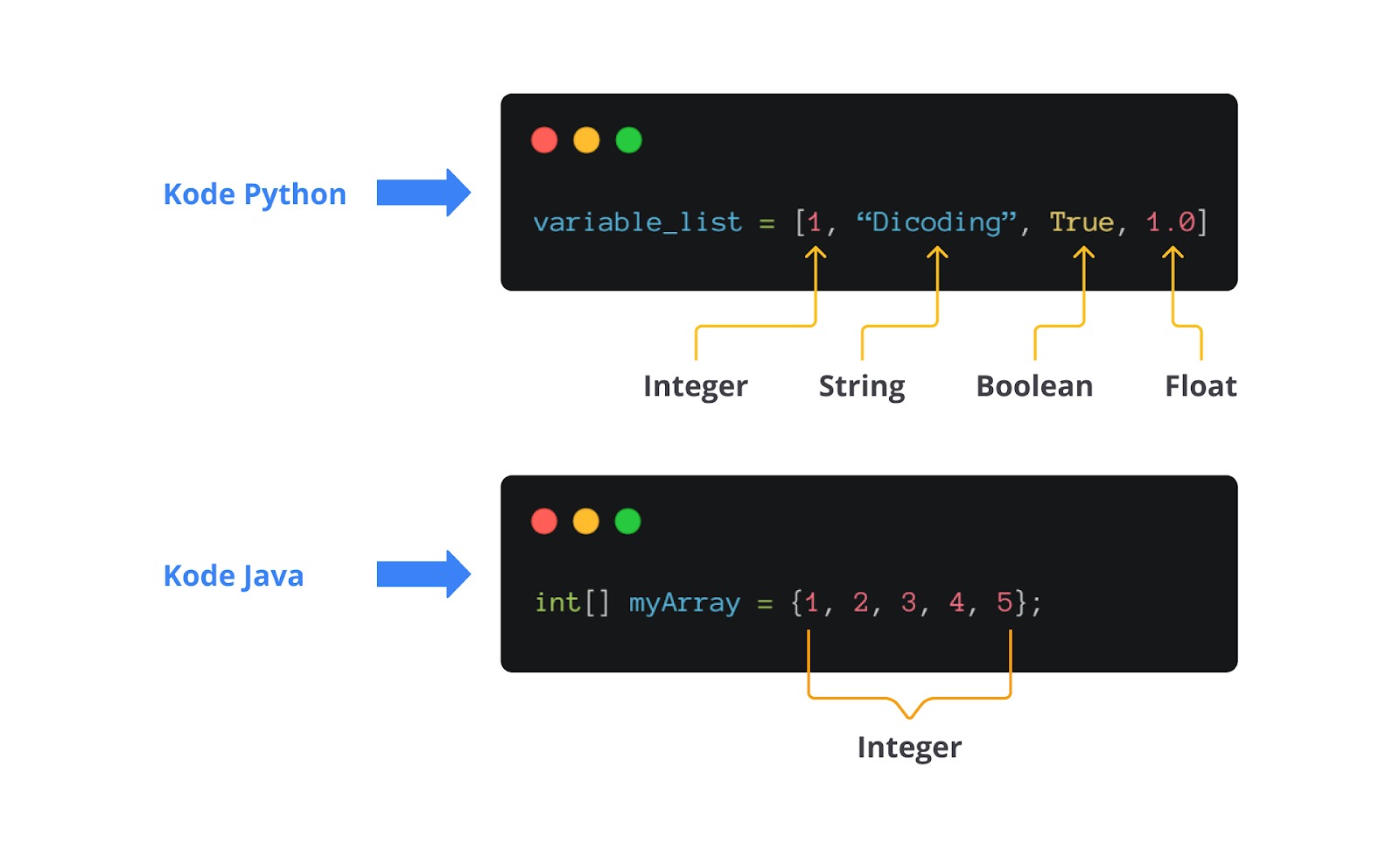
**Fundamental Array**

Python tidak memiliki tipe data array yang sering digunakan dalam bahasa pemrograman lain. Sebaliknya, Python memiliki tipe data list yang dapat dikatakan mirip, tetapi tak sama dengan array.

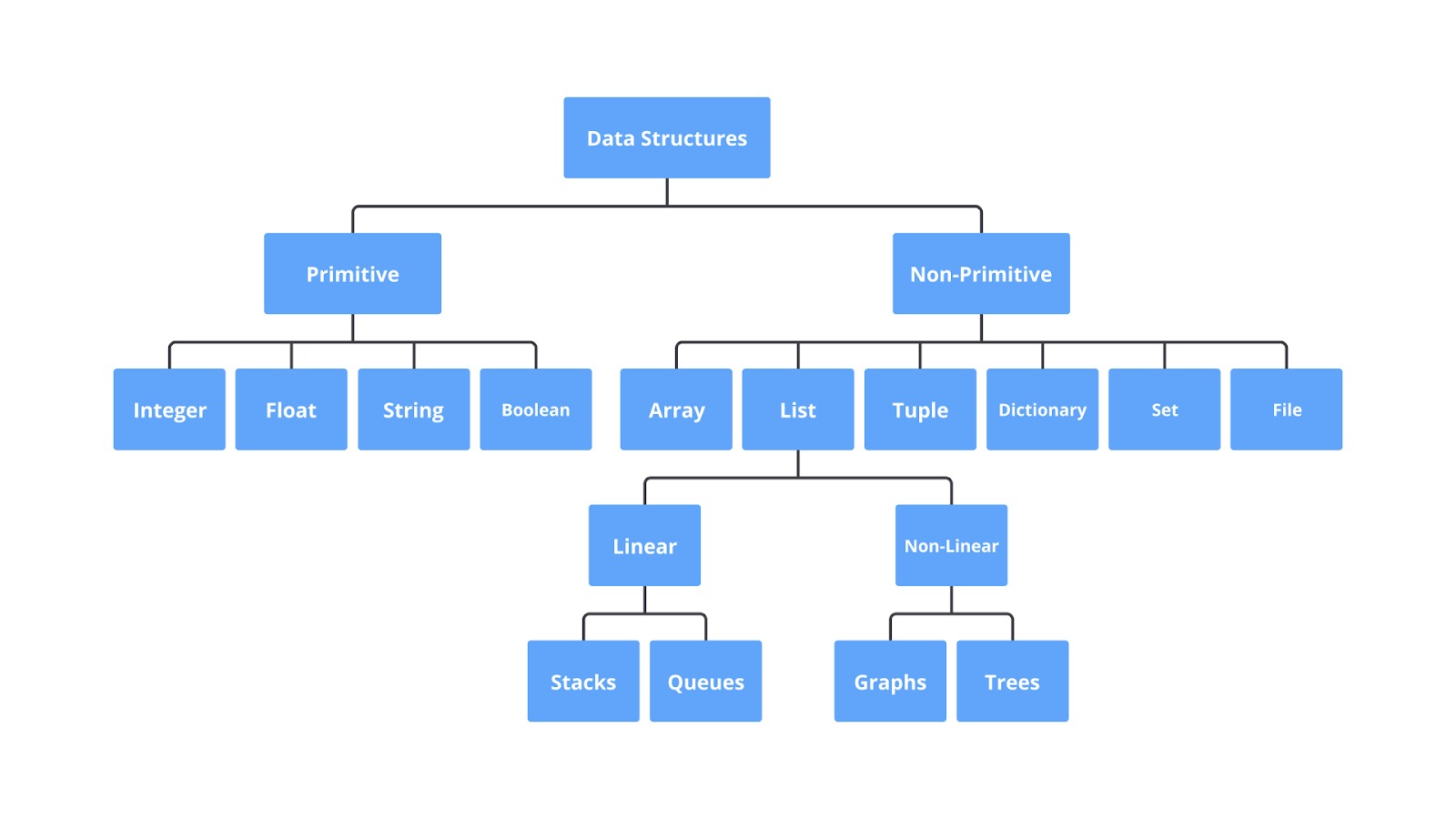
Perbedaan yang menonjol adalah cara array menyimpan nilai sangat berbeda dengan list. Pada array, nilai di dalamnya harus memiliki tipe data yang sama. Namun, pada list, nilai di dalamnya tidak harus memiliki tipe data yang sama.



Array bukan hanya sebuah tipe data, melainkan salah satu tipe struktur data berjenis linear. *Array*merupakan kata dalam bahasa Inggris yang jika diterjemahkan ke bahasa Indonesia memiliki arti "sebuah kelompok besar yang terdiri dari beberapa hal atau orang". Arti ini mirip dengan array atau tipe data list dalam Python, sebuah kelompok besar yang terdiri dari beberapa nilai atau data. Lalu, apa arti dari struktur data itu sendiri?

Struktur data adalah cara untuk mengatur dan menyimpan data sehingga data-data tersebut dapat diakses dan bekerja secara efisien. Dengan adanya struktur data, setiap data yang disimpan memiliki hubungan satu sama lain dan kita dapat beroperasi dengan setiap data tersebut.

Ketika mempelajari materi tipe data pada modul "Berinteraksi dengan Data", sebenarnya Anda telah mempelajari struktur data yang beragam jenisnya. Baik tipe data primitif maupun tipe data collection yang telah dibahas sebelumnya termasuk jenis struktur data Python.



Dari sini, kita harus bisa menyamakan persepsi bahwa array dan list merupakan hal yang berbeda dalam Python. Kendati demikian, Anda bisa menggunakan list sebagai array dalam Python.

Jika Anda benar-benar ingin menggunakan array, Anda pun bisa mendeklarasikan array dalam Python dengan menggunakan library atau modul Python. Salah satunya modul bernama "array".

Library merupakan sekumpulan kode yang telah dibuat oleh developer atau programmer dan disediakan kepada pengguna lain agar dapat digunakan ulang dalam pengembangan program atau perangkat lunak. Adapun modul merupakan file yang berisikan kode Python dan dapat digunakan kembali oleh programmer lainnya. Anda akan mempelajari library dan modul pada Python lebih jauh nanti.

import array

x = array.array("i",[1, 2, 3, 4, 5])

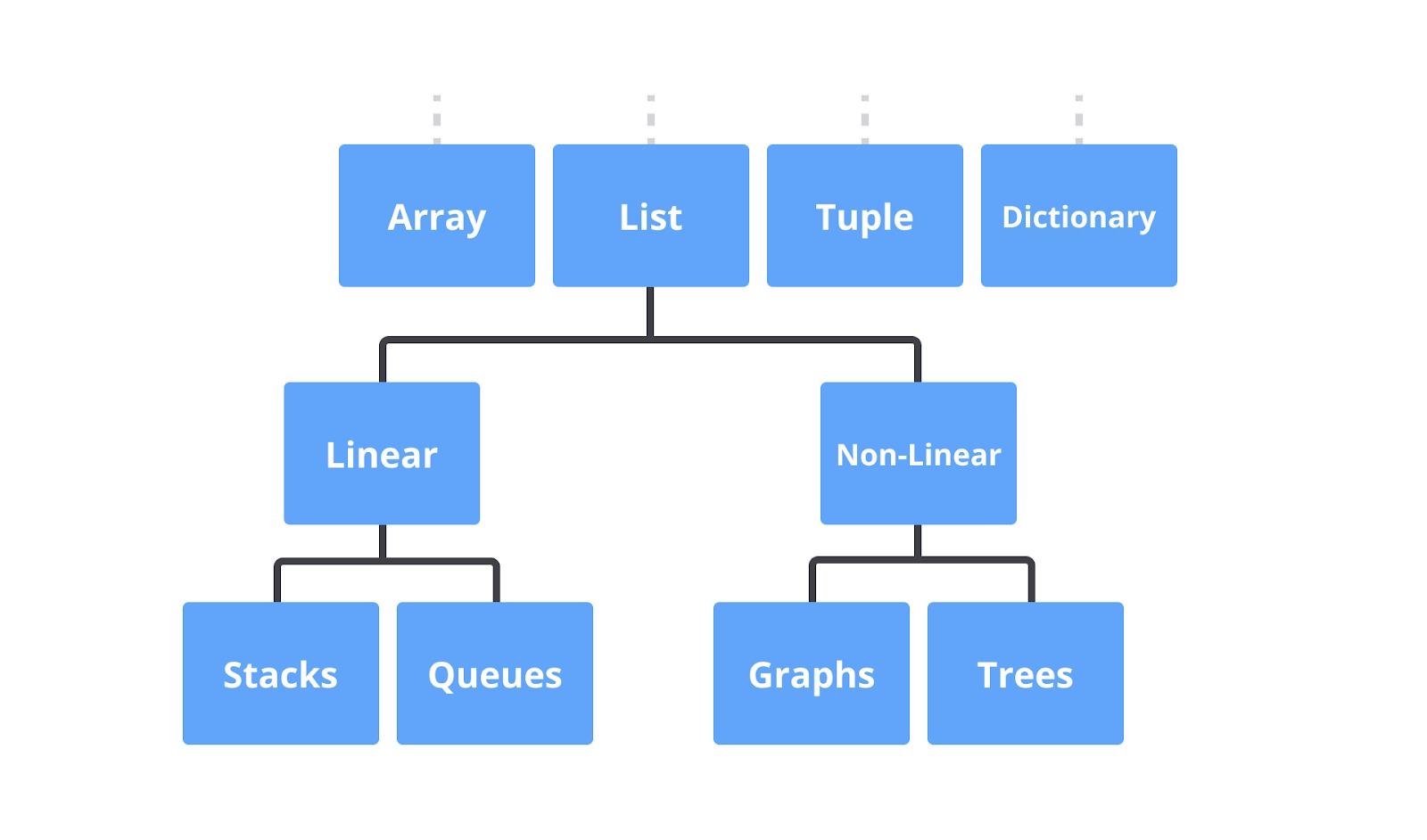
print(x)

Pada kode di atas, kita melakukan impor module array dengan memberikan sintaks "import array". Dengan mengimpor module, sekarang kita mempunyai beragam kode baru yang dapat digunakan. Contohnya fungsi "array()" yang digunakan untuk membuat array.

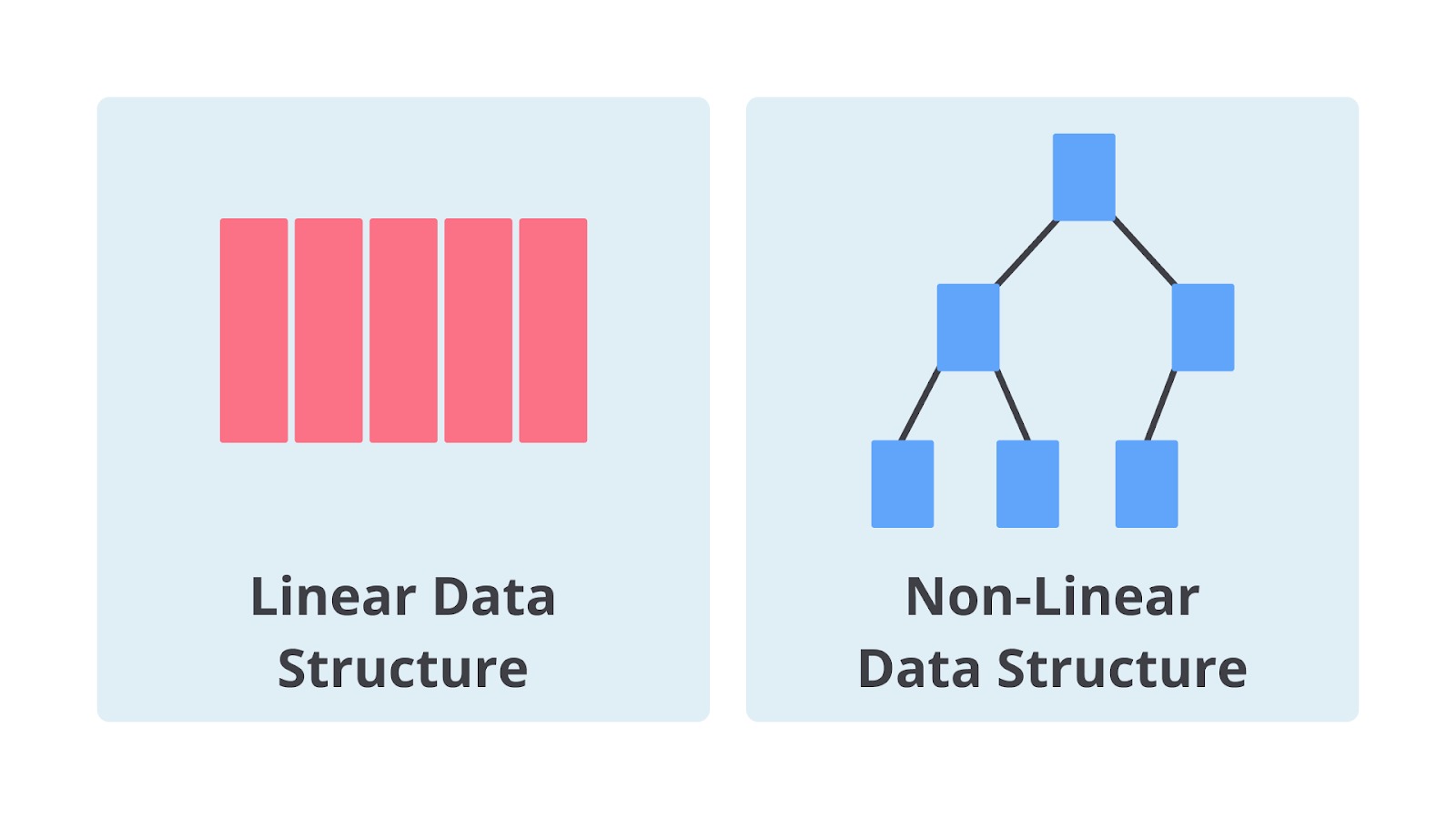
Pada contoh di atas, kita membuat array bertipe integer dengan menyatakan "i" sebelum array. Sekarang, coba Anda ubah nilai array "[1, 2, 3, 4, 5]" menjadi "[1, 2, 3, 4, 5, 'Dicoding']". Apa yang terjadi? Jika yang terjadi adalah **error**, hal ini disebabkan karena nilai atau elemen dalam array harus bertipe sama atau identik.

**Catatan**: Kelas ini akan menggunakan "list" sebagai "array". Anda bisa asumsikan ke depannya bahwa kata "list" dalam kelas ini juga dapat diartikan sebagai "array".

Dalam materi ini, kita akan menggunakan list sebagai struktur data array dari awal hingga materi selesai. Mari masuk kembali ke pembahasan list.

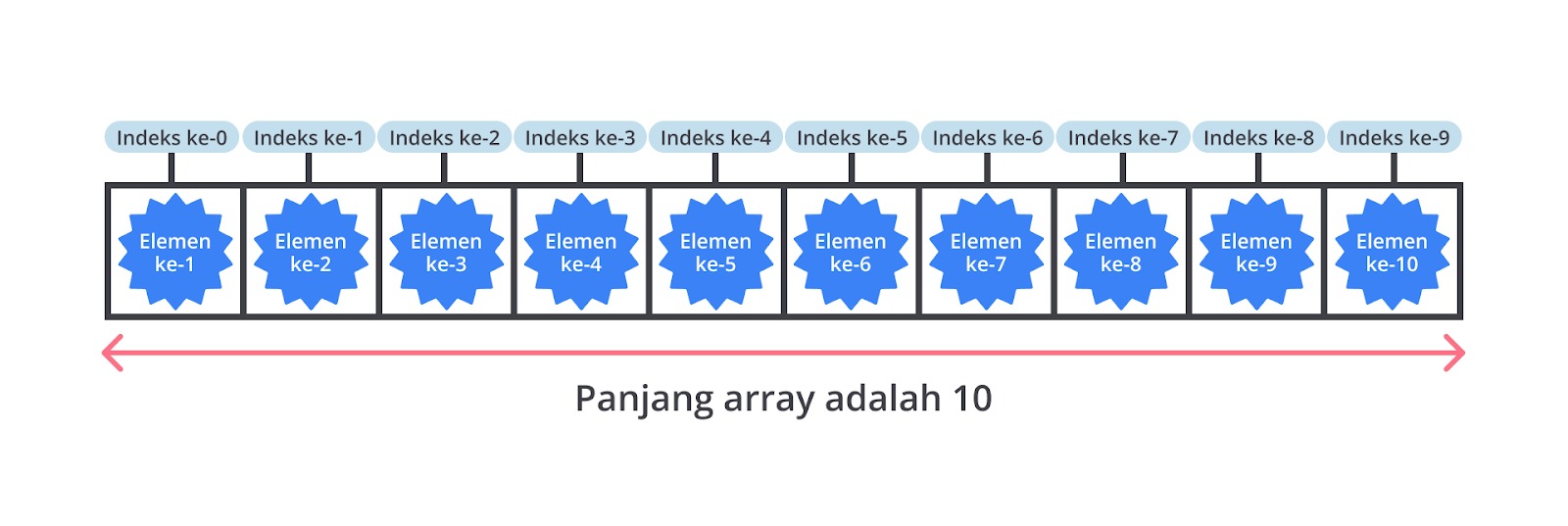


Sebagaimana yang sudah ditunjukkan pada gambar mengenai struktur data, list dapat dibagi kembali menjadi struktur data linear dan non-linear.



Struktur data linear adalah jenis struktur data yang elemen-elemen nilai di dalamnya disusun secara berurutan atau linear. Sebaliknya, struktur data non-linear merupakan jenis struktur data yang elemen-elemen nilai di dalamnya tidak disusun secara linear.

Array adalah salah satu jenis dari struktur data linear. Dalam konteks ini, array terdiri dari **kumpulan elemen bertipe data sama**dengan indeks yang berurutan atau linear.



Mari kita bedah satu per satu mengenai array berdasarkan gambar di atas.

1. Indeks: Posisi atau nomor yang digunakan untuk mengidentifikasi elemen-elemen dalam array. Indeks selalu dimulai dari 0.
2. Element: Nilai yang berada dalam suatu indeks. Elemen selalu dimulai dari 1. Contohnya jika nilai dari indeks ke-8 adalah "Dicoding", kita bisa sebut sebagai "elemen ke-9 pada array tersebut adalah Dicoding".
3. Array length: Panjang dari suatu array. Dalam gambar tersebut, panjang array adalah 10.

Mari lihat salah satu contoh kasus saat kita perlu menggunakan array sebagai solusi yang terbaik.

"Selepas berakhirnya semester genap, para guru dari SMA Dicoding perlu merekap semua nilai ujian akhir semester. Salah satunya adalah guru matematika, guru tersebut perlu merekap nilai dari seluruh siswa yang ada di kelas IPA 1. Guru tersebut membuat program menggunakan Python untuk mempermudah proses rekap nilai."

nama\_siswa1 = 90

:

nama\_siswa10 = 60

Pada kode di atas, program yang dibuat adalah menyimpan 10 nilai dengan menggunakan 10 variabel yang berbeda. Variabel pertama dimulai dengan "nama\_siswa1" dengan nilai siswanya adalah 90. Variabel kedua berlanjut dengan konsep yang sama hingga selesai.

Namun, Anda mungkin menyadari bahwa membuat program dengan cara tersebut tidak efektif dan membuat kodenya sulit dibaca. Bahkan ini baru 10 nama siswa, bagaimana dengan 100 siswa? 1000 siswa?

siswa = [90, 100, 50, 80, 85, 45, 80, 75, 50, 60]

print(siswa)

print(siswa[0])

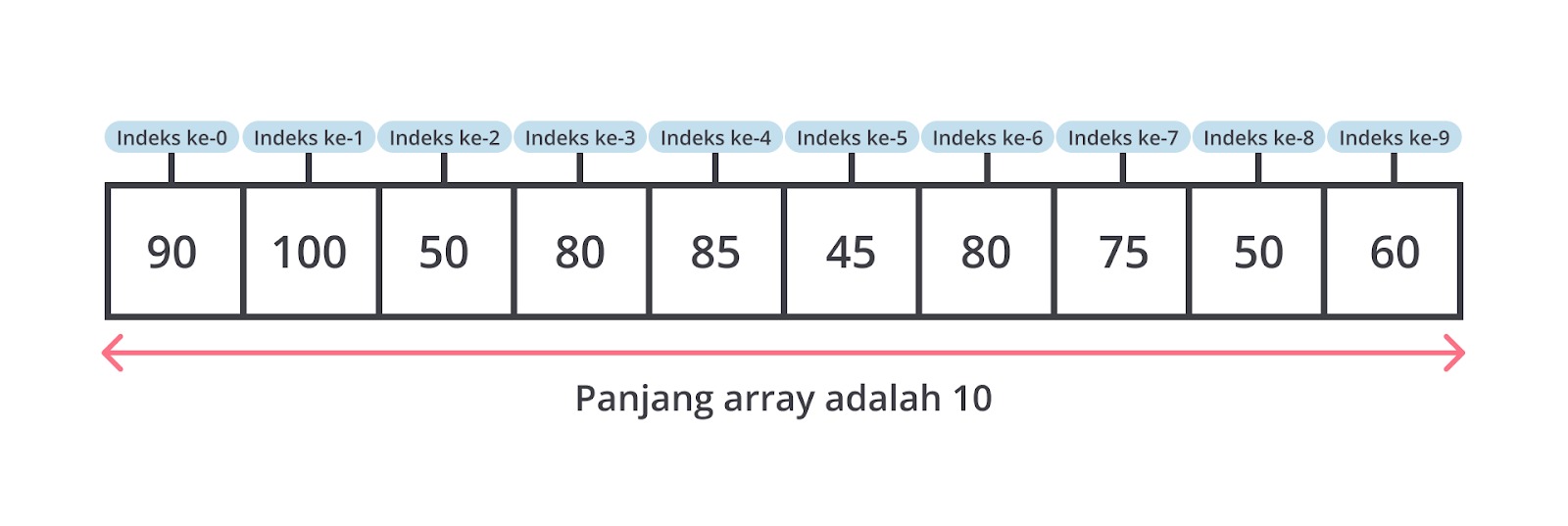
Output:

[90, 100, 50, 80, 85, 45, 80, 75, 50, 60]

90

Pada kode di atas, alih-alih membuat inisialisasi variabel yang berulang, Anda dapat membuat list untuk menyimpan seluruh nilai tersebut. Jika Anda ingin mendapatkan nilai pertama atau nilai tertentu, cukup lakukan **indexing**. Pada contoh di atas, kita menggunakan indexing untuk mengakses elemen pertama atau indeks 0.

Jika kita tarik ke konteks penyimpanan array secara teoretis, kode di atas dapat diilustrasikan seperti berikut.



Ilustrasi di atas menggambarkan kode yang sebelumnya telah kita buat. Seluruh nilai yang direkap oleh guru SMA Dicoding akan disimpan pada sebuah kotak yang disebut sebagai indeks tempat keseluruhan indeks tersebut dikelompokkan menjadi satu array.

**Implementasi Array dengan Python**

**Mendeklarasikan Array**

Pada materi sebelumnya, sudah disebutkan bahwa dalam Python kita dapat mendeklarasikan array menggunakan dua cara. Pertama dengan memanfaatkan list dan kedua menggunakan library Python.

Perlu Anda ingat, setiap elemen yang ada pada list sebetulnya disimpan pada satu memori. Jika list adalah "[1, 2, 3]", sebetulnya Anda memerintahkan memori komputer untuk menyimpan integer "1" ke dalam satu tempat memori, begitu pun integer "2" akan disimpan dalam satu tempat memori, dan seterusnya.

var\_list = [1,2,3]

for elemen in var\_list:

print(id(elemen))

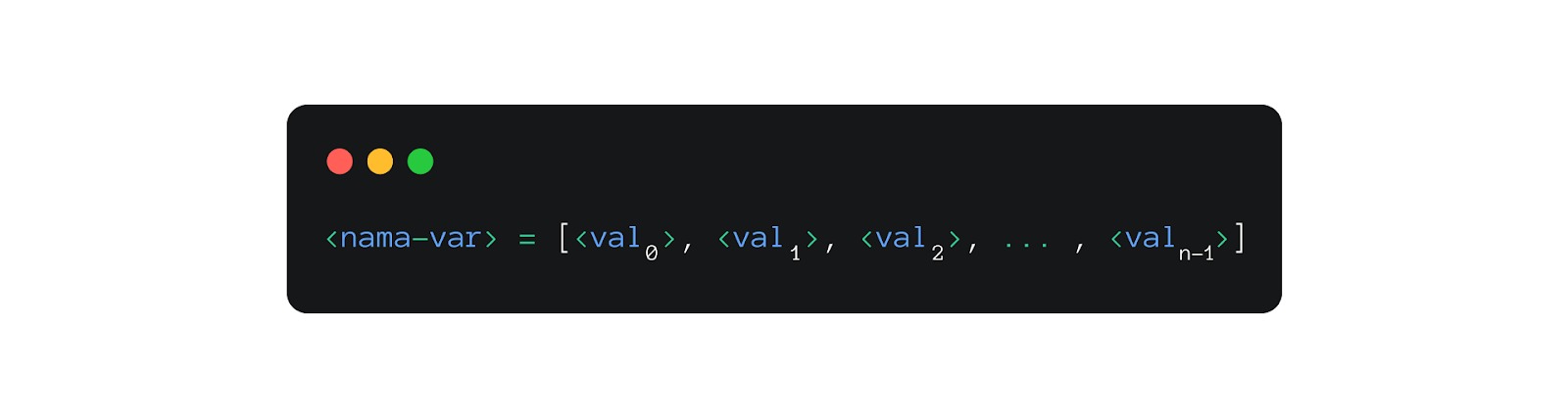
Ketika Anda menjalankan kode di atas, ia akan menghasilkan lokasi memori setiap elemen yang berada pada list. Lokasi tersebut bisa berubah jika Anda menjalankan ulang program, tetapi perhatikan bahwa semua elemen tersebut memiliki ID lokasi penyimpanan yang berbeda.

Sekarang mari lebih detailkan cara deklarasi array dalam Python menggunakan list. Ada dua cara untuk melakukan deklarasi array menggunakan list, yakni berikut.

**Mendefinisikan Isi Array**

Cara pertama adalah dengan mendeklarasikan variabel array sekaligus mendefinisikan isi array. Cara ini dilakukan jika kita sudah tahu nilai yang perlu diberikan.

Berikut adalah struktur mendeklarasikan variabel array dengan mendefinisikan isi array secara langsung.



**<nama-var>**merupakan nama variabel array yang dideklarasikan sebanyak **n** dengan elemen-elemennya adalah **<val0>, <val1>, <val2>, … , <valn-1>**. Perlu diingat bahwa elemen tersebut terurut berdasarkan indeks dari **0** hingga **n-1**.

var\_arr = [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

Pada variabel "var\_arr" kita menyimpan elemen bertipe integer dengan panjangnya adalah 10 elemen dan alamat setiap elemen array (indeks) adalah indeks ke-0 hingga 9.

**Mendefinisikan Nilai Default**

Jika tidak mengetahui nilai yang diberikan, kita dapat memberikan nilai default terlebih dahulu sebagai upaya untuk memberikan nilai awal. Umumnya, nilai default ini ditentukan karena kita tidak tahu nilai seharusnya.

Dalam prosesnya, kita bisa secara perlahan mengganti masing-masing nilai tersebut sesuai kebutuhan. Misal kita memiliki array "[0,0,0,0]", yang kemudian hari kita bisa memperbaruinya menjadi "[1,2,0,4]", dengan begitu kita bisa mengetahui bahwa indeks ke-2 pada array tersebut belum kita perbarui.

Nilai default ditentukan oleh kesepakatan bersama sesuai kebutuhan yang nilainya di luar dari rentang yang disepakati. Misalnya, tim Anda menentukan nilai dalam list harus berkisar dari 1 hingga 10. Kita bisa menyepakati "0" sebagai nilai default karena di luar jangkauan yang disepakati (1-10).

Berikut adalah struktur mendeklarasikan variabel array dengan mendefinisikan nilai default.



Jika Anda merasa familier dengan struktur tersebut, Anda benar. Struktur tersebut merupakan struktur yang sama dengan list comprehension. Anda dapat menginisialisasi variabel array dengan menggunakan list comprehension dan mendefinisikan nilai default. Pada materi List Comprehension, kita menggunakan ekspresi Namun, pada array kita menggunakan *default value* atau <default-val>.

Berikut adalah penjelasan lebih detail terkait struktur tersebut.

1. <nama-var> merupakan variabel yang Anda deklarasikan.
2. <default-val> merupakan nilai default yang Anda definisikan. Umumnya, programmer akan menggunakan nilai di luar range yang telah disepakati sebagai nilai default. Misalnya jika range nilai yang disepakati seharusnya 1 hingga 10, nilai default bisa kita definisikan dengan 0.
3. <n> merupakan ukuran panjangnya array.

var\_arr = [0 for i in range(10)]

Pada contoh di atas, kita membuat list dengan nilai default-nya adalah "0" sebanyak 10 elemen. Perhatikan bahwa <default-val> yang ada pada struktur sebelumnya diubah menjadi "0" untuk mendapatkan nilai default "0".

Dari sini, Anda dapat mengubah nilai default tersebut dengan nilai yang baru berdasarkan hasil suatu operasi. Misalkan pada contoh di bawah ini.

var\_arr = [0 for i in range(10)]

for i in range(10):

var\_arr[i] = i

Pada contoh di atas, kita membuat program untuk mengubah nilai default pada variabel array "var\_arr" dengan nilai 0 hingga 9. Output dari program tersebut adalah mengubah nilai yang awalnya adalah [0, 0, 0, 0, 0, 0, 0, 0, 0, 0] menjadi [0, 1, 2, 3, 4, 5, 6, 7, 8, 9].

**Mengakses Elemen Array**

Mengakses elemen array dalam Python tidak berbeda dengan mengakses elemen pada list. Hal ini karena kita menggunakan list sebagai "bentuk lain" dari array. Anda dapat melakukan metode indexing untuk mengakses elemen array. Berikut adalah struktur untuk melakukan hal tersebut.



**<namaVariabelArray>** merupakan nama variabel array yang sebelumnya telah Anda deklarasikan. **<indeks>** merupakan urutan indeks yang ingin Anda akses sehingga nilai atau elemen tersebut dapat diambil atau ditampilkan.

**Pemrosesan Sekuensial pada Array**

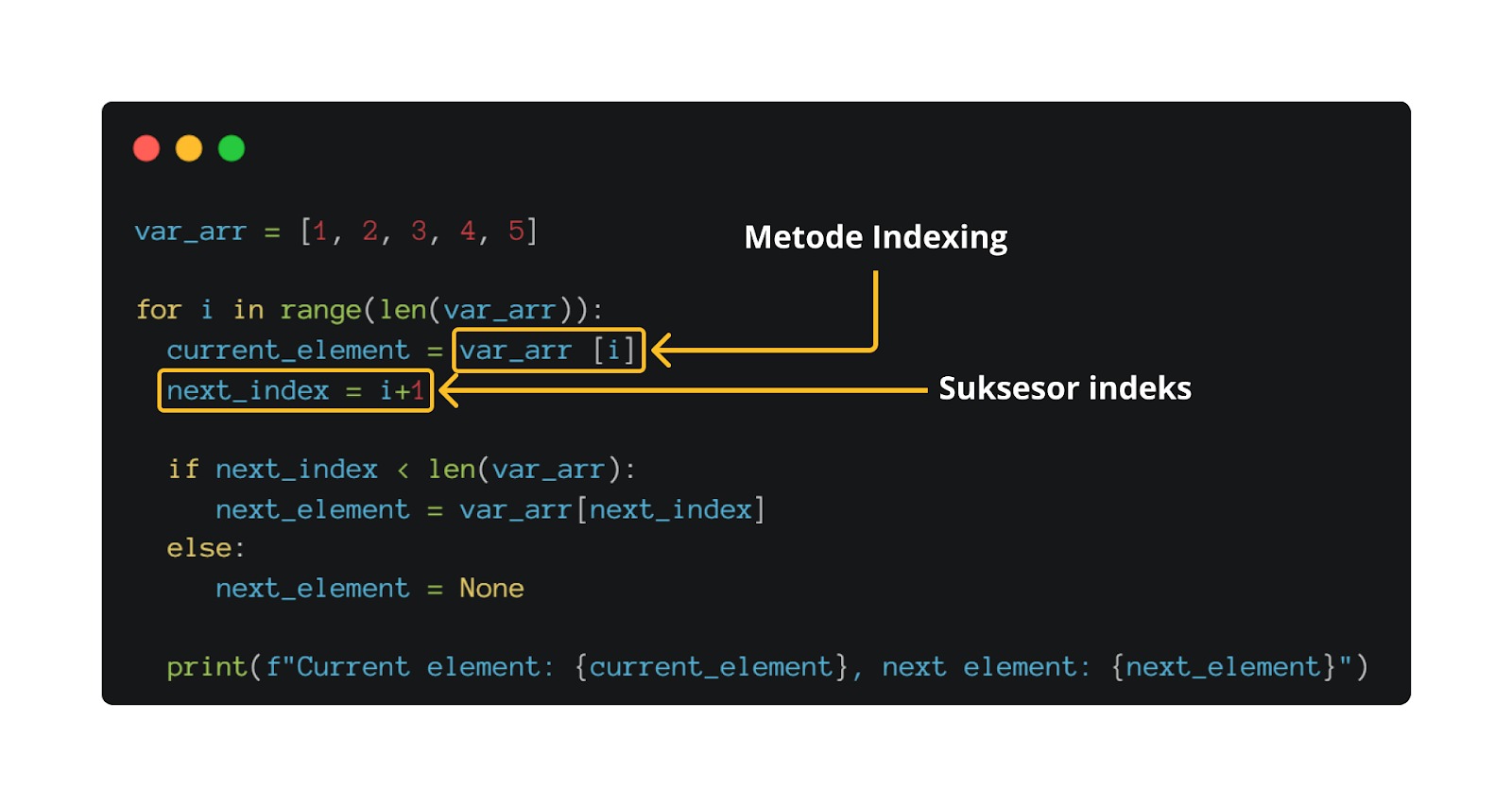
Pemrosesan array merujuk kepada operasi-operasi yang dilakukan pada elemen-elemen suatu array. Operasi ini melibatkan manipulasi hingga pengolahan elemen yang ada pada array.

Adapun pemrosesan sekuensial adalah sebuah pemrosesan setiap elemen array yang dimulai dari elemen pada indeks terkecil hingga terbesar. Pemrosesan sekuensial lebih sering menggunakan pengulangan (loop/iterasi) dalam setiap prosesnya.

Sebab pemrosesan sekuensial melibatkan semua elemen di dalamnya, ada beberapa hal yang perlu diperhatikan.

1. Setiap elemen array diakses secara langsung melalui indeksnya (metode indexing).
2. Elemen pertama (*first element*) adalah elemen array dengan indeks terkecil yang selalu dimulai dari 0.
3. Elemen selanjutnya (*next element*) dicapai melalui suksesor indeks.
4. Kondisi berhenti dicapai jika indeks yang diproses adalah indeks terbesar yang sudah terdefinisi.
5. Suatu array tidak boleh kosong, minimal memiliki satu elemen di dalamnya.

Membuat program untuk melakukan pemrosesan sekuensial array. Proses tersebut adalah mencetak setiap elemen yang berada pada variabel array "var\_arr" menggunakan perulangan loop.



Pertama-tama, kita menginisialisasi variabel "var\_arr" dengan nilai "[1, 2, 3, 4, 5]". Perulangan for digunakan untuk melakukan iterasi melalui setiap elemen array. **Variabel "i"** bertindak sebagai **indeks saat ini**yang digunakan untuk mengakses elemen dalam setiap iterasi atau perulangan.

Kemudian, setiap proses perulangan berlangsung, kita mengambil elemen saat ini menggunakan "var\_arr[i]" dan menyimpannya pada variabel "current\_element". Selanjutnya adalah mencari indeks berikutnya dengan cara menambahkan nilai "1" pada indeks saat ini atau "i".

Hasil dari operasi penjumlahan nilai "1" dengan indeks saat ini akan disimpan pada variabel "next\_index". "next\_index" berperan sebagai **"suksesor indeks"** yang merujuk pada indeks berikutnya berdasarkan indeks saat ini dengan menambahkan nilai "1".

Kemudian kita memeriksa "next\_index" berada dalam rentang indeks yang valid dalam array atau tidak. Jika iya, artinya masih ada elemen berikutnya, dan kita ambil elemen berikutnya tersebut menggunakan "var\_arr[next\_index]" serta menyimpannya dalam variabel "next\_element". Sebaliknya, jika "next\_index" tidak valid atau melebihi rentang array, artinya tidak ada elemen berikutnya sehingga kita menetapkan "next\_element" sebagai "None".

Pada langkah terakhir, kita mencetak nilai "current\_element" dan "next\_element" untuk menunjukkan perbedaan antara elemen sekarang dan selanjutnya.

Mencetak setiap elemen array menggunakan perulangan adalah satu di antara banyaknya contoh-contoh persoalan pemrosesan sekuensial pada array. Contoh lain dari pemrosesan array adalah berikut.

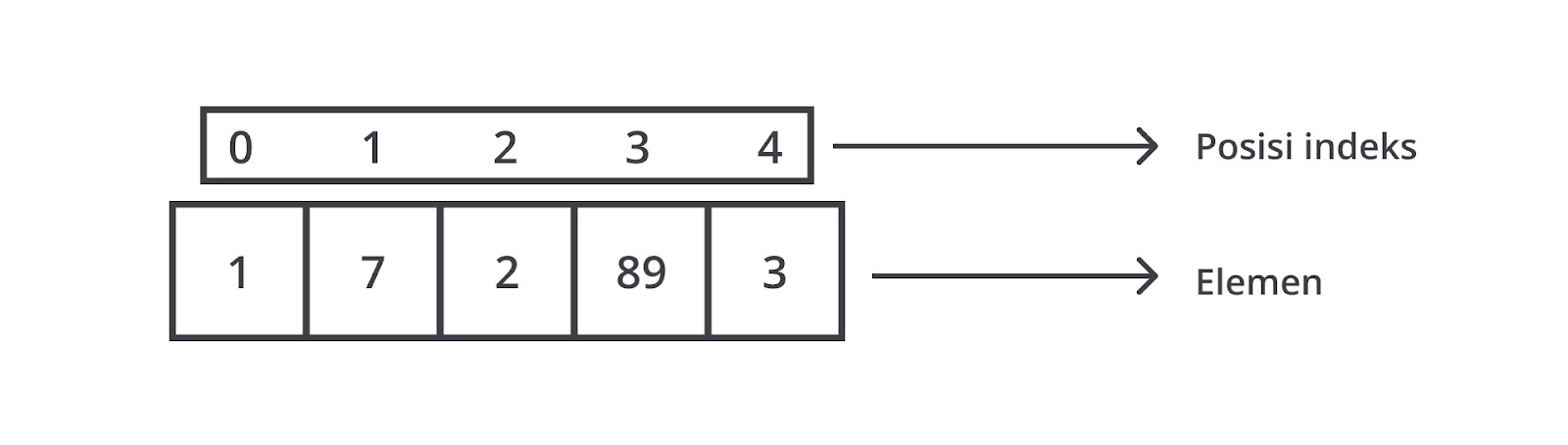
1. Mengisi array secara sekuensial.
2. Menghitung nilai rata-rata elemen array.
3. Mengalikan elemen array dengan suatu nilai.
4. Mencari nilai terbesar atau terkecil pada array.
5. Mencari indeks letak suatu nilai ditemukan pertama kali dalam array, dan sebagainya.

**Latihan Array**

Pada materi sebelumnya, kita telah memahami bahwa array adalah salah satu jenis dari struktur data linear. Array mengumpulkan elemen-elemen berdasarkan indeks secara berurutan atau linear.

Pemrosesan array merujuk pada operasi-operasi yang dilakukan pada elemen-elemen suatu array. Salah satu operasinya adalah pemrosesan sekuensial yang merupakan sebuah pemrosesan setiap elemen array, dimulai dari elemen pada indeks terkecil hingga indeks terbesar.

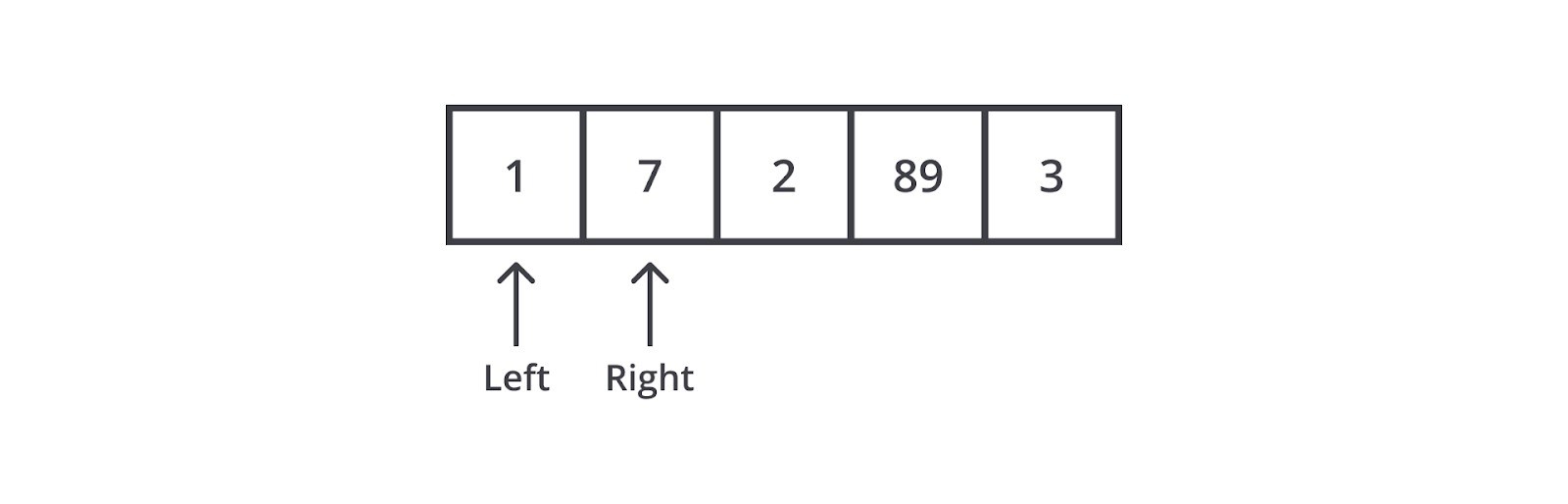
Dari sekian banyaknya contoh pemrosesan sekuensial pada array, mari kita pelajari salah satunya, yakni **mencari nilai terbesar dalam array**. Mari kita asumsikan memiliki sebuah array seperti pada gambar berikut.



Kita memiliki array yang beranggotakan nilai integer dengan elemen indeks ke-0 adalah 1, elemen indeks ke-1 adalah 7, elemen indeks ke-2 adalah 2, elemen indeks ke-3 adalah 89, elemen indeks ke-4 adalah 3.

Kita akan mencari nilai atau elemen terbesar dari array tersebut menggunakan algoritma *two pointers*. Algoritma adalah serangkaian langkah-langkah terstruktur yang dirancang untuk menyelesaikan suatu masalah atau mencapai suatu tujuan. Dalam hal ini, tujuan yang ingin dicapai adalah mencari nilai terbesar pada array.

Algoritma two pointers adalah algoritma yang memiliki pendekatan dengan cara memanipulasi atau memproses urutan data menggunakan dua penanda atau dua *pointer* secara bersamaan. Kedua pointer tersebut bisa kita sebut sebagai "left" dan "right".



Untuk memahami algoritma ini, perhatikan beberapa informasi berikut.

1. Pointer "left" akan berada pada indeks pertama dan menyatakan bahwa pointer "left" selalu menunjukkan nilai terbesar dalam array.
2. Pointer "right" akan selalu berada pada elemen selanjutnya dan membandingkannya dengan elemen pointer "left".

Mari kita bedah satu per satu setiap langkah-langkahnya, simak penjelasan berikut.

1. Pertama, kita memulai dengan dua pointer: pointer "left" pada elemen pertama (1) dan pointer "right" pada elemen berikutnya (7). Kita membandingkan nilai 7 dengan nilai 1. Sebab 7 lebih besar dari 1, kita mengganti nilai pointer "left" dari 1 menjadi 7.
2. Sekarang, pointer "left" berada pada elemen 7 dan pointer "right" berada pada elemen berikutnya (2). Kita membandingkan nilai 7 dengan 2. Sebab 2 tidak lebih besar dari 7, pointer "left" tetap pada nilai 7.
3. Pointer "right" berpindah ke elemen berikutnya (89), sementara pointer "left" tetap pada nilai 7. Kita membandingkan nilai 89 dengan 7. Sebab 89 lebih besar dari 7, pointer "left" berpindah ke nilai 89.
4. Sekarang, pointer "left" berada pada elemen 89 dan pointer "right" berada pada elemen berikutnya (3). Kita membandingkan nilai 89 dengan 3. Sebab 3 tidak lebih besar dari 89, pointer "left" tetap pada nilai 89.
5. Proses berakhir, nilai pada pointer "left" ditetapkan sebagai nilai terbesar dalam array.

Dengan demikian, kita menggunakan dua pointer untuk membandingkan dan mengganti nilai pointer "left" jika ada nilai yang lebih besar saat melintasi array. Pada akhirnya, nilai pada pointer "left" adalah nilai terbesar dalam array.

Sekarang Anda sudah paham secara teoretis cara algoritma two pointers bekerja untuk memproses array dalam mencari nilai terbesar. Selanjutnya, mari kita ubah penjelasan tersebut ke dalam program Python.

var\_arr = [1, 7, 2, 89, 3]

left\_pointer = var\_arr[0]

for i in range(1, len(var\_arr)):

right\_pointer = var\_arr[i]

if right\_pointer > left\_pointer:

left\_pointer = right\_pointer

print(left\_pointer)

Pada program di atas, hal pertama yang dilakukan adalah menginisialisasi variabel "var\_arr" dengan array "[1, 7, 2, 89, 3]". Kedua, kita menginisialisasi variabel "left\_pointer" dengan nilainya adalah indeks pertama variabel "var\_arr" (var\_arr[0]).

Kita menggunakan perulangan "for" untuk mengakses semua indeks dari indeks ke-1 hingga panjang array. Untuk mengetahui panjang array, kita menggunakan fungsi "len()", yang  bertujuan menghitung panjang atau banyaknya elemen dari list, set, dan string.

Dalam perulangan "for" tersebut kita mendeklarasikan variabel "right\_pointer" yang akan terus berpindah dari indeks ke-1 hingga indeks terakhir (akhir panjang array). Setelah memiliki "left\_pointer" dan "right\_pointer", kita membandingkan nilai keduanya. Jika "right\_pointer" lebih besar dari "left\_pointer", kita akan memperbarui nilai "left\_pointer" dengan nilai "right\_pointer".

Proses tersebut terjadi secara berulang sampai nilai yang tersimpan dalam "left\_pointer" dijadikan sebagai nilai maksimal dari array. Lalu, kita mencetak nilai tersebut dengan perintah "print(left\_pointer)".

result = total/len(var\_array)