UAS SEMESTER 1
SK500 – PEMROGRAMAN DALAM SAINS

Aditya Adiaksa 20922320
Akhmad Faeda Insani 20922313
Ahmad Mushawir 20922307
Dwi Putra Prasetya 20922312
Zainuddin 20922319

Bagian 1

1. *Buat suatu dictionary dengan Python yang dapat menyimpan informasi 4-5 jenis kopi dengan terlebih dahulu melakukan kuantifikasi terhadap bahan bahan yang belum memiliki nilai, dengan satuan untuk setiap bahan adalah sama. Cantumkan satuan yang digunakan untuk jumlah bahan. [Nilai: 20]*

Jawab:
Kami membuat beberapa menu minuman kopi dengan jenis dan bahan seperti yang tampil pada tabel dibawah ini:

BAHAN			MENU				
Material	IDR/ gram	Merek	Americano	Cappucino	Cafe Latte	Affogato	Macchiato
Coffe	168	Exelso	30	30	30	30	30
Milk	17	UHT Diamond	-	150	150	-	-
Sugar	16	Gulaku	-	-	-	-	-
Chocolate	133	Mllo	-	10	-	-	-
Water	3	Le Mineralle	155	-	-	-	-
Cream	73	Max Creamer	-	-	-	30	15
Ice Cream	33	Campina	-	-	-	-	-
Whiskey	1257	Jameson	-	-	-	-	-
Ice	1	Atlas Ice Cube	-	-	-	-	-

Script python untuk mernampilkan dictionary coffee

```
BLUE = '\033[94m'
RED = '\033[91m'
END = '\033[0m'

coffee_dict = {
    'material': ['Coffee', 'Milk', 'Sugar', 'Chocolate', 'Water', 'Cream', 'Ice Cream', 'Whiskey', 'Cube Ice'],
    'harga_pergram': [168, 17, 16, 133, 3, 73, 33, 1257, 1],    'americano': [30, 0, 0, 0, 155, 0, 0, 0, 0],
    'cappucino': [30, 150, 0, 10, 0, 0, 0, 0, 0],
    'caffelatte': [30, 150, 0, 0, 0, 0, 0, 0, 0],
    'afogato': [0, 30, 0, 0, 0, 30, 0, 0, 0],
    'machiato': [30, 0, 0, 0, 0, 15, 0, 0, 0]
}

print(RED + "  C" + BLUE + "O" + RED + "FF" + BLUE + "EE " + END + "MENU:")
print()

coffees = ['americano', 'cappucino', 'caffelatte', 'afogato', 'machiato']

for coffee in coffees:
    print(RED + coffee.capitalize() + ":" + END)
    for i in range(len(coffee_dict['material'])):
        material = coffee_dict['material'][i]
        quantity = coffee_dict[coffee][i]
        print("  - " + BLUE + material + ":" + END + " " + str(quantity) + " gram")
    print()
```

Output yang dihasilkan:

```
   COFFEE MENU:

Americano:
  - Coffee: 30 gram
  - Milk: 0 gram
  - Sugar: 0 gram
  - Chocolate: 0 gram
  - Water: 155 gram
  - Cream: 0 gram
  - Ice Cream: 0 gram
  - Whiskey: 0 gram
  - Cube Ice: 0 gram

Cappucino:
  - Coffee: 30 gram
  - Milk: 150 gram
  - Sugar: 0 gram
  - Chocolate: 10 gram
  - Water: 0 gram
  - Cream: 0 gram
  - Ice Cream: 0 gram
  - Whiskey: 0 gram
  - Cube Ice: 0 gram
```

```
Caffelatte:
  - Coffee: 30 gram
  - Milk: 150 gram
  - Sugar: 0 gram
  - Chocolate: 0 gram
  - Water: 0 gram
  - Cream: 0 gram
  - Ice Cream: 0 gram
  - Whiskey: 0 gram
  - Cube Ice: 0 gram

Afogato:
  - Coffee: 0 gram
  - Milk: 30 gram
  - Sugar: 0 gram
  - Chocolate: 0 gram
  - Water: 0 gram
  - Cream: 30 gram
  - Ice Cream: 0 gram
  - Whiskey: 0 gram
  - Cube Ice: 0 gram
```

```
Machiato:
  - Coffee: 30 gram
  - Milk: 0 gram
  - Sugar: 0 gram
  - Chocolate: 0 gram
  - Water: 0 gram
  - Cream: 15 gram
  - Ice Cream: 0 gram
  - Whiskey: 0 gram
  - Cube Ice: 0 gram
```

2. *Konversi data harga kopi yang semula dalam USD untuk setiap pound menjadi dalam IDR untuk per kilogram dan gambarkan grafiknya untuk suatu rentang waktu tertentu. [Nilai: 20]*

Jawab:
Untuk mengkonversi harga kopi USD/pound menjadi IDR/kilogram, kami menggunakan python dengan library pandas, numpy, dan matplotlib.pyplot.
Dalam program tersebut selain kami melakukan konversi harga, kami juga berusaha menampilkan data-data analisis lain yang dibutuhkan diantaranya:

a. Harga rata-rata setahun
b. Harga maksimal/ tertinggi pada tahun tersebut
c. Harga minimal/ terendah pada tahun tersebut
d. Harga Open/ harga di awal tahun
e. Harga Close/ harga di akhir tahun
f. Annual Change/ selisih kenaikan atau penurunan harga dibanding dengan tahun sebelumnya

Script python untuk konversi harga:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Membaca file CSV dengan memisahkan kolom menggunakan tanda koma
data = pd.read_csv('hargakopi.csv', delimiter=',')

# Mengubah nama kolom menjadi lowercase dan menghapus spasi
data.columns = data.columns.str.lower().str.replace(' ', '_')

# Mengubah format kolom 'year_month' menjadi datetime
data['year_month'] = pd.to_datetime(data['year_month'], format='%Y%m', errors='coerce')

# Menghapus baris yang memiliki nilai 'year_month' yang tidak valid
data = data.dropna(subset=['year_month'])

# Menambahkan kolom tahun
data['year'] = data['year_month'].dt.year

# Menampilkan harga asli dari file CSV di sebelah year
data['us_kg'] = data['price']

# Mengubah format kolom 'price' menjadi float dan mengkonversi ke Rupiah
data['price'] = data['price'].astype(float) * 14985.25
```

```python
# Menghitung harga_open dan harga_close per tahun
result = data.groupby('year').agg({'price': ['mean', 'max', 'min'], 'us_kg': 'first'})

# Menghitung harga_open
result['harga_open'] = data.groupby('year')['price'].first()

# Menghitung harga_close
result['harga_close'] = data.groupby('year')['price'].last()

# Menghitung perubahan tahunan dalam persentase (annual_change)
result['annual_change'] = ((result['us_kg'] - result['us_kg'].shift(1)) / result['us_kg']) * 100
result['annual_change'] = result['annual_change'].fillna(0)

# Memformat kolom annual_change dengan dua angka di belakang koma
result['annual_change'] = result['annual_change'].round(2)

# Konversi harga per pound menjadi harga per kilogram
result['us_lbs'] = result['us_kg'] / 2.20462

# Menampilkan hasil dengan kolom "us_perlbs" dan "us_perkg" di sebelah kanan kolom "year"
result = result.reset_index()
result = result[['year', 'us_lbs', 'us_kg', 'price', 'harga_open', 'harga_close', 'annual_change']]
result.columns = ['year', 'us_lbs', 'us_kg', 'rp_kg', 'rp_top', 'rp_min', 'rp_open', 'rp_close', '%']

# Menampilkan hasil tanpa index
print(result.to_string(index=False))
```

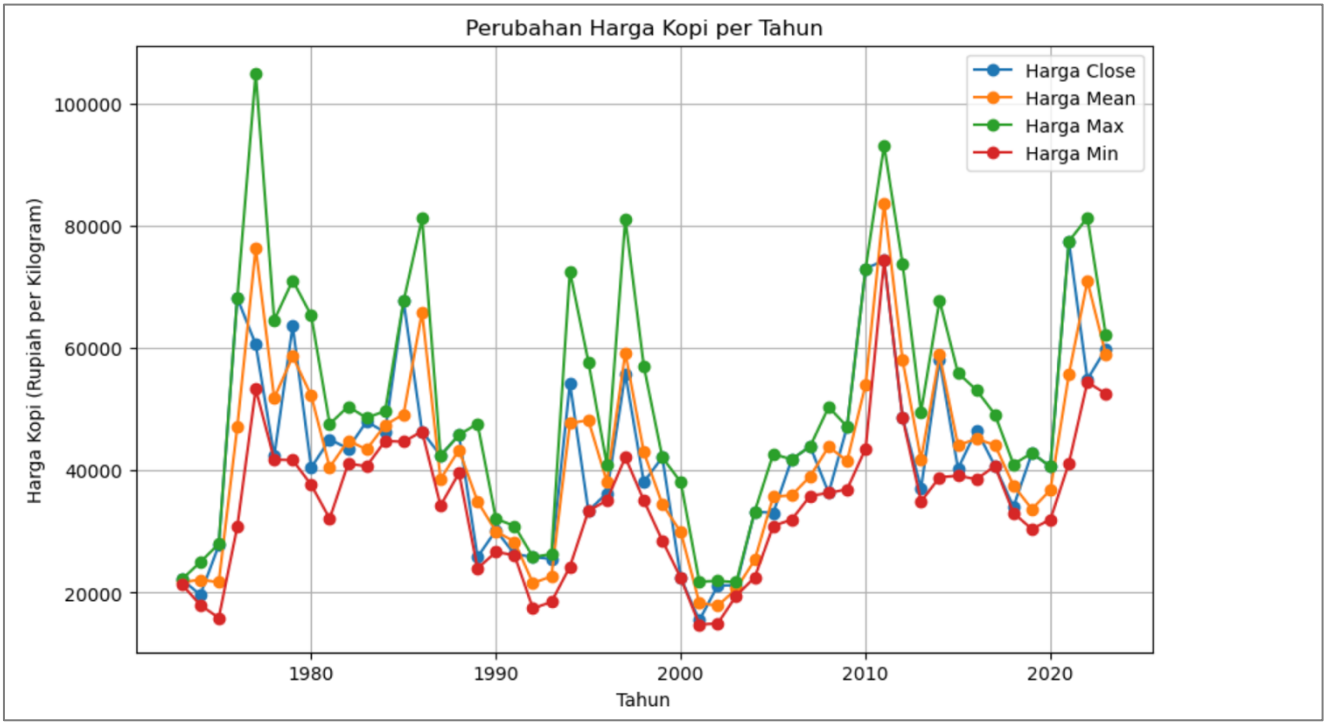
Adapun kurs yang kami gunakan adalah: 1US$ = 14.985,25 Rupiah

Output program yang dihasilkan:

year	us_lbs	us_kg	rp_kg	rp_top	rp_min	rp_open	rp_close	%
1973	0.65	1.44	21,707.84	22,169.41	21,372.20	21,637.54	22,169.41	0.00
1974	0.72	1.58	22,120.73	24,998.29	17,984.85	23,691.16	19,648.45	8.67
1975	0.56	1.24	21,720.81	27,974.55	15,881.57	18,533.07	27,746.71	-27.83
1976	0.93	2.05	47,073.62	68,174.65	30,783.03	30,783.03	68,174.65	39.79
1977	2.19	4.83	76,342.38	104,933.16	53,283.47	72,349.49	60,589.67	57.45
1978	1.95	4.31	51,831.90	64,559.71	41,761.72	64,559.71	42,329.39	-12.07
1979	1.30	2.87	58,621.37	71,055.17	41,672.13	42,979.70	63,609.38	-50.21
1980	1.72	3.80	52,251.24	65,327.34	37,639.42	56,920.17	40,437.70	24.49
1981	1.28	2.82	40,500.11	47,588.60	32,106.32	42,226.63	45,048.70	-34.80
1982	1.40	3.09	44,863.04	50,424.21	41,100.12	46,351.16	43,572.80	8.90
1983	1.28	2.82	43,419.18	48,575.43	40,671.70	42,275.49	47,994.20	-9.64
1984	1.42	3.13	47,406.92	49,693.74	44,804.88	46,920.32	46,203.82	9.90
1985	1.46	3.22	49,167.37	67,639.60	44,671.48	48,279.56	67,639.60	2.82
1986	2.41	5.32	65,888.90	81,191.15	46,373.54	79,660.15	46,373.54	39.39
1987	1.27	2.80	38,565.89	42,420.53	34,247.02	41,996.99	42,420.53	-89.68
1988	1.29	2.84	43,351.94	45,794.51	39,573.82	42,552.93	45,794.51	1.31
1989	1.44	3.18	34,941.42	47,637.81	23,975.32	47,637.81	25,821.04	10.67
1990	0.81	1.78	30,000.39	32,067.72	26,649.51	26,649.51	30,200.82	-78.76
1991	0.87	1.92	28,350.63	30,928.58	26,139.03	28,764.98	26,301.81	7.35
1992	0.77	1.71	21,570.12	25,839.97	17,418.42	25,567.37	25,839.97	-12.51
1993	0.67	1.47	22,666.88	26,294.73	18,489.78	21,986.46	25,507.22	-16.29
1994	0.73	1.62	47,697.98	72,550.26	24,241.38	24,241.38	54,234.50	9.30
1995	1.68	3.70	48,196.65	57,753.94	33,378.12	55,415.38	33,378.12	56.26
1996	1.06	2.34	38,148.65	40,927.61	35,085.33	35,085.33	36,280.64	-57.94
1997	1.28	2.81	59,248.50	81,098.04	42,144.18	42,144.18	55,808.74	16.75
1998	1.70	3.75	42,971.39	57,038.79	35,076.84	56,193.08	38,087.46	25.00
1999	1.12	2.48	34,473.83	42,133.09	28,525.88	37,164.09	42,133.09	-51.20
2000	1.15	2.54	29,895.89	38,059.98	22,395.36	38,059.98	22,395.36	2.35
2001	0.66	1.46	18,342.15	21,811.97	14,785.07	21,811.97	15,477.48	-74.49
2002	0.48	1.05	17,853.21	21,919.16	14,969.63	15,796.66	21,177.75	-38.08
2003	0.65	1.43	20,628.29	21,733.30	19,514.10	21,435.13	21,220.35	26.30
2004	0.73	1.60	25,358.08	33,279.28	22,500.57	23,952.32	33,279.28	10.51
2005	1.02	2.24	35,700.36	42,667.60	30,918.32	33,546.37	33,033.04	28.60
2006	1.19	2.63	35,919.37	41,797.08	31,945.62	39,428.76	41,797.08	14.92
2007	1.19	2.63	39,034.50	43,921.28	35,840.02	39,403.15	43,921.28	-0.06
2008	1.35	2.97	43,874.13	50,352.11	36,304.67	44,478.32	36,304.67	11.41
2009	1.17	2.57	41,520.28	47,141.10	36,905.44	38,517.61	47,141.10	-15.48

2010	1.40	3.09	54,067.52	72,980.32	43,571.20	46,311.64	72,980.32	16.83
2011	2.36	5.20	83,665.19	93,109.90	74,304.66	77,965.46	74,304.66	40.60
2012	2.23	4.92	58,022.11	73,765.68	48,588.14	73,765.68	48,588.14	-5.69
2013	1.50	3.31	41,752.95	49,573.16	34,912.02	49,573.16	37,066.58	-48.80
2014	1.18	2.59	58,839.16	67,755.03	38,865.82	38,865.82	58,021.84	-27.55
2015	1.69	3.73	44,113.38	55,921.91	39,123.61	55,921.91	40,287.04	30.50
2016	1.17	2.58	45,147.72	53,124.43	38,546.20	38,663.18	46,596.19	-44.64
2017	1.49	3.28	44,141.39	49,131.68	40,705.03	49,131.68	40,705.03	21.31
2018	1.24	2.74	37,522.38	41,002.57	33,019.00	41,002.57	34,046.97	-19.83
2019	1.03	2.28	33,672.12	42,826.98	30,397.35	34,149.40	42,826.98	-20.07
2020	1.14	2.50	36,790.49	40,609.95	31,988.42	37,504.71	40,609.95	8.95
2021	1.25	2.75	55,673.83	77,447.25	41,166.80	41,166.80	77,447.25	8.90
2022	2.36	5.21	70,949.12	81,315.72	54,414.95	78,113.20	54,865.53	47.30
2023	1.59	3.50	58,918.07	62,090.94	52,520.11	52,520.11	59,873.19	-48.73

```
plt.figure(figsize=(10, 6))
plt.plot(result['year'], result['rp_close'], marker='o', label='Harga Close')
plt.plot(result['year'], result['rp_kg'], marker='o', label='Harga Mean')
plt.plot(result['year'], result['rp_top'], marker='o', label='Harga Max')
plt.plot(result['year'], result['rp_min'], marker='o', label='Harga Min')
plt.xlabel('Tahun')
plt.ylabel('Harga Kopi (Rupiah per Kilogram)')
plt.title('Perubahan Harga Kopi per Tahun')
plt.legend()
plt.grid(True)
plt.show()
```



3. *Buat fungsi-fungsi untuk menghasilkan elemen elemen matriks A dan B dan buatlah kedua matriks tersebut, [Nilai: 20]*

Jawab:
Berikut adalah matriks yang dihasilkan dari python berdasarkan dari data A dan B yang merupakan data historical penjualan kopi di periode tertentu dan akan dianalisa untuk mendapatkan persamaan trendnya.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

#membaca data csv
data = pd.read_csv('matriks_kopi.csv')

#ambil kolom index sebagai data representasi data historical tahun
x = data['index']
#ambil kolom average_kg sebagai data rata-rata harga per kg setiap tahun
y = data['average_kg']

#function untuk membuat elemen matriks A
def elemA(p,q,x,y):
    sum = 0
    for xi in x:
        sum += xi**(p+q-2)
    return sum

#function untuk membuat elemen matriks B
def elemB(p,q,x,y):
    sum = 0
    N = len(x)
    for i in range(N):
        sum+=y[i]*(x[i]**(p+q-2))
    return sum

#function untuk menginisialisasi Matriks A
def inisialisasiA(M,x,y):
    mat = []
    for p in range(M+1):
        row=[]
        for q in range(M+1):
            apq = elemA(p+1,q+1,x,y)
            row.append(apq)
        mat.append(row)
    return mat

#function untuk menginisialisasi Matriks B
def inisialisasiB(M,x,y):
    mat = []
    for p in range(M+1):
        bpq = elemB(p+1, 1, x, y)
        mat.append(bpq)
    return mat
```

```
#function untuk print matriks agar lebih mudah dilihat
def printMatriks(mat):
    for row in mat:
        print(row)

#Menginisialisasi ordo matriks
M=2

#Membuat matriks A dan B
matA = inisialisasiA(M, x, y)
matB = inisialisasiB(M, x, y)

#Konversi matriks A dan B menjadi numpy array agar dapat diproses dengan numpy linalg
npA = np.array(matA)
npB = np.array(matB)

#menentukan nilai C (konstanta persamaan garis trend)
C = np.linalg.solve(npA, npB)

#membuat data trend
xx = [i*0.1 for i in range(1301)]
def ymodel(xx, C):
    yy=[]
    for x in xx:
        sum = 0
        for j in range(len(C)):
            sum+= C[j] * x**j
        yy.append(sum)
    return yy
yy = ymodel(xx,C)

#tampilkan matriks A, B, dan C
print('A = ')
printMatriks(matA)
print('B = ')
printMatriks(matB)
```

Berikut adalah output yang dihasilkan dari script diatas:

```
A =
[114, 6441, 487369]
[6441, 487369, 41486481]
[487369, 41486481, 3766875001]
B =
365.34397800600004
21450.327660113002
1720298.7876758326
```