DATA ANALYTICS

BAB 3 : DATA MANIPULATIONS

Praktikum 1

1.1 Tujuan

Mahasiswa mengenal materi Manipulasi Data menggunakan bahasa pemrograman Python antara lain mahasiswa akan mempelajari tentang penggunaan library Python seperti NumPy dan Pandas.

1.2 Ulasan Materi

1. **Apa itu NumPy?**

   NumPy, kependekan dari **Numerical Python**, adalah pustaka open-source fundamental yang digunakan untuk bekerja dengan array di Python. Menyediakan cara yang kuat dan efisien untuk menyimpan, memanipulasi, dan menganalisis data. Kecepatannya unggul dalam melakukan operasi numerik pada array data.

   Fitur utama NumPy:

   ndarray: Struktur data inti di NumPy adalah ndarray, objek array multidimensi yang dirancang untuk kecepatan dan efisiensi.

   Operasi Matematika: NumPy menawarkan berbagai fungsi untuk melakukan operasi matematika pada array..

2. **Mengapa Menggunakan NumPy dalam Analisis Data?**
   - ndarray NumPy yang dioptimalkan mengungguli daftar Python secara signifikan.
   - komprehensif yang dirancang khusus untuk komputasi numerik dan aljabar linier.
   - NumPy digunakan sebagai fondasi pustaka lain seperti Pandas dan Scikit-learn agar performa dan optimasi komputasi menjadi lebih baik.

3. **Memulai dengan NumPy**

   3.1 Mengimpor NumPy

```
import numpy as np
```

   3.2 Memeriksa Versi NumPy

```
print(np.__version__)
```

4. **Membuat Array menggunakan NumPy**

   4.1 Membuatan Array dari List

```
arr = np.array([1, 2, 3, 4, 5])
```

```
print(arr)
```

4.2 Membuatan Array Multidimensi

```
arr = np.array([[1, 2, 3], [4, 5, 6]])
print(arr)
```

4.3 Memeriksa Dimensi Array

```
a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]], [[1, 2, 3], [4, 5, 6]]])
print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```

4.4 Membuat Array dengan Dimensi Tertentu

```
arr = np.array([1, 2, 3, 4], ndmin=5)
print(arr)
print('jumlah dimensi:', arr.ndim)
```

5. Pengindeksan Array NumPy

5.1 Mengakses Elemen Array

```
import numpy as np
arr = np.array([1, 2, 3, 4])
print(arr[0])  # Cetak elemen pertama
print(arr[1])  # Cetak elemen kedua
print(arr[2] + arr[3])  # Akses elemen ketiga dan keempat, lalu
jumlahkan
```

5.2 Mengakses Array 2-D

```
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('Elemen ke-2 pada baris ke-1: ', arr[0, 1])
print('Elemen ke-5 pada baris ke-2: ', arr[1, 4])
```

5.3 Mengakses Array 3-D

```
arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11,
12]]])
print(arr[0, 1, 2])  # Cetak elemen ketiga dari array kedua di
array pertama
```

5.4 Pengindeksan Negatif

```
arr = np.array([[1,2,3,4,5], [6,7,8,9,10]])
print('Elemen terakhir dari dimensi ke-2: ', arr[1, -1])
```

6. Pemotongan Array NumPy (NumPy Array Slicing)

Pemotongan dalam Python berarti mengambil elemen dari satu indeks yang diberikan ke indeks lain yang diberikan.

Sintaks: [start:end:step]

start: Indeks awal (termasuk). Default-nya 0.

end: Indeks akhir (tidak termasuk).

step: Langkah pengambilan elemen. Default-nya 1.

Contoh:

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5, 6, 7])
# Potong elemen dari indeks 1 hingga 5 (tidak termasuk indeks 5)
print(arr[1:5])

# Potong elemen dari indeks ke-4 hingga akhir array
print(arr[4:])

# Potong elemen dari awal hingga indeks 4 (tidak termasuk)
print(arr[:4])
```

Pemotongan Negatif:

Gunakan operator minus (-) untuk merujuk ke indeks dari akhir:

```
print(arr[-3:-1])  # Potong dari indeks ke-3 dari akhir hingga
indeks ke-1 dari akhir
```

Pemotongan Array 2-D:

```
import numpy as np
arr = np.array([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10]])

# Dari elemen kedua, potong elemen dari indeks 1 ke indeks 4
(tidak termasuk)
print(arr[1, 1:4])

# Dari kedua elemen, kembalikan indeks 2
print(arr[0:2, 2])

# Dari kedua elemen, potong indeks 1 ke indeks 4 (tidak
termasuk), ini akan mengembalikan array 2-D
print(arr[0:2, 1:4])
```

7. **Tipe Data NumPy (NumPy Data Types)**
   Tipe Data NumPy:

   NumPy memiliki beberapa tipe data tambahan dibandingkan Python bawaan, direpresentasikan dengan satu karakter:

i - integer

b - boolean

u - unsigned integer

f - float

c - complex float

m - timedelta

M - datetime

O - object

S - string

U - unicode string

V - fixed chunk of memory untuk tipe data lain (void)

Memeriksa Tipe Data Array:

```
arr = np.array([1, 2, 3, 4])
print(arr.dtype)  # Cetak tipe data dari objek array
```

Membuat Array dengan Tipe Data Tertentu:

```
arr = np.array([1, 2, 3, 4], dtype='S')  # S untuk string
print(arr)
print(arr.dtype)
```

Konversi Tipe Data pada Array yang Ada:

```
arr = np.array([1.1, 2.1, 3.1])
newarr = arr.astype('i')  # Ubah tipe data dari float ke integer
menggunakan 'i'

print(newarr)
print(newarr.dtype)
```

8. **Perbedaan Antara copy() dan view():**

copy(): Memiliki data sendiri, perubahan pada salinan tidak memengaruhi array asli.

view(): Tidak memiliki data sendiri, perubahan pada tampilan memengaruhi array asli.

Membuat Salinan:

```
arr = np.array([1, 2, 3, 4, 5])  # Buat array
x = arr.copy()  # Buat salinan dari array
arr[0] = 42  # Ubah elemen pertama pada array asli

print(arr)  # Cetak array asli
```

```
print(x)  # Cetak salinan
```

Membuat Tampilan:

```
arr = np.array([1, 2, 3, 4, 5])  # Buat array
x = arr.view()  # Buat tampilan dari array
arr[0] = 42  # Ubah elemen pertama pada array asli

print(arr)  # Cetak array asli
print(x)  # Cetak tampilan
```

Tampilan AKAN terpengaruh oleh perubahan yang dilakukan pada array asli.

Melakukan Perubahan pada TAMPILAN (Make Changes in the VIEW):

Contoh:

```
arr = np.array([1, 2, 3, 4, 5])  # Buat array
x = arr.view()  # Buat tampilan dari array
x[0] = 31  # Ubah elemen pertama pada tampilan

print(arr)  # Cetak array asli
print(x)  # Cetak tampilan
```

Array asli AKAN terpengaruh oleh perubahan yang dilakukan pada tampilan.

- **Memeriksa Kepemilikan Data Array (Check if Array Owns its Data)**
  - Salinan memiliki data sendiri.
  - Tampilan tidak memiliki data sendiri.
  - arr.base mengembalikan None untuk salinan, dan merujuk ke objek asli untuk tampilan.

Contoh:

```
arr = np.array([1, 2, 3, 4, 5])
x = arr.copy()
y = arr.view()

print(x.base)  # Salinan (None)
print(y.base)  # Tampilan (arr)
```

9. **Bentuk Array NumPy (NumPy Array Shape)**
   - Bentuk array: Jumlah elemen di setiap dimensi.
   - arr.shape untuk mendapatkan bentuk.
   - Tupel bentuk menunjukkan jumlah elemen di setiap dimensi.

Contoh:

```
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
print(arr.shape)  # (2, 4)
```

10. Perubahan Bentuk Array NumPy (NumPy Array Reshaping)

- Ubah bentuk array. Dengan perubahan bentuk, kita dapat menambah atau menghapus dimensi atau mengubah jumlah elemen di setiap dimensi.
- arr.reshape(new_shape) untuk mengubah bentuk.
- Dimensi yang tidak diketahui (-1).
- arr.reshape(-1) untuk memipihkan array.

Contoh:

```
arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
newarr = arr.reshape(4, 3)  # Ubah bentuk menjadi 2D (4 baris, 3 elemen)
print(newarr)

newarr = arr.reshape(2, 3, 2)  # Ubah bentuk menjadi 3D (2 baris, 3 array, 2 elemen)
print(newarr)
```

11. Iterasi Array NumPy (NumPy Array Iterating)

- Loop for dasar untuk iterasi.
- nditer() untuk iterasi lanjutan.

Contoh:

```
arr = np.array([1, 2, 3])
for x in arr:
  print(x)

arr = np.array([[1, 2, 3], [4, 5, 6]])
for x in arr:
  print(x)

for x in arr:
  for y in x:
    print(y)

arr = np.array([[[1, 2, 3], [4, 5, 6]], [[7, 8, 9], [10, 11, 12]]])
for x in arr:
  print(x)

for x in arr:
  for y in x:
```

```
    for z in y:
      print(z)

arr = np.array([[[1, 2], [3, 4]], [[5, 6], [7, 8]]])
for x in np.nditer(arr):
  print(x)

arr = np.array([1, 2, 3])
for x in np.nditer(arr, flags=['buffered'], op_dtypes=['S']):
  print(x)

arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
for x in np.nditer(arr[:, ::2]):
  print(x)

arr = np.array([1, 2, 3])
for idx, x in np.ndenumerate(arr):
  print(idx, x)

arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8]])
for idx, x in np.ndenumerate(arr):
  print(idx, x)
```

12. Penggabungan Array NumPy (NumPy Joining Array)

- concatenate() untuk menggabungkan berdasarkan sumbu.
- stack() untuk menggabungkan di sepanjang sumbu baru.

Contoh:

```
# Buat array 1 dimensi
arr1 = np.array([1, 2, 3])
arr2 = np.array([4, 5, 6])

arr1 = np.array([[1, 2], [3, 4]])  # Buat array 2 dimensi
arr2 = np.array([[5, 6], [7, 8]])  # Buat array 2 dimensi

arr = np.concatenate((arr1, arr2))  # Gabungkan arr1 dan dan arr2
berdasarkan baris (axis=1)

arr_2 = np.concatenate((arr1, arr2), axis=1)  # Gabungkan arr1
dan arr2 berdasarkan baris (axis=1)

print(arr)
print(arr_2)
```

13. Pemisahan Array Dasar (Basic Array Splitting)

Membagi array menjadi beberapa bagian yang lebih kecil.

Gunakan np.array_split().

Contoh:

```
arr = np.array([1, 2, 3, 4, 5, 6])  # Buat array
newarr = np.array_split(arr, 3)  # Pisahkan menjadi 3 bagian
print(newarr)  # Cetak array yang dipisahkan
```

Pemisahan Menjadi Beberapa Array (Splitting into Multiple Arrays)

Mengakses elemen individual dari hasil pemisahan.

Contoh:

```
arr = np.array([1, 2, 3, 4, 5, 6])  # Buat array
newarr = np.array_split(arr, 3)  # Pisahkan menjadi 3 bagian
print(newarr[0])  # Akses array pertama dari hasil pemisahan
print(newarr[1])  # Akses array kedua dari hasil pemisahan
print(newarr[2])  # Akses array ketiga dari hasil pemisahan
```

Pemisahan Array 2 Dimensi (Splitting 2D Arrays)

Gunakan sintaks yang sama dengan pemisahan array 1 dimensi.

Gunakan np.array_split() dengan menentukan jumlah pemisahan dan sumbu yang ingin dipisahkan.

Contoh:

```
arr = np.array([[1, 2], [3, 4], [5, 6], [7, 8], [9, 10], [11,
12]])  # Buat array 2D
newarr = np.array_split(arr, 3)  # Pisahkan menjadi 3 array 2D
print(newarr)  # Cetak array 2D yang dipisahkan
```

Pemisahan berdasarkan Baris:

```
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12],
[13, 14, 15], [16, 17, 18]])  # Buat array 2D
newarr = np.array_split(arr, 3, axis=1)  # Pisahkan menjadi 3
array 2D berdasarkan baris
print(newarr)  # Cetak array 2D yang dipisahkan berdasarkan baris
```

Alternatif: Gunakan hsplit() untuk pemisahan baris.

```
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12],
[13, 14, 15], [16, 17, 18]])  # Buat array 2D
```

```
newarr = np.hsplit(arr, 3)  # Pisahkan arr menjadi 3 array 2D
berdasarkan baris
print(newarr)  # Cetak array 2D yang dipisahkan berdasarkan baris
```

Pemisahan dengan hsplit() (Splitting with hsplit())

Memecah array 2D menjadi beberapa array 2D lainnya.

Kebalikan dari hstack() (penggabungan horizontal).

Contoh:

```
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 12],
[13, 14, 15], [16, 17, 18]])  # Buat array 2D
newarr = np.hsplit(arr, 3)  # Pisahkan arr menjadi 3 array 2D
berdasarkan baris
print(newarr)# Cetak array 2D yang dipisahkan berdasarkan baris
```

14. Pencarian Elemen Dasar (Basic Element Search)

Menemukan nilai dan indeksnya dalam array.

Gunakan np.where().

Contoh:

```
arr = np.array([1, 2, 3, 4, 5, 4, 4])  # Buat array
x = np.where(arr == 4)  # Cari indeks di mana nilai = 4
print(x)  # Cetak indeks yang ditemukan
```

Pencarian pada Array Terurut (Searching on Sorted Arrays)

Gunakan np.searchsorted() untuk pencarian binary pada array terurut.

Mengembalikan indeks untuk mempertahankan urutan.

Contoh:

```
arr = np.array([6, 7, 8, 9])  # Buat array terurut
x = np.searchsorted(arr, 7)  # Cari indeks di mana nilai 7
seharusnya dimasukkan
print(x)  # Cetak indeks
```

Pencarian Sisi Kanan:

```
arr = np.array([6, 7, 8, 9])  # Buat array terurut
x = np.searchsorted(arr, 7, side='right')  # Cari indeks dari
kanan
```

```
print(x)  # Cetak indeks
```

Pencarian Beberapa Nilai:

```
arr = np.array([1, 3, 5, 7])  # Buat array terurut
x = np.searchsorted(arr, [2, 4, 6])  # Cari indeks untuk beberapa
nilai
print(x)  # Cetak indeks
```

Penjelasan:

np.searchsorted(arr, [2, 4, 6]): Mencari indeks di mana nilai 2, 4, dan 6 seharusnya dimasukkan.

Nilai yang dikembalikan adalah array [1 2 3], menunjukkan indeks untuk nilai-nilai tersebut.

15. Pengurutan Array (Array Sorting)

Pengurutan Dasar

Mengurutkan elemen dalam urutan tertentu (numerik atau alfabetik).

Gunakan np.sort().

Contoh:

```
arr = np.array([3, 2, 0, 1])  # Buat array
print(np.sort(arr))  # Urutkan dan cetak array
```

Catatan:

sort() menghasilkan salinan baru yang diurutkan, array asli tidak berubah.

Mendukung pengurutan string dan tipe data lain.

Pengurutan Array 2D

Contoh:

```
arr = np.array([[3, 2, 4], [5, 0, 1]])  # Buat array 2D
print(np.sort(arr))  # Urutkan dan cetak array 2D
```

16. Pemfilteran Array (Array Filtering)

Pemfilteran Dasar

Memilih elemen berdasarkan kondisi tertentu.

Gunakan daftar indeks Boolean (True/False).

Contoh:

```
arr = np.array([41, 42, 43, 44])  # Buat array
x = [True, False, True, False]  # Daftar filter
newarr = arr[x]  # Filter array
print(newarr)  # Cetak array hasil filter
```

Pembuatan Daftar Filter

Buat daftar filter berdasarkan kondisi.

Contoh:

```
arr = np.array([41, 42, 43, 44])  # Buat array
filter_arr = []  # Daftar filter kosong

for element in arr:
  if element > 42:
    filter_arr.append(True)
  else:
    filter_arr.append(False)

newarr = arr[filter_arr]
print(filter_arr)  # Cetak daftar filter
print(newarr)  # Cetak array hasil filter
```

Pemfilteran Langsung dari Array

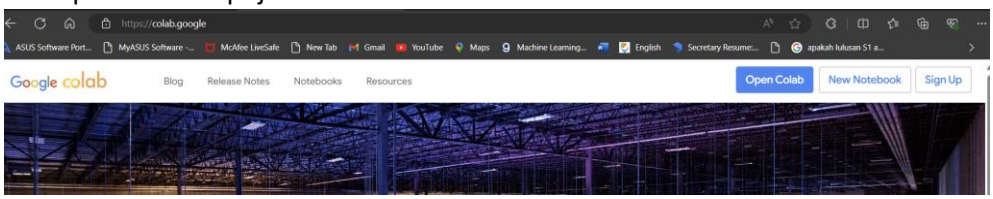
Gunakan array sebagai kondisi dalam ekspresi boolean.

Contoh:

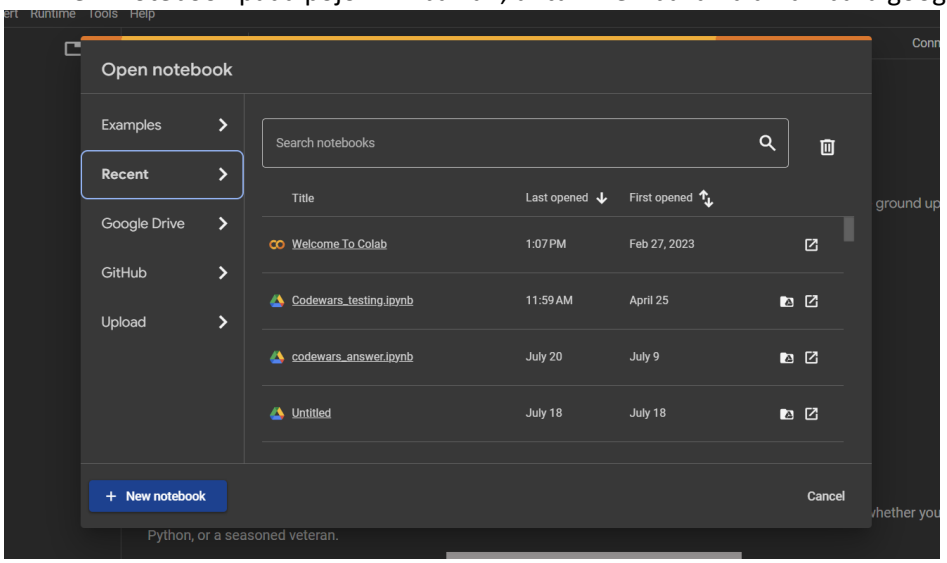
```
arr = np.array([41, 42, 43, 44])  # Buat array
filter_arr = arr > 42  # Pemfilteran elemen > 42
newarr = arr[filter_arr]
print(filter_arr)  # Cetak daftar filter
print(newarr)  # Cetak array hasil filter
```

1.4 Langkah Persiapan

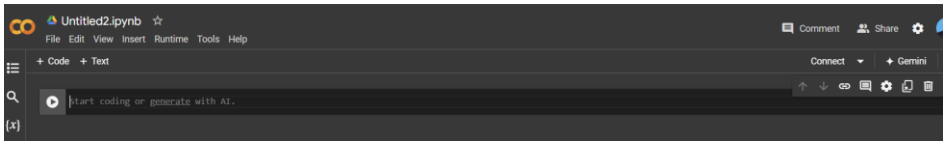
1. Membuka Google Colab
   a. Buka Google Colaboratory dengan link berikut https://colab.research.google.com/ .
   b. Klik Open Colab di pojok kanan atas



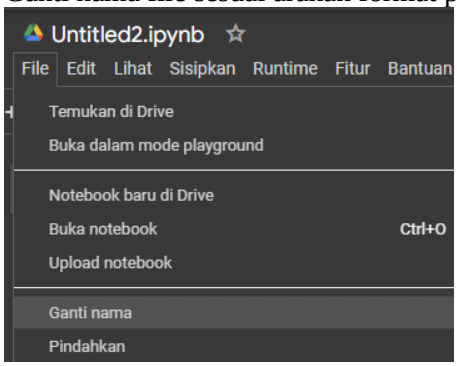
   c. Anda bisa login menggunakan akun Google.
   d. Klik New Notebook pada pojok kiri bawah, untuk membuka halaman baru google colab.



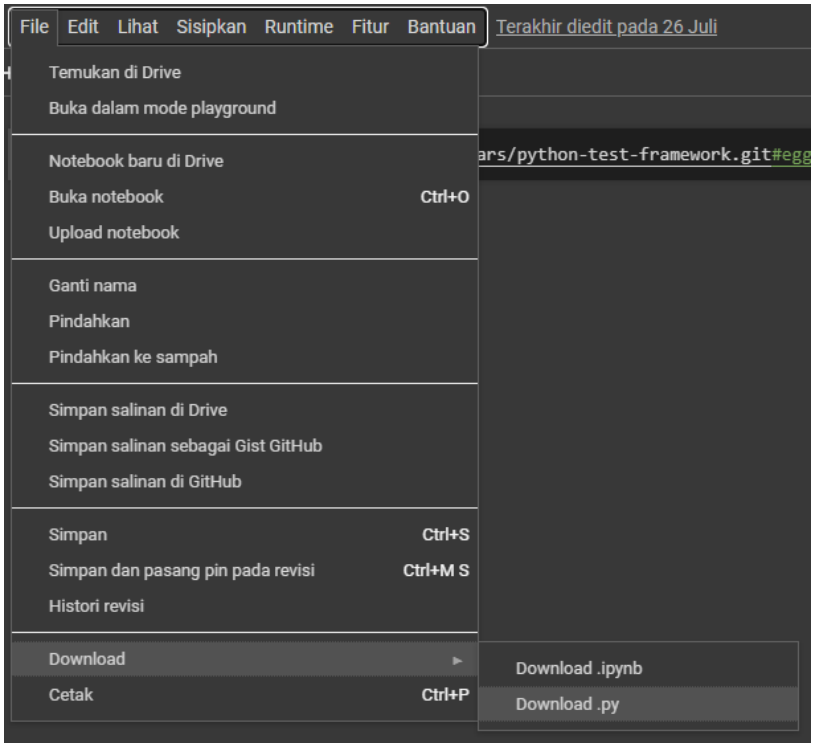
   e. Tampilan Google Colab.



   f. Ganti nama file sesuai arahan format pada praktikum



   g. Setelah selesai mengerjakan praktikum, download file dengan format (**.py**)


File
Edit
Lihat
Sisipkan
Runtime
Fitur
Bantuan
Terakhir diedit pada 26 Juli
Temukan di Drive
Buka dalam mode playground
Notebook baru di Drive
Buka notebook Ctrl+O
Upload notebook
Ganti nama
Pindahkan
Pindahkan ke sampah
Simpan salinan di Drive
Simpan salinan sebagai Gist GitHub
Simpan salinan di GitHub
Simpan Ctrl+S
Simpan dan pasang pin pada revisi Ctrl+M S
Histori revisi
Download
Cetak Ctrl+P
Download .ipynb
Download .py
ars/python-test-framework.git#egg

1.5 Contoh Studi Kasus

Memotong Array NumPy 3 Dimensi

Buatlah kode Python yang menggunakan slicing untuk memotong array NumPy 3 dimensi untuk mengambil kotak ke-1, baris ke-2, dan kolom ke-1 hingga ke-2:

Langkah-langkah:

1. **Import pustaka NumPy:** Import pustaka NumPy dengan alias np.

```
import numpy as np
```

2. **Buat array NumPy 3 dimensi:**

    - Define '**array_3d'** variable
    - Use np.array(), membuat array 3D ukuran 2x3x3

```
# Contoh array NumPy 3 dimensi
array_3d = np.array([
  [ [1, 2, 3], [4, 5, 6], [7, 8, 9] ],
  [ [10, 11, 12], [13, 14, 15], [16, 17, 18] ]
])
```

3. Gunakan slicing untuk memotong array. Sintaksnya adalah nama_array[kotak, baris, kolom]. Dalam kasus ini, ambil data pada kotak ke-1, baris ke-2, dan kolom ke-1 hingga ke-2. **array_3d[0, 1, 1:3]**

4. Simpan hasil potongan array dalam variabel **array_potongan**.

```
# Memotong array
array_potongan = array_3d  [0, 1, 1:3]  # Kotak ke-1, baris ke-2, kolom ke-1 hingga ke-2
```

5. Cetak hasil potongan array ke konsol untuk melihat isinya.

```
print("Array yang dipotong:")
print(array_potongan)
```

Output:

```
Array yang dipotong:
[5 6]
```

Tampilan Keseluruhan Kode

```
import numpy as np
# Contoh array NumPy 3 dimensi
array_3d = np.array([
  [ [1, 2, 3], [4, 5, 6], [7, 8, 9] ],
  [ [10, 11, 12], [13, 14, 15], [16, 17, 18] ]
])
array_potongan = array_3d [0, 1, 1:3]
print("Array yang dipotong:")
print(array_potongan)
```

1.6 Praktikum

Mengambil Data Spesifik Penjualan Produk

Skenario: Sebuah toko online memiliki data penjualan produk yang disimpan dalam array NumPy 3 dimensi. Array ini berisi data penjualan produk untuk tiga kategori (elektronik, pakaian, dan mainan) selama tiga bulan (Januari, Februari, dan Maret). Setiap elemen dalam array adalah jumlah produk yang terjual untuk kategori dan bulan tertentu.

Objektif: Menerapkan slicing untuk mengambil data yang spesifik pada Numpy Array 3D.

Langkah – Langkah:

1. Import pustaka NumPy: Import pustaka NumPy dengan alias np.
2. Buat array NumPy 3 dimensi:
    - Buat variable bernama '**data_penjualan**'
    - Buat array NumPy 3D ukuran 3x3x3 menggunakan **np.array()** untuk membuatnya.
    - Gunakan List berikut

```
data_penjualan = np.array([
    # Kategori 1 (Elektronik)
    [[10, 20, 30],  # Bulan 1
     [40, 50, 60],  # Bulan 2
     [70, 80, 90]], # Bulan 3

    # Kategori 2 (Pakaian)
    [[100, 110, 120], # Bulan 1
     [130, 140, 150], # Bulan 2
     [160, 170, 180]], # Bulan 3

    # Kategori 3 (Mainan)
    [[190, 200, 210], # Bulan 1
     [220, 230, 240], # Bulan 2
     [250, 260, 270]]  # Bulan 3
])
```

3. Gunakan Slicing untuk memotong array. Sintaksnya adalah nama_array[kotak, baris, kolom]. Ambil data pada Potong kotak ke-1, baris ke-2, dan kolom ke-1 hingga ke-2
4. Simpan hasil potongan array dalam variabel **data_spesifik**.
5. Cetak hasil potongan array ke konsol untuk melihat isinya. Gunakan **print()**
   Output:
   Array yang dipotong:
   ```
   [50 60]
   ```

6. Submit file dengan nama **answer_bab3_percobaan1.py** pastikan file disimpan dalam format Python file (**.py**)