**Abstraksi Data**

Abstraksi data merupakan kemampuan Anda untuk mengerti konteks dan merepresentasikannya menjadi bentuk lain sesuai dengan konteks masalahnya. Simak contoh skenario berikut. Anda memiliki sebuah angka bernilai 60 yang tidak mewakili apa pun, hanya angka 60 saja. Sekilas, Anda paham bahwa angka tersebut bernilai 60. Namun, apakah Anda mengetahui apa maksud dari angka tersebut? Apakah maksudnya suhu udara? Lembar kertas? Atau berat badan?



Sekarang, apa yang terjadi ketika Anda mengubah format angka sebelumnya yang hanya 60 menjadi seperti gambar berikut?



Apakah Anda melihat perbedaannya? Setelah ditambahkan satuan derajat celcius “ °C “, Anda tidak lagi berasumsi banyak hal seperti sebelumnya. Sebab, Anda sudah tahu pasti angka 60°C tersebut merepresentasikan derajat suhu.

Dari contoh tersebut, Anda pasti sudah paham bahwa kemampuan mengabstraksi data penting untuk memahami konteks. Sama seperti Anda yang perlu mempunyai kemampuan tersebut, komputer yang Anda gunakan pun memiliki kemampuan abstraksi data.

Ketika menuliskan data dalam pemrograman, komputer tidak akan mengetahui data yang dimaksud hingga Anda mendeklarasikan tipe datanya, sama seperti contoh di atas.

**Data Typing**

Sekarang Anda memahami bahwa komputer perlu mengetahui data yang diinginkan dengan cara melakukan deklarasi. Sebelum memahami berbagai tipe data yang umum digunakan dalam Python, Anda harus mengenal terlebih dahulu cara menuliskan data dalam pemrograman.

**Deklarasi dan Inisialisasi**

Deklarasi merujuk pada pembuatan variabel dengan menentukan tipe data dan nama variabelnya. Umumnya, ini dilakukan oleh bahasa pemrograman lainnya, seperti C/C++. Sementara inisialisasi merujuk kepada pemberian nilai awal pada variabel yang sebelumnya telah dideklarasikan. Kedua proses tersebut wajib dilakukan dalam bahasa pemrograman lain, seperti C. Beruntungnya Anda, Python tidak mengharuskan Anda untuk melakukan deklarasi tipe data variabel. Hal ini disebabkan Python merupakan bahasa pemrograman yang menerapkan *loosely typed*. Artinya, Anda tidak perlu mendeklarasikan tipe data variabel secara eksplisit. Python juga merupakan bahasa pemrograman yang menerapkan ***dynamic typing*.**

Artinya, Python adalah bahasa pemrograman yang hanya mengetahui tipe variabel saat program berjalan dan melakukan proses *assignment*. Hal ini memungkinkan kita untuk mengubah tipe data dari suatu variabel seiring berjalannya program.

**Tipe Data**

Sebagaimana yang telah dijelaskan bahwa setiap nilai yang digunakan dalam variabel adalah sebuah data. Data sendiri memiliki tipe yang berbeda-beda, kita dapat menemuinya dalam kehidupan sehari-hari. Simak kisah berikut untuk memahami data dalam kehidupan sehari-hari. Sekarang mari kita fokus terhadap berbagai tipe data pada Python. Dalam Python, tipe data dikelompokkan menjadi dua, yakni **tipe data primitif**dan **tipe data collection**.

**Tipe Data Primitif**

Tipe data primitif merupakan jenis paling dasar dalam pemrograman. Tipe data ini menyimpan *single value*. Berikut adalah berbagai tipe data primitif.

**Numbers**

Integer Bilangan bulat positif atau negatif dan tidak memiliki angka desimal.

Contoh: 1; -20; 999; dan 0.

Float Bilangan riil yang dapat mewakili bilangan bulat atau bilangan desimal.

Contoh: 3.14; 1; dan 4.01E+1

Complex Bilangan kompleks. (Kita tidak akan menggunakannya di kelas ini.)

Contoh: 1+2j

Tipe data primitif pertama, yakni *numbers* adalah tipe data angka berupa bilangan bulat, riil, dan kompleks.

Tipe data integer merupakan bilangan bulat positif atau negatif dan tidak memiliki angka desimal. Selanjutnya, tipe data float merupakan bilangan riil yang mewakili bilangan bulat dan angka desimal. Terakhir, tipe data complex yang merupakan representasi dari bilangan kompleks dalam matematika. Tipe data complexterdiri dari bilangan riil dan imajiner dengan bentuk “x +yj”, yaitu “x” adalah bilangan riil dan “y” adalah bilangan imajiner.

Ciri dari bilangan numbers adalah Anda dapat mengoperasikan bilangan tersebut dengan operasi matematika sederhana, seperti pertambahan (+), pengurangan (-), perkalian (\*), dan operasi matematika lainnya.

Perlu diperhatikan bahwa tipe data numbersbersifat *immutable* yang artinya nilai di dalamnya tidak dapat diubah. Sebenarnya, semua tipe data primitif atau *single-value* (numbers, boolean, string) sudah dapat dipastikan adalah *immutable* secara natural. Banyak programmer beranggapan bahwa dengan melakukan inisialisasi ulang variabel pada Python dapat memperbarui atau mengubah nilai tersebut. Nyatanya, tindakan tersebut menyebabkan program membuat objek baru dengan nilai baru alih-alih mengubahnya.

**Boolean**

Tipe data primitif kedua adalah boolean, yakni tipe data yang hanya bernilai TRUE atau FALSE. Tipe data ini merepresentasikan nilai kebenaran (*truth values*). Sebenarnya, setiap variabel yang memiliki nilai bisa dievaluasi dan menghasilkan nilai *true*. Hanya ada beberapa nilai yang akan dianggap bernilai *false*sebagai berikut.

1. Nilai yang sudah didefinisikan bernilai salah: **None dan False**.
2. Angka nol dari semua tipe numerik: **0, 0.0, 0j, Decimal(0), Fraction(0,1)**.
3. Urutan (*sequence*) dan koleksi (*collection*) yang kosong: **“”, (), {}, set(), range(0)**.

**String**

String merupakan karakter yang berurutan. Ketika Anda membuat variabel bernilai string tentu diawali dengan *single quote*(‘’) atau *double quote* (“”). Jalankan kode di bawah ini untuk mengetahui contoh tipe data string.

Variabel x yang menyimpan teks “Dicoding” adalah variabel dengan data bertipe string. Sederhananya, teks “Dicoding” tersebut adalah string.

Beberapa fakta menarik lainnya dari string Python adalah berikut.

1. Anda dapat menggunakan tiga *single quote* atau *double quote* untuk menyimpan string yang lebih dari satu baris (*multi-line*).

Pada kode di atas, Anda menampilkan string lebih dari satu baris (multi-line) menggunakan double quote (”””).

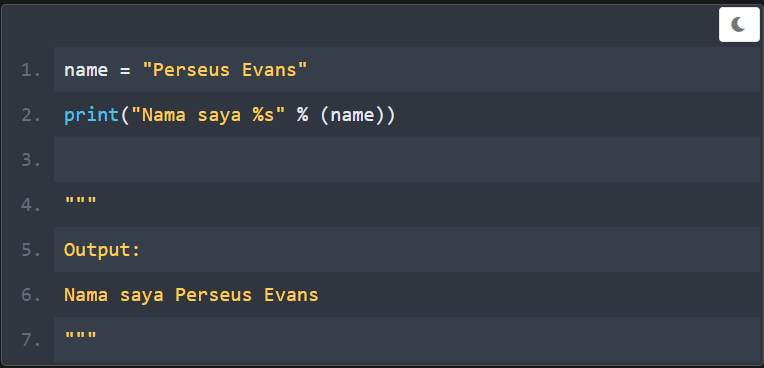
1. String merupakan urutan karakter yang setiap karakternya memiliki indeks. Anda dapat mengakses setiap karakter dari string (substring) dengan menggunakan metode indexing. Perlu diingat bahwa indeks selalu dimulai dari 0.

Metode indexing merupakan cara untuk mengambil spesifik elemen berdasarkan indeksnya.

1. Namun, Anda tidak dapat mengubah substring di dalamnya. Ini dikarenakan string pada Python bersifat immutable.
2. Anda dapat mengakses beberapa substring dengan menggunakan metode *indexing* dan *slicing*.  Metode *slicing*adalah cara yang sering digunakan untuk mendapatkan bagian dari suatu list atau array. Metode ini dapat diterapkan pada string untuk mengambil satu atau banyak substring.
3. Anda dapat menampilkan teks/string berdasarkan input dari pengguna dengan berbagai cara. Perhatikan metode di bawah ini dan jalankan kodenya menggunakan IDE atau notebook Anda.
   1. **Formatted String**

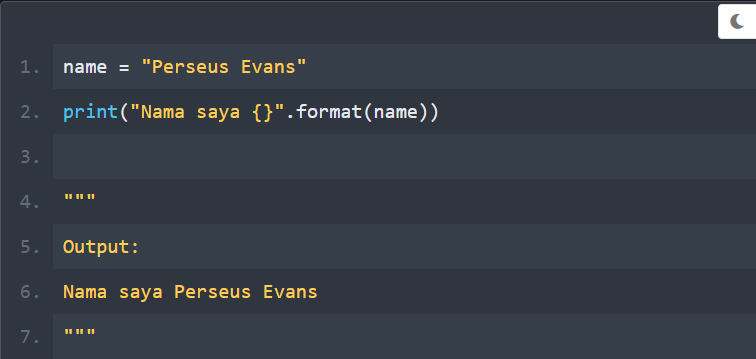
Metode ini diperuntukkan untuk menampilkan variabel bertipe string dengan menggunakan huruf “f” di depan string dan menempatkan variabel di dalam kurung kurawal.

* 1. **%-formatting**



Pada kode di atas, Anda menampilkan variabel string dengan menggunakan metode “%-formatting”. Metode ini adalah pendekatan lama yang masih didukung oleh Python. Metode ini menggunakan operator Modulo (%) untuk memasukkan nilai variabel ke dalam string dengan menggunakan format khusus yang ditentukan oleh tipe data variabel.

* 1. **str.format()**



Pada kode di atas, Anda menampilkan variabel string dengan menggunakan metode “str.format()”. Metode ini memungkinkan penggabungan variabel/nilai ke dalam string dengan menempatkan tanda kurung kurawal atau {} sebagai penempatan variabel. Sekilas mirip dengan formatted string, pembedanya adalah pada penggunaan “.format” setelah string.

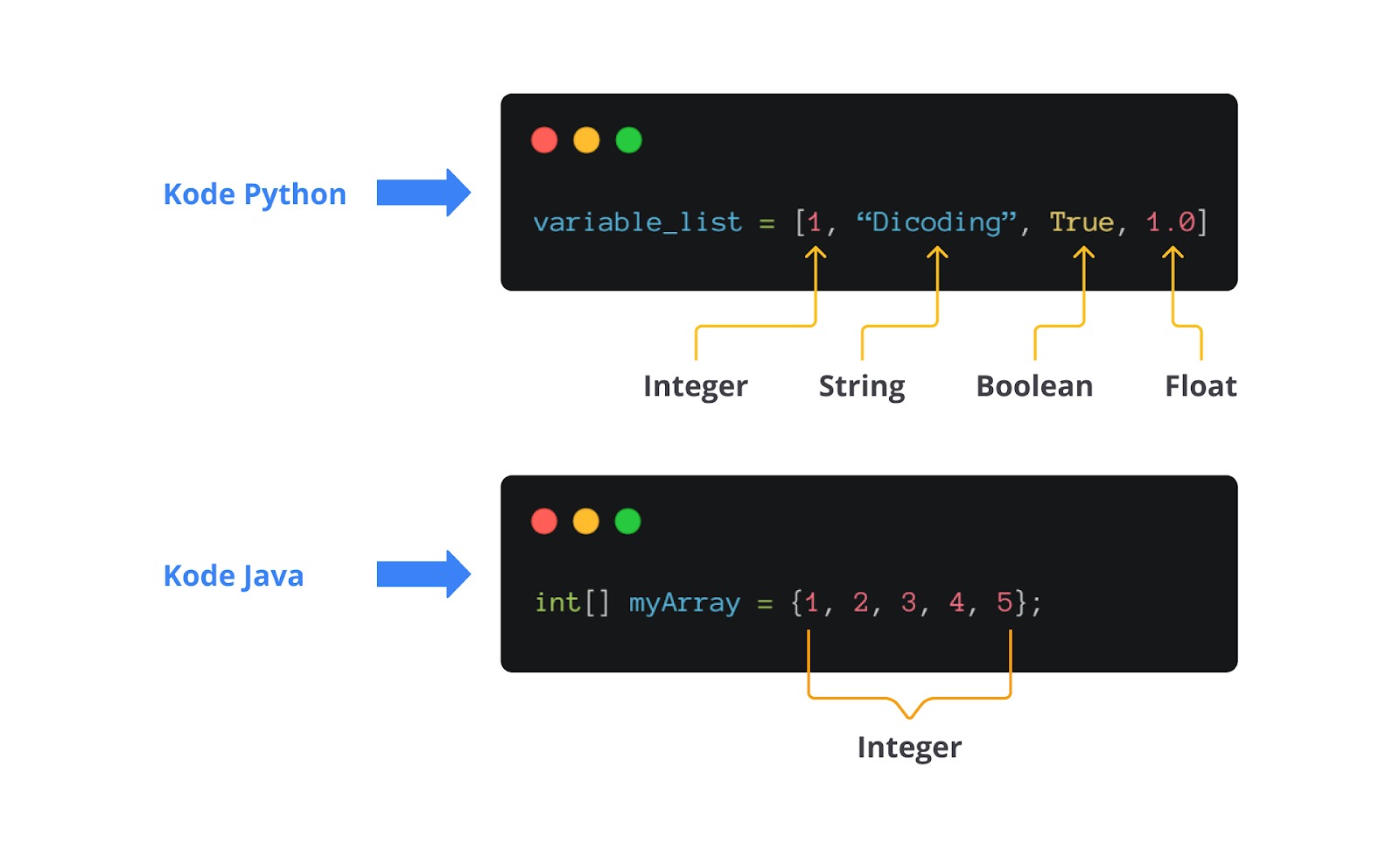
Masih banyak cara untuk menampilkan string. Anda bisa mendapatkan informasi lebih detail terkait string Python pada laman berikut. [https://docs.python.org/3/library/string.html](https://docs.python.org/3/library/string.html" \t "_blank).

**Tipe Data Collection**

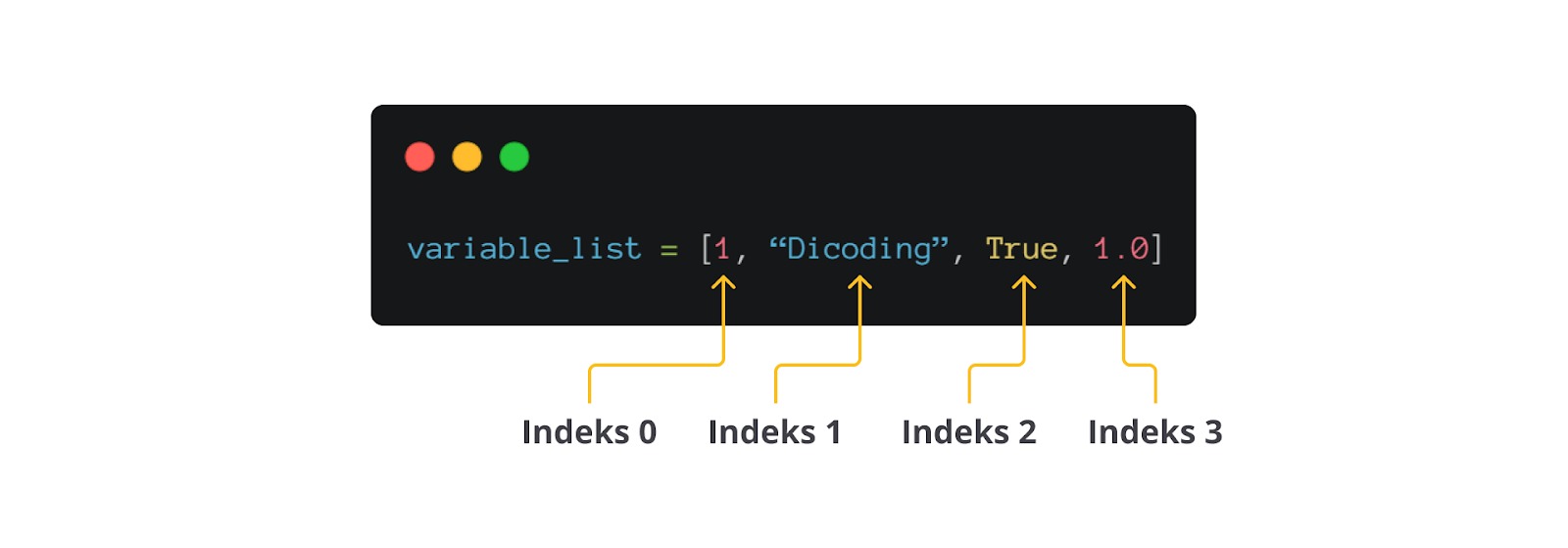
Tipe data collection merupakan tipe data yang menyimpan satu atau lebih data primitif sebagai satu kelompok. Dalam Python, berikut yang termasuk tipe data collection.

**List**

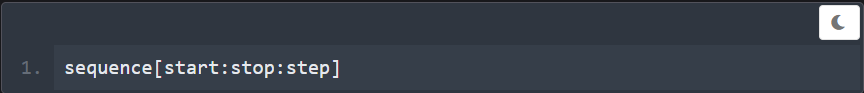
List merupakan jenis kumpulan data terurut (*ordered sequence*) dan salah satu tipe data yang sering digunakan pada Python. List dalam Python ini serupa, tetapi tak sama dengan array pada bahasa pemrograman lainnya. List Python tidak mengharuskan memiliki tipe data yang sama di dalamnya, sedangkan array sebaliknya.



Melakukan inisialisasi list pada Python cukup mudah, yakni menggunakan kurung siku “[]” dan setiap elemennya dipisahkan dengan koma. Berikut adalah implementasi list pada Python. Setiap data atau elemen dalam list memiliki indeks yang selalu dimulai dari 0. Anda dapat mengakses setiap indeks pada list dengan metode indexing. List Python bersifat *mutable*yang artinya nilai di dalamnya dapat diubah.

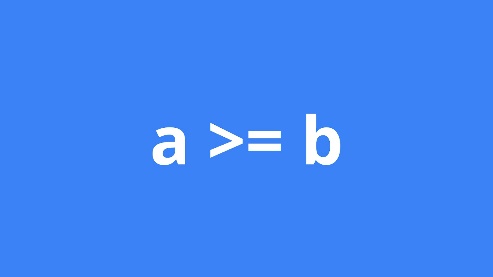


Konsep **indexing**merujuk kepada pengambilan data dalam Python berdasarkan indeksnya. Adapun konsep **slicing** merujuk pada pengambilan data berdasarkan indeks dari rentang tertentu.

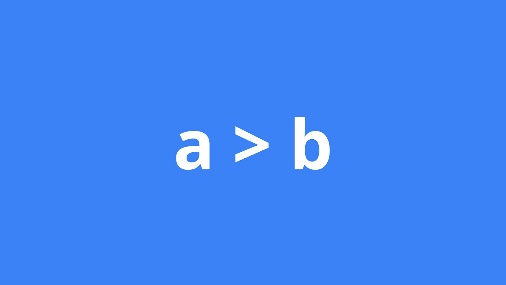


***Start***merupakan indeks pertama yang Anda ambil. ***Stop***merupakan indeks terakhir yang ingin Anda ambil. ***Step***merupakan jumlah elemen yang ingin Anda lewati di antara setiap elemen slice. Secara default, nilai step adalah 1.

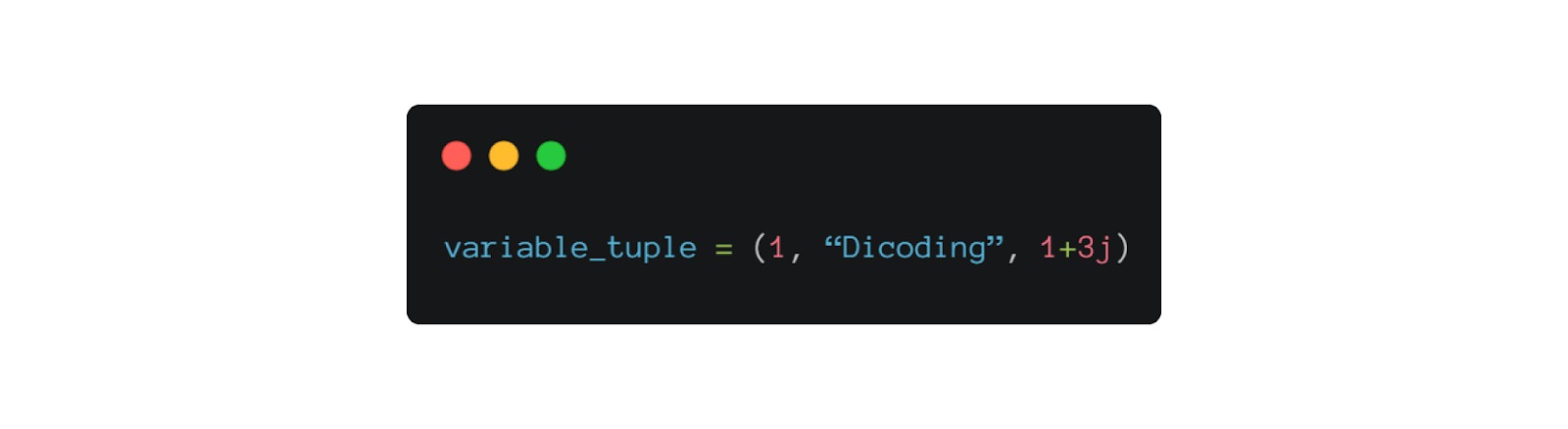
Hal penting yang harus Anda ingat adalah nilai ***start***bersifat inklusif sedangkan ***stop*** bersifat eksklusif. Masih ingat dengan konsep tersebut dalam matematika? Konsep ini menggambarkan batas tertentu dalam suatu interval. Jika suatu interval dikatakan inklusif, batas terakhir yang telah ditentukan akan dianggap sebagai bagian dari interval.



Namun, jika suatu interval dikatakan eksklusif, batas terakhir yang telah ditentukan tidak akan dianggap sebagai bagian dari interval.



**Tuple**



Tuple adalah jenis dari list yang tidak dapat diubah elemennya. Umumnya, tuple digunakan untuk data yang bersifat sekali deklarasi dan dapat dieksekusi lebih cepat. Tuple didefinisikan dengan kurung “()“ dan setiap elemen di dalamnya dipisahkan dengan koma.

Pada kode di atas, Anda dapat lihat bahwa nilai yang diapit tanda kurung “()” akan dianggap sebagai tuple oleh program. Anda juga dapat melakukan indexing dan slicingpada tuple sama seperti list. Tuple bersifat immutable yang artinya tidak dapat diubah.

**Set**

Set adalah kumpulan item bersifat unik, tanpa urutan (*unordered collection*), dan dapat diinisialisasikan dengan kurawal “{}” di mana setiap elemennya dipisahkan dengan koma.

Tidak sama seperti list, dalam set kita tidak bisa melakukan indeksing karena set tidak memiliki indeks. Hal ini merujuk pada definisi nya yang menyatakan bahwa set merupakan kumpulan item tanpa urutan. Perhatikan kode di bawah ini.

Program mengembalikan output '**set' object is not subscriptable**karena setiap nilai dalam set tidak memiliki indeks sehingga tidak bisa dilakukan *indexing*.



Selain tanpa urutan (unordered collection). Set juga bersifat unik, artinya, data yang Anda simpan pada set tidak akan ada duplikat. Anda dapat memanfaatkan hal ini untuk menghilangkan duplikat pada suatu data.

Anda dapat melihat bahwa nilai yang diapit tanda kurawal “{}” akan dianggap sebagai set oleh program dan nilai duplikat di dalamnya akan dihapus. Pada kode di atas pun, nilai 3 dan 2 yang duplikat telah dihapus.

Terakhir, set adalah himpunan dalam matematika. Ini maknanya Anda dapat melakukan operasi *union*(gabungan) dan *intersection* (irisan) pada set. Python menyediakan method “.union()” dan “.intersection()” untuk tipe data set.

Method merupakan tindakan atau operasi yang dapat dilakukan oleh suatu objek. Saat ini, tidak apa-apa jika kamu belum memahami sepenuhnya. Anda akan mempelajari lebih detail mengenai *method*pada modul Object-Oriented Programming (OOP).

**Dictionary**

Dictionary pada Python merupakan kumpulan pasangan *key-value*yang bersifat tidak berurutan. Dictionary dapat digunakan untuk menyimpan data kecil hingga besar. Pada Python, dictionary didefinisikan dengan kurawal dan tambahan definisi berikut.

1. Setiap elemen pasangan *key-value* dipisahkan dengan koma (,).
2. *Key* dan *value*dipisahkan dengan titik dua (:).
3. *Key* dan *value*dapat berupa tipe variabel/objek apa pun.

Pada kode di atas, sintaks yang diapit tanda kurawal “{}” dan memiliki pasangan key-value akan dianggap sebagai data bertipe dictionary oleh program.

Dalam mengambil setiap nilai/elemen pada dictionary, Anda harus mengetahui key (kunci) untuk mengakses setiap value-nya (nilai). Hal ini berbeda dengan tipe data sebelumnya yang cukup mengharuskan Anda untuk menyebutkan indeksnya saja.

**Konversi antara Tipe Data**

Setelah mengetahui berbagai tipe data pada Python, Anda pun dapat melakukan konversi antar tipe data dengan menggunakan beberapa fungsi.

Fungsi merupakan blok kode yang dapat dipanggil untuk melakukan tugas tertentu. Anda akan mempelajari fungsi lebih detail pada modul subprogram. Saat ini, Anda cukup memahami bahwa fungsi di bawah ini dapat melakukan operasi terhadap list, set, dan string.

Di bawah ini merupakan berbagai fungsi yang dapat digunakan untuk mengonversi data antar list, set, dan string.

**Transformasi Angka, Karakter, dan String**

Pada materi ini, Anda akan mempelajari berbagai cara mentransformasi tipe data string. Ada banyak method/metode yang dapat membantu Anda mentransformasi tipe data string.

**Mengubah Huruf Besar/Kecil**

Dalam kategori ini terdapat beberapa metode yang dapat digunakan untuk mengubah string menjadi huruf kapital (*UPPERCASE*) atau huruf kecil (*lowercase*). Kedua metode ini, baik upper() maupun lower() adalah metode bawaan dari string Python. Perlu diingat bahwa jika terdapat karakter bukan huruf (seperti simbol atau angka) yang tidak memiliki opsi kapital sehingga karakter tersebut tidak diubah.

**upper() :** Pada kode di atas, Anda mengubah string "dicoding" menjadi uppercasedengan menggunakan method .upper(). Hasilnya, string tersebut berubah menjadi "DICODING".

**lower() :** Pada kode di atas, Anda mengubah string "DICODING" menjadi lowercasedengan menggunakan method .lower(). Hasilnya, string tersebut berubah menjadi "dicoding".

**Awalan dan Akhiran**

Kategori ini adalah metode dalam string yang biasanya dipergunakan untuk menghapus karakter *whitespace* pada suatu string. Namun, selain *whitespace*bisa juga diperuntukkan untuk menghilangkan kata yang tidak diinginkan.

**rstrip()** Metode rstrip() menghapus *whitespace* pada sebelah kanan atau akhir string.

**lstrip()** Kebalikan dari rstrip(), lstrip() bertugas untuk menghapus *whitespace* pada sebelah kiri atau awal string.

**strip()** Metode ini bertugas untuk menghapus *whitespace*pada bagian awal dan akhir string.

Anda dapat mengganti *whitespace*dengan kata atau hal lainnya. Caranya adalah memberikan nilai pada ".strip()". Kode di atas menghapus kata "Code" pada variabel "kata".

**startswith()** Metode startswith() bertujuan untuk menemukan suatu kata pada awal string. Metode ini mengembalikan nilai **True**. Pada kode di atas, Anda mencari string "Dicoding" pada string pertama "Dicoding Indonesia". Operasi ini mengembalikan nilai True karena pada string "Dicoding Indonesia" memang diawali dengan string "Dicoding".

**endswith()** Metode endswith() bertujuan untuk menemukan suatu kata pada akhir string. Metode ini juga mengembalikan nilai **True**jika menemukannya, sedangkan jika tidak menemukan kata yang diinginkan, nilai False dikembalikan.

**Memisah dan Menggabung String**

Kategori selanjutnya adalah memisah dan menggabung string. Anda dapat menggunakan metode ini untuk tujuan tersebut dengan menggunakan dua metode berikut.

print(' '.join(['Dicoding','Indonesia', '!']))

**join()** Pada kode di atas, Anda menggabungkan string "Dicoding", "Indonesia", dan "!" yang telah disimpan pada variabel list. Perhatikan bahwa pada sintaks awal sebelum ".join()" Anda menambahkan *single quotes* di sana. Single quotes ini bermaksud agar Anda menentukan *delimiter* pada setiap kata/nilai yang ingin Anda gabungkan. Pada kode di atas, *delimiter*-nya adalah *whitespace*atau spasi.

**split()** Kebalikan dari metode sebelumnya, metode split() bertujuan untuk memisahkan kata/substring berdasarkan delimiter.

Pada kode di atas, Anda memisahkan string "Dicoding Indonesia !" menjadi "Dicoding",  "Indonesia", dan "!". Perhatikan bahwa kedua string tersebut disimpan sebagai list.

print('Dicoding Indonesia !'.split())

Anda juga bisa menggunakan delimiter ***newline* (\n)**untuk memisahkan setiap baris pada string multiline.

**Mengganti Elemen String**

Kategori selanjutnya merupakan metode yang bertujuan untuk mengganti elemen string di dalamnya dengan elemen string lainnya.

**replace()**

**string = "Ayo belajar Coding di Dicoding"**

**print(string.replace("Coding", "Pemrograman"))**

**Ayo belajar Pemrograman di Dicoding**

Perhatikan kode di atas, Anda mengubah kata "Coding" menjadi "Pemrograman". Perlu diingat bahwa replace() bersifat case-sensitive. Contohnya, Anda dapat lihat dalam kode di atas bahwa kata "coding" pada "Dicoding" tidak ikut berubah. Hanya kata "Coding" saja yang diubah menjadi "Pemrograman". Hal ini karena kata "coding" pada "Dicoding" berawalan huruf c kecil.

**Pengecekan String**

Kategori selanjutnya bertujuan untuk melakukan pengecekan pada string. Hasil dari kategori ini adalah mengembalikan nilai boolean True/False.

**isupper()**

isupper() akan mengembalikan nilai True jika semua huruf dalam string adalah huruf kapital (uppercase). Sebaliknya, jika ada satu huruf saja yang kecil/tidak uppercase, nilai False akan dikembalikan.

kata = 'DICODING'

print(kata.isupper())

True

Pada kode di atas, Anda memastikan variabel kata mengandung string UPPERCASE atau tidak. Jika iya, nilai "True" dikembalikan.

**islower()**

Kebalikannya, islower() akan mengembalikan nilai True jika semua huruf dalam string adalah huruf kecil (lowercase).

**isalpha()**

Metode ini mengembalikan nilai True jika semua karakter dalam string adalah huruf alfabet. Jika ada satu huruf lain yang bukan alfabet, seperti angka, nilai False akan dikembalikan.

**isalnum()**

Metode isalnum() mengembalikan nilai True jika karakter dalam string adalah alfanumerik, yaitu hanya huruf atau hanya angka atau berisi keduanya. Jika tidak, nilai False akan dikembalikan.

**isdecimal()**

Metode isdecimal() akan mengembalikan nilai True jika karakter dalam string berisi hanya angka/numerik. Jika tidak, nilai False akan dikembalikan.

**isspace()**

Metode isspace() akan mengembalikan nilai True jika string hanya berisi whitespace, seperti spasi, tab, newline, atau karakter *whitespace* lainnya.

**istitle()**

Metode istitle() mengembalikan nilai True jika string berisi huruf kapital pada setiap kata pertamanya. Jika tidak, nilai False dikembalikan.

**Formatting pada String**

Kategori terakhir yang akan kita bahas pada modul kali ini adalah *formatting* pada string. Dalam kategori ini terdapat beberapa metode, yaitu zfill(), rjust(), ljust(), center(), dll. Semua metode akan dijelaskan secara detail di bawah ini. Simak baik-baik, ya.

**zfill()**

Metode zfill() bertujuan untuk menambahkan nilai nol (0) di depan sebuah string dengan panjang tertentu. Tujuan dari metode ini adalah membantu untuk memastikan bahwa sebuah angka atau nilai memiliki panjang tetap, terutama ketika ingin menyimpan nilai dalam format yang konsisten.

teks = 'Code'

tambah\_number = teks.zfill(5)

print(tambah\_number)

Output:

0Code

Kita bedah kode di atas lebih detail.

1. Variabel **teks**menyimpan nilai string berupa "Code". Perlu dipahami bahwa kata "Code" hanya memiliki 4 huruf atau sederhananya panjang kata "Code" adalah 4.
2. Variabel **tambah\_number** menyimpan nilai variabel teks dengan memanggil metode zfill(5).  Angka 5 tersebut merupakan parameter untuk menentukan panjang yang diinginkan oleh string. Sederhananya, Anda memastikan bahwa panjang kata "Code" haruslah 5 dan bukan 4. Jadi, program akan menambahkan angka 0 di depan kata "Code" untuk memastikan bahwa panjangnya adalah 5.

Metode zfill() ini berguna ketika ingin memastikan bahwa angka atau nilai dalam string memiliki panjang tetap dan sesuai dengan format yang diinginkan. Metode zfill() dapat diterapkan untuk menetapkan nomor nota atau nomor antrian.

**rjust()**

Metode rjust() berguna untuk merapikan pencetakan teks. Dengan metode rjust() ini, Anda dapat membuat teks menjadi rata kanan sehingga tampilannya lebih rapi.

print('Dicoding'.rjust(20))

Output:

Dicoding

Berdasarkan kode di atas, perhatikan bahwa secara default, rjust() akan menambahkan *whitespace* untuk membuat teks menjorok ke sebelah kanan. Angka 20 yang Anda masukkan bersifat sama seperti pada zfill(). Metode rjust() akan memastikan bahwa panjang teksnya adalah 20 termasuk dengan kata "Dicoding".

Anda bisa menambahkan teks lain, tidak hanya *whitespace*.

**ljust()**

Selanjutnya adalah metode ljust(), metode ini adalah kebalikan dari metode rjust() yang bertujuan untuk membuat teks menjadi rata kiri.

print('Dicoding'.ljust(20))

Output:

Dicoding '

**center()**

Metode center() menjadikan teks rata tengah. Metode ini akan menambahkan *whitespace* di sebelah kiri dan kanan secara default. Anda juga bisa mengganti *whitespace*tersebut dengan karakter lain.

**String Literals**

Umumnya, string ditulis dengan mudah di Python, diapit oleh tanda petik tunggal. Namun, dalam kondisi tertentu, dibutuhkan petik tunggal di tengah string (misalnya, struktur kepemilikan dalam bahasa Inggris—Dicoding's Cat atau penyebutan Jum'at pada hari dalam bahasa Indonesia).

Python akan salah mengira bahwa string berakhir setelah huruf m dan selebihnya merupakan kode yang invalid. Namun, Python memperbolehkan untuk menggunakan petik dua seperti Anda menggunakan petik tunggal. Dalam kasus sebelumnya, Anda cukup mengetik seperti berikut.

st1 = "Jum'at"

Dalam contoh di atas, Python mengenali bahwa petik tunggal adalah bagian tidak terpisahkan dari string tersebut. Bagaimana jika kita memerlukan kedua jenis petik dalam string tunggal? Python menyediakan *escape character*.

*Escape character* memungkinkan Anda untuk menggunakan karakter yang sebelumnya tidak bisa dimasukkan dalam string. Umumnya diawali dengan *backslash* (\) dan diikuti karakter tertentu yang diinginkan. Contohnya, untuk petik tunggal Anda dapat menambahkan **\'**.

Ini merupakan cara paling aman untuk melakukan penambahan atau penyuntingan dalam variabel. Contohnya berikut.

st1 = 'Jum\'at'

Python mengetahui bahwa pada Jum\'at, sebelum petik terdapat *backslash* (\) yang menandakan petik tunggal sebagai bagian dari string dan bukan akhir dari string. *Escape character* \' dan \" memungkinkan Anda untuk memasukkan karakter ' dan '' dalam bagian string. Beberapa contoh *escape character* lainnya sebagai berikut.

* \' Single quote
* \" Double quote
* \t Tab
* \n Newline (*line break*)
* \\ Backslash

Raw String

Python juga menyediakan cara untuk mencetak string sesuai dengan apa pun input atau teks yang diberikan. Metode ini dinamakan raw strings. Umumnya, ia digunakan untuk regex atau beberapa implementasi lain yang sangat bergantung pada keberadaan backslash. Untuk mengimplementasikan raw strings, sisipkan huruf r sebelum pembuka string.

print(r'Dicoding\tIndonesia')

Output:

Dicoding\tIndonesia

**Operasi pada List, Set, dan String**

Dalam modul ini, Anda akan belajar mengenai contoh-contoh operasi pada list, set ,dan string. Banyak fungsi dalam Python yang dapat digunakan untuk melakukan operasi pada list, set, dan string.

**len()**

Fungsi len() bertujuan untuk menghitung panjang atau banyaknya elemen dari list, set, dan string. Khusus pada string, program akan menghitung jumlah karakternya.

**min() dan max()**

Selain menghitung panjang atau banyaknya elemen, Anda juga dapat mengetahui berapa nilai minimum dan maksimum dari suatu list menggunakan fungsi min() dan max().

**Count**

Fungsi count() digunakan untuk mengetahui berapa kali suatu objek muncul dalam list.

**In dan Not In**

**In** dan **not in** merupakan operator yang diperuntukkan untuk mengetahui nilai atau objek yang ada dalam list. Anda bisa menggunakan operator ini untuk memastikan suatu nilai ada dalam list bahkan dalam string. Operator **in** dan **not in** akan mengembalikan nilai boolean True atau False.

kalimat = "Belajar Python di Dicoding sangat menyenangkan"

print('Dicoding' in kalimat)

print('tidak' in kalimat)

print('Dicoding' not in kalimat)

print('tidak' not in kalimat)

Output:

True

False

False

True

Ada empat baris kode di atas. Pada baris pertama, Anda mencari kata atau substring "Dicoding" dalam variabel kalimat. Hasilnya, kata tersebut ada dalam variabel kalimat sehingga mengembalikan nilai True.

Hal ini berlaku sebaliknya pada baris kode ketiga, Anda justru memastikan bahwa substring "Dicoding" **tidak ada**dalam variabel kalimat. Hasilnya tentu False karena kita sudah tahu pada baris kode pertama bahwa substring "Dicoding" ada dalam variabel kalimat.

**Memberikan Nilai untuk Multiple Variable**

Dalam list atau tuple, terkadang Anda perlu memberikan nilai pada variabel-variabel tersebut. Secara konvensional, Anda bisa melakukan hal tersebut dengan menandai indeks yang diinginkan dan memberikan nilai satu per satu sesuai keinginan.

data = ['shirt', 'white', 'L']

apparel = data[0]

color = data[1]

size = data[2]

Dalam kode di atas, Anda mengakses indeks pertama pada variabel "data" yang merupakan list, lalu menyimpannya pada variabel baru bernama "apparel". Lalu, berlaku seterusnya, Anda mengakses indeks kedua serta ketiga dan menyimpannya pada variabel baru, masing-masing bernama "color" dan "size".

Alih-alih melakukannya satu persatu, Anda dapat melakukan hal tersebut sekaligus dalam Python.

Pada kode di atas, Anda melakukan hal yang sama dengan kode sebelumnya. Anda mengakses indeks 0, 1, dan 2 pada variabel "data" yang merupakan list. Namun, alih-alih melakukannya satu persatu, Anda melakukannya sekaligus dalam satu baris kode.

Perlu diperhatikan bahwa jumlah variabel yang ingin Anda masukkan haruslah sama dengan jumlah variabel yang ada pada list atau tuple. Pada variabel data di atas, list yang telah diinisialisasikan beranggota sebanyak tiga dan statement kedua juga melakukan inisialisasi tiga data, yakni "apparel, color, size". Tidak percaya? Silakan ubah kode **"apparel, color, size"** menjadi **"apparel, color, size, size\_chart"**.

**Sort**

Anda bisa menggunakan fungsi sort() untuk mengurutkan angka atau urutan huruf.

kendaraan = ['motor', 'mobil', 'helikopter', 'pesawat']

kendaraan.sort()

print(kendaraan)

Output:

['helikopter', 'mobil', 'motor', 'pesawat']

Pada kode di atas, Anda mengurutkan anggota variabel "kendaraan" yang merupakan list. Perhatikan cara fungsi sort() mengurutkan anggota di dalamnya. Anggota list merupakan string maka akan diurutkan berdasarkan huruf pertamanya dalam alfabet.

Beberapa hal yang perlu kamu ketahui mengenai sort sebagai berikut.

1. Anda dapat membalikkan urutan dengan cara berikut.

kendaraan = ['motor', 'mobil', 'helikopter', 'pesawat']

kendaraan.sort(reverse=True)

print(kendaraan)

Output:

['pesawat', 'motor', 'mobil', 'helikopter']

1. Metode sort tidak dapat mengurutkan list yang memiliki angka dan string sekaligus di dalamnya.

urutan = ['Dicoding', 1, 2, 'Indonesia', 3]

urutan.sort()

print(urutan)

Output:

TypeError: '<' not supported between instances of 'int' and 'str'

Pada kode di atas menghasilkan errorkarena Anda mengurutkan dua tipe data, yakni **string** dan **number**

1. Metode sort menggunakan urutan ASCII sehingga nilai huruf kapital (uppercase) akan lebih dahulu dibandingkan huruf kecil (lowercase). *ASCII (American Standard Code for Information Interchange) table* merupakan sebuah kode karakter yang memetakan set karakter yang umum digunakan ke dalam angka. Sederhananya, tabel ini menampilkan karakter-karakter ASCII beserta nilai angka yang mewakilinya. Metode sort() mengurutkan berdasarkan angka pada ASCII ini. Anda bisa meninjau lebih dalam mengenai ASCII pada link berikut: [ASCII Table](https://www.cs.cmu.edu/~pattis/15-1XX/common/handouts/ascii.html).  
     
   Untuk mengatasi masalah ini, Anda dapat memasukkan keyword "str.lower" pada parameter metode sort(). Jadi, sort() akan menganggap semua objek menggunakan huruf kecil, tanpa mengubah kondisi asli dari objek tersebut.