**Rangkuman Control Flow**

Kita sudah berada di penghujung materi Control Flow. Sampai sejauh ini, Anda telah memiliki pemahaman mengenai berbagai jenis kontrol dalam pemrograman, seperti percabangan, perulangan, hingga penanganan kesalahan. Mari kita rangkum secara saksama.

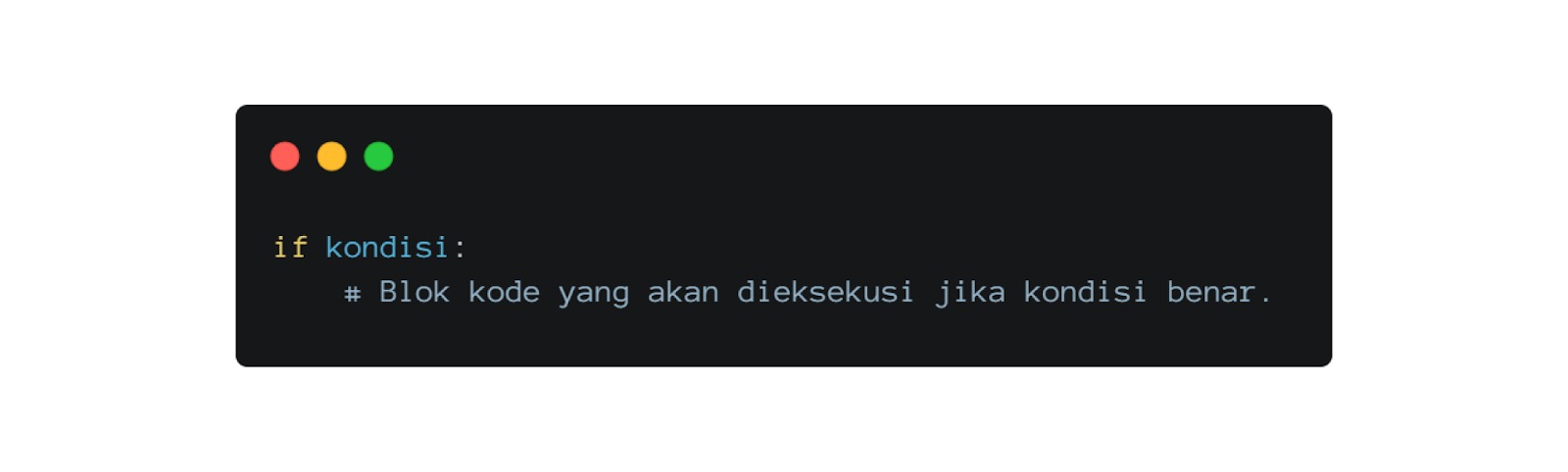
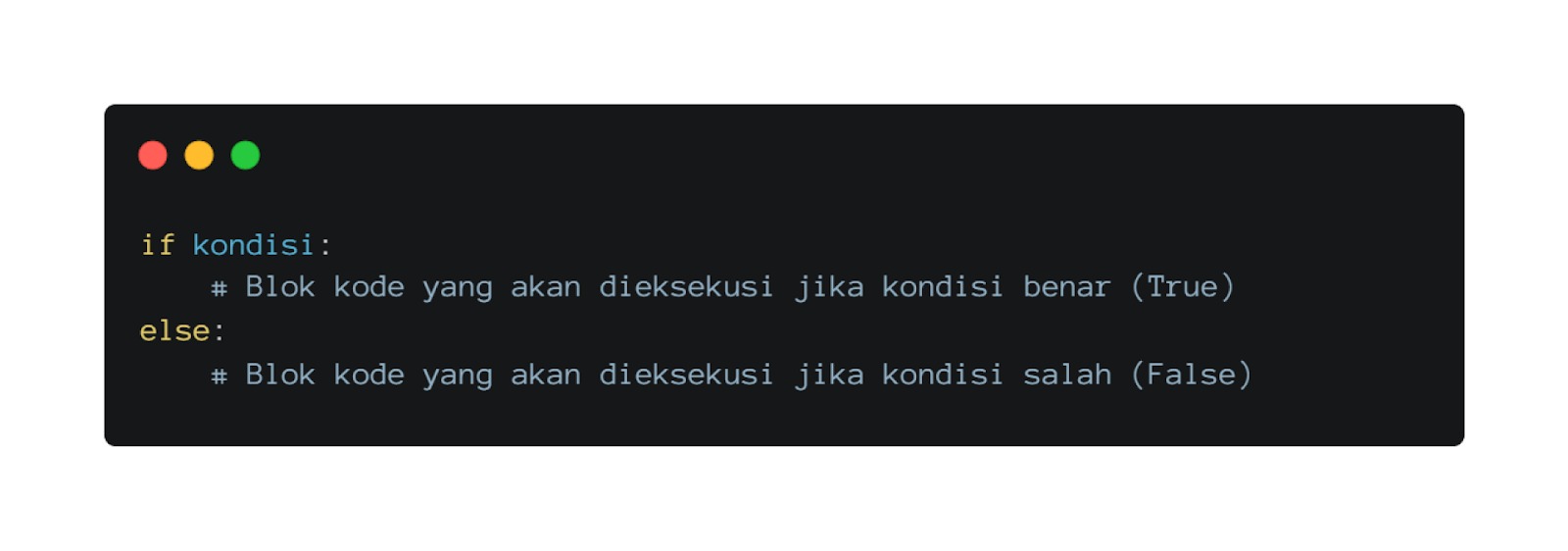
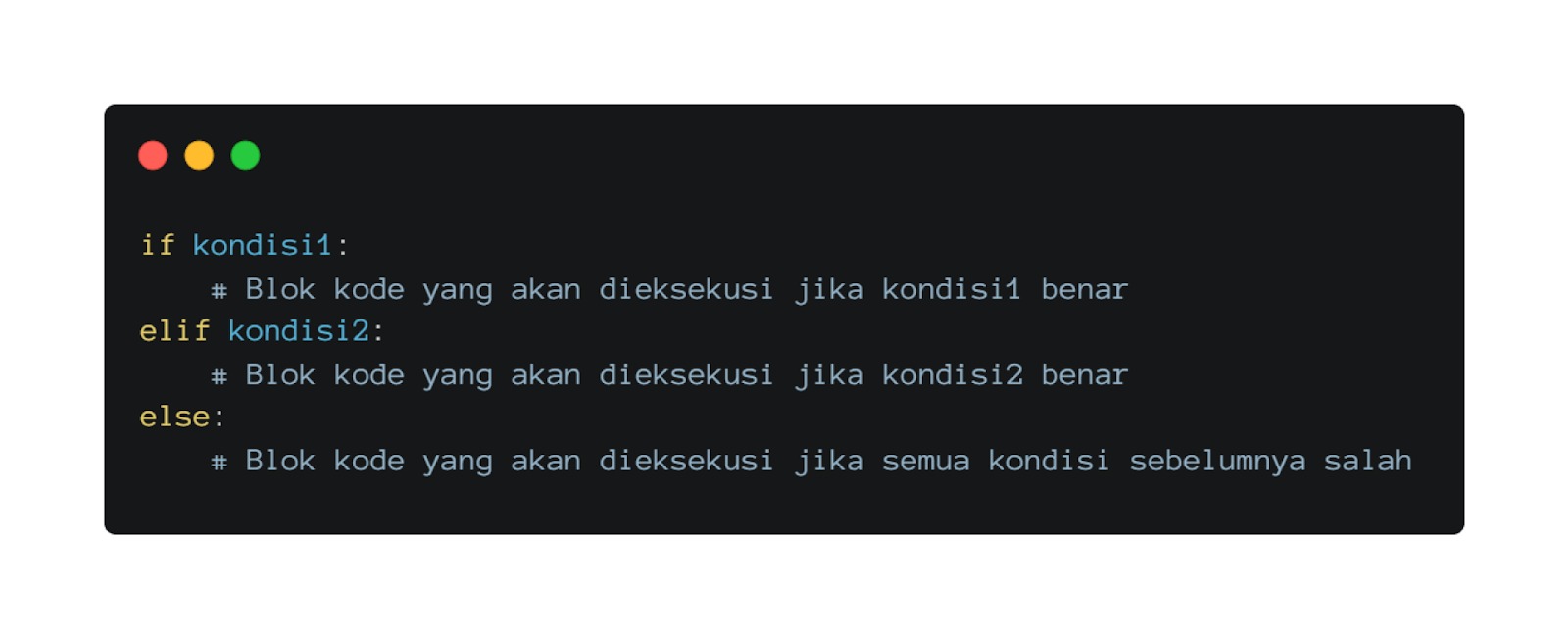
**Percabangan dan Ternary Operators**

**Percabangan**

Dalam pemrograman, sebuah kode program dapat berjalan berdasarkan kondisi tertentu. Maknanya, Anda dapat memberikan instruksi berdasarkan **"Jika-maka"**.

1. **Jika**Anda tidak menyelesaikan kelas Memulai Pemrograman dengan Python, **maka** Anda tidak lulus dari kelas Memulai Pemrograman dengan Python.
2. **Jika** variabel nama kurang dari dua, **maka** variabel tersebut tidak memenuhi kriteria kondisi.

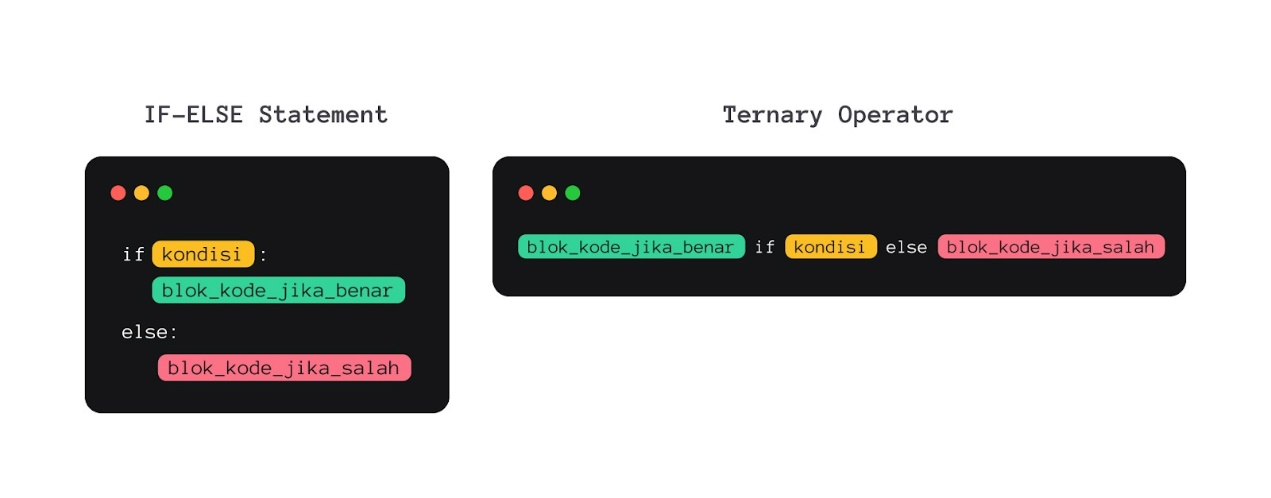
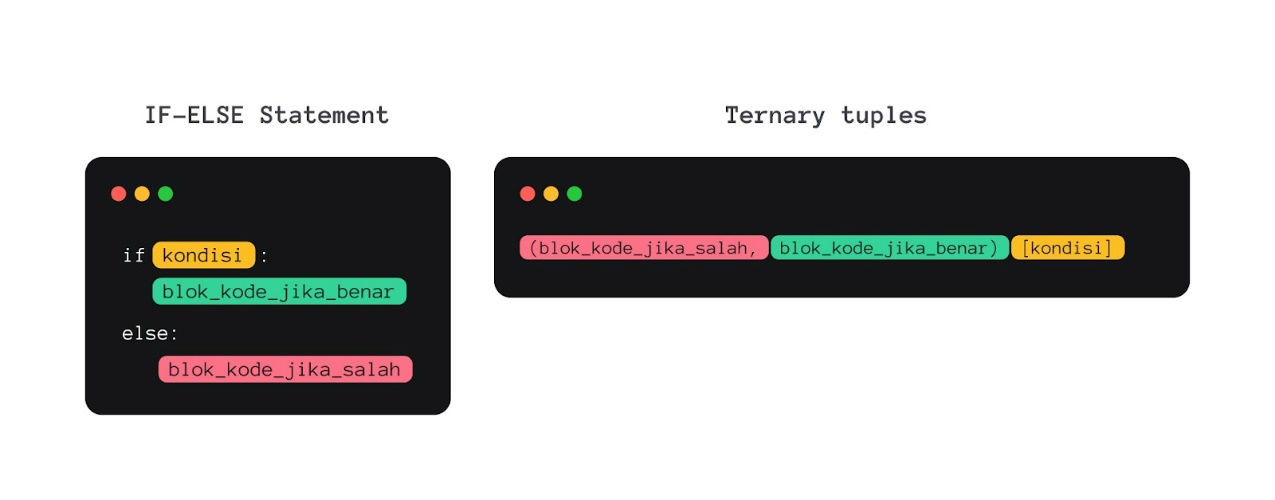
Statement atau sintaks untuk melakukan percabangan sebagai berikut.

1. **If**  
   If adalah statement Python yang akan mengecek nilai variabel di dalamnya memenuhi kriteria suatu kondisi atau tidak. Jika memenuhi kriteria, kondisi tersebut bernilai *true*. Jika tidak memenuhi kriteria, kondisi akan bernilai *false*. Jika kondisi if bernilai *true*, kode yang berada dalam blok kode if akan dieksekusi.  
     
   Struktur dari IF statement sebagai berikut.  
   [](https://www.dicoding.com/academies/86/tutorials/6416)Sebagai informasi tambahan, kita juga dapat menambahkan **'and'** atau **'or'**operator dalam kondisi percabangan.
2. **Else**  
   Else adalah statement yang menjadi jalan keluar saat kondisi atau hasil evaluasi if statement bernilai *false*. Ini maksudnya, program akan menjalani blok kode if terlebih dahulu dan jika hasilnya adalah false, program akan menjalankan else statement sebagai jalan keluar atau kondisi terakhir.  
     
   Jika kita gabungkan if dan else, struktur berikut dihasilkan.  
   [](https://www.dicoding.com/academies/86/tutorials/6416)
3. **ELIF**  
   Elif merupakan kependekan dari else if dan alternatif untuk if bertingkat atau *switch case*. Elif statement berada di posisi setelah if. Anda dapat menambahkan elif statement lebih dari satu karena tidak dibatasi dan opsional.  
     
   Struktur keseluruhan percabangan jika kita gabungkan antara if, elif, dan else sebagai berikut.  
   [](https://www.dicoding.com/academies/86/tutorials/6416)

**Ternary Operators**

Ternary operators termasuk conditional expressions pada Python. Conditional expressions adalah bentuk ekspresi yang bertujuan untuk mengevaluasi kondisi dan mengembalikan nilai berdasarkan hasil evaluasinya. Anda bisa asumsikan bahwa ternary operators ini merupakan versi one-liner dari if dan else.

Ada 2 tipe ternary sebagai berikut.

1. **Ternary Operators**  
   Ternary jenis ini adalah jenis expression yang umum digunakan. Struktur dari ternary operators sebagai berikut.  
   [](https://www.dicoding.com/academies/86/tutorials/6416)
2. **Ternary Tuples**  
   Jenis ternary kedua adalah jenis ternary yang menggunakan tipe data tuples.  
   [](https://www.dicoding.com/academies/86/tutorials/6416)

**Perulangan**

Perulangan adalah jenis kode program yang bertujuan untuk meningkatkan efektivitas program dalam membuat kode program berulang. Ada beberapa jenis perulangan dalam Python, yakni for, while, dan for bersarang.

**For**

For adalah sintaks dalam Python yang bersifat **definite iteration**. Definite iteration adalah sebuah proses iterasi atau perulangan yang jumlah pengulangannya ditentukan secara eksplisit sebelumnya.

 Format dari perulangan for sebagai berikut.

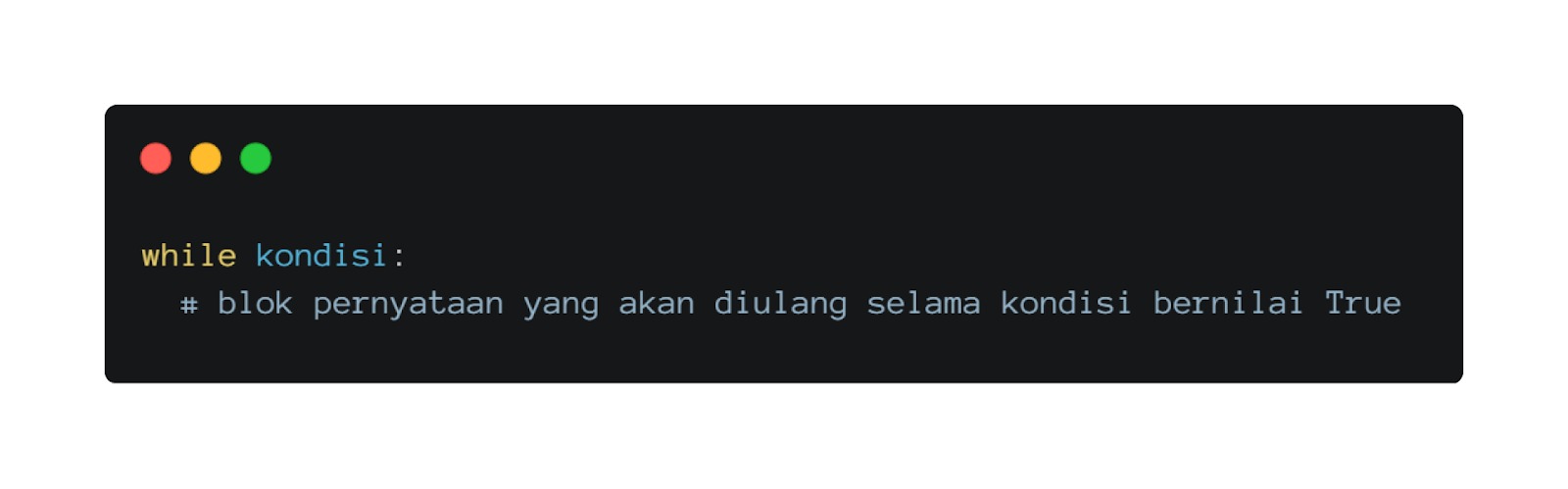
[](https://www.dicoding.com/academies/86/tutorials/6416)

[<iterable>](https://wiki.python.org/moin/Iterator)merupakan segala object dalam Python yang dapat diiterasi seperti list, tuple, hingga string. Ada pula <var> merupakan variabel yang akan mengambil elemen berikutnya dari <iterable> setiap kali iterasi berjalan.

**While**

While adalah sintaks dalam Python yang bersifat **indefinite iteration**. Indefinite iteration adalah sebuah proses iterasi yang akan berhenti ketika memenuhi kondisi tertentu.

Format dari perulangan while sebagai berikut.

[](https://www.dicoding.com/academies/86/tutorials/6416)

Kondisi merupakan ekspresi yang akan dievaluasi dan menghasilkan nilai true atau false. Selama hasil evaluasi bernilai true, program akan terus berjalan hingga menghasilkan nilai false.

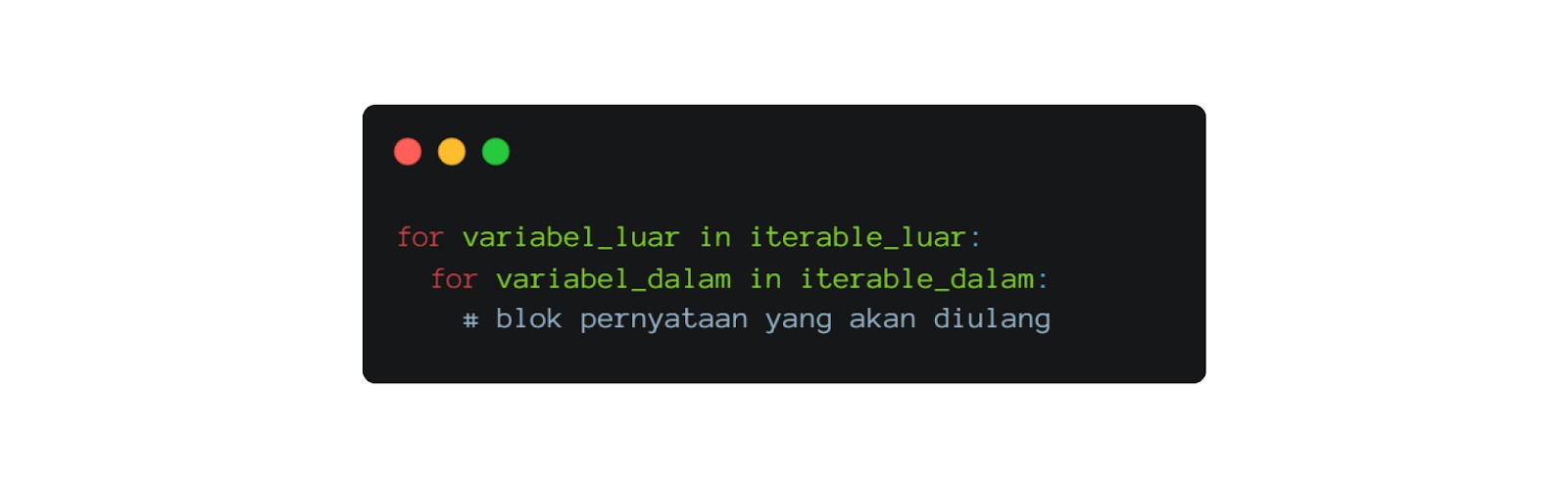
Namun, ada kondisi yang akan menyebabkan perulangan terus berjalan tanpa henti. Kondisi ini disebut dengan **infinite loop**, kebalikan dari **indefinite loop**. Kondisi **infinite loop** berarti kita melakukan perulangan yang tidak pernah memenuhi kondisi yang diinginkan. Contohnya ketika kita melakukan perulangan, tetapi tidak menambahkan increment di akhir kode.

1. counter = 1
2. while counter <= 5:
3. print(counter)

**For Bersarang**

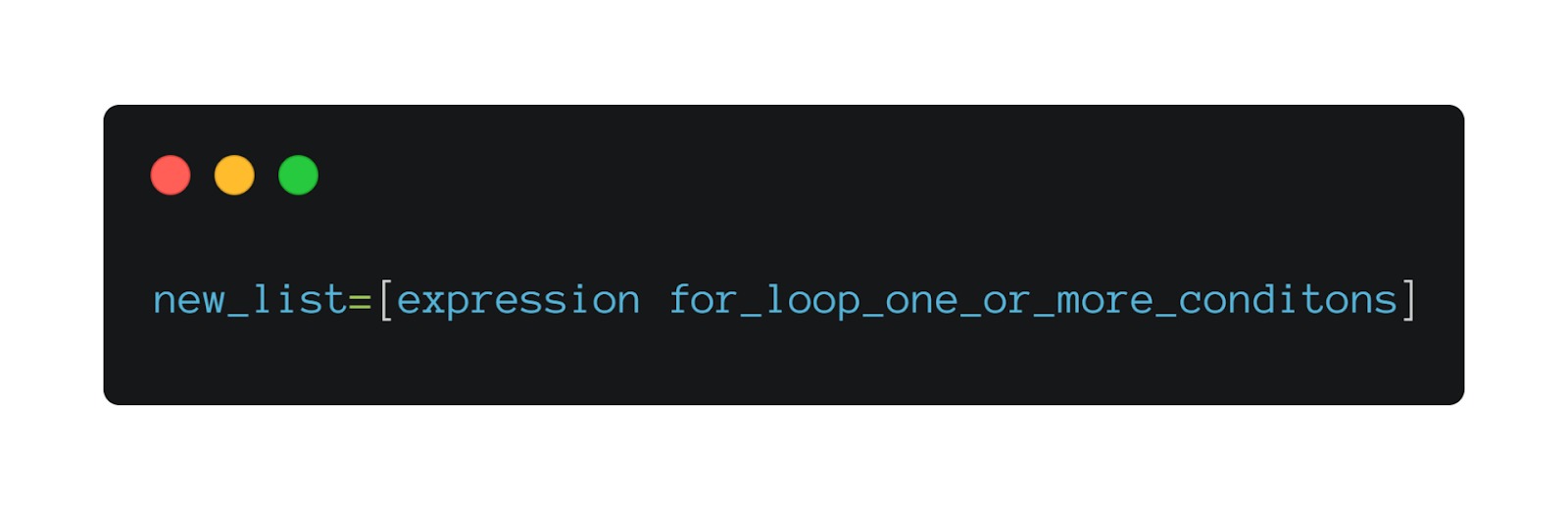
Ketika Anda membuat perulangan, sering kali menemukan perulangan dalam perulangan atau disebut sebagai nested loop.

Format dari nested loop sebagai berikut.

[](https://www.dicoding.com/academies/86/tutorials/6416)

Anda dapat asumsikan bahwa terdapat dua perulangan, yakni "perulangan luar" dan "perulangan dalam". Program akan melakukan "perulangan luar" terlebih dahulu, lalu akan melakukan "perulangan dalam". "variabel\_luar" akan mengambil nilai dari "iterable\_luar" sedangkan "variabel\_dalam" akan mengambil nilai dari "iterable\_dalam".

**Kontrol Perulangan**

1. **Break**  
   Break statement adalah pernyataan untuk menghentikan perulangan dan kemudian program akan otomatis keluar dari perulangan tersebut, lalu dilanjutkan dengan mengeksekusi blok perulangan selanjutnya.  
     
   Jika Anda memiliki perulangan yang bertingkat seperti for bersarang, break akan menghentikan perulangan sesuai dengan tingkatan atau letak perulangannya berada.
2. **Continue**  
   Continue statement adalah pernyataan untuk membuat iterasi berhenti, kemudian melanjutkan ke iterasi berikutnya. Continue seolah **mengabaikan**pernyataan (statement) yang berada antara continue hingga akhir blok.
3. **Else setelah For**  
   Pada Python juga dikenal else setelah for dengan fungsinya untuk perulangan yang bersifat pencarian. Else setelah for ini bisa dikatakan sebagai memberikan jalan keluar program saat pencarian tidak ditemukan.  
     
   Perlu diperhatikan oleh Anda, if dan else berkaitan walaupun berbeda blok. **Pada else setelah for, statement else tidak akan dieksekusi saat if pernah sekali saja benar.**Dengan kata lain, **break dalam if harus tidak terjadi untuk memicu else setelah for.**
4. **Else setelah While**  
   Berbeda dengan else setelah for, pada statement else setelah while,**blok statement else akan selalu dieksekusi saat kondisi pada while menjadi salah**.
5. **Pass**  
   Pass statement adalah pernyataan yang digunakan jika Anda menginginkan sebuah pernyataan atau blok pernyataan (statement), tetapi tidak ada tindakan atau program tidak melakukan apa pun.  
   Statement pass digunakan dalam situasi-situasi ketika Python memerlukan adanya pernyataan, tetapi tidak memiliki tindakan yang perlu dilakukan pada saat itu. Biasanya itu adalah kondisi ketika Anda membutuhkan *placeholder* untuk menunjukkan bahwa tidak ada operasi yang perlu dilakukan. Hal ini dapat membantu kita mengatur struktur kode secara rapi dan memungkinkan penambahan implementasi di kemudian hari.
6. **List Comprehension**  
   List comprehension adalah cara untuk menghasilkan list baru berdasarkan list atau *iterables* yang telah ada sebelumnya. Sintaks dasarnya sebagai berikut.  
   [](https://www.dicoding.com/academies/86/tutorials/6416)  
   Catatan:
   1. **new\_list**merupakan variabel yang dideklarasikan oleh Anda.
   2. **expression**merupakan ekspresi yang akan dijalankan seiring perulangan bernilai benar.
   3. **for\_loop\_one\_or\_more\_conditions**merupakan perulangan for yang Anda definisikan. Contohnya adalah "for n in angka" yang ada pada contoh sebelumnya.

**Penanganan Kesalahan (Error Handling and Exception Handling)**

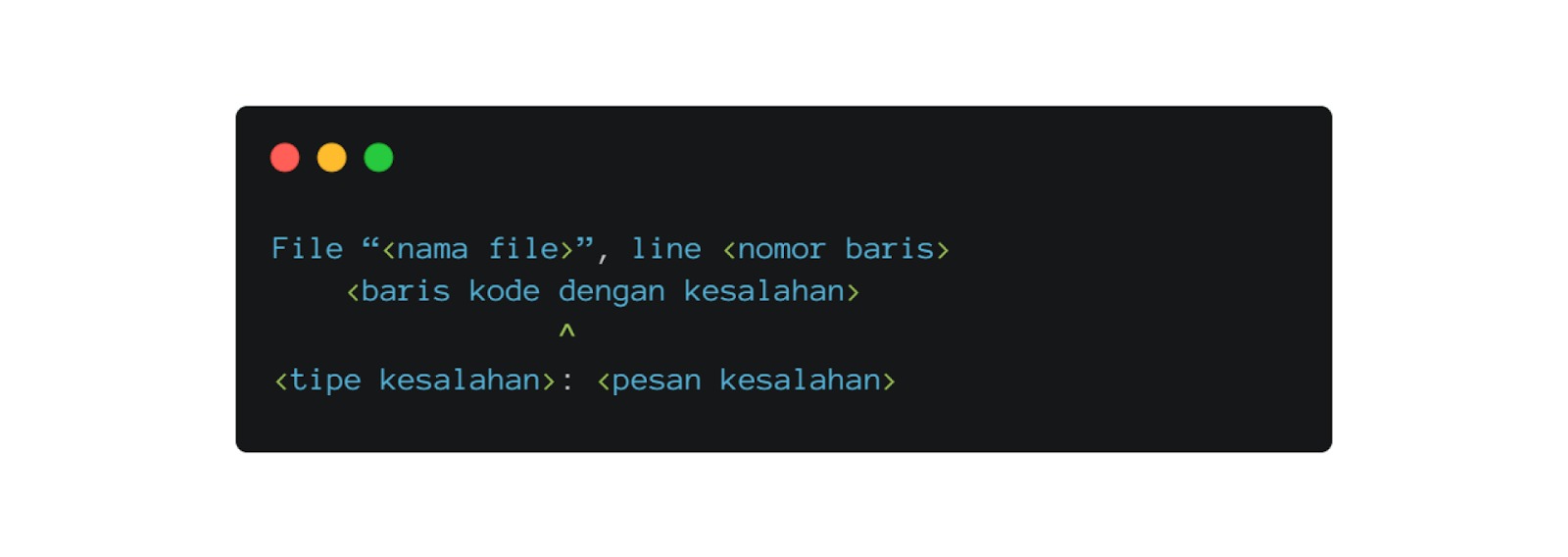
Saat Anda membuat program, sering kali menemukan setidaknya dua jenis kesalahan berdasarkan kejadiannya.

1. Kesalahan sintaks (*syntax errors*) atau sering disebut juga sebagai kesalahan penguraian (*parsing errors*).
2. Pengecualian (*exceptions*) atau sering disebut juga sebagai kesalahan saat beroperasi (*runtime* errors).

**Kesalahan Sintaks (Syntax Errors)**

Kesalahan sintaks (*syntax errors*) adalah jenis kesalahan yang terjadi ketika Python tidak mengerti perintah Anda. Ini mengakibatkan pesan kesalahan muncul sebelum program tersebut berjalan.

Secara umum struktur kesalahan sintaks sebagai berikut.

[](https://www.dicoding.com/academies/86/tutorials/6416)

Berikut adalah penjelasan detail satu per satu terkait poin di atas.

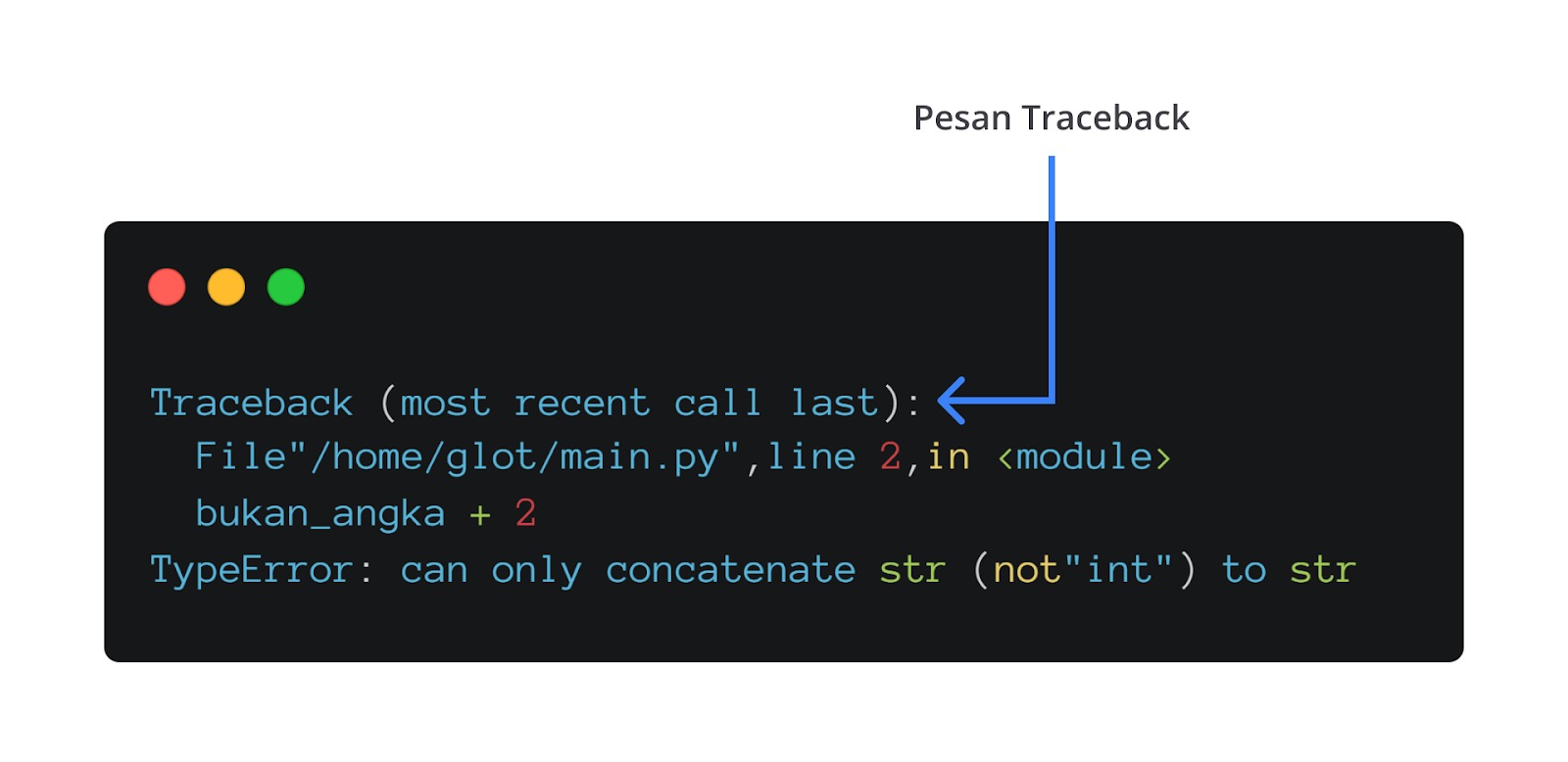
1. "<nama file>" merupakan file Python yang Anda eksekusi. Jika Anda menggunakan mode script melalui lokal komputer dan program Anda menghasilkan Error, pesan ini akan memunculkan nama script atau file Python Anda.
2. <nomor baris> merupakan nomor baris kode dalam file Anda yang mengalami kesalahan.
3. <baris kode> merupakan kode yang mengalami kesalahan dalam file Anda.
4. <tipe kesalahan> merupakan kelompok atau tipe kesalahan yang Anda alami, contohnya SyntaxErrordan IndentationError.
5. <pesan kesalahan> merupakan pesan detail kesalahan atau keterangan yang diberikan oleh program. Contohnya “invalid syntax” dan “expected an indented block”.

**Pengecualian (Exceptions)**

Pengecualian adalah kesalahan yang terjadi ketika Python mengerti perintah Anda, tetapi mendapatkan masalah saat mengikutinya. Umumnya, pengecualian bisa terjadi ketika aplikasi sudah mulai beroperasi.

Jenis kesalahan ini adalah kesalahan yang paling sering ditemui ketika Anda membuat kode program yang kompleks. Meskipun kode atau ekspresi dari Python yang Anda tulis sudah benar, ada kemungkinan terjadi kesalahan ketika perintah tersebut dieksekusi.

Secara umum, struktur pengecualian sama seperti kesalahan sintaksis. Namun, hal yang menjadi pembeda adalah pada struktur pengecualian memberikan pesan "traceback (most recent call last)".

[](https://www.dicoding.com/academies/86/tutorials/6416)

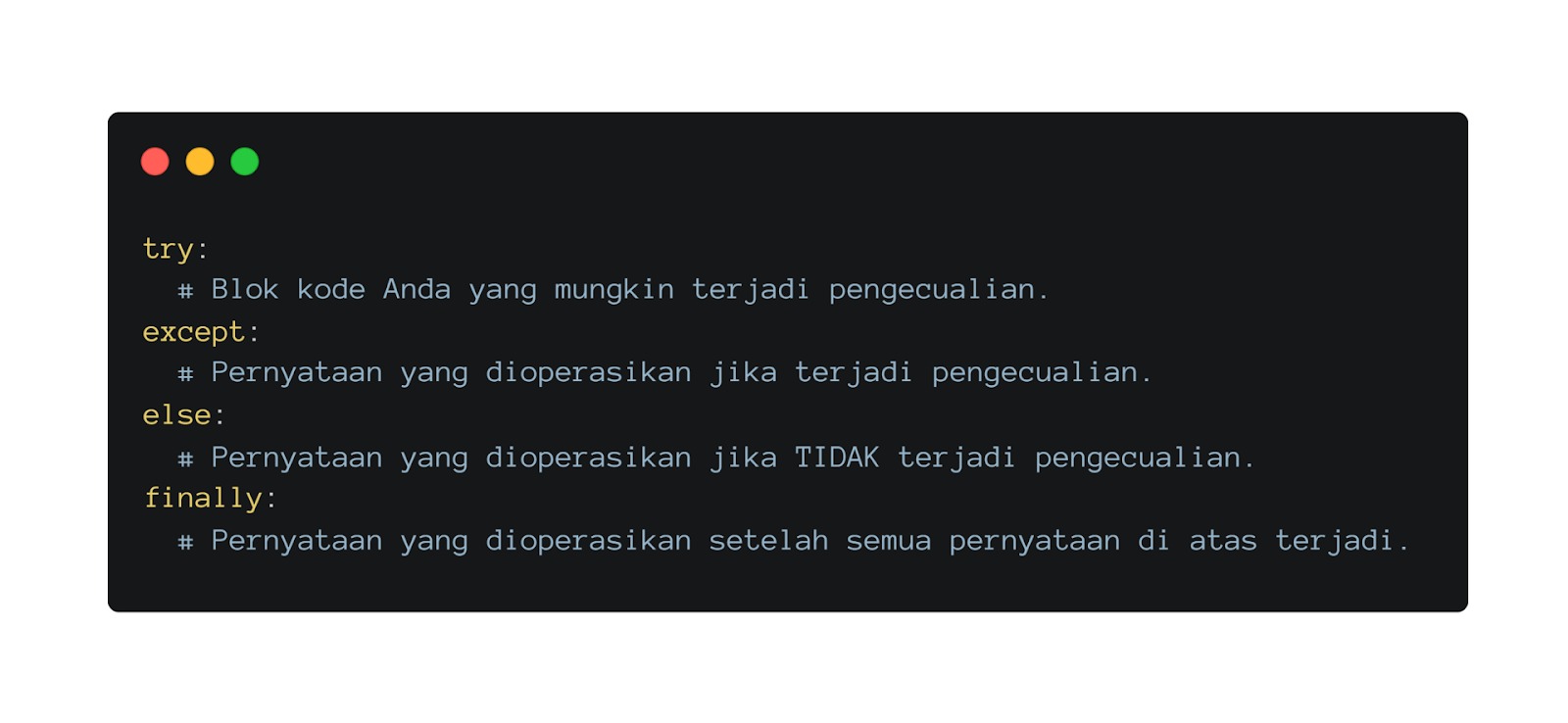
Pesan ini mengacu pada informasi yang ditampilkan ketika terjadi kesalahan atau pengecualian (*exception*). Pesan *traceback* ini menyediakan "jejak" dari kode yang dieksekusi sehingga Anda dapat melacak kembali jalur eksekusi program sebelum mencapai titik error.

**Penanganan Pengecualian**

Untuk menyelesaikan setiap masalah pengecualian, kita dapat membangun kode program dalam menangani hal tersebut.

Program Python yang Anda bangun dapat dilengkapi penanganan terhadap pengecualian dari tipe kesalahan yang Anda tentukan. Konsep ini dikenal dengan *exceptions handling* yang menggunakan pernyataan **try-except**untuk menangani pengecualian tersebut.

Secara keseluruhan, struktur lengkap dari penanganan pengecualian sebagai berikut.

[](https://www.dicoding.com/academies/86/tutorials/6416)

Pada try statement, program akan menjalankan blok kode yang mungkin terjadi pengecualian. Pada except statement, program akan mengeksekusi statement ini jika terjadi pengecualian. Pada else statement, program akan mengeksekusi statement ini jika **tidak** terjadi pengecualian. Pada finally statement, program akan mengeksekusi statement ini setelah semua pernyataan di atas terjadi.

**Raise Exception**

Jika sebelumnya kita menangani kesalahan yang **TIDAK DISENGAJA**, kali ini kita akan mempelajari cara menangani kesalahan yang **DISENGAJA**. Umumnya, ketika membuat kode program kita ingin membatasi program tersebut dengan kondisi tertentu.

Perlu diingat bahwa umumnya, raise digunakan bersamaan dengan if-else statement.

Berikut adalah contoh penerapan raise statement untuk menangani kesalahan atau pengecualian yang disengaja.

1. var = -1
2. if var < 0:
3. raise ValueError("Bilangan negatif tidak diperbolehkan")
4. else:
5. for i in range(var):
6. print(i+1)

9. """
10. Traceback (most recent call last):
11. File "/home/glot/main.py", line 3, in <module>
12. raise ValueError("Bilangan negatif tidak diperbolehkan")
13. ValueError: Bilangan negatif tidak diperbolehkan