## Pengantar Tipe Data

[](https://www.dicoding.com/academies/620/tutorials/32460?from=32455)

Halo! Selamat datang kembali di kelas ini. Pada materi sebelumnya, Anda sudah memahami konsep dasar pemrograman prosedural. Mulai dari mendefinisikan, mengenal prinsip dasar algoritma dan interpretasinya, hingga mengulik insight menarik di balik pemrograman prosedural sudah Anda pelajari.

Pada materi ini, kita akan mengenal perilaku tipe data dalam sebuah pemrograman prosedural. Untuk itu, Anda perlu belajar beberapa materi:

* penggambaran konsep tipe data dalam pemrograman;
* pengindentifikasian tipe data primitif dan tipe data lainnya; dan
* penerapan konsep tipe data dalam sebuah aplikasi.

Namun, sebelum lanjut ke penjelasan lebih dalam mengenai tipe data, sebaiknya Anda perlu tahu konsepnya terlebih dahulu. Yuk, kita simak dahulu.

### Mengenal Data

Pernahkah merasakan bahwa saat ini Anda sedang dikelilingi oleh informasi yang melimpah? Saat scrolling media sosial, kita menerima informasi berupa foto, komentar, hingga jumlah likes. Ketika melakukan transaksi keuangan, Anda sedang berinteraksi dengan beragam informasi, seperti nomor rekening, tanggal, dan jumlah transaksi. Bahkan secara tidak sadar, sejak dari kandungan pun hingga lahir menjadi bayi, Anda sudah memiliki informasi riwayat diri sendiri.

Lalu, apakah informasi tersebut dapat dikatakan sebagai data? Tentu saja. Jika kita menilik Kamus Besar Bahasa Indonesia (KBBI), pengertian [data](https://kbbi.kemdikbud.go.id/entri/data) dalam konteks komputer merupakan informasi dalam bentuk yang dapat diproses oleh komputer, seperti representasi digital dari teks, angka, gambar grafis, atau suara.

Saat ini, seluruh aktivitas yang kita lakukan akan berhubungan dengan data. Setiap langkah yang kita ambil selalu dapat dihitung dan diamati berdasarkan data yang dihasilkan. Hal ini terjadi karena ia terikat dengan aksi yang kita lakukan.

Berkaca dari hal tersebut, dapat disimpulkan bahwa setiap aksi selalu berjalan dengan data yang mendukung aksi tersebut. Data tidak mungkin digunakan ketika kita tidak ada kebutuhan yang mendasarinya. Data memiliki peran penting dalam mendukung keputusan dan tindakan yang kita ambil, serta memberikan landasan yang kuat untuk mencapai tujuan yang diinginkan.

|  |
| --- |
| Ketika bekerja merenovasi rumah, Bapak Fajar perlu mendefinisikan elemen-elemen rumah. Misalnya ukuran tanah, jumlah kamar, jumlah lantai, bahan bangunan, dan sebagainya perlu dipersiapkan untuk menunjang renovasi rumah. Elemen tersebut akan digunakan selama proses renovasi sebagai panduan dalam membangun rumah sesuai kebutuhan dan spesifikasi yang diinginkan.  [Bapak Fajar sedang melihat bahan bangunan untuk memastikan cukup untuk renovasi satu rumah.](https://www.dicoding.com/academies/620/tutorials/32460?from=32455)  Proses pendefinisian elemen tersebut tidak sembarangan. Penggunaan elemen tersebut sudah diperhitungkan oleh rekan kerja Bapak Fajar yang kompeten untuk digunakan secara optimal. Bapak Fajar tidak akan mengizinkan elemen baru jika tidak ada kebutuhan yang mendasarinya selama proses renovasi berlangsung.  Dengan menerapkan analogi di atas pada konteks pemrograman, kita dapat melihat bahwa proses pendefinisian elemen-elemen dalam renovasi rumah mirip dengan pemilihan tipe data dalam pemrograman. Seperti halnya Bapak mempertimbangkan dengan cermat elemen-elemen rumah sebelum merenovasi, pemrogram juga harus melakukan pemilihan tipe data yang tepat dan relevan sebelum memulai pengembangan aplikasi. Hal ini akan memastikan program berjalan dengan baik sesuai dengan kebutuhan dan spesifikasi yang ditetapkan. |

### Mengenal Variabel

Menurut pengertian sebelumnya, data dapat berupa teks, angka, gambar grafis, atau suara. Namun, dalam konteks pemrograman, data memiliki makna yang lebih spesifik. Data dalam konteks pemrograman merujuk pada informasi yang digunakan oleh program untuk melakukan operasi dan pengolahan instruksi tertentu.

Data dalam pemrograman dapat dibedakan berdasarkan tipenya, yaitu tipe data primitif, seperti string untuk teks, numerik untuk angka, boolean untuk pengambilan keputusan, dan tipe data non-primitif, seperti kumpulan data primitif dalam satu objek. Selain itu, data dapat dibedakan berdasarkan sifatnya, yaitu mutable, yang berarti nilainya dapat berubah, dan immutable, yang berarti nilainya tidak dapat berubah.

Lalu, bagaimana program dapat menyimpan informasi yang kita berikan untuk menyelesaikan permasalahan? Variabel adalah solusinya. Variabel ini dapat menyimpan data dalam bentuk boolean, angka, string, atau informasi lainnya. Variabel akan tersimpan dalam memori komputer dan dapat dipanggil serta digunakan saat kita butuhkan.

|  |
| --- |
| Saat bekerja di area pembangunan rumah, Bapak Fajar melihat beberapa rekan kerja membawa toolbox. Isi dari toolbox itu adalah perkakas yang beragam, mulai dari gergaji, palu, meteran, dll. Toolbox tersebut disimpan di area pembangunan dan akan diambil kembali apabila mereka membutuhkan perkakas selama proses pembangunan rumah.  [Rekan kerja Bapak Fajar membawa toolbox di area pembangunan rumah.](https://www.dicoding.com/academies/620/tutorials/32460?from=32455)  Layaknya pembangunan rumah, pembuatan program pun serupa dengan analogi toolbox tersebut. Toolbox itu ibaratkan sebuah wadah penyimpan informasi atau data yang bisa dipakai dan dipanggil sesuai kebutuhan kita. Apabila membutuhkan perkakas tertentu, kita dapat mengambil toolbox dan ambil perkakas yang ingin dipakai. |

Variabel merupakan **nama** yang tersimpan dalam memori komputer, sedangkan isi dari variabel tersebut merupakan **harga** (nilai) yang mengandung suatu informasi. Sebab variabel tersimpan di memori komputer, kita dapat memanggilnya kapan pun apabila kita membutuhkan informasi yang terkandung di dalamnya. Berikut adalah gambaran pendefinisian suatu variabel.

[](https://www.dicoding.com/academies/620/tutorials/32460?from=32455)

Dalam bahasa pemrograman Python, mendeklarasikan sebuah variabel seperti menuliskan sebuah rumus dalam matematika. Bagian kiri merupakan nama variabel, sedangkan bagian kanan berisi nilai atau data.

1. numFloors = 1
2. propertyType = "Rumah"

Terkadang kita butuh menyimpan sebuah nilai dengan informasi yang berubah-ubah. Saat ini variabel numFloors berisi 1. Mungkin di akhir program variabel tersebut berubah menjadi 2. Variabel tersebut memiliki sifat **mutable**. Artinya, nilai atau informasi dalam variabel dapat berubah suatu waktu.

1. numFloors = 1
2. propertyType = "Rumah"
4. # akhir program
5. numFloors = 2

Berbeda dengan mutable, variabel yang bersifat **immutable**tidak bisa berubah selama program berjalan. Ia akan terus bernilai sama mulai dari pendefinisian nilai awal hingga program berakhir.

Selain berdasarkan sifatnya, data dibedakan menjadi dua tipe, yakni **primitif** dan **non-primitif**. Tipe data yang sudah disebutkan sebelumnya seperti boolean, numerik, dan string adalah tipe data primitif. Namun, ada juga tipe data non-primitif yang memiliki kompleksitas yang lebih tinggi. Anda mulai penasaran? Yuk, lanjut ke materi berikutnya. Let’s go!

## Tipe Data Primitif

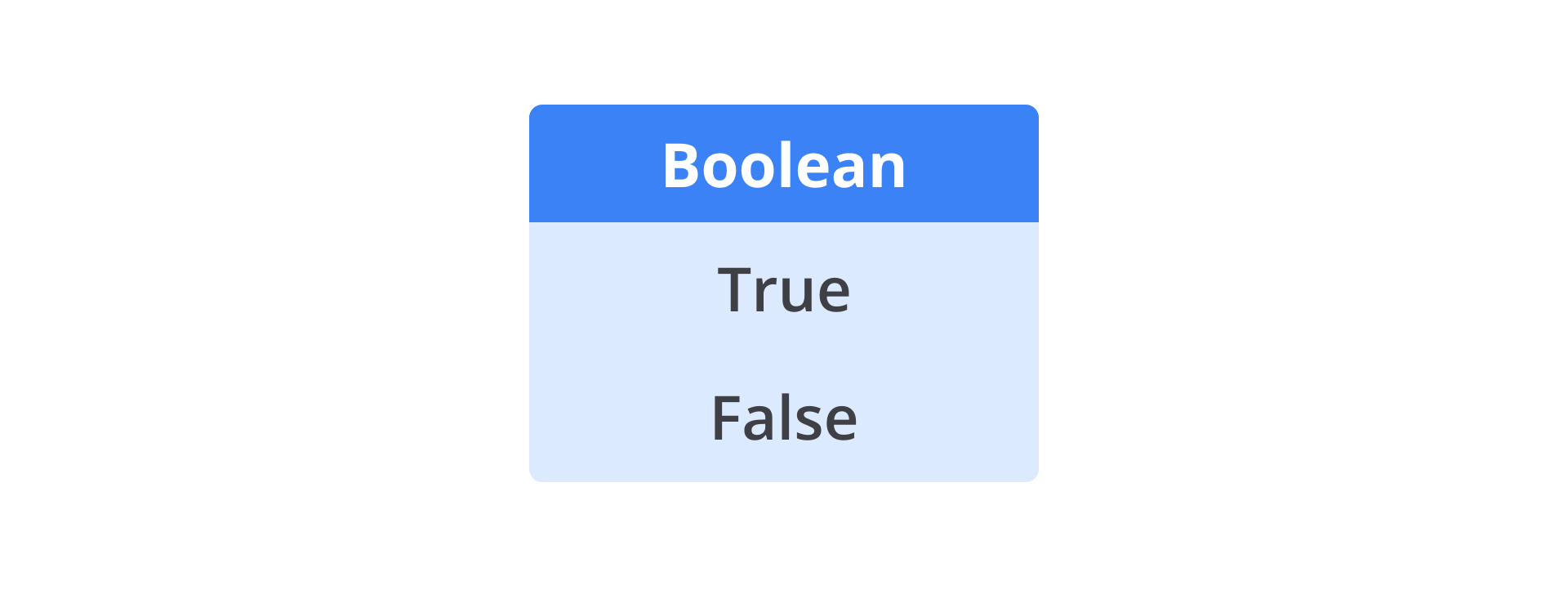
Pada penjelasan sebelumnya, kita sempat menyinggung beberapa tipe data, seperti boolean untuk pengambilan keputusan, numerik untuk angka, dan string untuk teks. Ketiga tipe data tersebut merupakan tipe data primitif dan bersifat mutable atau dapat berubah sesuai dengan yang kita inginkan.

Lalu, apa arti atau maksud dari kata primitif itu sendiri? Kata primitif ini merujuk pada elemen atau komponen dasar yang tersedia dalam bahasa pemrograman. Jika mendengar kata dasar, berarti elemen tersebut merupakan fondasi yang tidak dapat diuraikan lagi. Ia menjadi bagian utama yang berfungsi untuk membangun logika pemrograman.

|  |
| --- |
| Komponen seperti semen, pasir, batu, baja, kayu, serta bahan bangunan lainnya merupakan contoh bagian fundamental yang dapat digunakan dalam membangun rumah. Apabila tidak ada komponen tersebut, pembangunan rumah tidak akan berjalan dengan baik. Masing-masing komponen berperan penting untuk memastikan bangunan berdiri dengan kokoh dan aman.  [Bahan bangunan, seperti batu, semen, dan kayu.](https://www.dicoding.com/academies/620/tutorials/32465?from=32460)  Konsep yang sama berlaku dalam pemrograman. Komponen dasar memiliki peran penting dalam membangun program secara utuh. Logika dan perhitungan dalam sebuah program pasti dibangun berdasarkan komponen dasar ini. Layaknya bahan bangunan, komponen seperti string, numerik, dan boolean menjadi bagian fundamental yang harus tersedia dalam program yang dibangun. |

### Boolean

Boolean merupakan tipe data primitif yang hanya memiliki dua nilai, yaitu “**True**” dan “**False**”. Tipe data ini dapat Anda pakai untuk mengontrol alur program, pengambilan keputusan, dan pengujian suatu kondisi.

[](https://www.dicoding.com/academies/620/tutorials/32465?from=32460)

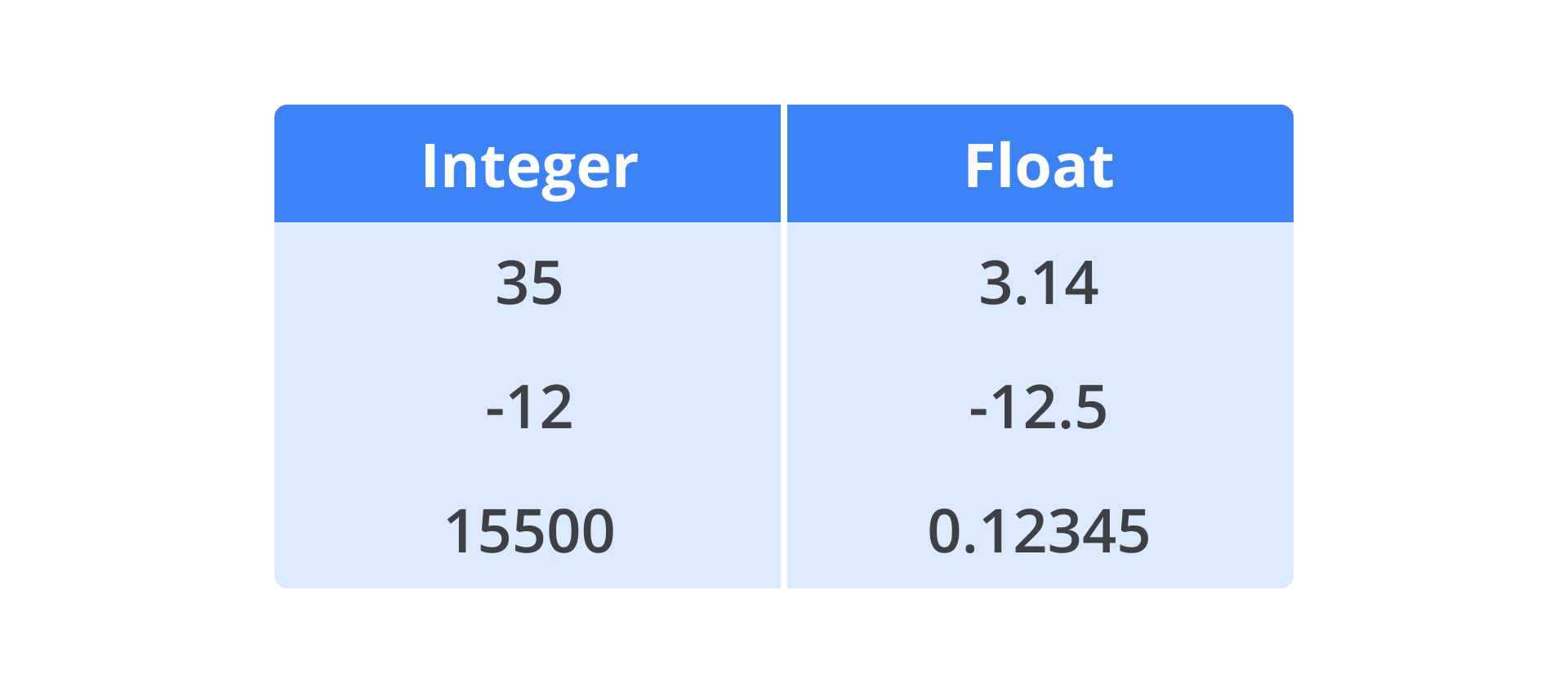
|  |
| --- |
| Bayangkan bila pemilik rumah yang dibangun oleh proyek Bapak Fajar datang ke area pembangunan. Mereka akan bertanya kepada Bapak Fajar, “Apakah rumah sudah selesai? Jika selesai, saya akan tinggal di rumah ini seminggu setelah pembangunan selesai. Namun, saya akan tinggal di rumah saudara apabila rumah belum selesai.”  Bapak Fajar menjawab, “Belum, Pak, rumah masih dalam proses pembangunan.” Mendengar jawaban dari Bapak Fajar, pemilik akan melakukan tindakan untuk tinggal di rumah saudara karena rumahnya belum selesai.  [Percakapan pemilik rumah dengan Bapak Fajar yang menanyakan rumah sudah selesai atau belum.](https://www.dicoding.com/academies/620/tutorials/32465?from=32460)  Konsep pengambilan keputusan yang diambil oleh pemilik rumah menggambarkan penggunaan boolean dalam pemrograman. Jika pertanyaan “Apakah rumah sudah selesai?” dilontarkan, terdapat dua opsi jawaban, yakni **sudah selesai** bersifat **True**, dan **belum selesai** bersifat **False**. Setiap opsi memiliki keluaran aksi yang berbeda, bahkan berlawanan. Itulah alasan boolean digunakan untuk mengontrol alur dan pengambilan keputusan dalam suatu program. |

Berikut adalah implementasi variabel bertipe data boolean dalam bahasa pemrograman Python.

1. hasTerrace = True
2. hasGarage = False

### Numerik

Numerik atau angka merupakan tipe data primitif yang umumnya berupa **bilangan bulat** dan **bilangan riil**. Bilangan bulat adalah angka tanpa pecahan atau desimal. Biasanya, bilangan bulat disandingkan dengan bilangan **integer** dalam pemrograman. Sebaliknya, angka dengan pecahan atau desimal disebut sebagai bilangan riil. Dalam pemrograman, bilangan riil disebut juga dengan **float**.

[](https://www.dicoding.com/academies/620/tutorials/32465?from=32460)

|  |
| --- |
| Ketika merencanakan desain rumah, Bapak Fajar memperkirakan jumlah ruang yang akan ada di bangunan tersebut. Perhitungan tersebut berdasarkan luas tanah yang akan digunakan. Dalam desainnya, Bapak Fajar memasukkan 2 kamar mandi dengan ukuran lebar 1,5 meter dan panjang 2 meter. Dengan menggunakan informasi ini, Bapak Fajar dapat merencanakan tata letak dan dimensi ruangan secara terperinci untuk memastikan rumah dibangun sesuai kebutuhan dan spesifikasi.  [Bapak mendesain denah untuk pembangunan rumah.](https://www.dicoding.com/academies/620/tutorials/32465?from=32460)  Informasi jumlah dan ukuran kamar mandi pada bangunan tersebut merupakan data bertipe numerik. Data tersebut digunakan untuk merepresentasikan angka yang terkait dengan perhitungan matematis atau perhitungan. Dalam hal ini, jumlah ruang tidur bernilai 2 dinyatakan dalam **integer**, sedangkan ukuran ruang tidur bernilai 1,5 dinyatakan dalam **float**. |

Penggunaan integer dan float ini sedikit berbeda. Dari nilainya sudah terlihat jelas bahwa integer atau bilangan bulat tidak memiliki komponen desimal sedangkan float memilikinya. Perbedaan tersebut membuat tipe data float memiliki tingkat presisi yang tinggi. Dengan begitu, Anda dapat menyimpan angka di belakang koma supaya proses operasi menghasilkan nilai yang akurat.

Berikut adalah contoh implementasi variabel bertipe numerik dalam bahasa pemrograman Python.

1. numberOfRestroom = 3
2. heightOfRestroom = 2
3. widthOfRestroom = 1.5

### String

String merupakan tipe data primitif yang digunakan untuk merepresentasikan teks atau urutan karakter. String dapat berisi kombinasi huruf, angka, simbol, dan spasi yang membentuk kata, kalimat, atau teks yang panjang. Teks tersebut harus diapit oleh tanda kutip ganda (“”) atau tunggal (‘’) agar teridentifikasi sebagai string oleh komputer.

[](https://www.dicoding.com/academies/620/tutorials/32465?from=32460)

|  |
| --- |
| Sebelum Bapak Fajar melakukan pembangunan rumah, ternyata pemilik tanah pernah mendapatkan selembar brosur tentang promosi rumah. Dia mendapatkan berbagai informasi penting tentang rumah, seperti alamat rumah, spesifikasi bangunan, dan fasilitas yang tersedia. Brosur tersebut memberikan gambaran yang jelas tentang perencanaan pembangunan rumah baru melalui informasi berupa teks.  [Pemilik rumah menunjukkan brosur rumah dengan bonus furnitur, yaitu lemari, kasur, shower, wastafel, dll.](https://www.dicoding.com/academies/620/tutorials/32465?from=32460)  Setiap kata, kalimat, atau paragraf dalam brosur tersebut merupakan representasi dari informasi bertipe data string. Informasi alamat rumah mewakili informasi string yang berisi angka dan nama jalan. Begitu pula dengan spesifikasi bangunan, seperti jumlah kamar, luas tanah, dan luas bangunan, dapat diwakili oleh string. |

Berikut adalah contoh implementasi variabel bertipe data string dalam bahasa pemrograman Python.

1. buildingArea = "88 meter persegi"
2. homeAddress = 'Jl. Batik Kumeli No. 50, Bandung'
3. facilities = "Pasar tradisional, tempat olahraga, SPBU"

**Catatan:**

Apabila ingin memastikan variabel bertipe data numerik, string, boolean, atau lainnya, Anda dapat menggunakan perintah [type()](https://docs.python.org/3/library/functions.html#type). Perintah tersebut akan mengembalikan tipe data dari suatu objek. Penerapan perintah type dapat Anda lihat pada contoh kode berikut.

1. a = 5
2. b = "Hello"
3. c = 4.5
4. d = False
6. print(type(a))  # <class 'int'>
7. print(type(b))  # <class 'str'>
8. print(type(c))  # <class 'float'>
9. print(type(d))  # <class 'bool'>

Perlu dipahami dengan saksama bahwa tipe data primitif merupakan elemen atau komponen dasar dalam pemrograman. Memahami tipe data primitif dengan baik akan membantu Anda menguasai fondasi utama dalam pemrograman. Tanpa pemahaman yang baik, Anda akan kesulitan untuk memahami dan mengimplementasikan konsep-konsep yang lebih kompleks dalam pemrograman. Apabila memiliki pertanyaan terkait materi ini, Anda bisa bertanya di Forum Diskusi, ya.

## Tipe Data Non-primitif

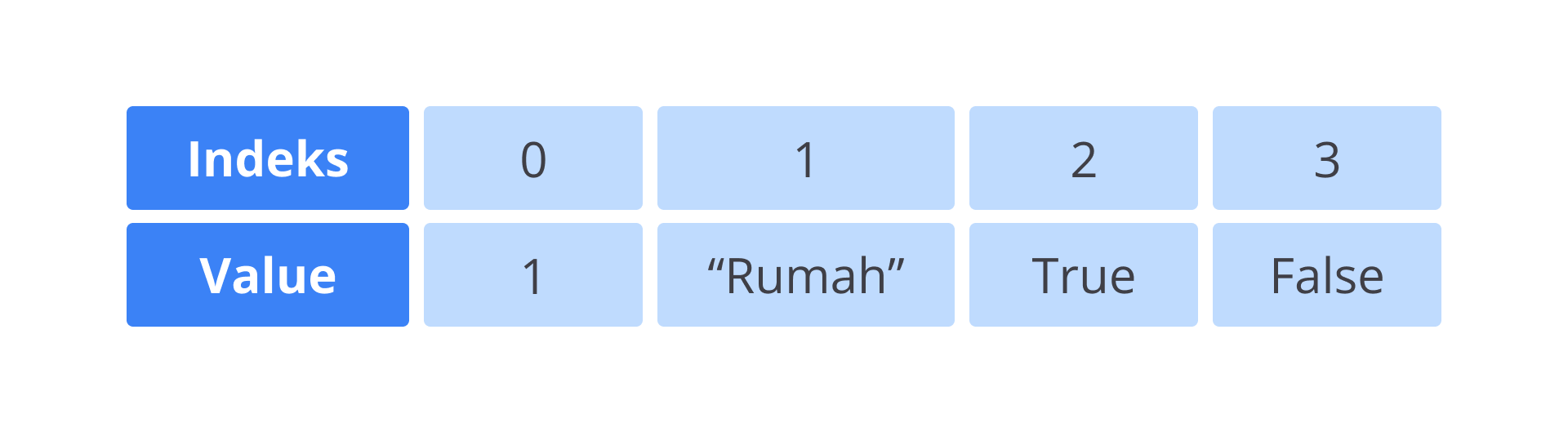
Jika materi sebelumnya adalah tipe data primitif, saat ini kita akan memasuki materi tipe data non-primitif. Tipe data non-primitif ini memungkinkan kita untuk menyimpan koleksi nilai bertipe data. Jadi, apakah kita bisa mengombinasikan beberapa tipe data primitif menjadi satu variabel? Tentu bisa. Yuk, kita menilik lebih jauh tentang tipe data non-primitif.

### List

Tipe data list merupakan kumpulan elemen atau nilai yang disusun dalam urutan tertentu. Isi dari tipe data ini dapat berupa string, numerik, boolean, bahkan tipe data lainnya, seperti list yang bersarang (nested list). Tidak perlu mengelompokkan satu jenis yang seragam, list dapat berisi berbagai tipe data dalam satu variabel.

[](https://www.dicoding.com/academies/620/tutorials/32470?from=32465)

Ketika ingin mendefinisikan nilai dalam sebuah list, Anda perlu memperhatikan posisi atau urutannya. Setiap elemen dalam list memiliki **indeks** yang menunjukkan posisi dalam suatu urutan. Indeks selalu dimulai dari angka 0 serta diakhiri dengan angka n-1 dan n adalah banyaknya elemen yang terisi.

[](https://www.dicoding.com/academies/620/tutorials/32470?from=32465)

|  |
| --- |
| Bapak Fajar memiliki daftar bahan bangunan saat melakukan inspeksi. Hal ini ditujukan untuk memudahkan Bapak Fajar dalam memeriksa bahan yang ada atau tidak tersedia pada proses pembangunan berlangsung. Daftar bahan bangunan terdiri dari pasir, semen, kayu, cat, dan keramik. Namun, Bapak Fajar lupa memasukkan beberapa bahan bangunan, seperti batu bata dan baja sehingga beliau menambahkannya ke daftar tersebut.  [Bapak Fajar sedang memasukkan beberapa bahan bangunan yang tidak tersedia pada daftar inspeksi.](https://www.dicoding.com/academies/620/tutorials/32470?from=32465)  Analogi di atas menggambarkan penggunaan list dalam proses pembangunan rumah. Daftar bahan bangunan diibaratkan list yang dapat berisi nilai atau informasi. Isi dari sebuah list dapat bertambah, berkurang, atau berubah bergantung dengan kebutuhan yang diinginkan. Dengan begitu, kita dapat mengelola daftar secara fleksibel dan mudah. |

Tipe data list ini bersifat mutable. Artinya, nilai dari sebuah list dapat berubah suatu waktu. Lalu, cara mendeklarasikan suatu variabel list harus dimulai dan diakhiri dengan kurung siku ‘[ ]’.

Berikut adalah contoh implementasi pendeklarasian variabel, pemanggilan suatu elemen, dan perubahan nilai elemen pada tipe data list dalam pemrograman Python.

1. # deklarasi variabel list
2. numbers = [23, 43, 15, 34]
4. # cara mengakses elemen dalam variabel list
5. numbers[0]    # 23
6. numbers[1]    # 43
7. numbers[2]    # 15
8. numbers[3]    # 34
10. # cara mengubah nilai elemen dalam variabel list
11. numbers[2] = 51
12. numbers[2]   # 51

Kelebihan utama dari list adalah tingkat fleksibilitasnya dalam mengelola elemennya. Kita dapat memanipulasi isi elemen suatu list dengan cara menambahkan, menghapus, dan mengubah sesuai dengan kebutuhan. Kemampuan ini memungkinkan kita untuk mengelola data secara dinamis selama program berlangsung.

1. # deklarasi variabel list numbers
2. numbers = [1, 2, 3]
4. # menambahkan elemen di indeks ke-4
5. numbers.insert(4, 5)    # [1, 2, 3, 5]
7. # mengubah elemen di indeks ke-4
8. numbers[3] = 4    # [1, 2, 3, 4]
10. # menghapus elemen di indeks ke-2
11. numbers.pop(2)    # [1, 2, 4]

Jika Anda perhatikan sejenak, proses deklarasi dan mengakses elemen dalam list serupa dengan array dalam bahasa pemrograman lainnya, seperti JavaScript. Namun, ada hal unik dari cara mendeklarasikan list dalam Python, yaitu Anda dapat mendefinisikan berbagai macam nilai dengan tipe data yang berbeda. Tidak hanya bertipe data primitif, Anda juga dapat memasukkan nilai bertipe data list, bahkan data non-primitif lainnya.

Berikut adalah contoh penerapan penggunaan list apabila diberi nilai dengan tipe data yang beragam.

1. # deklarasi variabel list
2. myList = [1, “Rumah”, 4.5, True, ["Kamar mandi", "Kamar tidur"]]
4. # cara mengakses elemen dalam variabel list
5. myList[0]   # 1
6. myList[1]   # "Rumah"
7. myList[3]   # True
8. myList[4]   # ["Kamar mandi", "Kamar tidur"]
10. # cara mengakses elemen dalam sublist
11. myList[4][0]   # "Kamar mandi"
12. myList[4][1]   # "Kamar tidur"

Fleksibilitas dalam memberikan nilai bertipe data non-primitif menjadi satu keunggulan bahasa pemrograman Python. Struktur data yang universal membuat Python mampu menyimpan dan mengelola nilai dalam kumpulan objek tertentu. Dengan begitu, Anda dapat memanipulasi data dengan data yang heterogen.

Perlu Anda ingat bahwa meskipun Python dapat memberikan fleksibilitas dalam mendeklarasikan variabel non-primitif, disarankan untuk menggunakannya secara konsisten. Hal ini akan membantu Anda dalam menjaga kejelasan dan kegunaan dari variabel tersebut.

### Tuple

Tuple merupakan jenis variabel bertipe data list. Namun, perbedaan utama antara tuple dan list terletak pada sifatnya yang tidak dapat diubah (immutable). Artinya, kita dapat mendefinisikan kumpulan nilai bertipe data primitif **tanpa adanya perubahan data** pada setiap elemennya. Cara mendeklarasikan suatu variabel tuple cukup dengan mengapit sekumpulan nilai menggunakan tanda kurung ‘( )’.

|  |
| --- |
| Pemilik rumah sedang melihat proses pembangunan rumah. Dia sedang mengecek dan membandingkan kondisi rumah dengan spesifikasi rumah pada brosur yang diberikan di awal sebelum pembangunan berlangsung. Ketika mengecek brosur, dia melihat jumlah, jenis, dan tambahan perabotan pada ruang kamar mandi yang akan dibangun. Dalam brosur tertulis dua, umum dan khusus; wastafel; shower; dan WC. Brosur tersebut menjadi panduan atau acuan bagi pemilik rumah untuk memastikan tidak ada pengurangan perabotan ataupun ruangan.  [Pemilik rumah sedang memeriksa kelengkapan furnitur di kamar mandi.](https://www.dicoding.com/academies/620/tutorials/32470?from=32465)  Dalam konteks pemrograman, daftar tersebut merupakan analogi dari tipe data tuple. Daftar yang tertulis dalam brosur menjadi acuan yang tidak boleh ditambah, dikurang, bahkan diubah. Daftar tersebut dapat digunakan sebagai pembanding untuk menjaga konsistensi dari awal hingga akhir pembangunan rumah. |

Berikut adalah contoh implementasi pendeklarasian variabel dan pemanggilan suatu elemen pada tipe data tuple dalam pemrograman Python.

1. # deklarasi variabel tuple
2. numbers = (23, 43, 15, 34)
4. # cara mengakses elemen dalam variabel tuple
5. numbers[0]    # 23
6. numbers[1]    # 43
7. numbers[2]    # 15
8. numbers[3]    # 34

Tuple memiliki posisi atau indeks seperti list untuk menunjukkan letak nilai dalam urutan. Namun, kita tidak dapat mengganti elemen dalam tuple karena tidak dapat diubah. Berikut adalah perbedaan mendasar antara list dan tuple.

| **Perbedaan** | **List** | **Tuple** |
| --- | --- | --- |
| Mutable/Immutable | Bersifat mutable atau nilai dapat berubah. | Bersifat immutable atau nilai tidak dapat berubah. |
| Karakter pengapit elemen | Diapit dengan tanda kurung siku ‘[ ]’. | Diapit dengan tanda kurung ‘( )’. |
| Operasi umum yang digunakan | Operasi penambahan, pengurangan, atau pengubahan elemen data. | Hanya mengakses elemen saja. |
| Cara mengakses elemen | Panggil berdasarkan indeks. | Panggil berdasarkan indeks. |

### Set

Set termasuk tipe data mutable yang berupa sekumpulan elemen unik dan tidak memiliki urutan tertentu. Mendengar kata unik, artinya tidak ada satu pun nilai elemen yang boleh sama. Elemen dalam variabel set tidak boleh diisi oleh nilai duplikasi karena sifatnya mirip dengan himpunan matematika.

|  |
| --- |
| Sebelum memulai pembangunan, pemilik rumah perlu menyiapkan beberapa surat untuk memenuhi persyaratan hukum dan administrasi. Hal ini ditujukan untuk kelegalan dalam mendirikan suatu bangunan. Surat-surat tersebut meliputi Izin Mendirikan Bangunan (IMB), Pembebasan Lahan, Perjanjian Kontrak, Izin Lingkungan, dan Izin Penggunaan Bangunan (IPM). Beberapa surat tersebut tidak mungkin dibuat berkali-kali karena akan terjadi duplikasi.  [Surat-surat untuk memenuhi persyaratan hukum dan administrasi saat membangun rumah.](https://www.dicoding.com/academies/620/tutorials/32470?from=32465)  Analogi di atas mirip dengan sifat tipe data set. Ketika membuat daftar menggunakan set, kita hanya perlu menyertakan elemen-elemen unik tanpa adanya duplikasi. Setiap elemen hanya perlu ditambahkan sekali dan tidak akan muncul lagi dalam daftar tersebut. Dengan demikian, elemen dalam daftar tidak perlu dibuat berkali-kali. |

Cara mendeklarasikan suatu variabel set cukup dengan mengapit sekumpulan nilai menggunakan tanda kurung kurawal ‘{ }’.

Berikut adalah contoh implementasi pendeklarasian variabel pada tipe data set dalam pemrograman Python.

1. # deklarasi variabel set
2. numbers = {23, 43, 15, 34}
4. numbers   # {23, 43, 15, 34}

Keunikan dari tipe data set ini memungkinkan kita untuk menolak penambahan nilai yang sama. Apabila kita mencoba menambahkan nilai sama dengan elemen yang sudah ada, ia tidak akan ditambahkan. Hal ini menjadikan set cocok digunakan untuk menyimpan sekumpulan nilai yang unik.

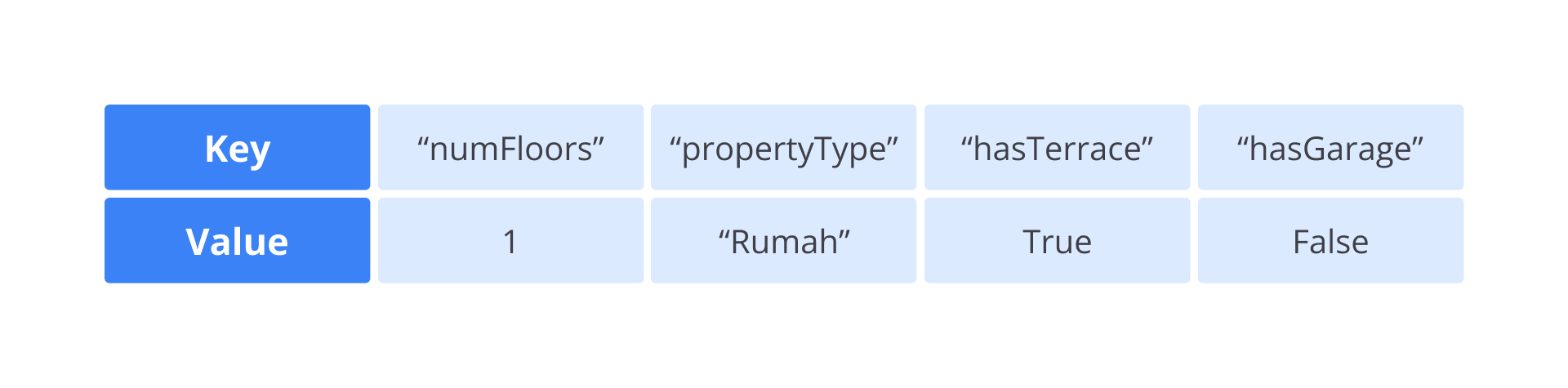
Tipe data set memiliki kegunaan lain, yaitu dapat melakukan operasi union (gabungan), intersection (irisan), dan difference (perbedaan). Hal ini selaras dengan sifatnya yang mirip dengan himpunan matematika yang mampu melakukan operasi-operasi tersebut.

1. # deklarasi variabel set a dan b
2. a = {1, 2, 3}
3. b = {2, 3, 4}
5. # operasi gabungan
6. a.union(b)    # {1, 2, 3, 4}
8. # operasi irisan
9. a.intersection(b)    # {2, 3}
11. # operasi perbedaan
12. a.symmetric\_difference(b)    # {1, 4}

Perlu diingat kembali bahwa set tidak bisa mengakses elemen berdasarkan indeks layaknya list dan tuple. Hal ini karena set tidak memiliki urutan. Untuk mengakses elemennya, Anda dapat menggunakan operasi seperti sebelumnya layaknya himpunan matematika.

### Dictionary

Dictionary merupakan tipe data mutable yang mampu menyimpan data dalam bentuk key-value. Apa maksudnya key-value? Key diibaratkan urutan dalam list sebagai identifier dari sebuah data, sedangkan value adalah data yang ingin kita simpan. Setiap key harus bernilai string unik karena merepresentasikan referensi untuk mengakses nilai. Tidak boleh adanya kesamaan key dalam suatu variabel dictionary.

[](https://www.dicoding.com/academies/620/tutorials/32470?from=32465)

|  |
| --- |
| Saat ingin merencanakan desain rumah, Bapak Fajar bercengkerama dengan pemilik proyek. Mereka mendiskusikan spesifikasi rumah yang akan dibangun pada kompleks perumahan. Akhirnya, mereka menyepakati bahwa seluruh rumah memiliki luas tanah 96 meter persegi, luas bangunan 88 meter persegi, lantai 2, kamar tidur 3, dan kamar mandi 2. Informasi tersebut dicatat oleh Bapak Fajar dan pemilik proyek supaya dapat digunakan sebagai acuan selama proses pembangunan berlangsung.  [Bapak Fajar dan pemilik rumah sedang berbincang membahas spesifikasi rumah di tanah kosong.](https://www.dicoding.com/academies/620/tutorials/32470?from=32465)  Jika dikaitkan dengan dictionary, kita dapat membuat sebuah variabel propertyDetail yang mewakili spesifikasi rumah tersebut. Kata kunci, seperti luas tanah, luas bangunan, lantai, kamar tidur, dan kamar mandi merupakan sebuah **key** atau identifier pada variabel dictionary. Sebaliknya, nilai yang menggambarkan identifier tersebut, seperti 96 meter persegi, 88 meter persegi, dst., merupakan **value** dari variabel dictionary.  Dengan menggunakan konsep dictionary, Bapak Fajar dan pemilik proyek dapat dengan mudah mengakses dan memeriksa spesifikasi rumah yang telah disepakati. Apabila ingin memastikan jumlah kamar tidur, mereka dapat mencari kata kunci “kamar tidur” dalam variabel dan melihat nilai yang ada padanya. |

Cara pendeklarasian variabel dictionary diapit oleh tanda kurung kurawal ‘{ }’ layaknya set.

Berikut adalah contoh implementasi pendeklarasian variabel, pemanggilan suatu elemen, dan perubahan nilai elemen pada tipe data dictionary dalam bahasa pemrograman Python.

1. # deklarasi variabel dictionary
2. propertyDetail = {"numFloors": 1, "propertyType": "Rumah", "hasTerrace": True, "hasGarage": False}
4. # cara mengakses elemen dalam variabel dictionary
5. propertyDetail["numFloors"]       # 1
6. propertyDetail["propertyType"]    # Rumah
7. propertyDetail["hasTerrace"]     # True
8. propertyDetail["hasGarage"]     # False
10. # cara mengubah nilai elemen dalam variabel dictionary
11. propertyDetail["hasTerrace"] = False
13. propertyDetail["hasTerrace"]   # False

Layaknya kamus (arti literal dari dictionary), ia dapat menyimpan dan mengakses value (informasi) berdasarkan key yang diberikan. Key berperan sebagai kata kunci yang digunakan untuk mengakses nilai terkait. Apabila kata kunci tersedia, kita dapat dengan mudah menemukan value atau informasi yang sesuai berdasarkan yang ingin dicari.

Dalam konteks pemrograman, tipe data non-primitif adalah tipe data yang memiliki tingkatan lebih advanced karena mengandung kumpulan tipe data primitif. Ibarat tas ransel atau koper, kita dapat mengisinya dengan berbagai barang. Akan tetapi, cara kita mengakses barang tersebut, diperlukan jalan yang beragam pula, mulai dari berdasarkan urutan, himpunan, bahkan kata kunci yang relevan dengan barang.

Apakah Anda sudah bisa membayangkan perbedaan dari setiap tipe data non-primitif? Untuk menguji pemahaman akan penggunaan data dalam bahasa pemrograman Python, terdapat beberapa kuis yang perlu Anda ikuti. Mau tahu bagaimana kuisnya? Yuk, segera ke materi berikutnya.

## Type Annotation pada Variabel

Fleksibilitas dalam menulis kode menjadi hal yang sangat diagungkan saat menjadi pengembang aplikasi Python. Mulai dari penulisan yang mudah, sederhana, mudah dipahami, dan masih banyak keuntungan lainnya saat menulis kode Python. Namun, ada yang perlu diperhatikan di balik fleksibilitas tersebut. Apa itu? Coba perhatikan kode berikut.

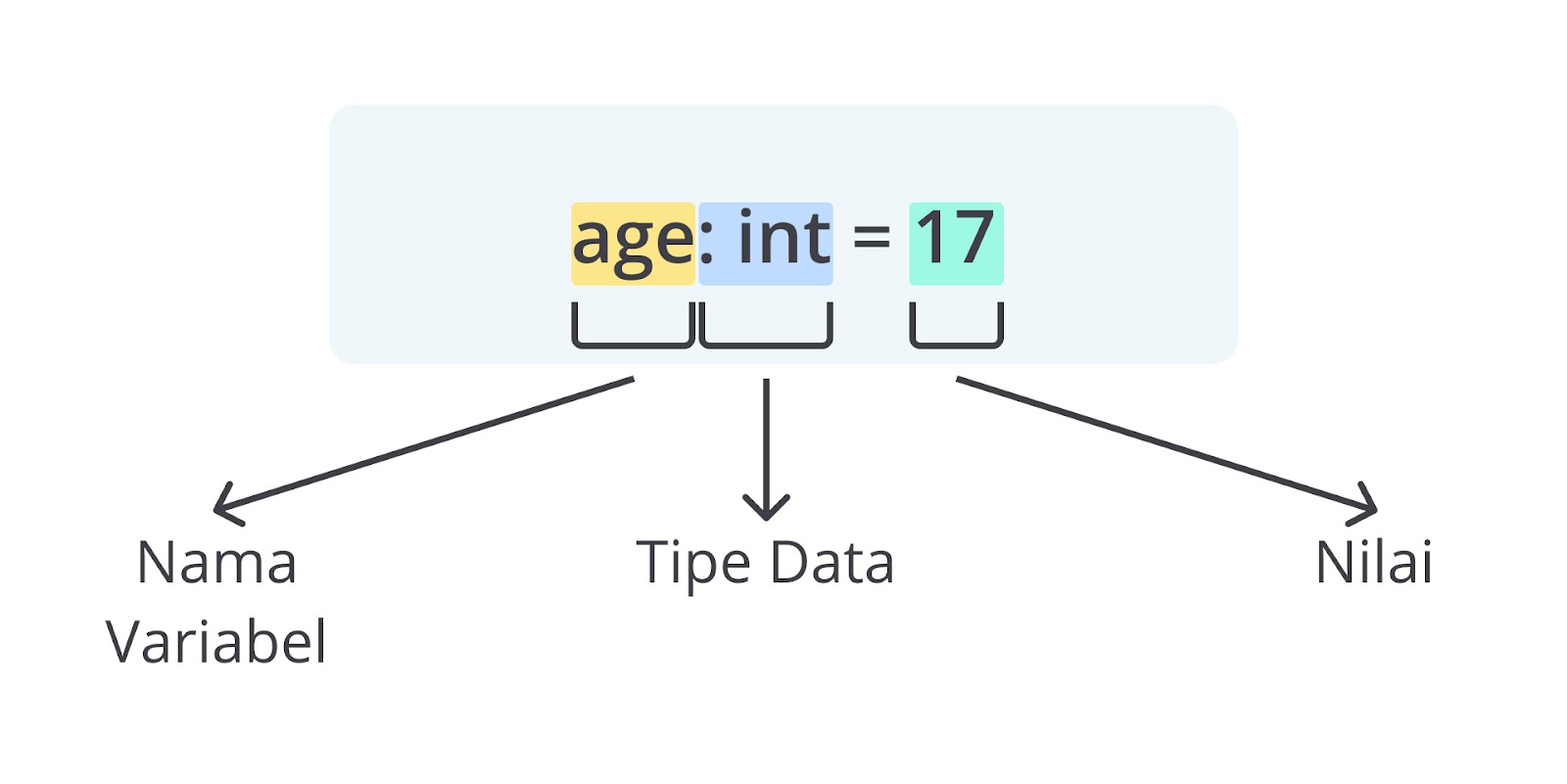
1. name = "Bima"
2. age = 17

Secara kasat mata dan eksekusi program, kode di atas tidak ada yang error dan berhasil dijalankan. Ada dua baris kode yang tertulis pada program dan nama variabel merepresentasikan nilainya. Coba Anda bayangkan apabila program Python berisi kode yang kompleks dan memiliki deklarasi yang cukup banyak. Bagaimana cara melacak tipe data dari variabel tersebut? Anda memerlukan banyak waktu untuk melacaknya, bukan?

Di sinilah penerapan anotasi perlu ditegaskan pada pemrograman Python. Apa itu anotasi? Dalam konteks pemrograman, anotasi adalah suatu mekanisme untuk menerangkan dan mempertegas penulisan kode. Anotasi ini dapat diibaratkan sebagai cara untuk menyatakan tipe data secara eksplisit pada suatu variabel.

Memang, pembuatan variabel pada Python tidak memerlukan deklarasi tipe data. Ia mampu mengetahui tipe data tersebut berdasarkan nilai dari variabel secara implisit. Namun, anotasi mampu membantu pembacaan kode, dokumentasi, dan analisis status yang disediakan oleh tools IDE, seperti linter atau type checking.

Anotasi tipe data dapat didefinisikan pada formula berikut.

[](https://www.dicoding.com/academies/620/tutorials/35173)

Anda dapat menyisipkan tipe data pada proses assignment setelah nama variabel, diikuti dengan tanda colon “:”, kemudian tipe data dapat disisipkan sebelum karakter sama dengan “=”. Alhasil, kode pada awal materi dapat diubah menjadi seperti berikut.

1. name**: str** = "Bima"
2. age**: int** = 17

Lalu, bagaimana dengan tipe data lainnya, seperti boolean, float, dan kumpulan data lainnya? Setiap tipe data memiliki anotasi tipe tersendiri. Berikut adalah beberapa anotasi yang perlu Anda tahu pada Python.

| **Tipe Data** | **Anotasi Tipe Data** |
| --- | --- |
| Boolean | bool |
| Integer | int |
| Float | float |
| String | str |
| List | list |
| Tuple | tuple |
| Set | set |
| Dictionary | dict |

Alhasil, Anda dapat menerapkan seluruh anotasi tersebut pada setiap deklarasi variabel yang ada.

1. # tipe data primitif
2. hasTerrace: bool = True
3. numberOfRoom: int = 3
4. roomArea: float = 9.5
5. buildingArea: str = "88 meter persegi"
7. # tipe data non-primitif
8. numbersList: list = [23, 43, 15, 34]
9. fruitsTuple: tuple = ("Apel", "Jeruk", "Semangka")
10. colorSet: set = {"Hijau", "Biru", "Merah"}
11. propertyDetailDict: dict = {"numFloors": 1, "propertyType": "Rumah", "hasTerrace": True, "hasGarage": False}

Untuk tipe data non-primitif, Anda bisa memberikan informasi yang lebih spesifik agar menegaskan tipe data dari isi elemennya. Hal ini berguna untuk mengetahui aksi yang seharusnya dilakukan pada variabel tersebut. Berikut contohnya.

1. # tipe data non-primitif
2. numbersList: **list[int]** = [23, 43, 15, 34]
3. fruitsTuple: **tuple[str]** = ("Apel", "Jeruk", "Semangka")
4. colorSet: **set[str]**= {"Hijau", "Biru", "Merah"}
5. propertyDetailDict: **dict[str, any]** = {"numFloors": 1, "propertyType": "Rumah", "hasTerrace": True, "hasGarage": False}

Mungkin Anda mempunyai pertanyaan, mengapa ada anotasi any? Tipe data apa lagi, tuh? Nah, anotasi **any** ini mengacu pada variabel yang memiliki nilai yang beragam.

Misalnya, variabel propertyDetailDict memiliki dua komponen, yaitu key dan value. Umumnya, key pada dictionary dapat bernilai string atau integer, asalkan tidak sama dengan key lainnya. Value pada dictionary dapat berisi elemen yang beragam, mulai dari integer, boolean, string, dll. Untuk itu, Python menyediakan anotasi **any** untuk menanggulangi kondisi tersebut.

Sejak Python 3.9+, kita dapat mendefinisikan tipe data pada kumpulan data lebih detail lagi. Tidak sekadar menuliskan anotasi any, tetapi kita dapat mencantumkan tipe data apa saja yang boleh tersedia pada kumpulan data tersebut.

Contohnya, sebuah list dapat menampung dua tipe data, yaitu **integer** dan **float**. Kita dapat mendeklarasikan keduanya dengan meletakkan jenis tipe data pada kurung siku “[ ]” yang dipisahkan dengan koma “,“. Berikut contohnya.

1. # tipe data non-primitif
2. numbersList:**list[int, float]**= [23, 52.3, 34]
3. fruitsTuple:**tuple[str, int, bool]**= ("Apel", "Jeruk", 10, True)
4. colorSet: **set[int, str]** = {225, "Biru", "Merah"}
5. propertyDetailDict: **dict[str, tuple[int, str]]**= {"numFloors": 1, "propertyType": "Rumah", "hasTerrace": True, "hasGarage": False}

Anda dapat menggunakan perintah **type** untuk memeriksa variabel sesuai anotasi atau tidak. Hal ini dapat Anda lakukan saat proses pengecekan atau debugging saja, ya. Berikut contohnya.

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/35173#run1-editor1)

1

name: str = "Bima"

2

age: int = 17

3

​

4

print(type(name))

5

print(type(age))

6

​

7

'''

8

Output:

9

<class 'str'>

10

<class 'int'>

11

'''

 Input  Reset

 Jalankan

### Keunggulan Menerapkan Type Annotations

Setelah belajar tentang anotasi pada Python, apakah kita harus selalu menuliskannya saat pembuatan variabel, padahal tidak perlu pun tidak masalah? Pertanyaan ini akan selalu timbul dalam benak Anda secara sadar atau tidak. Jawabannya cukup sederhana. Kita kembalikan kepada diri sendiri.

#### Dokumentasi Otomatis

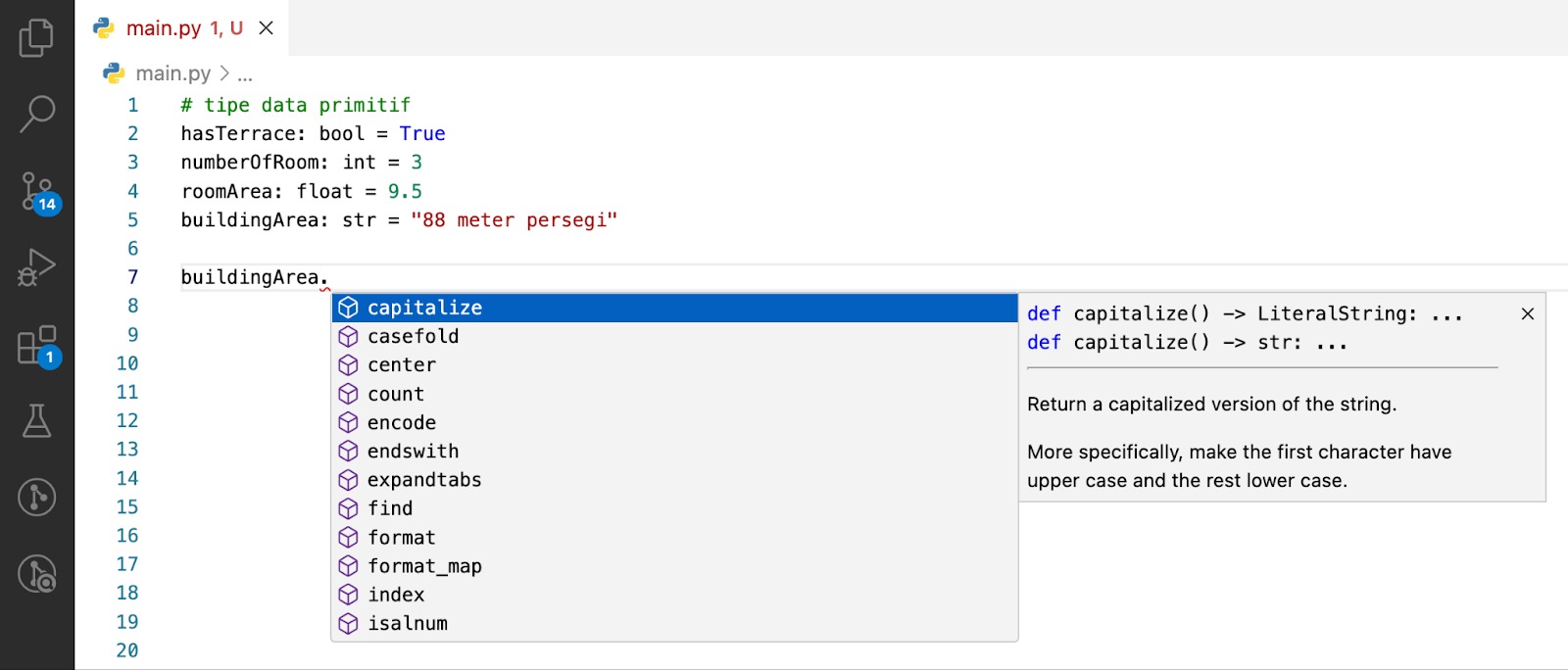
Tujuan adanya penulisan anotasi dalam pembuatan variabel adalah untuk mempermudah orang atau pengembang aplikasi lainnya dalam memahami kode. Anotasi dapat menjadi dokumentasi yang mudah untuk dibuat dan sederhana untuk dijelaskan ke pengembang aplikasi lain. Jadi, sebaiknya Anda mulai membiasakan diri untuk menulis anotasi pada variabel.

#### Autocomplete

Pada IDE, Anda diberikan berbagai macam tools untuk mendukung pengembangan aplikasi dalam berbagai macam bahasa pemrograman, salah satunya adalah Python. Penggunaan tools ini mampu mempercepat kinerja pengembang aplikasi dalam mengembangkan suatu program.

IDE menawarkan fitur autocomplete untuk menampilkan saran atau pilihan lengkap dalam proses mengetik kode. Salah satu IDE yang umum dipakai oleh pengembang aplikasi adalah Visual Studio Code. Dengan IDE tersebut, kita mampu menghemat waktu proses penulisan kode.

Dengan bantuan type annotation pada kode, Anda diberikan saran yang tepat sesuai dengan tipe datanya. Alhasil, kita dapat menjalankan operasi sesuai dengan tipe data yang diinginkan.

[](https://www.dicoding.com/academies/620/tutorials/35173)

#### Typechecker

Sebagaimana penjelasan sebelumnya, bahasa pemrograman Python adalah bahasa yang fleksibel. Anda dapat menuliskan type annotation atau tidak pada variabel, program tetap berjalan dengan semestinya.

Terkadang, kita mendeklarasikan sebuah nilai integer pada variabel tertentu. Kemudian, pada aksi berikutnya Anda melakukan proses assignment dengan memberikan harga baru bernilai string. Apakah program mengalami error? Tentu tidak. Program tetap berjalan dengan sesuai dengan aksi yang dijalankan.

1. numberOfRoom: int = 3
2. numberOfRoom = "3 ruangan"
3. print(numberOfRoom)
5. # Output: 3 ruangan

Kondisi ini yang perlu kita cegah karena variabel memiliki dua jenis tipe data yang berbeda. Oleh karena itu, kita memerlukan proses pengecekan agar kode program terhindar dari ambiguitas.

Anda dapat menginstal library **mypy** sebagai typechecker untuk memeriksa variabel dan fungsi agar sesuai dengan kaidah penulisan. Library tersebut akan memberikan peringatan kepada pengembang aplikasi ketika Anda melakukan assignment yang tidak semestinya.

Anda dapat menginstalnya melalui terminal dengan perintah berikut.

1. pip install mypy

Setelah library terinstal dengan benar, Anda dapat memeriksa kode program. Jadi, program yang sudah Anda tulis akan semakin rapi dan sesuai dengan tatanan penulisan kode Python.

Anda hanya menjalankan perintah mypy diikuti dengan nama file-nya. Secara otomatis, mypy akan menganalisis kode satu per satu untuk menemukan keganjalan.

$ **mypy main.py**

main.py:2: error: Incompatible types in assignment (expression has type "**str**", variable has type "**int**")  [assignment]

Found 1 error in 1 file (checked 1 source file)

Kesalahan pada proses assignment terindikasi dalam program. Anda dapat memperbaiki dahulu hingga sesuai dengan kaidah penulisan kode secara benar. Ketika kode sudah benar, mypy akan memberikan output “Success”. Output tersebut merepresentasikan bahwa kode yang sudah Anda tulis sudah benar.

$ **mypy main.py**

Success: no issues found in 1 source file

#### Berlatih Menemukan Error

Melalui analisis yang ketat, mypy membantu Anda mendeteksi potensi kesalahan tipe data yang mungkin terjadi, mengurangi peluang bug yang merugikan, dan waktu yang terbuang pada tahap debugging. Dengan praktik konsisten menggunakan mypy, kemampuan Anda dalam menghasilkan kode yang benar, mudah dibaca, dan mudah dipelihara akan terus berkembang. Jadi, Anda semakin waspada dalam melakukan berbagai macam aksi. Tidak hanya assignment, tetapi juga proses ekspresi, pengondisian, perulangan, dan aksi-aksi lainnya.

Materinya sangat insightful, bukan? Adanya konsep type annotation membuat Anda lebih menyadari betapa pentingnya menulis anotasi pada suatu variabel. Apabila Anda ingin mengeksplorasi lebih dalam terkait type annotation, silakan kunjungi beberapa tautan berikut.

* [Typing – Support for type hints](https://docs.python.org/3/library/typing.html).
* [Type inference and type annotations](https://mypy.readthedocs.io/en/stable/type_inference_and_annotations.html).
* [Type hints cheat sheet](https://mypy.readthedocs.io/en/stable/cheat_sheet_py3.html).

**Catatan**:

Setiap **Kuis Coding** yang akan Anda jalani memerlukan kode sesuai dengan prosedur. Agar proses submit kode berjalan dengan lancar, gunakanlah **type annotation** pada setiap variabel yang Anda buat. Dengan demikian, Anda akan terbiasa menghasilkan kode yang terstruktur dengan baik dan dokumentasi yang mudah dibaca oleh pengembang aplikasi lain.

## Rangkuman Membangun Fondasi dengan Tipe Data

Selamat! Anda sudah berada di penghujung materi terkait **Tipe Data**. Sejauh ini, Anda telah mengenal pengertian data, variabel, tipe data (primitif dan non-primitif), serta penerapan pembuatan variabel dalam bahasa pemrograman Python. Berikut adalah ringkasan materi yang sudah Anda pelajari.

* Menurut KBBI, [data](https://kbbi.kemdikbud.go.id/entri/data) dalam konteks komputer merupakan informasi dalam bentuk yang dapat diproses oleh komputer, seperti representasi digital dari teks, angka, gambar grafis, atau suara.
* Seluruh aktivitas yang kita lakukan akan berhubungan dengan data karena ia terikat dengan aksi yang dilakukan. Setiap langkah yang kita ambil dapat dihitung dan diamati berdasarkan data yang dihasilkan.
* Data tidak mungkin digunakan ketika kita tidak punya kebutuhan yang mendasarinya. Data memiliki peran penting dalam mendukung keputusan dan tindakan yang kita ambil, serta memberikan landasan kuat untuk mencapai tujuan yang diinginkan.
* Variabel dapat menyimpan data dalam bentuk boolean, angka, string, atau informasi lainnya. Variabel akan tersimpan dalam memori komputer dan dapat dipanggil serta digunakan saat kita butuhkan.
* Variabel adalah **nama** yang tersimpan dalam memori komputer, sedangkan isi dari variabel adalah **harga** (nilai) yang mengandung suatu informasi. Sebab variabel tersimpan di memori komputer, kita dapat memanggilnya kapan pun apabila kita membutuhkan informasi yang terkandung di dalamnya. Berikut adalah gambaran pendefinisian suatu variabel.  
  [](https://www.dicoding.com/academies/620/tutorials/32485?from=32480)
* Dalam bahasa pemrograman Python, mendeklarasikan sebuah variabel seperti menuliskan sebuah rumus dalam matematika. Bagian kiri merupakan nama variabel, sedangkan bagian kanan berisi nilai atau data.
  1. age = 35
  2. jobTitles = “Arsitek”
* Variabel memiliki dua sifat, yaitu **mutable** dan **immutable**.
  1. Variabel bersifat mutable adalah variabel yang dapat diubah nilainya setelah dideklarasikan. Artinya kita dapat memperbarui, mengubah, atau memodifikasi nilai yang tersimpan dalam variabel sepanjang program berjalan.
  2. Sebaliknya, variabel bersifat immutable adalah variabel yang nilainya tidak dapat diubah setelah dideklarasikan. Artinya, setelah variabel didefinisikan, nilainya akan tetap dan tidak akan berubah sepanjang program berlangsung.
* Selain sifatnya, data dibedakan menjadi dua tipe, yakni **primitif** dan **non-primitif**.
  1. Variabel bertipe data primitif adalah variabel dasar yang menjadi fondasi dalam membangun logika pemrograman. Ia meliputi boolean, numerik, dan string.
  2. Variabel bertipe data non-primitif adalah variabel tingkat lanjutan yang berupa sekumpulan data.
* **Boolean** merupakan tipe data primitif yang hanya memiliki dua nilai, yaitu “**True**” dan “**False**”. Tipe data ini dapat Anda pakai untuk mengontrol alur program, pengambilan keputusan, dan pengujian suatu kondisi.
* **Numerik** atau angka merupakan tipe data primitif yang umumnya berupa **bilangan bulat**dan**bilangan riil**. Bilangan bulat adalah angka tanpa pecahan atau desimal dan biasa kita sebut dengan integer dalam pemrograman. Sebaliknya, angka dengan pecahan atau desimal disebut dengan bilangan riil atau juga float dalam pemrograman.
* **String** adalah tipe data primitif yang digunakan untuk merepresentasikan teks atau urutan karakter. String dapat berisi kombinasi huruf, angka, simbol, dan spasi yang membentuk kata, kalimat, atau teks yang panjang.
* Berikut adalah contoh dan penerapan variabel bertipe data primitif.

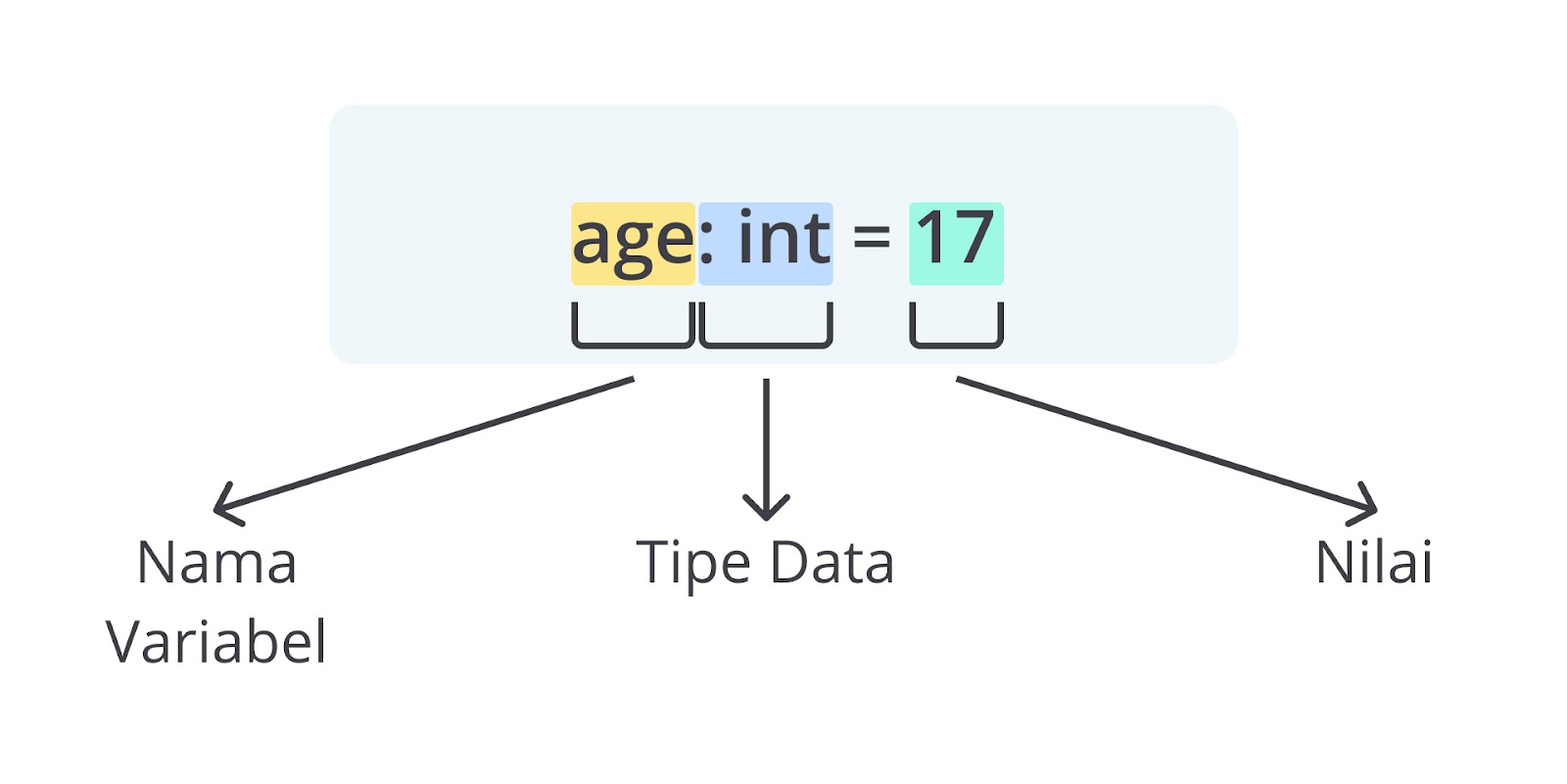
| **Tipe Data Primitif** | **Contoh Nilai** | **Deklarasi** |
| --- | --- | --- |
| Boolean | True, False | hasTerrace = True |
| Int | 1, -12, 144 | numberOfRoom = 3 |
| Float | 3.14, -12.5 | witdhOfRestroom = 1.5 |
| String | “Rumah”, ‘Jual’ | buildingArea = “88 meter persegi” |

* **List** merupakan kumpulan elemen atau nilai yang disusun dalam urutan tertentu. Isi dari tipe data ini dapat berupa string, numerik, boolean, bahkan tipe data lainnya, seperti list yang bersarang (nested list).
* **Tuple** merupakan jenis variabel bertipe data list. Namun, perbedaan utama antara tuple dan list terletak pada sifatnya yang tidak dapat diubah (immutable). Artinya, kita dapat mendefinisikan kumpulan nilai bertipe data primitif **tanpa adanya perubahan data** pada setiap elemennya.
* **Set** merupakan tipe data mutable yang berupa sekumpulan elemen unik dan tidak memiliki urutan tertentu. Dalam tipe data set, setiap elemen yang disimpan harus bersifat unik, artinya tidak ada duplikasi elemen yang diperbolehkan.
* **Dictionary** merupakan tipe data mutable yang mampu menyimpan data dalam bentuk key-value. Key diibaratkan urutan dalam list sebagai identifier dari sebuah data, sedangkan value adalah data yang ingin kita simpan. Setiap key harus bernilai string unik karena merepresentasikan referensi untuk mengakses nilai.
* Berikut adalah contoh penerapan tipe data non-primitif.

| **Tipe Data Non-primitif** | **Deklarasi** |
| --- | --- |
| List | propertyDetail = [1, "Rumah", True, False] |
| Tuple | propertyDetail = (1, "Rumah", True, False) |
| Set | propertyDetail = {1, "Rumah", True, False} |
| Dictionary | propertyDetail = {"numFloors": 1, "propertyType": "Rumah", "hasTerrace": True, "hasGarage": False} |

* Berikut adalah perbedaan mendasar antara list, tuple, set, dan dictionary.

| **Perbedaan** | **List** | **Tuple** | **Set** | **Dictionary** |
| --- | --- | --- | --- | --- |
| Mutable/Immutable | Bersifat mutable. | Bersifat immutable. | Bersifat mutable. | Bersifat mutable. |
| Karakter pengapit elemen | Tanda kurung siku ‘[ ]’. | Tanda kurung ‘( )’. | Tanda kurung kurawal ‘{ }’. | Tanda kurung kurawal ‘{ }’. |
| Operasi umum yang digunakan | Operasi penambahan, pengurangan, atau pengubahan elemen data. | Hanya mengakses elemen saja. | Operasi union (gabungan), intersection (irisan), dan difference (perbedaan). | Operasi penambahan, pengurangan, atau pengubahan elemen data. |
| Cara mengakses elemen | Panggil berdasarkan indeks. | Panggil berdasarkan indeks. | Panggil berdasarkan indeks. | Panggil berdasarkan key. |

* Dalam konteks pemrograman, anotasi adalah suatu mekanisme untuk menerangkan dan mempertegas penulisan kode.
* Anotasi dapat diibaratkan sebagai cara untuk menyatakan tipe data secara eksplisit pada suatu variabel.
* Anotasi pada tipe data dapat didefinisikan pada formula berikut.  
  [](https://www.dicoding.com/academies/620/tutorials/32485?from=32480)
* Berikut adalah beberapa anotasi yang perlu Anda tahu pada Python.

| **Tipe Data** | **Anotasi Tipe Data** |
| --- | --- |
| Boolean | bool |
| Integer | int |
| Float | float |
| String | str |
| List | list |
| Tuple | tuple |
| Set | set |
| Dictionary | dict |

* Berikut adalah keunggulan menerapkan type annotation dalam kode Anda.
  1. Mampu menulis dokumentasi secara otomatis.
  2. Mampu menerapkan autocomplete pada IDE.
  3. Mampu melatih penulisan kode melalui error yang ditemukan.