# **Python Dasar: 7 Aturan Sintaks**

**Bahasa pemrograman Python adalah bahasa yang dieksekusi oleh sebuah interpreter. Interpreter tersebut bertugas untuk memparsing sintaks python, dan kemudian mengkonversinya menjadi sebuah instruksi mesin satu persatu (maksudnya adalah satu baris-satu baris)** [**[1]**](https://www.bbc.co.uk/bitesize/guides/zgmpr82/revision/2)**.**

**Terdapat beberapa aturan penulisan sintaks pada bahasa pemrograman Python. Aturan-aturan ini harus ditaati oleh programmer agar interpreter bisa memparsing dan menjalankan aplikasi dengan baik. Jika tidak, maka aplikasi akan memproduksi sebuah error.**

## **Apa saja aturan-aturan penulisan sintaks python?**

**Aturan penulisan python terbagi menjadi beberapa sub bahasan; mulai dari struktur baris kode, statemen, komentar, penugasan, indentasi, dan lain sebagainya.**

**Secara umum, sintaks penulisan python bersifat:**

* **Case Sensitive**
* **Tidak Menggunakan Titik Koma**
* **Indentasi Sebagai Pembentuk Struktur**
* **Tidak ketat terhadap tipe data**
* **Human friendly**

**Kita akan bahas satu persatu.**

**NB: Beberapa contoh dari tutorial ini diambil dari materi-materi yang akan datang. Sehingga sangat wajar jika kita masih belum memahami beberapa istilah atau beberapa kode program yang dijadikan contoh.**

## **Case Sensitive**

**Bahasa pemrograman python bersifat *case sensitive*. Ia akan membedakan antara huruf kecil dan huruf besar walaupun sebuah kata itu terlihat sama.**

**Contoh:**

* **ibu\_kota = 'Jakarta'**
* **print(iBu\_kota)**

**Jika dijalankan, kita akan mendapatkan error:**

* **Exception has occurred: NameError**
* **name 'iBu\_kota' is not defined**
* **File "case-sensitive.py", line 3, in <module>**
* **print(iBu\_kota)**

**Kenapa?**

**Karena variabel yang kita definisikan adalah ibu\_kota, dengan huruf kecil semuanya. Sedangkan variabel yang berusaha kita panggil adalah iBu\_kota yang mana huruf B nya adalah huruf kapital, dan interpreter python menganggap keduanya berbeda.**

## **Tidak Menggunakan Titik Koma**

**Sebelum membahas lebih lanjut tentang titik koma, ada baiknya kita mengetahui terlebih dahulu pengertian statement pada bahasa pemrograman (lebih khusus pada python).**

### **Apa itu statement?**

**Statemen adalah sebuah pernyataan atau instruksi yang akan dieksekusi oleh mesin** [**[2]**](https://www.greenteapress.com/thinkpython/thinkCSpy/html/chap02.html#:~:text=A%20statement%20is%20an%20instruction%20that%20the%20Python%20interpreter%20can%20execute.)**. Interpreter python akan bertugas untuk menginterpretasikan setiap statemen menjadi perintah mesin yang sesuai.**

### **Penulisan Statement**

**Di dalam python, penulisan antar statemen tidak diakhiri dengan titik koma –*berbeda dengan bahasa pemrograman lain pada umumnya di mana setiap statement akan dibedakan berdasarkan adanya karakter titik koma (;)*.**

**Lalu, bagaimana cara interpreter python membedakan antar satu statemen dengan statemen lainnya?**

**Dengan karakter ganti baris (\n).**

**Setiap pergantian baris, interpreter akan menganggap bahwa sebuah statemen telah selesai.**

**Perhatikan contoh berikut:**

* **a = 5**
* **b = 8**
* **c = a + b**
* **print(c)**

**Pada program di atas, terdapat 4 buah statemen. Dan setiap statemen dipisahkan oleh sebuah karakter \n atau karakter yang menandakan bahwa suatu baris telah selesai.**

### **Kita tetap bisa menggunakan titik koma**

**Pada kasus-kasus tertentu, kita tetap bisa menggunakan titik koma.**

**Kapan?**

**Ketika terdapat lebih dari 1 statemen dalam satu baris.**

**Perhatikan contoh berikut:**

* **a = 5; b = 8; c = a + b;**
* **print(c)**

**Pada contoh di atas, terdapat 4 buah statement:**

* **3 di antaranya ditulis dalam satu baris sekaligus.**
* **dan sisanya ditulis pada baris yang berbeda.**

## **Satu Statemen Multi Baris**

**Kebalikannya…**

**Kita bisa memecah satu statemen menjadi lebih dari satu baris.**

**Dalam kasus-kasus tertentu, satu statemen bisa sangat panjang. Sehingga kode programnya melebihi ukuran layar.**

**Contoh:**

* **kondisi = 10 < 5 and 10 > 9 or 11 == 6 + 5 and 0 == 100 \* 5 / (25 - 15)**

**Hal ini tentu saja membuat kode program menjadi lebih susah dibaca, didebug, dan dipahami.**

**Kita inginnya kode program terlihat lebih ringkas dan mudah dibaca.**

**Sedangkan jika kita buat baris baru, statemen di atas akan terputus, yang kemudian akan dianggap error oleh interpreter python.**

**Solusinya?**

**Kita bisa memecah satu statemen panjang menjadi multiple baris dengan tanda *backslash* (\).**

**Seperti contoh berikut:**

* **kondisi = 10 < 5 \**
* **and 10 > 9 \**
* **or 11 == 6 + 5 \**
* **and 0 == 100 \* 5 / (25 - 15)**

**Meskipun kode program di atas memiliki total 4 baris, tapi interpreter tetap menganggapnya sebagai satu statemen utuh.**

## **Indentasi Sebagai Pembentuk Struktur**

**Pada bahasa pemrograman lain, umumnya indentasi adalah sesuatu yang tidak penting. Bahkan cenderung diabaikan oleh mesin.**

**Indentasi hanya digunakan untuk mempermudah manusia dalam membaca kode program.**

**Tapi tidak dengan python.**

**Dalam python, indentasi adalah hal yang super-super penting, karena ia bertugas untuk mendefinisikan struktur blok kode program.**

**Sehingga, melakukan kesalahan indentasi juga bisa berujung pada sebuah error (yang mungkin akan sulit dipecahkan bagi yang belum terbiasa). Atau dalam istilah lain: kita menggunakan indentasi untuk mengelompokkan blok kode program di dalam python.**

### **Apa itu Indentasi?**

**Indentasi adalah penulisan paragraf yang agak menjorok masuk ke dalam** [**[3]**](https://id.wikipedia.org/wiki/Wikipedia:Tutorial_(Halaman_pembicaraan)#:~:text=Indentasi%20adalah%20penulisan%20paragraf%20yang,maka%20paragraf%20akan%20semakin%20menjorok.)**. Biasanya jika kita membaca majalah atau koran, kita akan dapati indentasi pada kalimat awal setiap paragrafnya.**

### **Kesalahan-Kesalahan Penulisan Indentasi pada Python**

**Untuk memahami betapa pentingnya indentasi pada python, mari kita perhatikan contoh-contoh berikut tentang kesalahan indentasi yang umum terjadi pada bahasa pemrogrman python.**

**Perhatikan contoh berikut:**

* **print('Selamat')**
* **print('Pagi')**
* **print('Dunia')**

**Kode program di atas, secara sekilas tidak ada masalah. Tapi, ketika kita eksekusi, ternyata interpreter python memberitahukan kita bahwa terdapat error yang disebabkan oleh indentasi yang tidak pada tempatnya:**

* **IndentationError: unexpected indent**

**Yang benar harusnya tiga perintah print() berada pada satu indentasi, karena memang ketiganya berada dalam satu blok yang sama.**

* **print('Selamat')**
* **print('Pagi')**
* **print('Dunia')**

**Contoh kode program yang benar ✅:**

* **a = 10**
* **if a > 5:**
* **print('nilai a lebih dari 5')**

**Contoh yang salah ❌:**

* **a = 10**
* **if a > 5:**
* **print('nilai a lebih dari 5')**

**Pesan error:**

* **IndentationError: expected an indented block**

**Contoh yang salah tapi samar (kalau di-copy paste kemungkinan akan tetap working) ❌:**

* **a = 10**
* **if a > 5:**
* **print('nilai a lebih dari 5')**
* **print('nilai a lebih dari 5')**

**Pesan error:**

* **TabError: inconsistent use of tabs and spaces in indentation**

**Kenapa yang terakhir error?**

**Karena ke-tidak-konsisten-an penggunaan tab. Yang pertama menggunakan 4 spasi untuk tab, sedangkan yang kedua menggunakan 1 tab asli (bukan spasi).**

**Meski secara visual terlihat sama, tapi tidak bagi interpreter python.**

**Oiya, jika kita menggunakan teks editor atau IDE yang canggih, error seperti ini tidak akan terjadi karena IDE atau Teks Editor yang kita gunakan akan otomatis mengkonversi semua tab menjadi seragam, entah menggunakan spasi atau menggunakan tab itu sendiri.**

**Error seperti ini biasanya akan terjadi jika:**

* **Kita melakukan *copy-paste* kode program dari internet**
* **Dan kita tidak menggunakan teks editor atau IDE yang kekinian**

## **Tidak Ketat Terhadap Tipe Data**

**Sifat yang berikutnya adalah: insensitifitas terhadap tipe data.**

**Artinya:**

**Kita bisa memberi dan mengubah nilai apa pun dari tipe data apa pun ke dalam sebuah variabel.**

**Bagi yang baru mempelajari bahasa pemrograman, ini mungkin biasa saja. Tapi bagi *old-school* yang pernah mempelajari Java dan semisalnya, akan merasa aneh. Karena pada bahasa-bahasa tersebut, satu variabel hanya diperuntukkan untuk satu tipe data saja.**

**Contoh ketidak-ketat-an python terhadap tipe data:**

* ***# nilai awal berupa integer***
* **a = 5**
* ***# kita ubah menjadi string dan tidak error***
* **a = 'Nurul Huda'**

## **Tanda Petik dan Tanda Petik Dua**

**Dalam bahasa pemrograman python, kita bisa mendefinisikan string dengan tanda petik satu ' maupun tanda petik dua ".**

**Contoh yang benar ✅:**

* **nama = 'Nurul Huda'**
* **asal = "Indonesia"**

**Contoh yang salah ❌:**

* **nama = 'Nurul Huda"**
* **asal = "Indonesia'**

## **Penulisan Komentar**

**Komentar adalah sebuah baris kode atau statemen yang diabaikan oleh interpreter python. Ia hanya ditulis dengan tujuan agar dibaca oleh manusia, bukan mesin.**

**Komentar juga sangat penting sebagai penjelasan alur dari kode program yang kita tulis. Jika tidak, kita sendiri (si penulis kode) bisa lupa dan kebingungan jika harus menjelaskan kode program lama yang pernah kita tulis pada masa lalu.**

**Penulisan komentar pada python terdiri dari 2 jenis:**

* **satu baris**
* **dan multi baris**

**Komentar satu baris ditulis dengan tanda #. Sedangkan komentar lebih dari satu baris ditulis dengan *triple doublequote* (tanda petik dua sebanyak 3x).**

**Contoh:**

* ***# variabel a merepresentasikan panjang***
* **a = 5**
* **b = 10 *# variabel b merepresentasikan tinggi***
* **"""**
* **Dan variabel c merepresentasikan luas**
* **persegi dari hasil perkalian**
* **variabel a dan variabel b**
* **"""**
* **c = a \* b**

**Python adalah bahasa pemrograman yang** [**berorientasi objek secara menyeluruh**](https://jagongoding.com/python/menengah/oop)**. Artinya semua variabel di dalam python adalah sebuah objek. Meskipun begitu, tentu saja terdapat jenis-jenis tipe data pada Python yang harus kita pahami.**

**Sebelum kita mulai praktik, kita samakan dulu frekuensi pemahaman kita.**

# **Python Dasar: Tipe Data dan Variabel**

## **Apa itu variabel?**

****

**Anggap saja variabel adalah sebuah keranjang, tempat di mana kita bisa memasukkan sesuatu di dalamnya, yaitu data.**

**Di python, kita bisa memasukkan tipe data apa saja ke dalam keranjang (yakni variabel) tanpa harus mendefinisikan tipe datanya terlebih dahulu (hal ini berbeda dengan beberapa bahasa pemrograman lain yang mengharuskan kita mendefinisikan tipe data terlebih dahulu).**

## **Apa itu tipe data?**

**Tipe data –*sesuai namanya*– ia adalah jenis dari suatu data. Setiap data memiliki nilai, dan setiap nilai memiliki jenis. Ada data-data yang bertipe angka, ada pula yang bertipe huruf/karakter, ada juga yang bertipe benar/salah dan sebagainya.**

**Sebagai ibarat, kalau variabel adalah keranjang, maka tipe data adalah jenis barang atau jenis benda yang akan kita masukkan ke dalam keranjang tersebut.**

**Gambar di bawah ini saya kira bisa memberikan ilustrasi dasar bagaimana hubungan sebuah variabel dan tipe data.**

****

**Kita bisa lihat bahwa di dalam gambar di atas, terdapat banyak kotak dan banyak buah. Setiap kotak tertentu digunakan untuk menyimpan jenis buah tertentu.**

**Sehingga bisa kita tarik kesimpulan bahwa:**

* **Kotak keranjang merepresentasikan variabel.**
* **Buah merepresentasikan data.**
* **Dan jenis-jenis buah tersebut merepresentasikan tipe data.**

## **Cara Membuat Variabel**

**Secara singkat, membuat variabel di Python sangat mudah sekali. Kita hanya perlu menuliskan nama variabel lalu diikuti oleh nilai yang kita inginkan.**

**Perhatikan contoh skrip berikut ini:**

**nama = 'Nurul Huda'**

**usia = 24**

**sudah\_menikah = True**

**print('nama:', nama)**

**print('ujia:', usia)**

**print('sudah menikah:', sudah\_menikah)**

***Kode program 01***

**Jika kita eksekusi, program di atas akan menghasilkan output seperti berikut:**

**nama: Nurul Huda**

**ujia: 24**

**sudah menikah: True**

### **Penjelasan**

**Pada skrip di atas, kita membuat 3 buah variabel:**

* **nama**
* **usia**
* **sudah\_menikah**

**Masing-masing variabel kita berikan sebuah nilai.**

* **Variabel nama memiliki nilai "Nurul Huda"**
* **Variabel usia memiliki nilai 24**
* **Variabel sudah\_menikah memiliki nilai True**

**Lalu di baris selanjutnya, kita menampilkan isi dari masing-masing variabel menggunakan perintah print().**

**Mudah sekali bukan?**

**Ya, tentu saja! 😁**

## **Aturan Penamaan Variabel**

**Secara umum, kita bisa membuat nama variabel apa saja yang kita mau di dalam python. Akan tetapi, terdapat beberapa aturan dan pengecualian.**

**Berikut ini aturan-aturannya secara sederhana:**

1. **Nama variabel hanya boleh diawali oleh huruf atau underscore.**
2. **Nama variabel tidak boleh diawali oleh angka.**
3. **Nama variabel hanya bisa terdiri dari karakter *alpha-numeric* dan underscore (A-z, 0-9, and \_ )**
4. **Nama variabel bersifat case sensitive. Artinya variabel nama berbeda dengan Nama atau naMA**

**Sebagai contoh, berikut ini adalah variabel-variabel yang benar xdan variabel-variabel yang salah:**

1. **\_nama ✅**
2. **1nama ❌**
3. **nama depan ❌**
4. **namaDepan ✅**
5. **nama\_belakang ✅**
6. **nama%lengkap ❌**

## **Aturan Assignment**

**Aturan *assignment* atau aturan pemberian nilai terhadap variabel di dalam Python bisa selesai secara *multiple* mau pun secara *single*.**

**Cara *single* adalah dengan memberikan satu nilai terhadap satu variabel dalam satu baris, sedangkan cara *multiple* adalah dengan memberikan *multiple nilai* terhadap *multiple variables* dalam satu baris.**

**Perhatikan contoh berikut:**

**a, b, c = 1, 2, "Mantap"**

**print('a:', a)**

**print('b:', b)**

**print('c:', c)**

***# kita juga bisa memberikan satu nilai yang sama untuk***

***# beberapa variabel***

**d = e = f = 10**

**print('d:', d)**

**print('e:', e)**

**print('f:', f)**

**Jika dijalankan, program di atas akan menghasilkan output:**

**a: 1**

**b: 2**

**c: Mantap**

**d: 10**

**e: 10**

**f: 10**

## **Memeriksa Tipe Data Pada Python**

**Sebelum kita memasuki contoh-contoh tipe data dasar pada python, kita akan mempelajari cara untuk memeriksa atau mengetahui tipe data dari suatu variabel.**

**Untuk melakukannya, kita bisa menggunakan fungsi type() bawaan python.**

**Perhatikan kode program berikut:**

**a = 'Madura'**

**b = 50**

**print(type(a))**

**print(type(b))**

**Output:**

**<class 'str'>**

**<class 'int'>**

## **Jenis-Jenis Tipe Data Python**

**Jika kita lihat kembali kode program 01 di atas, maka kita akan mendapati bahwa data dari masing-masing 3 variabel memiliki tipe data yang berbeda-beda.**

**nama = 'Nurul Huda'**

**usia = 24**

**sudah\_menikah = True**

* **Variabel nama memiliki tipe data string (teks)**
* **Variabel usia memiliki tipe data number (numerik)**
* **Dan variabel sudah\_menikah memiliki tipe data boolean (benar/salah)**

**Sekarang, kita akan mencoba dan memahami lebih jauh tentang masing-masing dari tipe data di atas.**

## **Tipe Data Numbers (Numerik)**

**Yang pertama adalah tipe data numerik. Tipe data numerik adalah semua jenis tipe yang bersifat angka, bisa ditambah, bisa dikurangi, bisa dikali, bisa dibagi, dan lain sebagainya.**

**Ada beberapa tipe data numerik pada python; seperti integer, float, dan kompleks.**

### **Integer**

**Tipe data integer adalah tipe data bilangan bulat. Sehingga setiap variabel yang memiliki nilai bilangan bulat, maka ia akan dikategorikan sebagai integer.**

### **Float**

**Hampir sama dengan tipe data integer, hanya saja tipe data float dipergunakan untuk variabel-variabel yang memiliki nilai pecahan / desimal.**

### **Complex**

**Sedangkan tipe data numerik yang lainnya adalah tipe data complex, sesuai namanya, ini adalah tipe data yang kompleks. Ia merepresentasikan nilai imajiner.**

### **Mencoba tipe data numerik**

**Setelah perkenalan singkat dengan 3 tipe data numerik pada python, sekarang waktunya untuk kita mencoba masing-masing dari jenis tipe data tersebut.**

**Buka lagi teks editor kalian, lalu tuliskan kode program berikut:**

**panjang = 5**

**lebar = 10.5**

**luas = panjang \* lebar**

**print(panjang, '\*', lebar, '=', luas)**

**print("Tipe dari variabel panjang:", type(panjang))**

**print("Tipe dari variabel lebar:", type(lebar))**

**print("Tipe dari variabel luas:", type(luas))**

**Dengan memanggil fungsi type(nama\_variabel), kita akan bisa mengetahui tipe data dari sebuah variabel.**

**Jika dijalankan, kode program di atas akan menghasilkan output seperti berikut:**

**5 \* 10.5 = 52.5**

**Tipe dari variabel panjang: <class 'int'>**

**Tipe dari variabel lebar: <class 'float'>**

**Tipe dari variabel luas: <class 'float'>**

**Dan untuk tipe data complex, silakan coba tulis dan jalankan kode program berikut**

**a = 5j**

**b = 10j**

**c = a + b**

**print(a, '+', b, '=', c)**

**print('Tipe dari a:', type(a))**

**print('Tipe dari b:', type(b))**

**print('Tipe dari c:', type(c))**

**Output dari kode program di atas adalah:**

**5j + 10j = 15j**

**Tipe dari a: <class 'complex'>**

**Tipe dari b: <class 'complex'>**

**Tipe dari c: <class 'complex'>**

## **Tipe Data String (Teks)**

**Selanjutnya yang akan kita bahas adalah tipe data string. Ia adalah tipe data yang digunakan untuk menyimpan sebuah teks.**

**Data yang bertipe string harus diapit oleh tanda petik, baik tanda petik satu ('') mau pun tanda petik dua ("").**

**Silakan perhatikan contoh kode program berikut:**

**nama\_depan = "Wahit"**

**nama\_belakang = 'Abdulloh'**

**nama\_lengkap = nama\_depan + ' ' + nama\_belakang**

**usia = '12'**

**alamat = 'Bangkalan'**

**kata\_mutiara = "Don't judge a book by it's cover"**

**print(nama\_lengkap, '(' + usia + ')', ',', 'dari', alamat, ', kata mutiara:', kata\_mutiara)**

**print('\nTipe dari nama\_lengkap:', type(nama\_lengkap))**

**print('Tipe dari usia:', type(usia))**

**print('Tipe dari alamat:', type(alamat))**

**print('Tipe dari kata\_mutiara:', type(kata\_mutiara))**

**Jika kita menjalankan kode program di atas, hasil yang akan kita dapat adalah seperti berikut:**

**Wahit Abdulloh (12) , dari Bangkalan , kata mutiara: Don't judge a book by it's cover**

**Tipe dari nama\_lengkap: <class 'str'>**

**Tipe dari usia: <class 'str'>**

**Tipe dari alamat: <class 'str'>**

**Tipe dari kata\_mutiara: <class 'str'>**

### **Catatan**

**Coba perhatikan variabel usia, meskipun isinya adalah sebuah angka numerik, tetap saja di situ dia bertipe data string.**

**Kenapa? karena ia diapit oleh tanda petik.**

### **Lalu, apa perbedaan antara tipe data numerik dan tipe data teks (string)?**

**Perbedaannya terletak pada fungsi dan cara mengoperasikannya.**

**Misalkan kita ingin menambahkan dua buah variabel bertipe data numerik, yang kita dapatkan adalah hasil penjumlahannya.**

**Berbeda jika kita menambahkan dua buah variabel bertipe data string (teks), yang kita dapatkan adalah hasil penggabungan keduanya.**

**Perhatikan contoh berikut:**

***# penjumlahan dua data numerik***

**print(5 + 5) *# output 10***

***# penjumlahan dua data string***

**print('5' + '5') *# output 55***

**Output dari kode program di atas:**

**10**

**55**

**Oleh karena itu: pemilihan tipe data yang tepat sangatlah penting agar tidak terjadi pada kesalahan operasi.**

## **Tipe Data Boolean (Benar/Salah)**

**Selanjutnya adalah tipe data boolean.**

**Tipe data boolean adalah tipe data yang paling simpel dan mudah. Akan tetapi dia sangat penting sekali bahkan untuk membangun program/aplikasi skala besar sekalipun.**

**Tipe data boolean hanya memiliki dua buah nilai, yaitu; True dan False.**

**Nilai True untuk pernyataan bernilai benar, dan False untuk merepresentasikan pernyataan yang bernilai salah.**

**Simpelnya, kita bisa mempraktikkan kode program berikut:**

**saya\_orang\_indonesia = True**

**saya\_adalah\_robot = False**

**print('Apakah saya orang Indonesia?', saya\_orang\_indonesia)**

**print('Apakah saya adalah robot?', saya\_adalah\_robot)**

**print('Tipe dari saya\_orang\_indonesia', type(saya\_orang\_indonesia))**

**print('Tipe dari saya\_adalah\_robot', type(saya\_adalah\_robot))**

**Jika dijalankan, maka output yang akan kita dapatkan adalah:**

**Apakah saya orang Indonesia? True**

**Apakah saya adalah robot? False**

**Tipe dari saya\_orang\_indonesia <class 'bool'>**

**Tipe dari saya\_adalah\_robot <class 'bool'>**

**Tipe data Boolean adalah tipe data yang sangat penting. Ia bisa berfungsi untuk mengontrol laju dan alur dari program yang kita bangun. Tipe data Boolean sangat berkaitan erat dengan** [**operator logika**](https://jagongoding.com/python/dasar/operator/) **yang insyaallah akan kita bahas nanti pada seri** [**belajar python dasar**](https://jagongoding.com/python/dasar/overview) **ini.**

## **Tipe Data Canggih**

**Masih ada beberapa tipe data lagi di dalam bahasa pemrograman Python. Tipe data tersebut sebenarnya tidak terlalu canggih seperti yang dibayangkan, hanya saja saya katakan itu sebagai tipe data canggih karena cara kerjanya tidak seperti tipe data yang sudah kita bahas di atas.**

**Di antara tipe data tersebut adalah tipe data:**

* [**List**](https://jagongoding.com/python/dasar/list)
* [**Tuple**](https://jagongoding.com/python/dasar/tuple)
* [**Set**](https://jagongoding.com/python/dasar/set)
* [**Dictionary**](https://jagongoding.com/python/dasar/dictionary)

# **Python Dasar: Tipe Data Tuple**

## **Apa itu Tuple**

**Tuple adalah 1 dari 4 tipe data kolektif pada python yang berguna untuk menyimpan lebih dari satu nilai dalam satu variabel secara sekaligus** [**[1]**](https://www.w3schools.com/python/python_tuples.asp)**.**

**Tuple bersifat *ordered* (terurut) dan juga bersifat *unchangable* (tidak bisa diubah). *Ordered* berarti datanya bisa kita akses menggunakan indeks, dan unchangeable berarti datanya tidak akan pernah bisa diubah setelah pertama kali definisikan.**

**Dalam python, tipe data tuple didefinisikan dengan tanda kurung ().**

## **Apa bedanya Tuple dengan List?**

**Tuple sama saja dengan list. Dia sama-sama digunakan untuk menyimpan data himpunan. Sama-sama bisa menampung berbagai macam tipe data dalam satu himpunan. Hanya saja setelah diberi nilai, tuple tidak bisa diubah lagi. Hal ini berbeda dengan list.**

**Dari segi penulisan, list menggunakan kurung siku [] sedangkan tuple menggunakan kurung biasa ().**

## **Bagaimana cara membuat tuple?**

**Cara membuat tuple sangat mudah dan simpel. Pertama, siapkan dulu secangkir kopi ☕, lalu buka text editor favorit kalian sebelum kita mulai menulis kode 👨🏻‍💻.**

**Ada 3 cara untuk membuat tuple. Perhatikan contoh berikut:**

***# cara standar***

**tuple\_jenis\_kelamin = ('laki-laki', 'perempuan')**

***# tanpa kurung***

**tuple\_status\_perkawinan = 'menikah', 'lajang'**

***# menggunakan fungsi tuple()***

**tuple\_lulus = tuple(['lulus', 'tidak lulus'])**

### **Keterangan**

* **Cara yang pertama adalah cara standar dan paling dasar**
* **Cara yang kedua tanpa tanda kurung. Ini mungkin kelihatan agak aneh, tapi yang seperti ini normal di python 😏**
* **Cara yang ketiga adalah dengan menggunakan fungsi tuple() dan melemparkan list sebagai parameternya.**

### **Tuple kosong**

**Untuk membuat tuple kosong, kita cukup dengan menuliskan dua tanda kurung seperti berikut:**

**tuple\_kosong = ()**

### **Tuple yang hanya berisi satu item**

**Untuk mendefinisikan tuple yang hanya berisi satu item, kita tetap diharuskan menulis tanda koma.**

**tuple\_tunggal = (10,)**

**Kalau tidak, maka python akan menggap tanda kurungnya tidak ada, seperti contoh berikut:**

**print(type((10))) *# yang ini dianggap integer biasa***

**print(type((10,))) *# yang ini dianggap tuple***

**Output:**

**<class 'int'>**

**<class 'tuple'>**

## **Cara mengakses nilai tuple**

**Mengakses data pada tuple tidak jauh berbeda dengan cara mengakses data pada list, bahkan bisa kita bilang sama persis dalam keumumannya.**

**Kita bisa mengakses nilai pada tuple dengan langsung mendefinisikan indeks-nya seperti berikut:**

***# cara standar***

**tuple\_jenis\_kelamin = ('laki-laki', 'perempuan')**

**print(tuple\_jenis\_kelamin[1]) *# indeks satu***

**print(tuple\_jenis\_kelamin[0]) *# indeks nol***

**Output:**

**perempuan**

**laki-laki**

**Kita juga bisa mengakses nilai pada tuple dengan negatif indeks:**

**print(tuple\_jenis\_kelamin[-2])**

**print(tuple\_jenis\_kelamin[-1])**

**laki-laki**

**perempuan**

## **Slicing tuple**

***Slicing* adalah teknik memotong nilai dari sebuah tuple. Sintaksnya sama saja dengan teknis** [***slicing* di list**](https://jagongoding.com/python/dasar/list/#slicing-list)**. Tidak berbeda.**

**Untuk melakukan slicing, kita perlu mendefinisikan range indeks dengan pemisah tanda titik dua (:).**

**Perhatikan kode berikut:**

**tuple\_buah = ('Pisang', 'Nanas', 'Melon', 'Durian')**

**print(tuple\_buah[0:1])**

**print(tuple\_buah[0:2])**

**print(tuple\_buah[1:3])**

**print(tuple\_buah[0:-1])**

**print(tuple\_buah[-1:-3])**

**print(tuple\_buah[-1:3])**

**print(tuple\_buah[-3:-1])**

**Output:**

**('Pisang',)**

**('Pisang', 'Nanas')**

**('Nanas', 'Melon')**

**('Pisang', 'Nanas', 'Melon')**

**()**

**()**

**('Nanas', 'Melon')**

## **Slicing tanpa batas**

***Slicing* pada tuple juga bisa dilakukan tanpa harus mendefinisikan batas sebelah kanan mau pun sebelah kiri. Sebagaimana contoh berikut:**

**tuple\_buah = ('Pisang', 'Nanas', 'Melon', 'Durian')**

**print(tuple\_buah[0:])**

**print(tuple\_buah[1:])**

**print(tuple\_buah[2:])**

**print(tuple\_buah[3:])**

**print(tuple\_buah[:0])**

**print(tuple\_buah[:1])**

**print(tuple\_buah[:2])**

**print(tuple\_buah[:3])**

**print(tuple\_buah[:4])**

**Jika dijalankan, kode program di atas akan menghasilkan output:**

**('Pisang', 'Nanas', 'Melon', 'Durian')**

**('Nanas', 'Melon', 'Durian')**

**('Melon', 'Durian')**

**('Durian',)**

**()**

**('Pisang',)**

**('Pisang', 'Nanas')**

**('Pisang', 'Nanas', 'Melon')**

**('Pisang', 'Nanas', 'Melon', 'Durian')**

## **Mengubah data pada tuple**

**Kita telah mempelajari pada pertemuan sebelumnya bahwa kita bisa mengubah data pada list. Akan tetapi, berbeda dengan Tuple. Kita tidak bisa mengubah data pada tuple. Dan jika kita *kekeuh* berusaha mengubah data tertentu dari tuple, kita hanya akan mendapatkan error.**

**Perhatikan contoh berikut:**

***# mencoba edit data list***

**list\_siswa = ['Andi', 'Budi']**

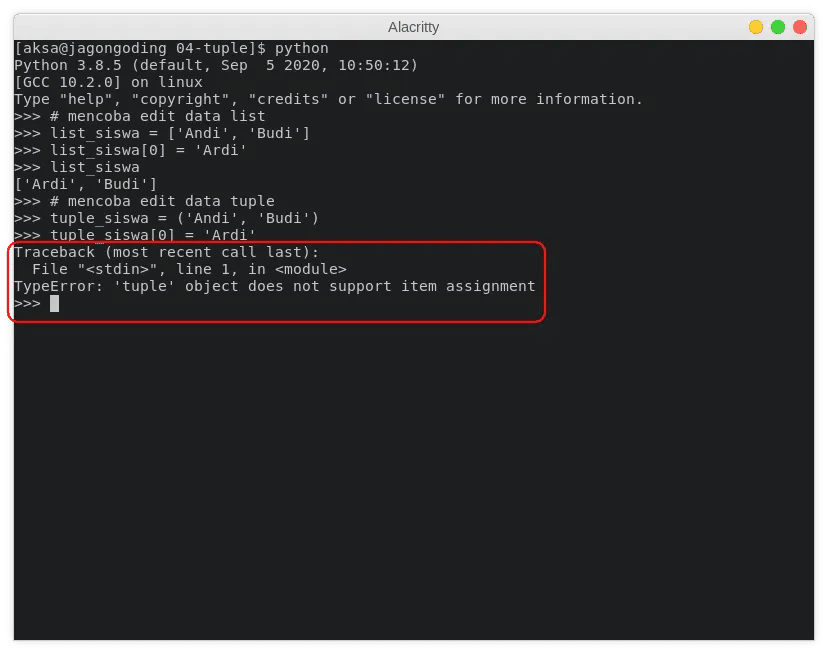
**list\_siswa[0] = 'Ardi'**

***# mencoba edit data tuple***

**tuple\_siswa = ('Andi', 'Budi')**

**tuple\_siswa[0] = 'Ardi'**

**Jika saya jalankan melalui python mode interaktif, berikut ini error yang saya dapatkan:**

****

**Jika kalian benar-benar ingin mengubah data yang ada pada tuple. Maka gunakanlah list! 😁**

**Tuple memang difungsikan untuk kasus-kasus tertentu di mana nilai data bersifat tetap, tidak akan berubah selama runtime. Seperti misalnya data jenis kelamin.**

## **Sequence Unpacking**

**Fitur selanjutnya dari Tuple adalah: *sequence unpacking*. Fitur ini berfungsi untuk mengekstrak isi dari tuple ke dalam variabel-variabel tunggal secara berurutan. Kita hanya perlu menggunakan** [**operator *assignment***](https://jagongoding.com/python/dasar/operator/#operator-penugasan) **standar (simbol sama dengan =) dan mendefinisikan nama variabel dengan koma** [**[2]**](https://www.pythoncontent.com/unpacking-sequence-in-python)**.**

**Contohnya adalah seperti ini:**

**siswa = ('Nurul Huda', 'Bangkalan', 24)**

***# ekstrak data atau juga dinamakan sequence unpacking***

**nama, asal, usia = siswa**

***# setiap variabel di atas akan memiliki nilai dari tiap isi tuple***

***# secara berurutan***

**print('Nama:', nama)**

**print('Asal:', asal)**

**print('Usia:', usia)**

**Jika kita jalankan, kita akan mendapatkan output seperti berikut:**

**Nama: Nurul Huda**

**Asal: Bangkalan**

**Usia: 24**

### **Penjelasan**

**Jika kita perhatikan kode program dan output di atas, kita bisa simpulkan bahwa:**

1. **Variabel nama memiliki nilai dari siswa[0]**
2. **Variabel asal memiliki nilai dari siswa[1]**
3. **Variabel usia memiliki nilai dari siswa[2]**

**Dan pengekstrakan nilai tersebut terjadi sesuai dengan urutan penulisan variabel, teman-teman bisa coba bolak-balikkan urutan variabelnya lalu lihat hasilnya seperti apa 👨🏻‍💻.**

## **Menggabungkan dua buah tuple atau lebih**

**Hal lain yang bisa kita lakukan dengan tuple adalah: menggabungkan beberapa tuple menjadi satu tuple baru.**

**Kita bisa melakukan hal tersebut menggunakan operator penjumlahan + seperti contoh berikut:**

**>>> a = (1, 2, 3)**

**>>> b = (50, 60, 70)**

**>>>**

**>>> c = a + b**

**>>> c**

**(1, 2, 3, 50, 60, 70)**

**>>>**

## **Fungsi-fungsi bawaan tuple**

**Sama seperti list, tuple juga memiliki fungsi-fungsi bawaan yang bisa kita gunakan seperti berikut:**

| **Nama** | **Keterangan** |
| --- | --- |
| **len()** | **Menghitung jumlah item pada tuple** |
| **max()** | **Mencari nilai paling besar dari sebuah tuple** |
| **min()** | **Mencari nilai paling kecil dari sebuah tuple** |

**Contoh penggunaan:**

**>>> nilai\_semester\_1 = (80, 90, 100, 88, 60)**

**>>> max(nilai\_semester\_1)**

**100**

**>>> min(nilai\_semester\_1)**

**60**

**>>> len(nilai\_semester\_1)**

**5**

# **Python Dasar: 7 Jenis Operator**

## **Apa itu Operator?**

**Operator di dalam Python adalah simbol khusus yang berfungsi untuk menjalankan suatu operasi tertentu, baik operasi aritmatika maupun operasi logika. Sedangkan nilai yang dioperasikan oleh operator dinamakan sebagai operan** [**[1]**](https://www.programiz.com/python-programming/operators)**.**

**Berikut ini salah satu contoh paling sederhana dari operator aritmatika pada Python:**

**>>> 10 + 5**

**15**

**Pada kode program di atas, tanda + adalah sebuah operator. Sedangkan angka 10 dan 5 keduanya merupakan operan.**

**Dari operasi tersebut, didapatkanlah sebuah hasil akhir berupa nilai integer yaitu 15.**

## **Jenis-jenis operator pada python**

**Terdapat 7 jenis operator pada python:**

* [**Operator aritmatika**](https://jagongoding.com/python/dasar/operator/#operator-aritmatika)
* [**Operator komparasi atau perbandingan**](https://jagongoding.com/python/dasar/operator/#operator-komparasi-atau-perbandingan)
* [**Operator penugasan**](https://jagongoding.com/python/dasar/operator/#operator-penugasan)
* [**Operator logika**](https://jagongoding.com/python/dasar/operator/#operator-logika)
* [**Operator keanggotaan (*membership*)**](https://jagongoding.com/python/dasar/operator/#operator-keanggotaan)
* [**Operator identitas**](https://jagongoding.com/python/dasar/operator/#operator-identitas)
* [**Operator *bitwise***](https://jagongoding.com/python/dasar/operator/#operator-bitwise)

**Markibas tusatu. Mari kita bahas satu-persatu! 👨‍💻**

## **Operator aritmatika**

**Operator matematika adalah operator yang kita gunakan untuk menghitung operasi matematika, mulai dari penjumlahan, pengurangan, perkalian, perpangkatan, dan lain sebagainya** [**[2]**](https://www.microfocus.com/documentation/silk-test/205/en/silktestclassic-help-en/STCLASSIC-F02D493B-ARITHMETICOPERATORS-REF.html)**.**

**Berikut ini tabel operator aritmatika pada python 🐍.**

| **Simbol** | **Nama** | **Tugas** | **Contoh** |
| --- | --- | --- | --- |
| **+** | **Penjumlahan** | **Menjumlahkan nilai operan** | **5 + 1 = 6** |
| **-** | **Pengurangan** | **Mengurangkan nilai operan** | **10 - 10 = 0** |
| **\*** | **Perkalian** | **Mengalikan nilai operan** | **2 \* 2 = 4** |
| **/** | **Pembagian** | **Membagi nilai operan** | **100 / 5 = 50** |
| **%** | **Modulus** | **Menghitung sisa hasil bagi dari operan** | **10 / 3 = 1** |
| **\*\*** | **Perpangkatan** | **Menghitung pangkat dari operan** | **2 \*\* 3 = 8** |
| **//** | **Pembagian Bulat** | **Membagi operan lalu membulatkannya dengan menghapus angka dibelakang koma** | **10 // 3 = 3** |

**Kita bisa mencoba satu-persatu dari masing-masing operator di atas.**

**a, b = 10, 3**

**print(a, '+', b, '=', a + b)**

**print(a, '-', b, '=', a - b)**

**print(a, '\*', b, '=', a \* b)**

**print(a, '/', b, '=', a / b)**

**print(a, '%', b, '=', a % b)**

**print(a, '\*\*', b, '=', a \*\* b)**

**print(a, '//', b, '=', a //b)**

**Output:**

**10 + 3 = 13**

**10 - 3 = 7**

**10 \* 3 = 30**

**10 / 3 = 3.3333333333333335**

**10 % 3 = 1**

**10 \*\* 3 = 1000**

**10 // 3 = 3**

## **Operator komparasi atau perbandingan**

**Operator perbandingan adalah operator yang bertugas untuk membandingkan antar dua operan. Apakah operan 1 lebih besar dari pada operan 2, atau apakah keduanya sama? Dan lain sebagainya.**

**Berikut ini adalah tabel operator perbandingan pada Python.**

| **Simbol** | **Nama** | **Contoh** | **Hasil** |
| --- | --- | --- | --- |
| **>** | **Lebih dari** | **5 > 5** | **False** |
| **<** | **Kurang dari** | **2 < 4** | **True** |
| **==** | **Sama dengan** | **10 == 10** | **True** |
| **!=** | **Tidak sama dengan** | **5 != 5** | **False** |
| **>=** | **Lebih dari atau sama dengan** | **10 >= 10** | **True** |
| **<=** | **Kurang dari atau sama dengan** | **9 <= 10** | **True** |

**Agar menjadi lebih jelas, mari kita gerak-gerakkan jari kita 😂**

***# buat variabel a dan b dengan teknik squence ordering***

**a, b = 5, 10**

**print(a, '>', b, '=', a > b)**

**print(a, '<', b, '=', a < b)**

**print(a, '==', b, '=', a == b)**

**print(a, '!=', b, '=', a != b)**

**print(a, '>=', b, '=', a >= b)**

**print(a, '<=', b, '=', a <= b)**

**Output dari kode program di atas adalah:**

**5 > 10 = False**

**5 < 10 = True**

**5 == 10 = False**

**5 != 10 = True**

**5 >= 10 = False**

**5 <= 10 = True**

## **Operator penugasan**

**Operator penugasan adalah operator yang digunakan untuk memberikan sebuah tugas terhadap suatu variabel. Atau dalam bahasa yang lebih manusiawi: operator penugasan adalah operator yang berfungsi untuk memberikan nilai ke dalam sebuah variabel.**

**Sebenarnya operator penugasan ini hanya ada 1 saja, yaitu operator =.**

**Akan tetapi, ada banyak *variant shortcut* yang memudahkan kita untuk melakukan operasi aritmatika atau operasi *bitwise* bersamaan dengan operasi penugasan.**

**Berikut ini adalah tabel operator penugasan pada Python.**

| **Operator** | **Contoh** | **Sama dengan** |
| --- | --- | --- |
| **=** | **a = 10** | **a = 10** |
| **+=** | **a += 5** | **a = a + 5** |
| **-=** | **a -= 3** | **a = a - 3** |
| **\*=** | **a \*= 6** | **a = a \* 6** |
| **/=** | **a /= 8** | **a = a / 8** |
| **%=** | **a %= 9** | **a = a % 9** |
| **//=** | **a //= 6** | **a = a // 6** |
| **\*\*=** | **a \*\*= 1** | **a = a \*\* 1** |
| **&=** | **a &= 2** | **a = a & 2** |
| **|=** | **a |= 3** | **a = a | 3** |
| **^=** | **a ^= 4** | **a = a ^ 4** |
| **>>=** | **a »= 4** | **a = a » 4** |
| **<<=** | **a «= 2** | **a = a « 2** |

**Agar lebih ahsan (baik), mari kita coba masing-masing dari operator penugasan di atas.**

***# penugasan pertama***

**a = 10**

**print('a = 10 -> ', a)**

**a += 5**

**print('a += 5 -> ', a)**

**a -= 3**

**print('a -= 3 -> ', a)**

**a \*= 6**

**print('a \*= 6 -> ', a)**

**a /= 8**

**print('a /= 8 -> ', a)**

***# karena a jadi float, kita ubah lagi menjadi integer***

**a = int(a)**

**a %= 9**

**print('a %= 9 -> ', a)**

**a //= 6**

**print('a //= 6 -> ', a)**

**a \*\*= 1**

**print('a \*\*= 1 -> ', a)**

**a &= 2**

**print('a &= 2 -> ', a)**

**a |= 3**

**print('a |= 3 -> ', a)**

**a ^= 4**

**print('a ^= 4 -> ', a)**

**a >>= 4**

**print('a >>= 4 -> ', a)**

**a <<= 2**

**print('a <<= 4 -> ', a)**

**Output dari kode program di atas adalah:**

**a = 10 -> 10**

**a += 5 -> 15**

**a -= 3 -> 12**

**a \*= 6 -> 72**

**a /= 8 -> 9.0**

**a %= 9 -> 0**

**a //= 6 -> 0**

**a \*\*= 1 -> 0**

**a &= 2 -> 0**

**a |= 3 -> 3**

**a ^= 4 -> 7**

**a >>= 4 -> 0**

**a <<= 4 -> 0**

**NB: beberapa operator terakhir di atas mengandung operasi bitwise. Operasi bitwise ada di bagian akhir dari pembahasan hari ini.**

## **Operator logika**

**Operator logika adalah operator yang sangat penting. Operator ini sangat berkaitan erat dengan operator perbandingan. Dan kedua-duanya juga mengembalikan nilai dengan tipe data yang sama yaitu boolean.**

**Berikut ini tabel dari operator logika pada python.**

| **Simbol** | **Tugas** | **Contoh** |
| --- | --- | --- |
| **and** | **Mengembalikan True jika dua statement sama-sama benar** | **True and True** |
| **or** | **Mengembalikan True jika salah satu statement bernilai benar** | **2 > 5 or 1 < 3** |
| **not** | **Menegasikan hasil. True menjadi False dan sebaliknya** | **not(1 > 5)** |

**Mari kita coba masing-masing dari operator di atas.**

**print(True and True)**

**print(1 + 2 == 3 and True)**

**print('----')**

**print(False or 1 > 5)**

**print(False or 5 > 2)**

**print('----')**

**print(not(1 > 5))**

**print(not(1 < 5))**

**Jika kita jalankan, kita akan mendapatkan output sebagai berikut:**

**True**

**True**

**----**

**False**

**True**

**----**

**True**

**False**

## **Operator keanggotaan**

**Python adalah bahasa pemrograman yang terbilang unik, ia memiliki operator khusus atau juga sering dinamakan sebagai operator spesial. Dinamakan spesial karena memang operator ini hanya ada di Python dan tidak ada di bahasa pemrograman lainnya.**

**Di antara operator spesial tersebut adalah operator keanggotaan. Atau di dalam bahasa inggris ia dinamakan *membership operator*.**

**Operator keanggotaan dalam python hanya memiliki dua varian, yaitu in dan not in. Berikut ini tabel lebih lengkapnya:**

| **Simbol** | **Tugas** |
| --- | --- |
| **in** | **Bernilai true jika suatu nilai ada di dalam *sequence*** |
| **not in** | **Bernilai false jika suatu nilai tidak ada di dalam *sequence*** |

**Agar lebih jelas, mari kita coba kode program berikut ini:**

**perusahaan = 'Microsoft'**

**list\_pulau = ['Jawa', 'Sumatra', 'Sulawesi']**

***# ini adalah dictionary, insyaallah akan kita pelajari***

***# di pertemuan-pertemuan yang akan datang***

**mahasiswa = {**

**'nama': 'Lendis Fabri',**

**'asal': 'Lamongan'**

**}**

**print(**

**"Apakah 'c' ada di variabel perusahaan?",**

**'c' in perusahaan**

**)**

**print(**

**"Apakah 'z' tidak ada di variabel perusahaan?",**

**'c' not in perusahaan**

**)**

**print(**

**"Apakah 'Madura' ada di variabel list\_pulau?",**

**'Madura' in perusahaan**

**)**

**print(**

**"Apakah 'Madura' tidak ada di variabel list\_pulau?",**

**'Madura' not in perusahaan**

**)**

**print(**

**"Apakah atribut 'nama' ada di variabel mahasiswa?",**

**'nama' in mahasiswa**

**)**

**Jika kita jalankan program di atas, kita akan mendapatkan output sebagai berikut:**

**Apakah 'c' ada di variabel perusahaan? True**

**Apakah 'z' tidak ada di variabel perusahaan? False**

**Apakah 'Madura' ada di variabel list\_pulau? False**

**Apakah 'Madura' tidak ada di variabel list\_pulau? True**

**Apakah atribut 'nama' ada di variabel mahasiswa? True**

## **Operator identitas**

**Selain operator keanggotaan, python juga masih memiliki operator spesial lainnya: yaitu operator identitas. Operator ini didefinisikan dengan is dan is not.**

**Tugasnya adalah untuk mengetahui apakah dua buah variabel memiliki nilai yang sama dan posisi yang sama di dalam memori. Karena tidak semua nilai yang sama memiliki tempat / posisi yang sama di dalam memori.**

| **Simbol** | **Tugas** |
| --- | --- |
| **is** | **Bernilai true jika dua variabel bersifat identik baik dari segi nilai mau pun penempatan lokasi di memory** |
| **is not** | **Bernilai false jika dua variabel tidak identik baik dari segi nilai mau pun penempatan lokasi di memory** |

**Untuk lebih jelasnya, markitkod. Mari kita koding 😁**

**Silakan tulis dan jalankan kode program di bawah ini:**

**a = 5**

**b = 5**

**list\_a = [1, 2, 3]**

**list\_b = [1, 2, 3]**

**nama\_a = 'budi'**

**nama\_b = 'budi'**

***# output True***

**print('a is b:', a is b)**

***# output False***

**print('a is not b:', a is not b)**

***# output False***

**print('list\_a is list\_b:', list\_a is list\_b)**

***# output True***

**print('list\_a == list\_b:', list\_a == list\_b)**

***# output True***

**print('nama\_a is nama\_b:', nama\_a is nama\_b)**

***# output False***

**print('nama\_a is not nama\_b:', nama\_a is not nama\_b)**

**Output:**

**a is b: True**

**a is not b: False**

**list\_a is list\_b: False**

**list\_a == list\_b: True**

**nama\_a is nama\_b: True**

**nama\_a is not nama\_b: False**

### **Penjelasan**

**Kita coba perhatikan output dari kode program di atas. Dari 3 pasang variabel yang kita tes, masing-masing memiliki nilai yang sama.**

1. **Variabel a dan b sama-sama memiliki nilai 5. Dan ketika kita periksa, mereka *equal* dan *identical*. Alias interpreter python menaruh nilai dari kedua variabel tersebut pada lokasi yang sama dalam memory.**
2. **Pada pasangan kedua, variabel list\_a dan list\_b juga memiliki nilai yang sama. Setelah kita tes, ternyata mereka berdua tidak *identical* meskipun nilai mereka berdua *equal*. Artinya, interpreter python menaruh nilai dari 2 variabel tersebut pada tempat yang berbeda pada memory.**
3. **Untuk pasangan yang terakhir, variabel nama\_a dan nama\_b yang bertipe data string, mereka berdua memiliki nilai yang *equal* dan juga *identical*.**

**Untuk mengetahui id atau lokasi penyimpanan suatu nilai pada python, kita bisa memanggil fungsi id().**

**>>> id('merah')**

**140442358081648**

**>>> id('merah')**

**140442358081776**

**>>>**

**Pada kode di atas, kita mendefinisikan nilai merah sebanyak 2x, dan python menempatkan dua nilai tersebut di lokasi yang berbeda di dalam memory.**

**Contoh berikutnya untuk nilai integer:**

**>>> id(10)**

**140442360371792**

**>>> id(10)**

**140442360371792**

**>>>**

**Di dalam kode di atas, kita mendefinisikan nilai 10 sebanyak 2x, akan tetapi python menempatkan keduanya dalam posisi yang sama.**

## **Operator bitwise**

**Operator terakhir yang akan kita bahasa dalam pertemuan ini adalah operator bitwise.**

**Operator bitwise adalah operator yang berhubungan dengan angka-angka biner.**

**Angka-angka biner adalah angka 0 dan 1. Dan pada hakikatnya hanya ini lah angka yang dipahami oleh mesin.**

**Sebelum kita mulai, kita bisa mengetahui nilai biner dari suatu angka desimal dengan melakukan perintah format() dengan parameter kedua berupa string '08b'. Berikut ini demonstrasi menggunakan python mode interaktif.**

**>>> *# biner dari angka 0***

**>>> print(format(0, '08b'))**

**00000000**

**>>> *# biner dari angka 1***

**>>> print(format(1, '08b'))**

**00000001**

**>>> *# biner dari angka 2***

**>>> print(format(2, '08b'))**

**00000010**

**>>> *# biner dari angka 37***

**>>> print(format(37, '08b'))**

**00100101**

**>>>**

**Setelah sedikit perkenalan dengan binary, berikut ini adalah tabel yang menjelaskan tentang operator bitwise pada python.**

| **Simbol** | **Nama** | **Tugas** |
| --- | --- | --- |
| **&** | **Bitwise AND** | **Mengembalikan bit 1 jika dua bit bernilai 1** |
| **|** | **Bitwise OR** | **Mengembalikan bit 1 jika salah satu bit bernilai 1** |
| **^** | **Bitwise XOR** | **Mengembalikan bit 1 jika hanya satu bit saja yang bernilai 1** |
| **-** | **Bitwise NOT** | **Membalikkan semua bit** |
| **>>** | **Bitwise right shift** | **Menggeser bit ke kanan dengan mendorong salinan digit sebelah kiri dan membiarkan digit sebalah kanan terlepas** |
| **<<** | **Bitwise left shift** | **Menggeser bit ke kiri dengan mendorong digit 0 dan membiarkan bit paling kiri terlepas** |

**Mari kita coba satu persatu dari kode operator bitwise di atas.**

**a = 1**

**b = 64**

**print('a =', a, '=', format(a, '08b'))**

**print('b =', b, '=', format(b, '08b'), '\n')**

**print('[and]')**

**print('a & b =', a & b)**

**print(format(a, '08b'), '&', format(b, '08b'), '=', format(a & b, '08b'), '\n')**

**print('[or]')**

**print('a | b =', a | b)**

**print(format(a, '08b'), '|', format(b, '08b'), '=', format(a | b, '08b'), '\n')**

**print('[xor]')**

**print('a ^ b =', a ^ b)**

**print(format(a, '08b'), '^', format(b, '08b'), '=', format(a ^ b, '08b'), '\n')**

**print('[not]')**

**print('~a ~b =', ~a, ~b)**

**print('~' + format(a, '08b'), '~' + format(b, '08b'), '=', format(~a, '08b'), format(~b, '08b'), '\n')**

**print('[shift right]')**

**print('a >> b =', a >> b)**

**print(format(a, '08b'), '>>', format(b, '08b'), '=', format(a >> b, '08b'), '\n')**

**print('[shift left]')**

**print('b << a =', b << a)**

**print(format(b, '08b'), '<<', format(a, '08b'), '=', format(b << a, '08b'), '\n')**

**Output dari program di atas adalah:**

**a = 1 = 00000001**

**b = 64 = 01000000**

**[and]**

**a & b = 0**

**00000001 & 01000000 = 00000000**

**[or]**

**a | b = 65**

**00000001 | 01000000 = 01000001**

**[xor]**

**a ^ b = 65**

**00000001 ^ 01000000 = 01000001**

**[not]**

**~a ~b = -2 -65**

**~00000001 ~01000000 = -0000010 -1000001**

**[shift right]**

**a >> b = 0**

**00000001 >> 01000000 = 00000000**

**[shift left]**

**b << a = 128**

**01000000 << 00000001 = 10000000**

# **Python Dasar: Tipe Data List**

## **Tipe Data List**

**Tipe data list adalah tipe data koleksi yang bersifat *ordered* (terurut) dan juga bersifat *changable* (bisa diubah). Tipe data ini bisa kita definisikan dengan tanda kurung siku [] di dalam Python.**

**Kita bisa abaikan dulu pengertiannya karena pada tahap ini, praktik lebih penting untuk membentuk pemahaman yang lebih dalam terhadap suatu tipe data.**

## **Bagaimana cara membuat list?**

**Langsung saja, seduh kopi anda, buka teks editor, lalu coba tuliskan kode program di bawah ini.**

***# list kosong***

**list\_kosong = []**

***# list yang berisi kumpulan string***

**list\_buah = ['Pisang', 'Nanas', 'Melon', 'Durian']**

***# list yang berisi kumpulan integer***

**list\_nilai = [80, 70, 90, 60]**

***# list campuran berbagai tipe data***

**list\_jawaban = [150, 33.33, 'Presiden Sukarno', False]**

***Kode program 1***

**Pada Kode program 1, kita lihat bahwa sebuah list didefinisikan menggunakan tanda kurung siku ([]).**

**Kita juga saksikan bahwa list pada python, bisa berisi berbagaimacam tipe data. Bisa terdiri dari tipe data yang sejenis mau pun dari tipe data yang berbeda-beda.**

## **Menampilkan List**

**Kita bisa menggunakan perintah print() untuk melihat isi dari sebuah list, baik secara menyeluruh maupun sebagian.**

**Pada Kode program 1 di atas, tambahkan kode program berikut ini untuk menampilkan semua isi dari list:**

**print('list\_kosong:', list\_kosong)**

**print('list\_buah:', list\_buah)**

**print('list\_nilai:', list\_nilai)**

**print('list\_jawaban:', list\_jawaban)**

**Jika dijalankan, kita akan mendapatkan output sebagai berikut:**

**list\_kosong: []**

**list\_buah: ['Pisang', 'Nanas', 'Melon', 'Durian']**

**list\_nilai: [80, 70, 90, 60]**

**list\_jawaban: [150, 33.33, 'Presiden Sukarno', False]**

**Kita juga bisa menampilkan isi tertentu dari list dengan menggunakan indeks. Setiap data pada list memiliki indeks sebagai alamat. Dan indeks adalah sebuah nilai integer dimulai dari 0 yang menjadi acuan di mana sebuah data disimpan di dalam list.**

**Tambahkan kode program berikut:**

**print(list\_buah[0])**

**print(list\_buah[2])**

**print(list\_buah[1])**

**print(list\_buah[3])**

**Output:**

**Pisang**

**Melon**

**Nanas**

**Durian**

**Kita juga bisa menggunakan indeks negatif untuk menampilkan data dari belakang. Perhatikan contoh berikut:**

**print(list\_buah[-1])**

**print(list\_buah[-2])**

**print(list\_buah[-3])**

**print(list\_buah[-4])**

**Output:**

**Durian**

**Melon**

**Nanas**

**Pisang**

**NB: yang perlu diperhatikan adalah: bahwa indeks negatif tidak dimulai dari 0, akan tetapi dimulai dari angka 1.**

## **Slicing List**

***Slicing* list adalah teknik untuk memotong nilai pada list. Maksudnya adalah: kita mengambil beberapa nilai dari anggota list dengan mendefinisikan indeks kiri dan indeks kanan.**

**Perhatikan contoh kode program berikut:**

**list\_buah = ['Pisang', 'Nanas', 'Melon', 'Durian']**

**print(list\_buah[0:1])**

**print(list\_buah[0:2])**

**print(list\_buah[1:3])**

**print(list\_buah[0:-1])**

**print(list\_buah[-1:-3])**

**print(list\_buah[-1:3])**

**print(list\_buah[-3:-1])**

**Jika kita eksekusi, program di atas akan menghasilkan output:**

**['Pisang']**

**['Pisang', 'Nanas']**

**['Nanas', 'Melon']**

**['Pisang', 'Nanas', 'Melon']**

**[]**

**[]**

**['Nanas', 'Melon']**

### **Keterangan**

* **parameter indeks sebelah kiri mendefinisikan awal indeks dari nilai yang akan ditampilkan.**
* **parameter indeks sebelah kanan mendefinisikan batas yang harus ditampilkan.**

### **Slicing tanpa batas**

**Kita juga bisa melakukan slicing data tanpa mendefinisikan indeks batas. Coba perhatikan contoh berikut:**

**list\_buah = ['Pisang', 'Nanas', 'Melon', 'Durian']**

**print(list\_buah[0:])**

**print(list\_buah[1:])**

**print(list\_buah[2:])**

**print(list\_buah[3:])**

**print(list\_buah[:0])**

**print(list\_buah[:1])**

**print(list\_buah[:2])**

**print(list\_buah[:3])**

**print(list\_buah[:4])**

**Kode program di atas akan menampilkan output sebagai berikut:**

**['Pisang', 'Nanas', 'Melon', 'Durian']**

**['Nanas', 'Melon', 'Durian']**

**['Melon', 'Durian']**

**['Durian']**

**[]**

**['Pisang']**

**['Pisang', 'Nanas']**

**['Pisang', 'Nanas', 'Melon']**

**['Pisang', 'Nanas', 'Melon', 'Durian']**

## **Mengubah data di dalam list**

**Pada awal pembahasan, dikatakan bahwa list adalah tipe data yang bersifat *changable* alias bisa diubah.**

**Bagaimana cara mengubah data di dalam list?**

**Caranya mudah, seperti mengubah nilai variabel pada umumnya. Perhatikan contoh berikut:**

**list\_buah = ['Pisang', 'Nanas', 'Melon', 'Durian']**

**print(list\_buah)**

***# ubah data pertama***

**list\_buah[0] = 'Jeruk'**

**print(list\_buah)**

***# ubah data terakhir***

**list\_buah[-1] = 'Mangga'**

**print(list\_buah)**

**Output:**

**['Pisang', 'Nanas', 'Melon', 'Durian']**

**['Jeruk', 'Nanas', 'Melon', 'Durian']**

**['Jeruk', 'Nanas', 'Melon', 'Mangga']**

### **Kita juga bisa mengubah data dalam range**

**Di dalam python, kita juga bisa mengubah data dalam *range* tertentu secara sekaligus. Caranya tidak jauh berbeda dengan apa yang telah kita pelajari pada poin *slicing data list*.**

**Pada kode program di atas, tambahkan lagi kode program berikut:**

***# ubah data dalam range***

**list\_buah[1:3] = ['Naga', 'Pepaya']**

**print(list\_buah)**

**Maka kita akan mendapati bahwa nilai Nanas dan Melon akan berubah menjadi Naga dan Pepaya.**

**Output:**

**['Jeruk', 'Naga', 'Pepaya', 'Mangga']**

## **Menambah item ke dalam list**

**Setelah kita mengubah data pada list, sekarang kita akan mencoba untuk menambahkan sebuah data baru ke dalam list.**

### **Menambah data di belakang**

**Yang pertama, kita bisa menggunakan fungsi append(). Fungsi ini menerima satu parameter, yang mana parameter tersebut akan dimasukkan sebagai nilai baru pada list, dan nilai baru tersebut berada pada akhir item.**

**list\_buah = ['Jeruk', 'Naga', 'Pepaya', 'Mangga']**

**print(list\_buah)**

***# tambah data di belakang list***

**list\_buah.append('Sirsak')**

**print(list\_buah)**

### **Menambah data di depan**

**Selain fungsi append(), kita juga bisa menambahkan item ke dalam list dengan menggunakan fungsi insert(). Fungsi insert ini menerima dua buah parameter:**

1. **Parameter pertama untuk mendefinisikan posisi indeks dari data yang akan dimasukkan**
2. **Parameter kedua untuk mendefinisikan nilai yang akan dimasukkan ke dalam list**

**Berikut ini contoh untuk memasukkan nilai Jambu ke dalam list\_buah pada indeks 0.**

***# tambah data di awal list***

**list\_buah.insert(0, 'Jambu')**

**print(list\_buah)**

### **Menambah data di mana pun**

**Tidak hanya terbatas indeks 0, kita juga bisa memasukkan nilai pada indeks berapa pun pada list.**

***# tambah data di index mana pun dalam list***

**list\_buah.insert(2, 'Manggis')**

**print(list\_buah)**

**Jika 3 potongan kode program di atas dijalankan, maka kita akan mendapatkan output seperti berikut:**

**['Jeruk', 'Naga', 'Pepaya', 'Mangga']**

**['Jeruk', 'Naga', 'Pepaya', 'Mangga', 'Sirsak']**

**['Jambu', 'Jeruk', 'Naga', 'Pepaya', 'Mangga', 'Sirsak']**

**['Jambu', 'Jeruk', 'Manggis', 'Naga', 'Pepaya', 'Mangga', 'Sirsak']**

## **Menghapus item dari list**

**Untuk menghapus item dari list, kita bisa menggunakan dua buah fungsi; fungsi pop() dan fungsi remove(), kita juga bisa menggunakan *statement* del.**

### **Menghapus item dengan fungsi pop()**

**Fungsi pop() akan mengambil item terakhir dari sebuah list, lalu menghapusnya. Karena ia juga “mengambil”, maka kita bisa menyimpan hasil kembalian dari fungsi pop() ke dalam sebuah variabel.**

**Perhatikan contoh berikut:**

**list\_angka = [1, 2, 3, 4, 5]**

**print(list\_angka)**

***# hapus satu angka di belakang***

**angka\_yang\_terhapus = list\_angka.pop()**

**print('angka yang terhapus:', angka\_yang\_terhapus)**

**print(list\_angka)**

**Output:**

**[1, 2, 3, 4, 5]**

**angka yang terhapus: 5**

**[1, 2, 3, 4]**

### **Menghapus dengan fungsi remove()**

**Selanjutnya adalah fungsi remove(). Fungsi ini akan menghapus data yang memiliki nilai yang sama dengan parameter yang dimasukkan. Perhatikan contoh berikut:**

**list\_buah = ['Mangga', 'Jambu', 'Jeruk', 'Jambu']**

**print(list\_buah)**

***# hapus item pertama dengan nilai 'Jambu'***

**list\_buah.remove('Jambu')**

**print(list\_buah)**

**Output:**

**['Mangga', 'Jambu', 'Jeruk', 'Jambu']**

**['Mangga', 'Jeruk', 'Jambu']**

### **Menghapus dengan statement del**

**Selanjutnya, kita juga bisa menghapus item pada list dengan menggunakan statement del. Dengan statement ini, kita bisa menghapus indeks berapa pun dari item list.**

**Perhatikan contoh berikut:**

**print('\n' \* 2)**

**list\_buah = ['Mangga', 'Jambu', 'Jeruk', 'Jambu']**

**print(list\_buah)**

**del list\_buah[1]**

**print(list\_buah)**

**del list\_buah[0:2]**

**print(list\_buah)**

**Output:**

**['Mangga', 'Jambu', 'Jeruk', 'Jambu']**

**['Mangga', 'Jeruk', 'Jambu']**

**['Jambu']**

## **Menggabungkan dua buah list atau lebih**

**Berikutnya hal umum yang biasa kita lakukan dengan list adalah: menggabungkan dua buah list (atau lebih) menjadi satu kesatuan.**

**Bisa jadi kita memiliki 3 list berikut:**

**a = [1, 2, 3]**

**b = ['a']**

**c = [True, 'b', False]**

**Kita bisa dengan mudah menggabungkan ketiganya menggunakan operator +.**

**listBaru = a + b + c**

**print(listBaru)**

**Program di atas akan menghasilkan output:**

**[1, 2, 3, 'a', True, 'b', False]**

## **Mengurutkan data**

**Terakhir tapi bukan yang paling akhir, kita bisa mengurutkan data list pada python dengan memanggil fungsi <list>.sort().**

**Perhatikan contoh berikut:**

**list\_buah = ['Mangga', 'Jeruk', 'Zaitun', 'Apel', 'Durian']**

**print(list\_buah)**

***# urutkan secara ascending***

**list\_buah.sort()**

**print(list\_buah)**

***# membalikkan posisi item list (tidak harus berurut)***

**list\_buah.reverse()**

**print(list\_buah)**

**Output:**

**['Mangga', 'Jeruk', 'Zaitun', 'Apel', 'Durian']**

**['Apel', 'Durian', 'Jeruk', 'Mangga', 'Zaitun']**

**['Zaitun', 'Mangga', 'Jeruk', 'Durian', 'Apel']**

## **Fungsi-fungsi bawaan list**

**Masih ada banyak sekali fungsi-fungsi list pada python yang belum kita bahas.**

**Secara umum, berikut ini di antara fungsi-fungsi list yang bisa kita manfaatkan untuk menyelesaikan berbagai macam permasalahan.**

| **Nama** | **Keterangan** |
| --- | --- |
| **append()** | **Menambahkan elemen baru pada list** |
| **clear()** | **Menghapus semua item pada list** |
| **copy()** | **Mengembalikan hasil duplikat dari list** |
| **count()** | **Mengembalikan jumlah item pada list sesuai yang didefinisikan** |
| **index()** | **Mengembalikan indeks pertama dari item yang sudah didefinisikan** |
| **insert()** | **Menambahakn item baru pada list pada posisi tertentu** |
| **pop()** | **Menghapus item terakhir pada list, atau juga bisa menghapus item pada posisi yang didefinisikan** |
| **remove()** | **Hapus item pada list sesuai dengan nilai yang didefinisikan** |
| **reverse()** | **Membalikkan posisi tiap item pada list** |
| **sort()** | **Mengurutkan list** |

# **Perbedaan List, Tuple, dan Set 🐍**

## **Perbedaan Secara Pengertian**

**Kita mulai dulu dari segi pengertian.**

**Secara umum, semua tipe data di atas merupakan tipe data kolektif. Artinya: ia bisa menyimpan lebih dari satu nilai dalam satu variabel sekaligus.**

**Bedanya:**

* **List adalah tipe data kolektif yang bersifat *mutable* dan *ordered*, alias bisa diedit-edit dan bisa diakses via indeks.**
* **Tuple adalah tipe data kolektif yang bersifat *immutable* dan *ordered*. Dalam artian ia sama dengan list, hanya saja tuple tidak bisa diedit-edit.**
* **Sedangkan Set adalah tipe data kolektif yang bersifat *unique*, *unordered*, dan *unchangeable* di mana semua nilainya harus unik, dan ia tidak bisa diakses via indeks (karena tidak berurut), dan dia tidak bisa diedit-edit (akan tetapi bisa ditambah dan dihapus).**

**Perhatikan tabel berikut untuk lebih jelasnya:**

| **Sifat** | **List** | **Tuple** | **Set** |
| --- | --- | --- | --- |
| **Ordered (Bisa diakses via indeks)** | **✅** | **✅** | **❌** |
| **Mutable (Bisa diubah via indeks)** | **✅** | **❌** | **❌** |
| **Changeable (Bisa ditambah anggota baru atau dihapus)** | **✅** | **❌** | **✅** |
| **Unique (Semua anggota harus unik)** | **❌** | **❌** | **✅** |

## **Perbedaan Sintaks Pembuatan**

**Masing-masing tipe data list, tuple, dan set memiliki sintaks pembuatan tersendiri. Di mana list menggunakan kurung siku ([]), tuple menggunakan kurung biasa (()) atau bisa juga tanpa kurung, dan set menggunakan tanda kurung kurawal ({}).**

**Perhatikan tabel berikut untuk lebih jelasnya:**

| **Tipe Data** | **Simbol** | **Contoh** |
| --- | --- | --- |
| **list** | **[]** | **a = [1, 2, 3]** |
| **tuple** | **()** | **a = (1, 2, 3) atau a = 1, 2, 3** |
| **set** | **{}** | **a = {1, 2, 3}** |

## **Perbedaan Cara Mengakses Nilai**

**Tipe data yang memiliki sifat “*ordered*” alias terurut (list dan tuple), cara mengakses nilainya bisa menggunakan indeks. Sedangkan variabel yang bersifat “*unordered*” alias tidak terurut seperti set, ia tidak bisa diakses secara eksplisit.**

**Contoh list dan tuple:**

**list\_a = [1, 2, 3]**

**print(list\_a[0])**

**print(list\_a[1])**

**print(list\_a[2])**

**tuple\_a = 10, 20, 30**

**print(tuple\_a[0])**

**print(tuple\_a[1])**

**print(tuple\_a[2])**

**Output:**

**1**

**2**

**3**

**10**

**20**

**30**

**Contoh set:**

**set\_b = { 15, 16, 17 }**

**print(set\_b[0])**

**print(set\_b[1])**

**print(set\_b[2])**

**Error:**

**Traceback (most recent call last):**

**File "<stdin>", line 1, in <module>**

**TypeError: 'set' object is not subscriptable**

## **Perbedaan Slicing Data**

**Untuk list dan tuple, tidak ada perbedaan dalam hal *slicing* data –karena keduanya bersifat *ordered*.**

**Ada pun set, ia tidak mendukung fitur *slicing* data.**

**>>> list\_a = [1, 2, 3, 4]**

**>>> tuple\_a = tuple(list\_a)**

**>>>**

**>>> list\_a[2:4]**

**[3, 4]**

**>>> tuple\_a[2:]**

**(3, 4)**

**>>>**

## **Perbedaan Ubah Data**

**Untuk ubah data ada dua tipe:**

1. **ubah data pada indeks tertentu (yang sudah dimasukkan sebelumnya)**
2. **ubah data dalam artian menambahkan data baru atau menghapus data yang sudah ada**

**Dalam dua tipe di atas, hanya tipe data list saja yang bisa melakukan semuanya.**

**Sedangkan set hanya bisa mengedit data untuk tipe yang kedua.**

**Ada pun tuple, maka ia tidak bisa melakukan edit data sama sekali, baik tipe yang pertama mau pun tipe yang kedua.**

**list\_a = [1, 2, 3]**

**print(f'Sebelum: {list\_a}')**

***# edit data yang paling tengah***

**list\_a[1] = 99**

**print(f'Sesudah: {list\_a}')**

**Output:**

**Sebelum: [1, 2, 3]**

**Sesudah: [1, 99, 3]**

## **Perbedaan Tambah Data**

**Untuk mengedit data yang dalam artian “menambah data”, maka hanya ada dua tipe data yang bisa melakukannya; yaitu list dan set. Ada pun tuple tidak bisa melakukannya.**

**Kita bisa menambahkan nilai baru kepada suatu list dengan memanggil fungsi list.append(), dan kita bisa melakukan hal yang sama dengan set menggunakan fungsi set.add().**

**Perhatikan contoh berikut:**

**list\_a = [1]**

**set\_a = {1, 39}**

**print('[Sebelum]')**

**print(f' list\t: {list\_a}')**

**print(f' set\t: {set\_a}')**

**list\_a.append(10)**

**list\_a.append(50)**

**set\_a.add(50)**

**set\_a.add(100)**

**print('[Sesudah]')**

**print(f' list\t: {list\_a}')**

**print(f' set\t: {set\_a}')**

**Output:**

**[Sebelum]**

**list : [1]**

**set : {1, 39}**

**[Sesudah]**

**list : [1, 10, 50]**

**set : {1, 50, 100, 39}**

## **Perbedaan Hapus Data**

**Sama seperti sebelumnya, hanya tipe data list dan set saja yang bisa melakukan hapus data.**

**Keduanya sama-sama menggunakan fungsi .remove().**

**Perhatikan contoh berikut:**

**list\_a = [1, 2, 3, 4]**

**set\_a = {1, 2, 3, 4}**

**print('[Sebelum]')**

**print(f' list\t: {list\_a}')**

**print(f' set\t: {set\_a}')**

**list\_a.remove(2)**

**list\_a.remove(4)**

**set\_a.remove(2)**

**set\_a.remove(4)**

**print('[Sesudah]')**

**print(f' list\t: {list\_a}')**

**print(f' set\t: {set\_a}')**

**Output:**

**[Sebelum]**

**list : [1, 2, 3, 4]**

**set : {1, 2, 3, 4}**

**[Sesudah]**

**list : [1, 3]**

**set : {1, 3}**

## **Perbedaan Operator Keanggotaan**

**Berikutnya adalah penggunaan** [**operator keanggotaan**](https://jagongoding.com/python/dasar/operator/#operator-keanggotaan) **in atau pun not in.**

**Dua operator ini bisa kita gunakan baik pada list, tuple, dan set.**

**Perhatikan contoh berikut:**

**>>> list\_a = [1, 2, 3]**

**>>> tuple\_a = 1, 2, 3**

**>>> set\_a = {1, 2, 3}**

**>>>**

**>>> 1 in list\_a**

**True**

**>>> 2 not in tuple\_a**

**False**

**>>> 3 in set\_a**

**True**

**>>>**

## **Perbedaan Ketika Dilakukan Perulangan**

**Yang terakhir tapi bukan yang paling akhir, kita bisa melakukan** [**perulangan for**](https://jagongoding.com/python/dasar/perulangan-for) **untuk ketiga tipe data tersebut.**

**Akan tetapi perlu diperhatikan bahwa tipe data set bertipe unordered, artinya bahwa data yang kita definisikan bisa jadi urutannya berbeda dengan data yang disimpan oleh sistem.**

**Perhatikan contoh berikut:**

**list\_a = [3, 10, 1]**

**tuple\_a = 3, 10, 1**

**set\_a = {3, 10, 1}**

**print('[LIST]')**

**for item in list\_a:**

**print(item)**

**print('\n[TUPLE]')**

**for item in tuple\_a:**

**print(item)**

**print('\n[SET]')**

**for item in set\_a:**

**print(item)**

**Output:**

**[LIST]**

**3**

**10**

**1**

**[TUPLE]**

**3**

**10**

**1**

**[SET]**

**1**

**10**

**3**

## **Kesimpulan**

**Tiga (dari 4) tipe data kolektif pada python –yaitu list, set dan tuple– memiliki berbagai macam kesamaan dan juga berbagai macam perbedaan.**

**Masing-masing sifat pada tiap tipe data tersebut memberikan kita keleluasaan untuk memilih mana tipe data yang pas untuk suatu kasus, dan mana tipe data yang cocok untuk suatu kasus yang lain.**

**Source :** [**Jago Ngoding**](https://jagongoding.com/)