Catatan Perulangan: `while` Loop

Konsep Dasar

Bayangkan `while` loop sebagai pernyataan `if` yang terus berulang. Jika `if` hanya memeriksa kondisi sekali, `while` akan terus **memeriksa kondisi** dan **menjalankan kodenya berulang-ulang** selama kondisi tersebut masih bernilai `True`.

Struktur Dasar

```
while kondisi:
    # Blok kode yang akan diulang (harus ada indentasi)
    # Penting: Harus ada sesuatu di sini yang bisa membuat kondisi menjadi False!
```

Contoh Sederhana

Contoh berikut mengurangi nilai `error` sampai di bawah 1.

```
# Inisialisasi variabel
error = 50.0

# Loop akan berjalan selama 'error' masih lebih besar dari 1
while error > 1:
    # Kurangi nilai error dengan membaginya 4
    error = error / 4
    # Cetak nilai error di setiap iterasi
    print(error)
```

Alur Kerja:

1. **Iterasi 1:** `50 > 1` (True) -> `error` menjadi `12.5`.
2. **Iterasi 2:** `12.5 > 1` (True) -> `error` menjadi `3.125`.
3. **Iterasi 3:** `3.125 > 1` (True) -> `error` menjadi `0.78125`.
4. **Iterasi 4:** `0.78125 > 1` (False) -> Loop berhenti.

Peringatan: Infinite Loop

Jika Anda lupa memasukkan kode yang bisa membuat kondisi `while` menjadi `False`, program Anda akan berjalan selamanya! Ini disebut **infinite loop**.

```
x = 1
while x > 0: # Kondisi ini akan selalu True
    print("Bantuan, saya terjebak!")
    # Tidak ada kode yang mengubah nilai x
```

Catatan Perulangan: `for` Loop

Konsep Dasar

Jika `while` loop berjalan berdasarkan sebuah *kondisi*, `for` loop akan **mengambil setiap item satu per satu dari sebuah urutan** (seperti `list` atau `string`) dan menjalankan blok kode untuk setiap item tersebut. Loop ini akan berhenti secara otomatis setelah semua item selesai diproses.

Struktur Dasar

```
for variabel_sementara in urutan_data:
    # Blok kode yang akan dijalankan untuk setiap item
    # 'variabel_sementara' akan berisi item yang sedang diproses
```

Contoh Menggunakan List

Melakukan iterasi pada setiap elemen dalam sebuah list.

```
# Data tinggi badan keluarga
fam = [1.73, 1.68, 1.71, 1.89]

# Loop akan mengambil setiap nilai tinggi badan satu per satu
for height in fam:
    print(height)
```

Mengakses Indeks dengan `enumerate()`

Terkadang, selain nilainya, kita juga butuh nomor indeksnya. Untuk itu, kita bisa membungkus urutan data kita dengan fungsi `enumerate()`.

```
for index, height in enumerate(fam):
    print("Indeks " + str(index) + ": " + str(height))
```

Contoh Output:

```
Indeks 0: 1.73
Indeks 1: 1.68
...
```

Loop pada String

`for` loop juga bisa digunakan untuk mengambil setiap karakter dari sebuah string.

```
for karakter in "mobil":
    print(karakter)
```

Catatan Looping pada Struktur Data

Looping pada Dictionaries

Untuk mendapatkan *key* dan *value* dari dictionary, kita harus menggunakan metode `.items()`.

```
world = { "afghanistan": 30.55, "albania": 2.77 }

for key, value in world.items():
    print(key + " -- " + str(value))

#alternatif modern
For key, value in word.items():
       print(F”{key} – {value}”)
```

Looping pada NumPy Array

Array 1D

Looping pada array 1D sama seperti pada list biasa.

```
import numpy as np
bmi = np.array([21.8, 20.9, 24.7])

for nilai_bmi in bmi:
    print(nilai_bmi)
```

Array 2D

Untuk mengakses setiap elemen satu per satu, gunakan `np.nditer()`.

```
np_2d = np.array([[1.73, 1.68], [65.4, 59.2]])

for elemen in np.nditer(np_2d):
    print(elemen)
```

Latihan Singkat

Diberikan dictionary berikut:
`data_mobil = {'merek': 'Toyota', 'tahun': 2019, 'harga': 250}`

Buatlah sebuah `for` loop menggunakan `.items()` untuk mencetak setiap key dan value dengan format `key: value`.

Looping pada Pandas DataFrame

Untuk melakukan perulangan pada setiap **baris** data dalam sebuah DataFrame, kita menggunakan metode `.iterrows()`. Metode ini sangat berguna karena di setiap perulangan, ia akan memberikan dua hal:

- **Label Indeks Baris:** Nama atau nomor indeks dari baris tersebut.
- **Data Baris:** Seluruh data pada baris tersebut dalam bentuk sebuah Pandas Series.

Struktur Dasar `.iterrows()`

Struktur dasar untuk menggunakan `.iterrows()` adalah sebagai berikut:

```
for label, baris in nama_dataframe.iterrows():
    # 'label' akan berisi indeks baris (misal: 'BR')
    # 'baris' akan berisi data pada baris tersebut
    print(f"Indeks: {label}")
    print(baris['nama_kolom'])
```

Contoh Penggunaan

Misalnya, kita ingin mencetak ibu kota dari setiap negara dalam DataFrame `brics`.

```
for label, baris in brics.iterrows():
    # 'baris' adalah Series, jadi kita bisa akses kolomnya
    print(label + ": " + baris['capital'])
```

Output yang Dihasilkan:

```
BR: Brasilia
RU: Moscow
IN: New Delhi
...
```

Latihan Singkat

Gunakan DataFrame `df_mobil` yang sudah kita buat sebelumnya. Buatlah sebuah `for` loop menggunakan `.iterrows()` untuk mencetak merek dan tahun setiap mobil dengan format:
`Merek Mobil: [NAMA MEREK], Tahun: [TAHUN]`.

Import pandas as pd

gunakan data mobil dari latihan sebelumnya nilai kali ini berupa list

Data_mobil = {

“merek”:[“Toyota”, “Honda”, “Suzuki”, “Daihatsu”, “Mitsubishi”],

“tahun”:[2019, 2020, 2018, 2019, 2021]

```
df_mobil = pd.DataFrame(data_mobil

# label adalah index baris adalah series atau kolo

for label, baris in df_mobil.iterrows():

        print(f”Merek Mobil: {baris[“merk”]}, Tahun:{baris[“tahun”]}”)
```

```
# itertuples baru ini

for row in df.itertuples():

    print(f"Merek: {row.merk}, Tahun: {row.tahun}")
```