PERTEMUAN KE-2:
PUSTAKA NUMPY DAN PANDAS

A. TUJUAN

Setelah menempuh praktikum ini diharapkan mahasiswa mampu

1. menggunakan paket Pandas Numpy
2. mampu menggunakan paket Pandas Pandas.

B. MATERI

Paket Numpy

Bahasa pemrograman Python menyediakan banyak pusata atau package untuk komputasi numerik, sains data dan *machine learning*. Salah satu package yang digunakan untuk membuat array adalah Numpy. NumPy bisa juga disebut dengan *Numerical Python*. Numpy adalah paket dasar Python untuk keperluan komputasi saintifik, misalnya perhitungan operasi array multi dimensi dan matriks beserta dengan kumpulan fungsi matematika tingkat tinggi. NumPy berada di dalam lisensi BSD yang memungkinkan programmer menggunakan ulang dengan beberapa batasan. Semua elemen dalam array NumPy harus memiliki tipe data yang sama, dan dengan demikian akan memiliki ukuran yang sama di memori. Array NumPy memfasilitasi operasi matematika tingkat lanjut dan jenis lain pada data dalam jumlah besar.

Saat ini semakin banyak paket ilmiah dan matematika berbasis Python yang menggunakan array NumPy. Berikut ini adalah beberapa fungsi yang disediakan oleh pustaka Numpy.

1. Pembuatan Array

Berikut ini adalah fungsi yang disediakan oleh Pustaka Numpy untuk pembuatan array.

- np.array([1, 2, 3]). np.array(list) digunakan untuk membuat array dari list pada Python.
- np.zeros((a,b)) digunakan untuk membuat array axb dengan nilai nol.
- np.ones((a,b)) digunakan untuk membuat array axb dengan nilai satu.
- np.full((a,b), c) digunakan untuk membuat array axb dengan semua elemen bernilai c.
- np.eye(a) digunakan untuk membuat matriks identitas axa.
- np.arange(0, 10, 2) digunakan untuk membuat array [0, 2, 4, 6, 8] (range dengan step). 2 adalah ukuran langkah dan 0 adalah nilai awal.
- np.linspace(0, 10, 5) menghasilkan 5 nilai yang dibagi rata antara 0 dan 10.
- np.random.rand(2,3) menghasilkan array 2x3 dengan nilai acak antara 0 dan 1.
- np.random.randint(1, 10, (3,3)) menghasilkan array 3x3 dengan angka acak dari 1-9.

2. Operasi Matematika

Berikut ini adalah operasi matematika yang disediakan oleh Pustaka Numpy.

- np.add(a, b), a + b digunakan untuk penjumlahan elemen.
- np.subtract(a, b), a - b digunakan untuk pengurangan.
- np.multiply(a, b), a * b digunakan untuk perkalian.
- np.divide(a, b), a / b digunakan untuk pembagian.
- np.exp(a) digunakan untuk eksponensial.
- np.sqrt(a) digunakan untuk akar kuadrat.
- np.log(a), np.log10(a) digunakan untuk logaritma alami dan basis 10.
- np.abs(a) digunakan untuk nilai absolut.
- np.round(a, 2) digunakan untuk pembulatan hingga 2 desimal.
- np.power(a, 3) digunakan untuk pangkat 3 dari setiap elemen.

3. Statistik & Agregasi

Berikut ini adalah operasi **Statistik & Agregasi** yang disediakan oleh Pustaka Numpy.

- np.sum(a) digunakan menjumlahkan elemen array.
- np.mean(a) digunakan untuk menghitung rata-rata.
- np.median(a) digunakan untuk menghitung Median.
- np.var(a) digunakan untuk menghitung Varians.
- np.std(a) digunakan untuk menghitung Standar deviasi.
- np.min(a), np.max(a) digunakan untuk menghitung Nilai minimum dan maksimum.
- np.percentile(a, 50) digunakan untuk menghitung Persentil ke-50 (median).
- np.corrcoef(a, b) digunakan untuk menghitung Korelasi antara array a dan b.

4. Operasi pada Array

Berikut ini adalah operasi **array** yang disediakan oleh Pustaka Numpy.

- a.shape digunakan untuk mendapatkan ukuran array.
- a.size digunakan untuk mendapatkan jumlah elemen dalam array.
- a.dtype digunakan untuk mendapatkan tipe data array.
- np.reshape(a, (2,3)) digunakan untuk mengubah bentuk array.
- np.transpose(a), a.T digunakan untuk mendapatkan transpos array.
- np.concatenate((a, b), axis=0) digunakan untuk menggabungkan array secara vertikal.
- np.concatenate((a, b), axis=1) digunakan untuk menggabungkan array secara horizontal.
- np.hstack((a, b)), np.vstack((a, b)) digunakan untuk Stack horizontal dan vertikal.

5. Operasi Matriks

Berikut ini adalah operasi matriks yang disediakan oleh Pustaka Numpy.

- np.dot(A, B) digunakan untuk menghitung perkalian matriks.
- np.matmul(A, B) digunakan untuk menghitung perkalian matriks alternatif.
- np.linalg.inv(A) digunakan untuk menghitung Invers matriks.
- np.linalg.det(A) digunakan untuk menghitung Determinan matriks.
- np.linalg.eig(A) digunakan untuk menghitung Eigenvalues dan eigenvectors.
- np.linalg.solve(A, b) digunakan untuk menghitung Menyelesaikan sistem persamaan linear.

Contoh 2.1.

Buatlah program untuk menggunakan fungsi statistik pada Numpy.

Jawaban

Kode Program 2.1. Program menggunakan fungsi statistik pada Numpy

No	Kode Program
1	`import numpy as np`
2	`data = np.array([10, 12, 23, 23, 16, 23, 21, 16, 19, 18])`
3	`mean = np.mean(data)  # Rata-rata`
4	`median = np.median(data)  # Median`
5	`variance = np.var(data)  # Varians`
6	`std_dev = np.std(data)  # Standar deviasi`
7	`minimum = np.min(data)  # Nilai minimum`
8	`maximum = np.max(data)  # Nilai maksimum`
9	`percentile_25 = np.percentile(data, 25)  # Persentil ke-25`
10	`percentile_75 = np.percentile(data, 75)  # Persentil ke-75`
11	`print(f"Data: {data}")`
12	`print(f"Mean (Rata-rata): {mean}")`
13	`print(f"Median: {median}")`
14	`print(f"Variance (Varians): {variance}")`
15	`print(f"Standard Deviation (Standar Deviasi): {std_dev}")`
16	`print(f"Min: {minimum}, Max: {maximum}")`
17	`print(f"25th Percentile: {percentile_25}")`
18	`print(f"75th Percentile: {percentile_75}")`

Paket Pandas

Pandas adalah alat analisis dan manipulasi data sumber terbuka yang cepat, kuat, fleksibel, dan mudah digunakan, dibangun di atas bahasa pemrograman Python. Pandas bertujuan untuk menjadi blok penyusun tingkat tinggi yang fundamental untuk melakukan analisis data di dunia nyata yang praktis dengan Python. Selain itu, ia memiliki tujuan yang lebih luas untuk menjadi alat analisis / manipulasi data *open-source* yang paling kuat dan fleksibel yang tersedia dalam bahasa apa pun.

Salah satu pemanfaatan dari paket pandas adalah membuka file CSV. File CSV (Comma Separated Values) merupakan file teks yang memiliki struktur untuk mengatur datanya. Seperti namanya, struktur yang digunakan CSV adalah koma. Koma digunakan untuk memisah data. Semua data dipisah dengan tanda koma. Koma ini disebut delimiters (pemisah). Selain menggunakan delimiter koma, ada juga CSV yang menggunakan tab (\t), titik dua (:), dan titik koma (;).

Pandas adalah pustaka Python yang sangat kuat untuk analisis dan manipulasi data. Berikut beberapa fungsi utama yang disediakan oleh Pandas:

1. Membuat DataFrame & Series

Berikut ini adalah fungsi yang disediakan oleh pandas dalam membuat DataFrame dan Series.

- pd.Series([1, 2, 3]) digunakan untuk membuat objek Series.
- pd.DataFrame({'A': [1, 2], 'B': [3, 4]}) digunakan untuk membuat DataFrame dari dictionary.
- pd.read_csv('data.csv') digunakan untuk membaca file CSV menjadi DataFrame.
- pd.read_excel('data.xlsx') digunakan untuk membaca file Excel.
- pd.read_json('data.json') digunakan untuk membaca file JSON.

2. Menampilkan & Mengeksplorasi Data

Berikut ini adalah fungsi yang disediakan oleh pandas untuk menampilkan dan mengeksplorasi data.

- df.head(5) digunakan untuk menampilkan 5 baris pertama.
- df.tail(5) digunakan untuk menampilkan 5 baris terakhir.
- df.shape digunakan untuk menampilkan (baris, kolom).
- df.info() digunakan untuk menampilkan informasi ringkasan DataFrame.
- df.describe() digunakan untuk menampilkan statistik dasar (mean, std, min, max, dll.).
- df.columns digunakan untuk menampilkan list nama kolom.
- df.dtypes digunakan untuk menampilkan tipe data setiap kolom.
- df.isnull().sum() digunakan untuk memeriksa jumlah missing values.

3. Seleksi Data

Berikut ini adalah fungsi yang disediakan oleh pandas untuk menyeleksi data.

- df['kolom'] digunakan untuk memilih satu kolom.
- df[['kolom1', 'kolom2']] digunakan untuk memilih beberapa kolom.
- df.loc[3] digunakan untuk memilih baris berdasarkan label index.
- df.iloc[3] digunakan untuk memilih baris berdasarkan posisi index.
- df.loc[df['kolom'] > 10] digunakan untuk memfilter data berdasarkan kondisi.

4. Manipulasi Data

Berikut ini adalah fungsi yang disediakan oleh pandas untuk memanipulasi data.

- df['kolom'].fillna(0) digunakan untuk mengisi missing values dengan 0.
- df.dropna() digunakan untuk menghapus baris dengan missing values.
- df.drop(columns=['kolom']) digunakan untuk menghapus kolom.
- df.rename(columns={'A': 'X'}) digunakan untuk mengganti nama kolom.
- df.sort_values(by='kolom', ascending=False) digunakan untuk mengurutkan data.

5. Agregasi & Statistik

Berikut ini adalah fungsi yang disediakan oleh pandas untuk agregasi dan statistik.

- df.mean() digunakan untuk menghitung rata-rata.
- df.median() digunakan untuk menghitung median.
- df.sum() digunakan untuk menjumlahkan data.
- df.min(), df.max() digunakan untuk menghitung milai minimum & maksimum.

- df.count() digunakan untuk menghitung jumlah elemen.
- df.groupby('kategori').mean() digunakan untuk mengelompokkan data lalu dihitung rata-ratanya.

6. Operasi String

Berikut ini adalah fungsi yang disediakan oleh pandas untuk operasi string.

- df['kolom'].str.lower() digunakan untuk mengubah teks ke huruf kecil.
- df['kolom'].str.contains('keyword') digunakan untuk memeriksa apakah ada teks tertentu.
- df['kolom'].str.replace('lama', 'baru') digunakan untuk mengganti teks dalam kolom.

7. Gabungan Data

Berikut ini adalah fungsi yang disediakan oleh pandas untuk menggabungkan data.

- pd.concat([df1, df2]) digunakan untuk menggabungkan dua DataFrame secara vertikal.
- df1.merge(df2, on='kolom') digunakan untuk merge dua DataFrame berdasarkan kolom.

8. Export Data

Berikut ini adalah fungsi yang disediakan oleh pandas untuk ekspor data.

- df.to_csv('hasil.csv', index=False) digunakan untuk menyimpan ke file CSV.
- df.to_excel('hasil.xlsx') digunakan untuk menyimpan ke file Excel.
- df.to_json('hasil.json') digunakan untuk menyimpan ke file JSON.

Contoh 2.2

Buatlah program untuk membuat DataFrame dari dictionary dan menyimpan hasilnya dalam file csv.

Jawaban

Kode Program 2.2. Program untuk membuat DataFrame dari dictionary dan menyimpan hasilnya dalam file csv.

No	Kode Program
1	`import pandas as pd`
2	`# Data siswa dalam bentuk dictionary`
3	`data_siswa = {`
4	`    'Nama': ['Ali', 'Budi', 'Cici', 'Dani', 'Eka'],`
5	`    'Usia': [17, 19, 18, 20, 21],`
6	`    'Kelas': ['10A', '11B', '10A', '12C', '12C'],`
7	`    'Nilai': [80, 65, 90, 72, 60]`
8	`}`
9	`# Menyimpan DataFrame ke dalam file CSV`
10	`df.to_csv('data_siswa.csv', index=False)`
11	`# Menampilkan 5 baris pertama`
12	`df.head()`

Contoh 2.3

Buatlah program untuk

a. menambahkan kolom Status yang berisi informasi apakah nilai siswa lebih besar atau sama dengan 75 (lulus) atau kurang dari 75 (gagal). Tampilkan DataFrame setelah kolom baru ditambahkan.
b. memfilter DataFrame untuk hanya menampilkan siswa yang berusia di atas 18 tahun dan memiliki nilai lebih dari 70.
c. menghitung rata-rata nilai seluruh siswa pada DataFrame.
d. menghitung rata-rata nilai siswa berdasarkan grup kelas.
e. menghitung rata-rata nilai siswa berdasarkan usia.

Jawaban

Kode Program 2.3. Program untuk contoh 2.3.

No	Kode Program
1	`import pandas as pd`
2	`# Data siswa dalam bentuk dictionary`
3	`data_siswa = {`
4	`    'Nama': ['Ali', 'Budi', 'Cici', 'Dani', 'Eka'],`

```
    'Usia': [17, 19, 18, 20, 21],
    'Kelas': ['10A', '11B', '10A', '12C', '12C'],
    'Nilai': [80, 65, 90, 72, 60]
}
# Membuat DataFrame dari dictionary
df = pd.DataFrame(data_siswa)
# a. Menambahkan kolom Status (Lulus / Gagal)
df['Status'] = df['Nilai'].apply(lambda x: 'Lulus' if x >= 75 else
'Gagal')
print("DataFrame setelah kolom Status ditambahkan:")
print(df)
# b. Memfilter DataFrame untuk siswa yang berusia di atas 18 tahun dan
nilai lebih dari 70
filtered_df = df[(df['Usia'] > 18) & (df['Nilai'] > 70)]
print("\nSiswa yang berusia di atas 18 tahun dan nilai lebih dari 70:")
print(filtered_df)
# c. Menghitung rata-rata nilai seluruh siswa
average_nilai = df['Nilai'].mean()
print("\nRata-rata nilai seluruh siswa:", average_nilai)
# d. Menghitung rata-rata nilai siswa berdasarkan grup kelas
average_by_class = df.groupby('Kelas')['Nilai'].mean()
print("\nRata-rata nilai siswa berdasarkan kelas:")
print(average_by_class)
# e. Menghitung rata-rata nilai siswa berdasarkan usia
average_by_age = df.groupby('Usia')['Nilai'].mean()
print("\nRata-rata nilai siswa berdasarkan usia:")
print(average_by_age)
```

Contoh 2.4

Diberikan DataFrame berisi data siswa dengan kolom Nama, Usia, dan Nilai.

a. Tambahkan kolom baru Status yang berisi informasi apakah nilai siswa lebih besar atau sama dengan 75 (lulus) atau kurang dari 75 (gagal), dengan menggunakan apply() dan lambda.
b. Tambahkan kolom baru Kategori_Usia yang mengelompokkan usia siswa menjadi 3 kategori:
    - "Remaja" jika usia kurang dari 18 tahun,
    - "Dewasa Muda" jika usia antara 18 dan 25 tahun,
    - "Dewasa" jika usia lebih dari 25 tahun.
c. Buat kolom baru Nilai_Berikut yang berisi nilai siswa ditambah 5 (dengan menggunakan fungsi apply() dan lambda).
d. Tentukan kolom Usia_Kategoris yang mengkategorikan usia menjadi:
    - "Tua" jika usia lebih dari atau sama dengan 30,
    - "Muda" jika usia lebih kecil dari 30.
e. Buat kolom Diskon yang memberikan diskon 10% jika nilai siswa lebih dari 80 dan 5% jika nilai siswa kurang dari atau sama dengan 80.

```
data_siswa = {
  'Nama': ['Ali', 'Budi', 'Cici', 'Dani', 'Eka'],
  'Usia': [17, 22, 18, 24, 29],
  'Nilai': [80, 65, 90, 72, 60]
}
```

Jawaban

Kode Program 2.4. Program untuk contoh 2.4.

No	Kode Program
1	`import pandas as pd`
2	`data_siswa = {`
3	`    'Nama': ['Ali', 'Budi', 'Cici', 'Dani', 'Eka'],`
4	`    'Usia': [17, 22, 18, 24, 29],`
5	`    'Nilai': [80, 65, 90, 72, 60]`

```
}
df = pd.DataFrame(data_siswa)
# 1. Menambahkan kolom Status (Lulus / Gagal) berdasarkan nilai
df['Status'] = df['Nilai'].apply(lambda x: 'Lulus' if x >= 75 else
'Gagal')
print("DataFrame setelah menambahkan kolom Status:")
print(df)
# 2. Menambahkan kolom Kategori_Usia berdasarkan usia
df['Kategori_Usia'] = df['Usia'].apply(lambda x: 'Remaja' if x < 18 else
('Dewasa Muda' if 18 <= x <= 25 else 'Dewasa'))
print("\nDataFrame setelah menambahkan kolom Kategori_Usia:")
print(df)
# 3. Menambahkan kolom Nilai_Berikut yang berisi nilai siswa + 5
df['Nilai_Berikut'] = df['Nilai'].apply(lambda x: x + 5)
print("\nDataFrame setelah menambahkan kolom Nilai_Berikut:")
print(df)
# 4. Menambahkan kolom Usia_Kategoris (Tua/Muda) berdasarkan usia
df['Usia_Kategoris'] = df['Usia'].apply(lambda x: 'Tua' if x >= 30 else
'Muda')
print("\nDataFrame setelah menambahkan kolom Usia_Kategoris:")
print(df)
# 5. Menambahkan kolom Diskon berdasarkan nilai
df['Diskon'] = df['Nilai'].apply(lambda x: 0.1 if x > 80 else 0.05)
print("\nDataFrame setelah menambahkan kolom Diskon:")
print(df)
```

C. TUGAS/LATIHAN SOAL

1. Buatlah program untuk operasi perkalian matriks dengan file paket Numpy.
2. Buatlah DataFrame menggunakan dictionary dengan data berikut:

   data = {
   'Nama': ['Ari', 'Budi', 'Cici', 'Deni', 'Eka'],
   'Usia': [23, 25, 22, 21, 24],
   'Nilai': [80, 75, 90, 65, 88],
   'Kelas': ['A', 'B', 'A', 'C', 'B']
   }
3. Tampilkan DataFrame yang sudah dibuat pada soal no 2.
4. Lakukan filtering DataFrame untuk menampilkan siswa yang berusia lebih dari 22 tahun dan memiliki nilai lebih dari 75.
5. Hitung median nilai per kelas.
6. Buat DataFrame lain dengan data berikut:

   data2 = {
   'Nama': ['Fikri', 'Gina', 'Hani'],
   'Usia': [26, 27, 28],
   'Nilai': [85, 78, 92],
   'Kelas': ['C', 'A', 'B']
   }

   Gabungkan kedua DataFrame tersebut berdasarkan kolom Nama.