Dasar-Dasar Python: Dictionaries (Kamus)

Seperti yang ditunjukkan di materi Anda, menggunakan *list* untuk menyimpan data yang saling berhubungan (seperti negara dan populasinya) bisa jadi kurang praktis. Anda harus tahu indeks numerik dari setiap data.

```
# Cara dengan list (kurang praktis)
countries = ["afghanistan", "albania",
"algeria"] pop = [30.55, 2.77, 39.21]

# Untuk tahu populasi Albania, kita harus cari
indeksnya dulu ind_alb = countries.index("albania") #
hasilnya 1 print(pop[ind_alb]) # baru kita bisa
dapatkan 2.77
```

Di sinilah **Dictionary** menjadi solusi yang lebih elegan dan intuitif.

Apa itu Dictionary?

Dictionary adalah tipe data yang menyimpan koleksi pasangan **kunci (key)** dan **nilai (value)**. Setiap kunci bersifat unik dan digunakan untuk mengakses nilainya.

- **Kunci (Key):** Pengenal unik untuk sebuah data (contoh: "albania"). Harus berupa tipe data yang tidak bisa diubah (*immutable*) seperti string atau angka.
- **Nilai (Value):** Data yang ingin kita simpan (contoh: 2.77). Bisa berupa tipe data apa pun.

Membuat Dictionary Pertama Anda

Kita bisa membuat dictionary dengan kurung kurawal `{}`.

Contoh:

```
# Membuat dictionary
world = {"afghanistan": 30.55, "albania": 2.77,
"algeria": 39.21}   # Mengakses nilai dengan kunci
(lebih mudah!) print(world["albania"])
```

Output:

```
2.77
```

Jauh lebih mudah dibaca, bukan? Kita tidak perlu lagi pusing dengan urutan atau indeks angka.

Latihan Singkat: Membuat Dictionary Ibu Kota

---

1. Buatlah sebuah dictionary bernama `ibu_kota` yang berisi data berikut:
    - Kunci "Indonesia", Nilai "Jakarta"
    - Kunci "Malaysia", Nilai "Kuala Lumpur"
    - Kunci "Thailand", Nilai "Bangkok"
2. Setelah itu, coba tampilkan ibu kota dari "Malaysia" menggunakan `print()`.

Lanjutan Dictionaries: Menambah & Mengubah Data

Selain mengambil data, kita juga bisa dengan mudah mengubah isi dari sebuah dictionary.

Menambah & Memperbarui Data

Caranya sangat mudah dan sintaksnya sama untuk kedua operasi ini. Anda cukup menunjuk sebuah kunci (key) dan memberinya nilai (value).

- Jika **kunci belum ada**, Python akan **menambahkan** pasangan kuncinilai yang baru.
- Jika **kunci sudah ada**, Python akan **memperbarui** nilainya dengan yang baru.

Contoh:

```
world = {"afghanistan": 30.55, "albania": 2.77,
"algeria": 39.21}
# 1. Menambah data baru
world["sealand"] = 0.000027
print(world)
# Output akan menyertakan 'sealand'

# 2. Memperbarui data yang sudah ada
world["albania"] = 2.81 # Nilai albania akan berubah dari
2.77 print(world)
```

Menghapus Data

Untuk menghapus sebuah item dari dictionary, kita menggunakan fungsi `del()`.

```
# Menghapus 'sealand' dari
dictionary
del(world["sealand"])
print(world)
# Output tidak akan lagi berisi 'sealand'
```

Mengecek Keberadaan Kunci

Terkadang, kita perlu tahu apakah sebuah kunci ada di dalam dictionary sebelum melakukan sesuatu. Kita bisa menggunakan kata kunci `in`.

```
print("sealand" in world) # Output: False (karena sudah
dihapus) print("albania" in world) # Output: True
```

Latihan Singkat: Memperbarui Data Ibu Kota

Sekarang, mari kita praktikkan pada dictionary `ibu_kota` yang sudah Anda buat.

```
ibu_kota = {"Indonesia": "Jakarta", "Malaysia": "Kuala
Lumpur",  "Thailand": "Bangkok"}
```

1. **Tambahkan** data baru: Ibu kota dari "Filipina" adalah "Manila".
2. Ternyata ada kesalahan data, **perbarui** ibu kota "Indonesia" menjadi "Nusantara".
3. **Hapus** data "Thailand" dari dictionary.
4. Tampilkan dictionary `ibu_kota` yang terakhir untuk melihat semua perubahannya.

Pengantar Pandas: Dari Dictionary ke DataFrame

Seperti yang dijelaskan di materi Anda, data di dunia nyata sering kali berbentuk tabel (tabular), dengan baris dan kolom. Pustaka (library) **Pandas** dirancang khusus untuk bekerja dengan data semacam ini secara efisien.

Struktur Data Utama: DataFrame

Struktur data inti di Pandas adalah **DataFrame**. Anggap saja ini adalah sebuah tabel super canggih, mirip seperti lembar kerja di Excel.

- Setiap **kolom** adalah sebuah variabel.
- Setiap **baris** adalah sebuah observasi atau catatan.

Keunggulan DataFrame adalah setiap kolom bisa memiliki tipe datanya sendiri (teks, angka, dll.), tidak seperti array NumPy yang harus seragam.

Cara Paling Umum: Membuat DataFrame dari Dictionary

Inilah mengapa kita belajar Dictionaries terlebih dahulu! Cara paling umum untuk membuat DataFrame dari awal adalah dengan menggunakan sebuah *dictionary* Python.

- **Kunci (Keys)** dari dictionary akan menjadi **nama kolom**.
- **Nilai (Values)**, yang biasanya berupa *list*, akan menjadi **data di dalam kolom** tersebut.

Contoh (dari materi Anda):

```
import pandas as pd

# 1. Buat dictionary terlebih dahulu
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],
    "population": [200.40, 143.50, 1252.00, 1357.00, 52.98]
}
```

```
# 2. Gunakan pd.DataFrame() untuk mengubahnya
brics = pd.DataFrame(data_brics)

# 3. Tampilkan hasilnya
print(brics)
```

Angka di paling kiri (0-4) adalah **indeks** baris, yang dibuat otomatis oleh Pandas.

Latihan Singkat: Membuat DataFrame Data Cuaca

1. Buatlah sebuah **dictionary** yang berisi data cuaca dari proyek Matplotlib kita sebelumnya.
    - `'bulan': ['Jan', 'Feb', 'Mar', ..., 'Des']`
    - `'suhu': [28, 29, 31, ..., 29]`
    - `'curah_hujan': [250, 310, 280, ..., 260]`
2. Dari dictionary tersebut, buatlah sebuah DataFrame Pandas bernama `df_cuaca`.
3. Tampilkan `df_cuaca` menggunakan `print()`.

Pandas: Mengakses Data dari DataFrame

Setelah data Anda tersimpan rapi dalam sebuah DataFrame, langkah berikutnya adalah mengambil bagian-bagian data yang Anda perlukan.

1. Menggunakan Kurung Siku `[]`

Ini adalah cara paling dasar dan memiliki dua fungsi utama:

a. Memilih Kolom (Column Selection)

Ini adalah penggunaan yang paling umum. Anda cukup menuliskan nama kolom di dalam kurung siku.

```
import pandas as pd

# Kita gunakan lagi DataFrame brics
```

```
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)

# Memilih satu kolom (hasilnya adalah Series)
print(brics["country"])

# Memilih beberapa kolom (gunakan list di dalamnya, hasilnya
DataFrame)
print(brics[["country", "capital"]])
```

b. Memilih Baris dengan Slicing (Row Slicing)

Anda juga bisa menggunakan `[]` untuk memilih baris, tetapi **hanya dengan teknik *slicing*** (seperti `[1:4]`).

```
# Memilih baris dari indeks 1 hingga sebelum 4
print(brics[1:4])
```

2. `.loc`: Memilih Berdasarkan Label

Karena menggunakan `[]` agak terbatas, Pandas menyediakan `.loc` untuk pemilihan berbasis **label (nama)**. Ini sangat kuat karena Anda tidak perlu tahu posisi angkanya.

```
# Mengatur indeks baris (row index)
brics.index = ["BR", "RU", "IN", "CH", "SA"]

# Memilih satu baris berdasarkan labelnya
print(brics.loc["RU"])

# Memilih beberapa baris
print(brics.loc[["RU", "IN", "CH"]])

# Memilih baris DAN kolom tertentu
print(brics.loc[["RU", "IN", "CH"], ["country", "capital"]])
```

3. `.iloc`: Memilih Berdasarkan Posisi Integer

`.iloc` bekerja mirip seperti `.loc`, tetapi menggunakan **posisi numerik (integer)**, persis seperti array NumPy. Ini berguna saat Anda tidak peduli dengan nama labelnya.

```
# Memilih baris di posisi ke-1 (Rusia)
print(brics.iloc[1])

# Memilih baris di posisi 1, 2, dan 3
print(brics.iloc[[1, 2, 3]])

# Memilih baris 1-3 dan kolom 0-1
print(brics.iloc[[1, 2, 3], [0, 1]])
```

Latihan Singkat: Memilih Data Cuaca

Gunakan `df_cuaca` yang sudah Anda buat sebelumnya.

1. Pilih dan tampilkan hanya kolom `'suhu'`.
2. Pilih dan tampilkan baris untuk 3 bulan pertama (indeks 0 sampai 2) menggunakan *slicing*.
3. Gunakan `.loc` untuk memilih data pada baris dengan indeks `0` (Januari).
4. Gunakan `.iloc` untuk memilih data pada baris di posisi ke-11 (Desember).