



Belajar Pemrograman Bahasa Python

Buku pendukung praktikum
mata kuliah **Program Komputer I**
Program Studi Sarjana Terapan Statistika Bisnis
Fakultas Vokasi ITS



Wahyu Wibowo
Brodjol Sutijo Suprih Ujama
Harun Al Azies

BELAJAR PEMROGRAMAN BAHASA PYTHON

Buku pendukung praktikum mata kuliah Program Komputer I
Program Studi Sarjana Terapan Statistika Bisnis
Fakultas Vokasi ITS

**Wahyu Wibowo
Brodjol Sutijo Suprih Ulama
Harun Al Azies**

 **itspress**
2020

Belajar Pemrograman Bahasa Python

Buku pendukung praktikum mata kuliah Program Komputer I
Program Studi Sarjana Terapan Statistika Bisnis
Fakultas Vokasi ITS

Penulis : Wahyu Wibowo, Brodjol Sutijo Suprih Ulama, Harun Al Azies
Desain Sampul :

© 2020, ITS Press, Surabaya

Hak cipta dilindungi undang-undang
Diterbitkan pertama kali oleh
ITS PRESS, Surabaya 2020

ISBN 978-602-5542-87-9



9 786025 542879

Anggota IKAPI dan APPTI

**Dilarang keras menerjemahkan, memfotokopi, atau memperbanyak
sebagian atau seluruh isi buku ini tanpa izin tertulis dari penerbit.**

Barangsiapa dengan sengaja dan tanpa hak melakukan perbuatan yang melanggar HAK CIPTA pada buku ini, akan dikenai sangsi sesuai undang-undang nomor 19 tahun 2002 pasal 72.

**Dicetak oleh Percetakan ITS Press
Isi di luar tanggung jawab percetakan**

Belajar Pemrograman Bahasa Python

Buku pendukung praktikum mata kuliah Program Komputer I
Program Studi Sarjana Terapan Statistika Bisnis
Fakultas Vokasi ITS

Wahyu Wibowo
Brodjol Sutijo Suprih Ulama
Harun Al Azies

*Buku ini tidak diperjualbelikan dan penulisannya ini didukung
melalui hibah Penelitian Dasar Unggulan Perguruan Tinggi
Program Deputi Bidang Penguatan Riset Dan Pengembangan
Institut Teknologi Sepuluh Nopember Tahun Anggaran 2020
dengan nomor kontrak : 1166/PKS/ITS/2020*

KATA PENGANTAR

Tim Penulis mengucapkan puji syukur ke hadirat Allah Yang Maha Kuasa atas terselesaikannya buku belajar pemrograman dalam bahasa Python.

Buku ini ditujukan sebagai buku pendukung kegiatan praktikum mata kuliah Program Komputer 1 yang merupakan mata kuliah wajib semester I mahasiswa Program Studi Sarjana Terapan Statistika Bisnis. Tentu, praktikum sangat penting sebagai cara efektif untuk belajar konsep dan ketrampilan pemrograman. Untuk meningkatkan pengetahuan dan ketrampilan, mahasiswa disarankan untuk membaca buku-buku lain seperti yang dituliskan diakhir setiap bab.

Penulis juga mengucapkan terima kasih kepada Institut Teknologi Sepuluh Nopember yang telah memberi dukungan penulisan buku ini melalui hibah Penelitian Dasar Unggulan Perguruan Tinggi Program Deputi Bidang Penguatan Riset Dan Pengembangan Institut Teknologi Sepuluh Nopember Tahun Anggaran 2020 dengan nomor kontrak : 1166/PKS/ITS/2020.

Demikian, semoga buku singkat ini bermanfaat meski disadari sepenuhnya masih banyak kekurangan, semoga bisa disempurnakan di waktu mendatang.

Surabaya, 2 Agustus 2020

Tim Penulis

DAFTAR ISI

KATA PENGANTAR	i
DAFTAR ISI.....	ii
BAB 1. PENGANTAR BAHASA PYTHON	1
1.1. Pengantar Python	1
1.2. Memulai Pemrograman Python	3
1.3. Instalasi Python Melalui Tools Anaconda	6
1.4. Belajar Python Dasar : Memahami Jupyter Notebook.....	15
1.5. Simulasi Hands on Jupyter Notebook	20
BAB 2. BILANGAN	23
2.1. Bilangan (Number).....	23
2.2. Operator Aritmatika	23
2.3. Fungsi <i>print</i>	24
2.4. Variabel	25
2.5. Fungsi <i>abs</i> , <i>int</i> , dan <i>round</i>	26
2.6. Operator Penugasan.....	28
2.7. Dua Operator Integer Lainnya.....	30
2.8. Tanda kurung, Urutan diutamakan.....	31
2.9. Tiga Jenis Kesalahan	31
BAB 3. STRING.....	36
3.1. Pengertian String	36
3.2. Variabel pada String	36
3.3. Indeks dan Irisan.....	37
3.4. Indeks Negatif.....	39
3.5. Batas Default untuk Irisan.....	39
3.6. Penggabungan string	40
3.7. Pengulangan String.....	40
3.8. Fungsi dan Metode String	40
3.9. Metode Dirantai	42
3.10. Fungsi <i>input</i>	42
3.11. Fungsi <i>int</i> , <i>float</i> , <i>eval</i> , dan <i>str</i>	43
3.12. Dokumentasi Internal.....	45
3.13. Garis Lanjutan	46

3.14. Mengindeks dan Mengiris Batas	47
BAB 4. OUTPUT	50
4.1. Pengertian Output	50
4.2. Argumen sep=" " dan end=" "	50
4.3. <i>Escape Character</i>	51
4.4. Argumen ljust (), rjust (), center ()	52
4.5. Metode Format	53
BAB 5. LIST, TUPLE DAN INTRO FILE	58
5.1. Pengertian List	58
5.2. Slices/Irisan	60
5.3. Metode split dan join	61
5.4. File Teks	63
5.5. Pengertian Tuple	64
5.6. Nested Lists	67
5.7. Objek yang Tidak Berubah dan Dapat Berubah	68
5.8. Menyalin List	69
5.9. Indexing, Deleting, and Slicing Out of Bounds	69
BAB 6. RELATIONAL DAN OPERATOR LOGIS	73
6.1. Nilai-nilai Karakter ASCII	73
6.2. Operator Relasional	74
6.3. Menyortir Item dalam List	77
6.4. Operator Logis	78
6.5. Evaluasi Short-Circuit	80
6.6. Tipe Data Boolean/bool	81
6.7. Tiga Metode Yang Mengembalikan Nilai Boolean	81
6.8. Ketentuan Penyederhanaan	83
BAB 7. DECISION STRUCTURE	87
7.1. Pengertian Kondisi If	87
7.2. Pengertian Kondisi If Else	90
7.3. Pengertian Kondisi If Else If /elif	92
7.4. Masukkan Validasi dengan Pernyataan if-elif-else	95
7.5. True and False	96
BAB 8. WHILE LOOP	100
8.1. Struktur Perulangan While Bahasa Python	100

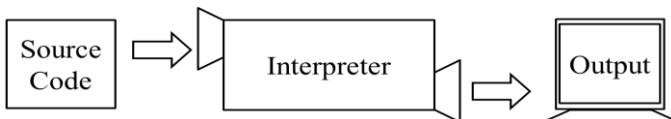
8.2. Infinite Loops	102
8.3. Statemen Break	103
8.4. Statemen Continue.....	104
8.5. Membuat Menu.....	105
BAB 9. FOR LOOP	109
9.1. Pengertian Struktur Perulangan For Bahasa Python.....	109
9.2. Penggunaan Function range()	111
9.3. Pass Statemen	113
9.4. Nested loop.....	113
BAB 10. FUNGSI	117
10.1. Fungsi (Function)	117
10.2. Fungsi Bawaan	117
10.3. Fungsi Buatan Pengguna	118
10.4. Fungsi Memiliki Satu Parameter.....	119
10.5. Melewati Nilai ke Fungsi	120
10.6. Fungsi Memiliki Beberapa Parameter.....	121
10.7. Fungsi Bernilai Boolean dan List	121
10.8. Fungsi yang Tidak Mengembalikan Nilai	122
10.9. Fungsi Tanpa Parameter.....	123
10.10. Ruang Lingkup (Scope) Variabel.....	123
10.11. Penamaan Konstanta	124

BAB 1. PENGANTAR BAHASA PYTHON

1.1. Pengantar Python

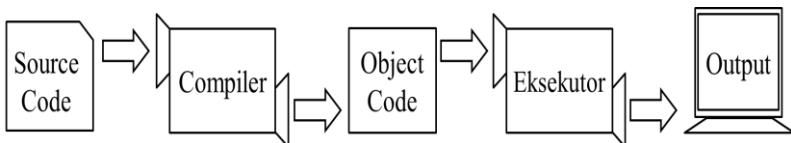
Guido van Rossum menjadi pencipta salah satu bahasa pemrograman yang popular yaitu python pada tahun 1990 di Belanda tepatnya di CWI atau *Centrum Wiskunde & Informatica*. Penciptaan python sendiri merupakan projek kelanjutan dari bahasa pemrograman yang telah ada sebelumnya, yaitu bahasa pemrograman jenis ABC. Versi 1.2 menjadi versi terakhir python yang dirilis oleh CWI pada tahun 1995, ditahun yang sama tepatnya di *Corporation for National Research Initiative* (CNRI) di negara Virginia Amerika Guido masih aktif melakukan projek pengembangan python. Pengembangan python terus dilakukan, dan pada tahun 2001, melalui *Python Software Foundation* (PSF) sebuah organisasi yang Guido gunakan untuk mengembangkan python, melalui PSF segala hal terkait pengembangan hingga hak intelektual python dilakukan. Sekilah informasi terkait penamaan python, sebenarnya nama python yang dipakai oleh Guido bukan berasal dari nama ular yang kita kenal. Melainkan nama grup komedi dari Negara Inggris yaitu Monty Python.

Python sebagai bahasa pemrograman yang populer dan komprehensif dengan menggabungkan kapabilitas, sintaksis kode yang jelas serta dilengkapi pustaka standar yang mempunyai fungssionalitas sangat besar. Python termasuk dari jajaran bahasa pemograman tingkat tinggi seperti bahasa pemograman C, C++, Java, Perl dan Pascal. Sedangkan bahasa pemograman tingkat rendah adalah bahasa mesin yaitu bahasa pemograman Assembly. Dalam bahasa pemrograman tingkat tinggi, terbagi menjadi dua jenis cara untuk memproses bahasa tingkat tinggi ke bahasa tingkat rendah, yaitu : compiler dan interpreter. Jenis pertama adalah interpreter, cara kerjanya cukup mudah, yaitu membaca sebuah program setiap baris yang ditulis dengan bahasa tingkat tinggi. Interpreter nantinya akan memproses langsung per baris untuk mengeluarkan outputnya.



Gambar 1.1. Visualisasi Proses Interpreter

Jenis aplikasi selanjutnya dalam bahasa pemrograman tingkat tinggi adalah kompiler. Cara kerja compiler berbeda dengan interpreter. Kompiler bekerja dengan cara keseluruhan kode program diterjemahkan dahulu sebelum menjalankan program tersebut. Gambar 1.2 menjelaskan bagaimana sebuah kompiler bekerja. Kode program disebut source code sedangkan program yang diterjemahkan disebut dengan object code atau executable. Sekali program tersebut di kompilasikan, Anda dapat mengeksekusinya berulang kali tanpa menerjemahkannya lagi kedalam object code.



Gambar 1.2. Visualisasi Proses Kompiler

Setiap bahasa pemrograman pasti memiliki ciri khas, namun ada beberapa komponen pada bahasa pemrograman komputer yang instruksi umumnya sama di semua bahasa pemrograman komputer, walaupun terkadang komponen lebih seringnya berbeda – beda.

- input** : Masukan dari keyboard, file, atau beberapa device.
- output** : Hasil / keluaran program ke monitor display, file, atau beberapa device.
- math** : Perhitungan matematika atau kalkulasi matematika seperti pengurangan, penjumlahan, perkalian, pembagian dan sebagainya.
- kondisi** : Memeriksa beberapa kondisi dan mengeksekusi beberapa perintah tertentu, sesuai dengan kondisi yang telah diperiksa.
- Perulangan** : Menjalankan beberapa perintah secara berulang - ulang kali, biasanya dengan beberapa variasi.

1.2. Memulai Pemrograman Python

Python merupakan bahasa *interpreter* seperti dijelaskan pada bab sebelumnya, cara kerja python membaca sebuah program setiap baris yang ditulis dengan bahasa tingkat tinggi. Interpreter nantinya akan memproses langsung per baris untuk mengeluarkan outputnya. Python akan secara otomatis eksekusi programnya jika terdapat kesalahan program di tengah eksekusi. Ada dua cara menjalankan *interpreter* python ini untuk selanjutnya kita hanya akan menyebut python saja. Yang pertama adalah dengan menggunakan program **command prompt** (cmd) di windows. Yang kedua menggunakan program IDLE yang merupakan bawaan python sendiri. Berikut merupakan operator yang biasa dipakai dalam bahasa pemrograman python.

Tabel 1.1. Operator Aritmatika

Operator	Deskripsi	Contoh
+	Penjumlahan	$3 + 4$ bernilai 7
-	Pengurangan	$8 - 1$ bernilai 7
*	Perkalian	$1 * 7$ bernilai 7
/	Pembagian	$7 / 1$ bernilai 7
//	Pembagian (dibulatkan kebawah)	$15 // 2$ bernilai 7
%	Sisa Bagi / Modulo	$13 \% 5$ bernilai 3

Tabel 1.2. Operator Assignment

Operator	Deskripsi	Contoh
=	Assignment	$N = 7$
+=	Penjumlahan	$N += 7$, N akan ditambah 7.
-=	Pengurangan	$N -= 7$, N akan dikurangi 7.
*=	Perkalian	$N *= 7$, N akan dikalikan 7.
//=	Pembagian (dibulatkan kebawah)	$N //= 7$, N akan dibagi 7 (dibulatkan kebawah)
%=	Sisa Bagi / Modulo	$N \%= 7$, N akan dimodulo 7.
=	Assignment	$N = 7$

Tabel 1.3. Operator Relasional

Operator	Deskripsi	Contoh True	Contoh False
----------	-----------	-------------	--------------

==	Sama dengan	$7 == 7$	$2 == 3$
!=	Tidak Sama dengan	$7 != 2$	$3 != 3$
<	Kurang dari	$7 < 8$	$7 < 7$
>	Lebih dari	$8 > 7$	$7 > 8$
<=	Kurang dari sama dengan	$7 <= 7$	$7 <= 8$
>=	Lebih dari sama dengan	$8 >= 7$	$2 >= 4$

Tabel 1.4. Operator Logika

Operator	Deskripsi	Contoh True	Contoh False
and	Dan	$(1 < 2) \text{ and } (3 == 3)$	$(1 == 2) \text{ and } (3 == 3)$
or	Atau	$(1 < 2) \text{ or } (4 == 3)$	$(3 < 2) \text{ or } (2 == 3)$
not	Negasi	$\text{not } (3 < 2)$	$\text{not } (1 > 2)$

a) Statement (Pernyataan) di Python

Setiap perintah yang dapat dieksekusi oleh Python disebut statement. Misalnya, `a = 1` adalah sebuah statement penugasan. Jenis statement penugasan dalam python akan dibahas lebih lanjut pada bab selanjutnya.

b) Statement Multibaris

Di Python, akhir dari sebuah statement adalah karakter baris baru (newline). Kita dapat membuat sebuah statement terdiri dari beberapa baris dengan menggunakan tanda backslash (\). Misalnya:

```
a = panjang1 + panjang2 +\
    panjang3 +\
    panjang4
```

Statement yang ada di dalam tanda kurung [], { }, dan () tidak memerlukan tanda \. Contohnya:

```
Nama_bulan = ['Januari', 'Maret', 'Juni', 'September']
```

c) Baris dan Indentasi

Python tidak menggunakan tanda { } untuk menandai blok / grup kode. Blok kode di python menggunakan tanda indentasi (spasi).

Jumlah spasi untuk setiap baris yang ada dalam satu blok kode harus sama. Contoh yang benar adalah sebagai berikut:

```
if nilai <= 5 :  
    print("Nilai merah")  
    print("Tidak Lulus")  
else :  
    print("Nilai biru")  
    print("Lulus")
```

Bila indentasi dalam satu grup kode tidak sama, python akan menampilkan sintaks error.

```
if True :  
    print("Jawab")  
    print("Benar")  
else :  
    print("Jawab")  
    print("Salah")
```

SyntaxError : unexpected indent

d) Tanda Kutip di Python

Python menggunakan tanda kutip tunggal ('), ganda ("'), maupun triple ("") atau ("""") untuk menandai string, sepanjang stringnya diawali oleh tanda kutip yang sama di awal dan akhir string. Tanda kutip tiga digunakan untuk string multibaris. Ketiga contoh berikut, semuanya adalah benar,

```
kata = 'kata'  
kalimat ="Ini adalah kalimat"  
paragraf ="""Ini adalah paragraf. Paragraf  
terdiri dari beberapa baris"""
```

e) Komentar di Python

Tanda pagar (#) digunakan untuk menandai komentar di python. Komentar tidak akan diproses oleh interpreter Python. Komentar hanya berguna untuk programmer untuk memudahkan memahami maksud dari kode.

```
#Komentar pertama
print("Statistika Bisnis" ) #Komentar kedua
```

Kode di atas akan menghasilkan keluaran:

```
Statistika Bisnis
```

Python tidak memiliki fitur komentar multibaris. Kita harus mengomentari satu persatu baris seperti berikut:

```
#Ini komentar
#Ini juga adalah komentar
#Ini juga masih komentar
```

1.3. Instalasi Python Melalui Tools Anaconda

Python dapat diinstall di berbagai platform salah satunya adalah sistem operasi windows. Tetapi dalam prakteknya python membutuhkan banyak library untuk memproses data serta memerlukan pengaturan terlebih dahulu sebelum dapat digunakan. Anaconda memberikan alternatif yang lebih baik dalam instalasi python beserta librariinya. Cukup download installer Anaconda dan sekali install python beserta banyak package lainnya sudah terpasang secara otomatis. Pengguna cukup menggunakan berbagai macam fitur tanpa harus repot menginstall satu persatu library python.



Gambar 1.3. Beberapa library python di Anaconda.

Anaconda adalah paket distribusi Python dari **Continuum Analytics** yang berisi paket Python ditambah beberapa paket tambahan untuk keperluan pemrograman data science, matematika hingga teknik dalam satu distribusi *platform* yang user friendly. File instalasi Anaconda dapat diunduh di

<https://www.anaconda.com/products/individual>.

Ada beberapa kelebihan Anaconda seperti:

- 1) Mudah dan Cepat dalam menginstall package untuk data science
- 2) Mudah dalam mengatur library, dependensi dan lingkungan dengan Conda
- 3) Library machine learning sudah tersedia seperti scikit-learn, TensorFlow, Theano untuk melakukan pengembangan dan pelatihan data
- 4) Library data analysis seperti numpy, pandas dan Numba
- 5) Library visualisasi yang sudah ada seperti Dash, Matplotlib, Bokeh, Datashader dan lainnya
- 6) Jupyter Notebook sebagai lingkungan IDE berbasis web yang mudah digunakan

Berikut merupakan tahapan instalasi anaconda. Pertama silahkan akses halaman resmi Anaconda di

<https://www.anaconda.com/products/individual>

Disana tersedia installer dalam 3 sistem operasi yaitu: windows, linux dan MacOS. Biasanya halaman download akan secara otomatis mengenali sistem operasi yang kita gunakan sehingga diarahkan di menu download sesuai dengan sistem operasi kita.



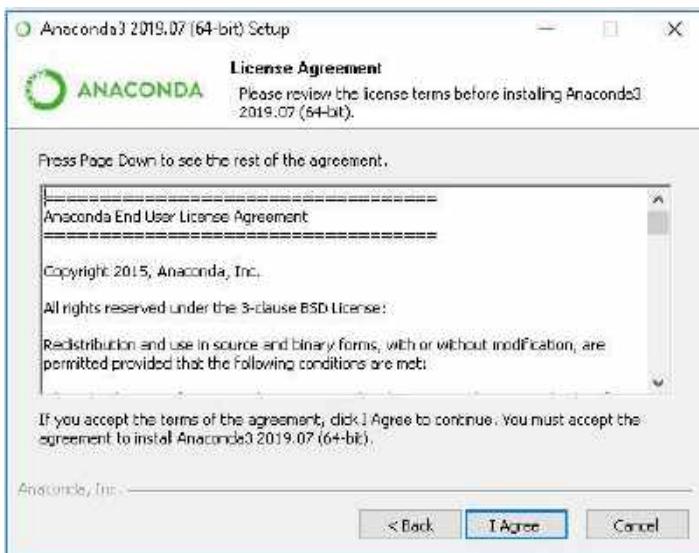
Gambar 1.4. Tampilan Anaconda Installer.

Anaconda memberikan pilihan installer python versi 2 dan python versi 3. Saat ini kita fokus ke Python 3 karena merupakan jenis python yang banyak digunakan sedangkan python 2 pada tahun 2020 akan mulai dihentikan pengembangannya. Versi Python 3 saat penulis mendownload installer adalah versi 3.7 . Bagi yang ingin mengunduh versi lamanya seperti 3.6, 3.5 dan seterusnya anda dapat mencari link archive di halaman official Anconda untuk melihat berbagai versi python yang tersedia.

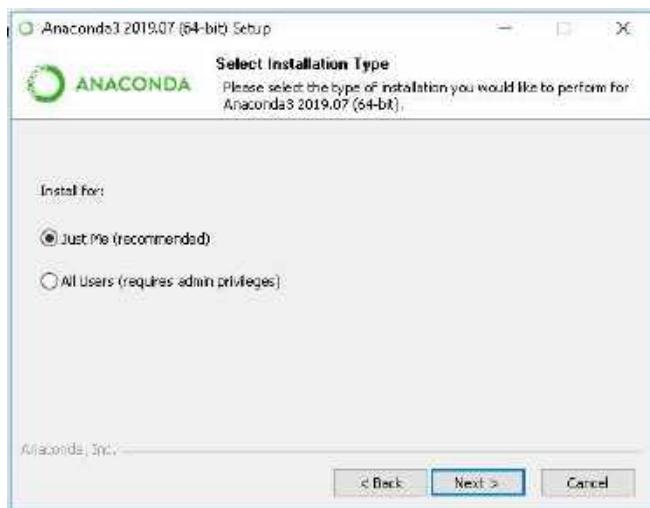
Setelah di download selanjutnya kita install dengan mengklik 2 kali (double click) pada file installer Anaconda sehingga muncul pop up seperti dibawah ini



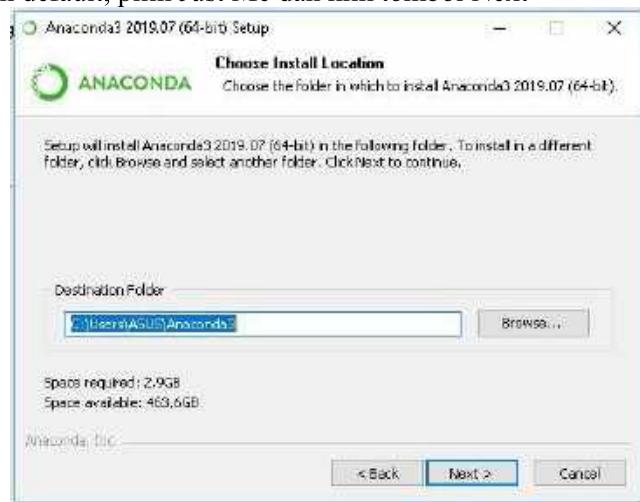
Tekan Tombol Next untuk melanjutkan



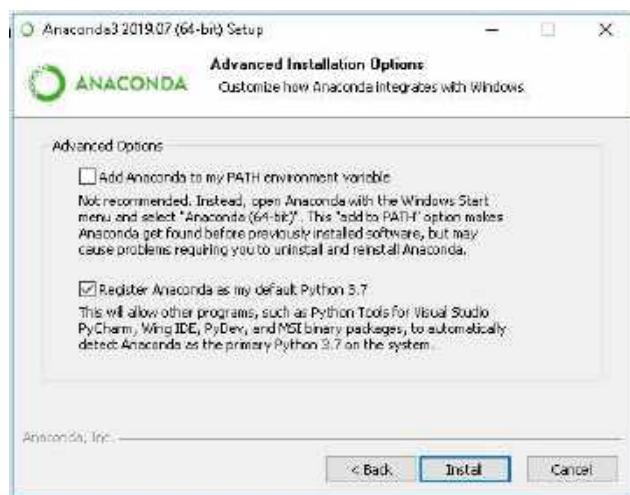
Anda akan masuk pada License Agreement. Pilih Tombol I Agree untuk melanjutkan



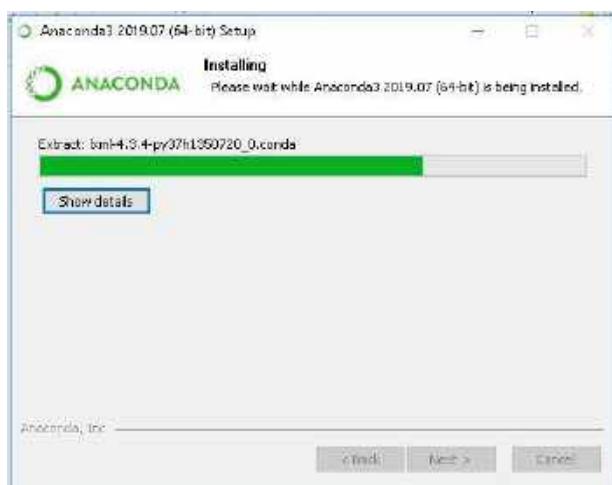
Biarkan default, pilih Just Me dan klik tombol Next



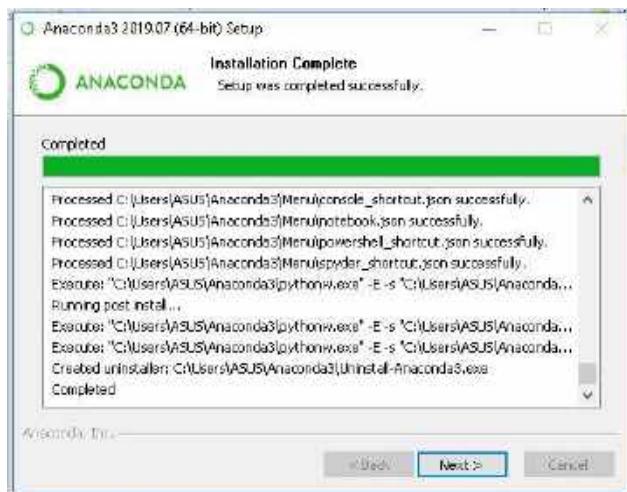
Pilihlah lokasi instalasi Anaconda. Biarkan default saja dan tekan tombol Next



Biarkan default dan tekan tombol Install



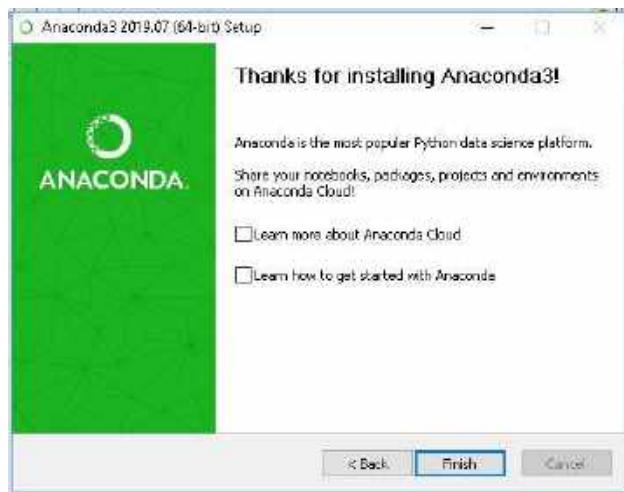
Proses instalasi sedang berlangsung. Tunggu beberapa menit dan ketika selesai tekan tombol Next



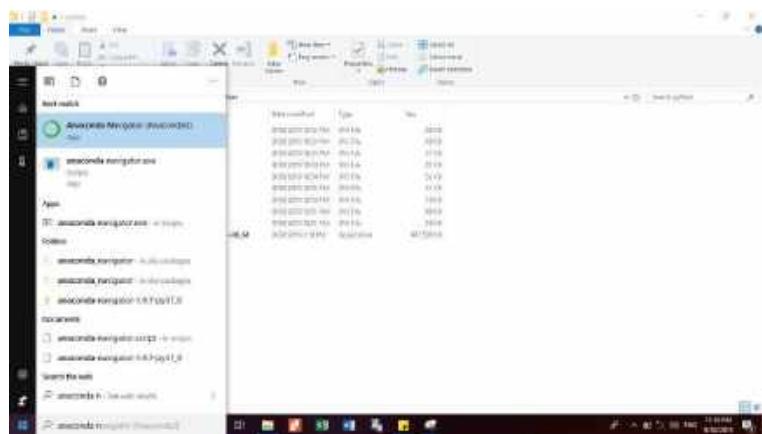
Muncul halaman informasi tambahan untuk rekomendasi IDE menggunakan pyCharm. Tekan Next saja



Halaman instalasi terakhir. Hilangkan centang dan tekan tombol Finish.

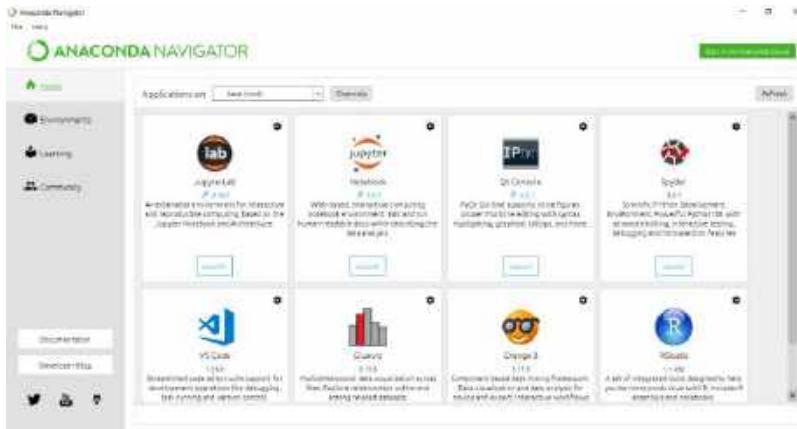


Selamat anda sudah berhasil menginstall python beserta banyak library. Untuk melakukan verifikasi coba buka Anaconda Navigator melalui tombol start di keyboard dan ketik “anaconda navigator”



Anaconda Navigator sebuah antarmuka (interface) dari Anaconda yang dimaksudkan agar para pengguna dipermudah dalam membuka aplikasi. Didalam anaconda navigator nantinya terdapat

banyak aplikasi. Selain aplikasi, anaconda navigator juga digunakan untuk memanajemen paket, environment, dan channel tanpa menggunakan command-line. Navigator ini akan mencari paket yang ada di Anaconda Cloud atau di repositori local Anaconda. Anaconda Navigator dapat mencari paket pada repositori cloud anaconda. Ada beberapa IDE yang dapat dijalankan melalui **Anaconda Navigator**. **Jupyter** dan **Spyder** adalah dua diantaranya. Pembaca dapat mencoba menjalankannya dengan mengklik button **Launch**



Melalui Anaconda Navigator, programmer dapat dengan mudah menjalankan dan mengelola paket Conda, *environment* dan kanal tanpa perlu menggunakan perintah *command prompt*. Terakhir, untuk mengecek versi anaconda dan python yang terinstall buka command prompt melalui start (seperti memanggil Anaconda Navigator) dan ketik “cmd” Untuk mengecek versi anaconda ketik conda –V



The screenshot shows a terminal window titled 'Anaconda Prompt (Anaconda3)'. The command 'conda -V' is run, displaying 'conda 4.7.10'. The command 'python -V' is run, displaying 'Python 3.7.3'. The prompt then shows three question marks, indicating the user is waiting for input.

```
(base) C:\Users\ASUS>conda -V
conda 4.7.10

(base) C:\Users\ASUS>python -V
Python 3.7.3

(base) C:\Users\ASUS>?
```

1.4. Belajar Python Dasar : Memahami Jupyter Notebook

Jupyter Notebook merupakan tool yang populer untuk mengolah data di python. Jupyter Notebook memungkinkan untuk mengintegrasikan antara kode dengan output di dalam satu dokumen secara interaktif. Jupyter Notebook sudah otomatis terinstall ketika kita telah menginstall python dengan anaconda.

a. Menjalankan Jupyter Notebook

Cara menjalankan Jupyter Notebook adalah dengan mengetik *jupyter notebook* di terminal.

Jupyter Notebook diakses melalui browser. Saat servis dijalankan anda akan diarahkan ke halaman browser pada alamat <http://localhost:8888/tree>

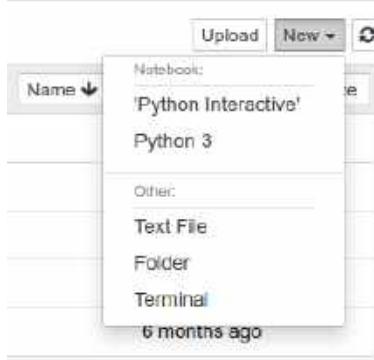
Halaman Jupyter Notebook akan muncul di lokasi folder dimana kita menjalankan sintaks *jupyter notebook*



Gambar 1.5. Halaman Jupyter Notebok

b. Operasi File / Folder

Di Jupyter Notebook kita bisa membuat file dan folder seperti kita membuatnya di halaman sistem operasi seperti Windows/Linux/Mac



Gambar 1.6. Membuat File / Folder di Jupyter Notebook

Saat membuat file atau folder di Jupyter Notebook secara otomatis akan bernama 'Untitled'



Gambar 1.7. Untitled File dan Folder

Untuk merubah nama folder centang “Untitled folder” kemudian tekan tombol **Rename** dan masukkan nama baru untuk Folder



Gambar 1.8. Rename Folder

Sedangkan untuk merubah nama file masuk ke file “Untitled.ipynb” dan klik judul **Untitled** disamping logo Jupyter dan masukkan nama file baru



Gambar 1.9. Rename File

c. Cell

Cell merupakan area untuk menaruh kode di Jupyter Notebook.



Gambar 1.10. Cell di Jupyter Notebook

Untuk menjalankan cell tekan SHIFT+ENTER di keyboard atau tombol RUN di Toolbar pada cell aktif sehingga kode akan dieksekusi dan menghasilkan output



```
In [1]: print("Statistik Eksis")
Statistik Eksis
```

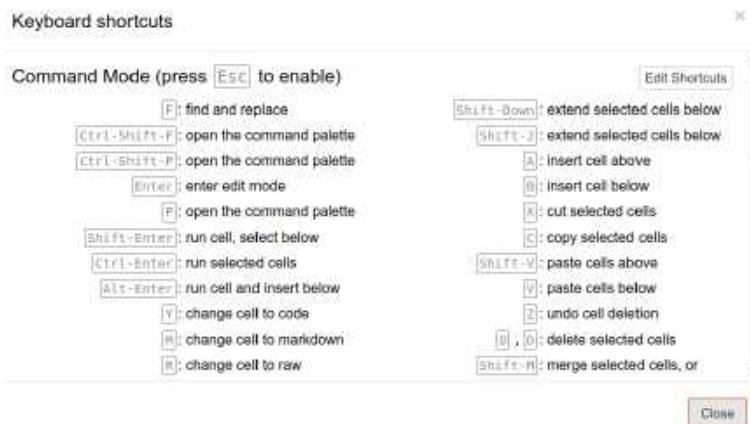
Gambar 1.11. Tampilan output pada python

Ada 2 tipe cell yaitu **code cell** dan **markdown cell**

- **Code cell** adalah cell untuk menulis dan meneksekusi kode python
- **markdown cell** adalah cell untuk menulis teks yang terformat. Tipe cell dapat dipilih melalui **Toolbar**

d. Keyboard Shortcuts

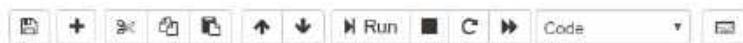
Keyboard Shortcuts digunakan untuk melakukan perintah dengan cepat menggunakan keyboard. Keyboard shortcut seperti menambah cell, menghapus cell, melakukan copy paste, menjalankan kode dan sebagainya. Detail keyboard shortcuts dapat di lihat di Menu **Help -> Keyboard Shortcuts**



Gambar 1.12. Detail Keyboard Shortcuts Jupyter Notebook

e. Toolbar

Toolbar mempunyai beberapa tombol shortcuts penting.



Gambar 1.13. Toolbar di Jupyter Notebook

Fungsi dari masing-masing simbol toolbar adalah sebagai berikut.

Tabel 1.5. Toolbar di Jupyter Notebook

Simbol	Keterangan	Deskripsi
	save and checkpoint	menyimpan file dan melakukan checkpoint
	insert cell below	menambah cell dibawah cell aktif
	cut selected cell	memotong cell aktif

Tabel 1.5. Toolbar di Jupyter Notebook

Simbol	Keterangan	Deskripsi
	copy selected cell	menyalin cell aktif
	paste cell below	menempel ke cell dibawah cell aktif
	move selected cell up	memindah cell aktif ke atas
	move selected cell down	memindah cell aktif ke bawah
	Run	menjalankan cell aktif
	interrupt kernel	interupsi kernel
	restart the kernel	memulai ulang kernel

	restart the kernel and re-run the whole notebook	memulai ulang kernel dan menjalankan seluruh kode notebook
	cell type	Tipe cell (code / markdown)
	command palette	shortcuts perintah

1.5. Simulasi Hands on Jupyter Notebook

Kita sudah belajar fitur Jupyter Notebook selanjutnya kita akan mencoba menggunakan Jupyter Notebook untuk ngoding python

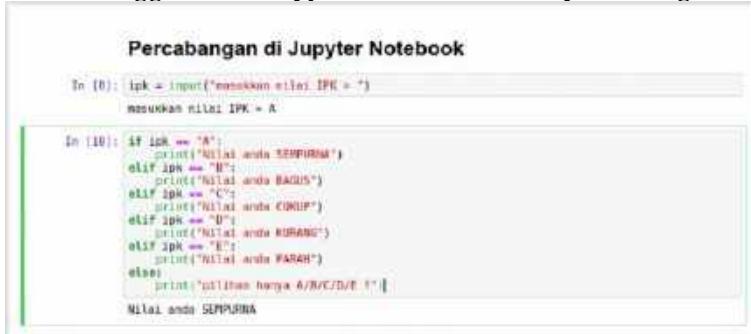
Case 1 : Menggunakan Jupyter Notebook untuk aritmatika dasar



The screenshot shows a Jupyter Notebook interface with two code cells. The first cell (In [1]) contains the assignment statements `a = 10` and `b = 5`. The second cell (In [2]) contains the code `print(a+b)`, `print(a-b)`, `print(a*b)`, and `print(a/b)`. The output of the second cell is `15`, `5`, `50`, and `2.0` respectively.

Gambar 1.14. Operasi aritmatika di Jupyter Notebook

Case 2 : Menggunakan Jupyter Notebook untuk percabangan



The screenshot shows a Jupyter Notebook interface with two code cells. The first cell (In [8]) contains the code `ipk = input("masukkan nilai IPK = ")` followed by the text "masukkan nilai IPK = A". The second cell (In [10]) contains a conditional statement that prints the grade based on the input IPK value. The output shows the input "Nilai anda SEMPURNA" and the printed grade "Nilai anda SEMPURNA".

```

In [8]: ipk = input("masukkan nilai IPK = ")
masukkan nilai IPK = A

In [10]: if ipk == "A":
    print("Nilai anda SEMPURNA")
elif ipk == "B+":
    print("Nilai anda BAGUS")
elif ipk == "C+":
    print("Nilai anda CDRUP")
elif ipk == "D":
    print("Nilai anda KURANG")
elif ipk == "E":
    print("Nilai anda #ARAH")
else:
    print("nilai anda hanya A/B/C/D/E !")
Nilai anda SEMPURNA

```

Gambar 1.15. Percabangan di Jupyter Notebook

Case 3 : Struktur Dictionary di Jupyter Notebook



The screenshot shows a Jupyter Notebook cell with two lines of Python code. The first line defines a dictionary named 'student' with various key-value pairs. The second line prints the 'student' dictionary, showing its contents.

```
In [13]: student = {  
    "name": "Taqiyyah",  
    "age": 26,  
    "tinggi": 177.6,  
    "hobi": ["olahraga", "jalan-jalan"],  
    "kontak": {  
        "website": "ngodingdata.com",  
        "email": "ngodingdata@gmail.com"  
    }  
}  
  
In [14]: print(student)  
{'name': 'Taqiyyah', 'age': 26, 'tinggi': 177.6, 'hobi': ['olahraga', 'jalan-jalan'], 'kontak': {'website': 'ngodingdata.com', 'email': 'ngodingdata@gmail.com'}}
```

Gambar 1.16. Dictionary di Jupyter Notebook

Jadi Jupyter Notebook merupakan tool interaktif untuk melakukan coding python khususnya untuk analisis data. Jupyter Notebook juga sangat fleksibel dan mudah digunakan.

Referensi

- Herho, Sandy H.S. 2017. *Tutorial Pemrograman Python 2 Untuk Pemula*. Program Studi Meteorologi, Fakultas Ilmu dan Teknologi Kebumian, Institut Teknologi Bandung. WCPL Press, Bandung.
- Pythonindo. (2020, 22 Juni). Sintaks Dasar Python. Diakses pada 22 Juni 2020, dari <https://www.pythonindo.com/sintaks-dasar-python/>
- Schneider, David I. 2016. *An Introduction to Programming Using Python Global Edition*. University of Maryland. British Library Cataloguing-in-Publication Data Pearson Education Limited
- Sianipar, RH. & Wadi, Hamzan. 2015. *Pemrograman Python: Teori dan Implementasi*. Informatika. Bandung.
- Supardi, Yuniar. 2017. *Semua Bisa Menjadi Programmer Python Basic*. PT. Elex Media Komputindo. Jakarta

BAB 2. BILANGAN

2.1. Bilangan (Number)

Dalam bahasa python pada umumnya untuk mengkategorikan sebuah angka terdapat 2 jenis. Disimbolkan sebagai **int** (kependekan dari integer) adalah jenis angka atau bilangan bulat. Sedangkan untuk bilangan pecahan atau desimal disimbolkan sebagai **float** (kependekan dari floating-point). Selain kedua jenis angka pada python, terdapat jenis angka lain yang disebut sebagai angka atau bilangan kompleks, yaitu suatu bilangan yang termasuk bilangan imajiner atau real.

Tabel 2.1. Tipe Angka

Angka	Type	Angka	Type
17	Int	77.77	Float
17.	float	-77	int

2.2. Operator Aritmatika

Lima operasi aritmatika dasar adalah penjumlahan, pengurangan, perkalian, pembagian, dan eksponensial. dalam Python operasi penambahan, pengurangan, dan operator lain dinotasikan dengan simbol standar (+), (-), dan (/), masing-masing. Namun, notasi untuk operator perkalian dan eksponensial berbeda dari notasi matematika biasa.

Tabel 2.2 Operator Aritmatika

Operator	Deskripsi	Contoh
+	Penjumlahan	$3 + 4$ bernilai 7
-	Pengurangan	$8 - 1$ bernilai 7
*	Perkalian	$1 * 7$ bernilai 7
/	Pembagian	$7 / 1$ bernilai 7

Hasil pembagian selalu berupa **float**, bahkan jika hasil bagi mengevaluasi menjadi bilangan bulat. Hasil dari operasi lain adalah **float** jika salah satu angka adalah **float** dan sebaliknya adalah **int**.

2.3. Fungsi *print*

Fungsi *print* digunakan untuk menampilkan angka pada monitor. Jika n adalah angka, maka pernyataan **print(n)** menampilkan nomor n. Kombinasi angka, operator aritmatika, dan tanda kurung yang dapat dievaluasi disebut ekspresi numerik. Fungsi *print* yang diterapkan pada ekspresi menampilkan hasil evaluasi ekspresi. Fungsi *print* tunggal dapat menampilkan beberapa nilai. Jika m, n, r, ... adalah angka (atau ekspresi numerik), kemudian pernyataan **print(m, n, r, ...)** menampilkan angka (atau nilai ekspresi numerik) satu demi satu yang dipisahkan oleh spasi. Fungsi *print* memanggil operasi baris baru yang menyebabkan fungsi cetak berikutnya untuk menampilkan hasilnya di awal baris baru.

Contoh 2.1

Buatlah sebuah program masing-masing dari lima operasi aritmatika standar menggunakan fungsi *print*. Kita mengetahui bahwa operator matematika telah dijelaskan pada Tabel 2.2 Secara sederhana programnya adalah sebagai berikut:

- Program Penjumlahan

```
print(3+4)  
[Run] 7
```

- Program Pengurangan

```
print(9-2)  
[Run] 7
```

- Program Perkalian

```
print(1*7)  
[Run] 7
```

- Program Pembagian

```
print(77/11)  
[Run] 7
```

- Program Perpangkatan

```
print(8**2)
[Run] 64
print(2* (3+4) )
[Run] 14
```

2.4. Variabel

Variabel adalah objek atau tempat yang digunakan untuk menyimpan nilai pada python yang berfungsi sebagai lokasi memori. Secara praktiknya, pada saat akan membuat sebuah variabel, maka hal yang harus dilakukan adalah menentukan lokasi untuk menyimpan data atau objek yang akan disimpan. Data atau objek yang dapat disimpan adalah bilangan, karakter (string), dan jenis objek lainnya.

Dalam python, variabel tidak perlu dideklarasikan, karena variabel dapat langsung secara otomatis terbentuk ketika user menugaskan atau menyimpan suatu nilai kedalam variabel. Hal ini biasanya dilakukan dengan memberikan tanda sama dengan (=) yang berfungsi untuk memberikan nilai ke variabel. Secara teknisnya, bagian kiri dari tanda = adalah penamaan atau nama variabel yang disebut sebagai lokasi memori untuk menyimpan data atau objek, sedangkan bagian kanan tanda = adalah nilai yang disimpan di dalam variabel. Sebagai contoh:

Contoh 2.2

Buatlah sebuah program untuk membuat suatu variabel yang berisi data/objek baik itu bilangan bulat (integer), pecahan (float), karakter (string), dan lain – lain serta hasilnya dikeluarkan menggunakan fungsi **print**. Secara sederhana programnya adalah sebagai berikut:

- Program Membuat Variabel untuk tipe data integer

Program berikut menghasilkan output 60. Hasil ini merupakan hasil dari objek/variabel “Berat”. variabel “Berat” berisikan data integer yaitu 60

```
Berat = 60
print(Berat)
[Run] 60
```

- Program Membuat Variabel untuk tipe data float

Program berikut menghasilkan output 170,5 hasil ini merupakan hasil dari objek/variabel “Tinggi”. Variabel “Tinggi” berisikan data float yaitu 170,5

```
Tinggi = 170.5
print(Tinggi)
[Run] 170.5
```

- Program Membuat Variabel untuk tipe data string

Program berikut menghasilkan output Hizam. Hasil ini merupakan hasil dari objek/variabel “Nama”. Variabel “Nama” berisikan data string yaitu Hizam.

```
Nama = 'Hizam'
print(Nama)
[Run] Hizam
```

2.5. Fungsi abs, int, dan round

Ada beberapa operasi umum yang dapat dilakukan pada angka selain operasi aritmatika standar. Misalnya, kita dapat membulatkan angka atau mengambil nilai absolutnya. Operasi ini dilakukan oleh fungsi bawaan. Fungsi mengaitkan dengan satu atau lebih nilai, disebut input, nilai tunggal yang disebut output. Fungsi tersebut dikatakan mengembalikan nilai output. Tiga fungsi yang dipertimbangkan dalam paragraf berikutnya memiliki input dan output numerik.

Tabel 2.3. Operator abs, int, dan round

Input	Output	Input	Output
-------	--------	-------	--------

abs(7)	7	int(-7.7)	-7
abs(0)	0	round(7.7)	8
abs(-7)	7	round(7.7177, 2)	7.72
int(7.7)	7	round(7.7177, 1)	7.7
int(7)	7		

Istilah di dalam tanda kurung dapat berupa angka (seperti yang ditunjukkan pada Tabel 2.3), variabel numerik, atau ekspresi numerik. Ekspresi dievaluasi untuk menghasilkan input.

Contoh 2.3

Buatlah sebuah program untuk mengevaluasi masing-masing dari tiga fungsi **abs**, **int**, dan **round** serta hasilnya dikeluarkan menggunakan fungsi **print**. Secara sederhana programnya adalah sebagai berikut:

- Program penggunaan fungsi **abs**

```
a = 7
b = 2
print(abs(1-(4*b)))
[Run] 7
```

Program tersebut menghasilkan output 7. Hasil ini merupakan hasil operasi matematika dari objek/variabel “a” dan “b”. Fungsi **abs** pada program tersebut berfungsi untuk mengabsolutkan hasil dari operasi matematika. Jika fungsi **abs** tidak digunakan, maka hasil operasi matematika dari program tersebut adalah -7

- Program penggunaan fungsi **int**

```
a = 7
b = 2
print(int((a**b)+0.8))
[Run] 49
```

Program tersebut menghasilkan output 49. Hasil ini merupakan hasil operasi matematika dari objek/variabel “a” dan “b”. Fungsi **int**

pada program tersebut berfungsi untuk pembulatan hasil dari operasi matematika. Pembulatan untuk fungsi *int* merupakan fungsi pembulatan kebawah. Jika fungsi *int* tidak digunakan, maka hasil operasi matematika dari program tersebut adalah 49.8

- Program penggunaan fungsi *round*

```
a = 7  
b = 2  
print(round((a/b)))  
[Run] 4
```

Program tersebut menghasilkan output 4. Hasil ini merupakan hasil operasi matematika dari objek/variabel “a” dan “b”. Fungsi *round* pada program tersebut berfungsi untuk pembulatan hasil dari operasi matematika. Pembulatan untuk fungsi *round* merupakan fungsi pembulatan keatas. Jika fungsi *round* tidak digunakan, maka hasil operasi matematika dari program tersebut adalah 3.5

2.6. Operator Penugasan

Karena ekspresi di sisi kanan pernyataan penugasan dievaluasi sebelum penugasan dibuat, pernyataan seperti **var** = **var** + 1 bermakna. Pertama-tama mengevaluasi ekspresi di sebelah kanan (yaitu, ia menambahkan 1 ke nilai variabel **var**) dan kemudian menetapkan jumlah ini ke variabel **var**. Efeknya adalah untuk meningkatkan nilai variabel **var** oleh 1. Dalam hal lokasi memori, pernyataan mengambil nilai **var** dari lokasi memori **var**, menggunakan untuk menghitung **var** + 1, dan kemudian menempatkan jumlah ke lokasi memori. Jenis perhitungan ini sangat umum sehingga Python menyediakan operator khusus untuk melaksanakannya. Pernyataan **var** = **var** + 1 dapat diubah dengan pernyataan yang memiliki makna sama yaitu **var** += 1 Secara umum, jika n memiliki nilai numerik, maka pernyataan **var** += n menambahkan nilai n ke nilai var. Operator += dikatakan melakukan tugas yang ditambah. Beberapa operator penugasan tambahan lainnya adalah -=, *=, /=, dan **=.

Contoh 2.4

Buatlah sebuah program untuk mengevaluasi operator penugasan serta hasilnya dikeluarkan menggunakan fungsi *print*. Secara sederhana programnya adalah sebagai berikut:

- Program penggunaan operator penugasan `+=`

```
num1 = 6
num1 += 1
print(num1)
[Run] 7
```

Program tersebut menghasilkan output 7. Penggunaan operator `+=` bertujuan untuk melakukan operasi matematika yaitu menjumlahkan nilai pada variabel num1 dengan angka 1.

- Program penggunaan operator penugasan `-=`

```
num2 = 7
num2 -= 5
print(num2)
[Run] 2
```

Program tersebut menghasilkan output 2. Penggunaan operator `-=` bertujuan untuk melakukan operasi matematika yaitu mengurangkan nilai pada variabel num2 dengan angka 5.

- Program penggunaan operator penugasan `/=`

```
num3 = 8
num3 /= 2
print(num3)
[Run] 4.0
```

Program tersebut menghasilkan output 4.0. Penggunaan operator `/=` bertujuan untuk melakukan operasi matematika yaitu membagi nilai pada variabel num3 dengan angka 2.

- Program penggunaan operator penugasan `**=`

```
num4 = 8
num4 **= 2
print(num4)
[Run] 64
```

Program tersebut menghasilkan output 64. Penggunaan operator `**=` bertujuan untuk melakukan operasi matematika yaitu perpangkatan nilai pada variabel num4 dengan angka 2.

2.7. Dua Operator Integer Lainnya

Selain lima operator aritmatika standar yang dibahas pada awal modul ini, operator divisi integer (tertulis `//`) dan operator modulus (tertulis `%`) juga tersedia dalam Python. Misalkan m dan n adalah sebuah bilangan bulat positif. Ketika Anda menggunakan pembagian panjang untuk membagi m dengan n , Anda mendapatkan hasil bagi bilangan bulat dan sisa bilangan bulat. Dalam Python, hasil bagi bilangan bulat dilambangkan $m // n$, dan sisa bilangan bulat dilambangkan $m \% n$. Misalnya,

Tabel 2.4. Dua Operator Integer Lainnya

Operator	Deskripsi	Contoh
<code>//</code>	Pembagian (dibulatkan kebawah)	$15 // 2$ bernilai 7
<code>%</code>	Sisa Bagi / Modulo	$13 \% 5$ bernilai 3

Contoh 2.5

Buatlah program untuk mengonversi 41 inci menjadi 3 kaki dan 5 inci. Secara sederhana program tersebut dapat disusun senagai berikut.

```
totalinc = 41
feet = totalinc//12
inc = totalinc % 12
print (feet,inc)
[Run] 3 5
```

2.8. Tanda kurung, Urutan diutamakan

Tanda kurung harus digunakan untuk mengklarifikasi arti suatu ungkapan. Ketika ada tanda kurung tidak mencukupi, operasi aritmatika dilakukan dalam urutan prioritas sebagai berikut:

1. istilah di dalam tanda kurung (dalam ke luar)
2. eksponensial
3. perkalian, pembagian (biasa dan bilangan bulat), modulus
4. penambahan dan pengurangan.

Jika terjadi seri, operasi paling kiri dilakukan terlebih dahulu. Misalnya, $8/2 * 3$ dievaluasi sebagai $(8/2) * 3$. Praktik pemrograman yang baik adalah menggunakan tanda kurung secara bebas sehingga Anda tidak perlu mengingat urutan prioritasnya. Misalnya, tulis $(2 * 3) + 4$ bukannya $2 * 3 + 4$ dan tulis $4 + (2 ** 3)$ alih-alih $4 + 2 ** 3$.

2.9. Tiga Jenis Kesalahan

Kesalahan tata bahasa dan tanda baca disebut kesalahan sintaksis. Beberapa pernyataan yang salah dan kesalahannya ditunjukkan pada Tabel 2.5.

Tabel 2.5. Jenis Kesalahan

Statement	Deskripsi
print(3))	Pernyataan itu berisi tanda kurung luar yang asing.
for = 5	Kata yang dicadangkan digunakan sebagai nama variabel.
print(2; 3)	Tanda titik koma harus berupa koma.

Kesalahan yang ditemukan saat program sedang berjalan disebut kesalahan runtime atau pengecualian. Beberapa pernyataan yang salah dan kesalahannya ditunjukkan pada Tabel 2.6.

Tabel 2.6. Jenis Kesalahan Runtime

Statement	Deskripsi
prmt(5)	Fungsi cetak salah eja
x += 1, when x has not been created	Python tidak mengetahui variabel x.

print(5 / 0)	Angka tidak dapat dibagi dengan nol.
--------------	--------------------------------------

Latihan Soal

Dalam latihan berikut tentukan output dan buatlah program yang sesuai berdasarkan perintah yang telah ditentukan

- Evaluasi hasil program berikut

a. $11-4\%9$	d. $10 ** 2$
b. $5\%5$	e. $9 \%1$
c. $7 // 4$	f. $(7 - 4) / 3$
- Eksekusi Aritmatika berikut.

a. $5 ** 10$	d. $100\%5$
b. $3 + (4 * 5)$	e. $7/4$
c. $(5 + 3) // 10$	f. $(2^3 - 1) + 5$
- Jelaskan perbedaan antara pernyataan penugasan **var1 = var2** dan pernyataan penugasan **var2 = var1**
- Lengkapi tabel dengan mengisi nilai setiap variabel setelah setiap baris kode dieksekusi.

	a	B
a = 3		
b = a + 5		
a = a ** b		
print((a/b) + 2)		
b = b % 2 + 0.7		

- Lengkapi tabel dengan mengisi nilai setiap variabel setelah setiap baris kode dieksekusi.

	Bal	Inter	WithDr
bal = 100			

inter = .05			
withDr = 25			
bal += (inter * bal)			
bal = bal - withDr			

- | | |
|---|--|
| 6. a = 7
b = 4
print(a * b ** 2) | 7. n = 6
n** = 4
print(n/5) |
| 8. totalBerries = 1000
totalCost = 450
eachBerry = totalCost/
totalBerries
print(eachBerry) | 9. points = 70
points += 70 * 10
print(points) |

Jabarkan kesalahan dari program berikut

10. $0.05 = \text{interest}$
 $\text{balance} = 800$
 $\text{print}(\text{interest} * \text{balance})$
Jabarkan kesalahannya

11. $a = 2$
 $b = 3$
 $a + b = c$
 $\text{print}(b)$

Tentukan hasil dari program berikut

- | | |
|--|--|
| 12. <code>int(7.75)</code> | 13. <code>int(9 - 2)</code> |
| 14. <code>round(-7.5)</code> | 15. <code>round(7.1254, 3)</code> |
| 16. <code>int(b * 0.5)</code> jika $b = 8$ | 17. <code>abs(a - 5)</code> jika $a = 7$ |

18. **Hitung Keuntungan.** Langkah-langkah berikut menghitung laba perusahaan.

 - (a) Buat pendapatan variabel dan berikan nilai 100.456.
 - (b) Buat biaya variabel dan berikan nilai 55.500.
 - (c) Buat variabel laba dan tetapkan perbedaan antara nilai-nilai variabel pendapatan dan biaya.
 - (d) Menampilkan nilai laba variabel.

19. **Keuntungan dari Saham.** Langkah-langkah berikut menghitung persentase laba dari penjualan saham.

 - (a) Buat variabel Harga Pembelian dan berikan nilai 20.

- (b) Buat variabel Harga Penjualan dan berikan nilai 30.
 - (c) Buat variabel Persentase Profit dan tetapkan 100 kali dari nilai selisih antara harga jual dan harga beli dibagi dengan harga pembelian.
 - (d) Menampilkan nilai variabel Profit variabel.
20. **Savings Account.** Langkah-langkah berikut menghitung saldo pada akhir tiga tahun ketika \$100 disetor pada awal setiap tahun dalam rekening tabungan dengan bunga 5% ditambah setiap tahun.
- (a) Buat saldo variabel dan berikan nilai 100.
 - (b) Tingkatkan nilai saldo variabel sebesar 5%, dan tambahkan 100 untuk itu.
 - (c) Tingkatkan nilai saldo variabel sebesar 5%, dan tambahkan 100 ke dalamnya.
 - (d) Meningkatkan nilai saldo variabel sebesar 5%.
 - (e) Menampilkan nilai keseimbangan (dibulatkan menjadi dua tempat desimal).

Referensi

- Pythonindo. (2020, 25 Juni). Variabel dan Tipe Data Python. Diakses pada 25 Juni 2020, dari <https://www.pythonindo.com/variabel-dan-tipe-data-python/>
- Schneider, David I. 2016. *An Introduction to Programming Using Python Global Edition*. University of Maryland. British Library Cataloguing-in-Publication Data Pearson Education Limited

BAB 3. STRING

3.1. Pengertian String

Selain kemampuan bekerja pada tipe data berupa angka, python juga dapat mengeksekusi tipe data bukan angka atau yang disebut sebagai tipe data string. String dalam program python disimbolkan dengan ‘*str*’. String adalah urutan karakter yang diperlakukan sebagai satu item. Karakter dalam string dapat berupa karakter apa pun yang ditemukan pada keyboard (seperti huruf, angka, tanda baca, dan spasi) dan banyak karakter khusus lainnya. Dalam program Python, string ditulis sebagai urutan karakter yang dikelilingi oleh tanda kutip tunggal (') atau tanda kutip ganda ("). Beberapa contoh string adalah sebagai berikut:

"Statistika Bisnis"

'Soekarno'

'59'

Tanda kutip pembukaan dan penutupan harus bertipe sama - baik tanda kutip ganda atau keduanya tanda kutip tunggal. Ketika sebuah string dikelilingi oleh tanda kutip ganda, sebuah kutipan tunggal dapat muncul secara langsung dalam string, tetapi bukan kutipan ganda. Demikian pula, sebuah string yang dikelilingi oleh tanda kutip tunggal dapat berisi penawaran ganda, tetapi bukan kutipan tunggal secara langsung.

3.2. Variabel pada String

Seperti pembahasan pada bab sebelumnya, jika variabel dalam python juga dapat berisi data atau objek dengan tipe data selain numeric yaitu tipe data string. Seperti halnya variabel yang diberi nilai numerik, variabel yang diberi nilai string dibuat (yaitu, muncul pertama kali mereka muncul dalam pernyataan penugasan. Ketika argumen dari fungsi *print* adalah string atau variabel yang memiliki nilai string, hanya karakter dalam tanda kutip terlampir yang

ditampilkan. Berikut merupakan contoh program menggambarkan penggunaan variabel.

```
s = 'Indonesia'  
print(s)  
[Run] Indonesia  
s = "Indonesia"  
print(s)  
[Run] Indonesia
```

3.3. Indeks dan Irisan

Dalam Python, posisi atau indeks karakter dalam string diidentifikasi dengan salah satu angka 0, 1, 2, 3, Misalnya, karakter pertama dari sebuah string dikatakan memiliki indeks 0, karakter kedua dikatakan memiliki indeks 1, dan seterusnya. Jika str1 adalah variabel string atau literal, maka str1 [i] adalah karakter string yang memiliki indeks i. Gambar 3.1 menunjukkan indeks karakter string "NUSA & BANGSA".

N	U	S	A		&		B	A	N	G	S	A
0	1	2	3	4	5	6	7	8	9	10	11	12

Gambar 3.1. Indeks karakter string "NUSA & BANGSA".

Substring atau irisan string adalah urutan karakter berturut-turut dari string. Jika str1 adalah string, maka str1 [m: n] adalah substring yang dimulai pada posisi m dan berakhir pada posisi n-1. Gambar 3.2 membantu memvisualisasikan irisan. Pikirkan indeks karakter yang menunjuk tepat ke kiri karakter. Maka "NUSA & BANGSA" [m: n] adalah urutan karakter antara panah yang diberi label dengan angka m dan n. Misalnya "NUSA & BANGSA" [2: 6] adalah substring "SA &"; yaitu, substring antara panah berlabel 2 dan panah berlabel 6

N	U	S	A		&		B	A	N	G	S	A
0	1	2	3	4	5	6	7	8	9	10	11	12

Gambar 3.2 Visualisasi irisan

Catatan: Jika $m \geq n$, yaitu, jika karakter di posisi m tidak di sebelah kiri karakter di posisi n , maka nilai $\text{str1}[m: n]$ akan menjadi string kosong (""), string tanpa karakter.

Jika subStr adalah string, maka $\text{str1}.find(\text{subStr})$ adalah indeks positif dari penampilan pertama subStr di str1 dengan pencarian yang dimulai di sisi kiri string. Nilai $\text{str1}.rfind(\text{subStr})$ adalah indeks positif dari penampilan pertama subStr di str1 dengan pencarian yang dimulai di sisi kanan string. Jika subStr tidak muncul di str1 , maka nilai yang dikembalikan oleh metode `find` and `rfind` akan menjadi -1 .

Contoh 3.1

Program berikut menggambarkan penggunaan indeks.

```
print("Indonesia"[1], "Indonesia"[5], "Indoneisa  
"[2:4])  
[Run] n e do
```

Program tersebut menghasilkan output `n e do`. Penjelasannya adalah sebagai berikut.

`"Indonesia"[1]` adalah program untuk mengeluarkan karakter pada indeks ke 1 pada kata `Indonesia`

`"Indonesia"[5]` adalah program untuk mengeluarkan karakter pada indeks ke 5 pada kata `Indonesia`

`"Indonesia"[2:4]` adalah program untuk mengeluarkan karakter pada indeks ke 2 sampai 3 pada kata `Indonesia`.

Contoh 3.2

Program berikut menggambarkan lokasi indeks suatu karakter

```
str1 = "Hello World!"  
print(str1.find('W'))  
[Run] 6
```

Program tersebut menghasilkan output `6`, artinya karakter huruf `W` pada kata `Hello World` berada pada indeks ke-6

3.4. Indeks Negatif

Indeks yang dibahas di atas menentukan posisi dari sisi kiri string. Python juga memungkinkan string untuk diindeks berdasarkan posisi mereka sehubungan dengan sisi kanan string dengan menggunakan angka negatif untuk indeks. Dengan pengindeksan negatif, karakter paling kanan diberi indeks -1, karakter di sebelah kirinya diberi indeks -2, dan seterusnya. Gambar 3.3 menunjukkan indeks negatif dari karakter string "NUSA & BANGSA".

N	U	S	A		&		B	A	N	G	S	A
13	12	11	10	9	8	7	6	5	4	3	2	1

Gambar 3.3. Indeks negatif dari karakter string "NUSA & BANGSA"

Contoh 3.3

Program berikut menggambarkan penggunaan indeks.

```
print("Python"[-1], "Python"[-4], "Python"[-5:-2])  
[Run] n t yth
```

Program tersebut menghasilkan output n t yth. Penjelasannya adalah sebagai berikut.

"Python"[-1] adalah program untuk mengeluarkan karakter pada indeks ke -1 pada kata Python

"Python"[-4] adalah program untuk mengeluarkan karakter pada indeks ke -4 pada kata Python

"Python"[-5:-2] adalah program untuk mengeluarkan karakter pada indeks ke -5 sampai -2 pada kata Python

3.5. Batas Default untuk Irisan

Dalam ekspresi str1 [m: n], satu atau kedua batas dapat dihilangkan. Jika demikian, batas kiri m default ke 0 dan batas kanan n default ke panjang string. Artinya, str1 [: n] terdiri dari semua

karakter dari awal string ke str1 [n-1], dan str1 [m:] terdiri dari semua karakter dari str1 [m] hingga akhir string. Slice str1 [:] adalah seluruh string str1

Contoh 3.4. Program berikut menggambarkan batasan default

```
print("Python"[-3:], "Python)[:3])  
[Run] hon Pyt
```

3.6. Penggabungan string

Dua string dapat digabungkan untuk membentuk string baru yang terdiri dari string yang disatukan. Operasi ini disebut **penggabungan** dan diwakili oleh tanda plus. Misalnya, "Statistika " + "Bisnis" adalah 'Statistika Bisnis'. Kombinasi string, tanda plus (+), fungsi, dan metode yang dapat dievaluasi untuk membentuk string disebut **ekspresi string**. Ketika ekspresi string muncul dalam pernyataan penugasan atau fungsi **print**, ekspresi string dievaluasi sebelum ditugaskan atau ditampilkan.

Contoh 3.5. Program menggambarkan penggabungan string

```
"Statistika " + "Bisnis"  
[Run] 'Statistika Bisnis'
```

3.7. Pengulangan String

Operator tanda bintang (*) dapat digunakan dengan string untuk berulang kali menyatukan string dengan dirinya sendiri. Jika str1 adalah string, variabel, atau ekspresi dan n adalah bilangan bulat positif, maka nilai **str1 * n** adalah gabungan n salinan dari nilai str1.

Contoh 3.6. Program menggambarkan pengulangan string

```
"Hidup ITS " * 3  
[Run] 'Hidup ITS Hidup ITS Hidup ITS '
```

3.8. Fungsi dan Metode String

Fungsi string beroperasi seperti fungsi numerik; dibutuhkan string sebagai input dan mengembalikan nilai. Metode string adalah

proses yang melakukan tugas pada string. Kita telah melihat dua contoh metode **find** dan **rfind**. Metode-metode ini melakukan tugas mencari indeks. Bentuk umum dari ekspresi yang menerapkan metode adalah

```
stringName.methodName ()
```

di mana tanda kurung mungkin mengandung nilai. Seperti fungsi numerik yang dibahas di bagian sebelumnya, fungsi dan metode string juga dapat diterapkan ke literal, variabel, dan ekspresi. Tabel 1 di bawah ini menjelaskan satu fungsi string dan enam metode string tambahan di mana str1 adalah string "Python". Beberapa metode string lebih lanjut akan disajikan dalam bab-bab selanjutnya.

Tabel 3.1. String operations (str1 = "Python").

Fungsi	Contoh	Output	Deskripsi
Len	len(str1)	6	Jumlah karakter dalam string
Upper	str1.upper()	"PYTHON"	Huruf besar setiap karakter alphabet
Lower	str1.lower()	"python"	Huruf kecil setiap karakter alphabet
Count	str1.count('th')	1	Jumlah kemunculan substring yang tidak tumpang tindih
capitalize	"coDE".capitalize()	"Code"	Mengkapitalisasi huruf pertama dari string dan menurunkan sisanya
Title	"staTistika bisnis".title()	"Statistika Bisnis"	Menggunakan huruf kapital dari huruf pertama dari setiap kata dalam string dan menurunkan sisanya
Tstrip	"ab ".rstrip()	"ab"	Menghilangkan spasi dari sisi kanan string

3.9. Metode Dirantai

Pertimbangkan dua baris kode berikut:

```
pujian = "Good Grade". upper ()  
numberOfGees = pujian.count ('G')
```

Kedua garis ini dapat digabungkan menjadi satu garis di bawah ini yang dikatakan **menghubungkan** kedua metode tersebut.

```
numberOfGees = "Good Grade" .upper () . count  
('G')
```

Metode berantai dieksekusi dari kiri ke kanan. Chaining sering menghasilkan kode yang lebih jelas karena menghilangkan variabel sementara, seperti variabel pujian di atas.

3.10. Fungsi input

Fungsi input meminta pengguna untuk memasukkan data. Pernyataan input tipikal adalah

```
kota = input ("Masukkan nama kota Anda:")
```

Ketika Python mencapai pernyataan ini, string "Masukkan nama kota Anda:" ditampilkan dan program berhenti. Setelah pengguna mengetik nama kotanya dan menekan tombol Enter (atau kembali), kota variabel ditetapkan nama kotanya. (Jika variabel belum dibuat sebelumnya, itu dibuat saat ini.) Bentuk umum dari pernyataan input adalah

```
variableName = input (prompt)
```

di mana prompt adalah string yang meminta respons dari pengguna.

Contoh 3.5

Berikut program meminta nama dari pengguna dan kemudian mem-parsing nama. Ketika program dijalankan, frasa "Masukkan nama lengkap:" muncul dan eksekusi program berhenti. Setelah pengguna mengetik kata-kata yang ditampilkan dalam warna hitam dan menekan tombol Enter (atau kembali), dua baris terakhir dari output ditampilkan.

```
fullName = input("Enter a full name: ")
n = fullName.rfind(" ")
print("Last name:", fullName[n+1:])
print("First name(s):", fullName[:n])
[Run] Enter a full name: BJ Habibie
      Last name: Habibie
      First name(s): BJ
```

3.11. Fungsi int, float, eval, dan str

Jika str1 adalah string yang berisi bilangan bulat, fungsi **int** akan mengubah string menjadi integer. Jika str1 adalah string yang berisi angka apa saja, fungsi **float** akan mengubah string menjadi angka titik-mengambang. (Fungsi **float** juga mengubah **integer** ke angka floating-point.) Jika str1 adalah string yang terdiri dari ekspresi numerik, fungsi **eval** akan mengevaluasi ekspresi menjadi integer atau angka floating-point yang sesuai.

Fungsi input selalu mengembalikan string. Namun, kombinasi fungsi input dan fungsi int, float, atau eval memungkinkan angka untuk dimasukkan ke dalam program. Misalnya, pertimbangkan tiga pernyataan berikut:

```
age = int (input ("Masukkan umur Anda:"))
age = float (input ("Masukkan umur Anda:"))
age = eval (input ("Masukkan umur Anda:"))
```

Misalkan pengguna merespons dengan bilangan bulat, katakanlah 25. Kemudian, setelah masing-masing pernyataan di atas telah ditanggapi, **print(age)** akan menampilkan 25, 25.0, dan 25, masing-masing. Namun, jika pengguna masih muda, ia mungkin merespons dengan

nomor 3 dari contoh 5. Dengan pernyataan input pertama, pesan kesalahan akan muncul. Setelah pernyataan input kedua atau ketiga dijalankan, fungsi cetak akan menampilkan nomor 3 dari contoh 5. Fungsi **eval** menghasilkan hasil yang baik dengan usia baik.

Fungsi **int** dan **float** mengeksekusi lebih cepat daripada fungsi **eval** dan lebih disukai oleh banyak programmer Python ketika mereka dapat digunakan dengan aman. Dalam buku ini kita akan menggunakan ketiga fungsi, tetapi akan mendukung fungsi **eval**.

Fungsi **int** dan **float** juga dapat diterapkan pada ekspresi numerik yang sesuai. Jika x adalah bilangan bulat, nilai **int** (x) adalah x. Jika x adalah angka titik-mengambang, fungsi **int** menghapus bagian desimal dari angka tersebut. Fungsi **float** beroperasi seperti yang diharapkan. Fungsi **eval** tidak dapat diterapkan pada literal numerik, variabel, atau ekspresi.

Tabel 5.2. Contoh Penerapan Fungsi **int** dan **float**

Contoh	Output	Contoh	Output
<code>int(4.8)</code>	4	<code>float(4.67)</code>	4.67
<code>int(-4.8)</code>	4	<code>float(-4)</code>	-4.0
<code>int(4)</code>	4	<code>float(0)</code>	0.0

Fungsi **str** mengubah angka menjadi representasi stringnya. Misalnya, nilai **str** (5.6) adalah "5.6" dan nilai **str** (5.) adalah "5.0". String tidak dapat digabungkan dengan angka. Namun, pernyataan itu tidak valid

```
strVar = numVar + '%'
```

dapat diganti dengan pernyataan yang valid

```
strVar = str (numVar) + '%'
```

yang menggabungkan dua string.

3.12. Dokumentasi Internal

Dokumentasi program adalah penyertaan komentar yang menentukan maksud program, tujuan variabel, dan tugas yang dilakukan oleh masing-masing bagian program. Untuk membuat pernyataan komentar, mulailah baris dengan tanda angka (#). Pernyataan seperti itu sepenuhnya diabaikan ketika program dijalankan. Komentar terkadang disebut komentar. Baris kode dapat didokumentasikan dengan menambahkan tanda angka.

Contoh 3.6

Penulisan ulang Contoh 3.5 berikut ini menggunakan dokumentasi. Komentar pertama menjelaskan keseluruhan program, komentar di baris ketiga memberikan arti dari sebuah variabel, dan komentar terakhir menjelaskan tujuan dari dua baris yang mengikutinya.

```
## Pisahkan nama menjadi dua bagian - nama
belakang dan nama depan.
fullName = input("Enter a full name: ")
n = fullName.rfind(" ") # indeks ruang sebelum
nama belakang
# Tampilkan informasi yang diinginkan.
print("Last name:", fullName[n+1:])
print("First name(s):", fullName[:n])
[Run] Enter a full name: BJ Habibie
      Last name: Habibie
      First name(s): BJ
```

informasi yang diinginkan, setelah akhir baris. Menekan Alt + 3 dan Alt + 4 dapat digunakan di IDLE untuk mengomentari dan menghapus komentar pada blok kode yang dipilih. Beberapa manfaat dokumentasi adalah sebagai berikut:

1. Orang lain dapat dengan mudah memahami program ini.
2. Anda dapat lebih memahami program ketika Anda membacanya nanti.
3. Program yang panjang lebih mudah dibaca karena tujuan masing-masing bagian dapat ditentukan secara sekilas.

Praktik pemrograman yang baik mengharuskan pemrogram mendokumentasikan kode mereka saat mereka menulisnya. Faktanya, banyak perusahaan perangkat lunak memerlukan tingkat dokumentasi tertentu sebelum mereka merilis perangkat lunak dan beberapa menilai kinerja programmer tentang seberapa baik kode mereka didokumentasikan.

3.13. Garis Lanjutan

Pernyataan panjang dapat dibagi menjadi dua atau lebih baris dengan mengakhiri setiap baris (kecuali yang terakhir) dengan karakter garis miring terbalik (\). Misalnya, seperti contoh berikut

```
print("Well written code is its own best
documentation")
[Run] Well written code is its own best
documentation
```

dapat ditulis sebagai

```
print("Well written code is its own" +\
      " best documentation")
[Run] Well written code is its own best
documentation
```

Python memiliki fitur yang dapat digunakan untuk menghilangkan kebutuhan garis lanjutan dengan karakter backslash. Kode apa pun yang diapit sepasang kurung dapat menjangkau beberapa baris. Karena ekspresi apa pun dapat ditutup dalam tanda kurung, fitur ini hampir selalu dapat digunakan. Misalnya, pernyataan di atas dapat ditulis sebagai

```
print("Well written code is its own" +"best
documentation")
[Run] Well written code is its own best
documentation
```

Metode kelanjutan garis ini telah menjadi gaya yang disukai oleh kebanyakan programmer Python dan akan digunakan kapan saja memungkinkan dalam buku teks ini.

3.14. Mengindeks dan Mengiris Batas

Python tidak mengizinkan indeks di luar batas untuk karakter individu string, tetapi memungkinkan indeks di luar batas untuk irisan. Misalnya, jika

str1 = "Python"

kemudian **print (str1 [7])** dan **print (str1 [-7])** memicu pesan kesalahan.

Jika indeks kiri dalam slice terlalu jauh negatif, slice akan mulai pada awal string, dan jika indeks kanan terlalu besar, slice akan pergi ke akhir string. Contohnya,

str1 [-10: 10] adalah "Python"

str1 [-10: 3] adalah "Pyt"

Latihan Soal

Dalam latihan berikut tentukan output dan buatlah program yang sesuai berdasarkan perintah yang telah ditentukan

1. **print("Institut Teknologi Sepuluh Nopember")**
2. **var = "Hello"**
print(var)
3. **"Python"[4]**
4. **"Python"[2:2]**
5. **"Python".find("Pyt")**
6. **"python/java/c++".split('/')**
7. **"8 Ball".title()**
8. **"Statistika Bisnis".lower().count('a')**
9. **"vocational".capitalize()**
10. **"Python"[-1*len("Python")-1:3]**
11. **a = 4**
b = 6
c = "Municipality"
d = "pal"

- ```
print(len(c))
print(c.upper())
print(c[a:b] + c[b + 4:])
print(c.find(d))
12. print('M' + ('m' * 3) * 2 + '.')
13. Nomer Telfon = 234-5678
 print("Nomer Telfon saya adalah" + phoneNumber)
14. quote = Saya Mahasiswa Statistika Bisnis.
 print(quote + ": " + "ITS")
15. usia = 19
 print("Usia: " + usia)
16. film = "the great gatsby".title()[:10].rstrip()
 print(film, len(film))
17. Buatlah program sederhana yang memotong karakter terakhir
 dari string dan program untuk memotong karakter pertama
 dari sebuah string.
18. Tulis sebuah program yang memiliki satu baris untuk setiap
 langkah. Baris yang menampilkan data harus menggunakan
 nama variabel yang diberikan. Langkah-langkah berikut
 memberikan nama dan tahun kelahiran seorang penemu
 terkenal.
 (a) Buat variabel firstName dan berikan nilai "Thomas".
 (b) Buat variabel middleName dan berikan nilai "Alva".
 (c) Buat variabel lastName dan berikan nilai "Edison".
 (d) Buat variabel YearOfBirth dan berikan nilai 1847.
 (e) Perlihatkan frasa "Tahun kelahiran" diikuti dengan nama
 lengkap penemu, diikuti oleh "adalah", dan tahun lahir
 penemu.
19. Jika n adalah jumlah detik antara petir dan guntur, badai
 berjarak $n / 5$ mil. Tulis program yang meminta jumlah detik
 antara petir dan guntur dan melaporkan jarak dari badai
 membulat ke dua desimal.
20. Tulis program untuk meminta nama tim baseball, jumlah
 pertandingan yang dimenangkan, dan jumlah pertandingan
 yang hilang sebagai input, lalu tampilkan nama tim dan
 persentase pertandingan yang dimenangkan.
```

## **Referensi**

- Herho, Sandy H.S. 2017. *Tutorial Pemrograman Python 2 Untuk Pemula*. Program Studi Meteorologi, Fakultas Ilmu dan Teknologi Kebumian, Institut Teknologi Bandung. WCPL Press, Bandung.
- Schneider, David I. 2016. *An Introduction to Programming Using Python Global Edition*. University of Maryland. British Library Cataloguing-in-Publication Data Pearson Education Limited

## BAB 4. OUTPUT

### 4.1. Pengertian Output

Pada bahasa python, cara untuk mengeluarkan hasil dari eksekusi suatu program perintah yang digunakan adalah fungsi **print()**. Output yang ditingkatkan dapat dihasilkan oleh fungsi cetak dengan dua argumen opsional dan penggunaan metode format. Beberapa kalimat pada fungsi **print**, dapat dipisahkan dengan memasukkan kata di dalam fungsi print, setiap argumennya dipisahkan dengan tanda (,) (koma). Berikut penggunaannya dalam kode program Python.

```
print("Departemen","Statistika","Bisnis")
[Run] Departemen Statistika Bisnis
```

### 4.2. Argumen `sep=" "` dan `end=" "`

Argumen `sep=" "` digunakan untuk memisahkan antar argumen dengan string kosong atau karakter lain. Sedangkan argumen `end=" "` digunakan untuk meletakkan suatu karakter setelah satu fungsi print berakhir. Contoh penggunaannya dijabarkan dalam potongan kode berikut.

**Tabel 4.1.** Statement Argumen `sep=" "` dan `end=" "`

| Statement                                                   | Output                                |
|-------------------------------------------------------------|---------------------------------------|
| print ("Departemen","Statistika","Bisnis",<br>sep="-")      | Departemen-Statistika-<br>Bisnis      |
| print("Departemen","Statistika","Bisnis",<br>sep="")        | DepartemenStatistikaBisnis            |
| print("Statistika","Bisnis", 1 , sep=" ")                   | Statistika Bisnis 1                   |
| print("Statistika", end=" ") print("Bisnis ITS")            | Statistika Bisnis ITS                 |
| print ("Departemen","Statistika","Bisnis",<br>end=" - ITS") | Departemen Statistika Bisnis<br>- ITS |

### 4.3. Escape Character

Di modul, disebutkan “Special Character”. Sebenarnya adalah **Escape Character**. *Escape character* adalah karakter-karakter yang diawali dengan tanda \ (backslash), dimana untuk masing-masing *escape character* memiliki makna tertentu yang digunakan dalam fungsi *print*. Macam-macam *escape character* akan didefinisikan sebagai berikut.

**Tabel 4.2. Escape Character**

| Escape Character | Output                                                         |
|------------------|----------------------------------------------------------------|
| \b               | Mundur Satu Spasi                                              |
| \f               | Mengganti Halaman ( <i>from feed</i> )                         |
| \n               | Mengganti baris baru                                           |
| \r               | Ke kolom pertama ke baris yang sama ( <i>carriage return</i> ) |
| \t               | Tabulasi horizontal ( <i>tab</i> )                             |
| \v               | Tabulasi vertical ( <i>tab</i> )                               |
| \'               | Karakter petik tunggal                                         |
| \”               | Karakter petik ganda                                           |

Agar lebih jelas, lihat penggunaannya pada potongan kode program berikut

#### Contoh 4.1 Escape Character

```
print ("Departemen, \nStatistika Bisnis \'DSB\'")
[Run] Departemen,
 Statistika Bisnis 'DSB'
print ("=====")
print ("Fakultas\t\t Departemenn")
print ("=====")
print ("Fak.Vokasi\t\t Statistika Bisnis\nFak.Creabiz\t\t
Manajemen Bisnis \nFak.Martech \t\t Teknik Kelautan")
[Run] =====
 Fakultas Departemenn
=====
 Fak.Vokasi Statistika Bisnis
 Fak.Creabiz Manajemen Bisnis
 Fak.Martech Teknik Kelautan
```

#### 4.4. Argumen ljust (), rjust (), center ()

Penyesuaian posisi atau penyelarasan string sering digunakan dalam banyak aplikasi sehari-hari. Dalam python ada beberapa fungsi yang membantu menyelaraskan string. Fungsi-fungsi ini adalah:

```
str.ljust(s, width[, fillchar])
str.rjust(s, width[, fillchar])
str.center(s, width[, fillchar])
```

Fungsi-fungsi ini masing-masing *left-justify*, *right-justify* dan *center* string dalam bidang lebar yang diberikan. Fungsi tersebut mengembalikan sebuah string yang lebarnya setidaknya karakter lebar, dibuat dengan melapisi string dengan fillchar karakter (default adalah spasi) sampai lebar yang diberikan di sisi kanan, kiri atau kedua sisi.

##### a) center()

Fungsi ini menyelaraskan string sesuai dengan lebar yang ditentukan dan mengisi sisa ruang baris dengan ruang kosong jika argumen 'fillchr' tidak diberikan.

##### Syntax :

```
center(len, fillchr)
```

**len** : Lebar string untuk mengembangkannya.

**fillchr (optional)**: Karakter untuk mengisi ruang yang tersisa.

##### Contoh 4.2 Argumen center()

```
data ="Statistika Bisnis"
print ("The original string is : \n", data,
"\n")
[Run] The original string is :
Statistika Bisnis
```

### b) **ljust()**

Fungsi ini menyelaraskan string kearah kiri sesuai dengan lebar yang ditentukan dan mengisi sisa ruang baris dengan ruang kosong jika argumen ‘fillchr’ tidak diteruskan.

*Syntax: ljust(len, fillchr)*

```
ljust(len, fillchr)
```

#### Contoh 4.3 Argumen ljust()

```
lstr = "Statistika Bisnis"
print ("The left aligned string is : ")
print (lstr.ljust(40, '-'))
[Run] The left aligned string is :
Statistika Bisnis-----
```

### c) **rjust()**

Fungsi ini menyejajarkan string kearah kanan sesuai dengan lebar yang ditentukan dan mengisi sisa ruang baris dengan ruang kosong jika argumen ‘fillchr’ tidak diberikan.

*Syntax:*

```
rjust(len, fillchr)
```

#### Contoh 4.4 Argumen rjust()

```
rstr = "Statistika Bisnis"
print ("The left aligned string is : ")
print (rstr.ljust(40, '-'))
[Run] The left aligned string is :
-----Statistika Bisnis
```

## 4.5. Metode Format

Metode format adalah tambahan yang cukup baru untuk Python yang dapat melakukan tugas yang sama dengan metode justification dan banyak lagi. Misalnya, ini dapat menempatkan ribuan pemisah dalam angka, pembulatan angka, dan mengubah angka menjadi persentase. Berikut akan didemonstrasikan kemampuan pemberian metode dan kemudian menyajikan beberapa fitur lainnya

Jika *str1* adalah string dan w adalah lebar bidang, maka pernyataan formulir

```
print("{0:<ws}".format(str1))
print("{0:^ws}".format(str1))
print("{0:>ws}".format(str1))
```

menghasilkan output yang sama dengan pernyataan

```
print(str1.ljust(w))
print(str1.center(w))
print(str1.rjust(w))
```

Jika *num* adalah angka dan w adalah lebar bidang, maka pernyataan formulir

```
print("{0:<wn}".format(num))
print("{0:^wn}".format(num))
print("{0:>wn}".format(num))
```

menghasilkan output yang sama dengan pernyataan

```
print(str(num).ljust(w))
print(str(num).center(w))
print(str(num).rjust(w))
```

Perhatikan bahwa metode format menerima angka secara langsung tidak harus dikonversi ke string. Simbol <, ^, dan > yang mendahului lebar masing-masing bidang menginstruksikan fungsi *print* masing-masing untuk menyelaraskan kiri, tengah, dan kanan.

Di setiap pernyataan di atas yang berisi metode format, ada argumen tunggal (*num*) dalam metode format. Seringkali ada beberapa argumen, yang disebut dengan posisi yang dihitung dari nol. Angka 0 sebelum titik dua pada kurung kurawal mengacu pada fakta bahwa *num* berada pada posisi ke-0. Ketika ada beberapa argumen, ada beberapa pasang kurung kurawal, dengan masing-masing

pasangan kurung kurawal terkait dengan argumen. Angka-angka sebelum titik dua di dalam masing-masing pasangan kurung kurawal memberikan posisi argumen yang diformat.

```
print("0123456789012345678901234567")
print("{0:^5s}{1:<20s}{2:>3s)".format("Rank",
"Nama", "IPK"))
print("{0:^5n}{1:<20s}{2:>3n)".format(1,
"Barry Hamdan", 3.98))
print("{0:^5n}{1:<20s}{2:>3n)".format(2,
"Hisam Aaron", 3.80))
print("{0:^5n}{1:<20s}{2:>3n)".format(3,
"Rizal Ruth", 3.65))
[Run] 0123456789012345678901234567
 Rank Nama IPK
 1 Barry Hamdan 3.98
 2 Hisam Aaron 3.8
 3 Rizal Ruth 3.65
```

Ketika angka sedang diformat, alih-alih menggunakan huruf n di dalam kurung kurawal, yang sesuai dengan semua jenis angka, kita menggunakan huruf d untuk bilangan bulat, huruf f untuk angka titik format *float*, dan simbol % untuk angka menjadi ditampilkan sebagai persentase. Ketika f dan% digunakan, mereka harus didahului oleh suatu periode dan bilangan bulat. Seluruh angka menentukan jumlah tempat desimal yang akan ditampilkan. Dalam masing-masing dari tiga kasus, kita juga dapat menentukan apakah kita ingin ribuan pemisah dengan memasukkan koma setelah nomor bidang-lebar.

Ketika metode format digunakan untuk memformat angka, right-justify adalah pemberanaran standar. Oleh karena itu, ketika tidak ada simbol <, ^, dan >, nomor tersebut akan ditampilkan dengan benar di bidangnya. Tabel 3 menunjukkan beberapa pernyataan dan hasil yang mereka hasilkan.

**Tabel 4.3.** Simulasi Metode Format Pada Angka

| Statemen                          | Output   | Keterangan                  |
|-----------------------------------|----------|-----------------------------|
| print("{0:10d)".format(12345678)} | 12345678 | angka adalah bilangan bulat |

|                                       |            |                               |
|---------------------------------------|------------|-------------------------------|
| print("{0:10,d}".format(12345678))    | 12,345,678 | Format ribuan                 |
| print("{0:10.2f}".format(1234.5678))  | 1234.57    | Pembulatan                    |
| print("{0:10,.2f}".format(1234.5678)) | 1,234.57   | Pembulatan dan format pemisah |

**Tabel 4.3.** Simulasi Metode Format Pada Angka

| Statemen                              | Output     | Keterangan                                  |
|---------------------------------------|------------|---------------------------------------------|
| print("{0:10,.3f}".format(1234.5678)) | 1,234.568  | Pembulatan dan format pemisah               |
| print("{0:10.2%}".format(12.345678))  | 1234.57%   | Format persentase % dan dibulatkan          |
| print("{0:10,.3%}".format(12.34569))  | 1,234.568% | Format persentase %, pemisah dan dibulatkan |

### Latihan Soal

Dalam latihan berikut tentukan output dan buatlah program yang sesuai berdasarkan perintah yang telah ditentukan

1. Tentukan output yang ditampilkan oleh baris kode berikut.
  - a. print("Harga: ", 'Rp', 7,700, sep="")
  - b. print("Akurasi: ", 90, '%', sep="")
  - c. print("Data", " Sci", "ence", sep="")
  - d. print("Akurasi: ", 90, '%', sep="")
  - e. print("Cerdas\tAmanah\n\tKreatif")
  - f. print("COUNTRY\t", "LAND AREA") print("India\t", 2.5, "million sq km") print("China\t", 9.6, "million sq km")
  - g. x = 10  
y = 8
  - h. str1 = "If {0:s} dream it, {0:s} do it. - Walt Disney"  
print(str1.format("you can"))
2. print("NUMBER\tSQUARE\tCUBE")  
print(str(2) + "\t" + str(2 \*\* 2) + "\t" + str(2 \*\* 3))  
print(str(3) + "\t" + str(3 \*\* 2) + "\t" + str(3 \*\* 3))

3. Lanjutkan program berikut yang menunjukkan hasil output berupa persentase mahasiswa fakultas Fakultas Vokasi (Vocations) di ITS. Pembagian persentase disesuaikan dengan akumulasi 100 persen

```
print("{0:20s}{1:s}".format("Departemen", "Persentase Mahasiswa"))
print("{0:14s}{1:10.1%}".format("Statistika Bisnis", .20))
```

4. Buatlah kode untuk mendapatkan harga yang harus dibayar untuk suatu produk dari data harga asli produk dan diskon yang didapatkan.

Harga Asli = Rp. 150.000

Diskon = 10%

Harga setelah diskon = Rp. 135.000

5. Buatlah kode untuk mendapatkan nilai laba bersih perusahaan dari data pendapatan dan pengeluaran tahunan perusahaan sebagai input, dan tampilkan laba bersih perusahaan (pendapatan dikurangi biaya).

Pendapatan = Rp. 550.000

Pengeluaran = Rp. 410.000

Laba Bersih = Rp. 140.000

## Referensi

Schneider, David I. 2016. *An Introduction to Programming Using Python Global Edition*. University of Maryland. British Library Cataloguing-in-Publication Data Pearson Education Limited

## BAB 5. LIST, TUPLE DAN INTRO FILE

### 5.1. Pengertian List

List adalah struktur data yang menyimpan data secara terurut, anda dapat menyimpan serangkaian data atau objek menggunakan list. Secara teknik, penulisan item list pada python menggunakan kurung siku [ ]. Kurung siku ini adalah tempat untuk menyimpan item kedalam list. List memiliki sifat **mutable**, artinya item yang ada didalam list dapat ditambah, dikurangi atau dapat juga mencari item yang terdapat didalam list. List dibuat dengan menulis item yang terlampir dalam tanda kurung siku, dengan item dipisahkan dengan koma. Beberapa contoh list adalah

```
["Statistika Bisnis", 2020, "ITS"]
[5, 10, 4, 5]
["spam", "ni"]
```

List biasanya ditugaskan untuk sebuah nama. Sebagai contoh, kita dapat menulis

```
team = ["Statistika Bisnis", 2020, "ITS"]
nums = [5, 10, 4, 5]
words = ["spam", "ni"]
```

**Tabel 5.1.** List operations. (daftar tim, angka, dan kata-kata diberikan di atas.)

| Fungsi atau Metode | Contoh     | Output | Deskripsi                                           |
|--------------------|------------|--------|-----------------------------------------------------|
| Len                | len(words) | 2      | Jumlah item dalam list                              |
| max                | max(nums)  | 10     | Nilai terbesar (item harus memiliki tipe yang sama) |
| min                | min(nums)  | 4      | Nilai terkecil (item harus memiliki tipe yang sama) |
| sum                | sum(nums)  | 24     | total (item harus berupa angka)                     |

|       |               |   |                               |
|-------|---------------|---|-------------------------------|
| count | nums.count(5) | 2 | jumlah kemunculan suatu objek |
|       |               |   |                               |

**Tabel 5.1.** List operations. (daftar tim, angka, dan kata-kata diberikan di atas.)

| Fungsi atau Metode | Contoh                  | Output                      | Deskripsi                                             |
|--------------------|-------------------------|-----------------------------|-------------------------------------------------------|
| index              | nums.index(4)           | 2                           | indeks kemunculan pertama suatu objek                 |
| reverse            | words.reverse()         | ["ni", "spam"]              | menbalik urutan item                                  |
| clear              | team.clear()            | []                          | [] adalah daftar kosong                               |
| append             | nums.append(7)          | [5, 10, 4, 5, 7]            | menyisipkan objek di akhir daftar                     |
| extend             | nums.extend([1, 2])     | [5, 10, 4, 5, 1, 2]         | menyisipkan item daftar baru di akhir daftar          |
| del                | del team[-1]            | ["Statistika Bisnis", 2020] | menghapus item dengan indeks yang dinyatakan          |
| remove             | nums.remove(5)          | [10, 4, 5]                  | menghapus kejadian pertama dari suatu objek           |
| insert             | words.insert(1, "wink") | ["spam", "wink", "ni"]      | masukkan item baru sebelum item indeks yang diberikan |
| +                  | ['a', 1] + [2, 'b']     | ['a', 1, 2, 'b']            | rangkaian; sama dengan ['a', 1].extend([2, 'b'])      |
| *                  | [0] * 3                 | [0, 0, 0]                   | pengulangan                                           |

Seperti dijelaskan bahwa sebuah list adalah struktur data yang digunakan untuk menyimpan data, list adalah contoh penggunaan obyek dan class. Ketika kita menggunakan variabel *i* dan mengisinya dengan nilai integer 5, sama dengan kita membuat obyek (instance) *i* dari class (tipe) int. Class mempunyai method, fungsi yang didefinisikan dalam class. Anda bisa menggunakan method ini pada obyek class tersebut. Sebagai contoh, Python menyediakan method append untuk class list. contoh\_list.append('item 1') akan

menambahkan string 'item 1' kedalam list contoh\_list. Perhatikan notasi titik untuk mengakses method pada obyek. Class juga mempunyai field yang sama halnya variabel yang digunakan hanya untuk class.

### Contoh 5.1

Program berikut meminta lima nilai sebagai input dan menampilkan rata-rata setelah memasukkan dua nilai terendah. Nilai ditempatkan ke daftar nilai, dua nilai terendah dihapus dari daftar, dan fungsi jumlah dan len digunakan untuk menghitung rata-rata nilai yang tersisa.

```
grades = [] # Create the variable grades and assign it
the empty list.
num = float(input("Enter the first grade: "))
grades.append(num)
num = float(input("Enter the second grade: "))
grades.append(num)
num = float(input("Enter the third grade: "))
grades.append(num)
num = float(input("Enter the fourth grade: "))
grades.append(num)
num = float(input("Enter the fifth grade: "))
grades.append(num)
minimumGrade = min(grades)
grades.remove(minimumGrade)
minimumGrade = min(grades)
grades.remove(minimumGrade)
average = sum(grades) / len(grades)
print("Average Grade: {:.2f}".format(average))
[Run] Enter the first grade: 90
 Enter the second grade: 87
 Enter the third grade: 80
 Enter the fourth grade: 79
 Enter the fifth grade: 91
 Average Grade: 89.33
```

## 5.2. Slices/Irisan

Slices adalah sublist yang ditentukan dengan notasi titik dua. Ini dianalogikan dengan seutas tali. Beberapa notasi irisan ditunjukkan pada Tabel 5.2

**Tabel 5.2.** Arti notasi irisan

| Notasi     | Output                                                                   |
|------------|--------------------------------------------------------------------------|
| list1[m:n] | list terdiri dari item-item dari list1 yang memiliki indeks m sampai n-1 |
| list1[:]   | list baru yang berisi item yang sama dengan list1                        |

**Tabel 5.2.** Arti notasi irisan

| Notasi    | Output                                                                               |
|-----------|--------------------------------------------------------------------------------------|
| list1[m:] | list yang terdiri dari item-item list1 dari list1 [m] hingga akhir list1             |
| list1[:m] | list yang terdiri dari item list1 dari awal list1 ke elemen yang memiliki indeks m-1 |

**Catatan:** Fungsi `del` dapat digunakan untuk menghapus slices dari list. Juga, jika item indeks m tidak di sebelah kiri item indeks n, maka `list1[m: n]` akan menjadi daftar kosong. Beberapa contoh irisan ditunjukkan pada Tabel 5.3.

**Tabel 5.3.** Contoh irisan di mana `list1 = ['a', 'b', 'c', 'd', 'e', 'f']`.

| Statement                        | Output                                                              |
|----------------------------------|---------------------------------------------------------------------|
| <code>list1[1:3]</code>          | <code>['b', 'c']</code>                                             |
| <code>list1[-4:-2]</code>        | <code>['c', 'd']</code>                                             |
| <code>list1[:4]</code>           | <code>['a', 'b', 'c', 'd']</code>                                   |
| <code>list1[4:]</code>           | <code>['e', 'f']</code>                                             |
| <code>list1[:]</code>            | <code>['a', 'b', 'c', 'd', 'e', 'f']</code>                         |
| <code>del list1[1:3]</code>      | <code>['a', 'd', 'e', 'f']</code>                                   |
| <code>list1[2:len(list1)]</code> | <code>['c', 'd', 'e', 'f']</code>                                   |
| <code>(list1[1:3])[1]</code>     | 'c' (Statement ini bisa ditulis dengan <code>list1[1:3][1]</code> ) |
| <code>list1[3:2]</code>          | <code>[]</code> , list tidak memiliki item; yaitu, daftar kosong    |

### 5.3. Metode `split` dan `join`

Sebuah string yang utuh dapat dimodifikasi dengan memecah atau menggabungkan menjadi sebuah string yang utuh kembali. Metode untuk memecah suatu string menjadi item-item dalam list disebut metode ***split***, sedangkan metode yang merupakan kebalikan dari ***split*** adalah metode ***join*** yang berfungsi untuk menggabungkan item-item list menjadi sebuah string yang utuh kembali

```
L = strVar.split(",")
```

Membuat list L yang berisi nilai string N + 1 sebagai itemnya. Artinya, item pertama dari daftar adalah teks sebelum koma pertama dari strVar, item kedua dari daftar adalah teks antara koma pertama dan kedua, ..., dan item terakhir dari daftar adalah teks yang mengikuti koma terakhir. String yang terdiri dari karakter koma disebut pemisah untuk pernyataan di atas. String apa pun dapat digunakan sebagai pemisah. (Tiga pemisah umum yang terdiri dari string karakter tunggal adalah ",", "\n", dan ""). Jika tidak ada pemisah yang ditentukan, metode split menggunakan spasi sebagai pemisah, di mana spasi adalah string yang karakternya adalah baris baru , tab vertikal, atau karakter spasi.

Metode join, yang merupakan kebalikan dari metode split, mengubah daftar string menjadi nilai string yang terdiri dari elemen-elemen dari daftar yang digabungkan menjadi satu dan dipisahkan oleh string yang ditentukan. Bentuk umum pernyataan menggunakan join dan memiliki string "," sebagai pemisah

```
strVar = ",".join(L)
```

### Contoh 5.2 Metode Split

Program berikut masing-masing menampilkan list = ['a', 'b', 'c']

```
list = ['a', 'b', 'c']
print("a,b,c".split(','))
print("a***b***c".split('***'))
print("a\nb\n c".split())
print("a b c".split())
[Run] ['a', 'b', 'c']
 ['a', 'b', 'c']
 ['a', 'b', 'c']
 ['a', 'b', 'c']
```

### Contoh 5.3 Metode Join

Program berikut menunjukkan bagaimana metode join dapat digunakan untuk menampilkan item dari list string.

```
line = ["Indonesia", "Raya", "Merdeka", "NKRI"]
print(" ".join(line))
krispies = ["Merah", "Kuning", "Hijau"]
print(", ".join(krispies))
[Run] Indonesia Raya Merdeka NKRI
 Merah, Kuning, Hijau
```

## 5.4. File Teks

Nilai-nilai yang digunakan dalam program Python berada di memori dan hilang ketika program berakhir. Namun, jika suatu program menulis nilai-nilai ke file pada perangkat penyimpanan (seperti hard disk atau flash drive), program Python apa pun dapat mengakses nilai di lain waktu. Artinya, file membuat penyimpanan data jangka panjang.

File teks adalah file sederhana yang terdiri dari baris teks tanpa pemformatan (yaitu, tanpa huruf tebal atau miring) yang dapat dibuat dan dibaca dengan Notepad (di PC) atau TextEdit (di Mac). Biasanya, file teks memiliki ekstensi txt. File teks sebenarnya dapat dibuat dengan pengolah kata apa pun. Misalnya, setelah dokumen dibuat di Word, Anda dapat memanggil Save As dan kemudian pilih "Save as type: Plain Text (\*.txt)" untuk menyimpan dokumen sebagai file teks. Juga, file teks apa pun yang ada dapat dibuka dan diedit di Word. Setiap baris file teks (kecuali mungkin baris terakhir) berakhir dengan karakter baris baru. Garis-garis file teks dapat ditempatkan ke dalam list dengan kode program

```
infile = open("Data.txt", 'r')
listName = [line.rstrip() for line in infile]
infile.close()
```

Jika data dalam file teks adalah semua angka, proses dalam paragraf sebelumnya menghasilkan daftar yang terdiri dari string, dengan setiap string memegang angka. Untuk file angka, kita dapat menempatkan angka ke dalam daftar dengan kode program

```
infile = open("Data.txt", 'r')
listName = [eval(line) for line in infile]
infile.close()
```

## 5.5. Pengertian Tuple

Tuple secara penggunaan sejenis dengan list namun yang mebedakan adalah sifat tuple yang tidak bisa diubah setelah didefinisikan (immutable). Elemen pada tuple bersifat tetap atau tidak dapat dihapus dan diubah. Maka dari itu tuple disebut immutable sequence. Tuple dibuat dengan menspesifikasikan item, tuple dipisahkan menggunakan tanda koma dan opsional diapit dengan tanda kurung. Tupel menggunakan tanda kurung, sedangkan list menggunakan tanda kurung siku.

### Contoh 5.4 Fungsi Tuple

Program berikut menunjukkan bahwa tupel memiliki beberapa fungsi yang sama dengan list

```
t = 5, 7, 6, 2, 1
print(t)
print(len(t), max(t), min(t), sum(t))
print(t[0], t[-1], t[:2])
[Run] (5, 7, 6, 2, 1)
 5 7 1 21
 5 1 (5, 7)
```

Tupel kosong ditulis sebagai dua tanda kurung yang tidak berisi apa-apa, contohnya : tup1 = (); Untuk menulis tupel yang berisi satu nilai, Anda harus memasukkan koma, meskipun hanya ada satu nilai, contohnya : tup1 = (70,) Seperti indeks String, indeks tuple mulai dari 0, dan mereka dapat diiris, digabungkan, dan seterusnya.

#### a. Mengakses nilai tuple

Seperti dijelaskan bahwa nilai dalam tuple tidak dapat diubah, namun kita dapat mengambil data yang tersimpan didalam tuple dengan cara menggunakan tanda kurung siku untuk mengiris beserta indeks atau indeks untuk mendapatkan nilai yang tersedia pada indeks tersebut. Sebagai contoh :

```
tup1 = ('fisika', 'kimia', 1993, 2017)
tup2 = (1, 2, 3, 4, 5, 6, 7)
print ("tup1[0]: ", tup1[0])
print ("tup2[1:5]: ", tup2[1:5])
[Run] tup1[0]: fisika
tup2[1:5]: (2, 3, 4, 5)
```

### b. Mengupdate nilai tuple

Tuple tidak berubah, yang berarti tidak dapat memperbarui atau mengubah nilai elemen tupel. Anda dapat mengambil bagian dari tupel yang ada untuk membuat tupel baru seperti ditunjukkan oleh contoh berikut.

```
tup1 = (12, 34.56)
tup2 = ('abc', 'xyz')
Aksi seperti dibawah ini tidak bisa dilakukan pada
tuple python
Karena memang nilai pada tuple python tidak bisa
diubah
tup1[0] = 100;
Jadi, buatlah tuple baru sebagai berikut
tup3 = tup1 + tup2
print (tup3)
[Run] (12, 34.56, 'abc', 'xyz')
```

### c. Menghapus elemen tuple

Menghapus elemen tuple secara individu tidak dapat dilakukan, hal ini karena prinsip tuple yang immutable. Yang dapat dilakukan adalah menghapus tuple itu sendiri dengan menggunakan statemen *del*. Sebagai contoh

```
tup = ('fisika', 'kimia', 1993, 2017);
print (tup)
del tup
print (tup)
[Run] -----

NameError
Traceback (most recent call last)
<ipython-input-8-8130c03e2c82> in <module>
----> 1 print(tup)

NameError: name 'tup' is not defined
```

Tuple merespons semua operasi urutan umum yang telah dijelaskan pada String di bab sebelumnya. Khusus untuk operator + dan \* sama seperti String respon tuple yang artinya penggabungan dan pengulangan kecuali hasilnya adalah tupel baru, bukan string. Dibawah ini adalah tabel daftar operasi dasar pada Tuple python.

**Tabel 5.4.** Operasi Dasar Pada Tuple Python

| Python Expression                          | Hasil                                | Penjelasan    |
|--------------------------------------------|--------------------------------------|---------------|
| len((1, 2, 3))                             | 3                                    | Length        |
| (1, 2, 3) + (4, 5, 6)                      | (1, 2, 3, 4, 5, 6)                   | Concatenation |
| ('Halo!',) * 4                             | ('Halo!', 'Halo!', 'Halo!', 'Halo!') | Repetition    |
| 3 in (1, 2, 3)                             | True                                 | Membership    |
| for x in (1,2,3) :<br>print (x, end = ' ') | 1 2 3                                | Iteration     |

Pernyataan seperti

**(x, y, z) = (5, 6, 7)**

menciptakan tiga variabel dan memberikan nilai kepada mereka.

Pernyataan itu juga bisa dituliskan

**x, y, z = 5, 6, 7**

yang dapat dianggap membuat tiga penugasan variabel dengan satu pernyataan.

### Contoh 5.5 Nilai Swap

Program berikut menukar nilai dua variabel. Intinya, baris ketiga dari program ini adalah menetapkan tuple (6, 5) ke tuple (x, y) atau memasukkan nilai x=5 dan y=6 pada tuple x,y.

```
x = 5
y = 6
x, y = y, x
print(x, y)
[Run] 6 5
```

## 5.6. Nested Lists

Sejauh ini, semua item dalam list dan tupel adalah angka atau string. Namun, item juga bisa berupa list atau tupel. List dan tupel memainkan peran penting dalam menganalisis data. Jika L adalah list tupel, maka L [0] adalah tupel pertama, L [0] [0] adalah item pertama dari tupel pertama, L [-1] (sama dengan L [len (L) -1 ]) adalah tupel terakhir, dan L [-1] [- 1] adalah item terakhir dari tupel terakhir. Ekspresi seperti L [0] [0] dapat dianggap sebagai (L [0]) [0].

### Contoh 5.6

Berikut disajikan data empat wilayah dengan populasi terpadat di Indonesia. Wilayah daftar berisi empat tupel, dengan masing-masing tupel memberi nama dan populasi 2020 (dalam jutaan) dari wilayah dengan jumlah populasi terpadat berdasarkan data BPS. Program berikut menampilkan populasi tahun 2020 Provinsi Jawa Timur dan menghitung total populasi ke empat Provinsi.

```

provinsi = [("Jawa Barat", 49935,7), ("Jawa Timur", 39886.3),
 ("Jawa Tengah", 34940,1), ("Sumatera Utara", 14703,5)]
print("Jumlah Populasi Penduduk Tahun 2020 Provinsi",
 provinsi[1][0], "sebesar", provinsi[1][1],
 "juta jiwa.")

```

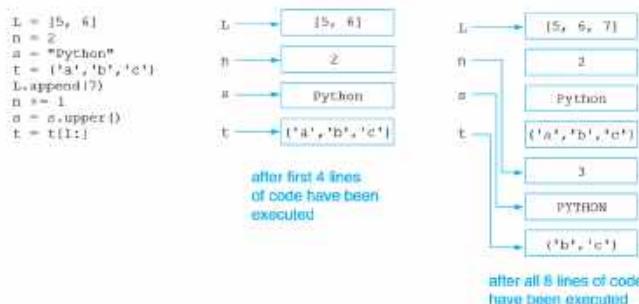
```

[Run] Jumlah Populasi Penduduk Tahun 2020 Provinsi Jawa Timur
sebesar 39886.3 juta jiwa.

```

## 5.7. Objek yang Tidak Berubah dan Dapat Berubah

Objek atau variabel adalah suatu entitas yang menyimpan data dan memiliki operasi dan / atau metode yang dapat memanipulasi data. Bilangan, string, list, dan tupel adalah objek. Ketika sebuah variabel dibuat dengan pernyataan penugasan, nilai di sisi kanan menjadi objek di memori, dan referensi variabel (yaitu, menunjuk ke) objek itu. Ketika daftar diubah, perubahan dilakukan ke objek di lokasi memori daftar. Namun, ketika variabel yang nilainya berupa angka, string, atau tuple, nilainya berubah, Python menunjuk lokasi memori baru untuk menyimpan nilai baru dan variabel referensi bahwa objek baru. Kami mengatakan bahwa daftar dapat diubah di tempat, tetapi angka, string, dan tupel tidak bisa. Objek yang dapat diubah di tempat disebut bisa berubah, dan objek yang tidak dapat diubah di tempat disebut tidak berubah. Gambar 1 menunjukkan delapan baris kode dan alokasi memori setelah empat baris kode pertama telah dieksekusi dan setelah semua delapan baris telah dieksekusi.



**Gambar 5.1.** Alokasi memori yang terkait dengan suatu program.  
(Sumber : David, 2016)

## 5.8. Menyalin List

Jika variabel var1 memiliki nilai yang dapat diubah (seperti list), maka pernyataan dari bentuk var2 = var1 menghasilkan var2 merujuk objek yang sama dengan var1. Oleh karena itu, setiap perubahan pada nilai var2 mempengaruhi nilai var1. Pertimbangkan empat baris kode berikut:

```
list1 = ['a', 'b'] # Lists are mutable objects.
list2 = list1 # list2 will point to the same
 # memory location as list1
list2[1] = 'c' # Changes the value of the second
 # item in the list object
print(list2[1])
[Run] c
```

Pada baris kode kedua, daftar variabel merujuk lokasi memori yang sama dengan list1. Oleh karena itu, setiap perubahan pada item dalam list2 menghasilkan perubahan yang sama dalam nilai list1. Efek ini tidak akan terjadi jika baris kedua kode diubah menjadi list2 = list (list1) atau list2 = list1 [:]. Dalam kasus tersebut, list2 akan menunjuk ke objek di lokasi memori berbeda yang berisi nilai yang sama dengan list1. Maka baris ketiga kode tidak akan memengaruhi lokasi memori yang ditunjuk oleh list1 dan outputnya adalah ['a', 'b'].

## 5.9. Indexing, Deleting, and Slicing Out of Bounds

Python tidak mengizinkan pengindeksan di luar batas untuk masing-masing item dalam daftar dan tupel, tetapi mengizinkannya untuk irisan. Misalnya, jika

```
list1 = [1, 2, 3, 4, 5]
```

kemudian **print (list1 [7])**, **print (list1 [-7])**, dan **del list1 [7]** memicu pesan kesalahan *Traceback IndexError*. Jika indeks kiri dalam slice terlalu jauh negatif, slice akan mulai di awal daftar, dan jika indeks kanan terlalu besar, slice akan pergi ke akhir daftar. Misalnya,

```
list1 [-10: 10] adalah [1, 2, 3, 4, 5]
list1 [-10: 3] adalah [1, 2, 3]
list1[3:10] is [4, 5]
del list1[3:7] is [1, 2, 3]
```

### Latihan Soal

Dalam latihan berikut tentukan output dan buatlah program yang sesuai berdasarkan perintah yang telah ditentukan

1. Tentukan output yang ditampilkan oleh baris kode berikut.  
departemen = [("Statistika Bisnis", "Vokasi"), ("Aktuaria", "Scintecis"), ("Manajemen Bisnis", "Creabiz")]
(name, fakl) = (departemen[0][0], departemen[0][1])
print(name, " Masuk kedalam Fakultas ", fakl, !,sep = "")
2. a = 3
b = 4
print((a + b,))
print((a + b))
print()
3. Apakah pernyataan s = 'a' + 'b' dan s = "" .join ([ 'a', 'b' ]) memberikan nilai yang sama ke variabel s?
4. Asumsikan bahwa negara daftar berisi nama-nama lima puluh negara di dunia, dan tentukan output yang ditampilkan oleh baris kode.
countries = [ "Japan", "India", "Algeria", "Brazil", "Angola", "England", "Argentina", "Portugal", "China", "Australia", "Austria", "Ghana", "Bahamas", "Bangladesh", "Belgium", "Bhutan", "Bosnia", "Cameroon", "Canada", "Denmark" ]
  - a. print(countries[2], ", ", countries[-1])
  - b. countries.insert(5, "Germany")
print(countries[5])
  - c. del countries[4]
print(countries[4])
  - d. countries[1] = "Poland"
print(countries[:3])
  - e. print(countries[2:len(countries)])

5. Asumsikan bahwa list1 berisi 100 item. Tentukan jumlah item di setiap irisan.
- list1[-8:]
  - list1[:8]
  - list1[:]
  - list1[-8:-1]
  - list1[8:8]
  - list1[1:-1]
6. Asumsikan bahwa daftar nums = [7, 3, 9, 1], dan tentukan output yang ditampilkan oleh baris kode.
- print("Largest Number:", max(nums))
  - print("Length:", len(nums)) list1[:]
  - print("Total:", sum(nums)) list1[8:8]
  - print("Number lot", sum(nums) / list(nums))
7. tuple1 = ("Institut", "Teknologi", "Sepuluh", "Nopember", "Departemen", "Statistika", "Bisnis")  
 tuple2 = tuple1[4:] + tuple1[:4]  
 print(" ".join(tuple2))) Asumsikan bahwa daftar nums = [7, 3, 9, 1], dan tentukan output yang ditampilkan oleh baris kode.
8. allDay = " around- the-  
 clock"  
 print(allDay.split('-'))
9. phoneNumber = "9876543219"  
 list1 = list(phoneNumber)  
 list1.insert(3, '-')  
 list1.insert(7, '-')  
 phoneNumber = "".join(list1)  
 print(phoneNumber)
10. word = "diary"  
 list1 = list(word)  
 list1.insert(3, list1[1])  
 del list1[1]  
 word = "".join(list1)  
 print(word)

## **Referensi**

- Belajarpython. (2020, 6 Juli). Tuple Python. Diakses pada 6 Juli 2020, dari <https://belajarpython.com/tutorial/tuple-python>
- Schneider, David I. 2016. *An Introduction to Programming Using Python Global Edition*. University of Maryland. British Library Cataloguing-in-Publication Data Pearson Education Limited

## BAB 6. RELATIONAL DAN OPERATOR LOGIS

### 6.1. Nilai-nilai Karakter ASCII

Masing-masing dari 47 tombol di bagian tengah mesin tik keyboard dapat menghasilkan dua karakter, dengan total 94 karakter. Menambahkan 1 untuk karakter yang dihasilkan oleh spasi membuat 95 karakter. Terkait dengan karakter-karakter ini adalah angka mulai dari 32 hingga 126. Nilai-nilai ini, yang disebut nilai ASCII karakter, Tabel 6.1 menunjukkan beberapa nilai ASCII.

**Tabel 6.1.** Karakter Nilai-nilai ASCII

| No | Karakter | No  | Karakter |
|----|----------|-----|----------|
| 32 | (space)  | 66  | B        |
| 33 | !        | 90  | Z        |
| 34 | “        | 97  | A        |
| 35 | #        | 98  | b        |
| 48 | 0        | 122 | z        |
| 49 | 1        | 123 | {        |
| 57 | 9        | 125 | }        |
| 65 | A        | 126 | ~        |

Standar ASCII juga memberikan karakter ke beberapa angka di atas 126. Tabel 6.2 menunjukkan beberapa nilai ASCII yang lebih tinggi.

**Tabel 6.2.** Karakter Nilai-nilai ASCII yang lebih tinggi

| No  | Karakter | No  | Karakter |
|-----|----------|-----|----------|
| 162 | ¢        | 181 | µ        |
| 169 | ©        | 188 | ¼        |
| 176 | °        | 189 | ½        |
| 177 | ±        | 190 | ¾        |
| 178 | ²        | 247 | ÷        |
| 179 | ³        | 248 | Ø        |

Jika `n` adalah angka non-negatif, maka

```
chr(n)
```

adalah string karakter tunggal yang terdiri dari karakter dengan nilai ASCII `n`. Jika `str` adalah string karakter tunggal, maka

```
ord(str)
```

adalah nilai karakter dari ASCII. Misalnya, pernyataan

```
print(chr(65))
```

menampilkan angka 65

```
print(ord('A'))
```

menampilkan huruf A.

Rangkaian dapat digunakan dengan `chr` untuk mendapatkan string menggunakan karakter ASCII yang lebih tinggi. Misalnya, pernyataan

```
print("32" + chr(176) + " Fahrenheit")
```

menampilkan 32° Fahrenheit.

## 6.2. Operator Relasional

Operator relasional kurang dari (`<`) dapat diterapkan pada angka, string, dan objek lainnya. Suatu angka **a** dikatakan lebih kecil dari angka **b** jika **a** terletak di sebelah kiri **b** pada garis bilangan. Misalnya,  $2 < 5$ ,  $-5 < -2$ , dan  $0 < 3.5$ .

String **a** dikatakan kurang dari string **b** jika **a** mendahului **b** saat menggunakan karakter ASCII. Digit mendahului huruf besar, yang mendahului huruf kecil. Dua string dibandingkan karakter dengan karakter (bekerja dari kiri ke kanan) untuk menentukan string mana yang harus mendahului yang lain. Jadi, "kucing"  $<$  "anjing", "kereta"  $<$  "kucing", "kucing"  $<$  "katalog", "9W"  $<$  "kelelawar", "Anjing"  $<$  "kucing", dan "sales\_99"  $<$  "sales\_retail". Jenis statemen ini disebut

**leksikografis.** Tabel 6.3. menunjukkan operator relasional yang berbeda dan artinya.

**Tabel 6.3.** Operator Relasional

| Operator                            | Contoh     | Penjelasan                                                                                                       |
|-------------------------------------|------------|------------------------------------------------------------------------------------------------------------------|
| Sama dengan $==$                    | $1 == 1$   | bernilai True Jika masing-masing operan memiliki nilai yang sama, maka kondisi bernilai benar atau True.         |
| Tidak sama dengan $!=$              | $2 != 2$   | bernilai False Akan menghasilkan nilai kebalikan dari kondisi sebenarnya.                                        |
| Lebih besar dari $>$                | $7 > 3$    | bernilai True Jika nilai operan kiri lebih besar dari nilai operan kanan, maka kondisi menjadi benar.            |
| Lebih kecil dari $<$                | $7 < 3$    | bernilai True Jika nilai operan kiri lebih kecil dari nilai operan kanan, maka kondisi menjadi benar.            |
| Lebih besar atau sama dengan $\geq$ | $7 \geq 3$ | bernilai True Jika nilai operan kiri lebih besar dari nilai operan kanan, atau sama, maka kondisi menjadi benar. |
| Lebih kecil atau sama dengan $\leq$ | $7 \leq 3$ | bernilai True Jika nilai operan kiri lebih kecil dari nilai operan kanan, atau sama, maka kondisi menjadi benar. |
| in, not in                          |            | Membership (Keanggotaan)                                                                                         |

### Contoh 6.1 Operator Relasional

Tentukan apakah masing-masing kondisi berikut dievaluasi menjadi Benar atau Salah.

- $7 \leq 7$
- $7 < 6$
- “cara”  $<$  “cari”
- “Daging”  $<$  “daging”
- “fun” in “refunded”

### Pembahasan.

- Benar.** Notasi  $\leq$  berarti "kurang dari atau sama dengan." Artinya, kondisinya benar asalkan salah satu dari dua situasi berlaku. Yang kedua (sama dengan) berlaku.

- b) **Salah.** Notasi  $<$  berarti "benar-benar kurang dari" dan tidak ada angka yang bisa benar-benar kurang dari itu sendiri.
- c) **Benar.** Karakter string dibandingkan satu per satu yang bekerja dari kiri ke kanan. Karena dua karakter pertama cocok, karakter ketiga menentukan urutan.
- d) **Benar.** Karena huruf besar mendahului huruf kecil dalam tabel ASCII, karakter pertama "Daging" mendahului karakter pertama "daging".
- e) **Benar.** String "fun" adalah "refunded" [2:5], sebuah substring dari "refunded".

### Contoh 6.2 Operator Relasional

Misalkan variabel **a** dan **b** memiliki nilai 4 dan 3, dan variabel **c** dan **d** memiliki nilai "hello" dan "bye". Apakah kondisi berikut ini benar atau salah?

- a)  $(a + b) < (2 * a)$
- b)  $(\text{len}(c) - b) == (a/2)$
- c)  $c < ("good" + d)$

#### Pembahasan.

- a) **Benar.** Nilai  $a + b$  adalah 7 dan nilai  $2 * a$  adalah 8. Karena  $7 < 8$ , kondisinya benar.
- b) **Benar**, karena nilai  $\text{len}(c) - b$  adalah 2, sama dengan  $(a / 2)$ .
- c) **Salah.** Kondisi "halo"  $<$  "good bye" salah, karena huruf **h** setelah **g** dalam alphabet

**Int** dapat dibandingkan dengan **float**. Jika tidak, nilai dari berbagai jenis tidak dapat dibandingkan. Misalnya, string tidak dapat dibandingkan dengan angka.

Operator relasional dapat diterapkan ke list atau tupel. Agar dua list atau dua tupel menjadi sama, mereka harus memiliki panjang yang sama dan item yang sesuai harus memiliki nilai yang sama. Nilai kebenaran dari kondisi ditentukan dengan membandingkan item yang sesuai berturut-turut sampai dua item berbeda (atau tidak dapat dibandingkan) atau sampai salah satu urutan kehabisan item. Pasangan item pertama yang memiliki nilai berbeda menentukan nilai kebenaran dari kondisi tersebut. Jika salah satu urutan kehabisan item dan semua

pasangan item cocok, maka urutan yang lebih pendek dikatakan lebih rendah dari keduanya. Beberapa contoh perbandingan yang memiliki nilai kebenaran Benar adalah sebagai berikut:

```
[3, 5] < [3, 7]
[3, 5] < [3, 5, 6]
[3, 5, 7] < [3, 7, 2]
[7, "three", 5] < [7, "two", 2]
```

Ketika operator **in** diterapkan ke list atau tuple, itu harus dianggap berarti item. Dua contoh adalah sebagai berikut:

```
'b' in ['a', 'b', 'c']
'B' not in ('a', 'b', 'c')
```

### 6.3. Menyortir Item dalam List

Item dalam list tempat setiap pasangan item dapat dibandingkan dapat dipesan dengan metode pengurutan. Pernyataan

```
list1.sort ()
```

perubahan *list1* ke daftar yang memiliki item yang sama, tetapi dalam urutan menaik baik secara numerik atau leksikografis yang sesuai.

#### Contoh 6.3 Sortir List

Program berikut menggambarkan bagaimana Python membuat dua list sederhana.

```
list1 = [6, 4, -5, 3.5]
list1.sort()
print(list1)
list2 = ["ha", "hi", 'B', '7']
list2.sort()
print(list2)
[Run] [-5, 3.5, 4, 6]
['7', 'B', 'ha', 'hi']
```

#### Contoh 6.4 Sortir List

Program berikut menggambarkan bagaimana Python membuat item dalam daftar string yang rumit. Catatan: chr(177) adalah ± dan chr(162) adalah ¢

```
list1 = [chr(177), "cari", "cara", "Daging",
"daging", "8-ball", "7" + chr(162)]
list1.sort()
print(list1)
[Run] ['7¢', '8-ball', 'Daging', 'cara',
'cari', 'daging', '±']
```

Program berikut membuat item list dalam tupel

```
monarchs = [("Hasan", 5), ("Husein", 2), ("Hisam",
6), ("Hasim", 1)]
monarchs.sort()
print(monarchs)
[Run] [('Hasan', 5), ('Hasim', 1), ('Hisam',
6), ('Husein', 2)]
```

## 6.4. Operator Logis

Pemrograman sering membutuhkan kondisi yang lebih kompleks daripada yang dipertimbangkan sejauh ini. Sebagai contoh, misalkan kita ingin menyatakan bahwa nilai variabel str1 adalah string dengan panjang 10 dan berisi substring "gram". Kondisi Python yang tepat adalah

```
len(str1) == 10) and ("gram" in str1)
```

Kondisi ini adalah kombinasi dari kondisi (len (str1) == 10) dan kondisi ("gram" di str1) dengan operator logis **and**. Tiga operator logis utama adalah **and**, **or**, dan **not**. Kondisi yang menggunakan operator ini disebut kondisi gabungan. Jika **cond1** dan **cond2** adalah kondisi, maka kondisi gabungan

```
cond1 and cond2
```

benar jika kedua kondisi tersebut benar. Kalau tidak, itu salah. Kondisi gabungan

cond1 or cond2

benar jika salah satu (atau keduanya) dari kedua kondisi itu benar. Kalau tidak, itu salah. Kondisi gabungan

not cond1

benar jika kondisinya salah, dan salah jika kondisinya benar.

### Contoh 6.5 Operator yang logis

Misalkan variabel n memiliki nilai 4 dan variabel answ memiliki nilai "Y". Tentukan apakah masing-masing kondisi berikut dievaluasi menjadi Benar atau Salah.

- a)  $(2 < n)$  and  $(n < 6)$
- b)  $(2 < n)$  or  $(n == 6)$
- c) not  $(n < 6)$
- d)  $(answ == "Y")$  or  $(answ == "y")$
- e)  $(answ == "Y")$  and  $(answ == "y")$
- f) not  $(answ == "y")$
- g)  $((2 < n)$  and  $(n == 5 + 1))$  or  $(answ == "No")$
- h)  $((n == 2)$  and  $(n == 7))$  or  $(answ == "Y")$
- i)  $(n == 2)$  and  $((n == 7)$  or  $(answ == "Y"))$

### Pembahasan

- a) **Benar**, karena kondisi  $(2 < 4)$  dan  $(4 < 6)$  keduanya benar.
- b) **Benar**, karena kondisi  $(2 < 4)$  benar. Fakta bahwa kondisi  $(4 == 6)$  salah tidak memengaruhi kesimpulan. Satu-satunya persyaratan adalah bahwa setidaknya satu dari dua kondisi itu benar.
- c) **Salah**, karena  $(4 < 6)$  benar.

- d) **Benar**, karena kondisi pertama menjadi ("Y" == "Y") ketika nilai variabel answ diganti dengan answ.
- e) **Salah**, karena kondisi kedua salah. Sebenarnya, kondisi gabungan ini salah untuk setiap nilai answ
- f) **Benar**, karena ("Y" == "y") salah.
- g) **Salah**, Dalam ekspresi logis ini, kondisi gabungan ((2 < n) dan (n == 5 + 1)) dan kondisi sederhana (answ == "Tidak") digabungkan oleh operator logika **or**. Karena kedua kondisi ini salah, maka secara keseluruhan salah.
- h) **Benar**, karena kondisinya mengikuti atau benar.
- i) **Salah**, karena kondisi pertama salah. (Membandingkan (h) dan (i) menunjukkan perlunya menggunakan tanda kurung untuk menentukan pengelompokan yang dimaksud.)

## 6.5. Evaluasi Short-Circuit

Ketika Python menemukan kondisi gabungan (cond1 and cond2), pertama-tama mengevaluasi cond1. Jika cond1 salah, Python menyadari bahwa kondisi gabungan salah dan karenanya tidak repot untuk mengevaluasi cond2. Demikian pula, ketika Python menemukan kondisi gabungan (cond1 or cond2), pertama-tama mengevaluasi cond1. Jika cond1 benar, Python menyadari bahwa kondisi gabungan benar dan oleh karena itu tidak repot-repot untuk mengevaluasi cond2. Proses ini disebut **evaluasi short-circuit**.

Beberapa bahasa pemrograman mengevaluasi kedua bagian dari kondisi gabungan sebelum memberikan nilai pada kondisi gabungan. Jika demikian, evaluasi kondisinya

```
number != 0) and (m == (n / number))
```

akan menyebabkan program macet dan menampilkan pesan kesalahan ketika angka memiliki nilai 0. Namun, karena **evaluasi short-circuit**, evaluasi kondisi majemuk ini tidak akan pernah menyebabkan masalah dalam Python. **evaluasi short-circuit** terkadang meningkatkan kinerja suatu program. Misalnya, ketika evaluasi cond2 memakan waktu.

## 6.6. Tipe Data Boolean/bool

Selain tipe data **int** dan **float**, dalam python juga ada tipe data boolean. Tipe data ini berupa biner yang berisi dengan salah satu dari 2 nilai: True atau False. Tipe data boolean banyak dipakai untuk percabangan kode program atau untuk memutuskan apa yang mesti dijalankan ketika sebuah kondisi terjadi. Sebagai contoh, kita bisa membuat kode program untuk menentukan apakah status kelulusan berdasarkan input dari pengguna. Untuk keperluan ini kita harus mengecek terlebih dahulu apakah status tersebut bisa dibagi 2 (untuk status lulus atau tidak). Tipe data boolean bisa dipakai untuk menampung kondisi seperti ini, yakni benar atau salah (*True* atau *False*).

Pernyataan program

```
print(condition)
```

akan menampilkan Benar atau Salah. Objek Benar dan Salah dikatakan memiliki tipe data Boolean atau tipe data bool. Baris-baris berikut kode menampilkan hasil **False**

```
x = 5
print((3 + x) < 7)
```

Baris-baris kode berikut menampilkan hasil **True**:

```
x = 2
y = 3
var = x < y
print(var)
```

## 6.7. Tiga Metode Yang Mengembalikan Nilai Boolean

Jika str1 dan str2 adalah string, maka kondisinya

```
str1.startswith(str2)
```

memiliki nilai True jika dan hanya jika str1 dimulai dengan str2, dan kondisinya

```
str1.endswith(str2)
```

memiliki nilai True jika dan hanya jika str1 diakhiri dengan str2.

Misalnya, dua kondisi berikut ini benar:

```
"fantastic".startswith("fan")
"fantastic".endswith("stic")
```

Jika var1 memiliki nilai "fantastis" dan var2 memiliki nilai "Fant", maka dua kondisi berikut ini salah:

```
var1.startswith(var2)
"elephant".endswith(var2)
```

Jika item adalah literal atau variabel, maka kondisi formulir

```
isinstance(item, dataType)
```

memiliki nilai **True** jika dan hanya jika item memiliki tipe data yang ditentukan, di mana tipe data adalah semua tipe data (seperti int, float, str, bool, list, atau tuple). Sebagai contoh, kondisi `isinstance ("32", int)` memiliki nilai **False** dan kondisi `isinstance (32, int)` memiliki nilai **True**.

Tabel 6.5 menunjukkan beberapa metode string lain yang mengembalikan nilai Boolean. Dalam tabel, asumsikan bahwa str1 bukan string kosong. Setiap metode dalam tabel mengembalikan False ketika str1 adalah string kosong.

**Tabel 6.5.** Metode yang mengembalikan **True** atau **False**

| Metode                      | Pengembalian True saat                                                                       |
|-----------------------------|----------------------------------------------------------------------------------------------|
| <code>str1.isdigit()</code> | semua karakter str1 adalah digit                                                             |
| <code>str1.isalpha()</code> | semua karakter str1 adalah huruf-huruf alfabet                                               |
| <code>str1.isalnum()</code> | semua karakter str1 adalah huruf alfabet atau digit                                          |
| <code>str1.islower()</code> | str1 memiliki setidaknya 1 karakter alfabet dan semua karakter alfabetnya adalah huruf kecil |
| <code>str1.isupper()</code> | str1 memiliki setidaknya 1 karakter alfabet dan semua karakter alfabetnya adalah huruf besar |
| <code>str1.isspace()</code> | str1 hanya berisi karakter spasi                                                             |

## 6.8. Ketentuan Penyederhanaan

List atau tuple kadang-kadang dapat digunakan untuk menyederhanakan kondisi gabungan yang mengandung operator logis. Misalnya, kondisi gabungan

```
(state == "MD") or (state == "VA") or (state ==
"WV") or (state == "DE")
```

dapat diganti dengan

state in ["MD", "VA", "WV", "DE"]

Terkadang kondisi gabungan yang melibatkan ketimpangan dapat dituliskan dalam bentuk yang lebih jelas. Misalnya kondisinya

$x \geq 10$ ) and  $(x \leq 20)$

dapat diganti dengan kondisi

$$10 < x \leq 20$$

dan kondisi

(x <= 10) or (x > 20)

dapat diganti dengan kondisi

not(10 < x <= 20)

Dua prinsip logika, yang dikenal sebagai Hukum De Morgan, adalah sebagai berikut:

```
not (cond1 and cond2) sama dengan not (cond1)
 or not (cond2)
not (cond1) or not (cond2) sama dengan not
 (cond1 and cond2)
```

Hukum De Morgan dapat diterapkan dari kiri ke kanan atau dari kanan ke kiri. Misalnya, menurut Hukum De Morgan, kondisi gabungan

```
not((temperature >= 80) and (humidity <= 60))
```

dan kondisi gabungan

```
not(len(word) == 5) and not(word.startswith('A'))
sama dengan
not((len(word) == 5) or (word.startswith('A')))
```

### Latihan Soal

Dalam latihan berikut tentukan output dan buatlah program yang sesuai berdasarkan perintah yang telah ditentukan

1. letter = 'D'  
print(letter + " is 4 positions before " + chr(ord(letter) + 4) +  
" alphabetically.")
2. print('S'+chr(116)+'atistika'+ ' Bisnis')
3. list1 = [50, 3, 12, 27, 10]  
list1.sort()  
print("Minimum:", list1[0])  
print("Maximum:", list1[-1])
4. Asumsikan nilai a adalah 4 dan nilai b adalah 2,99 dan tentukan apakah kondisi tersebut dievaluasi menjadi Benar atau Salah. Kemudian, gunakan fungsi cetak untuk mengonfirmasi jawaban Anda? Jelaskan alasannya
  - a)  $(a * a < b) \text{ or } \text{not}(a * a < a)$
  - b)  $3 * a == 2 * b$
  - c)  $(a * a < b) \text{ or } \text{not}(a * a < a)$
  - d)  $((a == b) \text{ or } \text{not}(b < a)) \text{ and } ((a < b) \text{ or } (b == a + 1))$
  - e)  $\text{not}((a < b) \text{ and } (a < (b + a)))$
5. Tentukan apakah kondisi berikut dievaluasi Benar atau Salah. Jelaskan alasannya
  - a) 'Sigma' > 'Gamma'
  - b)  $7 < 34 \text{ and } (\text{not } ("7" > "34" \text{ or } "7" == "34"))$
  - c) '1' < 'first'
6. Tentukan apakah kedua kondisi tersebut setara atau tidak — yaitu, apakah keduanya akan mengevaluasi ke True atau keduanya mengevaluasi ke False untuk nilai variabel apa pun yang muncul di dalamnya.
  - a)  $a <= b; (a < b) \text{ or } (a == b)$
  - b)  $\text{not}((a == b) \text{ and } (a == c)); (a != b) \text{ or } (a != c)$

- c)  $\text{not}((a == b) \text{ or } (a == c)); (a != b) \text{ and } (a != c)$   
 d)  $\text{not}(a >= b); (a <= b) \text{ and } \text{not}(a == b)$   
 e)  $(a < b) \text{ and } ((a > d) \text{ or } (a > e));$   
 $((a < b) \text{ and } (a > d)) \text{ or } ((a < b) \text{ and } (a > e))$
7. Tulis kondisi yang setara dengan negasi dari kondisi yang diberikan. (Misalnya,  $a != B$  setara dengan negasi  $a == B$ .)
- a)  $a > b$   
 b)  $(a < b) \text{ and } (c != d)$   
 c)  $a <= b$   
 d)  $(a == b) \text{ or } (a == d)$   
 e)  $\text{not}((a == b) \text{ or } (a > b))$   
 f)  $(a != "") \text{ and } (a < b) \text{ and } (\text{len}(a) < 5)$
8. Sederhanakan :
- a)  $(\text{ans} == 'Y') \text{ or } (\text{ans} == 'y') \text{ or } (\text{ans} == \text{"Yes"}) \text{ or } (\text{ans} == \text{"yes"}).$   
 b)  $(\text{name} == \text{"Athos"}) \text{ or } (\text{name} == \text{"Porthos"}) \text{ or } (\text{name} == \text{"Aramis"})$   
 c)  $(\text{year} == 2010) \text{ or } (\text{year} == 2011) \text{ or } (\text{year} == 2012) \text{ or } (\text{year} == 2013)$
9. Tentukan apakah Benar atau Salah ditampilkan.
- a) `num = "1234.56"`  
`print(isinstance(num, float))`  
 b) `str1 = "Departemen"`  
`print(str1.startswith('t') and str1.endswith('t'))`  
 c) `str1 = "Ekonomi Fiskal"`  
`print(str1.startswith(str1[0:4]))`
10. Tentukan apakah kondisi berikut dievaluasi Benar atau Salah. Jelaskan alasannya
- d) "Institut Teknologi sepuluh  
`nopember".endswith("o",2,6)`  
 e) "a b" in "Statistika Binsis"  
 f)  $\text{not}('C' == 'c') \text{ or } (\text{"coding"} < \text{"Coding"})$

## Referensi

Schneider, David I. 2016. *An Introduction to Programming Using Python Global Edition*. University of Maryland. British Library Cataloguing-in-Publication Data Pearson Education Limited

## BAB 7. DECISION STRUCTURE

### 7.1. Pengertian Kondisi If

Kondisi **if** pada python merupakan salah satu bentuk percabangan. Struktur **if** digunakan jika tidak terpenuhinya suatu kondisi, maka akan dieksekusi pada kondisi yang lain, dengan aturan penulisan sebagai berikut:

```
if condition:
 # Kode program yang dijalankan jika
 condition bernilai True
 # Kode program yang dijalankan jika
 condition bernilai True
```

Bagian **condition** adalah bagian yang berfungsi sebagai penentu dari struktur percabangan. Jika *condition* terpenuhi (menghasilkan nilai **True**), blok kode program akan dijalankan. Jika *condition* tidak terpenuhi (menghasilkan nilai **False**), blok kode program tidak akan dijalankan. **Condition** biasanya terdiri dari operasi perbandingan, misalnya apakah variabel a berisi angka 10, atau variabel password berisi string ‘rahasia’.

Blok kode program dalam bahasa Python ditandai dengan tanda titik dua setelah penulisan condition, kemudian diikuti satu atau beberapa baris dengan awalan *whitespace* di sisi kiri (boleh berupa spasi atau tab). Spasi di sisi kiri ini dikenal dengan istilah **indentation**.

#### Contoh 7.1 Program Percabangan If Bahasa Python

```
a = 12
b = 10
if a > b:
 print('Variabel a lebih besar dari variabel b')
 [Run] Variabel a lebih besar dari variabel
 b
```

Di awal kode program isi variabel a adalah angka 12 dan variabel b dengan angka 10. Kemudian di baris 4 terdapat kondisi if  $a > b$ , yakni apakah variabel a berisi angka yang lebih besar dari b?

Apakah 12 lebih besar dari 10? betul (True), maka perintah di baris 3 akan dijalankan. Dalam bahasa Python, identitas perlu ketika karakter spasi di awal baris 4, karena inilah penanda blok if. Jika baris ini tidak ‘dijorokkan’ satu atau beberapa spasi, maka akan terjadi error:

```
a = 12
b = 10
if a > b:
 print('Variabel a lebih besar dari variabel b') # baris ini akan error
 [Run] File "<ipython-input-3-b1a08b5a0a3e>",
line 5
 print('Variabel a lebih besar dari variabel b')
baris ini akan error
 ^
IndentationError: expected an indented block
```

Jika kita ingin menambah perintah lain di blok if yang sama, tulis baris baru dengan awalan spasi yang sama:

```
a = 12
b = 10

if a > b:
 print('Variabel a lebih besar dari variabel b')
 print('Sedang belajar bahasa Python')

[Run] Variabel a lebih besar dari variabel
 b
 Sedang belajar bahasa Python
```

Jika kondisi if ini tidak terpenuhi, maka blok kode program tidak akan di eksekusi. Berikut contohnya:

```
a = 8
b = 10

if a > b:
 print('Variabel a lebih besar dari variabel b')
 print('Sedang belajar bahasa Python ')
```

Kode program ini tidak akan menampilkan hasil apa-apa, karena variabel a saya isi dengan angka 8, sehingga kondisi **if a > b** menghasilkan nilai **False**. Namun akan berbeda jika ditulis seperti ini:

```
a = 8
b = 10

if a > b:
 print('Variabel a lebih besar dari variabel b')
print('Sedang belajar bahasa Python ')
 [Run] Sedang belajar bahasa Python
```

Kali ini perintah di baris 6 sudah tidak berada di dalam blok if (perhatikan perbedaan spasi di awal). Artinya, apapun hasil kondisi if, perintah di baris 6 akan selalu di jalankan. Bagaimana dengan membuat beberapa kondisi if? tidak ada masalah. Berikut contoh kode programnya:

### Contoh 7.2 Program Percabangan If Bahasa Python Beberapa Kondisi

```
a = 12
b = 12
if a > b:
 print('Variabel a lebih besar dari variabel b')
if a < b:
 print('Variabel a lebih kecil dari variabel b')
if a == b:
 print('Variabel a sama dengan variabel b')
 [Run] Variabel a sama dengan variabel b
```

Kode program ini merupakan hasil modifikasi dari kode sebelumnya. Di sini saya membuat 3 buah kondisi, yakni **if a > b**, **if a < b**, dan **if a == b**. Setiap kondisi if akan diperiksa, dan jika operasi perbandingan menghasilkan nilai true, maka blok kode program tersebut akan diproses. Silahkan anda coba ubah isi variabel a dan b untuk melihat blok kode program mana yang akan dijalankan.

### Contoh 7.3 Program Percabangan If Bahasa Python Beberapa Kondisi

Buat kode program yang bisa menebak apakah angka yang diinput merupakan bilangan genap atau bilangan ganjil:

```
a = 7

if (a % 2) == 0:
 print('Variabel a berisi angka genap')
if (a % 2) != 0:
 print('Variabel a berisi angka ganjil')
[Run] Variabel a berisi angka ganjil
```

Sekarang kondisi yang diperiksa adalah **if (a % 2) == 0** dan **(a % 2) != 0**. Di dalam bahasa Python, tanda persen (%) merupakan operator modulus yang dipakai untuk mencari sisa hasil bagi. Kondisi pertama, yakni **if (a % 2) == 0** akan bernilai **True** jika variabel a habis dibagi 2. Ini artinya variabel a berisi angka genap. Sedangkan kondisi kedua, yakni **if (a % 2) != 0** akan bernilai **True** jika variabel a tidak habis dibagi 2. Ini artinya variabel a berisi angka ganjil.

## 7.2. Pengertian Kondisi If Else

Modifikasi tambahan dari kondisi if yang sudah kita pelajari sebelumnya akan sedikit dimodifikasi untuk kondisi If Else. Blok kode program If tetap akan dijalankan ketika kondisi True, namun sekarang terdapat tambahan bagian Else akan dijalankan ketika kondisi False. Berikut format dasarnya dalam Python:

```
if condition:
 # Kode program yang dijalankan jika
 condition bernilai True
 # Kode program yang dijalankan jika
 condition bernilai True
else:
 # Kode program yang dijalankan jika
 condition bernilai False
 # Kode program yang dijalankan jika
 condition bernilai False
```

Bagian **condition** berperan sebagai penentu dari struktur percabangan ini. Jika *condition* terpenuhi (menghasilkan nilai **True**,

blok kode program milik **If** akan dijalankan. Jika *condition* tidak terpenuhi (menghasilkan nilai **False**), blok kode program bagian **Else**-lah yang akan diproses.

Sama seperti di pembahasan tentang kondisi if, dalam bahasa Python sebuah blok kode program ditandai dengan karakter spasi atau tab di awal baris. Mari kita lihat contoh prakteknya.

#### Contoh 7.4 Program Percabangan If Else

Pada tutorial sebelumnya kita telah melihat program pencari bilangan genap / ganjil menggunakan 2 buah kondisi **If** sebagai berikut:

```
a = 7
if (a % 2) == 0:
 print('Variabel a berisi angka genap')
if (a % 2) != 0:
 print('Variabel a berisi angka ganjil')

[Run] Variabel a berisi angka ganjil
```

Alur ini sebenarnya akan lebih sederhana (dan lebih efisien) jika kita ubah ke dalam struktur **If Else**. Jika sebuah angka tidak genap, maka pasti itu adalah angka ganjil. Sehingga jika kondisi **if (a % 2 == 0)** tidak terpenuhi (**False**), maka variabel a pasti berisi angka ganjil. Dengan demikian kode programnya bisa saya tulis ulang sebagai berikut:

```
a = 7

if (a % 2) == 0:
 print('Variabel a berisi angka genap')
else:
 print('Variabel a berisi angka ganjil')

[Run] Variabel a berisi angka ganjil
```

Sekarang jika kondisi **if (a % 2) == 0** menghasilkan **False**, bagian **Else** lah yang akan di proses. Kode program akan jadi lebih efisien karena pemeriksaan kondisi hanya perlu dilakukan 1 kali saja. Berikut contoh lain dari struktur kondisi If Else:

```

nilai = 76

if nilai >= 75:
 print('Selamat, anda lulus')
else:
 print('Maaf, silahkan coba lagi tahun depan')
[Run] Selamat, anda lulus

```

Di sini saya membuat kondisi **if nilai >= 75**, dimana jika variabel **nilai** berisi angka lebih besar atau sama dengan 75 maka jalankan perintah **print('Selamat, anda lulus')**. Jika tidak, blok **Else** lah yang akan di eksekusi, yakni **print('Maaf, silahkan coba lagi tahun depan')**.

### 7.3. Pengertian Kondisi If Else If /elif

Pada dasarnya, kondisi If Else If adalah sebuah struktur logika program yang di dapat dengan cara menyambung beberapa kondisi If Else menjadi sebuah kesatuan. (Elif adalah kependekan dari "else if."). Jika kondisi pertama bernilai False, maka kode program akan lanjut ke kondisi If di bawahnya. Jika ternyata tidak juga terpenuhi, akan lanjut lagi ke kondisi If di bawahnya, dst hingga blok Else terakhir atau terdapat kondisi If yang bernilai True. Berikut format dasar penulisan kondisi **If Else If** dalam bahasa Python:

```

if condition_1:
 # Kode program yang dijalankan jika
 condition_1 bernilai True
 # Kode program yang dijalankan jika
 condition_1 bernilai True
elif condition_2:
 # Kode program yang dijalankan jika
 condition_2 bernilai True
 # Kode program yang dijalankan jika
 condition_2 bernilai True
elif condition_3:
 # Kode program yang dijalankan jika
 condition_3 bernilai True
 # Kode program yang dijalankan jika
 condition_3 bernilai True
else:
 # Kode program yang dijalankan jika semua
 kondisi tidak terpenuhi
 # Kode program yang dijalankan jika semua
 kondisi tidak terpenuhi

```

Yang perlu menjadi perhatian, di dalam bahasa Python, perintahnya bukan **else if**, tapi **elif**. Ini cukup berbeda dengan kebanyakan bahasa pemrograman lain.

### Contoh 7.5 Program Percabangan If Else if

Membuat sebuah kode program untuk menampilkan teks dari sebuah angka nilai. Variabel asal berisi sebuah huruf antara ‘A’ – ‘E’. Kemudian kode program akan menampilkan hasil tampilan yang berbeda-beda untuk setiap huruf, termasuk jika huruf tersebut di luar ‘A’ – ‘E’.

```
nilai = 'D'

if nilai == 'A':
 print('Pertahankan')
elif nilai == 'B':
 print('Harus lebih baik lagi')
elif nilai == 'C':
 print('Perbanyak belajar')
elif nilai == 'D':
 print('Jangan keseringan main')
elif nilai == 'E':
 print('Kebanyakan bolos...')
else:
 print('Maaf, format nilai tidak sesuai')
 [Run] Jangan keseringan main
```

Di baris 1 mendefinisikan sebuah variabel **nilai** dengan huruf ‘D’.

Mulai dari baris 3 hingga 14, terdapat 5 buah pemeriksaan kondisi, yakni satu untuk setiap block **elif**. Dalam setiap kondisi, isi variabel **nilai** akan diuji apakah itu berupa karakter ‘A’, ‘B’, hingga ‘E’. Jika salah satu kondisi ini terpenuhi, maka block kode program yang sesuai akan dijalankan.

Jika ternyata nilai inputan bukan salah satu dari karakter ‘A’ – ‘E’, maka **block Else** di baris 14 lah yang akan dijalankan.

Setiap kondisi dari **block elif** ini bisa diisi dengan perbandingan yang lebih kompleks, seperti contoh berikut:

```
nilai = 65
print('Nilai:',nilai)

if nilai >= 90:
 print('Pertahankan')
elif (nilai >= 80) and (nilai < 90):
 print('Harus lebih baik lagi')
elif (nilai >= 60) and (nilai < 80):
 print('Perbanyak belajar')
elif (nilai >= 40) and (nilai < 60):
 print('Jangan keseringan main')
elif nilai < 40:
 print('Kebanyakan bolos...')
else:
 print('Maaf, format nilai tidak sesuai')

[Run] Nilai: 65
 Perbanyak belajar
```

Di sini memodifikasi sedikit kode program kita sebelumnya. Sekarang nilai inputan berupa angka antara 0 hingga 100. Angka inputan ini ditampung ke dalam variabel **nilai**.

Di baris 4, isi dari variabel **nilai** di periksa apakah berisi angka yang lebih dari 90. Jika iya, tampilkan teks “**Pertahankan!**”.

Jika kondisi di baris 4 tidak terpenuhi (yang artinya isi variabel **nilai** kurang dari 90), maka kode program akan lanjut ke kondisi **elif** berikutnya di baris 6. Di sini saya menggabungkan dua buah kondisi menggunakan operator logika **and**. Kondisi **elif (nilai >= 80) and (nilai < 90)** hanya akan terpenuhi jika isi variabel **nilai** berada dalam rentang 80 sampai 89.

Ketika membuat kondisi perbandingan, kita harus hati-hati dengan penggunaan tanda, apakah ingin menggunakan tanda lebih

besar saja ( $>$ ) atau tanda lebih besar sama dengan ( $\geq$ ) karena bisa mempengaruhi hasil akhir.

Jika ternyata kondisi ini tidak dipenuhi juga (artinya isi variabel **nilai** kurang dari 80), program akan lanjut ke kondisi **elif (nilai >= 60) and (nilai < 80)** di baris 8, yakni apakah **nilai** berada dalam rentang 60 – 79. Demikian seterusnya hingga kondisi terakhir **elif nilai < 40** di baris 12. Jika semua kondisi tidak terpenuhi, jalankan **block Else** di baris 14.

## 7.4. Masukkan Validasi dengan Pernyataan if-elif-else

Misalkan suatu program meminta pengguna untuk memasukkan nomor, dan kemudian menggunakan nomor tersebut dalam perhitungan. Jika pengguna tidak memasukkan nomor atau memasukkan nomor yang tidak pantas, program akan macet. Metode bernilai Boolean `isdigit` dapat digunakan untuk mencegah hal ini terjadi.

### Contoh 7.6 Input Validasi

Program berikut menggunakan metode `isdigit` untuk menjaga terhadap input yang tidak benar.

```
Request two numbers and find their sum. Validate the
input.
num1 = input("Enter first number: ")
num2 = input("Enter second number: ")
Display sum if entries are valid. Otherwise, inform
the user where invalid entries were made.
if num1.isdigit() and num2.isdigit():
 print("The sum is", str(eval(num1) + eval(num2)) +
 ".")
elif not num1.isdigit():
 if not num2.isdigit():
 print("Neither entry was a proper number.")
 else:
 print("The first entry was not a proper number.")
else:
 print("The second entry was not a proper number.")

[Run] Enter first number: 1
Enter second number: 2
The sum is 3.
```

## 7.5. True and False

Setiap objek memiliki nilai kebenaran yang terkait dengannya dan karenanya dapat digunakan sebagai syarat. Ketika angka digunakan sebagai kondisi, 0 dievaluasi menjadi False dan semua angka lainnya dievaluasi menjadi True. Tentu saja, objek Benar dan Salah masing-masing mengevaluasi untuk Benar dan Salah. Sebuah string, daftar, atau tuple yang digunakan sebagai suatu kondisi mengevaluasi ke False jika itu kosong, dan sebaliknya mengevaluasi ke True.

### Contoh 7.7 True or False

Program berikut menggambarkan nilai kebenaran objek.

```
Illustrate Boolean values.
if 7:
 print("A nonzero number is true.")
else:
 print("The number zero is false.")
if []:
 print("A nonempty list is true.")
else:
 print("An empty list is false.")
if ["spam"]:
 print("A nonempty list is true.")
else:
 print("The empty list is false.")

[Run] A nonzero number is true.
An empty list is false.
A nonempty list is true.
```

### Latihan Soal

Dalam latihan berikut tentukan output dan buatlah program yang sesuai berdasarkan perintah yang telah ditentukan

1.  $x = 1$   
if  $x > 0$ :

- ```
    print("Nilai %x adalah besar dari 0" % x )
2. nilai = 3
    if(nilai > 7):
        print("Selamat Anda Lulus")
    else:
        print("Maaf Anda Tidak Lulus")
3. number = 5
    if number < 0:
        print("negative")
    else:
        if number == 0:
            print("zero")
        else:
            print("positive")
4. a = 5
    if (a > 2) and ((a == 3) or (a < 7)):
        print("Halo")
5. Identifikasi kesalahan, nyatakan tipe setiap kesalahan (sintaks, runtime, atau logika), dan koreksi blok kode.
n = eval(input("Enter a number: "))
if 'n'%2 == 0:
    print("The number is an even number.")
else:
    print("The number is an odd number.")
6. Identifikasi kesalahan, nyatakan tipe setiap kesalahan (sintaks, runtime, atau logika), dan koreksi blok kode.
departemen = "Business Statistics"
if major == " Business Statistics" Or "Computer Science":
    print("Yes" )
7. Identifikasi kesalahan, nyatakan tipe setiap kesalahan (sintaks, runtime, atau logika), dan koreksi blok kode.
number = 6
if number > 5 and < 9:
    print("Yes")
else:
```

- ```
 print("No")
```
8. Sederhanakan
- ```
    if (a == 7):
```
- ```
 print("tujuh")
```
- ```
    elif (a != 7):
```
- ```
 print("sebelas")
```
9. Buatlah program untuk menentukan kategori umur seseorang dengan ketentuan sebagai berikut
- Dewasa = 25 tahun keatas
- Remaja = 17-25 tahun keatas
10. Buatlah program untuk menentukan suatu rumah tangga wajib melakukan pembayaran pajak rumah jika memiliki gaji diatas UMR. dengan ketentuan sebagai berikut
- gaji = Rp4.000.000 (standar gaji UMR = Rp..3.500.000)
- Status = Berkeluarga
- Status Kepemilikan Rumah = Milik Pribadi

## Referensi

- DuniaIlkom. (2020, 11 Juli). Percabangan Kondisi If Else If Bahasa Python. Diakses pada 11 Juli 2020, dari <https://www.duniaIlkom.com/tutorial-belajar-python-percabangan-kondisi-if-else-if-bahasa-python/>
- Schneider, David I. 2016. *An Introduction to Programming Using Python Global Edition*. University of Maryland. British Library Cataloguing-in-Publication Data Pearson Education Limited

## BAB 8. WHILE LOOP

### 8.1. Struktur Perulangan While Bahasa Python

Kode program dalam python yang bermaksud untuk melakukan perulangan atau mengulang beberapa baris perintah disebut dengan **loop** atau struktur perulangan. Didalam struktur perulangan terdapat tiga komponen yaitu komponen **awal perulangan**, komponen **saat perulangan** dan kondisi yang harus dipenuhi agar **perulangan berhenti**. Berikut format dasar struktur perulangan **while** dalam bahasa Python:

```
start;
while condition:
 # kode program yang akan diulang
 # kode program yang akan diulang
 Increment
```

Penjelasan dari format dasar struktur perulangan adalah sebagai berikut.

1. Bagian **start** adalah bagian awal yang berupa perintah inisialisasi variabel. Misalnya **i = 0**
2. **Condition** adalah bagian yang menjadi syarat atau ketentuan dalam struktur perulangan yang harus dipenuhi agar perulangan berjalan, misalnya **i < 5**.
3. Perintah **increment** di dalam block perulangan yang di pakai untuk menaikkan nilai variabel counter, misalnya dengan perintah **i = i +1**.

#### Contoh 8.1 Program Perulangan While

Berikut kode program perulangan While untuk menampilkan teks “Statistika Bisnis” sebanyak 5 kali:

```
i = 1
while i <= 5:
 print('Statistika Bisnis')
 i += 1
[Run] Statistika Bisnis
 Statistika Bisnis
 Statistika Bisnis
 Statistika Bisnis
 Statistika Bisnis
```

Di baris pertama terdapat perintah untuk menginput angka 1 ke dalam variabel **i**. Nantinya, variabel **i** ini akan menjadi *variabel counter* yang dipakai untuk menentukan jumlah perulangan.

Proses perulangan di mulai di baris 2. Perintah **while i <= 5** artinya, selama nilai variabel **i** kurang atau sama dengan 5, maka jalankan perulangan.

Di dalam blok perulangan terdapat perintah **print('Statistika Bisnis')** di baris 3. Ini dipakai untuk menampilkan teks “Statistika Bisnis“. Kemudian di baris 4 terdapat perintah *increment*, yakni **i += 1**. Perintah merupakan penulisan singkat dari **i = i + 1**, yang berfungsi untuk menaikkan nilai variabel **i** sebanyak 1 angka dalam setiap perulangan. Perulangan while di atas akan di ulang sebanyak 5 kali, mulai dari **i = 1**, **i = 2**, **i = 3**, **i = 4**, hingga **i = 5**. Ketika nilai *variabel counter* **i** sudah mencapai 6, maka kondisi **while i <= 5** tidak terpenuhi lagi (False), sehingga perulangan berhenti. Di dalam blok perulangan, kita juga bisa mengakses nilai dari variabel counter **i**:

```
i = 1
while i <= 5:
 print('Statistika Bisnis', i)
 i += 1
[Run] Statistika Bisnis 1
 Statistika Bisnis 2
 Statistika Bisnis 3
 Statistika Bisnis 4
 Statistika Bisnis 5
```

Bagaimana dengan perulangan menurun. Kita tinggal mengatur **kondisi awal**, **kondisi akhir**, serta proses **decrement**:

```
i = 1
while i <= 5:
 print('Statistika Bisnis', i)
 i -= 1
[Run] Statistika Bisnis 10
 Statistika Bisnis 9
 Statistika Bisnis 9
 Statistika Bisnis 7
 Statistika Bisnis 6
```

Di sini nilai awal variabel counter **i** sama dengan angka 10. Kondisi perulangan adalah **while i > 5**, artinya selama nilai variabel **i** di atas 5, jalankan perulangan. Dan karena kita ingin membuat perulangan menurun, maka dipakai perintah decrement **i = 1** atau sama dengan **i = i - 1** yang akan mengurangi nilai variabel **i** sebanyak 1 angka dalam setiap iterasi.

## 8.2. Infinite Loops

Seperti dijelaskan pada bagian sebelumnya, jika salah satu komponen dalam struktur perulangan adalah adanya perintah agar perulangan dapat berhenti. Perintah tersebut adalah perintah **increment**. Apabila perintah **increment** tidak dijalankan maka struktur perulangan tidak bisa berjalan dengan benar atau dengan kata lain, perulangan akan berjalan terus menerus. Kondisi demikian disebut **infinity loop**. Berikut contohnya:

### Contoh 8.2 Program Infinite Loops

```
i = 1
while i <= 5:
 print('Statistika Bisnis')
[Run] Statistika Bisnis
 Statistika Bisnis
 Statistika Bisnis
 Statistika Bisnis
 Statistika Bisnis
.....
```

Jika anda menjalankan kode program di atas, teks “**Statistika Bisnis**” akan ditampilkan terus menerus, tidak pernah selesai. Penyebabnya karena kondisi **while i <= 5** akan selalu bernilai **True**. Di dalam blok perulangan tidak ada perintah yang bisa mengubah nilai variabel **i** agar kondisi **while i <= 5** bernilai **False**.

Untuk menghentikan *infinity loop*, tekan kombinasi **CTRL + C** dari dalam jendela hasil. Atau bisa juga dengan tutup paksa aplikasi IDLE Python.

Di dalam Python, kita juga harus hati-hati dengan penggunaan spasi, karena itu adalah penanda blok perulangan. Kode program di bawah ini juga akan jadi infinity loop:

```
i = 1
while i <= 5:
 print('Statistika Bisnis') #infinity loop
i += 1
 [Run] Statistika Bisnis
 Statistika Bisnis
 Statistika Bisnis
 Statistika Bisnis
 Statistika Bisnis

```

Alasannya karena perintah increment di baris 4 bukan lagi berada di dalam blok perulangan, tapi berada setelah perulangan.

### 8.3. Statemen Break

Pada program pengulangan while dan for ada yang disebut sebagai statemen **Break** yang merupakan pernyataan yang akan membuat sebuah program berhenti atau keluar dari suatu blok pengulangan.

#### Contoh 8.3 Program Statemen Break

```
angka = 4
for i in range(0,10):
 print(i)
 if i is angka:
 print("angka ditemukan",i)
 break #untuk menghentikan proses
else:
 print("angka tidak ditemukan",angka)

[Run] 0
1
2
3
4
angka ditemukan 4
```

Pada perulangan for diatas jika kita tidak menggunakan fungsi break maka secara otomatis for akan melooping data sebanyak 10 kali sampai dengan batas yang ditentukan, namun kita juga bisa menghentikan proses loopingan tersebut, jika angka yang kita masukan telah ditemukan, maka proses looping akan berhenti. seperti pada script diatas kita akan menghentikan proses looping pada angka ke-4 jika loopingan telah sampai di angka ke-4 maka semua proses looping akan berhenti. Namun angka yang kita masukan tidak ditemukan maka **else** akan menampilkan pesan “**angka tidak ditemukan**”

#### 8.4. Statemen Continue

Selain statemen brake, dalam python juga ada statemen continue yang bertugas jika ada program di bawah statement tidak akan di eksekusi. Atau bisa kita katakan untuk melewati perintah yang ada di bawahnya. Statement continue menyebabkan program langsung melanjut ke step / interval berikutnya dan mengabaikan (skip) baris kode di bawahnya (yang satu blok).

#### Contoh 8.4 Program Statemen Continue

```
for i in range(1,5):
 if i is 2:
 print("nilai",i,"ditemukan")
 continue
 print("nilai setelah continue")
 # nilai tidak akan dicetak jika proses
terpenuhi
 print("nilai saat ini adalah",i)

[Run] nilai saat ini adalah 1
 nilai 2 ditemukan
 nilai saat ini adalah 3
 nilai saat ini adalah 4
```

pada script diatas bisa kita lihat bahwa nilai yang terdapat setelah continue tidak akan diproses, dan nilai tersebut akan selalu di ulang sampai proses berakhir.

## 8.5. Membuat Menu

Mengakses menu adalah salah satu tugas mendasar dari program interaktif. Pengguna membuat pilihan sampai dia memutuskan untuk berhenti.

### Contoh 8.5 Program Membuat Menu

Program berikut menggunakan menu untuk mendapatkan fakta tentang Indonesia.

```

Display facts about Indonesia.
print("Masukkan nomor menu")
print("Tentang Identitas Indonesia.\n")
print("1. Ibukota")
print("2. Burung Nasional")
print("3. Bunga Nasional")
print("4. Selesai\n")
while True:
 num = int(input("Silahkan pilih menu yang tersedia: "))
 if num == 1:
 print("Jakarta adalah ibukota dari Indonesia.")
 elif num == 2:
 print("Merak Bru adalah burung nasional Indonesia.")
 elif num == 3:
 print("Bunga Melati Putih adalah bunga nasional Indonesia.")
 elif num == 4:
 break
[Run] Masukkan nomor menu
Tentang Identitas Indonesia.

1. Ibukota
2. Burung nasional
3. Bunga nasional
4. Selesai

```

## Latihan Soal

Dalam latihan berikut tentukan output dan buatlah program yang sesuai berdasarkan perintah yang telah ditentukan

1. list1 = [7, 7, 10, 8]
 

```

total = 0
while list1:
 total += list1[0]
 list1 = list1[1:]
```

- print(total)
2. berhitung=5  
while berhitung  $\geq 2$ :  
    print(berhitung)  
    berhitung -=1
3. x = "Statistika Bisnis"  
while x:  
    print(x, ' ')  
    x = x[1:]
4. Buat kode yang lebih sederhana dan lebih jelas yang melakukan tugas yang sama dengan kode berikut.  
L = [2, 4, 6, 8]  
total = 0  
while L != []:  
    total += L[0]  
    L = L[1:]  
    print(total)
5. Identifikasi kesalahan dari program berikut.
- |                                               |                                                                                                        |
|-----------------------------------------------|--------------------------------------------------------------------------------------------------------|
| a) q = 1<br>while q!= 0:<br>q=q-2<br>print(q) | b) list1 = ['H', 'e', 'l', 'l',<br>'o']<br>i = len(list1)<br>while i > 1:<br>i -= 1<br>print(list1[i]) |
|-----------------------------------------------|--------------------------------------------------------------------------------------------------------|
6. Buatlah program untuk menampilkan perulangan dari list atau tuple.
7. Buatlah program untuk memunculkan 9 bilangan ganjil pertama
8. Buatlah kode program perulangan **While** dengan statemen **break**
9. Buatlah program untuk menentukan nilai variabel **angka** 0 sampai 9 dengan syarat angka 7 tidak ditampilkan.
10. Buatlah kode program perulangan **While** untuk membuat deret kelipatan 3 dari 3 sampai dengan 99

## Referensi

Pythonindo. (2020, 21 Juli). Perulangan. Diakses pada 21 Juli 2020, dari <https://www.pythonindo.com/perulangan/>

Schneider, David I. 2016. *An Introduction to Programming Using Python Global Edition*. University of Maryland. British Library Cataloguing-in-Publication Data Pearson Education Limited

## BAB 9. FOR LOOP

### 9.1. Pengertian Struktur Perulangan For Bahasa Python

Perulangan for pada python untuk memproses suatu array atau himpunan. Struktur perulangan **for** dalam python

```
data = [a, b, ...]
for i in data:
 # kode program yang akan diulang
 # kode program yang akan diulang
```

Dalam format ini, di awal saya mendefinisikan variabel **data** sebagai sebuah array atau himpunan (salah satu dari tipe data). Perulangan for akan dijalankan sebanyak jumlah element yang ada di dalam variabel **data**. Sepanjang perulangan, variabel **i** akan berisi element yang sedang di proses. Contoh kode program berikut akan memberikan pemahaman yang lebih.

#### Contoh 9.1 Program Perulangan For

Kode program dibawah ini bertujuan untuk menampilkan nama warna yang suda didefinisikan terlebih dahulu dalam sebuah variabel dengan menggunakan perulangan **for**:

```
warna = ['Merah', 'Biru', 'Kuning', 'Biru']
for i in warna:
 print(i)

[Run] Merah
 Biru
 Kuning
 Biru
```

Di baris 1 program terdapat suatu variabel dengan data tipe list dengan nama variabel warna. Variabel warna terdapat empat element atau nama warna.

### Contoh 9.2 Program Perulangan For Tipe Dataset

Perulangan for juga bisa dipakai untuk tipe data lain, misalnya tipe data set:

```
warna = {'Merah', 'Biru', 'Kuning', 'Biru'}
for i in warna:
 print(i)
```

```
[Run] Biru
Kuning
Merah
```

Perhatikan perubahan tanda kurung di baris 1. Sekarang saya menggunakan kurung kurawal yang merupakan cara pembuatan tipe data set di dalam Python. Hasilnya, nama warna hanya tampil 3 buah. ini karena perilaku dari tipe data set Python, dimana jika terdapat data yang berulang, data tersebut tidak akan disimpan. Dalam hal ini, warna ‘biru’ tertulis sebanyak 2 kali.

### Contoh 3 Program Perulangan For Tipe Data String

```
web = 'Statistika Bisnis'
for huruf in web:
 print(huruf)
```

[Run] S

```
t
a
t
i
s
t
i
k
a
```

```
B
i
s
n
i
s
```

Ini karena di dalam bahasa Python, tipe data string merupakan sebuah array.

### Contoh 9.4 Program Perulangan For Tipe Data Tuple

```
nama = ['budi', 'andi', 'rudi', 'sandi']
usia = [20, 18, 22, 19]
for i in range(len(nama)) :
 print(nama[i], ' berusia ', usia[i], ' tahun')
```

[Run] budi berusia 20 tahun  
andi berusia 18 tahun  
rudi berusia 22 tahun  
sandi berusia 19 tahun

## 9.2. Penggunaan Function range()

Fungsi range() merupakan fungsi yang menghasilkan list. Fungsi ini akan menciptakan sebuah list baru dengan rentang nilai tertentu. Jika m dan n memiliki nilai integer dan  $m < n$ , maka fungsinya

## range (m, n)

menghasilkan urutan bilangan bulat m, m + 1, m + 2, c, n-1. Yaitu, urutan dimulai dengan m, dan 1 berulang kali ditambahkan ke m sampai angka tepat sebelum n tercapai. Beberapa contoh adalah sebagai berikut:

**range (10)** akan menghasilkan [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

**range(4, 12)** akan menghasilkan [4, 5, 6, 7, 8, 9, 10, 11]

**range(10,50,5)** akan menghasilkan: [10, 15, 20, 25, 30, 35, 40, 45]

list dari 10 sampai 50 dengan interval 5

Struktur perulangan di bahasa Python sepintas tidak memungkinkan kita untuk membuat perulangan angka naik, misalnya dari 1, 2, 3, dst. Namun ini bisa dibuat dengan bantuan fungsi atau function **range()**. Fungsi **range()** bisa dipakai untuk membuat deret angka, yang kemudian menjadi inputan ke dalam perulangan for. Berikut contoh penggunaannya:

### Contoh 9.5 Program Penggunaan Function range()

```
for i in range(7):
 print(i)
```

```
[Run] 0
1
2
3
4
5
6
```

Di sini perintah **range(5)** akan membuat 5 buah deret yang dimulai dari angka 0, 1, 2, 3 dan 4. Yang harus diperhatikan, nilai maksimal dari **range(5)** adalah 4, karena angka dimulai dari 0, bukan 1. Namun kita bisa mengatur jangkauan yang diinginkan. Caranya, tambah angka kedua ke dalam function **range()**:

### 9.3. Pass Statement

Selain statemen break dan continue yang telah dijelaskan pada bab sebelumnya, didalam python juga terdapat statemen **pass** yang mengakibatkan program tidak melakukan tindakan apa-apa. Perintah pass biasanya digunakan untuk mengabaikan suatu blok statemen baik itu berupa perulangan, pengkondision, class, dan fungsi yang belum didefinisikan badan programnya agar tidak terjadi error ketika proses compilasi.

#### Contoh 9.6 Program Pass Statement

```
for i in range(1,5):
 if i is 2:
 print("nilai",i,"ditemukan")
 pass
 print("nilai setelah pass")
 print("nilai saat ini adalah",i)

[Run] nilai saat ini adalah 1
 nilai 2 ditemukan
 nilai setelah pass
 nilai saat ini adalah 2
 nilai saat ini adalah 3
 nilai saat ini adalah 4
```

### 9.4. Nested loop

Penggunaan loop dalam loop dapat dilakukan dalam python, hal ini disebut sebagai nested loop atau loop bersarang. Tentu hal ini akan berimbang pada penggunaan memori. Contoh penggunaan loop bersarang

#### Contoh 9.7 Program Nested loop

```

for m in range(1, 6):
 for n in range(1, 6):
 print(m, 'x', n, '=', m * n, "\t", end="")
 print()

```

[Run]

|            |            |            |            |
|------------|------------|------------|------------|
| 1 x 1 = 1  | 1 x 2 = 2  | 1 x 3 = 3  | 1 x 4 = 4  |
| 1 x 5 = 5  |            |            |            |
| 2 x 1 = 2  | 2 x 2 = 4  | 2 x 3 = 6  | 2 x 4 = 8  |
| 2 x 5 = 10 |            |            |            |
| 3 x 1 = 3  | 3 x 2 = 6  | 3 x 3 = 9  | 3 x 4 = 12 |
| 3 x 5 = 15 |            |            |            |
| 4 x 1 = 4  | 4 x 2 = 8  | 4 x 3 = 12 | 4 x 4 = 16 |
| 4 x 5 = 20 |            |            |            |
| 5 x 1 = 5  | 5 x 2 = 10 | 5 x 3 = 15 | 5 x 4 = 20 |
| 5 x 5 = 25 |            |            |            |

### Latihan Soal

Dalam latihan berikut tentukan output dan buatlah program yang sesuai berdasarkan perintah yang telah ditentukan

1. Tentukan urutan yang dihasilkan oleh fungsi range berikut.
  - a) range(10, 1, -1)
  - b) range(1,10)
  - c) range(1, 10, 3)
2. Tentukan fungsi range yang menghasilkan urutan berikut
  - a) [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
  - b) [1,3,5,7,9]
  - c) [1,11,21,31,41,51]
3. tentukan output yang dihasilkan
  - a) boroughs = ("Manhattan", "Bronx", "Brooklyn", "Queens", "Staten Island")
   
minLetters = 100
   
for borough in boroughs:
   
 if len(borough) < minLetters:
   
 minLetters = len(borough)
   
 print("The shortest word has length", minLetters)
4. Buat ulang program menggunakan perulangan for
  - a) num = 7
  - b) print("Statistika Bisnis")  
while num <= 49:  
 print(num)

```
num += 2
print("Statistika
Bisnis")
print("Statistika
Bisnis")
```



6. Buatlah program untuk menampilkan perulangan dari list atau tuple.
  7. Program untuk menemukan jumlah bilangan dalam satu list [7, 5, 9, 8, 4, 2, 6, 4, 1]
  8. Buatlah kode program perulangan **for** untuk mencetak angka -1 s/d 8
  9. Buatlah kode program Python Mencari bilangan ganjil dengan for
  10. Buatlah program implementasi Nested Loop untuk mencetak bilangan prima dari 2 sampai 30.

## **Referensi**

Schneider, David I. 2016. *An Introduction to Programming Using Python Global Edition*. University of Maryland. British Library Cataloguing-in-Publication Data Pearson Education Limited

# BAB 10. FUNGSI

## 10.1. Fungsi (Function)

Fungsi digunakan untuk memecah masalah kompleks menjadi masalah kecil yang harus diselesaikan satu per satu. Fungsi memungkinkan kita untuk menulis dan membaca suatu program sedemikian rupa sehingga kita pertama-tama fokus pada tugas dan kemudian pada bagaimana menyelesaikan setiap tugas.. Ada dua jenis fungsi - fungsi yang dirancang untuk mengembalikan nilai dan fungsi yang mengeksekusi baris kode tanpa bermaksud mengembalikan nilai. Tipe fungsi kedua sering menampilkan output dengan pernyataan cetak atau membuat file. Kami akan mulai dengan membahas fungsi yang dirancang untuk mengembalikan nilai. Dalam python terdapat dua fungsi secara umum.

- **Fungsi Bawaan**, adalah fungsi yang sudah tersedia didalam python dalam format default ataupun format library .
- **Fungsi Buatan Pengguna**, adalah fungsi yang tidak disediakan oleh python, tetapi dibuat sendiri oleh user.

Seperti pada bahasa pemrograman pada umumnya, fungsi di python juga memiliki struktur sebagai berikut.

1. Statemen `def` menjadi statemen awalan pada python yang kemudian diikuti dengan nama fungsi.
2. Pada python, argument parameter dapat tersedia ataupun tidak pada fungsi.
3. Tanda titik dua digunakan untuk menandakan awal pendefinisian isi dari fungsi yang terdiri dari statemen
4. `return` menjadi penanda bagian akhir dari suatu fungsi, kemudian akan memberikan suatu output balik kepada program yang memanggil tersebut. Statemen `return` pada fungsi sifatnya opsional.

## 10.2. Fungsi Bawaan

Python memiliki banyak fungsi bawaan. Dalam satu hal, fungsinya seperti program miniatur. Mereka menerima input, mereka

memproses input, dan mereka menghasilkan output. Beberapa fungsi bawaan yang kami temui sebelumnya tercantum pada Tabel 10.1

**Tabel 10.1.** Contoh Fungsi Bawan

| Fungsi | Contoh                    | Input         | Output |
|--------|---------------------------|---------------|--------|
| Int    | int(7.6) adalah 7         | Angka/number  | Number |
| Chr    | chr(65) adalah 'A'        | Angka/number  | String |
| Ord    | ord('A') adalah 65        | String        | Number |
| Round  | round(2.34, 1) adalah 2.3 | Number,number | number |

Output untuk masing-masing dari empat fungsi dalam tabel di atas adalah nilai tunggal. Suatu fungsi dikatakan mengembalikan outputnya. Sebagai contoh, dalam contoh pertama dari Tabel 10.1, kita mengatakan bahwa fungsi int mengembalikan nilai 2. Item di dalam tanda kurung disebut argumen. Tiga fungsi pertama pada Tabel 10.1 memiliki satu argumen dan fungsi keempat memiliki dua argumen. Argumen dapat berupa literal (seperti pada Tabel 10.1), variabel, atau jenis ekspresi lainnya. Baris kode berikut menggambarkan penggunaan literal, variabel, dan ekspresi sebagai argumen untuk fungsi int. Baris ketiga kode dikatakan memanggil (atau, memanggil) fungsi int dan meneruskan nilai num1 ke fungsi.

```
num = int(3.7) # literal as an argument
num1 = 2.6
num2 = int(num1) # variable as an argument
num1 = 1.3
num2 = int(2 * num1) # expression as an argument
```

### 10.3. Fungsi Buatan Pengguna

Selain menggunakan fungsi bawaan, kita dapat mendefinisikan fungsi kita sendiri (disebut fungsi yang ditentukan pengguna) yang mengembalikan nilai. Fungsi seperti itu biasanya didefinisikan oleh pernyataan kode sebagai berikut

```
def function_name(parameters):
 """function_opsional"""
 statement(s)
 return [expression]
```

Penjelasannya dari sintaks fungsi di atas:

1. Statemen `def` menjadi kata awalan disuatu, kemudian diikuti oleh nama fungsinya (`function_name`). Karakter tanda kurung dan tanda titik dua (:) menandai header (kepala) fungsi.
2. Parameter / argumen adalah input dari luar yang akan diproses di dalam tubuh fungsi.
3. `"function_opsional"` sebuah string untuk penjelasan fungsi, `"function_opsional"` bisa tidak digunakan. Letak dari `"function_opsional"` paling atas setelah baris `def`.
4. Baris selanjutnya berisikan pernyataan statemen dan diberikan indentasi
5. Baris terakhir adalah statemen `return`, yang berguna untuk mengembalikan suatu nilai `expression` dari fungsi. Statement `return` bersifat opsional

#### 10.4. Fungsi Memiliki Satu Parameter

Berikut merupakan contoh fungsi yang memiliki satu parameter. Contoh tentang konversi suhu. Program berikut menggunakan fungsi ini `fahrenheitToCelsius`. Di sebelah baris terakhir program, `celsiusTemp = fahrenheitToCelsius(fahrenheitTemp)`, memanggil (artinya, menjalankan) fungsinya. Nilai argumen `fahrenheitTemp` diberikan ke parameter `t` pada header fungsi. (Kami mengatakan bahwa nilai `fahrenheitTemp` diteruskan ke parameter `t`.) Setelah fungsi melakukan perhitungan menggunakan nilai parameter `t`, pernyataan pengembalian menentukan bahwa nilai yang dihitung adalah output dari fungsi `fahrenheitToCelsius`. Nilai itu diberikan ke variabel `celsiusTemp`. Program kemudian menggunakan nilai variabel saat menampilkan output

#### Contoh 10.1 Fungsi Memiliki Satu Parameter

```

def fahrenheitToCelsius(t):
 ## Convert Fahrenheit temperature to Celsius.
 convertedTemperature = (5 / 9) * (t - 32)
 return convertedTemperature
fahrenheitTemp = eval(input("Enter a temperature in
degrees Fahrenheit: "))
celsiusTemp = fahrenheitToCelsius(fahrenheitTemp)
print("Celsius equivalent:", celsiusTemp,
"degrees")

[Run] Enter a temperature in degrees
Fahrenheit: 50
 Celsius equivalent: 10.0 degrees

```

## 10.5. Melewati Nilai ke Fungsi

Jika argumen dalam panggilan fungsi adalah variabel, objek yang ditunjuk oleh variabel argumen (bukan variabel argumen itu sendiri) diteruskan ke variabel parameter. Oleh karena itu, jika objek tidak dapat diubah dan fungsinya mengubah nilai variabel parameter, tidak akan terjadi perubahan pada objek yang ditunjukkan oleh variabel argumen. Bahkan jika dua variabel memiliki nama yang sama, mereka diperlakukan sebagai variabel yang sama sekali berbeda. Oleh karena itu, ketika variabel argumen menunjuk ke objek numerik, string, atau tuple, tidak ada kemungkinan bahwa nilai variabel argumen akan diubah oleh pemanggilan fungsi.

### Contoh 10.2 Melewati Nilai ke Fungsi

Program berikut menunjukkan bahwa meskipun nilai parameter **num** dalam definisi fungsi diubah, tidak ada perubahan dalam nilai argumen num di bagian program yang disebut fungsi.

```
def triple(num):
 num = 3 * num
 return num
num = 2
print(triple(num))
print(num)
 [Run] 6
 2
```

## 10.6. Fungsi Memiliki Beberapa Parameter

Fungsi berikut memiliki lebih dari satu parameter. Berikut adalah contoh fungsi dengan beberapa parameter. Setiap parameter dipisahkan dengan tanda koma dan nilai parameter diisi oleh variabel multiple.

### Contoh 6.3 Fungsi Memiliki Beberapa Parameter

```
#fungsi hitung
def hitung(num1,num2):
 return num1 * num2
#memanggil fungsi dan mengisi parameter
multiple=hitung(1,7)
print("Hasil Operasi Perkalian :", multiple)
 [Run] Hasil Operasi Perkalian : 7
```

## 10.7. Fungsi Bernilai Boolean dan List

Sejauh ini, nilai yang dikembalikan oleh fungsi adalah angka atau string. Namun, suatu fungsi dapat mengembalikan semua jenis nilai. Dua program berikut ini menggunakan fungsi yang mengembalikan nilai Boolean dan fungsi yang mengembalikan list.

### Contoh 10.4 Fungsi Bernilai Boolean dan List

Kata vokal adalah kata yang berisi setiap vokal. Beberapa contoh kata vokal adalah sequoia, facetious, dan dialog. Program berikut menggunakan fungsi bernilai Boolean untuk menentukan apakah input kata oleh pengguna adalah kata vokal. Fungsi katavokal memeriksa kata untuk vokal satu per satu dan berakhir ketika sebuah vokal ditemukan hilang atau setelah semua vokal dianggap.

```

def katavokal(kata):
 kata = kata.upper()
 vokal = ('a', 'e', 'i', 'o', 'u')
 for vowel in vokal :
 if vowel not in kata:
 return False
 return True
kata = input("Masukkan kata: ")

if katavokal(kata):
 print(kata, "berisi seluruh kata vokal.")
else:
 print(kata, "tidak berisi seluruh kata vokal.")
[Run] Masukkan kata: Statistika Bisnis
Statistika Bisnis tidak berisi seluruh kata
vokal.

```

## 10.8. Fungsi yang Tidak Mengembalikan Nilai

Fungsi yang tidak mengembalikan nilai terlihat seperti fungsi yang dibahas di atas dengan pengecualian bahwa mereka tidak mengandung pernyataan pengembalian. Pada fungsi demikian kemungkinan ada tau mungkin tidak memiliki parameter dan dipanggil dengan menempatkan nama mereka (bersama dengan argumen mereka) sebagai pernyataan pada satu baris. Berikut adalah contoh sebuah fungsi dengan parameter **identitas**, kemudian outputnya **hello identitas**, sesuai **identitas** yang diisi ke dalamnya.

### Contoh 10.5 Fungsi Tanpa Return Values

```

def say_hello(identitas):
 print('hello', identitas)

say_hello('Akbar')

[Run] hello Akbar.

```

Untuk diingat, **nama** pada deklarasi fungsi (baris 1) disebut **parameter**, sedangkan nilai yang digunakan sebagai isi dari parameter nama pada saat fungsi dipanggil (baris 4), yakni ‘Akbar’ disebut sebagai **argumen**.

## 10.9. Fungsi Tanpa Parameter

Berikut adalah contoh fungsi tanpa parameter dan nilai return yang akan menampilkan kalimat “Halo ITS”.

### Contoh 10.6 Fungsi Tanpa Parameter

```
#deklarasi fungsi
def kalimat():
 print("Halo ITS")
#Pemanggil Fungsi
kalimat()
```

[Run] Halo ITS

## 10.10. Ruang Lingkup (Scope) Variabel

Variabel memiliki *scope lokal* adalah variabel yang yang didefinisikan di dalam fungsi yang hanya dapat diakses didalam fungsi yang telah didefinisikan, sedangkan variabel yang didefinisikan di luar fungsi memiliki *scope global* yang bisa diakses dari seluruh tempat dimanapun di dalam program. Berikut adalah contohnya:

### Contoh 10.7 Scope Variabel

```
total = 0
Variabel global
Definisi fungsi
def sum(arg1, arg2):
 """Menambahkan variabel dan mengembalikan hasilnya."""
 total = arg1 + arg2;
 # total di sini adalah variabel lokal
 print ("Di dalam fungsi nilai total : ", total)
 return total

Pemanggilan fungsi sum
sum(10, 20)
print ("Di luar fungsi, nilai total : ", total)

[Run] Di dalam fungsi nilai total : 30
 Di luar fungsi, nilai total : 0
```

Perhatikan bagaimana variabel total di dalam dan di luar fungsi adalah dua variabel yang berbeda.

### 10.11. Penamaan Konstanta

Suatu program terkadang menggunakan konstanta khusus yang akan digunakan beberapa kali dalam program. Konstanta semacam itu mungkin merujuk pada suku bunga atau usia minimum. Salah satu konvensi yang digunakan oleh pemrogram adalah membuat variabel global yang namanya ditulis dalam huruf besar dengan kata-kata yang dipisahkan oleh karakter garis bawah, dan menetapkan konstanta untuknya. Beberapa contoh adalah sebagai berikut:

```
INTEREST_RATE = 0.04
MINIMUM_VOTING AGE = 18
BOOK_TITLE = "Programming with Python"
```

Konvensi penamaan khusus mengingatkan programmer bahwa tidak ada penugasan kembali ke variabel yang harus dilakukan selama pelaksanaan program. Karena Python memungkinkan penugasan kembali ke variabel apa pun, programmer bertanggung jawab untuk tidak mengubah nilai variabel. Konstanta seperti itu disebut konstanta bernama. Beberapa contoh pernyataan menggunakan konstanta

```
interestEarned = INTEREST_RATE * amountDeposited
if (age >= MINIMUM_VOTING AGE):
 print("You are eligible to vote.")
 print("The title of the book is", BOOK_TITLE + ".")
```

Meskipun nilai konstanta seperti INTEREST\_RATE tidak akan berubah selama eksekusi suatu program, nilainya mungkin perlu diubah di lain waktu. Pemrogram dapat menyesuaikan diri dengan perubahan ini dengan mengubah hanya satu baris kode di bagian atas program alih-alih mencari seluruh program untuk setiap kemunculan suku bunga lama.

### Latihan Soal

Dalam latihan berikut tentukan output dan buatlah program yang sesuai berdasarkan perintah yang telah ditentukan

1. Tentukan output yang dihasilkan

```
def main():
 print(uc('h'))
 print(uc('w'))
def uc(letter):
 if letter == 'h':
 return 'H'
 else:
 return letter
```

3. Tentukan output yang dihasilkan

```
PLANE_RIDE_COST = 200
def main():
 noOfDays = 3
 cost = PLANE_RIDE_COST + noOfDays * 20
 print("Total cost: ${0:.2f}".format(cost))
 main()
```

4. Tentukan output yang dihasilkan

```
nama = "Ak"
def main():
 global nama
 Namalain = getName()
 nama += Namalain
 print(nama)
def getName():
 nama = "bar"
 return nama
```

2. Tentukan output yang dihasilkan

```
x = 7
def main():
 x = 5
 f()
 print(x)
def f():
 print(x)
main()
```

5. Tentukan output yang dihasilkan

```
def main():
 num = 70
 = triple(num)
 print(num)
def triple(num):
 num = 10
 * num
```

- main() main()
6. Buatlah contoh 2 fungsi tanpa return values dengan satu dan lebih parameter
  7. Buatlah sebuah fungsi untuk menampilkan variabel nama dan NRP.
  8. Buatlah fungsi untuk menghitung luas segitiga. Beri penamaan fungsi tersebut dengan nama fungsi **luas\_segitiga**
  9. Buatlah fungsi untuk menghitung total harga yang harus dikeluarkan untuk biaya kos dengan ketentuan sebagai berikut. Beri nama fungsi **Bayar\_kos**  
Biaya Kos = Rp. 850.000  
Keterlambatan = 5 hari  
Denda Terlambat = Rp. 50.000/hari  
bayar
  10. Buatlah fungsi untuk mengonversi **cos** menjadi **sin**. Beri penamaan fungsi tersebut dengan nama fungsi **trigonometri**

## Referensi

- David I. Schneider. 2016. An Introduction to Programming Using Python Global Edition. University of Maryland. British Library Cataloguing-in-Publication Data Pearson Education Limited
- Pythonindo. (2020, 18 Juli). Fugsi. Diakses pada 18 Juli 2020, dari <https://www.pythonindo.com/fungsi/>
- RH. Sianipar dan Hamzan Wadi, 2015. “Pemrograman Python: Teori dan Implementasi”, Informatika, Bandung.
- Sandy H.S. Herho Program, 2017. Tutorial Pemrograman Python 2 Untuk Pemula. Studi Meteorologi, Fakultas Ilmu dan Teknologi Kebumian, Institut Teknologi Bandung. WCPL Press, Bandung.