KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET, DAN TEKNOLOGI
REPUBLIK INDONESIA, 2021

Informatika untuk SMA Kelas X

Penulis : Hanson Prihantoro Putro

ISBN : 978-602-244-506-7

Bab 6
Analisis Data

Sumber: https://www.michiganstateuniversityonline.com/resources/business-analytics/types-of-data-analytics-and-how-to-apply-them/

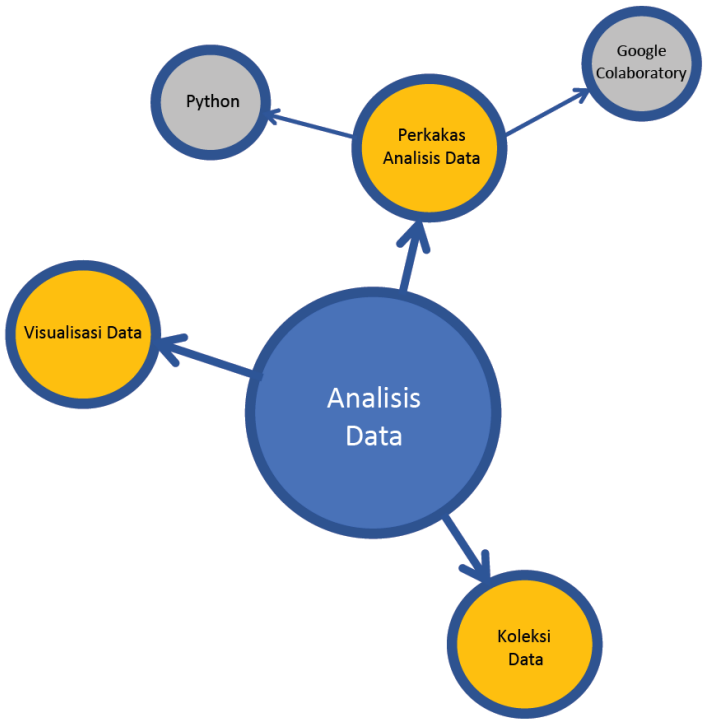
Tujuan Pembelajaran

Setelah mempelajari bab ini, kalian mampu (a) menggunakan alat bantu untuk menulis, menjalankan, dan mengembangkan program analisis data; (b) memahami pengkoleksian data melalui berbagai cara, khususnya secara otomatis melalui perangkat; (c) memahami transformasi data; (d) melakukan interpretasi data dan memahami aspek privasi dan keamanan data.

Pertanyaan Pemantik

Bagaimana cara untuk mengumpulkan banyak data secara otomatis dan menampilkannya agar kita mudah memahami data tersebut?

Peta Konsep



Gambar 6.1 Peta Konsep Analisis Data

Apersepsi

Apakah kalian pernah memakai mesin pencari seperti Google, Bing, Yahoo atau lainnya untuk mencari data? Dengan mengetikkan satu atau lebih kata kunci, semua halaman web yang mengandung apa yang kalian ketikkan tersebut muncul. Bagaimana mesin pencari melakukan itu?

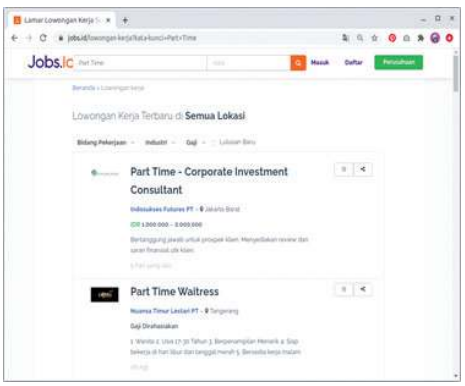
Kata Kunci

Analisis Data, Google Colaboratory, Koleksi Data, Web Scraping, Visualisasi Data

Pengantar Analisis Data

Pernahkah kalian mengalami saat sedang mencari informasi menemukan bahwa informasinya bertebaran di beberapa, bahkan di banyak halaman web? Mungkin pada saat itu kalian melakukan pencarian dengan mesin pencari dan mendapatkan hasil berupa daftar halaman yang memuat informasi yang kalian cari. Contohnya misalnya kalian ingin membanding-bandingkan harga barang dari beberapa situs toko *online*. Kalian harus membuka halamannya satu per satu, kemudian membandingkan, dan memutuskan akan membeli barang sesuai dengan kriteria yang kalian tetapkan. Jika membuka semua halaman web tersebut harus dilakukan manual, akan sangat melelahkan dan membosankan bukan? Bayangkan jika ada robot yang dapat melakukannya untuk kalian, sehingga mudah untuk menganasis. Tentunya pengambilan keputusan akan menjadi lebih cepat. Robot itu tidak harus berwujud seperti manusia *lho*. Robot dapat berupa sebuah perangkat lunak komputer. Nah, pada unit ini kalian akan membuat sebuah robot seperti ini yang akan membantu kalian.

Setelah lulus kuliah, apa yang kalian lakukan jika ingin mencari pekerjaan? Atau mungkin selama kuliah juga ingin mencari pekerjaan sambilan (*part time*)? Untuk mencari lowongan pekerjaan, kita menggunakan koran atau pun halaman web, misalnya JobsID. Coba ketik https://www.jobs.id/ di *browser*. Di halaman ini, kalian dapat mencari pekerjaan dengan memasukkan kata kunci pekerjaan di *form* pencarian yang ada, misalnya pekerjaan sambilan atau "*part time*", seperti diperlihatkan pada Gambar 6.2.



Gambar 6.2 Contoh Tampilan Halaman Website Pencarian Kerja

Sumber: Dokumen Kemendikbud, 2021

Dari halaman tersebut kita peroleh daftar pekerjaan yang ditampilkan cukup detail dalam beberapa halaman. Dengan tampilan tersebut, kita perlu melakukan *scroll* sekitar delapan kali untuk mendapatkan sekitar 23 lowongan pekerjaan. Bagaimana caranya kita dapat merangkum lowongan pekerjaan tersebut sehingga lebih mudah untuk melihat dan memilih pekerjaan yang kita inginkan?

Dalam kajian analisis data, terdapat teknik yang disebut sebagai *scraping*. *Scraping* adalah salah satu bentuk penyalinan, di mana data tertentu dikumpulkan dan disalin dari sebuah halaman web, bisa ke dalam basis data, *spreadsheet* atau tampilan tertentu untuk pengambilan atau analisis data. *Scraping* bisa dilakukan dengan menggunakan sebuah bahasa pemrograman yang mendukung. Dengan

scraping, kita bisa mendapatkan rangkuman dari suatu halaman web sehingga 23 lowongan yang sebelumnya ditampilkan dalam beberapa *scroll* kini menjadi diringkas dalam satu tampilan dengan data penting yang kita inginkan saja. Dalam aktivitas di unit pembelajaran ini, kita akan melakukan scraping dengan membuat program yang mengambil data dari sebuah halaman *website.*

Untuk membangun sebuah *scraper*, kita perlu menentukan bahasa pemrograman yang memudahkan kita. Setiap bahasa pemrograman akan diimplementasi oleh sebuah lingkungan pengembangan terintegrasi (*Integrated Development Environment*/IDE) tertentu. Ada banyak pilihan bahasa pemrograman beserta IDE-nya. Disini kita akan menggunakan bahasa pemrograman Python. Bahasa Python dipilih karena menjadi bahasa yang sering digunakan dalam analisis data. Python memiliki banyak sekali fungsi dan *library (*pustaka) yang memudahkan kita untuk melakukan analisis data, salah satunya untuk melakukan *scraping* ini.

Kemudian untuk membuat program Python, kita memerlukan alat bantu IDE atau lingkungan kerja untuk menulis dan menjalankan program Python tersebut. Salah satu alat bantu yang bisa kita gunakan ini yaitu *Google Collaboratory* atau *Google Colab*. *Google Colab* dipilih karena dapat digunakan secara online untuk mengambil data dari *website* yang *online* pula. Selain itu, banyak fungsi dan pustaka Python yang sudah terpasang dalam *Google Colab* sehingga bisa langsung digunakan.

Kita akan mulai dengan pengenalan alat bantu *Google Colab* dan Python. Setelah cukup mengenal dua alat bantu analisis data ini, mulailah kita membuat proyek untuk *scraping* halaman *website* yang dipilih. Kemudian setelah itu, data hasil *scraping* ini divisualisasikan ke dalam berbagai bentuk diagram dan *chart*.

A. Pengenalan Perkakas Analisis Data

1. Google Colaboratory (Google Colab)

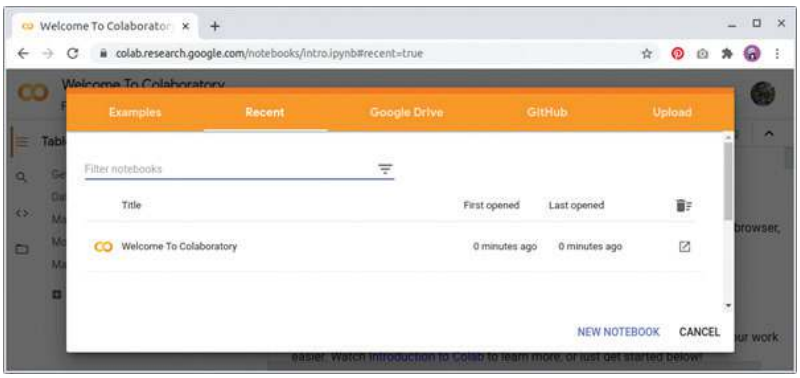
Aktivitas AD-K10-01-P: Mengenal Google Colab

Aktivitas ini dilakukan untuk memulai mengenal alat/lingkungan analisis data, salah satunya ialah Google Colab. Google Colab atau Google Colaboratory adalah salah satu lingkungan pengembangan aplikasi terintegrasi yang disediakan oleh Google secara *online* (*Online* IDE). Karena sifatnya yang *online*, maka pengguna tidak perlu melakukan instalasi dan

dapat langsung menggunakan Google Colab untuk menulis program dan melakukan pengolahan data dari Internet. Selain itu, Google Colab juga memiliki banyak fungsi serta *library* yang dapat digunakan untuk membantu pengolahan data, termasuk untuk melakukan *scraping*.

Sebelum membuat program, berikut beberapa langkah mengenal penggunaan Google Colab.

1. Buka Google Colaboratory melalui link https://colab.research.google.com/ Jika diminta untuk *Sign-In*, silakan masuk dengan menggunakan akun Google/GMail.
2. Buat catatan baru melalui pilihan NEW NOTEBOOK seperti diperlihatkan pada Gambar 6.3. *Notebook* adalah penamaan untuk *file* kerja di dalam Google Colab, tempat membuat berbagai macam dokumen, termasuk teks dokumen/catatan dan teks kode program Python.

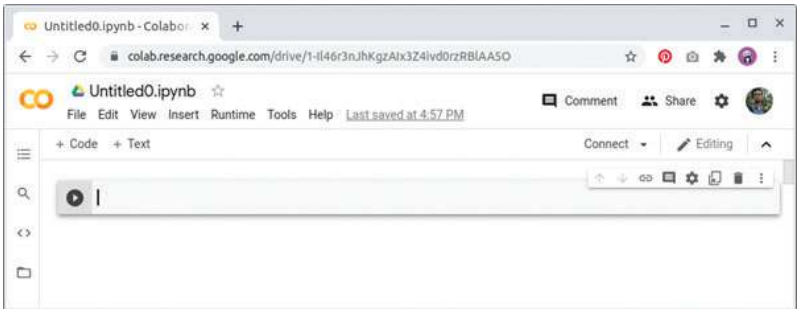


Gambar 6.3 Tampilan Google Colab Saat Dibuka Pertama Kali
Sumber: Dokumen Kemendikbud, 2021

Perhatikan bahwa Google Colab memiliki beberapa bagian area kerja yang hampir serupa dengan berbagai IDE yang lain. Beberapa objek yang ada di Google Colab diperlihatkan pada Gambar 6.3, yaitu seperti berikut.

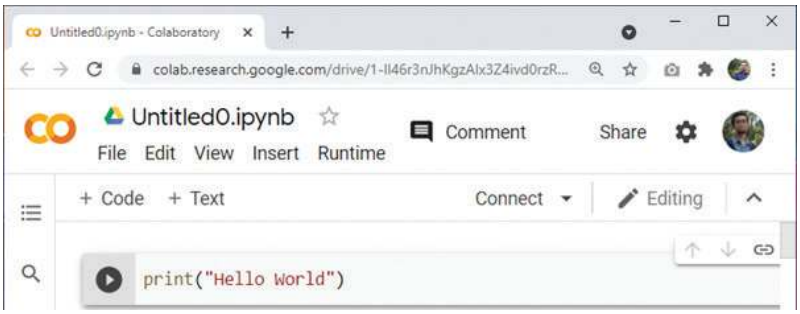
a. Nama *Notebook*, yang merupakan nama *file* berekstensi ipynb (Ipython Notebook). Kita bisa mengganti nama *notebook* dengan mengklik nama notebook tersebut.
b. *Star*, yang menandakan tingkat kepentingan file ini.
c. *Header* Kanan, yang berisi pilihan komentar, pilihan untuk berbagi (*share*), pengaturan (*setting*) file, serta pengaturan akun.
d. Menubar, yang berisi pilihan menu seperti *File, Edit, View* dan lain sebagainya.
e. Panel Kiri, yang berisi beberapa ikon menu seperti Daftar Isi, Pencarian, Penyisipan Kode, Pengelolaan File, dan Pemilihan Perintah.
f. *Toolbar* Atas, yang berisi pilihan ikon untuk Penambahan Kode atau Teks, Pilihan Koneksi, dan Pilihan Menutup Menubar.

g. Konten *Notebook*, yang berisi tulisan kode program atau teks yang kita tulis.
h. *Cell Toolbar,* yang berisi pilihan ikon untuk mengatur sel pada konten Pemindahan Atas atau bawah, Koneksi Antarsel, Penambahan komentar, Pengaturan Editor, Penggandaan, Penghapusan Sel dan lain sebagainya.



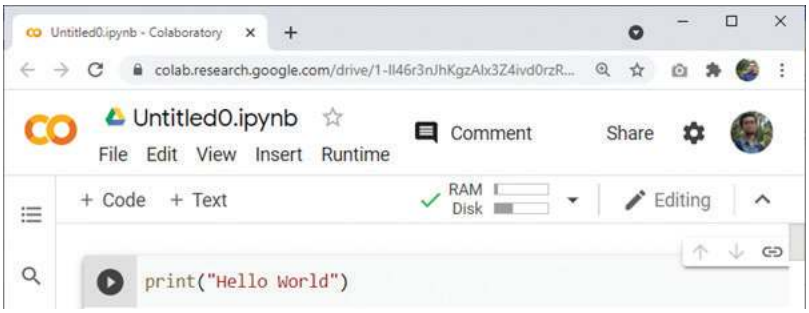
Gambar 6.4 Area Kerja Google Colab
Sumber: Dokumen Kemendikbud, 2021

3. Mulailah untuk membuat kode program Python dengan mencetak sebuah teks seperti contoh pada Gambar 6.5. Ketik satu baris kode tersebut di area Konten Notebook.



Gambar 6.5 Contoh sederhana kode program Python
Sumber: Dokumen Kemendikbud, 2021

4. Untuk menguji hasil dari kode program tersebut, tekan tombol "Run cell" (bulat dan segitiga) yang ada di pojok kiri area Konten Notebook. Jika berhasil, layar *output* akan menghasilkan teks sesuai yang diharapkan di bawah Konten Notebook seperti pada Gambar 6.6. Setelah berhasil, kita siap untuk membuat program Python menggunakan Google Colab ini.



Gambar 6.6 Hasil Keluaran Contoh Sederhana Kode Program Python
Sumber: Dokumen Kemendikbud, 2021

2. Python

Bahasa Pemrograman Python

Aktivitas AD-K10-02-P: Mengenal Python

Aktivitas pemanasan ini dilakukan untuk memulai mengenal sebuah bahasa pemrograman yang menyediakan *library* untuk analisis data, yaitu Python.

Untuk membantu berkomunikasi dengan komputer, kita perlu sebuah bahasa yang dipahami baik oleh manusia maupun komputer, dalam hal ini bahasa Pemrograman. Secara umum, bahasa pemrograman adalah bahasa yang digunakan untuk membuat program yang akan memberikan perintah kepada komputer untuk melakukan sesuatu. Ada berbagai jenis bahasa pemrograman yang dapat digunakan untuk membuat program. Salah satu bahasa pemrograman yang sering digunakan untuk melakukan analisis data karena menyediakan *library* untuk berbagai tahap proses analisis data, ialah bahasa Python. Python merupakan bahasa pemrograman yang cukup populer, seperti halnya bahasa C yang digunakan dalam unit pembelajaran Algoritma dan Pemrograman. Pada bagian ini, akan dibahas beberapa dasar pemrograman Python, terutama yang terkait dengan analisis data. Perhatikan bahwa pemrograman di materi analisis data ini hanya digunakan sebagai alat bantu. Algoritma dan pemrograman juga akan sedikit disinggung dalam unit pembelajaran ini sebagai bentuk latihan dan eksplorasi, memahami eksekusi yang terjadi khususnya dalam setiap proses analisis data. Jadi, kalian tidak perlu terlalu bingung memikirkan aturan bahasa pemrograman Python yang diberikan.

a. Cetak Data (*print*)

Perintah yang paling sederhana dalam bahasa pemrograman ialah perintah untuk mencetak suatu nilai atau data (*output*). Seperti halnya pada Gambar 6.6 yang sudah disajikan sebelumnya, mencetak data dalam Python dapat

dilakukan dengan menggunakan perintah `print`. Kalian dapat mengubah parameter atau isi di dalam tanda kurung `print` dengan data yang lain, misalnya menggantinya dengan "Selamat Datang". Setelah kode program dieksekusi atau dengan menekan tombol *Run*, program akan menampilkan data seperti yang diharapkan.

Latihan 1

1. Gantilah data dalam *print* dengan `print("2 + 3")`, kemudian jalankan (*Run*). Apa hasilnya?
2. Kemudian, ganti kembali data dalam `print` dengan `print(2 + 3)`. Apa hasilnya?
3. Ganti kembali data dalam *print* dengan `print("2" + "3")`. Apa hasilnya?
4. Apa makna dari penggunaan tanda petik tersebut?

Tanda petik digunakan untuk mencetak data *string*, yang berupa nilai alfanumerik dan tanda baca, yang tidak memiliki nilai aritmatika. Artinya, *string* "2" ditambah (atau lebih tepatnya dijejerkan) dengan *string* "3", hasilnya *string* "23". Tidak masuk akal string "2" dikurangi atau dikalikan *string* "3", sedangkan jika bilangan 2 dikurangi atau dikalikan 3, pasti ada nilai bilangannya. Sampai di sini, apakah kalian paham? Silakan, diskusikan dengan teman dan guru kalian.

Pemrograman Python di sini serupa dengan pemrograman C di unit pembelajaran Algoritma Pemrograman. Di program-program selanjutnya, kalian akan belajar Python dengan pendekatan prosedural seperti halnya belajar C. Untuk melihat hubungannya, kalian bisa melihat kembali tabel perbandingan bahasa Python dan bahasa C di unit pembelajaran Algoritma Pemrograman.

b. Pemberian Nilai Data (Assignment)

Selain mencetak langsung dari data yang ada di dalam tanda kurung, data lain yang pengisiannya di luar tanda kurung juga dapat dicetak. Pengisian atau pemberian nilai data ini disebut *assignment. Assignment* dilakukan dengan menyediakan sebuah nama (variabel) yang kemudian diisi dengan suatu nilai data menggunakan tanda sama dengan ("="). Perhatikan baris perintah pertama pada Gambar 6.7. Dalam baris tersebut, variabel bil1 diisi dengan nilai 10. Sebelum ke baris keempat, perhatikan catatan berikut.

```
bil1 = 10
bil2 = 5
jumlah = bil1 + bil2
jumlah
```

Gambar 6.7 Contoh Assignment dalam Operasi Penjumlahan Sederhana
Sumber: Dokumen Kemendikbud, 2021

Latihan 2

1. Jika diperhatikan, ada berapa nama variabel yang digunakan dalam Gambar 6.6?
2. Apa makna dari baris ke-2 dan ke-3 dalam Gambar 6.6?

Setelah dilakukan penjumlahan dan data nilai hasilnya disimpan ke dalam variabel jumlah, data ini kemudian dicetak pada baris 4. Kita dapat melihat hasil cetak atau keluaran (output) dari variabel jumlah setelah program ditulis lengkap kemudian di-*Run*. Perhatikan bahwa mencetak dapat dilakukan cukup dengan menuliskan nama variabel yang akan dicetak tanpa menggunakan perintah *print*, khususnya jika berada di baris terakhir kode program.

Latihan 3

Sebelum perintah menjumlahkan (baris 3), tambahkan baris perintah untuk mencetak bil2 tanpa menggunakan perintah *print*.

```
bil1 = 10
bil2 = 5
bil2
jumlah = bil1 + bil2
jumlah
```

1. Dapatkah `bil2` tercetak di hasil eksekusi?
2. Kemudian, sekarang tambahkan perintah *print* pada baris untuk mencetak `bil2` dan `jumlah`.

```
bil1 = 10
bil2 = 5
print(bil2)
jumlah = bil1 + bil2
print(jumlah)
```

Dapatkah kedua variabel tersebut tercetak?

c. Banyak Nilai untuk Satu Variabel (*Array*)

Latihan 4

1. Ketikkan kode program berikut, kemudian lihat hasil eksekusinya.

```
bil1 = 10
print(bil1)
bil1 = 5
print(bil1)
```

2. Apakah *output* dari variabel `bil1` pada baris keempat? Nilai 5 saja? Ke manakah nilai 10-nya?

Perhatikan bahwa pada variabel `bil1`, `bil2`, dan `jumlah`, kita menyimpan data dengan satu nilai saja untuk setiap variabel. Jika kita mengisi kembali variabel dengan nilai yang lain, nilai yang lama akan hilang atau ditimpa dengan nilai baru. Namun demikian, kita masih bisa menyimpan lebih dari satu nilai dalam satu variabel saja, yang disebut variabel array. Dalam kehidupan sehari-hari, kita akan menemui data yang memiliki banyak nilai, termasuk daftar nilai mata pelajaran Informatika sebuah kelas, misalnya. Menuliskan sebuah *array* dapat dilakukan dengan menggunakan tanda kurung siku seperti diperlihatkan pada Gambar 6.8 baris pertama.

```
data = [10, 9, 7, 8, 10, 8]
print(data[1])
data[2] = 10
print(data)
data.append(9)
print(data)
```

Gambar 6.8 Hasil Keluaran Contoh Sederhana Kode Program Python
Sumber: Dokumen Kemendikbud, 2021

Karena satu variabel menyimpan banyak nilai, kita dapat menyebutkan nilai-nilai tersebut sebagai nilai pertama (atau elemen indeks pertama), elemen kedua, elemen ketiga, dan seterusnya. Dalam bahasa pemrograman, mengakses nilai ke-n dari suatu array dapat dilakukan dengan menggunakan kurung siku persis setelah nama *variabel array* tersebut, misalnya pada baris kedua Gambar 6.8.

Latihan 5

1. Ketikkan baris perintah **pertama** dan **kedua** saja pada kode program Gambar 6.7 (tanpa kode baris ketiga dan seterusnya), kemudian jalankan programnya. Apakah *output*-nya bilangan 9?
2. Padahal, kita ingin mencetak data ke-1 pada baris kedua tersebut. Mengapa tidak muncul bilangan 10?

Indeks array dalam bahasa Python atau di hampir semua bahasa pemrograman dimulai dengan perhitungan nol, disebut indeks ke-0. Bilangan 9 seperti yang tercetak dari Gambar 6.7 ialah elemen dengan indeks ke-1 atau indeks pertama.

d. Penelusuran Data/Pengulangan (*Loop: for*)

Perhatikan bahwa dalam Gambar 6.8 baris pertama, kita hanya mencetak satu elemen sebuah data array pada indeks pertama. Pada baris ketiga dan kelima, kita mencetak banyak elemen, tetapi sebagai satu data utuh. Bagaimana jika kita ingin mencetak array sebagai elemen-elemen terpisah? Misalnya, kita ingin mencetak setiap elemen dengan memberikan keterangan indeksnya. Kita dapat menggunakan konsep Perulangan atau *Loop* dengan menggunakan perintah **for**, seperti ditunjukkan Gambar 6.9. Perintah **for** digunakan dengan menyertakan sebuah variabel baru untuk menelusuri setiap elemen di dalam variabel yang diberikan (setelah perintah in). Perhatikan baris ketiga Gambar 6.9.

```
data = [10, 9, 7, 8, 10, 8]
indeks = 0
for elemen in data:
  print("Elemen ke ", indeks, " = ", elemen)
  indeks = indeks + 1
```

Gambar 6.9 Contoh Penggunaan for dalam Python
Sumber: Dokumen Kemendikbud, 2021

Karena for merupakan perintah blok (mengandung beberapa perintah lain), perlu diakhiri dengan tanda titik dua. Perintah-perintah yang dimaksudkan untuk ikut berulang mengikuti for ini dibuat menjorok ke kanan misalnya 1 tab. Perhatikan penulisan baris keempat dan kelima Gambar 6.9.

Latihan 6

1. Perhatikan baris kelima Gambar 6.9. Mengapa nomor indeks yang dicetak di baris sebelumnya, perlu ditambah 1 setiap perulangan?
2. Bagaimana jika diganti, indeks ditambah dengan 2? Apa keluarannya?
3. Bagaimana jika dipindah, penambahan indeks dilakukan di baris keempat? Apa *output*-nya?

Kita dapat menggunakan perintah lain untuk mengulang, seperti perintah **while**. Kita dapat bereksplorasi terkait penggunaan **while** ini.

Latihan 7

Cari tahu penggunaan **while** dalam bahasa Python!

e. Uji Kondisi (*Branch: try*)

Perhatikan bahwa kode program pada Gambar 6.9, *array* diisi dengan satu jenis data, yaitu data bilangan. Namun demikian, sebuah *array* bisa diisi dengan data yang bervariasi, misal data bilangan dan data tekstual (karakter dengan tanda petik). Namun, bagaimana pengolahan data yang bervariasi seperti ini?

Latihan 8

Ketik kode program pada Gambar 6.10! Kode program tersebut mencetak semua data di dalam *array* yang kemudian dibagi dua. Namun demikian, pengolahan seperti ini menghasilkan *error*.
Error apa yang muncul? Apa maknanya?

```
data = [10, 9, "Cindi", 4, "8"]
for x in data:
  print(x / 2)

5.0
4.5
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent
<ipython-input-5-39c5096a6524> in <module>()
      1 data = [10, 9, "Cindi", 4, "8"]
      2 for x in data:
----> 3   print(x / 2)

TypeError: unsupported operand type(s) for /: 'str' and 'int'
```

Gambar 6.10 Contoh Pengolahan Data yang Bervariasi
Sumber: Dokumen Kemendikbud, 2021

Perhatikan bahwa data yang dicetak dengan dibagi dua terlebih dahulu ini memiliki variasi data seperti data bilangan dan data tekstual. Padahal, pembagian hanya bisa dilakukan pada bilangan dan oleh bilangan. Maka, akan terjadi kesalahan jika pembagian dilakukan pada data bilangan dan data tekstual. Perhatikan pesan kesalahannya berbunyi "*Line 3. Unsupported operand types* **for** /", yang dibagi harus berupa bilangan dengan bilangan, bukan string/teks.

Lalu, bagaimana jika kondisi data yang akan diolah bervariasi seperti di atas? Ada kalanya, *user* atau pengguna program memasukkan data yang benar. Namun ada kalanya, mereka melakukan kesalahan saat memasukkan data. Oleh karena itu, program yang dibuat harus memiliki mekanisme untuk menangani variasi data tersebut. Bagaimana penanganannya? Misalnya, jika data yang ada berupa bilangan, data langsung dibagi. Namun, jika data tidak bisa dibagi, berikan pesan bahwa data bukan merupakan angka/bilangan. Penanganan ini dalam konsep pemrograman disebut konsep percabangan atau branch. Salah satu perintah dalam Python yang dapat digunakan dalam percabangan ialah pasangan perintah **try** dan **except** seperti diperlihatkan pada Gambar 6.11.

```
data = [10, 9, "Cindi", 4, "8"]
for x in data:
  try:
    print(x / 2)
  except:
    print("Bukan bilangan")
```

Gambar 6.11 Contoh Penggunaan try-except dalam Python
Sumber: Dokumen Kemendikbud, 2021

Secara umum, penggunaan blok **try-except** dapat dijelaskan dengan terjemahannya. Kita mencoba (trying) untuk mengeksekusi perintah-perintah (baris setelah **try).** Jika terjadi kesalahan, eksekusi akan dilempar/lompat ke baris except dan mengeksekusi baris di dalamnya. Jika tidak terjadi kesalahan hingga **akhir** perintah dalam **try**, akan keluar dari blok perintah **try-except** (dalam hal ini kembali ke perintah **for** untuk perulangan berikutnya). Dari mekanisme ini, muncul proses percabangan di mana ada pemrosesan data yang membagi bilangan dan mencetak hasil pembagiannya, dan ada pemrosesan data yang tidak mencetak bilangan, tetapi mencetak pesan "Bukan bilangan".

Latihan 9

Ketik kode program pada Gambar 6.11! Berapa bilangan yang tercetak?

Blok perintah **try-except** merupakan salah satu perintah yang digunakan untuk kasus percabangan untuk menguji suatu kondisi, dalam hal ini kasus kesalahan atau tidak. Ada perintah lain yang dapat digunakan untuk menangani kasus percabangan (*branch*), seperti perintah **if** dan blok perintah **if-else.**

Latihan 10

Cari tahu penggunaan branch if atau **if-else** dalam bahasa Python!

f. Pustaka Kode (*Library import*)

Saat memerlukan informasi yang mungkin belum pernah dipelajari dan tidak ditemukan di buku mata pelajaran yang kalian miliki, kalian akan pergi ke perpustakaan yang menyimpan banyak sekali buku dan informasi yang dapat kita cari. Seperti halnya perpustakaan sekolah, fitur *library* atau pustaka di sebuah pemrograman memungkinkan kita untuk mendapatkan fungsionalitas yang tidak ada di program kita. Fungsionalitas ini disediakan oleh penyedia bahasa pemrograman atau komunitas tertentu yang memang sering menggunakan fungsionalitas tersebut. *Library* atau pustaka adalah koleksi program dan paket yang tersedia untuk berbagai penggunaan.

Sebagai contoh, terdapat *library* **Pandas.** Kalian dapat mencoba untuk mengetikkan kode program seperti pada Gambar 6.12 sehingga menampilkan keluaran seperti pada Gambar 6.13. Perhatikan bahwa sebelum *library* **Pandas** digunakan di Baris 6 (**pandas.DataFrame**), kita perlu **import** terlebih dahulu *library* tersebut di Baris 1. *Library* **Pandas** ini dapat kalian pelajari lebih detil dari halaman webnya di https://pandas.pydata.org/.

```
import pandas

data = [10, 9, 7, 8, 10]
siswa = ["Andi", "Budi", "Cindi", "Dedy", "Edi"]

nilai = pandas.DataFrame({
    "Nama": siswa,
    "Nilai": data
})
nilai
```

Gambar 6.12 Contoh sederhana penggunaan import pustaka Pandas
Sumber: Dokumen Kemendikbud, 2021

	Nama	Nilai
0	Andi	10
1	Budi	9
2	Cindi	7
3	Dedy	8
4	Edi	10

Gambar 6.13 Hasil keluaran contoh sederhana penggunaan import pustaka Pandas
Sumber: Dokumen Kemendikbud, 2021

B. Koleksi Data

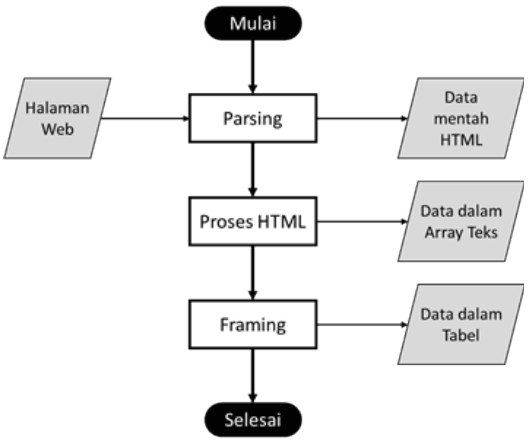
Web Scraping

Aktivitas AD-K10-03-P: Proyek *Web Scraping*

Aktivitas inti dilakukan untuk mengenal proses analisis data, khususnya dalam hal mengoleksi data dari situs web, yang dikenal dengan istilah web *scraping*.

Gambar 6.14 menunjukkan beberapa langkah yang bisa diikuti untuk melakukan *scraping* dalam bahasa Python menggunakan editor *online* Google Colab. Setelah editor siap digunakan, *scraping* bisa dimulai dengan proses parsing. Parsing adalah mengambil kode program dari sebuah halaman website secara utuh yang masih dalam bentuk kode HTML. Selanjutnya, kode HTML tersebut diproses setiap elemennya untuk mendapatkan data yang penting yang akan dirangkum. Hasilnya berupa kumpulan data yang diperlukan saja (yang diambil dari data mentah HTML yang utuh sebelumnya). Hasil keluaran sebelumnya masih berupa daftar atau *array* teks data pekerjaan yang

mungkin masih sulit dibaca. Data tersebut perlu ditampilkan secara lebih tertata sehingga mudah dibaca. Proses membingkai data atau *framing* ini bisa dilakukan dengan mudah jika data sudah diperoleh. Salah satu tampilan yang memudahkan pembacaan daftar teks tersebut ialah dalam bentuk tabel. Dengan demikian, kita peroleh hasil dari proses *scraping* ini dalam bentuk tabel data.



Gambar 6.14 Alur Proses Web Scraping
Sumber: Dokumen Kemendikbud, 2021

Mari, ikuti pembuatan proyek *scraping* tersebut dengan mengikuti langkah-langkah berikut.

1. Buka Google Colaboratory melalui link https://colab.research.google.com/. Kemudian, buka catatan baru melalui menu File > New notebook.
2. Parsing salah satu alamat website lowongan pekerjaan. Gambar 6.15 menunjukkan kode untuk melakukan parsing alamat website Lowongan Pekerjaan yang digunakan sebelumnya. Ketik beberapa baris kode program berikut di layar Google Colab kalian.

```
import requests
import pandas as pd
from bs4 import BeautifulSoup

th = "https://www.jobs.id/lowongan-kerja?kata-kunci=part time"
halaman = requests.get(th)
hasil = BeautifulSoup(halaman.content, 'html.parser')
print(hasil)
```

Gambar 6.15 Kode Program untuk Parsing Sebuah Alamat Web
Sumber: Dokumen Kemendikbud, 2021

Jika dijalankan, kode tersebut akan menyalin kode program yang ada di alamat website yang di-*request* seperti dicontohkan pada Gambar 6.16.

Gambar 6.16 Hasil Keluaran dari Kode Program Parsing Website Lowongan Pekerjaan
Sumber: Dokumen Kemendikbud, 2021

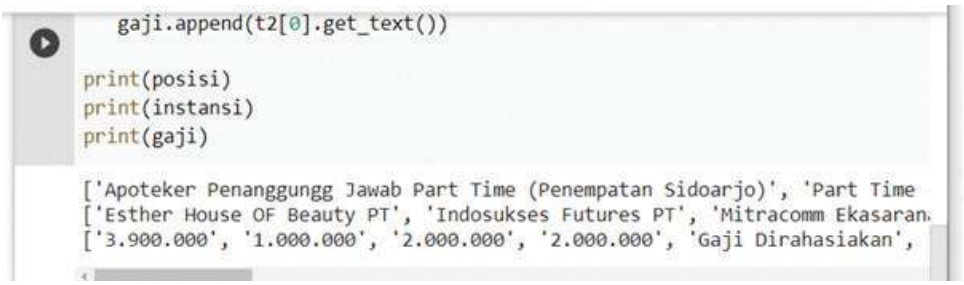
3. Olah kode HTML tersebut hingga kalian bisa mengambil data posisi pekerjaannya, instansi yang memberikan pekerjaan, serta gaji yang ditawarkan seperti diperlihatkan Gambar 6.17. Tiga data ini bisa diperoleh dari elemen kode website yang disebut *tag*. Setiap tiga elemen tersebut tersimpan di *tag* tertentu untuk kemudian diambil seperti diperlihatkan pada Gambar 6.17. Lanjutkan kode program yang ada pada Gambar 6.15 sehingga lengkap seperti pada Gambar 6.17.

```
1  import requests
2  import pandas as pd
3  from bs4 import BeautifulSoup
4
5  th = "https://www.jobs.id/lowongan-kerja?kata-kunci=part time"
6  halaman = requests.get(th)
7  hasil = BeautifulSoup(halaman.content, 'html.parser')
8  lowkers = hasil.find_all(class_="single-job-ads")
9
10 posisi = []
11 instansi = []
12 gaji = []
13
14 for p in lowkers:
15  t1 = p.select("h3")
16  t2 = t1[0].select("a")
17  posisi.append(t2[0].get_text())
18
19  t1 = p.select("p")
20  t2 = t1[0].select("a")
```

```
21   try:
22     instansi.append(t2[0].get_text())
23   except:
24     instansi.append("-")
25
26   t2 = t1[1].select("span")
27   try:
28     gaji.append(t2[1].get_text())
29   except:
30     gaji.append(t2[0].get_text())
31
32 print(posisi)
33 print(instansi)
34 print(gaji)
```

Gambar 6.17 Kode Program untuk Mengoleksi Data Lowongan Pekerjaan
Sumber: Dokumen Kemendikbud, 2021

Jika kode program tersebut dijalankan di Google Colab, keluaran yang dihasilkan berupa ringkasan data dari yang diharapkan saja, seperti diperlihatkan pada Gambar 6.18.



Gambar 6.18 Hasil Keluaran dari Kode Program Koleksi Data Pekerjaan
Sumber: Dokumen Kemendikbud, 2021

4. Terakhir, bingkai data array tersebut ke dalam tabel yang ditampilkan oleh Python seperti diperlihatkan kode pada Gambar 6.19. Ketikkan kode program tersebut menggantikan baris 32-34 yang sudah diketik pada Gambar 6.17.

```
31
32 lowker = pd.DataFrame({
33    "Posisi": posisi,
34    "Instansi": instansi,
35    "Gaji":gaji
36 })
37 lowker
```

Gambar 6.19 Kode Program untuk Membingkai Data ke dalam Tabel
Sumber: Dokumen Kemendikbud, 2021

```
    "Gaji":gaji
})
lowker
```

	Posisi	Instansi	Gaji
0	Apoteker Penanggungg Jawab Part Time (Penempat...	Esther House OF Beauty PT	3.900.000
1	Part Time - Corporate Investment Consultant	Indosukses Futures PT	1.000.000
2	Customer Care Part Time	Mitracomm Ekasarana PT	2.000.000
3	Call Center Officer Part Time	Mitracomm Ekasarana PT	2.000.000
4	CALL Center PART TIME - Jakarta	Mitracomm Ekasarana PT	Gaji Dirahasiakan
5	Deputy Division Head (Production Assistant Man...	Indo Tirta Abadi PT	Gaji Dirahasiakan

Gambar 6.20 Contoh Hasil Keluaran dari Kode Program yang Menampilkan Tabel

Sumber: Dokumen Kemendikbud, 2021

Jika kode program tersebut dijalankan di Google Colab, keluaran yang dihasilkan berupa tabel dari data yang telah dikoleksi, seperti diperlihatkan pada Gambar 6.20.

Gambar 6.20. menunjukkan hasil akhir proses *scraping* data dari website Jobs ID untuk mengoleksi data lowongan pekerjaan yang meringkasnya ke dalam sebuah tabel. Bagaimana kalian membaca lowongan pekerjaan dari Gambar 6.1 sebagai halaman web asli dan Gambar 6.20 sebagai hasil *scraping*? Dengan penyajian seperti pada Gambar 6.20, data yang diperoleh akan lebih mudah dipahami dan kemudian diolah/dianalisis. Perhatikan bahwa tampilan tersebut merupakan data saat dibuat. Bisa jadi, keluaran tidak sama persis, bahkan mungkin juga terjadi *error.*

Langkah-langkah *scraping* menggunakan bahasa Python seperti ini dapat dipelajari secara lebih detail dari tutorial yang ada di internet seperti misalnya di https://www.dataquest.io/blog/web-scraping-tutorial-python/. *Scraping* juga dapat dilakukan menggunakan bahasa lain (PHP atau R-Language) dan editor (*Integrated Development Environment*) lain.

C. Visualisasi Data

Aktivitas AD-K10-04-P: Proyek Visualisasi Data

Aktivitas lanjutan ini dilakukan untuk mengenal visualisasi data.

Pada proyek sebelumnya, data website diambil (*scraping*) dan ditampilkan dalam bentuk tabel. Tabel adalah salah satu bentuk analisis data dasar. Kita bisa melihat persebaran data secara baris per baris dalam bentuk tekstual. Analisis data dapat dilanjutkan dengan mengubah data tekstual tersebut menjadi data visual sehingga lebih mudah untuk dipahami. Data divisualkan

dalam berbagai diagram seperti diagram batang, diagram lingkaran, diagram garis, dan lain sebagainya. Berikut ini data pada proyek sebelumnya akan disajikan dalam diagram batang.

Sebelum data bisa diolah dan disajikan menjadi diagram, data tersebut perlu dipersiapkan terlebih dahulu. Dalam analisis data, proses persiapan ini disebut pra-pemrosesan data (*data preprocessing*). Setelah dipersiapkan, baru kemudian data bisa diolah dan divisualisasikan.

1. Pra-pemrosesan Data

Pra-pemrosesan data dilakukan agar data siap untuk diolah. Data perlu dipersiapkan karena bisa jadi yang kita peroleh masih mentah, banyak terdapat kesalahan sehingga tidak bisa dihitung untuk dibuat visualisasinya. Sebagai contoh, pada kasus sebelumnya, data yang diolah, yaitu data gaji, masih ada yang berupa data teks seperti "Dirahasiakan". Gaji juga masih mengandung titik yang mengelompokkan tiga angka. Padahal, dalam bilangan (integer), titik bermakna pecahan. Oleh karena itu, kita perlu mengubah data gaji yang masih bertipe teks ini ke dalam tipe bilangan seperti diperlihatkan pada Gambar 6.21. Baris 27-34 berikut perlu menggantikan kode program Baris 27-30 pada Gambar 6.17.

```
26   t2 = t1[1].select("span")
27   try:
28    xgaji = t2[1].get_text()
29   except:
30    xgaji = t2[0].get_text()
31   xgaji = xgaji.replace(".","")
32   if (xgaji=="Gaji Dirahasiakan"):
33    xgaji = 0
34   gaji.append(xgaji);
```

Gambar 6.21 Kode Program untuk Preprocessing dalam Menata Data Gaji

Sumber: Dokumen Kemendikbud, 2021

2. Visualisasi Data: Barchart

Untuk membuat tampilan grafik atau diagram, kita dapat menggunakan *library* Python, yaitu Plotly. *Library* ini perlu ditambahkan terlebih dahulu di kode program. Cara penambahannya menggunakan **import** seperti yang sudah dipelajari sebelumnya, seperti berikut.

```
import plotly.express as px
```

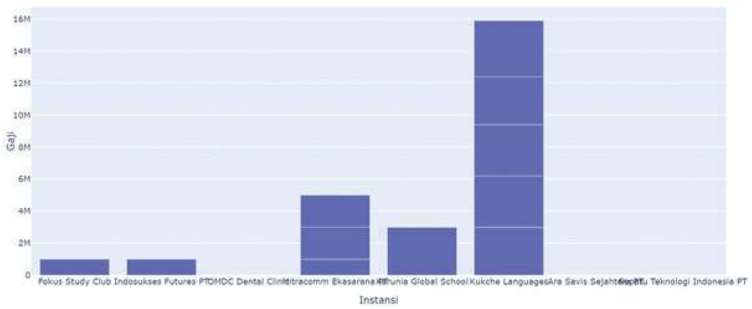
Selanjutnya, setelah *library*-nya di-import, kita dapat menggunakan fungsionalitas pada *library* Plotly untuk membuat diagram. Sebagai contoh, kita dapat membuat diagram batang (barchart) menggunakan fungsi bar() seperti diperlihatkan pada Gambar 6.22.

```
39
40  lowker = pd.DataFrame({
41    "Posisi": posisi,
42    "Instansi": instansi,
43    "Gaji":gaji
44  })
45
46  fig = px.bar(lowker, x='Instansi', y='Gaji')
47  fig.show()
```

Gambar 6.22 Kode Program untuk Memvisualisasi Data dalam Diagram Batang

Sumber: Dokumen Kemendikbud, 2021

Dari kode program tersebut, kita akan menghasilkan sebuah diagram batang seperti yang dicontohkan pada Gambar 6.23.



Gambar 6.23 Contoh Hasil Keluaran dari Kode Program Visualisasi Data

Sumber: Dokumen Kemendikbud, 2021

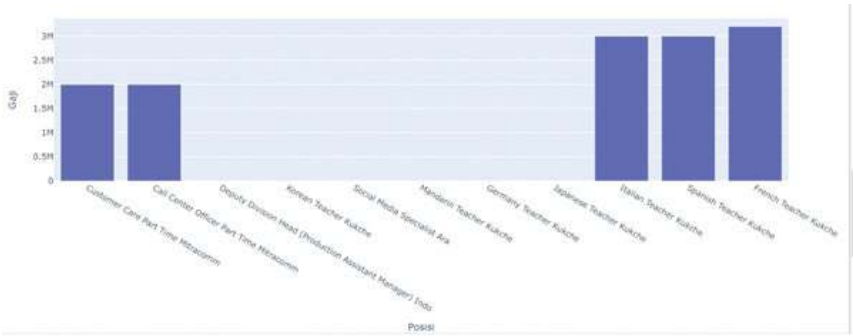
3. Prapemrosesan Data Lanjut

Jika diperhatikan pada Gambar 6.23, beberapa lowongan pekerjaan terlihat menumpuk. Beberapa posisi lowongan pada satu instansi menjadi satu batang saja. Hal ini menyebabkan nominal gaji yang ditunjukkan menjadi tidak valid. Ini terjadi karena kita mengelompokkan lowongan berdasarkan instansinya (Gambar 6.22 Baris 46). Bagaimana jika dikelompokkan berdasarkan posisinya? Tentunya, hal ini bukan menjadi solusi karena bisa jadi terdapat beberapa lowongan dengan posisi yang sama dan akan tertumpuk juga. Jadi, bagaimana solusinya?

Solusinya ialah mengelompokkan lowongan pekerjaan berdasarkan gabungan posisi dan instansi penyedia lowongan. Oleh karena itu, kita perlu mengubah kembali kode pada Gambar 6.17 sehingga posisi dan instansi tidak berada dalam variabel yang berbeda, tetapi disatukan dalam satu variabel (dalam hal ini variabel posisi saja) seperti diperlihatkan pada Gambar 6.24 Baris 18-28 yang menggantikan kode program sebelumnya. Selanjutnya, bingkai data tersebut ditampilkan cukup untuk dua item saja, yaitu posisi dan gaji seperti pada Baris 41-42. Terakhir, untuk menampilkan diagram batangnya, item yang digunakan diganti dari instansi menjadi posisi seperti pada Baris 45 Gambar 6.24 (menggantikan Baris 46 Gambar 6.22). Dengan demikian, setiap lowongan akan diperlihatkan secara terpisah seperti pada Gambar 6.25.

```
16    t1 = p.select("h3")
17    t2 = t1[0].select("a")
18    t3 = t2[0].get_text()
19
20    t1 = p.select("p")
21    t2 = t1[0].select("a")
22    try:
23      t2 = t2[0].get_text()
24    except:
25      t2 = ""
26    if (t2.find(" ")>=0):
27      t3 = t3 + " " + t2[0:t2.find(" ")]
28    posisi.append(t3)
..
40  lowker = pd.DataFrame({
41     "Posisi": posisi,
42     "Gaji":gaji
43  })
44
45  fig = px.bar(lowker, x='Posisi', y='Gaji')
46  fig.show()
```

Gambar 6.24 Kode Program untuk *Prcprocccsing* dalam Mengelompokkan Posisi dan Institusi
Sumber: Dokumen Kemendikbud, 2021



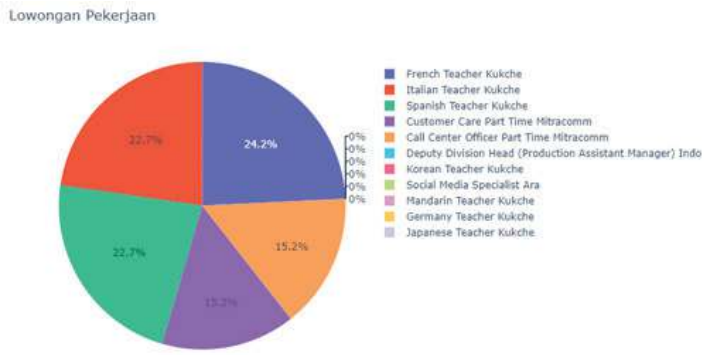
Gambar 6.25 Contoh Hasil Keluaran Setelah Posisi dan Institusi Dikelompokkan
Sumber: Dokumen Kemendikbud, 2021

4. Visualisasi Data dengan Diagram lain

Setiap diagram dan *chart* memiliki kegunaan masing-masing sesuai karakteristiknya. Diagram batang, misalnya, digunakan untuk memperlihatkan beberapa item serupa yang perlu dibandingkan. Di sisi lain, diagram lingkaran digunakan untuk memperlihatkan proporsi dari beberapa item yang bisa menjadi bagian utuh dari suatu kasus. Untuk membuat diagram lainnya, kalian dapat mempelajari penggunaanya di alamat ini https://plotly.

com/python/basic-charts/ Sebagai contoh, dapat dicoba untuk membuat diagram lingkaran (*pie chart*) seperti diperlihatkan pada Gambar 6.26. Untuk melakukannya, kode program yang sebelumnya menampilkan diagram batang (Gambar 6.24 Baris 45) diganti menjadi kode untuk menampilkan diagram lingkaran seperti berikut.

```
fig = px.pie(lowker, values='Gaji', names='Posisi', title='Lowongan')
```



Gambar 6.26 Contoh Hasil Keluaran dari Kode Program Visualisasi Data Menggunakan Diagram Lingkaran
Sumber: Dokumen Kemendikbud, 2021

Menarik ,bukan? Bagaimana dengan diagram yang lain?

Latihan 11

Untuk membuat diagram *scatter* (diagram titik) dari data lowongan pekerjaan ini, bagaimana kodenya?

Sebagai catatan, sebenarnya, diagram lingkaran dan diagram scatter kurang tepat jika digunakan dalam kasus ini. Namun demikian, kita dapat menggunakannya sebagai contoh untuk dipelajari pembuatannya karena datanya sudah tersedia. Dapatkah kalian mencari contoh kasus lain yang tepat divisualisasikan menggunakan kedua diagram tersebut?

Ayo Kita Renungkan

Pada unit pembelajaran ini, kalian telah mempelajari tentang bagaimana mengimplementasi sebuah *web scraper* dengan menggunakan *library* dari bahasa Python. Hal ini menjadi alternatif cara yang menggantikan cara sebelumnya, yaitu membuka secara manual satu per satu halaman webnya. Program web scraper ini akan melakukannya untuk kalian. Dari sini, ternyata, kalian telah melakukan aktivitas pemrograman seperti halnya yang dilakukan pada unit Algoritma Pemrograman.

Tantangan Berpikir

Jika kalian perhatikan, apa bedanya aktivitas pemrograman yang dilakukan di unit Analisis Data ini dibandingkan dengan yang dilakukan di unit Algoritma Pemrograman? Pertanyaan renungan berikutnya, apa bedanya mesin scraper yang kalian buat dan mesin pencari yang sering kalian gunakan?

Target *Scraping*

Kita telah melakukan web *scraping* untuk web yang terbuka dan dapat dilihat oleh publik. Jadi, program scraper, seperti halnya seseorang yang secara manual melakukannya, akan dapat membuka suatu halaman web karena informasi yang ditampilkan pada halaman web tersebut ialah informasi publik. Bagaimana dengan website yang tidak publik? Misalnya, kita hanya dapat menampilkan email setelah melakukan login. Informasi email kita bukan merupakan informasi publik, melainkan informasi privat. Saat kita menampilkan halaman kita di sosial media, hanya kita dan teman-teman kita yang dapat melihat. Halaman di sosial media ini tidak publik dan juga tidak privat, tetapi terbatas kepada sekumpulan tertentu. Apakah kita bisa membuat scraper untuk informasi yang tidak publik? Bolehkah kita mengambil data yang privat seperti itu? Diskusikan dengan teman-teman!

Tentang Perkakas

Kalian telah memanfaatkan *library* Python untuk melakukan *scraping*. Program *scraper* yang kalian tulis hanya untuk "membungkus" *library* tersebut agar dapat dijalankan. Menurut kalian, bagaimana program *scraper* melakukan pengambilan isi halaman web?

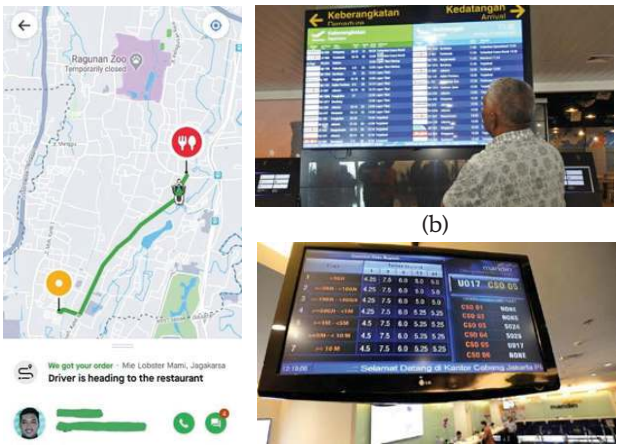
Library untuk *scraping* hanya merupakan salah satu *library* Python untuk analisis data secara keseluruhan. Python tidak hanya menyediakan *library* untuk *scraping*, tetapi juga menyediakan banyak *library* untuk keperluan lainnya. Jika kalian membutuhkan untuk keperluan lain, kalian akan dapat memanfaatkan seperti kalian memanfaatkan *library* untuk *scraping*. Eksplorasilah, *library* apa lagi yang dimiliki Python?

Sistem Visualisasi Real-Time

Saat kalian melakukan pemesanan ojek *online*, pelanggan dapat melacak posisi pengendara atau *driver* sedang di mana saat ini (Gambar 6.27.a). Pada beberapa tempat misalnya di bandara (Gambar 6.27.b), di halte bus atau di tempat umum, seringkali ada tampilan yang selalu diubah sesuai dengan keadaan saat itu. Pada sistem penerbangan di bandara, misalnya, setiap kali ada pesawat berangkat atau pergi, tampilan akan diubah. Kemudian di bank, tampilan kurs mata uang hari ini juga selalu diperbarui karena kurs tersebut sering berubah (Gambar 6.27.c). Data kurs ini diambil dari sistem komputer Bank Indonesia (Bank Sentral).

Berbagai data yang sering berubah tersebut menunjukkan bahwa yang ditampilkan ialah data pada waktu saat itu juga atau yang disebut real-time. Visualisasi data real-time ini dapat dilihat pada Gambar 6.26. Untuk kepentingan yang lebih besar, data *real-time* ini divisualisasikan dalam sistem yang disebut *dashboard*. Jika ingin belajar lebih jauh tentang sistem *dashboard*, kalian dapat mempelajari *website* https://dash-gallery.plotly.host/dash-web-trader/.

Rangkaian program komputer dibutuhkan dari berbagai sumber data. Program komputer tersebut akan berkolaborasi untuk akhirnya menampilkan data yang dilihat pengguna. Bayangkan bahwa setiap mesin/agen pengolah tersebut adalah sebuah program komputer yang berfungsi sesuai peran masing-masing, seperti manusia yang bersinergi di dunia dunia nyata. Namun, dalam hal ini, program komputer merupakan program yang aktif bekerja, bukan di dunia fisikal, tetapi di dunia digital. Dapatkah kalian membayangkan hal ini? Buatlah ilustrasi kerja sama yang dilakukan oleh para agen pengolah tersebut! Buatlah infografis dalam bentuk diagram untuk 3 sistem yang diilustrasikan pada Gambar 6.26.



Gambar 6.27 Contoh Visualisasi Data Secara *Real-Time*

Ada satu catatan menarik mengenai aplikasi ojek *online* yang bisa didiskusikan. Jika kalian pernah melakukan pemesanan ojek *online*, kalian lihat bahwa sepeda motor bergerak, tetapi tidak lancar. Mengapa ini bisa terjadi?

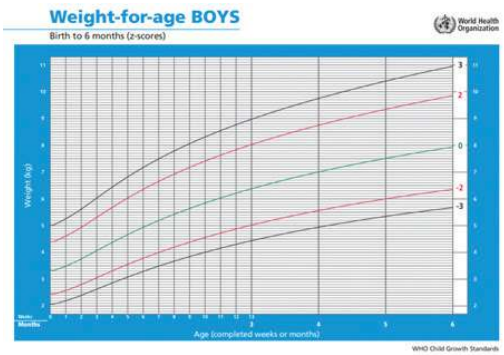
Kembali ke proyek web *scraping*, program scraper ini hanya merupakan sebagian kecil dari rangkaian analisis data. Rangkaian yang lengkap ialah mengambil, memroses, menyajikan dalam visualisasi yang memudahkan pembaca, menganalisis, menginterpretasi, dan menyimpulkan atau mengambil keputusan. Dari ilustrasi rangkaian proses analisis data tersebut, kita dapat ketahui bahwa pengambilan data dapat dilakukan dengan program, misalnya dengan program yang telah kalian buat.

Selanjutnya, dari rangkaian proses tersebut, menurut kalian, apakah semuanya dapat diotomasi dengan menulis programnya? Proses mana yang dapat digantikan oleh program komputer, dan mana yang sulit atau bahkan tidak mungkin dijadikan program komputer? Kalaupun dapat digantikan oleh program komputer, apakah memang lebih menguntungkan untuk membuat

programnya? Jelaskan alasan kalian!

Tentang Prediksi

Prediksi dari data dapat dilakukan dengan mudah jika divisualisasikan. Misalnya, dokter dapat menggunakan kurva pertumbuhan berat badan bayi untuk memprediksi perkembangan bayi dan menyatakan kesehatannya (Gambar 6.28).



Gambar 6.28 Visualisasi untuk Data Berat Badan sebagai Alat Memprediksi Pertumbuhan Ideal Bayi

Dengan pengalaman melakukan web *scraping* tentang lowongan pekerjaan, kalian dapat mencoba untuk membuat grafik mengenai tren kebutuhan *programmer*. Misalnya, kalian dapat mengambil data lowongan tahun 2019 dan tahun 2020, atau data lowongan setiap bulan di tahun 2021. Selalu menaik, bukan? Dari data tersebut, kalian dapat memperkirakan kira-kira berapa lowongan pekerjaan yang ada di bulan depan.

Uji Kompetensi

1. Jelaskan langkah koleksi data/*scraping* dengan bahasa kalian sendiri!
2. Proyek *web scraping* sebelumnya mencontohkan salah satu lowongan pekerjaan, yaitu "*part-time*". Bagian mana yang harus diubah agar scraping tersebut menampilkan lowongan pekerjaan lain, misalnya "*programmer*"?
3. Proyek *web scraping* sebelumnya menampilkan informasi posisi, instansi, dan gaji suatu lowongan pekerjaan yang ditampilkan dalam tiga kolom tabel. Dapatkan kalian menambah satu informasi lagi, yaitu Lokasi ke dalam tabel lowongan pekerjaan tersebut? Bagaimana analisis data untuk struktur HTML website lowongan pekerjaan tersebut?
4. Cari lowongan pekerjaan yang terdapat di koran/majalah (dilakukan secara manual/*unplugged*) atau *website* (dilakukan dengan bantuan komputer/*plugged*)! Rangkum seperti tabel yang dicontohkan pada Aktivitas 3! Tabel dapat ditulis tangan dalam selembar kertas (*unplugged*) atau ditulis di Aplikasi Pengolah Angka (*plugged*). Proses apa saja yang kalian lakukan jika analisis data dilakukan secara manual, bukan otomatis seperti pada aktivitas *web scraping*?

5. Cari lowongan pekerjaan dari *website* lain, misalnya JobsDB yang berada di alamat https://id.jobsdb.com/id yang memiliki struktur HTML yang sederhana dan mudah dianalisis. Dapatkah kalian melakukan *scraping* dari website tersebut untuk mendapatkan rangkuman lowongan pekerjaan yang tersedia dalam sebuah tabel? Kalian dapat menggunakan tabel berikut untuk membantu pekerjaan kalian.

No	Aktivitas	Checklist (√)	Keterangan Output	Jawaban
1.	Parsing HTML Website	☐	Berapa baris kode HTML yang diperoleh?	
2.	Proses struktur HTML	☐	Berapa *array* informasi dari setiap lowongan yang diperoleh?	
3.	*Framing* data	☐	Berapa baris lowongan pekerjaan dari tabel yang ditampilkan?	

Ingin Tahu Lebih?

Jika kalian tertarik dengan materi ini dan ingin mempelajari lebih lanjut, kalian dapat mengakses ke link berikut ini:

1. Jobs ID (2020). Info Lowongan Kerja Terbaru dan Populer 2020. Diakses dari http://jobs.id
2. Wikipedia (2020). Web Scraping. Diakses dari https://en.wikipedia.org/wiki/Web_scraping
3. Google Colaboratory (2020). Welcome to Colaboratory. Diakses dari https://colab.research.google.com/
4. Dataquest (2020). Tutorial: Web Scraping with Python using Beautiful Soup. Diakses dari https://www.dataquest.io/blog/web-scraping-tutorial-python/
5. PyData (2021). Pandas: Python Data Analysis *Library*. Diakses dari https://pandas.pydata.org/
6. Ploty (2021). Plotly Python Open Source Graphing *Library* Basic Chart. Diakses dati https://plotly.com/python/basic-charts/
7. Plotly (2021). Plotly | Dash. Diakses dari https://dash-gallery.plotly.host/dash-web-trader/