## Pengantar Pemrograman Prosedural

[](https://www.dicoding.com/academies/620/tutorials/32430?from=32425)

Pada materi sebelumnya, Anda sudah memahami perbedaan antara program dan bahasa pemrograman. Mulai dari pengertian dan perbedaannya, penjelasan paradigma pemrograman yang ada, bahasa pemrograman berdasarkan paradigma, hingga perspektif pengembang aplikasi dalam membuat program sudah Anda pelajari.

Pada materi ini, kita akan belajar lebih dalam mengenai pemrograman prosedural. Untuk itu, Anda perlu belajar beberapa materi:

* penggambaran konsep dasar pemrograman prosedural;
* pengenalan prinsip dasar algoritma dalam pemrograman prosedural; dan
* interpretasi algoritma dalam suatu pemrograman.

Penasaran? Yuk, kita masuk ke materi selanjutnya.

### Belajar Pemrograman Prosedural

Pemrograman prosedural merupakan program yang disusun secara struktural atau langkah demi langkah untuk mencapai keluaran yang diharapkan. Setiap instruksi harus dipikirkan dari awal hingga akhir dan akan dieksekusi satu per satu secara sekuensial.

Mungkin dalam benak Anda memiliki beberapa pertanyaan mengenai pemrograman prosedural seperti berikut.

* Mengapa kita perlu belajar lebih dalam mengenai paradigma prosedural?
* Begitu pentingkah belajar pemrograman prosedural dibandingkan dengan paradigma lainnya?
* Kapan waktu yang tepat bagi kita untuk menerapkan pemrograman prosedural?

|  |
| --- |
| Pada saat Ibu Fika sibuk memasak untuk seluruh anggota keluarganya, beliau meminta Bapak Fajar membelikan bahan masakan di pasar. Ibu Fika berkata dengan jelas kepada Bapak Fajar, “Pergilah ke pasar untuk membeli 1 saset garam, 1 kg gula pasir, dan 2 ikat sayur bayam. Selepas itu, mampirlah ke apotek untuk membeli 1 botol obat batuk.” Bapak Fajar akan mengingatnya dan membeli kebutuhan yang ibu perlukan.  [Bapak Fajar sedang memegang catatan dan menghafal daftar belanjaan.](https://www.dicoding.com/academies/620/tutorials/32430?from=32425)  Tidak jarang orang tua menyuruh kita untuk membelikan kebutuhan, seperti yang dilakukan Ibu Fika. Apabila berada di posisi Bapak Fajar, apa yang akan Anda lakukan? Kita akan pergi ke pasar untuk membeli kebutuhan masak, kemudian singgah ke apotek untuk membeli obat batuk. Setelah itu, kita kembali ke rumah dengan barang-barang yang Ibu butuhkan.  Melihat cerita di atas, hal yang kita lakukan pada saat membeli barang-barang tersebut merupakan aksi secara prosedural. Langkah demi langkah kita lakukan sesuai dengan hal yang dibutuhkan. Setelah pergi ke pasar, Anda akan pergi ke apotek, dan kembali dengan barang yang diinginkan. Anda melakukan setiap aksi secara berurutan. |

Salah satu alasan kita belajar pemrograman prosedural, yaitu memecahkan dan menyelesaikan masalah secara berurutan. Kita akan lebih mudah menyelesaikan permasalahan dalam bentuk program karena terbiasa menjalankan aksi secara beruntun. Hal ini memungkinkan kita untuk mengasah cara berpikir secara runtut, terorganisasi, dan memahami alur jalannya program.

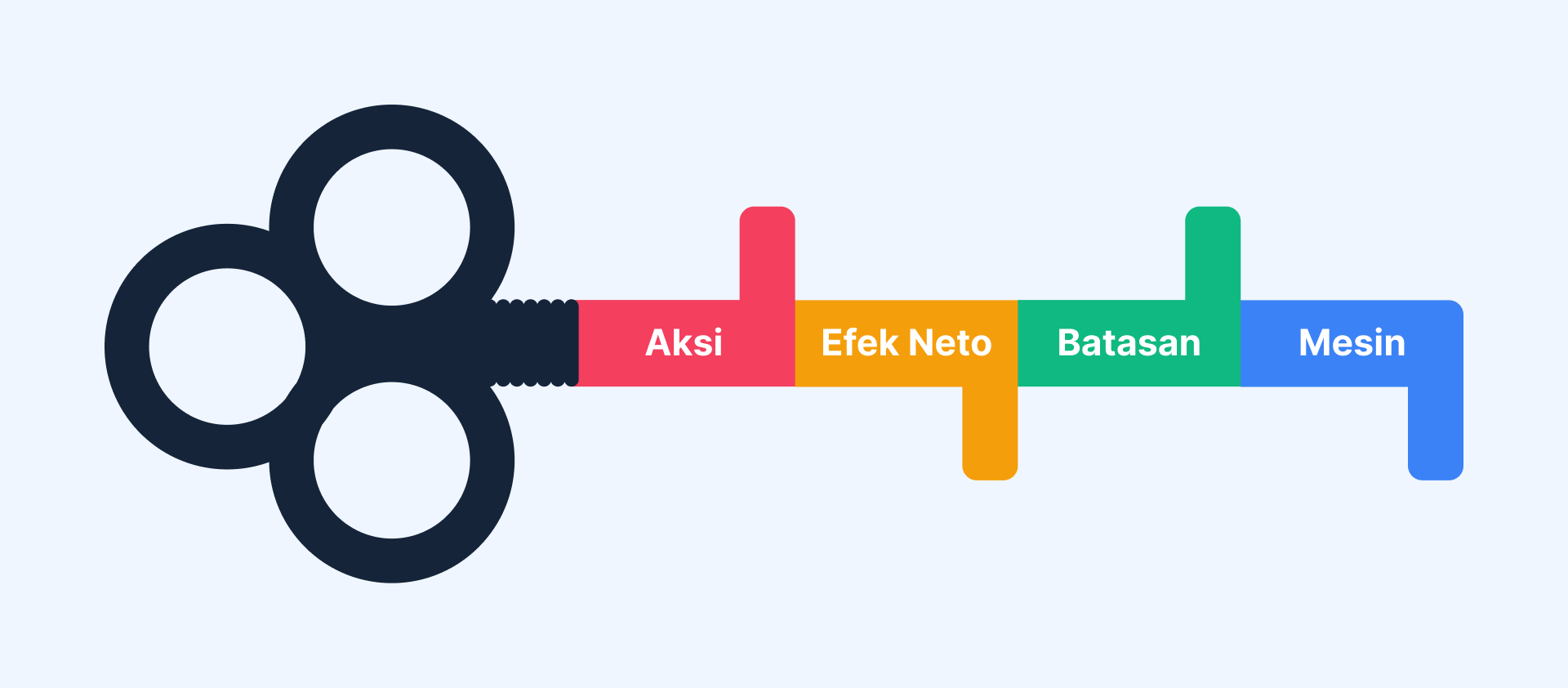
Bagi Anda yang ingin meniti karier dalam bidang pemrograman, langkah awal yang paling mudah dalam memahami pemrograman adalah belajar pemrograman prosedural. Untuk berpikir secara logis, Anda perlu mengenal konsep fundamental dalam pemrograman, seperti algoritma, pengolahan data, pengendalian aliran eksekusi, pembagian aksi menjadi sub-aksi, dll. Hal ini akan mudah dipelajari ketika Anda belajar pemrograman prosedural.

Lalu, kapan kita menerapkan pemrograman prosedural? Kita tahu bahwa pemilihan paradigma pemrograman bergantung pada kebutuhan dan kompleksitas tugas yang dihadapi. Namun, kita dapat menggunakan paradigma prosedural ketika ingin menyelesaikan masalah yang relatif sederhana. Selain itu, paradigma prosedural dapat kita pilih ketika ingin fokus pada algoritma atau langkah aksi yang berurutan.

### Insight di Balik Pemrograman Prosedural

Secara pengertian mungkin kita sudah paham tentang pemrograman prosedural. Namun, apakah Anda tahu insight atau pengetahuan menarik di balik pemrograman prosedural? Mari kita kupas lebih dalam melalui pembahasan ini.

Ada empat kunci utama yang perlu Anda pahami dalam pemrograman prosedural, yaitu **Aksi**, **Efek Neto**, **Batasan**, dan **Mesin**.

[](https://www.dicoding.com/academies/620/tutorials/32430?from=32425)

#### Aksi

Kita tahu bahwa pemrograman prosedural terdiri dari blok-blok instruksi yang dieksekusi secara berurutan. Blok tersebut serupa dengan aksi atau kejadian yang terjadi pada suatu durasi waktu tertentu. Boleh jadi ada sebuah aksi yang terjadi dalam kurun waktu yang cukup lama, tetapi tetap memiliki batas waktu yang melekat padanya. Tidak mungkin sebuah aksi terjadi dengan durasi waktu tak terbatas.

Ketika blok instruksi dijalankan secara sekuensial, artinya aksi berikutnya akan dijalankan setelah aksi sebelumnya berakhir. Apabila aksi pertama belum selesai, aksi berikutnya tidak mungkin dikerjakan. Alih-alih menjalankan aksi berikutnya, bisa saja aksi pertama tidak berjalan dengan baik sehingga tidak dapat menjalankan aksi selanjutnya. Nah, ini masuk ke poin berikutnya, yaitu **efek neto**.

#### Efek Neto

Apa itu efek neto? Efek neto berasal dari frasa bahasa inggris net effect. Dalam modul ini, kita akan selalu menyebut net effect dengan efek neto. Ini dapat diartikan sebagai reaksi atau hasil akhir yang didapatkan melalui aksi tertentu setelah mempertimbangkan semua faktor yang terlibat. Efek neto ini dapat berupa efek positif, negatif, bahkan netral, bergantung dengan dampak yang ditimbulkan dari aksi tersebut.

Dalam pengembangan suatu program, efek neto ini akan selalu hadir dalam setiap aksi yang kita lakukan, baik positif maupun negatif. Efek positif dapat Anda terima ketika program menghasilkan keluaran sesuai dengan yang diharapkan. Alih-alih mendapatkan efek positif, Anda mungkin dapat menerima efek negatif ketika program mengalami kesalahan atau error. Itulah alasan kita perlu mempertimbangkan efek neto dalam setiap aksi yang diimplementasikan pada program.

#### Batasan

Kunci berikutnya adalah batasan. Ketika aksi memiliki sebuah durasi waktu tertentu, pada saat yang bersamaan sebuah aksi memiliki batasan. Maksudnya adalah suatu aksi dapat terjadi atau terlaksana ketika ada batasan-batasan yang terdefinisi dengan baik. Mereka dideklarasikan di awal sebelum aksi itu dijalankan.

Bagaimana jika sebuah aksi tidak memiliki batasan yang jelas? Aksi yang tidak memiliki sebuah batasan yang jelas akan memiliki cakupan atau interpretasi yang luas. Anda perlu mendeskripsikan aksi lebih detail supaya penerjemah atau mesin mampu mengetahui dan memahami hasil yang diharapkan dari program.

#### Mesin

Kunci terakhir yaitu mesin. Aksi yang sudah tertulis pada sebuah program akan diterjemahkan oleh mesin sehingga menghasilkan keluaran. Setiap aksi dalam program akan diubah menjadi bentuk yang dapat dipahami mesin. Oleh karena itu, aksi yang tertulis dalam program harus sesuai dengan aturan bahasa pemrograman yang digunakan. Jika tidak, mesin tidak mampu menerjemahkan isi program dan akan memberikan penolakan berupa pesan error atau semacamnya.

Bagaimana dengan penjelasan insight di atas? Sudah cukupkah? Tenang. Jika Anda masih belum paham tak masalah. Penjelasan di atas hanyalah sneak peek dari materi selanjutnya. Tunggu apa lagi, yuk, segera ke materi berikutnya!

## Algoritma dalam Pemrograman Prosedural

Pemrograman prosedural selalu berkaitan dengan aksi yang dieksekusi secara sekuensial. Di balik itu, eksekusi aksi harus diimbangi dengan algoritma atau alur logika yang benar. Tanpa adanya algoritma yang benar, kemungkinan besar output yang dihasilkan tidak akan sesuai dengan harapan.

Lalu, apa sih maksud dari algoritma itu? Algoritma merupakan langkah-langkah atau urutan aksi logis dan terstruktur yang digunakan untuk menyelesaikan suatu tugas atau mencapai tujuan tertentu. Terkadang, suatu algoritma terdiri dari beberapa bagian yang disebut sub-algoritma apabila aksi tersebut bisa diuraikan dalam urutan aksi yang lebih sederhana.

Algoritma dapat dikatakan sebagai langkah kita untuk menghasilkan efek neto ketika sedang mengeksekusi suatu aksi. Dalam kehidupan sehari-hari, kita sering menjumpai sebuah algoritma, seperti penunjuk arah melalui peta, resep kue, panduan penggunaan peralatan elektronik, dll. Seluruh algoritma itu dirancang dengan harapan mendapatkan efek positif atau hasil yang diinginkan.

Memahami algoritma itu penting. Mengetahui algoritma atau alur yang tepat dalam menyelesaikan permasalahan memberikan dampak yang signifikan. Jika tidak percaya, simak cerita Ibu Fika dalam memotong sayur berikut.

Suatu hari, Ibu Fika sedang memotong sayur untuk mempersiapkan makan malam. Dimulai dengan mengambil sayur, menyiapkan talenan, memotong sayur, dan meletakkan potongan sayur ke dalam sebuah wadah bersih. Aktivitas Ibu Fika dilakukan secara runtut supaya tidak menimbulkan hal-hal yang tak diinginkan.

[](https://www.dicoding.com/academies/620/tutorials/32435?from=32430)

Kegiatan yang dilakukan oleh Ibu Fika di atas adalah contoh serangkaian **aksi**dalam memotong sayur. Kumpulan aksi tersebut terdiri dari berbagai aksi kecil, mulai dari mengambil sayur sampai meletakkan potongan sayur ke dalam wadah. Dengan kata lain, aksi besar yang ingin dicapai biasanya terdiri dari serangkaian sub-aksi yang lebih detail. Setiap sub-aksi Ibu Fika lakukan demi mencapai tujuan yang diinginkan.

Walaupun Ibu Fika telah melakukan serangkaian aksi dengan saksama, bisa jadi tidak berjalan sesuai rencana. Ada faktor eksternal yang mengakibatkan aksi memotong sayur gagal menghasilkan potongan sayur. Mungkin saja pisau tidak ditemukan di area dapur sehingga Ibu tidak bisa memotong sayur tersebut.

[](https://www.dicoding.com/academies/620/tutorials/32435?from=32430)

Ya, **efek neto** terdapat pada setiap aksi yang sedang berjalan. Ketika berhasil memotong sayur, Ibu Fika menghasilkan efek neto positif karena proses berjalan dengan baik dan benar. Jika pisau dapur menghilang, alih-alih menghasilkan potongan sayur, Ibu Fika hanya terdiam dan tidak bisa melanjutkan memotong. Efek yang dihasilkan oleh Ibu Fika adalah efek neto negatif. Beliau gagal menghasilkan keluaran sesuai dengan keinginannya.

Pada hari yang lain, Ibu Fika melakukan serangkaian aksi memotong sayur, tetapi ada sedikit perbedaan. Beliau memulai dengan menyiapkan talenan, bukan mengambil sayur terlebih dahulu. Ibu Fika dapat menukarkan urutan dua aksi tersebut karena talenan dan sayur berada di tempat yang sama. Kedua aksi tersebut bersifat independen atau tidak ada keterikatan satu sama lain sehingga dapat dikerjakan secara kontinu. Jika kita bayangkan, urutan aksinya menjadi seperti berikut.

[](https://www.dicoding.com/academies/620/tutorials/32435?from=32430)

Runtutan aksi tersebut dapat dilakukan selama tidak mengubah perilaku Ibu Fika dalam memotong sayur. Maksudnya bagaimana? Jika susunan kedua langkah tersebut diubah menjadi menyiapkan talenan dahulu, lalu mengambil sayur, Ibu Fika tidak mendapatkan efek neto apa pun alias netral. Asalkan tidak menghasilkan efek neto negatif, tidak masalah jika kedua langkah tersebut berubah posisi, bahkan dijadikan satu aksi sekaligus.

Berbeda cerita jika aksi memotong didahulukan. Ibu Fika tidak dapat memotong sayur karena tidak ada talenan dan sayur yang disiapkan. Alhasil, aksi Ibu Fika menghasilkan efek neto negatif yang disebabkan oleh tidak adanya prasyarat memotong sayur.

[](https://www.dicoding.com/academies/620/tutorials/32435?from=32430)

Kita tahu bahwa setiap aksi terjadi pada selang waktu yang terbatas. Aksi memiliki dua kondisi (dalam hal waktu), yaitu kondisi sebelum aksi berjalan pada saat t0 dan kondisi setelah aksi berhenti saat t1. Dengan demikian, sebuah aksi dapat dikatakan memiliki efek neto dengan membandingkan keadaan pada saat t0 (sebelum aksi) dan t1 (setelah aksi).

Kondisi tersebut dapat dikatakan sebagai **batasan** sebuah aksi. Aksi akan berjalan selama durasi (t1-t0) satuan waktu. Apakah batasan hanya sekadar dalam hal waktu? Tentu tidak. Aksi Ibu Fika memotong sayur menimbulkan banyak pertanyaan, seperti berikut.

* Apakah sayur harus dibeli dahulu atau sudah tersedia di dapur?
* Apakah sayur akan diolah menjadi masakan yang siap dimakan?
* Kapan ibu mulai memotong sayur, pagi hari, siang hari, atau malam hari?
* dst.

Dalam hal ini, kita harus membatasi dengan jelas keadaan awal dan akhir yang menjadi landasan memotong sayur supaya mampu merencanakan efek neto sesuai keinginan. Untuk itu, tentukan beberapa batasan seperti berikut.

* Initial state (keadaan awal t0) akan terjadi ketika ibu ingin memasak, sayur sudah ada dan dalam keadaan bersih, serta mangkuk dan talenan tersedia di laci dapur.
* Final state (keadaan akhir t1) terjadi ketika sayur dalam keadaan terpotong di mangkuk, siap untuk dimasak.

Pemberian batasan di atas merupakan sebuah kesepakatan antara pembuat aturan (Ibu Fika) dan orang lain yang ingin mengikuti serangkaian aksi memotong sayur. Beberapa batasan lainnya mungkin dapat didefinisikan di awal sebelum aksi berjalan. Mungkin saja terdapat batasan yang belum terdefinisi dengan baik sehingga menimbulkan efek neto negatif. Jadi, perlu adanya kesepakatan antara dua belah pihak supaya menghasilkan efek neto positif.

Itulah cerita Ibu Fika dalam menjalankan aksi memotong sayur.

Dalam membuat sebuah program, Anda perlu merangkai alur penyelesaian. Serangkaian alur tersebut dapat dikatakan sebagai algoritma. Ibarat resep atau panduan, algoritma mengarahkan kita untuk menyelesaikan instruksi satu per satu secara terperinci. Ia terdiri dari serangkaian **aksi** kecil yang menuntut kita untuk mencapai tujuan yang diinginkan.

[](https://www.dicoding.com/academies/620/tutorials/32435?from=32430)

Adakalanya, program tidak berjalan dengan mulus. Baik dari faktor internal maupun eksternal, program bisa saja menghasilkan error. Error pada faktor internal dapat terjadi di dalam program itu sendiri berupa kesalahan sintaksis dan logika, sedangkan faktor eksternal dipengaruhi oleh sumber eksternal berupa kesalahan input, gangguan jaringan, tidak kompatibel, dll.

Itulah sebab **efek neto** berlaku pada setiap aksi yang sedang berjalan. Apabila program berhasil menyelesaikan permasalahan dan menghasilkan keluaran yang sesuai dengan keinginan, efek neto positif yang dihasilkan oleh program. Sebaliknya, efek neto negatif dapat dihasilkan bila program terjadi error dan tidak menghasilkan apa yang kita harapkan.

Di sisi lain, efek neto dapat berupa netral alias tidak positif ataupun negatif. Kok bisa? Jelas bisa. Efek neto netral terjadi ketika tidak ada perubahan yang terjadi selama menjalankan instruksi program. Dengan kata lain, program tidak memiliki efek apa pun atau tidak berdampak sedikit pun setelah instruksi program berjalan. Contohnya, mengambil data dari sumber eksternal, perhitungan matematis, dan proses lainnya yang tidak mengubah atau mempengaruhi instruksi lainnya.

Pada dasarnya, efek neto dapat dianalisis dengan membandingkan keadaan sebelum dan sesudah aksi berjalan. Pada keadaan aksi selama t0 hingga t1, kita mampu mengamati apakah aksi atau instruksi menghasilkan efek neto positif, negatif, ataupun netral. Analisis efek neto ini penting untuk diperhatikan karena membantu kita memahami dampak dari aksi atau instruksi yang dilakukan dalam program.

Selain efek neto, sebuah aksi memiliki **batasan** yang mengikat dan tidak terhindarkan. Mulai dari durasi pelaksanaan aksi hingga keadaan sebelum dan sesudah aksi berjalan. Apabila batasan tidak terdefinisi dengan baik, program dapat mengalami error yang disebabkan oleh banyak hal, seperti

* lamanya durasi sebuah aksi berjalan;
* kelebihan penggunaan memori;
* ketidaksesuaian jenis, ukuran, atau format input dan output;
* kurangnya kapabilitas perangkat dalam mengelola data;
* ketidaksamaan sistem operasi, perangkat keras, atau versi aplikasi; dll.

Batasan-batasan tersebut seharusnya didefinisikan di awal sebelum program berjalan. Dengan begitu, program akan menghasilkan efek neto positif dan menyelesaikan permasalahan yang sedang diselesaikan.

Itulah konsep algoritma dalam pemrograman prosedural. Cukup singkat, bukan?

Materi di atas hanya membahas tiga kunci di balik pemrograman prosedural, yaitu aksi, efek neto, dan batasan. Satu kunci berikutnya akan dijelaskan pada materi berikutnya. Apakah Anda sudah tidak sabar? Yuk, lanjut ke materi berikutnya.

## Interpretasi Algoritma dalam Pemrograman

Sebagai pelajar SMA, Bima pernah mendapatkan mata pelajaran TIK (Teknologi Informasi dan Komunikasi). Dia pernah mendapatkan tugas oleh gurunya untuk menuliskan algoritma atau program di kertas. Bagi seorang murid, hal tersebut membuatnya merasa heran terhadap pemberian tugas semacam itu. Namun, bagi guru, memberikan tugas menulis algoritma atau program di kertas memiliki tujuan yang lebih dalam, yaitu melatih kemampuan pemikiran analitis, pemecahan masalah, serta pemahaman konsep dasar dalam pemrograman.

[](https://www.dicoding.com/academies/620/tutorials/32440?from=32435)

Namun, pernahkah Anda membayangkan cara untuk memeriksa algoritma yang berupa tulisan? Tidak mudah untuk menentukan bahwa teks tersebut adalah algoritma yang benar. Butuh keahlian khusus untuk meyakinkan bahwa algoritma tersebut benar sesuai dengan kaidah dan permasalahan yang ada. Untuk itu, kita membutuhkan sesuatu yang didesain khusus untuk memeriksa sebuah algoritma pemrograman secara cepat dan benar.

Dalam pemrograman, **mesin** yang mampu menyelesaikan masalah tersebut. Ia dirancang untuk menerjemahkan instruksi yang diberikan dan menjalankannya secara berurutan. Mesin akan menjalankan instruksi satu per satu berdasarkan aliran algoritma yang telah terdefinisi pada program.

Mesin berperan sebagai pelaksana instruksi yang ditulis melalui program. Ia bekerja berdasarkan aturan dan bahasa yang telah ditetapkan serta disepakati dalam environment pemrograman yang digunakan. Setiap bahasa pemrograman memiliki compiler atau interpreter untuk menerjemahkan program menjadi kode biner. Dari kode itulah, mesin mampu memahami alur dan maksud dari program itu sendiri.

[](https://www.dicoding.com/academies/620/tutorials/32440?from=32435)

Jika kita melihat pengertian secara umum, mesin merupakan suatu mekanisme yang menjalankan aksi primitif (dasar) atau instruksi sederhana. Selain itu, mesin dapat menjalankan serangkaian aksi secara terstruktur dan berurutan. Dengan kata lain, apa pun yang dapat menjalankan instruksi dasar dan menjalankannya secara berurutan dapat **dikatakan sebagai mesin**.

|  |
| --- |
| Pada saat Ibu Fika ingin mengajarkan anaknya memotong sayur, beliau mulai memperagakan aksi tersebut. Ibu Fika menyampaikan perintah secara lugas agar mereka mampu menirukan dan mengerjakan hal yang sama. Setelah itu, Bima berhasil memotong sayur yang benar berkat Ibu.  [Bima sedang memperagakan aksi memotong sayur bayam yang dilakukan oleh Ibu di dapur.](https://www.dicoding.com/academies/620/tutorials/32440?from=32435)  Berdasarkan cerita di atas, Ibu Fika dan anaknya berhasil melakukan serangkaian aksi memotong sayur. Mereka mengikuti algoritma yang ada untuk mengerjakan serangkaian aksi tersebut. Artinya, mereka termasuk mesin yang dapat mengerjakan beberapa aksi secara sekuensial. Jika demikian, arti dari kata mesin adalah **pengeksekusi atau pelaku dari algoritma**. |

Kita tidak dapat mengatakan semua hal yang dapat menjalankan aksi sebagai mesin. Ada suatu mekanisme yang hanya mengerjakan satu hal yang sama, contohnya charger. Ia tidak dapat disebut sebagai mesin karena menjalankan satu aksi saja, yaitu perantara pada proses pengisian daya baterai pada perangkat. Hal penting yang dapat dikerjakan oleh mesin adalah aksi-aksi sekuensial. Dengan begitu, ia mampu menjalankan beberapa aksi secara sekuensial dan merespons aksi yang telah dilakukan sebelumnya.

[](https://www.dicoding.com/academies/620/tutorials/32440?from=32435)

Berbeda cerita ketika kita dihadapkan dengan telepon genggam. Untuk mendengarkan suara, ada beberapa tahapan aksi yang harus dijalankan oleh telepon genggam, seperti mengubah suara menjadi sinyal listrik, dikirimkan melalui media komunikasi, tersampaikan pada penerima dengan mengonversi sinyal listrik menjadi suara. Setiap aksi dijalankan secara sekuensial dan dapat merespons aksi apabila terjadi kesalahan.

[](https://www.dicoding.com/academies/620/tutorials/32440?from=32435)

Jika kita memahami lebih dalam bahwa mesin dapat menjalankan beberapa aksi secara sekuensial, artinya mesin memiliki sekumpulan peralatan. Ia dibuat sedemikian rupa untuk menyelesaikan instruksi tertentu dengan jumlah peralatan yang terbatas. Setiap peralatan memiliki tugas dan fungsi spesifik serta berbeda satu sama lain.

Ketika kita memberikan program kepada mesin, ia akan “patuh” mengerjakan dan menyelesaikannya berdasarkan algoritma yang ada. Secara umum, mesin tidak memiliki inisiatif atau penafsiran dalam menjalankan instruksi. Ia hanya menjalankan instruksi berdasarkan algoritma yang tercantum dalam program. Mesin melakukannya dengan ketelitian dan konsistensi yang tinggi.

Berkaca dari hal tersebut, kita sebagai pengembang aplikasi perlu menyesuaikan pemahaman dengan mesin. Kita tidak boleh sembarangan menulis program dan dijalankan ke mesin. Kita perlu belajar menerjemahkan algoritma yang kita buat menjadi bahasa pemrograman tertentu supaya mesin paham terhadap setiap instruksi yang ada.

Di sisi lain, ketelitian dalam menulis kode program juga perlu dipelajari untuk menjadi pengembang aplikasi. Ketidaksesuaian menulis kode akan berdampak langsung pada cara mesin menjalankan program tersebut. Mesin tidak akan dapat memahami instruksi dengan benar apabila program mengalami kesalahan logika atau kode.

[](https://www.dicoding.com/academies/620/tutorials/32440?from=32435)

Oleh karena itu, penting bagi pengembang aplikasi untuk memperhatikan setiap instruksi yang dituangkan dalam bentuk kode. Mesin hanya mengikuti aturan dan kita pun harus mengikuti aturan yang berlaku. Dengan ketelitian yang tinggi, pengembang aplikasi dapat memastikan bahwa mesin berjalan dengan benar dan menghasilkan keluaran sesuai dengan harapan.

## Python sebagai Pemrograman Prosedural

Pada penjelasan sebelumnya, kita telah belajar beberapa bahasa pemrograman berdasarkan paradigma atau cara berpikir dalam menyelesaikan permasalahan. Salah satu dari paradigma pemrograman adalah pemrograman prosedural.

Lalu, mengapa pemrograman prosedural perlu dipelajari? Paradigma ini sangat cocok digunakan ketika Anda ingin memulai belajar bahasa pemrograman. Hal ini disebabkan karena cara memprogram dengan prosedural serupa cara berpikir kita dalam menyelesaikan persoalan.

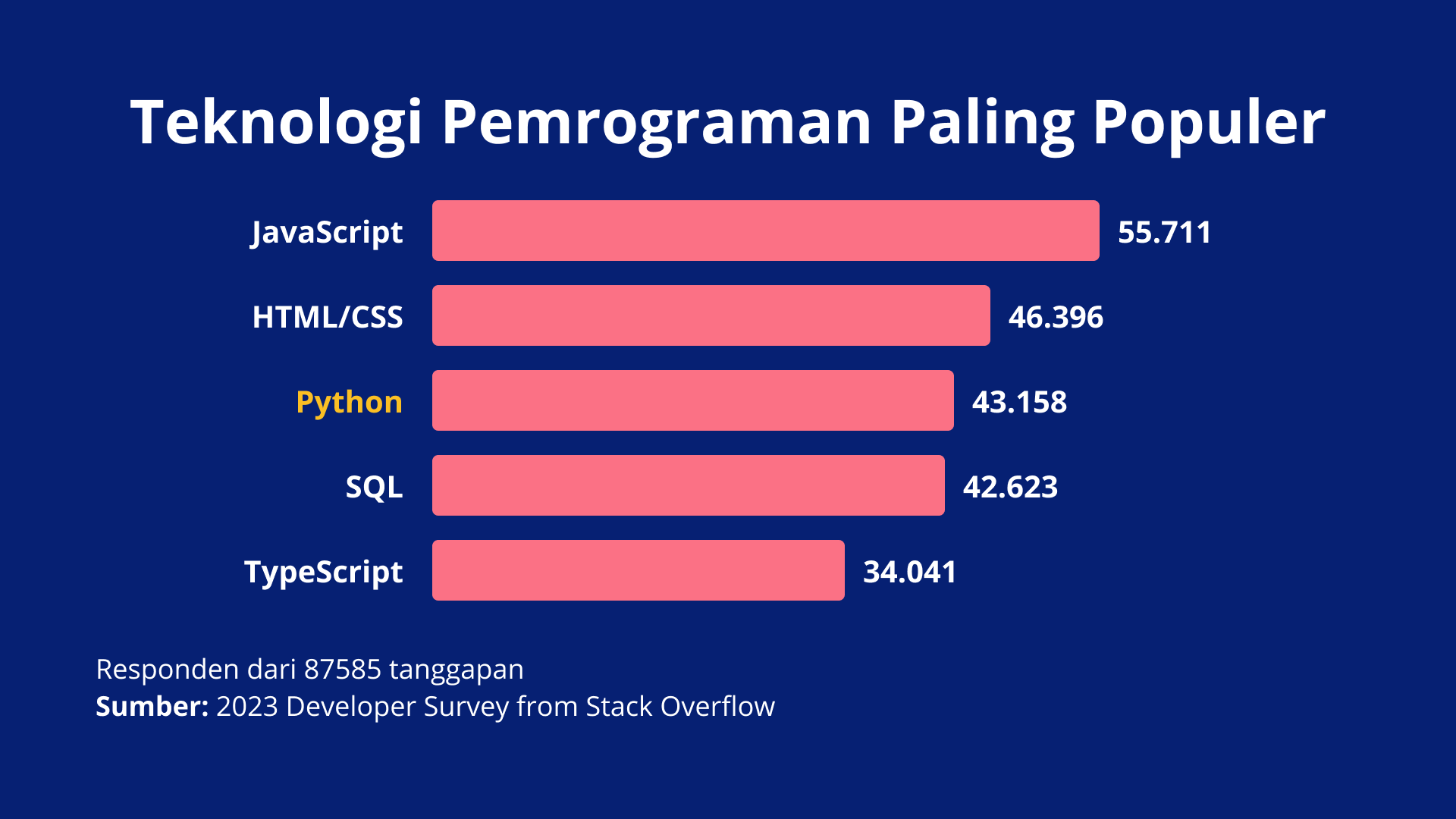
[](https://www.dicoding.com/academies/620/tutorials/32445?from=32440)

Pertanyaan berikutnya muncul di benak Anda, bahasa pemrograman apa yang paling cocok untuk pemula? Bahasa pemrograman dapat dipilih berdasarkan beberapa hal, seperti

* tingkat kompleksitas bahasa dalam menerjemahkan algoritma menjadi kode;
* kemudahan dalam memahami sintaksis kode;
* memiliki komunitas yang cukup besar dan aktif;
* memiliki resource library yang cukup banyak; serta
* beragamnya kegunaan dengan menggunakan bahasa tersebut.

Python menjadi salah satu bahasa pemrograman yang cocok untuk kita yang ingin belajar pemrograman, terlebih untuk paradigma pemrograman. Mengapa demikian? Python adalah bahasa pemrograman yang mudah dipahami. Selain itu, bahasa ini berfokus pada keterbacaan kode sehingga membuatnya sangat ramah bagi pemula.

Berdasarkan survei yang dikelola oleh [Jetbrains](https://www.jetbrains.com/lp/devecosystem-2022/python/#what-do-you-use-python-for-), Python memiliki banyak kegunaan, seperti analisis data, pengembangan web, machine learning, dan masih banyak lainnya. Di samping itu, Python menduduki lima peringkat teratas pada kategori teknologi paling populer dalam survei yang diselenggarakan oleh [Stack Overflow](https://survey.stackoverflow.co/2023/#section-most-popular-technologies-other-tools).

[](https://www.dicoding.com/academies/620/tutorials/32445?from=32440)

Itulah beberapa alasan Python cocok digunakan untuk pemula. Python adalah bahasa multi-paradigma dan kita dapat memanfaatkan konsep prosedural dalam menyelesaikan setiap permasalahan. Dengan begitu, algoritma yang sudah dibuat dapat dengan mudah ditransformasi menjadi kode yang mudah dibaca dan dipahami.

Saat ini, Anda sudah memahami konsep dasar dan insight menarik di balik pemrograman prosedural. Pada titik ini, Anda belum memulai ngoding, ya? Sabar. Sebelum memulai menulis kode, kita perlu tahu konsep dasar dan fakta unik dari pemrograman itu sendiri. Dengan begitu, kita akan paham alur pengerjaan dalam menyelesaikan permasalahan melalui kode dan semakin tertarik untuk melakukannya.

Setelah materi ini, Anda akan mulai membuat program Python berdasarkan algoritma penyelesaian masalah yang ada. Sudah tidak sabar? Yuk, beranjak ke materi berikutnya.