## Pengantar Nilai, Input, Output, dan Ekspresi

[](https://www.dicoding.com/academies/620/tutorials/34913?from=32490)

Dalam materi sebelumnya, Anda sudah memahami konsep dasar tipe data pada pemrograman. Mulai dari pengenalan data, variabel, tipe data (primitif dan non-primitif), hingga penerapan pembuatan variabel dalam bahasa pemrograman Python sudah Anda pelajari.

Pada materi ini, kita akan mengenal perilaku sebuah nilai, input, output, dan ekspresi dalam pemrograman. Untuk itu, Anda perlu belajar beberapa materi mengenai konsep berikut:

* pemberian nilai dan assignment dalam pembuatan variabel;
* menerima dan menampilkan data melalui eksternal ataupun internal perangkat;
* ekspresi sederhana dalam menggunakan operator aritmetika, relasional, dan logika; serta
* penerapan tingkat lanjut dalam membuat variabel.

Dari daftar di atas, apakah Anda sudah bisa membayangkan hal yang akan dipelajari? Mungkin sebelum memulai materi, kita akan melakukan senam otak. Kita sudah lama tidak melakukan senam otak sebelum memulai materi. Senam ini membawa Anda lebih mudah menggambarkan materi yang akan dibahas. Silakan amati analogi di bawah ini, ya.

Ketika Bima sedang sekolah, tiba-tiba guru mendadak memberikan ujian kepada seluruh murid di kelas. Ujian tersebut berupa soal mencongak yang harus dijawab oleh murid di kertas kosong. Dengan sigap, Bima dan teman-teman lainnya mengambil kertas kosong dan mulai mendengarkan soal yang diberikan guru. Guru memberikan lima pertanyaan yang cukup mudah. Bima dengan cakap menjawab pertanyaan dibandingkan teman lainnya karena ia rajin belajar.

[](https://www.dicoding.com/academies/620/tutorials/34913?from=32490)

Setelah selesai memberikan lima pertanyaan, seluruh murid mengumpulkan jawaban kepada guru. Guru menghabiskan waktu yang cukup banyak untuk memeriksa dan menganalisis seluruh jawaban murid. Akhirnya, guru memberikan penilaian dan feedback (umpan balik) kepada murid supaya mereka mengetahui jawabannya benar atau salah berdasarkan analisis guru.

Secara tidak langsung, kita sebagai murid di sekolah pernah mengalami kejadian hal serupa dengan yang Bima alami. Dari cerita tentang guru memberikan soal dan murid menjawab pertanyaan, kita dapat menarik pembelajaran berdasarkan cuplikan kejadian tersebut, khususnya yang dijalani oleh Bima.

Bayangkan interaksi guru dengan murid adalah program. Saat memberikan soal, secara tidak langsung guru meminta jawaban murid. Di situlah proses **input** terjadi, yakni ketika murid memberikan jawaban atas pertanyaan guru. Selama proses menunggu dan menerima jawaban, guru tidak akan bisa mengolah atau meresponsnya.

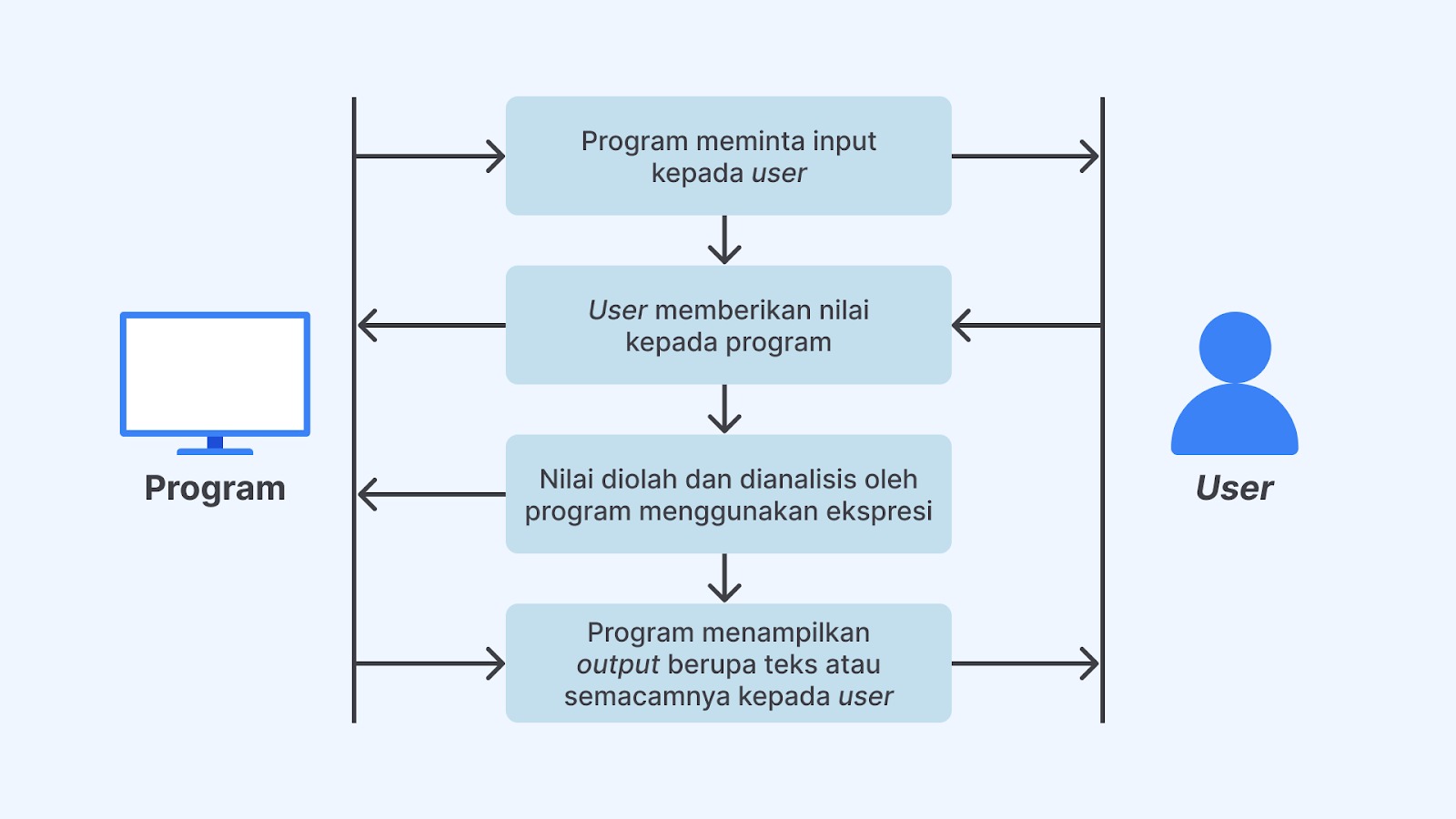
Di sisi murid, mereka memberikan sebuah jawaban kepada guru. Jawaban tersebut dapat merepresentasikan suatu **nilai**, informasi, atau hasil yang perlu diberikan kepada guru. Dengan begitu, informasi dapat diolah oleh guru secepatnya.

Setelah menjawab semua pertanyaan, seluruh murid mengumpulkan jawabannya kepada guru. Untuk memastikan bahwa jawaban dianalisis dengan teliti dan benar, guru membutuhkan waktu yang cukup banyak. Proses menganalisis dan menilai yang dilakukan oleh guru merupakan pemahaman dan keterampilan dalam menyelesaikan suatu permasalahan. Beliau mengombinasikan pernyataan atau jawaban satu dengan lainnya menggunakan pernyataan **ekspresi**, seperti logika, matematis, ataupun relasional. Semua itu dilakukan untuk memutuskan hasil akhir berupa nilai atau feedback yang komprehensif.

Setelah seluruh informasi terkumpul dan proses ekspresi dilakukan, guru memberikan nilai atau feedback secara langsung. Pada saat inilah terjadi proses timbal balik. Murid akan mendapatkan keluaran yang bermakna bagi mereka. Peristiwa ini menggambarkan **output** yang diberikan seorang guru kepada murid dalam bentuk pernyataan, nilai, atau feedback setelah melakukan pemeriksaan dan pengolahan data.

[](https://www.dicoding.com/academies/620/tutorials/34913?from=32490)

Serupa dengan proses nilai-input-ekspresi-output dalam pemrograman. Ketika membuat suatu algoritma untuk meminta nilai tertentu kepada user (pengguna), terjadi proses permintaan **input**. Setelah pengguna memberikan jawaban ke program, terjadi proses pemberian **nilai** ke program. Kemudian, program menganalisis dan mengolah menggunakan **ekspresi** tertentu. Terakhir, program akan memberikan **output** kepada user untuk menyatakan hasil akhir.

[](https://www.dicoding.com/academies/620/tutorials/34913?from=32490)

Nah, apakah Anda sudah memiliki gambaran terhadap materi yang akan dibahas? Tenang, ini hanyalah gambaran umum yang perlu Anda bayangkan supaya mudah dalam menerima materi. Sudah siap menerima materi kali ini? Tak perlu berlama-lama lagi. Yuk, kita berpindah ke materi inti.

## Harga dan Assignment

Algoritma yang kita bangun dalam suatu program dapat berupa rangkaian langkah-langkah atau instruksi yang dirancang secara sistematis. Ada batasan yang mengikat pada suatu algoritma supaya program dapat berjalan sesuai dengan tugasnya. Jadi, program pun dapat menciptakan hasil yang kita harapkan.

Namun, tak jarang batasan tersebut didefinisikan secara manual oleh pengembang aplikasi. Penentuan batasan berupa konstanta atau nilai tertentu dapat mengontrol tindakan program selama proses pengolahan data berlangsung.

Tidak hanya itu, kita mampu memanipulasi batasan tertentu secara fleksibel berdasarkan algoritma yang sedang berjalan. Untuk itu, kita perlu mengetahui cara untuk menentukan batasan dalam suatu permasalahan yang ada.

Mari kita telaah materi terkait cara menentukan batasan berdasarkan harga dan assignment dengan saksama.

### Harga

Pada materi terkait variabel, kita sudah belajar bahwa harga adalah nilai yang mengandung informasi. Harga dapat direpresentasikan sebagai suatu nilai yang bertipe data primitif atau non-primitif. Pada pemrograman, harga digunakan untuk menyimpan dan memanipulasi data yang diperlukan dalam suatu program. Ketika mendefinisikan variabel, kita memberikan harga padanya sebagai inisialisasi awal.

Harga ini diibaratkan sebagai acuan atau batasan saat kita mengeksekusi program. Mulai dari perhitungan matematis, membandingkan nilai, bahkan menjalankan logika program dapat dilakukan melalui pendefinisian harga. Dengan adanya harga, kita bisa membatasi dinamika program untuk mengikuti alur algoritma agar sesuai dengan harapan.

|  |
| --- |
| Ketika jam istirahat, Bima mampir ke kantin sekolah untuk membeli mi ayam bakso. Tujuan Bima datang ke kantin sekolah hanya untuk membeli mi ayam bakso, bukan yang lainnya. Saat sudah di depan stan mi ayam, dia berkata ke pelayan, “Bu, saya beli mi ayam bakso.” Secara otomatis, Bima memberikan harga atau batasan pemesanan kepada pelayan untuk menyediakan mi ayam bakso, bukan mi ayam telur puyuh, apalagi nasi goreng.  [Bima melihat menu kantin pada kedai mi ayam dan sedang memesan mi ayam bakso.](https://www.dicoding.com/academies/620/tutorials/34918?from=34913)  Begitu juga dengan pemrograman. Ketika menjalankan algoritma pemrograman, pastikan Anda memberikan harga sesuai dengan yang diminta. Dengan memberikan nilai atau batasan yang tepat, kita dapat memastikan bahwa program berjalan sesuai keinginan dan menghasilkan keluaran yang benar. |

Nilai dapat diperoleh dan didefinisikan melalui berbagai macam cara. Berikut adalah cara mendefinisikan nilai dalam pemrograman.

* Melalui konstanta yang diberikan ke variabel.
* Melalui hasil perhitungan suatu ekspresi.
* Melalui hasil pengembalian suatu fungsi.
* Melalui variabel yang sudah ada.

Berikut adalah contoh kode pendefinisian nilai apabila diterapkan pada pemrograman Python.

1. # Melalui konstanta nama dan harga makanan
2. food = "mi ayam bakso"
3. myMoney = 15000
4. foodPrice = 10000
6. # Melalui hasil pengurangan yang diperoleh dari ekspresi aritmetika
7. change = 15000 - 10000
9. # Melalui hasil pengembalian fungsi pengecekan uang palsu
10. isCounterfeit = counterfeitCheck()
12. # Melalui variabel foodPrice yang sudah didefinisikan sebelumnya
13. foodDiscount = foodPrice \* 15%

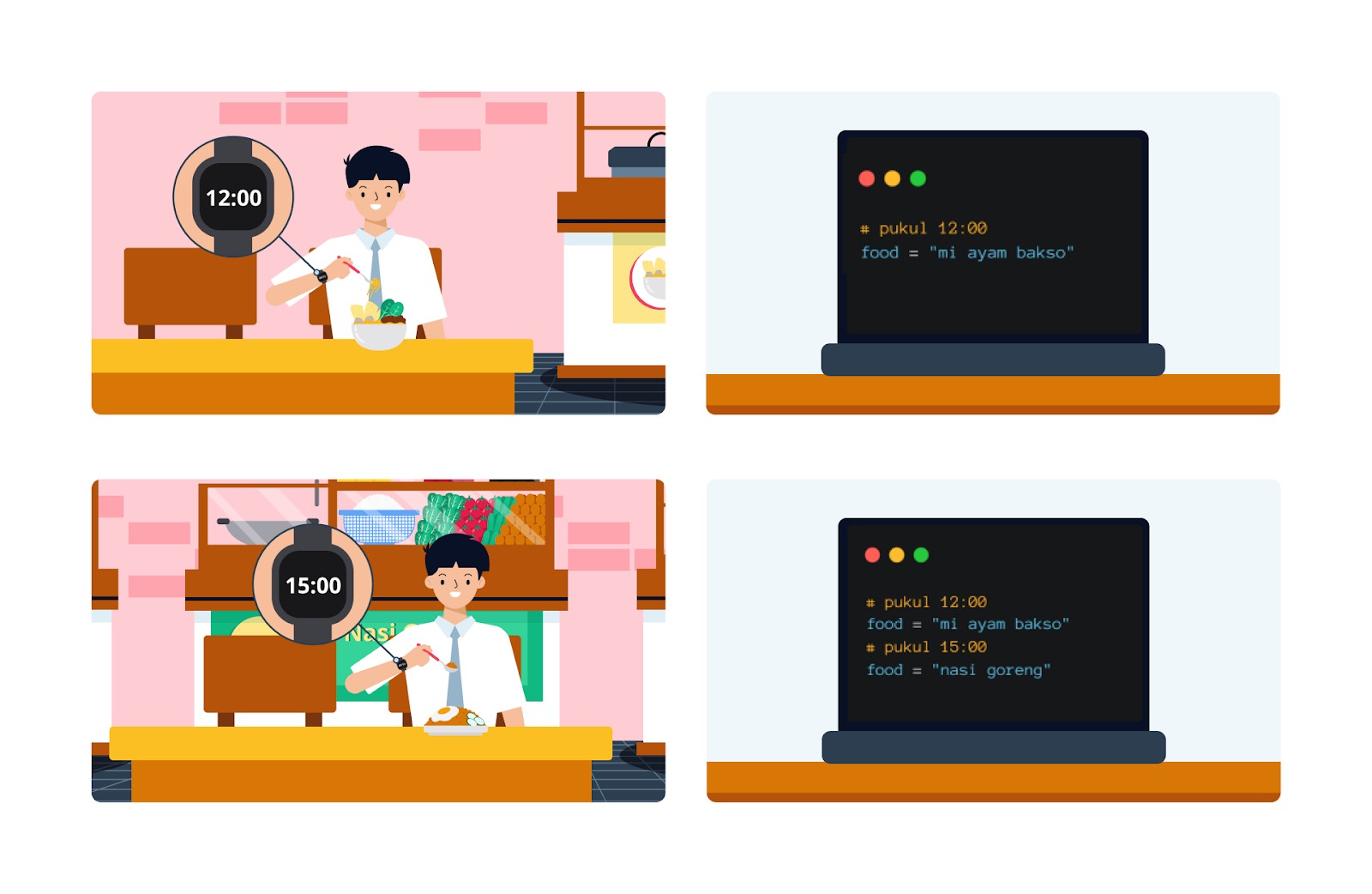
Berikut adalah penjelasan kode di atas.

* Nilai pada variabel food, myMoney, dan foodPrice diperoleh melalui konstanta statik. Ia tidak akan berubah selama konstanta tersebut melekat pada variabel tersebut.
* Nilai pada variabel change didapatkan melalui proses ekspresi aritmetika. Terjadi proses pengurangan pada dua nilai yang menimbulkan nilai baru.
* Nilai pada variabel isCounterfeit diperoleh melalui suatu fungsi. Ia mengembalikan nilai berdasarkan output yang menghasilkan nilai tertentu.
* Nilai pada variabel foodDiscount diperoleh dari proses ekspresi perkalian dari kombinasi dua nilai, pertama adalah foodPrice, nilai yang sudah didefinisikan sebelumnya, dan 15% nilai dari konstanta baru.

### Assignment

Suatu konstanta memiliki nilai tetap yang terdefinisi di awal program. Ia tidak akan berubah selama program berjalan. Namun, adakalanya nilai atau batasan suatu aksi dapat berubah seiring berjalannya program dan tidak hanya sekali, tetapi bisa berkali-kali. Oleh karena itu, nilai dapat dimanipulasi atau berubah sesuai dengan kebutuhan. Itulah alasan perlu adanya proses assignment.

Lalu, apa itu **assignment**? Jadi, **assignment** adalah instruksi primitif untuk memberikan nilai pada suatu variabel. Dalam assignment, nilai lama yang telah disimpan dalam variabel akan digantikan oleh nilai baru yang ditentukan. Dengan kata lain, assignment memperbarui variabel dengan nilai yang baru diberikan.

[](https://www.dicoding.com/academies/620/tutorials/34918?from=34913)

Sebenarnya, hal ini sudah kita lakukan saat membuat suatu variabel. Pemberian nilai ke variabel dilakukan dengan mengikuti ketentuan berikut.

* Ruas kiri adalah nama variabel.
* Ruas kanan adalah nilai atau batasan yang ingin diberikan pada variabel.

[](https://www.dicoding.com/academies/620/tutorials/34918?from=34913)

Kedua ruas tersebut dipisahkan dengan karakter sama dengan ‘=’ sebagai tanda penyerahan nilai kepada variabel. Hal ini menjadi kesepakatan suatu pemrograman yang harus Anda taati. Apabila dilanggar, Anda akan menerima pesan error dan program tidak bisa berjalan.

Mungkin Anda berpikir dan bertanya, “Apa yang terjadi ketika kita melakukan assignment pada variabel yang belum terdefinisi?” Ya, variabel yang belum terdefinisi akan mengalami proses inisialisasi awal apabila kita melakukan assignment. Pada saat itu juga variabel akan dibuat dengan nilai yang baru.

Dalam beberapa bahasa pemrograman, variabel yang belum terdefinisi akan diberi nilai default, seperti nilai nol pada data numerik atau nilai kosong pada data string. Namun, perlu Anda perhatikan bahwa tidak semua bahasa pemrograman menerapkan tindakan inisialisasi yang sama. Dalam pemrograman Python, Anda harus mendeklarasikan variabel dengan nilai atau batasan yang jelas supaya tidak terjadi error.

Mari kita simak kembali cerita Bima yang ingin makan di kantin sekolah berikut.  
Setelah memesan mi ayam bakso, Bima sadar makanannya tak kunjung datang, padahal perutnya sudah meronta-ronta. Bima pun menghampiri kembali stan mi ayam bakso.

[](https://www.dicoding.com/academies/620/tutorials/34918?from=34913)

Namun, apa jadinya bila Bima tidak fokus dan malah meminta hal yang belum dipesan? Ibarat Bima memesan makanan “mi ayam bakso”, tetapi malah meminta minuman “es teh”, padahal dia belum memesannya. Pelayan atau sistem tidak akan menemukan pesanan yang sesuai dengan permintaan. Hal ini serupa dengan pemrograman. Bila memanggil variabel yang belum terdefinisi, program akan mengalami error.

[](https://www.dicoding.com/academies/620/tutorials/34918?from=34913)

Dalam melakukan proses assignment, secara tidak langsung kita membuat sebuah nama variabel yang menyimpan nilai. Akan tetapi, tahukah Anda kalau ada aturan dalam menuliskan nama variabel? Anda tidak boleh sembarangan dalam membuat nama variabel. Dalam Python, ada beberapa aturan khusus yang perlu diikuti [6].

* Nama variabel bisa diawali dengan karakter alfabet atau underscore ‘\_’.
* Nama variabel tidak boleh diawali dengan angka.
* Nama variabel bisa mengandung karakter alfabet, angka, dan underscore.
* Nama variabel menerapkan case-sensitive atau sensitif terhadap huruf besar atau kecil.
* Nama variabel tidak boleh sama dengan nama tipe atau prosedur (aksi) pada pemrograman yang dipakai.

Meskipun banyak bahasa pemrograman memiliki aturan yang sama, Python menerapkan peraturan di atas supaya memudahkan dalam proses assignment. Untuk lebih jelasnya, berikut adalah beberapa contoh nama variabel yang benar dan salah.

| **Contoh Nama Variabel yang Benar** | **Contoh Nama Variabel yang salah** |
| --- | --- |
| 1. food = "mi ayam bakso" 2. myfood = "mi ayam bakso" 3. myFood = "mi ayam bakso" 4. my\_food = "mi ayam bakso" 5. \_my\_food = "mi ayam bakso" 6. \_myFood = "mi ayam bakso" 7. MyFood = "mi ayam bakso" 8. MYFOOD = "mi ayam bakso" 9. myfood2 = "nasi goreng" | 1. # Nama variabel diawali dengan angka 2. 2myfood = "nasi goreng" 4. # Nama variabel mengandung karakter selain alfaber, angka, dan underscore 5. my-food = "mi ayam bakso" 6. my food = "mi ayam bakso" 8. # Nama variabel sama dengan nama tipe atau prosedur pada pemrograman 9. True = "mi ayam bakso" 10. False = "mi ayam bakso" 11. and = "mi ayam bakso" |

Setelah memahami materi Harga dan Assignment, Anda memiliki wawasan lebih terkait cara menentukan nilai dan mendeklarasi variabel dalam program. Proses mendefinisikan nilai dan assignment mampu mengontrol tindakan dan membatasi alur program agar sesuai dengan keluaran yang kita harapkan.

Ada satu cara lagi untuk mendefinisikan nilai selain melalui proses assignment, yaitu membaca nilai dari device (peranti) internal atau eksternal. Hal yang paling mudah adalah meminta user memberikan nilai melalui console menggunakan perintah **input**. Mau tahu bagaimana cara penerapannya? Yuk, beralih ke materi berikutnya yang membahas input dan output. Let’s go!

## Input dan Output

Apakah kamu pernah menemukan program atau aplikasi yang interaktif? Interaksi antara pengguna dan program menjadi hal penting dalam memastikan pengalaman yang baik. Interaksi memungkinkan pengguna untuk berkomunikasi dengan program melalui memberikan input dan menerima output.

|  |
| --- |
| Saat melakukan transaksi jual beli di kantin sekolah, Bima ditanya oleh penjual mi ayam, “Ingin memesan apa?” Pertanyaan tersebut dijawab oleh Bima, “Aku ingin memesan mi ayam bakso.” Penjual akan memproses atau memasak pesanan yang diminta supaya memenuhi kebutuhan Bima. Setelah proses memasak selesai, penjual akan merespons Bima dengan cara menghidangkan pesanan mi ayam bakso.  [Bima memesan mi ayam bakso kepada pramusaji. Pramusaji segera menyiapkan pesanan dan menghidangkannya ke Bima.](https://www.dicoding.com/academies/620/tutorials/34923?from=34918)  Pada ilustrasi di atas, terjadi interaksi dalam sistem jual beli antara Bima dengan penjual mi ayam. Hal ini sama dengan sistem interaksi antara pengguna dengan program. Program dapat  menerima masukan dari pengguna berupa teks, angka, atau apa pun yang dibutuhkan. Kemudian, program akan memproses masukan berdasarkan data yang diterima. Setelah diproses, program dapat mengembalikan atau menampilkan output dengan harapan pengguna mampu memahami maksud dari sistem tersebut. |

Lalu, bagaimana program Python mengimplementasikan input dan output? Mudahnya, kita dapat menerapkan fungsi yang telah disediakan oleh Python, yaitu**input()** dan **print()**. Mari kita ulas satu per satu supaya lebih jelas.

### input()

Fungsi input() adalah perintah bawaan yang disediakan oleh Python untuk menerima masukan dari pengguna. Ketika perintah ini dipanggil, program akan terhenti sejenak menunggu pengguna memasukkan data melalui keyboard. Pengguna dapat mengirim data dengan menekan tombol “**Enter**” pada keyboard.

Lalu, apa yang diterima oleh program? Setelah pengguna mengetik dan mengirim data, program menerima data dalam bentuk string atau teks. Meskipun pengguna memasukkan data berupa angka, fungsi input menganggapnya sebagai string. Hal ini terjadi karena fungsi input() mengambil masukan sebagai teks dan mengembalikan nilai dalam bentuk string.

Berikut adalah contoh program yang menerapkan fungsi input().

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/34923?from=34918#run1-editor1)

1

total = input("Berapa porsi makan yang Anda beli?")

2

​

3

"""

4

Output:

5

Berapa porsi makan yang Anda beli? 3

6

"""

 Input  Reset

 Jalankan

|  |
| --- |
| **Catatan**:  Pada interactive code di atas, Anda dapat memanfaatkan menu input untuk memberikan masukan dengan menekan tombol **Input** hingga muncul kolom baru seperti di bawah ini.  [Ilustrasi tombol Input pada Interactive Code.](https://www.dicoding.com/academies/620/tutorials/34923?from=34918)  Anda bisa isi kolom input dengan nilai berupa string. Satu baris akan diproses oleh satu fungsi input. Apabila memiliki input lebih dari satu, Anda dapat memisahkan nilai input dengan menambahkan baris baru. Berikut contohnya.  [Ilustrasi cara kerja penggunaan input pada Interactive Code.](https://www.dicoding.com/academies/620/tutorials/34923?from=34918) |

Saat mengisi data “3”, terjadi proses assignment. Data akan berubah menjadi nilai yang diterima oleh variabel total. Secara otomatis, variabel total berisi teks “3”. Ingat, proses input hanya menerima data dalam bentuk string.

Lalu, bagaimana jika kita ingin mencatat data numerik? Sama dengan sebelumnya, seluruh data yang diterima melalui perintah input() akan menjadi sebuah string atau teks. Namun, Anda tidak perlu khawatir. Python memiliki cara untuk mengubah nilai bertipe data string menjadi numerik.

Python memiliki beberapa perintah built-in untuk mengubah nilai menjadi tipe data yang kita inginkan. Berikut perintah built-in yang bisa Anda pakai saat ingin mengonversi nilai ke tipe data tertentu.

| **Fungsi Built-in** | **Deskripsi Fungsi** | **Penerapan** |
| --- | --- | --- |
| int() | Mengonversi nilai menjadi tipe data integer. | int(3.14)  # 3  int("35")  # 35 |
| float() | Mengonversi nilai menjadi tipe data float. | float(13)  # 13.0  float("3.14")  # 3.14 |
| str() | Mengonversi nilai menjadi tipe data string. | str(3.14)  # "3.14"  str(35)  # "35" |

Apabila perintah built-in diterapkan pada program, kode akan menjadi seperti berikut.

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/34923?from=34918#run2-editor1)

1

totalFood = int(input("Berapa total pesanan yang ingin Anda pesan? "))

2

​

3

"""

4

Output:

5

Berapa total pesanan yang ingin Anda pesan? 2

6

"""

 Input  Reset

 Jalankan

Ketika program di atas dijalankan, angka “2” sebagai string yang dimasukkan oleh pengguna akan dikonversi menjadi integer.

### print()

Lalu, bagaimana kita mampu menampilkan atau merespons pengguna melalui program? Tenang, ada fungsi bawaan Python bernama print(). Ini merupakan perintah untuk mencetak teks ke console atau layar. Dengan menggunakan perintah ini, kita bisa menampilkan teks, nilai variabel, atau hasil operasi sehingga pengguna dapat melihatnya.

Inilah contoh penggunaan fungsi print() pada program Python.

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/34923?from=34918#run3-editor1)

1

food = "Mi ayam bakso"

2

​

3

print("Bima sedang memesan", food)

4

​

5

"""

6

Output:

7

Bima sedang memesan Mi ayam bakso

8

"""

 Input  Reset

 Jalankan

Anda bisa menyematkan teks, variabel, dan ekspresi tertentu pada perintah print() dengan dipisahkan oleh karakter koma ’,’ tambah ‘+’, atau persen ‘%’. Anda dapat memakai salah satu dari karakter tersebut. Namun, perlu Anda perhatikan bahwa

* karakter **koma** ‘,’ akan menggabungkan seluruh elemen yang dipisahkan oleh spasi;
* karakter **tambah** ‘+’ masih memerlukan spasi tambahan; dan
* karakter **persen** ‘%’ memerlukan tambahan operator untuk menyisipkan variabel.

Berikut adalah contoh menyematkan teks ataupun variabel dalam sebuah perintah print().

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/34923?from=34918#run4-editor1)

1

food = "Mi ayam bakso"

2

​

3

print("Bima sedang memesan", food)

4

print("Bima sedang memesan "+ food)

5

print("Bima sedang memesan %s" % (food))

6

​

7

"""

8

Output:

9

Bima sedang memesan Mi ayam bakso

10

Bima sedang memesan Mi ayam bakso

11

Bima sedang memesan Mi ayam bakso

12

​

13

"""

 Input  Reset

 Jalankan

Menyematkan variabel dengan operator persen ‘%’ memiliki cara yang unik. Setiap tipe data memiliki cara berbeda supaya nilai dapat ditampilkan pada sebuah teks. Berikut adalah cara menyisipkan variabel dengan berbagai tipe data.

* Operator %d untuk integer.
* Operator %f untuk float.
* Operator %s untuk string.
* Operator %x untuk hexadesimal.
* Operator %o untuk octal.

Seiring berjalannya waktu, Python melakukan pembaruan dalam menyisipkan variabel. Meskipun masih bisa digunakan, cara di atas (operator persen) dirasa kurang fleksibel dan sulit dibaca apabila digunakan dalam kasus yang kompleks. Oleh karena itu, Python pada versi 3 merekomendasikan cara baru yang lebih modern dan fleksibel, yaitu dengan formatting **f-string** atau metode ‘**.format()**’.

Kedua formatting tersebut memiliki konsep yang serupa dengan menyisipkan variabel melalui karakter kurung kurawal ‘{ }’. Bedanya, pada f-string, kita menandai string dengan awalan karakter ‘f’; sedangkan pada format(), konfigurasi diberikan di akhir string.

Lebih jelasnya, berikut adalah contoh penerapan formatting string menggunakan f-string dan format().

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/34923?from=34918#run5-editor1)

1

food = "Mi ayam bakso"

2

totalFood = 2

3

​

4

print(f"Bima sedang memesan {totalFood} {food}")

5

print("Bima sedang memesan {} {}".format(totalFood, food))

6

​

7

​

8

"""

9

Output:

10

Bima sedang memesan 2 Mi ayam bakso

11

Bima sedang memesan 2 Mi ayam bakso

12

"""

 Input  Reset

 Jalankan

Sebagaimana penjelasan sebelumnya bahwa penerapan formatting f-string atau format() sangat direkomendasikan dalam menyisipkan variabel karena sangat fleksibel. Penulisan angka menggunakan format yang jelas dapat diimplementasikan dengan formatting tersebut.

Misalnya, kita ingin menampilkan angka dengan jumlah digit desimal yang spesifik atau menambahkan angka di belakang koma. Metode formatting berfungsi agar kita bisa memastikan bahwa angka yang ditampilkan ke pengguna memiliki format sesuai dengan kebutuhan.

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/34923?from=34918#run6-editor1)

1

food = "Mi ayam bakso"

2

totalFood = 2

3

money = 20000

4

​

5

print(f"Bima sedang memesan {totalFood} {food} dan membayarnya dengan uang sebesar Rp{money:,}")

6

# atau

7

print("Bima sedang memesan {} {} dan membayarnya dengan uang sebesar Rp{:,}".format(totalFood, food, money))

8

​

9

​

10

"""

11

Output:

12

Bima sedang memesan 2 Mi ayam bakso dan membayarnya dengan uang sebesar Rp20,000

13

Bima sedang memesan 2 Mi ayam bakso dan membayarnya dengan uang sebesar Rp20,000

14

"""

 Input  Reset

 Jalankan

**Catatan:**  
Pada bahasa pemrograman Python, output dari penulisan ribuan dipisahkan dengan karakter koma “,”, sedangkan penulisan angka desimal menggunakan karakter titik “.”.

Saat membuat suatu sistem yang interaktif, perlu adanya algoritma yang tepat dalam menyampaikan fungsionalitas program. Ada perintah input untuk meminta masukan dari pengguna, kemudian diolah menjadi data yang akan diterima pengguna. Lalu, perintah output digunakan untuk menampilkan hasil akhirnya ke pengguna.

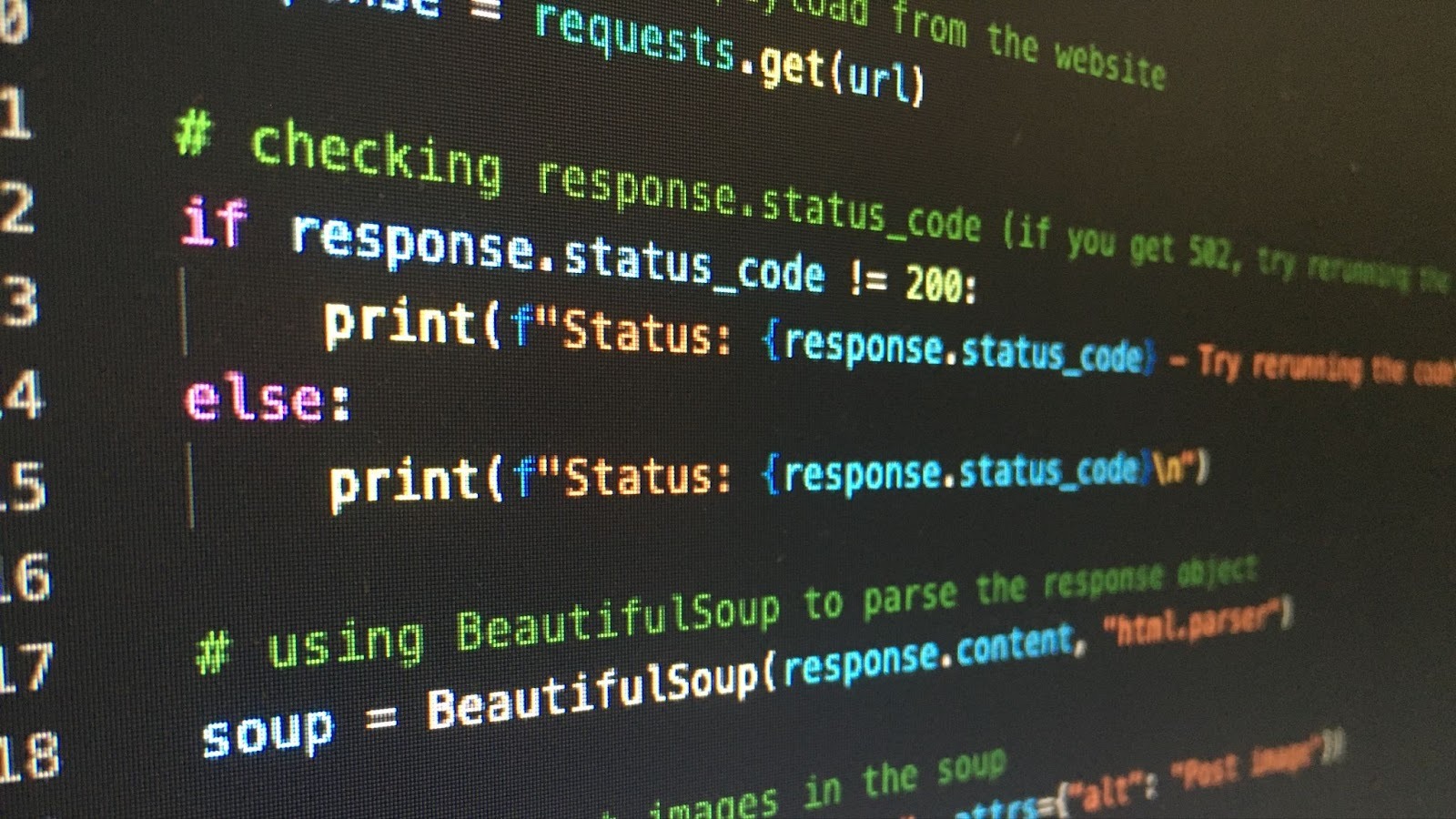
Bagaimana dengan materinya, menarik sekali, ya? Apabila Anda ingin menambah wawasan terkait dengan input dan output pada Python, silakan kunjungi tautan berikut.

* [Input and Output](https://docs.python.org/3/tutorial/inputoutput.html)
* [String - Common string operations](https://docs.python.org/3.11/library/string.html)
* [Built-in Functions](https://docs.python.org/3/library/functions.html)

## Ekspresi

Mengembangkan program, baik sederhana maupun kompleks, terkadang melibatkan lebih dari sekadar alur logis yang lurus. Alurnya mungkin terlihat mengalir begitu saja dari sisi prosedural dan algoritma.

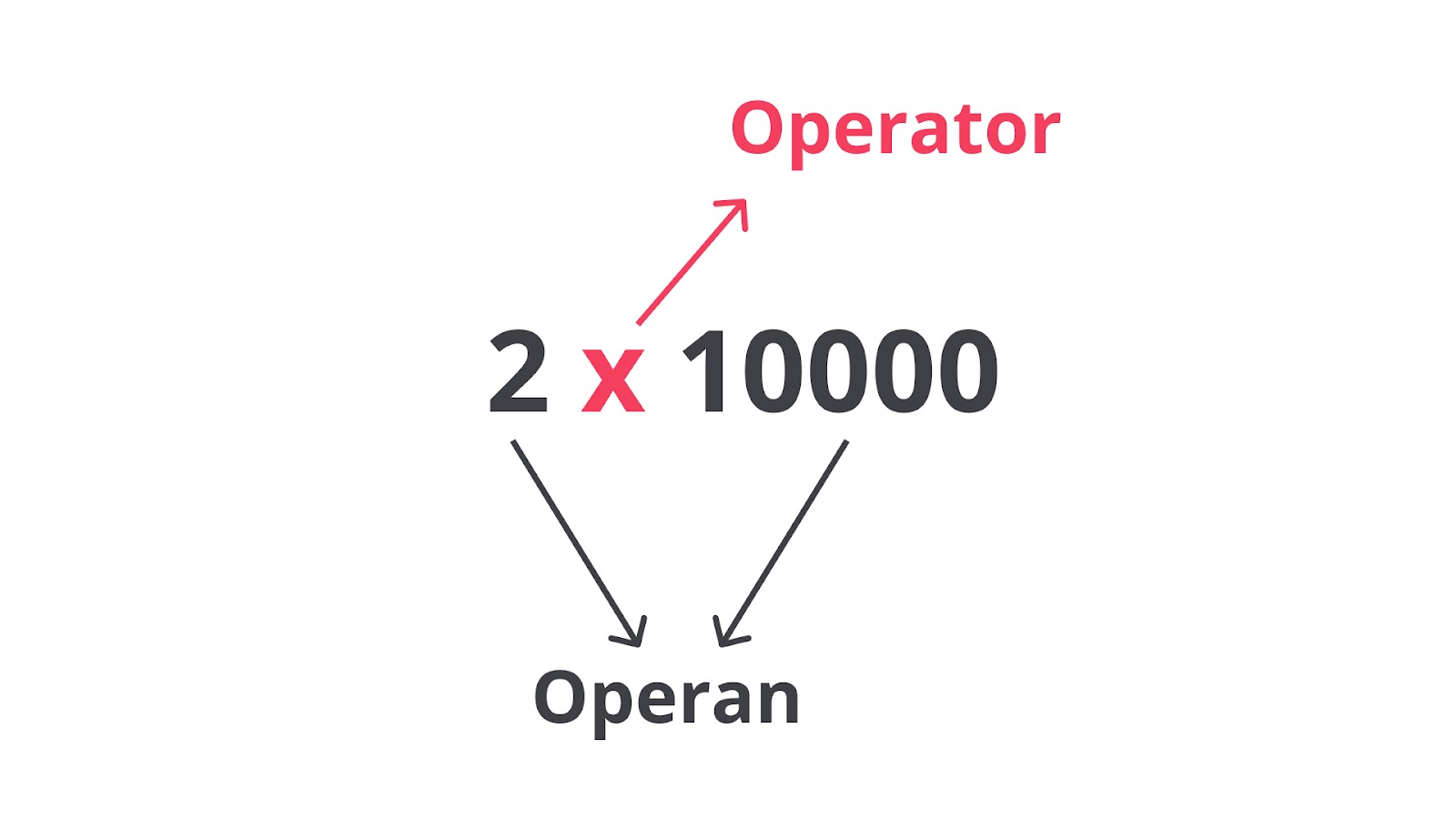
Namun, saat dikonversi dalam bahasa pemrograman ada pernyataan-pernyataan yang harus dievaluasi dan diolah. Itulah sebabnya ekspresi dapat terjadi dalam suatu program.

[](https://www.dicoding.com/academies/620/tutorials/34928?from=34923)

Maksudnya ekspresi itu apa, sih? Raut muka seseorangkah? Dalam pemrograman, jawabannya tentu bukan. Ekspresi adalah kombinasi dari nilai, variabel, operasi, dan assignment yang digunakan untuk melakukan perhitungan, pengambilan keputusan, dan manipulasi data dalam program. Dengan adanya ekspresi, kita dapat membandingkan nilai, melakukan komputasi matematis, serta membuat alur algoritma yang beragam.

|  |
| --- |
| [Bima sedang memesan dua mi ayam bakso dan menanyakan harganya. Pramusaji mi ayam pun menjawab, "Satu mi ayam bakso harganya 10 ribu rupiah. Total pembayarannya 20 ribu rupiah."](https://www.dicoding.com/academies/620/tutorials/34928?from=34923)  Dalam percakapan di atas, transaksi jual beli di kantin sekolah Bima terjadi. Saat memesan dua mi ayam bakso, secara otomatis pembayaran yang harus dibayar oleh Bima menjadi dua kali dari harga satuannya. Jika harga satuannya adalah 10 ribu rupiah, total pembayaran adalah 2 kali 10 ribu rupiah sehingga menjadi 20 ribu rupiah.  Pada saat bersamaan, penjual mi ayam sedang melakukan komputasi matematis, yakni mengalikan harga satuan dengan jumlah yang ingin dibeli Bima. Komputasi ini adalah salah satu ekspresi, yaitu operator, yang mampu mengalkulasi harga dengan operan tertentu. Dengan begitu, penjual mi dapat dengan mudah menghitung harga dan memberikan informasi akhir terkait biaya yang harus Bima bayar. |

Perlu Anda ingat bahwa ada tata cara untuk menulis suatu ekspresi. Apakah begitu terikat dengan aturan sampai ada tata caranya? Algoritma yang kita tulis dikonversi ke bahasa pemrograman, pasti ada rule yang harus dipatuhi. Struktur ekspresi umumnya memiliki **dua operan** yang dipisahkan oleh **satu operator**.

[](https://www.dicoding.com/academies/620/tutorials/34928?from=34923)

Sebagaimana yang kita ketahui bahwa ekspresi digunakan untuk melakukan proses pengambilan keputusan, perhitungan, dan manipulasi data dalam suatu program. Namun, pada materi sebelumnya, kita sudah membahas terkait manipulasi data string dengan perintah print.

Kita dapat membandingkan satu atau dua nilai yang menghasilkan nilai boolean menggunakan operator logika dan relasional. Operator logika adalah membandingkan satu atau dua operan bertipe boolean, sedangkan operator relasional adalah membandingkan dua operan bertipe numerik atau string.

### Operator Logika

Tujuan dari adanya operator logika adalah untuk menguji suatu kondisi antara boolean satu dan lainnya. Berikut adalah operator logika yang dapat Anda pakai.

| **Operator** | **Deskripsi** |
| --- | --- |
| and | Logika **and** yang hanya menghasilkan True, jika kedua operan bernilai True. |
| or | Logika **or** yang menghasilkan True, jika salah satu dari kedua operan bernilai True. |
| not | Logika **not** atau negasi bertujuan untuk membalikkan nilai logika dari operannya. |

Berikut adalah tabel kebenaran apabila operator logika diterapkan dalam suatu operasi boolean.

| **Operasi** | **Hasil** |
| --- | --- |
| True and True | True |
| True and False | False |
| False and True | False |
| False and False | False |
| True or True | True |
| True or False | True |
| False or True | True |
| False or False | False |
| not True | False |
| not False | True |

Masih bingung? Mari lihat kode di bawah ini.

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/34928?from=34923#run1-editor1)

1

print("Operator and")

2

print(True and True)

3

print(True and False)

4

print(False and True)

5

print(False and False)

6

print("Operator or")

7

print(True or True)

8

print(True or False)

9

print(False or True)

10

print(False or False)

11

print("Operator not")

12

print(not True)

13

print(not False)

14

​

15

'''

16

Output:

17

Operator and

18

True

19

False

20

False

 Input  Reset

 Jalankan

### Operator Relasional

Ada beberapa tipe data yang dapat menerapkan operator relasional, yaitu numerik dan string. Operator ini digunakan untuk membandingkan dua nilai. Dengan adanya operator ini, kita dapat menjalankan proses pengambilan keputusan berdasarkan perbandingan dua nilai.

Berikut adalah operator relasional yang umum digunakan saat membandingkan dua nilai.

| **Operator (Numerik)** | **Deskripsi** |
| --- | --- |
| Sama dengan (==) | Menghasilkan True, jika kedua operan bernilai sama. |
| Tidak Sama dengan (!=) | Menghasilkan True, jika kedua operan tidak bernilai sama. |
| Lebih Besar dari (>) | Menghasilkan True, jika operan kiri lebih besar dari operan kanan. |
| Kurang dari (<) | Menghasilkan True, jika operan kanan lebih besar dari operan kiri. |
| Lebih Besar dari Sama dengan (>=) | Menghasilkan True, jika operan kiri lebih besar atau sama dengan operan kanan. |
| Kurang dari Sama dengan (<=) | Menghasilkan True, jika operan kanan lebih besar atau sama dengan operan kiri. |

Berikut penerapan operator relasional pada data bertipe numerik.

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/34928?from=34923#run2-editor1)

1

print(10 == 4)

2

print(10 != 4)

3

print(10 > 4)

4

print(10 < 4)

5

print(10 >= 4)

6

print(10 <= 4)

7

​

8

'''

9

Output:

10

False

11

True

12

True

13

False

14

True

15

False

16

'''

 Input  Reset

 Jalankan

Mungkin Anda penasaran, bagaimana dengan tipe data string? Secara logika, tipe data string dapat menggunakan seluruh operator relasional. Cara membandingkan data numerik dan string cukup berbeda. Program menerapkan urutan leksikografis untuk membandingkan dua nilai string, yaitu berdasarkan urutan karakter dalam alfabet. Dengan begitu, program tetap menghasilkan keluaran berupa boolean.

| **Operator (String)** | **Deskripsi** |
| --- | --- |
| Sama dengan (==) | Menghasilkan True, jika kedua operan bernilai sama. |
| Tidak Sama dengan (!=) | Menghasilkan True, jika kedua operan tidak bernilai sama. |
| Lebih Besar dari (>) | Membandingkan urutan leksikografis (berdasarkan abjad) antara dua string dan memeriksa bahwa string pertama lebih besar dari string kedua.  Menghasilkan True, jika urutan leksikografis string pertama lebih besar dari string kedua. |
| Kurang dari (<) | Menghasilkan True, jika urutan leksikografis string pertama lebih kecil dari string kedua. |
| Lebih Besar dari Sama dengan (>=) | Menghasilkan True, jika urutan leksikografis string pertama lebih besar atau sama dengan string kedua. |
| Kurang dari Sama dengan (<=) | Menghasilkan True, jika urutan leksikografis string pertama lebih kecil atau sama dengan string kedua. |

Berikut contoh kode Python apabila string menerapkan operator relasional.

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/34928?from=34923#run3-editor1)

1

print("Arsitek" == "Pelajar")

2

print("Arsitek" != "Pelajar")

3

print("Arsitek" > "Pelajar")

4

print("Arsitek" < "Pelajar")

5

print("Arsitek" >= "Pelajar")

6

print("Arsitek" <= "Pelajar")

7

​

8

'''

9

Output:

10

False

11

True

12

False

13

True

14

False

15

True

16

'''

 Input  Reset

 Jalankan

### Operator Aritmetika

Dalam pemrograman, operasi matematis hanya dapat dilakukan pada data bertipe numerik, seperti penjumlahan, pengurangan, perkalian, pembagian, dan masih banyak lainnya. Operasi matematis juga memerlukan sebuah operator yang merepresentasikan proses perhitungan dua bilangan.

Berikut adalah operator aritmetika yang umum digunakan saat melakukan operasi matematis.

| **Operator** | **Deskripsi** |
| --- | --- |
| Penjumlahan (+) | Menambahkan nilai dari kedua operan. |
| Pengurangan (-) | Mengurangi nilai dari kedua operan. |
| Perkalian (\*) | Mengalikan nilai dari kedua operan. |
| Pembagian riil (/) | Membagi nilai dari kedua operan. Jika operan adalah float, hasil operasi adalah bilangan riil. |
| Modulo (%) | Sisa hasil pembagian nilai dari dua operan. |
| Pangkat (\*\*) | Memangkatkan nilai dari dua operan. |

Berikut adalah contoh kode Python apabila operasi aritmetika diterapkan pada data numerik.

* [**main.py**](https://www.dicoding.com/academies/620/tutorials/34928?from=34923#run4-editor1)

1

print(10 + 4)

2

print(10 - 4)

3

print(10 \* 4)

4

print(10 / 4)

5

print(10 % 4)

6

print(10 \*\* 4)

7

​

8

'''

9

Output:

10

14

11

6

12

40

13

2.5

14

2

15

10000

16

'''

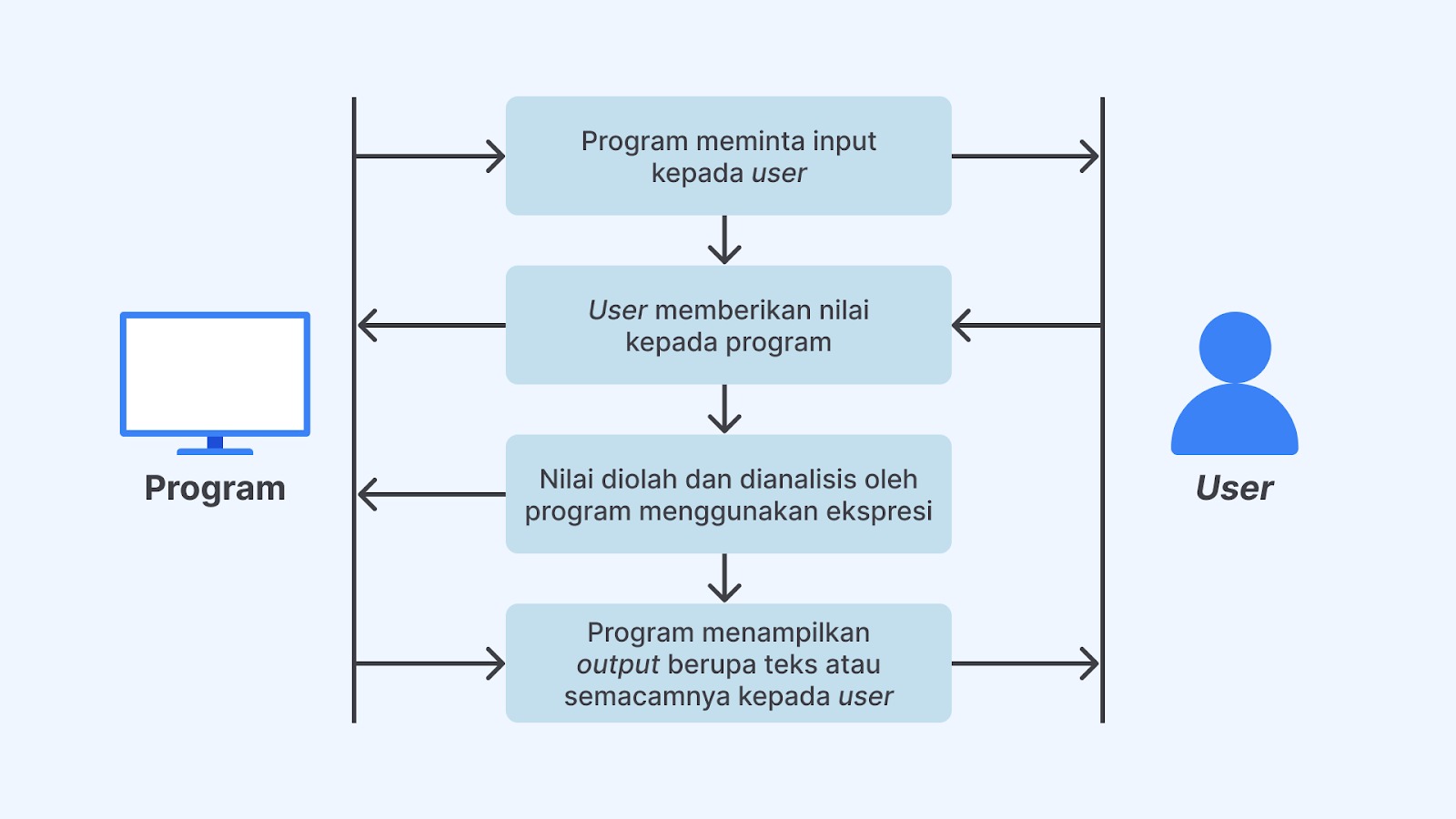
 Input  Reset

 Jalankan

Dengan ekspresi, kita dapat melakukan perhitungan, membandingkan pernyataan, dan memanipulasi teks dalam suatu program. Pemahaman materi ekspresi dalam pemrograman prosedural menjadi hal yang penting dilakukan bagi pengembang aplikasi. Hal ini karena ekspresi ini akan sering dipakai lebih dalam pada pernyataan kondisional dan perulangan. Kedua hal tersebut materi tersebut akan dibahas pada kelas ini. So, stay tuned!

## Rangkuman Berinteraksi dengan Nilai, Input, Output, dan Ekspresi

Selamat! Anda sudah berada di penghujung materi terkait Nilai, Input, Output, dan Ekspresi. Sejauh ini, Anda telah mengenal cara berinteraksi antara program dan pengguna melalui proses assignment, input, output, dan komputasi ekspresi. Berikut adalah ringkasan materi yang sudah Anda pelajari.

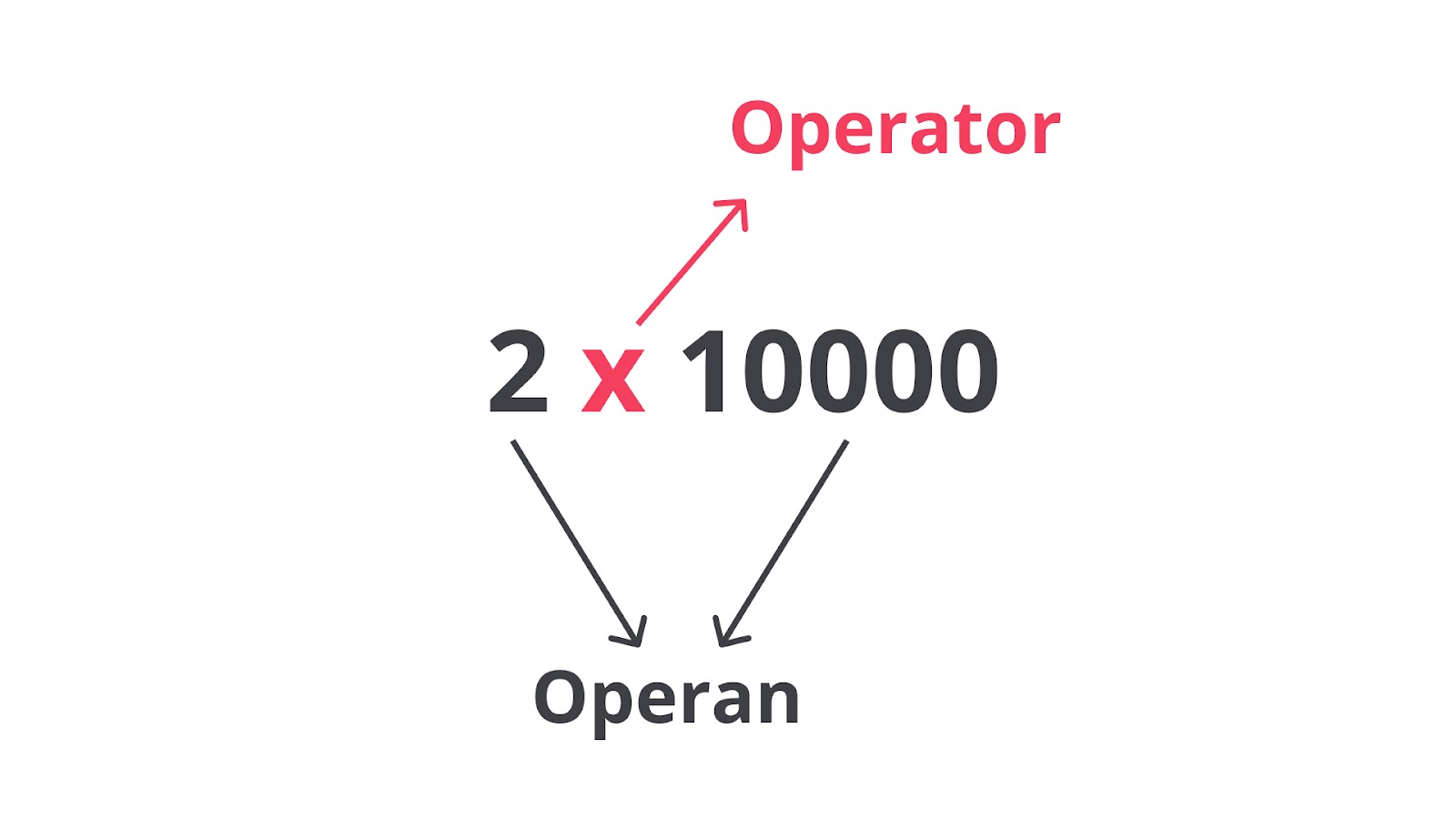
* Hadirnya interaksi memungkinkan pengguna berkomunikasi dengan program melalui pemberian masukan dan menampilkan data.
* Ketika membuat suatu algoritma untuk meminta nilai tertentu kepada pengguna, terjadi proses permintaan **input**. Setelah pengguna memberikan jawaban ke program, terjadi proses pemberian **nilai** ke program. Kemudian, program menganalisis dan mengolah menggunakan **ekspresi** tertentu. Terakhir, program akan memberikan **output** kepada user untuk menyatakan hasil akhir.  
  [](https://www.dicoding.com/academies/620/tutorials/34943?from=34938)
* Penentuan batasan algoritma bisa berupa konstanta atau nilai tertentu yang dapat mengontrol tindakan program selama proses pengolahan data berlangsung. Tidak hanya itu, kita mampu memanipulasi batasan secara fleksibel berdasarkan algoritma yang sedang berjalan.
* Harga adalah nilai atau batasan saat kita mengeksekusi program. Mulai dari perhitungan matematis, membandingkan nilai, bahkan menjalankan logika program dapat dilakukan melalui pendefinisian harga. Dengan adanya harga, kita bisa membatasi dinamika program agar mengikuti alur algoritma yang sesuai dengan harapan.
  + Nilai dapat diperoleh dan didefinisikan melalui berbagai macam cara. Berikut adalah cara mendefinisikan nilai dalam pemrograman.
  + Melalui konstanta yang diberikan ke variabel.
  + Melalui hasil perhitungan suatu ekspresi.
  + Melalui hasil pengembalian suatu fungsi.
  + Melalui konstanta tanpa diberikan ke variabel tertentu.  
      
    Berikut adalah contoh kode pendefinisian nilai apabila diterapkan pada pemrograman Python.
    1. # Melalui konstanta nama makanan dan jumlah uang yang diberikan ke variabel
    2. food = "mi ayam bakso"
    3. myMoney = 15000
    4. foodPrice = 12000
    6. # Melalui hasil perhitungan kembalian menggunakan ekspresi aritmetika
    7. change = myMoney - foodPrice
    9. # Melalui hasil pengembalian fungsi pengecekan uang palsu
    10. isCounterfeit = counterfeitCheck()
    12. # Melalui konstanta diskon tanpa diberikan ke variabel tertentu
    13. foodDiscount = foodPrice \* 15%
* Assignment adalah instruksi primitif untuk memberikan nilai pada suatu variabel. Dalam assignment, nilai lama yang mungkin telah disimpan pada variabel akan digantikan oleh nilai baru yang ditentukan. Kata lainnya, assignment memperbarui nilai variabel dengan nilai yang baru diberikan.
* Pemberian nilai dari suatu nilai pada variabel dilakukan dengan mengikuti ketentuan berikut.
  + Ruas kiri adalah nama variabel.
  + Ruas kanan adalah nilai atau batasan yang ingin diberikan pada variabel.
  + Kedua ruas dipisahkan dengan karakter sama dengan ‘=’ sebagai tanda penyerahan nilai kepada variabel.  
    [](https://www.dicoding.com/academies/620/tutorials/34943?from=34938)
* “Apa yang terjadi ketika kita melakukan assignment pada variabel yang belum terdefinisi?” Ya, variabel yang belum terdefinisi akan mengalami proses inisialisasi awal apabila kita melakukan assignment. Pada saat itu juga variabel akan dibuat dengan nilai yang baru.
* Dalam beberapa bahasa pemrograman, variabel yang belum terdefinisi akan diberi nilai default, seperti nilai nol pada data numerik atau nilai kosong pada data string. Dalam pemrograman Python, Anda harus mendeklarasikan variabel dengan nilai atau batasan yang jelas supaya tidak terjadi error.
* Dalam pemrograman Python, ada beberapa peraturan khusus yang perlu diikuti untuk membuat nama variabel.
  + Nama variabel bisa diawali dengan karakter alfabet atau underscore ‘\_’.
  + Nama variabel tidak boleh diawali dengan angka.
  + Nama variabel bisa mengandung karakter alfabet, angka, dan underscore.
  + Nama variabel menerapkan case-sensitive atau sensitif terhadap huruf besar atau kecil.
  + Nama variabel tidak boleh sama dengan nama tipe atau prosedur (aksi) pada pemrograman yang dipakai.

| **Contoh Nama Variabel yang Benar** | **Contoh Nama Variabel yang Salah** |
| --- | --- |
| * + 1. food = "mi ayam bakso"     2. myfood = "mi ayam bakso"     3. myFood = "mi ayam bakso"     4. my\_food = "mi ayam bakso"     5. \_my\_food = "mi ayam bakso"     6. \_myFood = "mi ayam bakso"     7. MyFood = "mi ayam bakso"     8. MYFOOD = "mi ayam bakso"     9. myfood2 = "nasi goreng" | * + 1. 2myfood = "nasi goreng"     2. my-food = "mi ayam bakso"     3. my food = "mi ayam bakso"     4. True = "mi ayam bakso"     5. False = "mi ayam bakso"     6. and = "mi ayam bakso" |

* Fungsi input() adalah perintah bawaan yang disediakan oleh Python untuk menerima masukan dari pengguna.
* Ketika perintah input() dipanggil, program akan terhenti sejenak menunggu pengguna memasukkan data melalui keyboard. Pengguna dapat mengirim data dengan menekan tombol “**Enter**” pada keyboard.
* Setelah pengguna mengetik dan mengirim data, program menerima data dalam bentuk string atau teks. Meskipun pengguna memasukkan data berupa angka, fungsi input menganggapnya sebagai string. Hal ini terjadi karena fungsi input() mengambil masukan sebagai teks dan mengembalikan nilai dalam bentuk string.
  + total = input("Berapa porsi makan yang Anda beli?")
  + """
  + Output:
  + Berapa porsi makan yang Anda beli? 3
  + """
* Python memiliki beberapa perintah built-in untuk mengubah nilai menjadi tipe data yang kita inginkan. Berikut perintah built-in yang bisa Anda pakai saat ingin mengonversi nilai ke tipe data tertentu.

| **Perintah Built-in** | **Fungsi** | **Penerapan** |
| --- | --- | --- |
| int() | Mengonversi nilai menjadi tipe data integer. | int(3.14)  # 3  int("35")  # 35 |
| float() | Mengonversi nilai menjadi tipe data float. | float(13)  # 13.0  float("3.14")  # 3.14 |
| str() | Mengonversi nilai menjadi tipe data string. | str(3.14)  # "3.14"  str(35)  # "35" |

* Perintah print() merupakan perintah untuk mencetak teks ke consoleatau layar. Dengan menggunakan perintah ini, kita bisa menampilkan teks, nilai variabel, atau hasil operasi sehingga pengguna dapat melihatnya.
  + food = "Mi ayam bakso"
  + print("Bima sedang memesan", food)
  + """
  + Output:
  + Bima sedang memesan Mi ayam bakso
  + """
* Anda bisa menyematkan teks, variabel, dan ekspresi tertentu pada perintah print() dengan dipisahkan oleh karakter koma ’,’ tambah ‘+’, atau persen ‘%’. Anda dapat memakai salah satu dari karakter tersebut. Namun, perlu Anda perhatikan bahwa
  + karakter **koma** ‘,’ akan menggabungkan seluruh elemen yang dipisahkan oleh spasi;
  + karakter **tambah** ‘+’ masih memerlukan spasi tambahan; dan
  + karakter **persen**‘%’ memerlukan tambahan operator untuk menyisipkan variabel.
    1. food = "Mi ayam bakso"
    3. print("Bima sedang memesan", food)
    4. print("Bima sedang memesan "+ food)
    5. print("Bima sedang memesan %s" % (food))
    7. """
    8. Output:
    9. Bima sedang memesan Mi ayam bakso
    10. Bima sedang memesan Mi ayam bakso
    11. Bima sedang memesan Mi ayam bakso
    12. """
* Kedua formatting**f-string** dan ‘**.format()**’ tersebut memiliki konsep yang serupa dengan menyisipkan variabel melalui karakter kurung kurawal ‘{ }’. Bedanya, f-string dapat dilakukan dengan menandai string yang diawali karakter ‘f’, sedangkan format() dengan memberikan konfigurasi di akhir string.
  + food = "Mi ayam bakso"
  + totalFood = 2
  + print(f"Bima sedang memesan {totalFood} {food}")
  + print("Bima sedang memesan {} {}".format(totalFood, food))

  + """
  + Output:
  + Bima sedang memesan 2 Mi ayam bakso
  + Bima sedang memesan 2 Mi ayam bakso
  + """
* Ekspresi adalah kombinasi dari nilai, variabel, operasi, dan assignment yang digunakan untuk melakukan perhitungan, pengambilan keputusan, dan manipulasi data dalam program. Dengan adanya ekspresi, kita dapat membandingkan nilai, melakukan komputasi matematis, serta membuat alur algoritma yang beragam.
* Struktur ekspresi umumnya memiliki **dua operan** yang dipisahkan oleh **satu operator**.  
  [](https://www.dicoding.com/academies/620/tutorials/34943?from=34938)
* Dalam boolean, ada operator logika yang memungkinkan kita melakukan operasi logika dengan membandingkan dua operan bertipe boolean. Tujuannya adalah untuk menguji suatu kondisi antara boolean satu dan lainnya. Berikut adalah operator logika yang dapat dipakai.

| **Operator** | **Deskripsi** |
| --- | --- |
| and | Logika **and** yang hanya menghasilkan True, jika kedua operan bernilai True. |
| or | Logika **or** yang menghasilkan True, jika salah satu dari kedua operan bernilai True. |
| not | Logika **not** atau negasi bertujuan untuk membalikkan nilai logika dari operannya. |

* Dalam tipe numerik, ada operator aritmetika yang mampu melakukan operasi matematis, seperti penjumlahan, pengurangan, perkalian, pembagian, dan masih banyak lainnya. Berikut adalah operator aritmetika yang umum digunakan saat melakukan operasi matematis.

| **Operator (numerik)** | **Deskripsi** | **Contoh** | **Hasil** |
| --- | --- | --- | --- |
| Penjumlahan (+) | Menambahkan nilai dari kedua operan. | 10 + 4 | 14 |
| Pengurangan (-) | Mengurangi nilai dari kedua operan. | 10 - 4 | 6 |
| Perkalian (\*) | Mengalikan nilai dari kedua operan. | 10 \* 4 | 40 |
| Pembagian Riil (/) | Membagi nilai dari kedua operan. Jika operan adalah float, hasil operasi adalah  bilangan riil. | 10 / 4 | 2.5 |
| Modulo (%) | Sisa hasil pembagian nilai dari dua operan. | 10 % 4 | 2 |
| Pangkat (\*\*) | Memangkatkan nilai dari dua operan. | 10 \*\* 4 | 10000 |

* Dalam string, ada operator relasional yang dapat melakukan operasi pembanding antara dua string layaknya tipe data numerik. Operator ini berguna untuk membandingkan dua nilai string dalam proses pengambilan keputusan. Berikut adalah operator relasional yang digunakan untuk membandingkan dua nilai string.

| **Operator (String)** | **Deskripsi** | **Contoh** | **Hasil** |
| --- | --- | --- | --- |
| Sama dengan (==) | Menghasilkan True, jika kedua operan bernilai sama. | “Arsitek” == “Pelajar” | False |
| Tidak Sama dengan (!=) | Menghasilkan True, jika kedua operan tidak bernilai sama. | “Arsitek” != “Pelajar” | True |
| Lebih Besar dari (>) | Membandingkan urutan leksikografis (urutan berdasarkan abjad) antara dua string dan memeriksa apakah string pertama lebih besar dari string kedua.  Menghasilkan True, urutan leksikografis string pertama lebih besar dari string kedua. | “Arsitek” > “Pelajar” | False |
| Kurang dari (<) | Menghasilkan True, urutan leksikografis string pertama lebih kecil dari string kedua. | “Arsitek” < “Pelajar” | True |
| Lebih Besar dari Sama dengan (>=) | Menghasilkan True, urutan leksikografis string pertama lebih besar atau sama dengan string kedua. | “Arsitek” >= “Pelajar” | False |
| Kurang dari Sama dengan (<=) | Menghasilkan True, urutan leksikografis string pertama lebih kecil atau sama dengan string kedua. | “Arsitek” <= “Pelajar” | True |

* Selain operator aritmetika, tipe data numerik dapat menerapkan operator relasional yang mampu melakukan operasi pembanding. Tujuannya adalah untuk membandingkan dua nilai numerik dan menghasilkan nilai boolean. Berikut adalah operator relasional yang umum digunakan saat membandingkan dua nilai numerik.

| **Operator (Numerik)** | **Deskripsi** | **Contoh** | **Hasil** |
| --- | --- | --- | --- |
| Sama dengan (==) | Menghasilkan True, jika kedua operan bernilai sama. | 10 == 4 | False |
| Tidak Sama dengan (!=) | Menghasilkan True, jika kedua operan tidak bernilai sama. | 10 != 4 | True |
| Lebih Besar dari (>) | Menghasilkan True, jika operan kiri lebih besar dari operan kanan. | 10 > 4 | True |
| Kurang dari (<) | Menghasilkan True, jika operan kanan lebih besar dari operan kiri. | 10 < 4 | False |
| Lebih Besar dari Sama dengan (>=) | Menghasilkan True, jika operan kiri lebih besar atau sama dengan operan kanan. | 10 >= 4 | True |
| Kurang dari Sama dengan (<=) | Menghasilkan True, jika operan kanan lebih besar atau sama dengan operan kiri. | 10 <= 4 | False |