**Dasar Logika: Operator Perbandingan (Comparison Operators)**

Sebelum kita bisa memfilter data, kita harus bisa "bertanya" pada data tersebut. Di sinilah operator perbandingan berperan. Operator ini digunakan untuk membandingkan dua nilai dan hasilnya selalu berupa nilai **Boolean**, yaitu True atau False.

| **Operator** | **Arti** | **Contoh** | **Hasil** |
| --- | --- | --- | --- |
| == | Sama dengan | 5 == 5 | True |
| != | Tidak sama dengan | 5 != 4 | True |
| < | Kurang dari | 4 < 5 | True |
| > | Lebih dari | 5 > 4 | True |
| <= | Kurang dari atau sama | 5 <= 5 | True |
| >= | Lebih dari atau sama | 5 >= 5 | True |

**Operator Perbandingan Utama**

**Perbandingan pada NumPy Array**

Seperti yang ditunjukkan di materi Anda, kehebatan operator ini muncul saat diterapkan pada array NumPy. Operasi perbandingan akan dilakukan pada **setiap elemen** array, menghasilkan sebuah array baru yang berisi nilai True dan False.

**Contoh:**

import numpy as np

bmi = np.array([21.8, 20.9, 24.7, 21.4])

# "Tanyakan" pada array: Elemen mana saja yang nilainya lebih dari 23?

apakah\_lebih\_besar = bmi > 23

print(apakah\_lebih\_besar)

**Outputnya:**

[False False True False]

Array boolean inilah yang nantinya akan kita gunakan sebagai "saringan" atau "filter" untuk memilih data di Pandas.

**Latihan Singkat**

Coba prediksi hasil dari kode berikut tanpa menjalankannya:

1. "jakarta" == "Jakarta"
2. tinggi = np.array([170, 165, 180, 155])  
   tinggi >= 170
3. 5 != 5.0

**Dasar Logika: Operator Boolean (and, or, not)**

Operator Boolean digunakan untuk menggabungkan atau membalikkan nilai True/False. Ini memungkinkan kita membuat "pertanyaan" yang lebih kompleks pada data kita.

**Operator Boolean Utama**

1. **and (dan):** Menghasilkan True hanya jika **kedua** kondisi bernilai True.
   * True and True → True
   * True and False → False
   * **Contoh:** x > 5 and x < 15
2. **or (atau):** Menghasilkan True jika **salah satu** (atau kedua) kondisi bernilai True.
   * True or False → True
   * False or False → False
   * **Contoh:** y < 7 or y > 13
3. **not (bukan/tidak):** Membalikkan nilai boolean.
   * not True → False
   * not False → True

**Menggunakan Operator Boolean pada NumPy**

Menggunakan and dan or secara langsung pada array NumPy akan menyebabkan ValueError. Sebagai solusinya, NumPy menyediakan fungsi khusus:

* np.logical\_and(kondisi1, kondisi2)
* np.logical\_or(kondisi1, kondisi2)
* np.logical\_not(kondisi)

**Contoh:**

import numpy as np

bmi = np.array([21.8, 20.9, 21.7, 24.7, 21.4])

# Mencari BMI yang lebih besar dari 21 DAN lebih kecil dari 22

filter\_logis = np.logical\_and(bmi > 21, bmi < 22)

print(filter\_logis)

# Output: [ True False True False True]

# Menggunakan filter tersebut untuk mengambil nilainya

print(bmi[filter\_logis])

# Output: [21.8 21.7 21.4]

**Latihan Singkat**

Diberikan sebuah array nilai siswa: nilai = np.array([85, 92, 78, 65, 95])

Tuliskan satu baris kode menggunakan fungsi logika NumPy untuk membuat filter yang memilih semua nilai yang:

* Lebih besar dari 80 **DAN** lebih kecil dari 95.

**Dasar Logika: Pernyataan Kondisional (if, elif, else)**

Pernyataan kondisional memungkinkan alur program Anda bercabang. Artinya, program bisa memutuskan blok kode mana yang akan dijalankan berdasarkan apakah suatu kondisi bernilai True atau False.

**1. if (Jika)**

Ini adalah bentuk paling dasar. Kode di dalam blok if hanya akan dijalankan **jika** kondisinya True.

**Struktur:**

if kondisi:

# Lakukan sesuatu di sini (harus ada indentasi)

**Contoh:**

z = 4

if z % 2 == 0: # Kondisi ini True

print("z adalah bilangan genap")

**2. else (Jika tidak)**

else memberikan blok kode alternatif yang akan dijalankan jika kondisi if ternyata False.

**Struktur:**

if kondisi:

# Jalankan ini jika True

else:

# Jalankan ini jika False

**Contoh:**

z = 5

if z % 2 == 0: # Kondisi ini False

print("z adalah bilangan genap")

else:

print("z adalah bilangan ganjil")

**3. elif (Jika tidak, maka jika...)**

elif adalah singkatan dari "else if". Ini memungkinkan Anda untuk memeriksa **beberapa kondisi secara berurutan**. Python akan berhenti di kondisi elif pertama yang True.

**Struktur:**

if kondisi\_1:

# Jalankan ini jika kondisi\_1 True

elif kondisi\_2:

# Jalankan ini jika kondisi\_1 False, TAPI kondisi\_2 True

else:

# Jalankan ini jika semua kondisi di atas False

**Contoh:**

z = 3

if z % 2 == 0: # False

print("z habis dibagi 2")

elif z % 3 == 0: # True, maka blok ini dijalankan dan berhenti

print("z habis dibagi 3")

else:

print("z tidak habis dibagi 2 ataupun 3")

**Latihan Singkat**

Diberikan variabel area = 12. Buatlah sebuah struktur if-elif-else untuk menentukan kategori ruangan berdasarkan luasnya:

* Jika area lebih besar dari 15, cetak "ruangan besar".
* Jika tidak, periksa apakah area lebih besar dari 10. Jika ya, cetak "ruangan sedang".
* Jika tidak memenuhi keduanya, cetak "ruangan kecil".

**Menerapkan Logika: Memfilter DataFrame Pandas**

Inilah saatnya kita menggunakan "saringan" boolean yang sudah kita buat untuk memilih baris-baris tertentu dari data kita. Prosesnya, seperti yang dijelaskan di materi Anda, terdiri dari 3 langkah sederhana.

**Proses Filtering 3 Langkah**

Mari kita gunakan DataFrame brics sebagai contoh untuk memilih negara dengan luas area lebih dari 8 juta km².

# DataFrame kita

import pandas as pd

data\_brics = {

"country": ["Brazil", "Russia", "India", "China", "South Africa"],

"capital": ["Brasilia", "Moscow", "New Delhi", "Beijing", "Pretoria"],

"area": [8.516, 17.100, 3.286, 9.597, 1.221]

}

brics = pd.DataFrame(data\_brics)

**Langkah 1: Pilih Kolom**

Pertama, kita ambil kolom yang ingin kita filter. Hasilnya adalah sebuah Pandas Series.

brics['area']

**Langkah 2: Buat Kondisi (Boolean Mask)**

Kemudian, kita terapkan operator perbandingan pada kolom tersebut. Ini akan menghasilkan Series baru yang berisi True dan False.

is\_huge = brics['area'] > 8

# is\_huge akan berisi: [True, True, False, True, False]

**Langkah 3: Gunakan Masker untuk Memfilter**

Terakhir, kita masukkan *boolean mask* (is\_huge) ke dalam kurung siku [] dari DataFrame asli. Pandas akan menampilkan semua baris di mana nilainya True.

brics[is\_huge]

Anda juga bisa menggabungkan semuanya dalam satu baris:

brics[brics['area'] > 8]

**Memfilter dengan Beberapa Kondisi**

Sama seperti di NumPy, kita bisa menggunakan np.logical\_and() dan np.logical\_or() untuk menggabungkan beberapa kondisi.

**Contoh:** Memilih negara dengan area lebih dari 8 Juta km² **DAN** kurang dari 10 Juta km².

import numpy as np

filter\_kondisi = np.logical\_and(brics['area'] > 8, brics['area'] < 10)

brics[filter\_kondisi]

**Latihan Singkat**

Gunakan df\_mobil dari proyek kita sebelumnya.

data\_mobil = {

'merek': ['Toyota', 'Honda', 'Suzuki', 'Daihatsu', 'Mitsubishi'],

'tahun': [2019, 2020, 2018, 2019, 2021],

'harga\_juta': [250, 280, 180, 170, 300]

}

df\_mobil = pd.DataFrame(data\_mobil)

1. Buatlah filter untuk memilih mobil yang harganya **di bawah 200 juta**.
2. Buatlah filter untuk memilih mobil yang tahunnya **lebih dari 2019 DAN** harganya **di atas 250 juta**.

import numpy as np

filter\_mobil = (

(df['jarak\_tempuh'] < 30000) &

(df['transmisi'] == 'Automatic') &

(df['harga'] > 280\_000\_000)

)

mobil\_cocok = df[filter\_mobil]

print(mobil\_cocok)

filter\_mobil = np.logical\_and(

df['jarak\_tempuh'] < 30000,

np.logical\_and(

df['transmisi'] == 'Automatic',

df['harga'] > 280\_000\_000

)

)

mobil\_cocok = df[filter\_mobil]