Beginning with Python: The Statistical Programming Module

Statistika dan Dasar Pemograman

Oleh Tim PkM(Pengabdian kepada Masyarakat)

Teamsar Muliadi Panggabean,S.Kom,PGCert
Ike Fitriyaningsih,S.Si,M.Si
Hernawati Susanti Samosir, S.ST., M.Kom.

Beginning with Python: The Statistical Programming Module

Prepared by:

PkM – Del Institute of Technology © 2019

This page is intentionally left blank

Daftar Isi

Chapter 1. Statistical Programming secara singkat

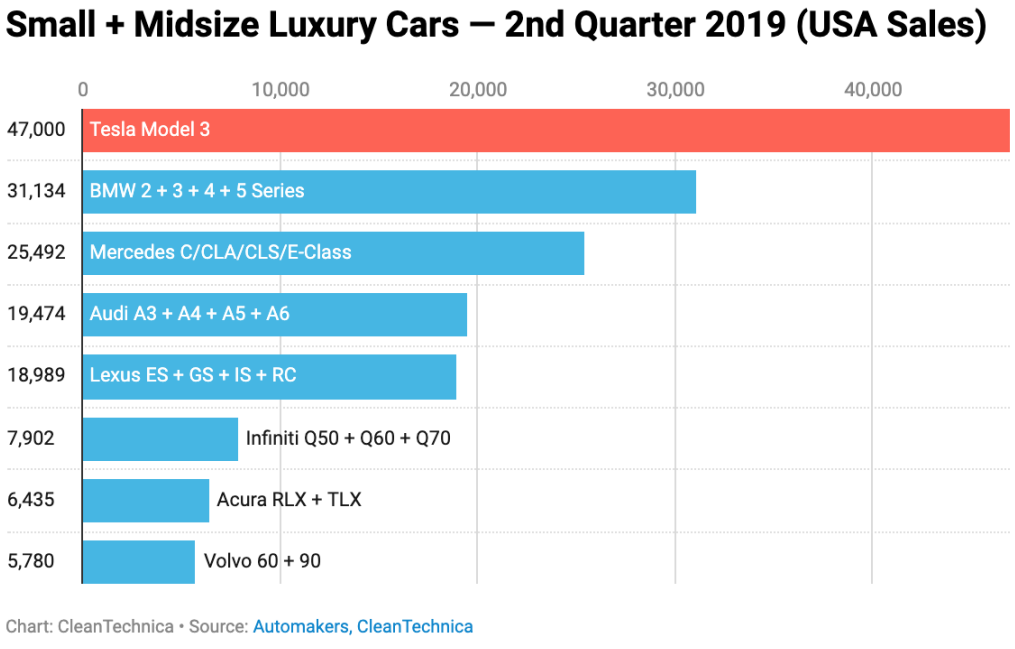
1.1 Apa yang dimaksud dengan statistical programming?

Statistical Programming mengacu kepada teknik komputasi yang dapat membantu pembaca di dalam menganalisa data. Para manager di perusahaan biasanya menggunakan teknik ini untuk mendapatkan ide atau gambaran umum terhadap kinerja perusahaanya, dan data yang digunakan cukup besar. Data tersebut dapat berupa penjualan produk selama setahun, pendapatan/pengeluaran per-triwulan, atau bahkan persaingan produk serupa dalam kurun waktu 5 tahun. Yang menjadi pertanyaanya adalah bagaimana data ini dapat memberi *insight* bagi si pembaca? Caranya cukup mudah, data terlebih dahulu diolah dengan berbagai pendekatan statistika, kemudian hasilnya akan ditampilkan dalam bentuk grafik. Perlu anda ingat bahwa kegiatan pengolahan data ini dilakukan secara otomatis dengan menggunakan bahasa pemrograman. Ada banyak bahasa pemrograman saat ini, dan yang paling umum digunakan untuk pengolahan data adalah *Python programming*.

1.2 Mengapa statistika?

Jika anda adalah seorang manager di sebuah perusahaan otomotif, dan berencana ingin meluncurkan mobil dengan model terbaru, anda harus terlebih dahulu menganalisa apakah mobil tersebut laku dipasar atau tidak. Banyak faktor yang dapat menyebabkan kegagalan memasarkan mobil tersebut, apakah design yang kurang baik, produk kompetitor lebih bagus, atau bahkan stabilitas ekonomi yang kurang bagik di suatu negara. Ketidakpastian ini menyebabkan anda tidak dapat mengambil keputusan kapan anda sebaiknya memasarkan produk terbaru.

Salah salah satu cara yang dapat anda lakukan adalah dengan menggunakan pendekatan statistik. Contohnya dengan menggunakan data penjualan 5 tahun terakhir dapat dihitung kemungkinan sukses atau tidaknya penjualan mobil dengan model terbaru.



1.3 Mengapa Python programming?

Python programming merupakan pemrograman yang berfokuskan pada kemudahaan untuk dibaca. Tidak seperti bahasa pemrograman lainnya, python membantu para programmer, khususnya programmer pemula, untuk dapat melakukan koding seminimal mungkin. Bahasa pemrograman python dikembangkan pada tahun 1991 oleh Guido Van Rossum. Pengembang bahasa pemrograman ini menciptakan bahasa python untuk kesenangan. Saat ini python banyak digunakan diberbagai organisasi. Python sendiri memiliki banyak library yang dapat membantu para pengembang di dalam membangun aplikasi.

Chapter 2. Bermain dengan Python Programming

2.1 Deklarasi *variable* secara singkat

Di setiap bahasa pemrograman, tanpa terkecuali, hal pertama yang harus anda ketahui dan mampu untuk menerapkan adalah mendeklarasikan *variable*. Apa yang dimaksud dengan *variable*? Coba bayangkan anda diminta untuk menyediakan secangkir teh untuk rekan anda. Pertama sekali anda mengambil cangkir, kemudian menuangkan beberapa air, gula dan teh ke dalamnya, dan mengaduknya. Anda dapat berpikir bahwa cangkir tersebut adalah variable, dan isi (teh) di dalam cangkir itu adalah *value*. Untuk lebih jelasnya, lihat contoh di bawah ini:

```
nama = "Budi Sanjaya Samosir"
alamat = "Jl. Sitoluama no. 12, Laguboti"
usia = 24
```

Seperti yang telah anda lihat pada contoh di atas, disana terdapat pendeklarasian tiga buah variable, yaitu: nama, alamat dan usia. Masing-masing *variable* ini di-isi dengan *value* (nilai) yang berbeda-beda. Sebagai contoh, *variable* nama di-isi dengan nilai "Budi Sanjaya Samosir", dan usia di-isi dengan sebuah angka, 24.

Ingat, ada beberapa aturan ketika mendeklarasikan sebuah *variable*:

- Pendeklarasian sebuah *variable* tidak boleh ada spasi di tengah-tengah nama *variable*. Contoh: nama saya => nama_saya.
- Pendeklarasian sebuah variable tidak boleh ada mengandung angka (terkecuali di akhir nama variable) dan special karakter di depan nama variable (terkecuali underscore "_"). Contoh: 123nama (tidak boleh), !$nama (tidak boleh), _nama (boleh), nama123 (boleh), nama!$@ (tidak boleh).

Latihan!

Anda dapat mengakses url berikut http://172.22.12.10:8888/tree/Workout, kemudian buat sebuah folder di dalamnya. Ganti nama folder yang baru saja anda buat dengan nama sekolah anda, contoh: SMAN02Balige. Di dalam folder tersebut, buat sebuah file jupyter notebook dan beri nama "W01-Variable". Anda akan menggunakan file ini sebagai antar-muka meng-code python. Catatan: anda dapat meminta pertolongan kepada instruktor disekitar, bila diperlukan. Tugas anda deklarasikan 5 buah *variable* yang dapat menyimpan informasi, seperti: nama_saya, usia_saya, no_hp_saya, alamat_sekolah, dan hobi.

2.2 Pengenalan Singkat tentang Operator Aritmatika dan Karakter Spesial

Melakukan operasi matematika dengan cepat dan tepat merupakan salah satu alasan mengapa orang berpindah dari melakukan perhitungan manual ke otomatis. Semua yang anda butuhkan sudah ada di setiap bahasa pemrograman. Di bawah ini, beberapa operator aritmatika dan dapat juga diterapkan di bahasa pemrograman python:

No	Operator	Symbol	Example
1	Addition	+	5 + 2 = 7
2	Subtraction	-	8 - 3 = 5
3	Multiplication	*	2 * 4 = 8
4	Division	/	6 / 3 = 2
5	Modulus	%	18 % 5 = 3

Selain operator aritmatika di atas, ada juga '*special operator*' yang sering digunakan untuk perbandingan, dan masing-masing operator menghasilkan keluaran **true** (untuk 'ya') atau **false** (untuk 'tidak').

Relational Operator		
1	**Operator**	**Meaning**
2	>	Is greater than
3	>=	Is greater than or equal to
4	<	Is less than
5	<=	Is less than or equal to
6	==	Is equal to
7	!=	Is **not** equal to

Logical Operator		
No	**Operator**	**Meaning**
1	&&	And
2	||	Or
3	!	Not

Beberapa aturan untuk '*logical operator*':

Operator '**&&**' (and)		
1	**Expression**	**Answer**
2	true **and** true	true
3	true **and** false	false
4	false **and** true	false
5	false **and** false	false
6	not false	true
7	not true	false

Operator '||' (or)		
1	**Expression**	**Answer**
2	true **or** true	true
3	true **or** false	true
4	false **or** true	true
5	false **or** false	false

Example		
No	**Expression**	**Answer**
1	3 > 6	false
2	8 >= (8+1 - 1)	true
3	7 < 2	false
4	(5+2) <= (8-1)	true
5	2 == (7 % 2)	false
6	(3 > 6) **and** (7 < 2)	false
7	(8 >= 2) **or** (7 < 2)	true

Latihan!

Buat sebuah *file jupyter notebook* baru dan ganti namanya dengan "W2-Arithmetic_Operator". Deklarasikan tiga buah *variable*, seperti: angka1, angka2, dan hasil. Lakukan operasi aritmatika di atas, misalnya: pengurangan, penambahan, perkalian, dsb. *Hint*: lakukan pemberian value angka ke masing-masing *variable* (terkecuali *variable* hasil), lalu lakukan operasi aritmatika. Cetak keluaran hasil dengan menggunakan fungsi **print**. Contoh: print("Hasil penjumlahan angka1 dan angka2:" + hasil).

2.3 Pengenalan Singkat Repetisi (Loop)

Repetisi atau *loop* sangat sering digunakan di setiap bahasa pemrograman. Bayangkan anda diminta untuk melakukan sesuatu hal yang sama sampai seseorang berkata "berhenti". Hal yang sama di dunia pemrograman, anda dapat melakukan repetisi sampai kode program anda menunjukan tanda berhenti. Loop (repetisi) memiliki "batas bawah" dan "batas akhir". Batas bawah menunjukan posisi anda pertama sekali dan batas akhir merupakan penanda untuk memberhentikan proses loop itu. Untuk lebih jelasnya, silahkan anda lihat kode program di bawah (anda dapat membuat *file jupyter notebook* yang baru untuk menjalankan kode di bawah ini, beri nama *jupyter notebook* anda dengan "C01-Loop"):

```
lower_bound = 0
upper_bound = 5

for num in range(lower_bound, upper_bound):
    print("Ini angka {}".format(num))
```

```
Output:
Ini angka 0
Ini angka 1
Ini angka 2
Ini angka 3
Ini angka 4
```

Seperti yang telah anda lihat, disana terdapat dua buah *variable* yang dideklarasikan dengan *value* **nol** dan **lima**. Angka nol merupakan batas bawah dan angka lima merupakan batas atas. Ini artinya perulangan akan **dimulai** dari angka **nol** dan **berakhir** pada **angka** empat **bukan lima**. Mengapa demikian? Bahasa pemrograman python "menganut" paham batas atas - 1.

2.4 Pengenalan Singkat Kondisional

Dalam kehidupan sehari-hari tanpa disadari, anda sering mengambil keputusan bila diberikan beberapa pilihan. Pengambilan keputusan ini tentunya dengan mempertimbangkan beberapa kondisi yang ada, misal: saya akan mengikuti pelatihan statistika di kampus IT Del bila cuaca cerah dan ada angkutan umum. Bila tidak, saya tidak akan mengikuti kesempatan itu.

Di dalam bahasa pemrograman pun terdapat konep "kondisi" atau *conditional statement* yang sering diterapkan oleh *programmer*. Lihat contoh di bawah ini, yang merupakan terjemahan dari kehidupan sehari-hari menjadi kode program *python*:

```
weather = 'cerah'
isPublicTransportAvailable = True

if weather is 'cerah' and isPublicTransportAvailable is True:
   print("Saya berangkat mengikuti pelatihan")
else:
   print("Saya tidak berangkat mengikuti pelatihan")
```

Bila anda perhatikan *variable* 'isPublicTransportAvailable', disana diisikan dengan nilai 'True'. Perlu anda ketahui, di dalam bahasa pemrograman 'True' memiliki makna 'Ya/Ada' sedangkan 'False' artinya 'Tidak/Tidak Ada'.

Penerapan *conditional statement* ditandai dengan kata kunci 'if' (jika) dan 'else' (lainnya). Untuk kata kunci 'is' memiliki arti 'adalah' dan 'and' merupakan *logical operator* (lihat contoh sebelumnya di atas). Sehingga bila dibahasa-manusiakan kode program di atas menjadi jika weather adalah 'cerah' dan isPublicTransportAvailable adalah 'Ya/Ada' maka cetak 'Saya berangkat mengikuti pelatihan', lainnya 'saya tidak berangkat mengikuti pelatihan'. Jalankan kode program di atas di dalam jupyter notebook anda untuk melihat keluarannya. Lakukan percobaan lainnya dengan men-set nilai 'False' ke *variable* 'isPublicTransportAvailable' atau menggantikan nilai 'cerah' dengan 'hujan'. Anda juga dapat menggantikan kata kunci 'and' menjadi 'or', dan lihat perbedaanya.

Contoh bilangan ganjil-genap:

```
batas_atas = 10
batas_bawah = 0

for angka in range(batas_bawah, batas_atas):
    if angka % 2 != 0:
       print("{} adalah bilangan ganjil".format(angka))
    else:
       print("{} adalah bilangan genap".format(angka))
```

Contoh bilangan factorial:

```
batas_atas = 8
batas_bawah = 1
hasil = 1

for angka in range(batas_bawah, batas_atas):
```

```
    hasil = hasil * angka

print("Bilangan factorial 7 adalah {}".format(hasil))
```

Latihan!

Buatlah sebuah *file* jupyter notebook yang baru, dan *rename* dengan nama "W3-Loop_Conditional_Stmnt". Dengan memanfaatkan jupyter notebook tersebut, buatlah program yang dapat mencetak hasil penjumlahan bilangan genap dan bilangan ganjil. Lalu tampilkan kedua hasil tersebut.

2.5 A Quick Intro to Functions/Procedure

Berbicara mengenai **fungsi**, dapat diibaratkan dengan proses menanak nasi dengan *rice cooker*. Ketika anda ingin memasak nasi, anda menuangkan beberapa mangkuk beras dan sejumlah air ke dalam *rice cooker* lalu mencolokannya ke *outlet* listrik, kemudian dalam beberapa menit beras tersebut berubah menjadi 'nasi'. Akan tetapi, *rice cooker* tidak dapat digunakan untuk menggoreng ikan, atau menumis sayur, karena kebermanfaatan *rice cooker* adalah **hanya** menanak nasi (hasilnya).

Di dalam bahasa pemrograman, fungsi memiliki arti satu blok kode program yang dapat digunakan untuk menghasilkan keluaran yang spesifik (layaknya *rice cooker*). Sebuah fungsi di bahasa pemrograman *python* ditandai dengan kata kunci **def**, dan (biasanya) membutuhkan inputan (atau disebut juga dengan *parameter* sebuah fungsi) dan menghasilkan keluaran yang unik, yang ditandai dengan kata kunci **return**. Parameter sebuah fungsi bisa lebih dari satu. Bila dianalogikan dengan proses menanak nasi; beras, air dan listrik adalah inputan (parameter) yang dibutuhkan oleh *rice cooker*. Untuk lebih jelasnya, silahkan lihat kode program di bawah ini:

```
def MenanakNasi(banyakMangkukBeras, banyakGelasAir, jlhListrik):
    hasil = banyakMangkukBeras * banyakGelasAir * jlhListrik

    return hasil
```

Contoh di atas merupakan sebuah fungsi untuk menanak nasi (asumsi saja untuk menanak nasi diperlukan perkalian ketiga parameter fungsi yang ada, seperti: **banyakMangkukBeras**, **banyakGelasAir**, **jlhListrik**. Hasilnya kemudian diberikan dengan menggunakan kata kunci **return**.

Setelah anda mengerti contoh di atas, perlu juga anda ketahui bahwa di dalam bahasa pemrograman ada yang namanya **prosedur**. Prosedur mirip dengan **fungsi**, akan tetapi bedanya, prosedur **tidak** menghasilkan apapun. Contohnya dapat dianalogikan dengan sapu. Sebatang sapu hanya memerlukan inputan dalam bentuk tenaga, akan tetapi kebermanfaatan sapu tidak menghasilkan apapun yang dapat dipegang, hanya kondisi bersih saja. Contoh prosedur yang nyata yang telah anda gunakan adalah prosedur **print**. Lihat contoh-contoh sebelumnya. Prosedur **print** hanya digunakan untuk mencetak. Untuk lebih jelasnya, lihat contoh di bawah ini:

```
def CetakPenjumlahanBilanganGanjil(angka):
    hasil = 0
    for bilangan in range(1, angka+1):
```

```
        if bilangan % 2 != 0:
            hasil = hasil + bilangan
    print("Penjumlahan bilangan ganjil adalaj {}".format(hasil))
```

Ketika anda telah berhasil membuat fungsi ataupun prosedur, anda dapat menggunakannya dengan cara "memanggil" nama fungsi/prosedur tersebut. Lihat contoh di bawah ini cara memanggil prosedur:

```
CetakPenjumlahanBilanganGanjil(10)
```

Kode program di atas menunjukan bahwa prosedur **CetakPenjumlahanBilanganGanjil** akan menghitung total jumlah bilangan ganjil dari 1 sampai 10 (1, 3, 5, 7 dan 9).

Lalu bagaimana dengan memanggil **fungsi**? Karena **fungsi** menghasilkan sebuah *output* (keluaran) maka *output* tersebut harus "ditampung" ke dalam sebuah *variable*. Lihat contoh di bawah ini:

```
def JumlahSeluruhBilanganGanjil(angka):
    hasil = 0
    for bilangan in range(1, angka+1):
        if bilangan % 2 != 0:
            hasil = hasil + bilangan
    return hasil

tampung_hasil = JumlahSeluruhBilanganGanjil(10)
```

Kode program di atas menunjukan sebuah prosedur **JumlahSluruhBilanganGanjil** akan menghitung total jumlah bilangan ganjil dari 1 sampai 10, dan *output* nya akan disimpan ke dalam *variable* **tampung_hasil**.

Latihan!

Buatlah sebuah *file* jupyter notebook yang baru, dan *rename* dengan nama "W4-Function_Procedure". Dengan memanfaatkan jupyter notebook tersebut, buatlah dua fungsi yang dapat menghitung penjumlahan bilangan genap dan bilangan ganjil. Kemudian, buatlah **sebuah** prosedur untuk mencetak kedua hasil tersebut.

2.6 A Quick Intro to Array & Dictionary

Apakah anda pernah tinggal di dalam asrama? Sebuah asrama hanya menampung banyak **wanita saja** atau **pria saja**. Lalu apa hubungannya dengan *array*? Ya, *array* dapat dianalogikan sebagai asrama yang dapat menampung banyak nilai (value) namun homogen.

Pendeklarasian *array* hampir sama dengan *variable* namun berbeda ketika pemberian *value*. *ariable* hanya dapat menampung **sebuah *value*** (nilai) saja, misal `usia_saya = 35`, sedangkan *array* dapat menampung lebih dari satu nilai melalui sebuah *variable* yang dideklarasikan. Untuk membuat sebuah *array*, anda cukup menggunakan '[]'. Untuk lebih jelasnya, lihat contoh di bawah ini:

```
data_nilai_mahasiswa = []
```

Selamat! Anda baru saja berhasil mendeklarasikan sebuah *array* dengan nama 'data_nilai_mahasiswa'. Kini saatnya anda menambahkan beberapa data ke *array* 'data_nilai_mahasiswa'. Lihat contoh di bawah ini:

```
data_nilai_mahasiswa.append(74)
data_nilai_mahasiswa.append(45)
data_nilai_mahasiswa.append(87)
data_nilai_mahasiswa.append(66)
data_nilai_mahasiswa.append(60)
data_nilai_mahasiswa.append(92)
```

Bila anda perhatikan kata kunci 'append', ini merupakan sebuah **prosedur** untuk menambahkan data ke dalam sebuah *array*.

Lalu bagaimana caranya mengakses (melihat) data satu per satu dari sebuah *array*? Melalui index *array*. Perlu anda ketahui (hampir) seluruh bahasa pemrograman memiliki index awal nya adalah **0**, sedangkan index terakhir adalah **n-1** dimana 'n' merupakan banyaknya data di dalam sebuah *array*. Dari contoh di atas, 'n' nya adalah 6. Oleh karena itu index terakhir nya adalah 6-1 = 5. Pertanyaan sederhana, bila anda ingin mengambil angka **87** pada contoh di atas, index berapakah angka tersebut terletak? Index ke-2. Bagaimana dengan angka **60**? Index ke-4. Untuk lebih jelasnya lihat contoh di bawah ini:

```
print(data_nilai_mahasiswa[2])
print(data_nilai_mahasiswa[4])
```

Dengan menggunakan konsep perulangan (loop), berikut contoh mencetak seluruh data yang telah disimpan di *array* dengan menggunakan index:

```
for idx in range(0, 6):
    print(data_nilai_mahasiswa[idx])
```

Bila anda perhatikan contoh di atas, 'range(0, 6)' maksudnya adalah perulangan akan dimulai dari angka **0** sampai **n-1** (atau 5).

Kadang kala bila data yang ada di dalam sebuah *array* cukup banyak, tidak dimungkinkan untuk dihitung secara manual (satu per satu). Oleh karena itu, *python* telah menyediakan sebuah fungsi 'len' yang dapat menghitung secara otomatis total data, berikut contohnya:

```
for idx in range (0, len(data_nilai_mahasiswa)):
    print(data_nilai_mahasiswa[idx])
```

Cara lain untuk melihat data *array* tanpa menggunakan index adalah sebagai berikut:

```
for val in data_nilai_mahasiswa:
    print(val)
```

Latihan!

Buatlah sebuah *file* jupyter notebook yang baru, dan *rename* dengan nama "W5-Array". Dengan memanfaatkan jupyter notebook tersebut, buatlah sebuah *array* yang mampu menampung sembarang angka. Kemudian dengan konsep *loop*, lalukan perhitungan bilangan genap dan ganjil dari data yang anda masukan ke dalam *array* tersebut.

Perlu anda ketahui, di dalam bahasa pemrograman yang lebih *advanced* terdapat *dictionary* untuk menyimpan banyak data (*value*) yang *homogen* layaknya *array*. Sesuai dengan namanya *dictionary* (kamus) memiliki 'key' dan 'value'. Bayangkan ketika anda menggunakan kamus Bahasa Indonesia – Bahasa Inggris, anda akan mencari

arti/terjemahan sebuah kata melalui kata kunci 'key'. Begitu juga di bahasa pemrograman, anda akan mendapatkan sebuah *value* dari sebuah *dictionary* dengan 'key' tersebut. Perhatikan contoh di bawah ini bagaimana mendeklarasikan sebuah *dictionary* di bahasa pemrograman *python*:

```
data_nilai = {}
```

Untuk menambahkan data ke dalam *dictionary*, anda dapat melakukanya dengan cara:

```
data_nilai['key_1'] = 74
data_nilai['key_2'] = 45
data_nilai['key_3'] = 87
data_nilai['key_4'] = 66
data_nilai['key_5'] = 60
data_nilai['key_6'] = 92
```

Bila anda perhatikan kode program di atas, 'data_nilai' merupakan sebuah *dictionary*. 'data_nilai' ini menyimpan 6 data, dan masing-masing data memiliki 'key', misal: key_1, key_2, dst. **Ingat!** Anda tidak dapat menambahkan data dengan 'key' yang sama. Artinya, setiap 'key' itu unik. Lihat contoh yang **salah** berikut ini:

```
data_nilai['key_1'] = 74
data_nilai['key_1'] = 104
```

Meskipun contoh di atas tidak memunculkan *error*, akan tetapi maknanya sudah berbeda. Pada *line*-1 'data_nilai' diisikan dengan data (*value*) '74'. Akan tetapi, pada *line* selanjutnya, datanya sudah berubah menjadi '104'. Ini disebut juga dengan *updating data*. Bila anda tidak bertujuan untuk melakukan perubahaan data berdasarkan 'key' yang spesifik, seperti contoh di atas, melainkan ingin menambah data baru ke dalam sebuah *dictionary*, maka contoh di atas tidak tepat untuk digunakan. Begitu pula sebaliknya.

Untuk melihat data di dalam sebuah *dictionary*, anda dapat melakukan iterasi seperti contoh di atas:

```
for t_key in data_nilai:
    print(data_nilai[t_key])
```

Latihan!

Buatlah sebuah *file* jupyter notebook yang baru, dan *rename* dengan nama "W6-Dictionary". Dengan memanfaatkan jupyter notebook tersebut, buatlah sebuah *dictionary* yang mampu menampung sembarang angka. Kemudian dengan konsep *loop*, lalukan perhitungan bilangan genap dan ganjil dari data yang anda masukan ke dalam *dictionary* tersebut.

Chapter 3. Dasar-dasar Statistika

3.1 Pengenalan Statistika

Kadang kita terjebak dalam pengertian kata statistik dan statistika. **Statistik** merupakan kumpulan data, bilangan atau non bilangan yang disajikan sedemikian rupa (biasanya dalam bentuk tabel atau grafik) yang menggambarkan suatu persoalan atau keadaan. Statistik lebih banyak digunakan untuk menggambarkan keadaan atau permasalahan, seperti:

1. Pencataan banyaknya penduduk
2. Pencatatan hasil pertanian di suatu daerah, dan semacamnya.

Statistika adalah ilmu/pengetahuan yang berhubungan dengan cara-cara pengumpulan, penyajian, pengolahan dan analisis data. Statistika digunakan sebagai cara ilmiah untuk mengumpulkan, menyusun, meringkas dan menyajikan data yang dikenal sebagai **statistika deskriptif**. Lebih lanjut statistika merupakan cara untuk mengolah data tersebut dan menarik kesimpulan yang teliti dan keputusan yang logic dari pengolahan data tersebut (**statistika inferensia**).

Data adalah hasil pengamatan terhadap fenomena atau gejala, baik berupa kebendaan ataupun fenomena peristiwa dari objek yang diteliti. Data juga diartikan sebagai kumpulan angka, fakta, fenomena atau keadaan lainnya yang merupakan hasil pengamatan, pengukuran, atau pencacahan dan sebagainya terhadap variabel dari suatu obyek kajian, yang berfungsi dapat membedakan objek yang satu dengan lainnya pada variabel yang sama. Data juga komponen utama untuk membanggun sebuah sistem cerdas. Dalam dunia teknologi, *big data* berarti terobosan baru yang berkaitan dengan mengolah, menyimpan, dan menganalisis data dalam berbagai format, dimana jumlah data yang diolah, disimpan dan di analisis sangatlah besar dan data yang bertambah dengan cepat. Oleh karena itu, *science*, *technology*, *engineering*, and *mathematics* (STEM) dan *computer science* menjadi suatu tren di dunia pendidikan saat ini. Di negara lain seperti Amerika Serikat (AS), Finlandia, dan Australia, *coding* sudah dibuat dalam kurikulum yang berlaku dari jenjang SD (Sekolah Dasar) hingga SMA (Sekolah Menengah Atas).
Statistik dasar yang dilakukan umumnya tergantung pada kebutuhan dan tujuan penelitian atau pihak pengguna. Bidang kajian atau cakupan statistik deskriptif meliputi:

1. Penyajian data yaitu: distribusi frekuensi; penyajian grafik, bagan dan diagram.
2. Peringkasan/klasifikasi data dikelompokkan menjadi:
    a. **Ukuran tendensi sentral** (pemusatan): mean atau rata-rata, median atau nilai tengah, modus.
    b. **Ukuran letak** (lokasi/pembagian distribusi): kuartil, desil, persentil.
    c. **Ukuran dispersi** (sebaran): selang (range), varians dan deviasi standar, koefisien keragaman, simpangan baku atau kesalahan baku dari rata-rata (standard error).

3.2 Penyajian Data

Berdasarkan sumbernya, data dibagi menjadi dua, yaitu **data primer** dan **data sekunder**. Data primer bersumber dari pengamatan langsung dari lapangan sedangkan data sekunder didapatkan dari data yang telah dikumpulkan oleh pihak lain misalnya pemerintah, badan atau peneliti sebelumnya. Dalam bidang informatika kita mengenal *data dummy* merupakan data buatan yang biasanya digunakan untuk *testing* aplikasi. Tabel 1 adalah *data dummy* warga suatu daerah.

Table 1. Data Warga RT Xx RW Yy Desa Zzz

Jenis Kelamin	Pekerjaan	Usia
Pria	Pegawai	25
Wanita	PNS	42
Pria	Swasta	43
Wanita	Pedagang	37
Pria	PNS	35
Wanita	Swasta	41
Pria	Pedagang	35
Wanita	Swasta	28
Wanita	Pedagang	37
Pria	Pegawai	36
Wanita	PNS	29
Pria	Swasta	27
Wanita	Pedagang	36
Pria	PNS	44
Wanita	Swasta	43
Pria	Pedagang	38
Wanita	Swasta	25
Pria	Pedagang	34
Wanita	Pegawai	31
Wanita	PNS	45

1. Distribusi Frekuensi

Distribusi frekuensi adalah daftar nilai data (bisa nilai individual atau nilai data yang sudah dikelompokan) yang disertai dengan nilai frekuensi atau kemunculan data yang sesuai. Distribusi frekuensi variabel "Gender" dari data yang terdapat pada Tabel 2 merupakan contoh tabel distribusi frekuensi untuk nilai individual.

Table 2. Tabel Banyaknya Warga RT Xx RW Yy Desa Zzz Berdasarkan Gender

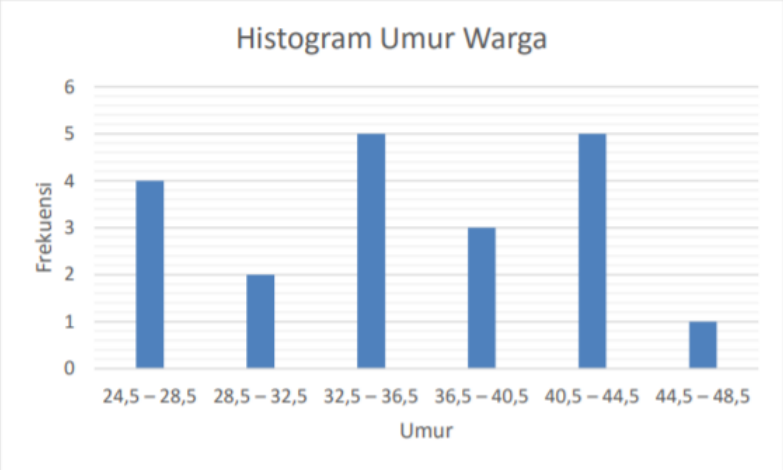
Jenis Kelamin	Usia
Pria	9
Wanita	11
Total	**20**

2. Grafik Batang (*bar chart*)

Grafik batang adalah grafik yang penyajian datanya mengunakan batang atau persegi panjang. Grafik batang atau sering kita kenal dengan sebutan histogram. Grafik batang dipakai untuk memperlihatkan perbedaan tingkat nilai dari beberapa aspek pada suatu data. Grafik batang Tabel 3 terdapat pada Gambar 1.

Table 3. Tabel Distribusi Frekuensi Umur Warga XYZ

Interval	**Frekuensi**
24.5-28.5	4
28.8-32.5	2
32.5-36.5	5
36.5-40.5	3
40.5-44.5	5
44.5-48.5	1
Total	**20**



Gambar 1. Histogram Umur Warga XYZ

3.3 Peringkasan Data

Peringkasan data adalah bagian dari statistika deskriptif yang digunakan untuk melihat ukuran pemusatan, penyebaran maupun letak data.

1. Ukuran tendensi sentral (pemusatan)
   a. Mean atau rata-rata (*average*)
      Rata-rata atau rataan merupakan ukuran pemusatan data. Selain itu, rataan merupakan parameter dari distribusi normal. Rumus rataan untuk data tunggal dan kelompok adalah berbeda.
      Rataan data tunggal:

$$\bar{X} = \frac{\sum_{i=1}^{n} X_i}{n}$$

di mana:
$\bar{X}$: rataan
X_i : data ke-i
n : banyaknya data

Rataan data kelompok:

$$\bar{X} = \frac{\sum_{i=1}^{n} x_{ti} \cdot f_i}{\sum_{i=1}^{n} f_i}$$

di mana:
$\bar{X}$: rataan
x_{ti} : nilai tengah kelompok data ke-i
f_i : frekuensi kelompok data ke-i

b. Median atau nilai tengah
Selain merupakan ukuran pemusatan, median juga merupakan ukuran letak yang menyatakan data yang berada di tengah setelah diurutkan. Median data tunggal didapatkan dengan 2 rumus yang berbeda antara yang banyaknya data (n) ganjil dan genap.
Median data tunggal:
- n ganjil

$$Me = X_{\frac{n+1}{2}}$$

- n genap

$$Me = \frac{X_{\frac{n}{2}} + X_{\frac{n}{2}+1}}{2}$$

Median data kelompok:

$$Me = T_b + \frac{\frac{n}{2} + f_s}{f_m} \times k$$

di mana:
Me : median
n : banyak data
T_b : tepi bawah kelas median, kelas median adalah data ke $\frac{n}{2}$
f_s : frekuensi kelompok data sebelum kelas median
f_m : frekuensi kelompok kelas median
k : panjang kelas interval dalam selang

c. Modus (*mode*)
Modus adalah data yang paling sering muncul. Data tunggal dapat memiliki satu modus (unimodus), lebih dari satu modus (multimodus) atau tidak punya modus (unmodus). Sedangkan pada data berkelompok, nilia modus dihitung dengan:

$$M_o = T_b + \frac{d_1 + d_2}{f_m} \times k$$

di mana:
M_o : modus
T_b : tepi bawah kelas modus, kelas modus adalah kelas dengan frekuensi terbanyak
d_1 : selisih frekuensi kelas modus dengan kelas sebelumnya
d_1 : selisih frekuensi kelas modus dengan kelas setelahnya
f_m : frekuensi kelompok kelas modus
k : panjang interval kelas dalam selah

2. Kuartil sebagai ukuran letak/ lokasi/pembagian distribusi
   Kuartil dibagi menjadi kuartil 1 (Q_1), kuartil 2 atau median (Q_2=Me) dan kuartil 3 (Q_3). Kuartil 1 dan 3 pada data tunggal diperoleh dengan seperti mencari median dari data baru setelah median ditemukan. Kuartil 1 adalah median data sebelum median sebenarnya sedangkan Kuartil 3 adalah median data setelah median sebenarnya. Kuartil data kelompok adalah sebagai berikut:

$$Q_i = T_b + \frac{\frac{i \times n}{4} + f_s}{f_m} \times k$$

   di mana:
   Q_i : kuartil ke-i
   n : banyaknya data
   T_b: tepi bawah kelas kuartil, kelas kuartil adalah data ke $\frac{i \times n}{4}$
   f_s : frekuensi kelompok data sebelum kelas kuartil
   f_m : frekuensi kelompok kelas kuartil
   k : panjang kelas interval dalam selang

3. Varians dan standar deviasi sebagai ukuran dispersi (sebaran)
   Standar deviasi atau simpangan baku adalah akar dari varians. Varians menyatakan kuadrat dari simpangan data terhadap rataanya. Pada didtribusi normal, stsndar deviasi adalah jarak rataan terhadap titik belok kurva.
   <u>Varians data tunggal:</u>

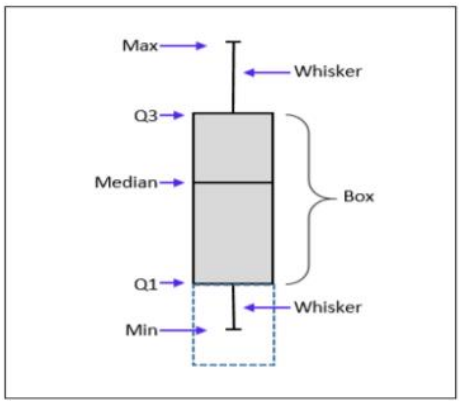
$$s^2 = \frac{\sum_{i=1}^{n}(X_i - \bar{X})}{n-1}$$

   <u>Varians data kelompok:</u>

$$s^2 = \frac{\sum_{i=1}^{n}(x_{ti} - \bar{X})^2 \times f_i}{\sum_{i=1}^{n} f_i}$$

4. Box Plot (visualisasi pemusatan dan sebaran data)

Boxplot disebut juga Peta Wiskher. Boxplot merupakan ringkasan dari batas bawah (BB) atau Min, Q_1, Q_2= Me dan Q_3 serta batas atas (BA) atau Max. Data yang nilainya kurang dari batas bawah atau lebih dari batas atas disebut sebagai outlier (data pencilan). Tampilan Box-Plot dapat dilihat pada Gambar 2 di bawah.



Gambar 2. Box-Plot

Batas Bawah (BB) dan Batas Atas (BA) diperoleh dari:

$$Max = Q_3 + IQR$$
$$Min = Q_1 - IQR$$

di mana:

$$IQR = \frac{3}{2}(Q_3 - Q_1)$$

Chapter 4. Working out with Python Library

Setelah anda mempelajari dasar pemrograman *python* dan statistika di atas, sekarang bagaimana anda dapat memanfaatkan kode program untuk mempermudah pengolahan data. Melakukan kegiatan statistik biasanya melibatkan data yang cukup banyak dan kompleks. Mengolah data tersebut dengan contoh kode program di atas cukup sulit, meskipun dimungkinkan. Bahasa pemrograman *python* termasuk bahasa pemrograman *advanced* di mana bahasa pemrograman ini telah menyediakan banyak *library* untuk melakukan kegiatan statistika. Apa maksudnya? Analoginya seperti "dapur", anda memerlukan alat untuk menanak nasi (*rice cooker*) pasti berada di dapur. Anda memerlukan alat untuk memotong sayuran/buah-buahan, pasti alat tersebut berada di dapur. Begitu pula dengan *library* yang ada di bahasa pemrograman. *Library* dapat diibaratkan seperti dapur, yang menyediakan berbagai "alat" untuk memudahkan anda menyelesaikan pekerjaan komputasi, seperti menghasilkan grafik distribusi frekuensi, menghitung rerata secara otomatis, mendapatkan hasil standard deviasi, dsb. Adapun *library* yang akan anda pakai untuk pemrograman statistika ini adalah: numpy, pandas, dan matplotlib. Untuk lebih jelasnya ketikan kode program berikut pada *jupyter notebook* anda:

```
1  import os
   import pandas as pd
   import numpy as np
   import matplotlib.pyplot as plt
```

Pada kode program di atas (line 1), anda melakukan 'impor' *library* yang dibutuhkan untuk kegiatan statistik.

```
2  current_dir = os.getcwd()
   file_data = "{}/{}".format(current_dir, "data_warga_xyz.txt")
   df = pd.read_csv(file_data, delimiter = ',')
```

Pada line ke-2 menunjukan cara membaca *file* data dengan pembatas koma (csv). Data yang dibaca akan disimpan dalam sebuah *variable* yaitu 'df'.

```
3  df
```

Pada line ke-3 bagaimana cara melihat data yang telah disimpan ke dalam *variable* 'df'.

```
4  df.describe()
```

Pada line ke-4, prosedur *describe* akan menampilkan informasi statistik dari data yang anda miliki, seperti: count (banyaknya data), mean (rerata), std (standar deviasi), min (nilai minimum dari usia), 25% (Q_1), 50% (Q_2 / median), 75% (Q_3), dan max (nilai maximum dari usia).

```
5  ax = df['Gender'].value_counts().plot.bar(rot=0)
```

Pada line ke-5 menunjukan bagaimana melakukan *plotting data* menjadi sebuah grafik perbandingan jumlah 'wanita' dengan 'pria'. Bila anda ingin melihat detail angka jumlah 'pria' dan 'wanita', anda dapat melihat pada line ke-6 di bawah ini:

```
6 df['Gender'].value_counts().to_frame()
```

Dengan cara yang sama anda dapat menampilkan informasi statistik untuk pekerjaan warga XYZ.

```
7 df['Pekerjaan'].value_counts().to_frame()
```

Pada line ke-7 menunjukan bagaimana melihat informasi statistik pekerjaan warga XYZ. Untuk melakukan *plotting data* pekerjaan, anda dapat melakukannya seperti kode program line ke-5 di atas, cukup dengan mengganti 'Gender' menjadi 'Pekerjaan'. Lihat kode program di bawah ini:

```
8 ax = df['Pekerjaan'].value_counts().plot.bar(rot=0)
```

Untuk dapat menampilkan grafik usia dalam bentuk box-plot, lihat line ke-9 di bawah ini:

```
9 cx = df.plot.box(rot=0)
```

Latihan!

Buatlah sebuah *file* jupyter notebook yang baru, dan *rename* dengan nama "W7-Library". Dengan mengikuti contoh di atas, tampilkanlah informasi statistik untuk dataset data_pengguna_inet.csv.