, 60 - n)  
  
# Ініціалізуємо вектор b3  
b3 = np.zeros\_like(b1)  
  
# Створюємо вектор b3 за вказаним правилом  
for i in range(len(b1)):  
 if b1[i] < b2[i]:  
 b3[i] = 2  
  
# Виведення результатів  
print("b1:", b1)  
print("b2:", b2)  
print("b3:", b3)  
  
  
#11. Написати функцію, яка визначає відстань від точки A з координатами (2n, -n) до середини відрізку BC, який задано точками B (2n, n/2)) та C (1 - n, n + 2).  
print('-'\*20)  
print('#11')  
  
def distance\_from\_A\_to\_midpoint(n):  
 # Координати точок A, B, C  
 A = np.array([2 \* n, -n])  
 B = np.array([2 \* n, n / 2])  
 C = np.array([1 - n, n + 2])  
  
 # Знаходимо координати середини відрізку BC  
 midpoint\_BC = (B + C) / 2  
  
 # Обчислюємо відстань між A і серединою BC  
 distance = np.linalg.norm(A - midpoint\_BC)  
  
 return distance  
  
result = distance\_from\_A\_to\_midpoint(n)  
print(f"Відстань від точки A до середини відрізку BC: {result}")

**результат:**

--------------------

#1

[75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47, 46, 45, 44, 43, 42, 41, 40, 39, 38, 37, 36, 35, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6]

['П', 'о', 'г', 'о', 'д', 'а', 'є', 'в']

--------------------

#2

[14.0, 13.735294117647058, 13.470588235294118, 13.205882352941176, 12.941176470588236, 12.676470588235293, 12.411764705882353, 12.147058823529411, 11.882352941176471, 11.617647058823529, 11.352941176470589, 11.088235294117647, 10.823529411764707, 10.558823529411764, 10.294117647058822, 10.029411764705882, 9.764705882352942, 9.5, 9.235294117647058, 8.970588235294118, 8.705882352941178, 8.441176470588236, 8.176470588235293, 7.911764705882353, 7.647058823529412, 7.382352941176471, 7.117647058823529, 6.852941176470588, 6.588235294117647, 6.323529411764706, 6.0588235294117645, 5.794117647058824, 5.529411764705882, 5.264705882352942, 5.0]

[10, 10.2, 10.4, 10.6, 10.8, 11.0, 11.2, 11.4, 11.6, 11.8, 12.0, 12.2, 12.4, 12.6, 12.8, 13.0, 13.2, 13.4, 13.6, 13.8, 14.0, 14.2, 14.4, 14.6, 14.8, 15.0, 15.2, 15.4, 15.6, 15.8, 16.0, 16.2, 16.4, 16.6, 16.8, 17.0, 17.2, 17.4, 17.6, 17.8, 18.0, 18.2, 18.4, 18.6, 18.8, 19.0, 19.2, 19.4, 19.6, 19.8, 20.0, 20.2, 20.4, 20.6, 20.8, 21.0, 21.2, 21.4, 21.6, 21.8, 22.0, 22.2, 22.4, 22.6, 22.8, 23.0, 23.2, 23.4, 23.6, 23.8, 24.0, 24.2, 24.4, 24.6, 24.8, 25.0, 25.2, 25.4, 25.6, 25.8, 26.0, 26.2, 26.4, 26.6, 26.8, 27.0, 27.2, 27.4, 27.6, 27.8, 28.0, 28.2, 28.4, 28.6, 28.8, 29.0, 29.2, 29.4, 29.6, 29.8, 30.0]

--------------------

#3

[[14.0, 13.735294117647058, 13.470588235294118], [12.941176470588236, 12.676470588235293, 12.411764705882353], [11.882352941176471, 11.617647058823529, 11.352941176470589], [10.823529411764707, 10.558823529411764, 10.294117647058822]]

[[ 1. 4. -3.]

[ 2. 5. -2.]

[ 3. 6. -1.]

[ 4. 7. 0.]

[ 5. 8. 1.]

[ 6. 9. 2.]

[ 7. 10. 3.]

[ 8. 1. 4.]

[ 9. 2. 5.]

[10. 3. 6.]

[11. 4. 7.]

[12. 5. 8.]

[13. 6. 9.]

[-3. 7. 10.]

[-2. 8. 1.]

[-1. 9. 2.]

[ 0. 10. 3.]

[ 1. 11. 4.]

[ 2. 12. 5.]

[ 3. 13. 6.]]

--------------------

#4

6.0

[2.0, 5.0, 8.0, 11.0, 14.0, 17.0, 20.0, 13.0, 16.0, 19.0, 22.0, 25.0, 28.0, 14.0, 7.0, 10.0, 13.0, 16.0, 19.0, 22.0]

--------------------

#5a

[[10 11 12 13]

[14 15 16 17]

[18 19 20 21]

[22 23 24 25]

[26 27 28 29]]

#5b

[10 11 12 13]

#5c

[[ 8 8 8 8]

[14 15 16 17]

[18 19 20 21]

[22 23 24 25]

[26 27 28 29]]

#5d

[[8 8 8 8]

[2 2 2 2]

[2 2 2 2]

[2 2 2 2]

[2 2 2 2]]

--------------------

#6

[ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24

25 26 27 28 29 30 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9]

--------------------

#7

Сума всіх елементів: 64.0

Добуток всіх елементів: 268435456.0

--------------------

#8

v1 (Назви видів промислової продукції): ['Крейда', 'Піски будівельні, такі як глинисті, каолінові, полевошпатові (крім кременистих та металоносних пісків)', 'Камінь дроблений (щебінь), який використовується як наповнювач бетону, для дорожнього покриття та подібних цілей (крім гальки, гравію та креміня)', 'Каолін, крім кальцинованого', 'Глини каолінові інші (комова і пластична)']

v2 (Одиниці вимірювання обсягів виробництва): ['тис.т / thsd.t', 'млн.т / mln.t', 'млн.т / mln.t', 'тис.т / thsd.t', 'тис.т / thsd.t']

v3 (Обсяги виробництва у 2018 р.): [237.0, 13.5, 60.4, 2092.0, 148.0]

--------------------

#9

Перші п'ять рядків:

0 ... 10

0 Руди залізні неагломеровані ... 161.0

1 Концентрати залізорудні неагломеровані ... 60.5

2 Концентрати залізорудні агломеровані ... 53.1

3 Гіпс і ангідрит ... 1.4

4 Вапняк, флюс вапняковий та інший вапняковий ка... ... 6.1

[5 rows x 11 columns]

Результати:

index 4 10 Відсоток 2018 від 2011 Абсолютний приріст

0 0 176.0 161.0 -4.046243 -7.0

1 1 67.1 60.5 -8.872180 -5.9

2 2 64.6 53.1 -21.118012 -13.6

3 3 2.2 1.4 -34.782609 -0.8

4 4 20.6 6.1 -71.052632 -16.2

--------------------

#10

b1: [ 8.35935026 12.00403276 9.67747018 7.99133052 8.69535183 12.69606031

10.2665911 12.50594932 9.03170885 11.37615661 6.97566584 8.85936978

11.1074315 5.40122659 9.76107246 8.34693767 8.79175502 9.67818136

10.59936122 10.54067251 14.93701629 11.72485819 11.0106064 9.61563574

6.15355944 8.21944823 9.01622295 8.72289439 9.31904561 8.30352209

9.73875365 15.30008152 6.29627822 14.62659156 9.50035971 6.5173601

10.48018235 7.89632344 9.06360254 7.94720088 8.89599532 8.42005346

11.98013651 8.28228063 12.27604305 7.41796763 8.48086735 8.83835591

12.92540145 11.89028465 10.5709512 7.04343912 10.78024357 11.14046742

12.52776698]

b2: [10.75375655 14.30299749 5.78948486 5.75848222 12.78221716 8.92742933

9.79636809 13.79372012 14.6773928 9.76407925 7.39125337 9.27727562

10.17038601 10.80069675 14.73223023 9.85792651 11.88108079 11.54036936

12.95508603 10.85089852 11.58086343 5.70868213 13.48783173 12.8399753

13.6619308 12.07913585 11.19407736 10.28363143 9.2777521 12.1118206

9.87275676 11.5285083 13.81402775 8.1483156 8.80571802 10.53313324

14.46765627 14.64419076 7.09138703 8.8432317 11.84555539 8.47768069

10.21302896 9.95641617 6.98504742 7.68539776 14.78333359 13.74513783

8.09405999 13.28070684 7.59679537 5.39674335 14.145057 10.28133105

14.45773638]

b3: [2. 2. 0. 0. 2. 0. 0. 2. 2. 0. 2. 2. 0. 2. 2. 2. 2. 2. 2. 2. 0. 0. 2. 2.

2. 2. 2. 2. 0. 2. 2. 0. 2. 0. 0. 2. 2. 2. 0. 2. 2. 2. 0. 2. 0. 2. 2. 2.

0. 2. 0. 0. 2. 0. 2.]

--------------------

#11

Відстань від точки A до середини відрізку BC: 12.002603884157804

Process finished with exit code 0