Topik : 1.4 Federated Computation Primitives

Objective : Memahami tff.federated\_map, tff.federated\_mean

Task : Implementasi operasi sederhana FL: sum, mean antar client

Source : <https://www.tensorflow.org/federated/tutorials/custom_federated_algorithms_1>

**Intended Uses**

Bayangkan Anda ingin melatih sebuah model AI (kecerdasan artifisial), tetapi datanya tidak terkumpul di satu server pusat. Data tersebut tersebar di ribuan perangkat milik pengguna, seperti ponsel atau laptop. **Federated Learning** adalah teknik yang melatih model secara langsung di perangkat-perangkat tersebut tanpa harus mengirim data pribadi pengguna ke server pusat.

Federated Core (FC) Adalah “Mesin” atau lingkungan pemrograman di dalam TensorFlow Federated (TFF) yang memungkinkan kita menulis logika untuk Scenario seperti ini. FC memberikan kita alat untuk mengekspresikan dua hal penting :

1. Kode Komputasi ( Menggunakan Tensorflow) : misalnya, kode untuk melatih model sedikit demi sedikit di ponsel seorang pengguna
2. Kode Komunikasi Terdistribusi : Misalnya, cara mengumpulkan hasil pelatihan dari ribuan ponsel, kemudian menggabungkannya (rata – rata), dan mengirimkan model yang telah diperbarui Kembali ke semua ponsel.

Perbedaan Federated Core (FC) & tf.contrib.distribute

|  |  |  |
| --- | --- | --- |
|  | Federated Core | Tf.contrib.distribute |
| Tujuan | Memberikan control penuh untuk menciptakan algoritma baru yang berjalan di atas data yang memang sengaja terpencar dan tidak boleh di pindahkan | Membuat Pelatihan yang sudah ada menjadi lebih cepat dan efisien |
| Cara Kerja | Data tetap di perangkat pengguna (di luar "gedung"). FC menyediakan bahasa pemrograman untuk mengatur bagaimana model berkelana ke perangkat, berhitung di sana, dan bagaimana hasilnya dikombinasikan. | Anda punya model dan data yang sudah terkumpul di sebuah pusat data (data center) yang penuh dengan GPU/TPU. Framework ini membagi-bagi data dan komputasi ke across semua prosesor tersebut secara otomatis. |

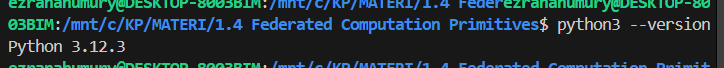
Federated Core ditujukan untuk para peneliti dan engineer yang ingin menciptakan algoritma Federated Learning yang baru. Misalnya, jika anda ingin mencoba cara baru untuk merata – ratakan model, atau pola komunikasi yang berbeda antar perangkat.

 FC beroperasi pada level yang setara dengan **pseudocode** (kode semu) dalam makalah penelitian. Anda bisa mendeskripsikan ide ("kirim model ke klien, latih, ambil update, rata-ratakan") tanpa harus pusing dengan detail teknis jaringan seperti protokol socket dan pertukaran pesan point-to-point.

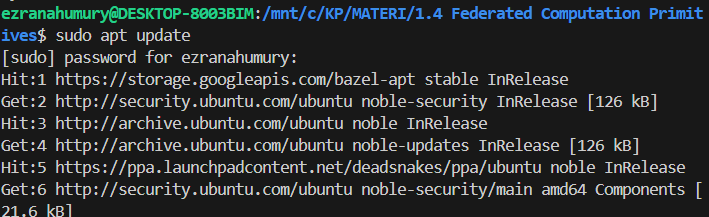
**Pengujian TFF**

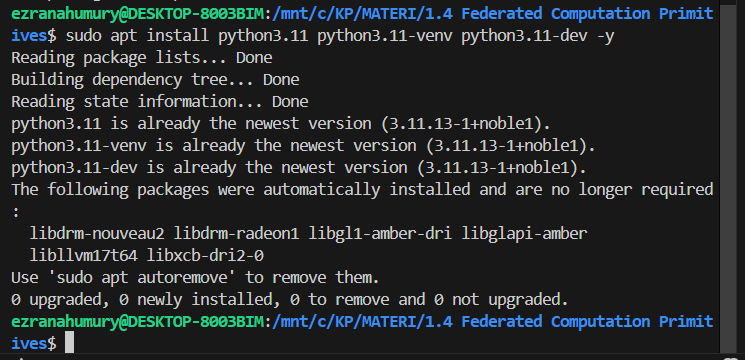
Mencoba menjalankan “Hello World” untuk memastikan bahwa lingkungan sudah diatur dengan benar :

1. Melakukan pengecekan Versi Python dikarenkan TFF hanya mensupport Python Versi 3.9 – 3.11

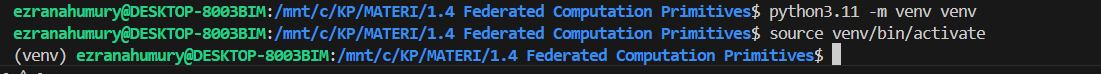


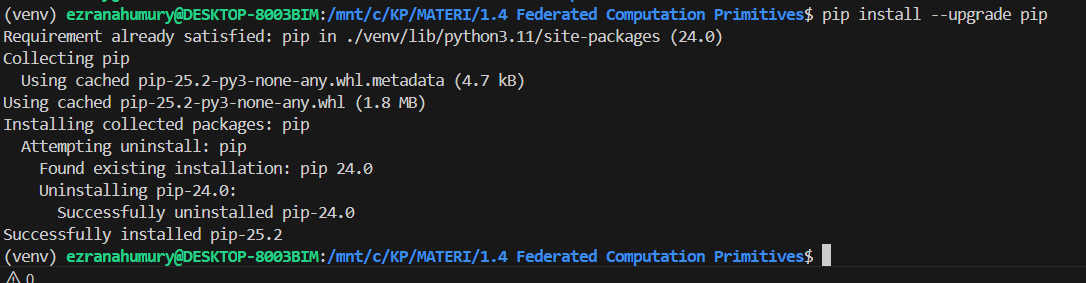
1. Melakukan penginstallan python terbaru yang mensupport TFF



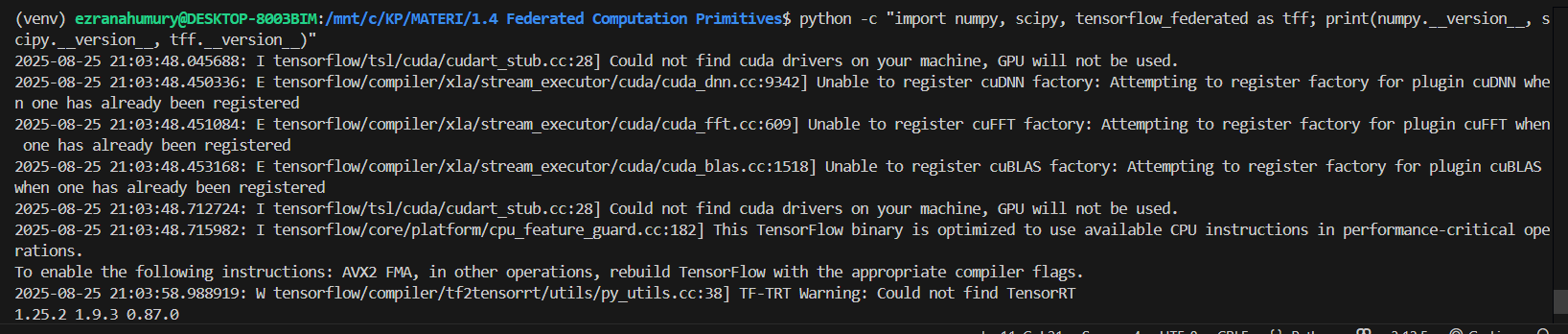


1. Membuat dan mengaktivkan environment

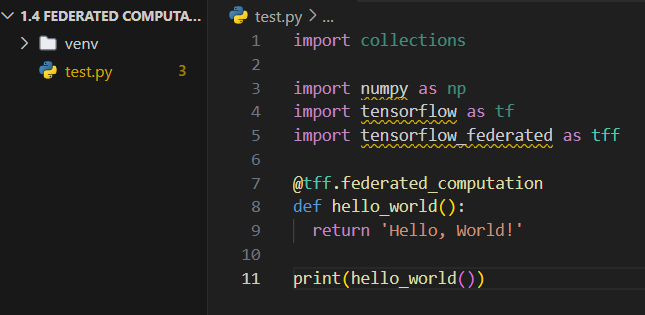


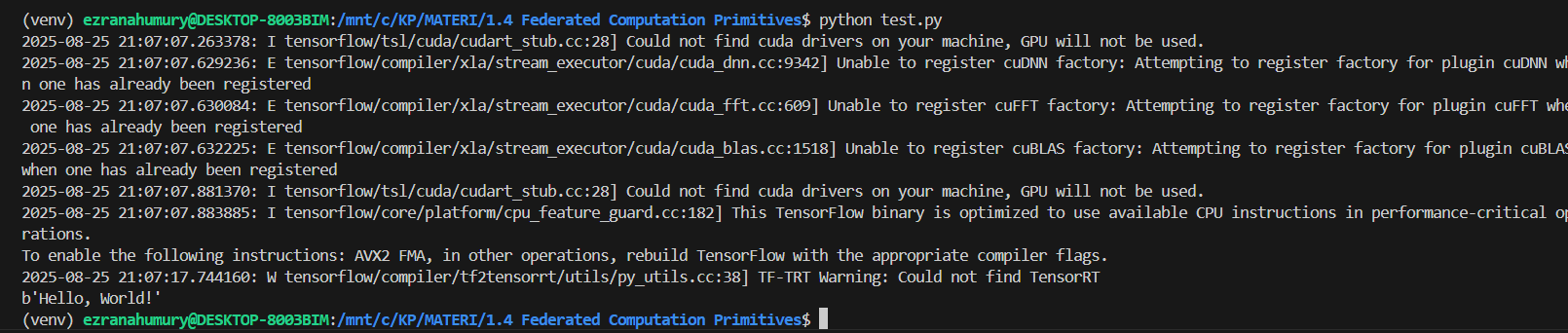


1. Melakukan pengecekan versi Numpy, SciPy, TFF



1. Membuat file untuk melakukan pengecekan TFF dengan mencoba “Hello World”





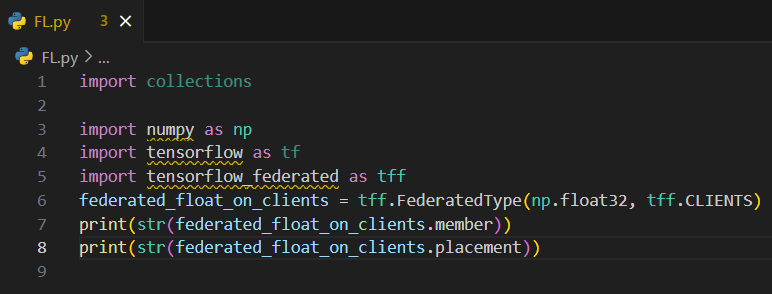
**FEDERATED DATA**

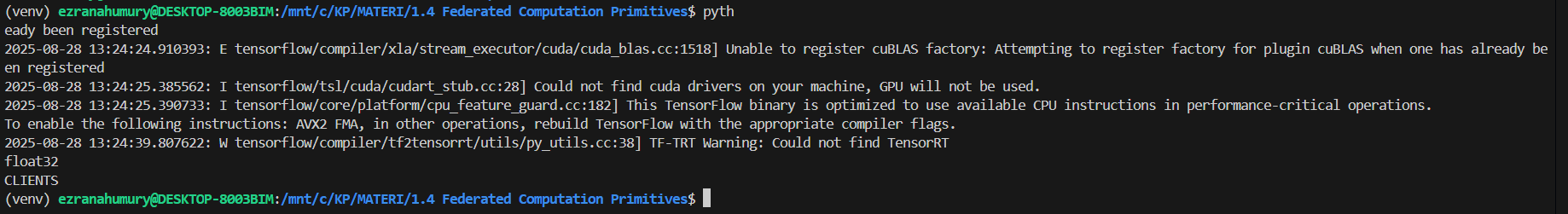
Salah satu fitur pembeda dalam TFF Adalah kemampuannya untuk mengeskpresikan komputasi berbasis TensorFlow pada data terfederasi secara ringkas. Dalam tutorial ini, kita akan menggunakan istilah Data Terfederasi untuk merujuk pada Kumpulan item data yang dihosting di sekelompok perangkat dalam system terdistribusi. Misalnya, aplikasi yang berjalan pada perangkat seluler dapat mengumpulkan data dan menyimpannya secara lokal, tanpa mengunggahnya ke lokasi terpusat. Atau, sejumlah sensor terdistribusi dapat mengumpulkan dan menyimpan pembacaan suhu di lokacsi masing-masing.

Data terfederasi seperti contoh diatas diperlakukan dalam TFF sebagai entitas utama (First-class citizens), artinya data tersebut dapat muncul sebagai parameter dan hasil fungsi, dan mereka memiliki tipe

Poin penting yang harus dipahami adalah bahwa kita memodelkan seluruh kumpulan item data di semua perangkat (misalnya, seluruh kumpulan pembacaan suhu dari semua sensor dalam sebuah array terdistribusi) sebagai satu nilai terfederasi tunggal.

Contoh cara mendefinisikan tipe float terfederasi yang dihosting oleh sekelompok perangkat klien dalam TFF. Kumpulan pembacaan suhu yang terwujud di sepanjang array sensor terdistribusi dapat dimodelkan sebagai nilai dari tipe terfederasi ini.



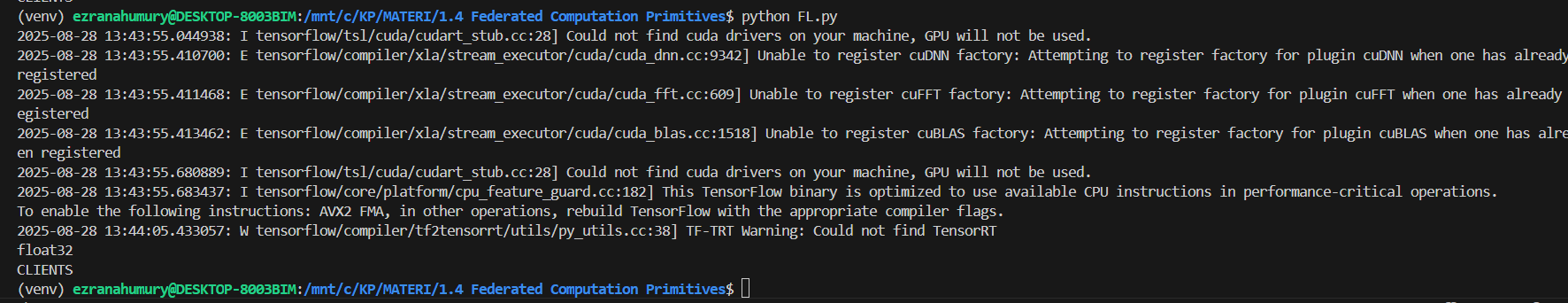


Jenis federasi di TFF didefinisikan dengan menentukan jenis T dan konstituen anggotanya ( item data yang berada pada perangkat individu) dan jenis G yaitu perangkat yang nilai federasi jenis ini di-host ( ditambahkan sedikit informasi opsional)

|  |  |  |
| --- | --- | --- |
|  | Jenis T | Jenis G |
| Code |  |  |
| Pengertian | Tipe dari data individual yang berada pada setiap perangkat secara terpisah. | Kelompok atau lokasi di mana nilai federasi tersebut "hidup" atau dikumpulkan. Ini menunjukkan cakupan dari nilai tersebut. |
| Contoh | Float, String | Client, Server |

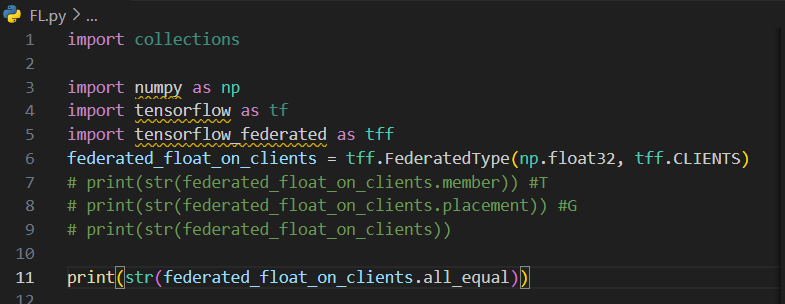
Jenis Federasi dengan konstituen anggota T dan Penempatan G dapat direpresentasikan kompak (digabung) sebagai {T}@G, seperti di bawah ini

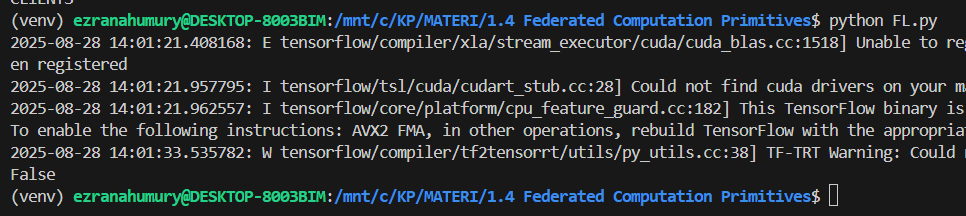




Tipe Federasi du TFF hadir dalam dua rasa :

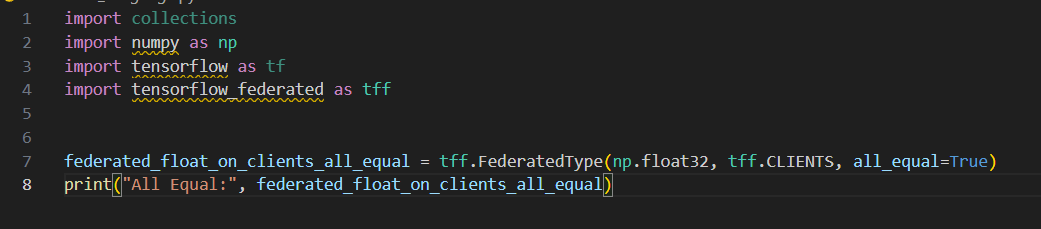
1. Tipe Dimana konstituen anggota dari nilai federasi mungkin berbeda ( ini Adalah default dan paling umum ) code nya ( all\_equal=False )
2. Tipe yang diketahui semuanya sama (harus sama semua) codenya ( all\_equal=True )

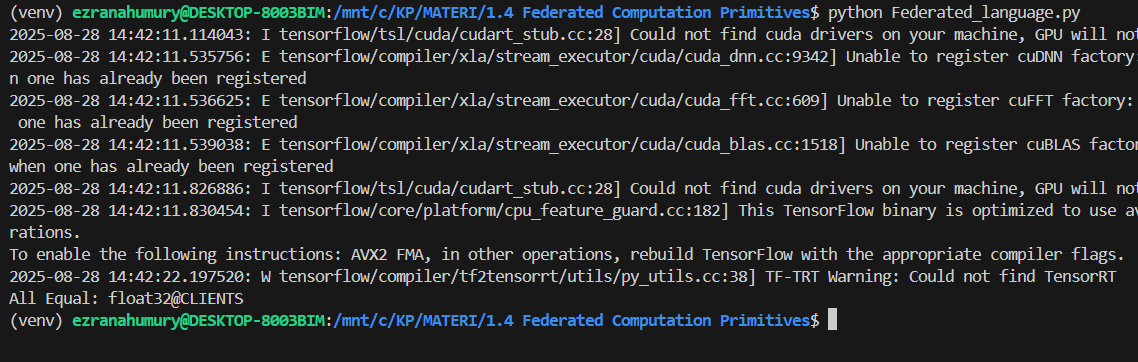




T@G vs {T}@G

|  |  |
| --- | --- |
| {T}@G | T@G |
| all\_equal=False | all\_equal=True |
| multi-set (kumpulan di mana anggota bisa berulang) dari nilai-nilai bertipe T yang berasal dari penempatan G. Nilai-nilai ini boleh berbeda-beda antara satu perangkat dengan perangkat lainnya. | Sebuah nilai bertipe T yang **sama persis** ada (atau dibroadcast) ke **setiap perangkat** dalam penempatan G. |
| **Kurung Kurawal**{ } melambangkan sebuah **kumpulan** atau **tas** (bag) yang berisi banyak item. | **Tidak ada kurung kurawal** menandakan bahwa ini adalah **sebuah nilai tunggal**, bukan sebuah kumpulan. Nilai tunggal ini diduplikasi dan hadir di semua perangkat di G. |

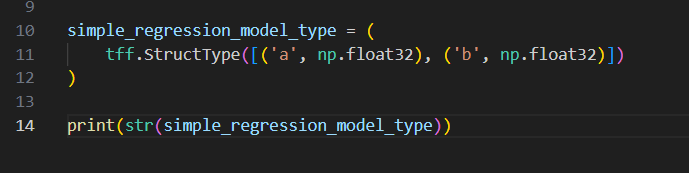


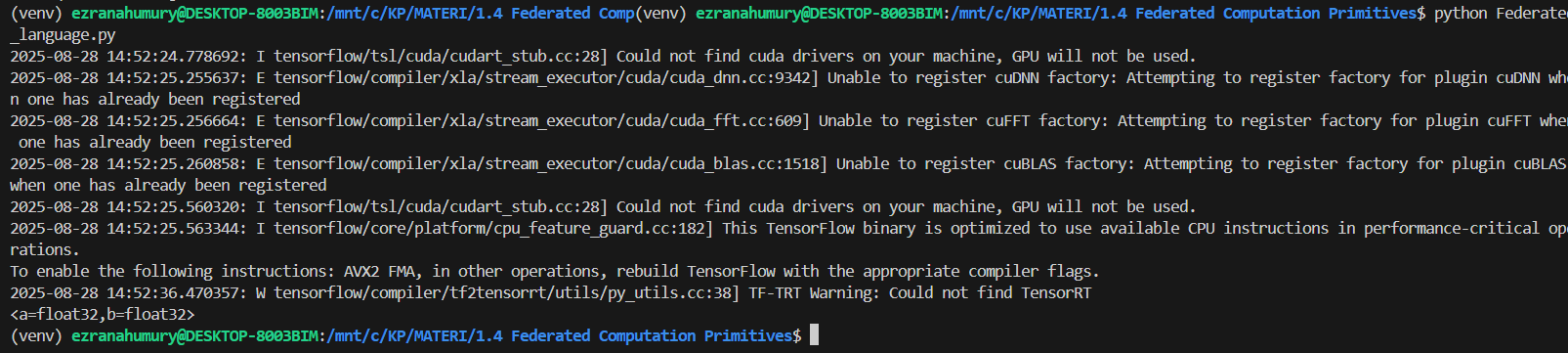


Salah satu contoh nilai gabungan dari jenis tersebut yang mungkin muncul dalam scenario praktis Adalah hyperparameter (seperti kecepatan belajar, norma kliping, dll) yang telah disiarkan oleh server ke sekelompok perangkat yang berpartisipasi dalam pelatihan gabungan.

Contoh lain adalah seperangkat parameter untuk model pembelajaran mesin yang telah dilatih sebelumnya di server, yang kemudian disiarkan ke sekelompok perangkat klien, di mana parameter tersebut dapat dipersonalisasi untuk setiap pengguna.

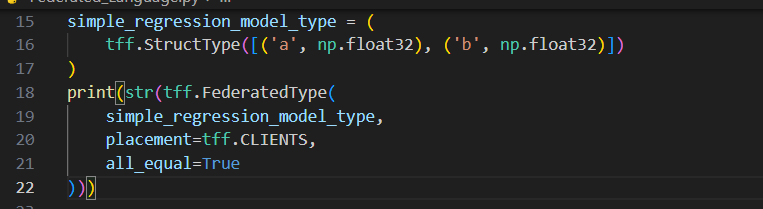
Sebagai contoh, misalkan kita memiliki sepasang float32 parameter a dan b untuk model regresi linier sederhana satu dimensi. Kita dapat membangun tipe (non-federasi) model tersebut untuk digunakan dalam TFF sebagai berikut. Kawat gigi sudut < > di tipe string dicetak adalah notasi TFF kompak untuk bernama atau tidak disebutkan namanya tupel.

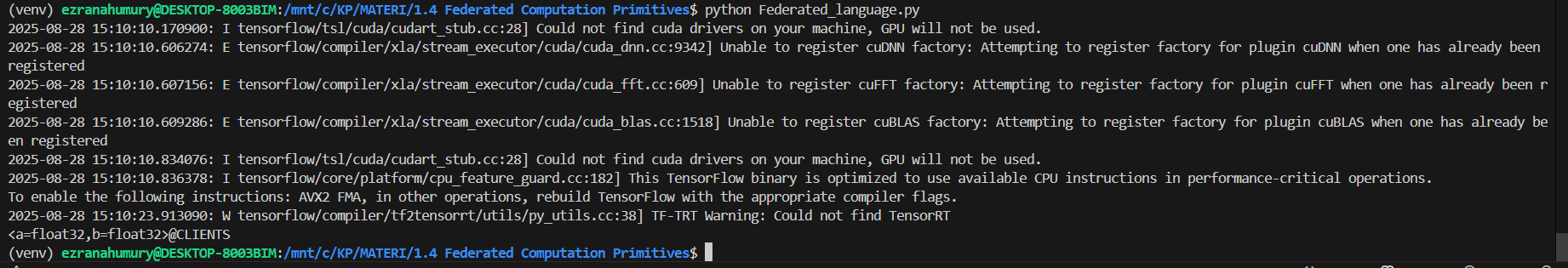




Penting untuk dipahami bahwa tf.float32 hanyalah notasi singkat yang mewakili skenario paling dasar—sebuah tensor dengan bentuk kosong shape=[] (hanya satu angka skalar). Dalam dunia nyata, data federated learning hampir selalu lebih kompleks dari sekadar satu angka; data tersebut bisa berupa vektor bobot model, gambar, atau kumpulan teks. Oleh karena itu, untuk kejelasan dan ketepatan, selalu gunakan tff.TensorType secara eksplisit dengan menentukan shape-nya.

Ketika model ini disiarkan ke klien, jenis nilai gabungan yang dapat di hasilkan dapat direpresentasikan seperti di bawah ini :





Per simetri dengan *mengapung Federasi* di atas, kita akan merujuk pada jenis seperti *tupel federasi.* Lebih umum, kita akan sering menggunakan *XYZ Federasi* istilah untuk merujuk ke nilai federasi di mana konstituen anggota yang *XYZ* -seperti. Dengan demikian, kita akan berbicara tentang hal-hal seperti *tupel federasi,* *urutan federasi,* *model federasi,* dan sebagainya.

Apa itu "Isomorfisme" antara T dan T@G?

Bayangkan Anda memiliki **sebuah file PDF** (T). Itu hanya ada di komputer server Anda.

Sekarang, Anda **mengirim file PDF yang sama persis itu ke 100 laptop karyawan** (T@tff.CLIENTS). Sekarang ada 100 salinan file yang identik.

**"Isomorfisme"** artinya kedua keadaan ini pada intinya adalah setara untuk banyak keperluan:

1. **Memiliki 1 file di server** (T)
2. **Memiliki 100 salinan file yang sama di laptop karyawan** (T@tff.CLIENTS)

Kedua keadaan merepresentasikan **informasi yang sama persis**. Jika isi file di server berubah, semua salinan di laptop juga harus diperbarui agar tetap sama.

Kegunaan Membuat Jenis T@G?

Membuat jenis T@G itu seperti \*\* memberikan alamat pengiriman pada sebuah paket\*\*.

* **Isi paketnya** adalah T (contoh: sebuah buku).
* **Alamatnya (**G**)** menentukan apakah paket itu ada di **gudang pusat** (@SERVER) atau sudah **diantarkan ke setiap toko cabang** (@CLIENTS).

**Placements**

TFF dirancang dengan filosofi yang berbeda dari kebanyakan framework. Alih-alih berfokus pada di mana suatu operasi dijalankan, TFF berfokus pada di mana data berada dan bagaimana data itu berubah. **Placement)** dalam TFF adalah sifat dari **data itu sendiri**, bukan dari operasinya. Data bisa berada di server (@SERVER) atau di kumpulan klien (@CLIENTS).

Dengan secara eksplisit menandai data sebagai {T}@CLIENTS (data sensitif di perangkat user), TFF dapat secara otomatis mencegah operasi yang salah. Framework ini dirancang untuk **tidak pernah mengizinkan** data mentah dari klien dibawa keluar dari konteks yang aman. Semua operasi yang Anda tulis akan berlaku untuk **seluruh grup klien secara kolektif**, bukan untuk individu tertentu.

TFF menyediakan dua "alamat" dasar yang sudah jadi untuk memudahkan pemrograman:

1. tff.SERVER →  **"Pusat Komando"**. Biasanya hanya ada satu. Di sinilah model global disimpan dan proses koordinasi dilakukan.
2. tff.CLIENTS → **"Lapangan"** atau **"Perangkat User"**. Ini bisa mewakili ribuan ponsel, sensor, atau database yang terpisah-pisah.

TFF menggunakan model **client-server** yang intuitif:

* **Server** bertindak sebagai koordinator pusat.
* **Klien** adalah perangkat-perangkat individual yang memiliki data.

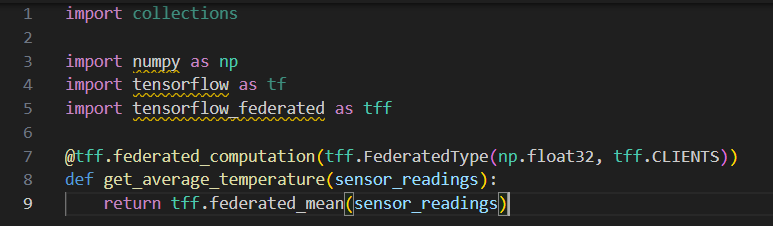
**Fleksibilitas yang Penting:**  
Yang menarik, TFF **tidak mendikte secara kaku** apa sebenarnya CLIENTS dan SERVER itu. Ini memberimu kebebasan:

* tff.CLIENTS bisa berarti 1.000 **ponsel**, atau 50 **database** yang tersebar, atau bahkan 3 **komputer** dalam lab.
* tff.SERVER bisa berupa sebuah **aplikasi cloud**, sebuah **komputer tunggal**, atau bahkan sebuah **proses dalam simulator**.

**Federated Computations**

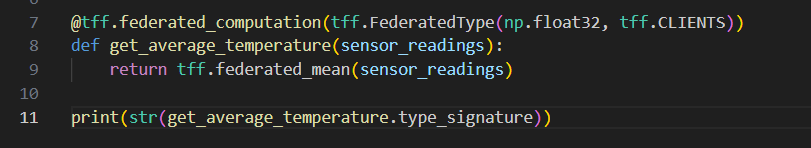
TFF dirancang sebagai lingkungan pemrograman fungsional bertipe kuat yang mendukung pengembangan modular. Unit dasar dari komposisi di TFF merupakan Federated Computations yang merupakan bagian dari logika yang dapat menerima nilai – nilai federasi sebagai masukan dan Kembali nilai – nilai federasi sebagai sebuah output.

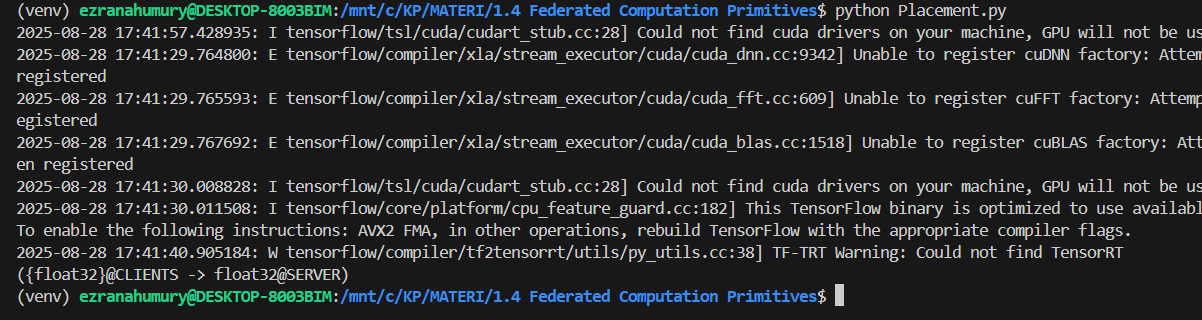
Inilah cara menentukan perhitungan yang menghitung rata – rata suhu yang di laporkan oleh larik sensor :



Kode yang dihasilkan oleh tff.federated\_computation bukanlah tensorflow, juga bukan python. Melainkan spesifikasi system terdistribusi dalam internal platform-independent glue language.

Pertama, mari kita bermain dengan definisi sedikit. Perhitungan TFF umumnya dimodelkan sebagai fungsi - dengan atau tanpa parameter, tetapi dengan tanda tangan tipe yang terdefinisi dengan baik. Anda dapat mencetak tanda tangan jenis perhitungan dengan query yang type\_signature properti, seperti yang ditunjukkan di bawah ini.





Tanda tangan tipe memberi tahu kita bahwa komputasi menerima kumpulan pembacaan sensor yang berbeda pada perangkat klien, dan mengembalikan rata-rata tunggal di server.

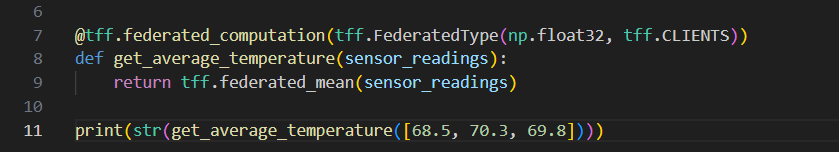
Dalam banyak scenario praktis, perhitungan yang mewakili tugas singkat atas cenderung menerima masukan dan melaporkan keluaran di server- hal ini mencerminkan gagasan bahwa perhitungan mungkin dipicu oleh query yang berasal dan berakhir di server.

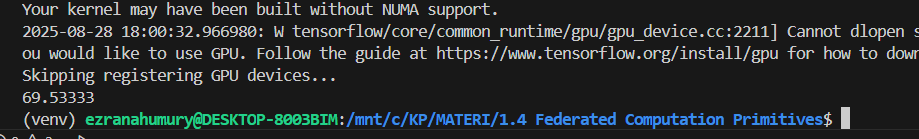
Secara umum, Anda tidak boleh memandang komputasi federasi sebagai sesuatu yang berjalan di server atau dieksekusi oleh server. Server hanyalah salah satu jenis peserta dalam komputasi federasi. Dalam memikirkan mekanisme komputasi semacam ini, sebaiknya selalu berangkat dari perspektif jaringan global secara keseluruhan, bukan perspektif koordinator terpusat tunggal.

Secara umum, jenis tanda tangan fungsional kompak direpresentasikan sebagai (T -> U) untuk jenis T dan U input dan output, masing-masing. Jenis parameter formal (seperti sensor\_readings dalam hal ini) ditetapkan sebagai argumen untuk dekorator. Anda tidak perlu menentukan jenis hasil - itu ditentukan secara otomatis.

**Executing Federated Computations**

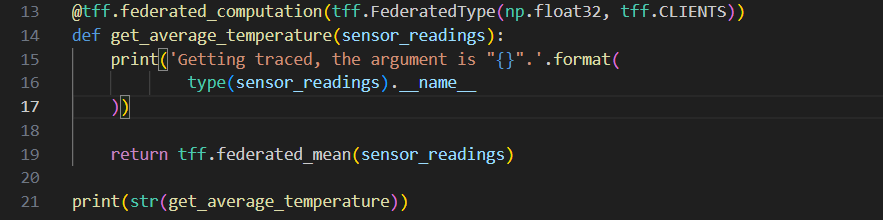
Untuk mendukung pengembangan dan debugging, TFF memungkinkan kita untuk secara langsung menjalankan komputasi yang didefinisikan dengan cara ini sebagai fungsi python, seperti yang ditunjukkan di bawah ini. Dimana perhitungan mengharapkan nilai tipe Federasi dengan all\_equal bit set ke False , Anda dapat memberikannya sebagai daftar biasa dalam Python. Sedangkan untuk tipe federasi dengan bit all\_equal disetel ke True, Anda dapat langsung memberikan konstituen anggota tunggalnya. Inilah cara hasilnya dilaporkan kembali kepada Anda.

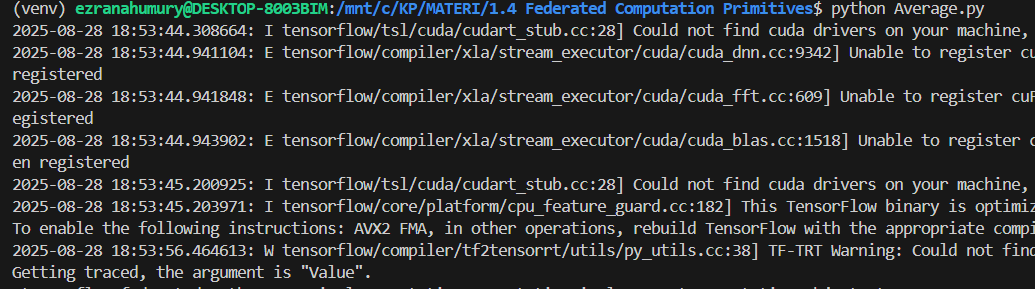




Ketika interpreter python bertemu fungsi dihiasi denga tff.federated\_computation, interpreter tersebut akan menelusuri pernyataan – pernyataaan dalam tubuh fungsi tersebut sekali ( pada saat definisi) dan kemudian membangun representasi serial dari logika komputasi tersebut untuk digunakan dimasa depan (baik untuk eksekusi, maupun untuk dimasukkan sebagai sub-komponen ke dalam komputasi lain).

Anda dapat memverifikasi ini dengan menambahkan pernyataan cetak, sebagai berikut:





Perhitungan TFF mengekspresikan perilaku global system terdistribusi, berbeda dengan program python yang mengekspresikan perilaku local masing – masing peserta. Kita dapat melihat bahwa dalam contoh sederhana di atas, dengan operator khusus tff.federated\_mean yang menerima data pada perangkat klien, tapi deposit hasil pada server.

Operator tff.federated\_mean tidak dapat dengan mudah dimodelkan sebagai operator biasa di Python, karena tidak mengeksekusi secara lokal - seperti disebutkan sebelumnya, itu merupakan sistem terdistribusi yang koordinat perilaku beberapa peserta sistem. Kita akan mengacu ke operator seperti *operator federasi,* untuk membedakan mereka dari biasa operator (lokal) di Python.

**Composing federated computations**

Perhitungan get\_average\_computation dapat di panggil dalam tubuh fungsi python lain yang dihiasi dengan tff.federated\_computation. Hal tersebut akan menyebabkan itu tertanam dalam tubuh induk, banyak cara yang sama tff.federated\_mean tertanam dalam tubuh sendiri sebelumnya.

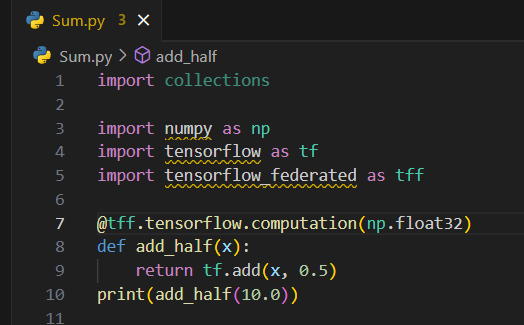
Pembatasan penting untuk menyadari adalah bahwa tubuh fungsi Python dihiasi dengan tff.federated\_computation harus terdiri hanya dari operator federasi, yaitu, mereka tidak dapat secara langsung mengandung operasi TensorFlow. Misalnya, Anda tidak bisa langsung menggunakan tf.nest antarmuka untuk menambah sepasang nilai federasi. Kode TensorFlow harus terbatas pada blok kode dihiasi dengan tff.tf\_computation dibahas dalam bagian berikut. Hanya ketika dibungkus dengan cara ini dapat kode TensorFlow dibungkus dipanggil dalam tubuh seorang tff.federated\_computation .

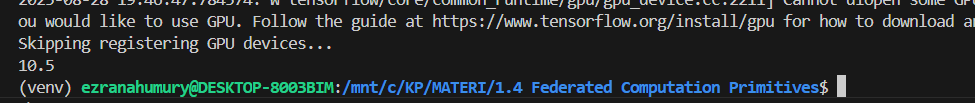
Alasan untuk pemisahan ini adalah teknis (sulit untuk mengelabui operator seperti tf.add bekerja dengan non-tensor) serta arsitektur. Bahasa perhitungan federasi (yaitu, logika yang dibangun dari tubuh serial fungsi Python dihiasi dengan tff.federated\_computation ) dirancang untuk melayani sebagai bahasa lem platform-independen. Bahasa lem ini saat ini digunakan untuk membangun sistem terdistribusi dari bagian tertanam kode TensorFlow (terbatas tff.tf\_computation blok). Dalam kepenuhan waktu, kami mengantisipasi kebutuhan untuk bagian menanamkan dari lainnya, logika non-TensorFlow, seperti query database relasional yang mungkin mewakili pipa input, semua terhubung bersama-sama menggunakan bahasa lem sama ( tff.federated\_computation blok).

**Logika TensorFlow**

TFF dirancang untuk digunakan dengan tensorflow. Dengan demikian, Sebagian besar kode yang akan ditulis dalam TFF kemungkinan Adalah TensorFlow biasa ( yaitu, yang dijalankan secara local). Untuk menggunakan kode tersebut dengan TFF, seperti disebutkan di atas, hanya perlu dihiasi dengan tff.tf\_computation.

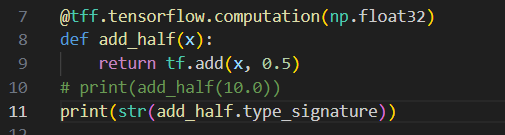
Sebagai contoh, berikut Adalah cara untuk menerapkan fungsi yang mengambil nomor dan menambahkan 0.5 :





Untuk menyematkan blok penyusun yang dapat digunakan kembali yang diimplementasikan menggunakan kode TensorFlow di badan komputasi gabungan, blok penyusun tersebut harus memenuhi properti tertentu - seperti dilacak dan diserialisasikan pada waktu definisi, memiliki tanda tangan tipe, dll. Ini umumnya memerlukan beberapa bentuk dekorator.

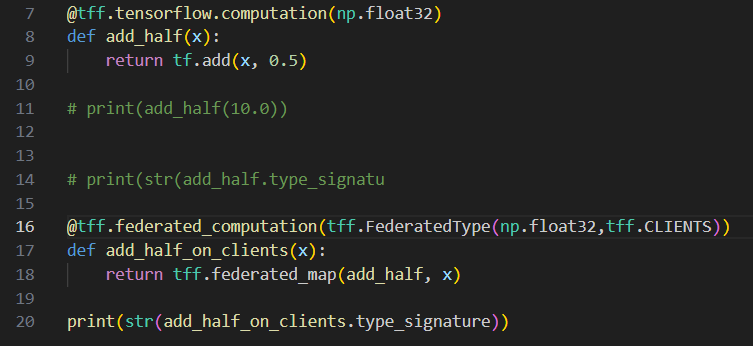
Secara khusus, ia memiliki tanda tangan tipe TFF.

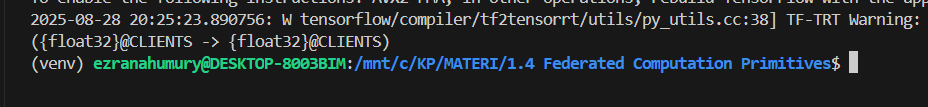




Perhatikan tanda tangan jenis ini tidak memiliki penempatan. Komputasi TensorFlow tidak dapat menggunakan atau mengembalikan tipe federasi.

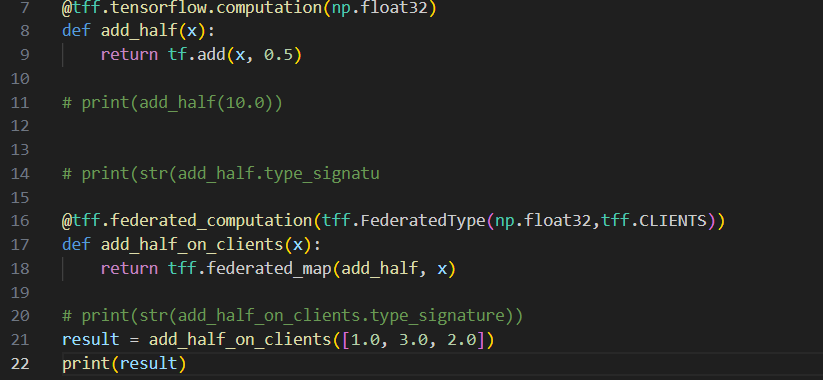
Anda sekarang dapat juga menggunakan add\_half sebagai sebuah blok bangunan dalam perhitungan lainnya. Sebagai contoh, berikut adalah cara Anda dapat menggunakan tff.federated\_map operator untuk menerapkan add\_half pointwise ke semua konstituen anggota pelampung Federasi pada perangkat klien.

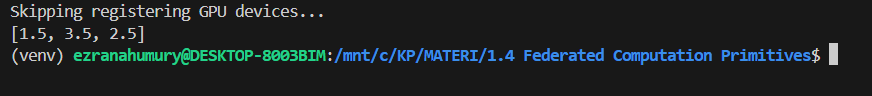




**Menjalankan Komputasi Tensorflow**

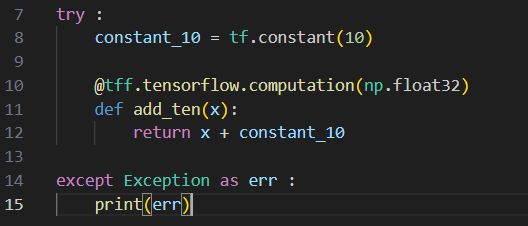
Pelaksanaan perhitungan didefinisikan dengan tff.tf\_computation mengikuti aturan yang sama seperti yang dijelaskan untuk tff.\_federated\_computation. Mereka dapat dipanggil sebagai callable biasa dengan python, sebagai berikut :





perhitungan add\_half\_on\_clients dengan cara ini mensimulasikan proses didistribusikan. Data dikonsumsi pada klien, dan dikembalikan pada klien. Memang, perhitungan ini membuat setiap klien melakukan tindakan lokal. Tidak ada tffSERVER disebutkan secara eksplisit dalam sistem ini (bahkan jika dalam prakteknya, mendalangi pemrosesan tersebut mungkin melibatkan satu). Pikirkan perhitungan didefinisikan dengan cara ini sebagai konseptual analog dengan Map panggung MapReduce .

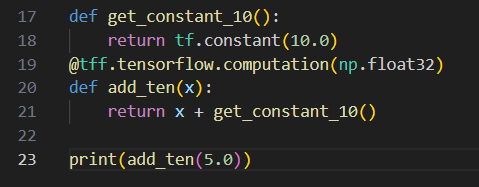
Perbedaan satu-satunya antara metode Python yang diberi dekorasi dengan federated\_language.federated\_computation dan yang diberi dekorasi dengan tff.tensorflow.computation adalah bahwa yang terakhir diserialisasikan sebagai grafik TensorFlow (sedangkan yang pertama tidak diperbolehkan mengandung kode TensorFlow yang tertanam langsung di dalamnya).





Di atas gagal karena constant\_10 telah dibangun di luar grafik yang tff.tf\_computation konstruksi internal di tubuh add\_ten selama proses serialisasi.

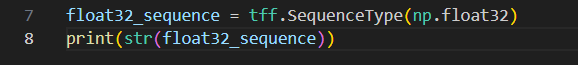
Di sisi lain, memanggil fungsi python yang memodifikasi grafik saat saat dipanggil di dalam tff.tf\_computation baik-baik saja:

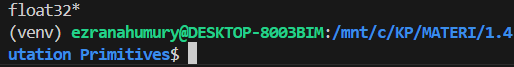


**Working with tf.data.Datasets**

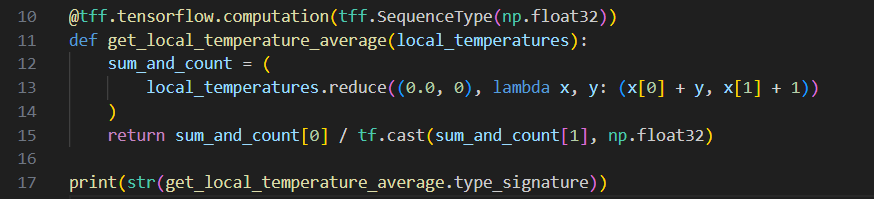
Sebuah fitur unik dari tff.tf\_computation s adalah bahwa mereka memungkinkan Anda untuk bekerja dengan tf.data.Dataset s didefinisikan secara abstrak sebagai parameter formal dengan kode Anda. Parameter yang akan diwakili dalam TensorFlow sebagai data set harus dideklarasikan menggunakan tff.SequenceType konstruktor.

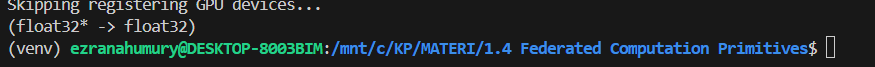
Sebagai contoh, jenis spesifikasi tff.SequenceType(tf.float32) mendefinisikan urutan abstrak elemen mengapung di TFF. Urutan dapat berisi tensor, atau struktur bersarang yang kompleks (kita akan melihat contohnya nanti). The ringkas representasi dari urutan T -typed item adalah T\* .





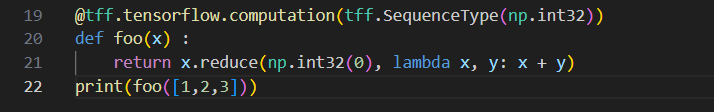
Misalkan dalam contoh sensor suhu kita, setiap sensor tidak hanya menyimpan satu pembacaan suhu, tetapi beberapa pembacaan. Berikut adalah cara Anda dapat mendefinisikan perhitungan TFF di TensorFlow yang menghitung rata-rata suhu dalam satu set data lokal menggunakan operator tf.data.Dataset.reduce.

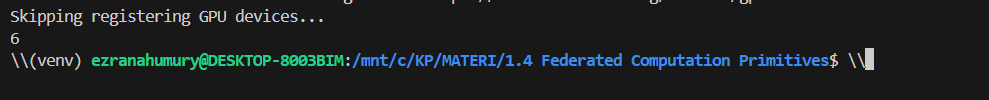




Dalam tubuh metode yang diberi dekorasi dengan tff.tf\_computation, parameter formal dari tipe urutan TFF diwakili sebagai objek yang berperilaku seperti tf.data.Dataset, yaitu mendukung properti dan metode yang sama (saat ini mereka tidak diimplementasikan sebagai subkelas dari tipe tersebut - hal ini mungkin berubah seiring dengan perkembangan dukungan untuk kumpulan data di TensorFlow).

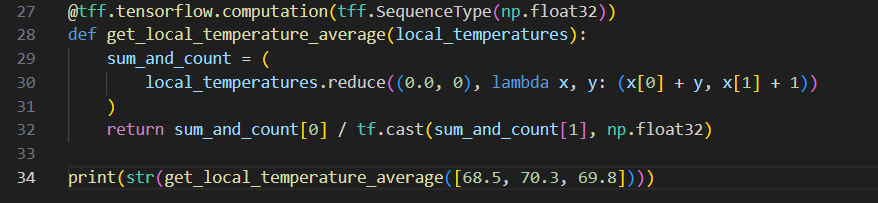
Anda dapat dengan mudah memverifikasi hal ini sebagai berikut :

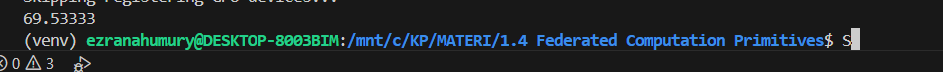




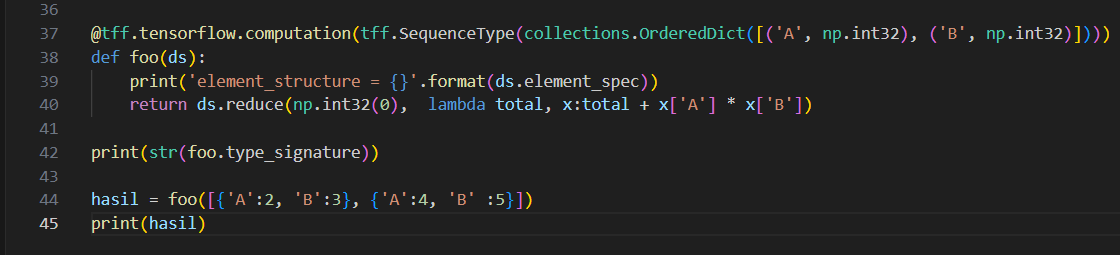
Berbeda dengan tf.data.Dataset biasa, objek-objek serupa dataset ini hanyalah placeholder. Mereka tidak mengandung elemen apa pun, karena mewakili parameter bertipe urutan abstrak yang akan diikat ke data konkret saat digunakan dalam konteks konkret. Dukungan untuk dataset placeholder yang didefinisikan secara abstrak masih terbatas pada tahap ini, dan pada awal pengembangan TFF, Anda mungkin menemui beberapa batasan, tetapi kita tidak perlu khawatir tentang hal itu dalam tutorial ini (silakan merujuk ke halaman dokumentasi untuk detailnya).

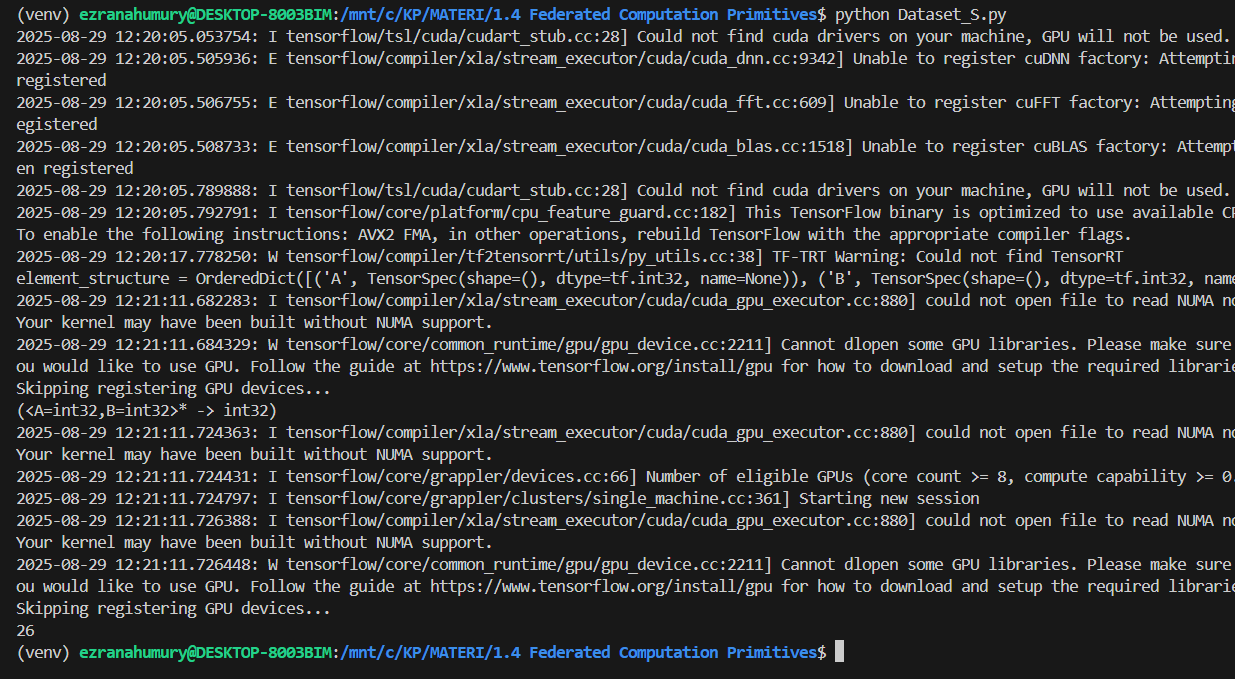
Saat menjalankan perhitungan secara lokal yang menerima urutan dalam mode simulasi, seperti dalam tutorial ini, Anda dapat memasukkan urutan sebagai daftar Python, seperti di bawah ini (serta dengan cara lain, misalnya sebagai tf.data.Dataset dalam mode eager, tetapi untuk saat ini, kita akan menjaga kesederhanaannya).





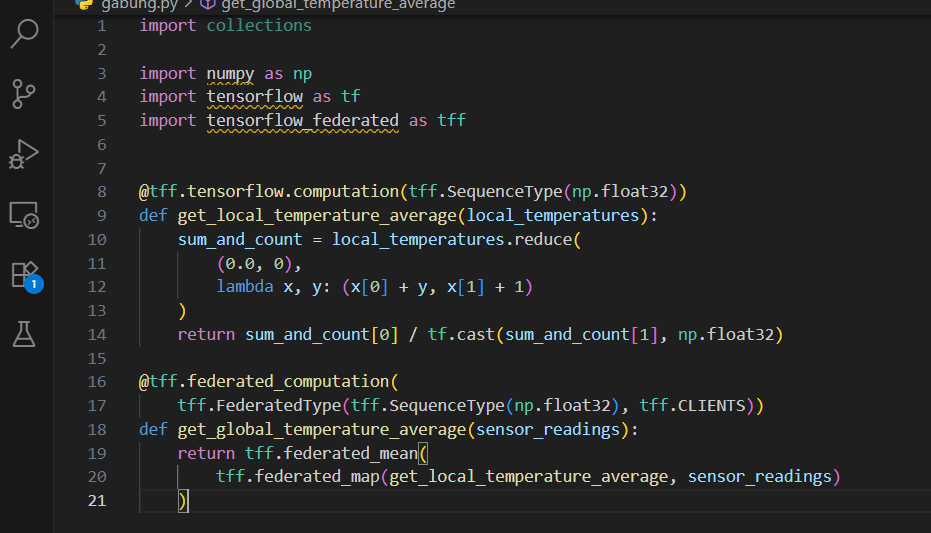
Seperti semua jenis TFF lainnya, urutan seperti yang didefinisikan di atas dapat menggunakan konstruktor tff.StructType untuk mendefinisikan struktur bersarang. Misalnya, berikut adalah cara mendeklarasikan perhitungan yang menerima urutan pasangan A, B, dan mengembalikan jumlah hasil perkaliannya. Kami menyertakan pernyataan pelacakan dalam tubuh perhitungan agar Anda dapat melihat bagaimana tanda tangan jenis TFF diterjemahkan menjadi output\_types dan output\_shapes dataset.





**PENGGABUNGAN SEMUANYA**

Menggunakan komputasi tensoflow dalam pengaturan federasi. Misalkan kita memiliki sekelompok sensor yang masing – masing memiliki urutan pembacaan suhu local. Kita dapat menghitung rata- rata suhu global dengan mengumpulkan rata – rata local dari sensor – sensor tersebut sebagai berikut :



Ini bukan rata-rata sederhana dari semua pembacaan suhu lokal dari semua klien, karena hal itu akan memerlukan penimbangan kontribusi dari klien yang berbeda berdasarkan jumlah pembacaan yang mereka simpan secara lokal. Kami menyisakan hal ini sebagai latihan bagi pembaca untuk memperbarui kode di atas; operator tff.federated\_mean menerima bobot sebagai argumen kedua opsional (diharapkan berupa federated float).

Input untuk get\_global\_temperature\_average kini menjadi urutan float federated. Urutan federated adalah cara umum untuk mewakili data di perangkat dalam pembelajaran federated, dengan elemen urutan biasanya mewakili batch data.





Menjalankan perhitungan secara lokal pada sampel data di Python. Perhatikan bahwa cara kita memberikan input sekarang berupa daftar dari daftar. Daftar luar mengiterasi perangkat dalam grup yang diwakili oleh tff.CLIENTS, sedangkan daftar dalam mengiterasi elemen dalam urutan lokal masing-masing perangkat.

